Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Theoretical Computer Science 298 (2003) 179–206

Theoretical
Computer Science

www.elsevier.com/locate/tcs

Polynomial-time identification of very simple grammars from positive data☆

Takashi Yokomori

Department of Mathematics, School of Education, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan

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. This class of languages is also known as the class of left Szilard languages of context-free grammars.

After providing some properties of very simple languages, we show that the class of very simple grammars is polynomial-time identifiable in the limit from positive data in the following sense. That is, we show that there effectively exists an algorithm that, given a target very simple grammar G_* over alphabet Σ, 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 $O(m)$, and the total number of prediction errors made by the algorithm is bounded by $O(n)$, where n is the size of G_*, $m = \mathrm{Max}\{N^{|\Sigma|+1}, |\Sigma|^3\}$ and N is the total length of all positive data provided.

Keywords: Very simple grammars; Identification in the limit from positive data; Polynomial-time learning

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" [3], a computational approach to learning theory has been again receiving much attention, and a numerous number of intensive work on grammatical inference

☆ A preliminary version appeared in the Proceedings of COLT'91, Santa Cruz, CA, 1991.

E-mail address: yokomori@mn.waseda.ac.jp (T. Yokomori).

0304-3975/03/$ - see front matter
PII: S0304-3975(02)00423-1

has been reported. From the practical point of view, there are, we believe, two major requirements for an inductive inference algorithm: the identification algorithm must have a good *time efficiency* and hopefully run with only *positive data* (*examples*).

Angluin [1] has given several conditions for a class of languages to be identifiable in the limit from positive data, and presented some examples of identifiable classes. She has also proposed subclasses of regular languages called k-reversible languages (where $k \geqslant 0$) and shown these classes are identifiable in the limit from positive data, requiring a polynomial time for updating conjectures [2]. Wright [22] proposes a sufficient condition for a language class to be identifiable in the limit from positive data, while Sato and Umayahara [18] introduce another sufficient condition for the identifiability in the limit from positive data and discuss the relationship to Wright's condition.

On the other hand, as for efficient identifiability results on subclasses of regular languages, Tanida and Yokomori [20,23] have introduced a new subclass of regular languages called strictly deterministic regular languages and shown its identifiability from positive data. Further, in [6,24], another subclass of regular languages called strictly k-testable languages is studied and shown to be identifiable in the limit from positive data, together with its application results to pattern recognition and DNA sequence analysis, respectively.

Motivated by a question posed by Angluin, however, one natural question has been recognized to be significant: In what sense we should analyse the time complexity of an "in-the-limit" algorithm? 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. By making a slight modification of his definition in [17], we propose the following definition for polynomial-time identifiability in the limit from positive data.

Informally, we say a class of grammars $\mathscr{R}$ is *polynomial-time identifiable in the limit from positive data* if and only if there is an algorithm A which, given r in $\mathscr{R}$, identifies r' in $\mathscr{R}$ equivalent to r in the limit from positive data, with the property that there exist polynomials p and q such that for any n, for any r 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.

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 the author's knowledge, the notion of a very simple grammar was originally introduced in [5] in the study of some types of Thue systems and the equivalence problem for simple deterministic grammars by Korenjak and Hopcroft [10]. Informally, a context-free grammar G is *very simple* if the righthand side of each rule starts with distinct terminal symbol, possibly followed by a nonterminal string. For example, the context-free grammar $G_1 = (\{E, R\}, \{+, -, *, (,), a\}, P_1, E)$, where $P_1 = \{E \to +EE, E \to -EE, E \to *EE, E \to (ER, E \to a, R \to)\}$ generates a subset of the arithmetic expressions in prefix notation. Further, consider the context-free grammar $G_2 = (\{S, T, G\}, \{a, c, d, \bar{a}, \bar{c}\}, P_2, S)$, where $P_2 = \{S \to aST, S \to cSG, S \to d, T \to \bar{a}, G \to \bar{c}\}$ generating $\{xd\bar{x}^R | x \in \{a, c\}^*\}$. Under the interpretation that $\bar{a} = t$, $\bar{c} = g$, this language represents biological "palindromes" to form secondary structures in nucleic acids. These grammars G_1, G_2 are typically very simple, and, thus, very simple grammars can capture the non-regularity feature

necessary for describing a certain class of popular languages. (Also, see Example 1 below.) The class of very simple languages forms a proper subclass of the simple deterministic languages and is incomparable to the class of regular languages. On the other hand, Mäkinen has discussed the left Szilard languages of context-free grammars in great detail, investigating the derivational properties of context-free grammars [12], and it turns out that the class of left Szilard languages of context-free grammars coincides with the class of very simple languages.

After providing some properties of very simple languages, we show that the class of very simple grammars is polynomial-time identifiable in the limit in the sense above. In fact, the identification of the class is achieved using only positive data, and this result is in marked contrast with the fact that the class of regular languages is not at all identifiable in the limit from positive data only [7].

Thus, the main result in this paper provides an interesting instance of a language class containing nonregular languages which is polynomial-time identifiable in the limit. As a corollary, it immediately follows that the class of very simple grammars is polynomial-time identifiable via only equivalence queries, provided that only positive counterexamples are supplied in the identification process.

2. Definitions

2.1. Basic definitions and notations

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., [9].)

Let Σ be a finite alphabet and Σ^* be the set of all finite length strings over Σ. Further, let $\Sigma^+ = \Sigma^* - \{\lambda\}$, where λ is the null string. By $len(u)$ we denote the length of a string u. A *language over* Σ is a subset of Σ^*. For a string w in Σ^*, $\mathrm{alph}(w)$ denotes the set of terminal symbols appearing in w. For a language L, let $\mathrm{alph}(L) = \bigcup_{w\in L} \mathrm{alph}(w)$.

Let h be a homomorphism from Σ^* to Δ^*. Then, h is called a *coding* if and only if for all a in Σ, $h(a)$ is in Δ. Further, a coding h is called a *renaming* if and only if it is a bijection.

Let $G = (V_N, \Sigma, P, S)$ be a context-free grammar (CFG). The language generated by G is defined by $\{w \in \Sigma^* | S \Rightarrow^* w \text{ in } G\}$.

By $\mathrm{Label}(P)$ we denote the set of labels of all rules in P. Then, define $D_L(G)$ as $\{\pi_1 \cdots \pi_n \in \mathrm{Label}(P)^* | S \Rightarrow_L^{\pi_1} a_1\alpha_1 \Rightarrow_L \cdots \Rightarrow_L^{\pi_n} a_1 \cdots a_n \in \Sigma^*\}$, where each π_i is a label of a rule used in the ith step, and $\Rightarrow_L$ indicates the left-most derivation. The language $D_L(G)$ is called *left Szilard language* of G.

Let $G_i = (V_{Ni}, \Sigma, P_i, S_i)$ $(i = 1, 2)$ be *CFGs*. Then, G_1 and G_2 are *isomorphic* if and only if there is a renaming h from $(V_{N1} \cup \Sigma)^*$ to $(V_{N2} \cup \Sigma)^*$ such that $h(S_1) = S_2$, the restriction of h to Σ is the identity, and $P_2 = \{h(X) \to h(\alpha) | X \to \alpha \in P_1\}$. A *CFG* $G = (V_N, \Sigma, P, S)$ is *linear* if and only if any rule of G is of the form either $A \to uBv$ or $A \to v$, where $A, B \in V_N$, $u, v \in \Sigma^*$.

For a language L over Σ and x in Σ^*, let $x\backslash L = \{y | xy \in L\}$. The cardinality of a set S is denoted by $|S|$.

A language L is said to have *the prefix-free property* if and only if for any $x \in \Sigma^+$, $y \in \Sigma^*$, both x and xy are in L implies that $y = \lambda$.

A (directed finite) labeled graph is an ordered triple (V, E, Σ), where Σ is a finite alphabet of labels, V is a finite set of nodes and $E = \{(p, a, q) | p, q \in V, a \in \Sigma\}$ is a finite set of edges.

A sequence of edges: $\sigma = (p_1, u_1, q_1), (p_2, u_2, q_2), \ldots, (p_n, u_n, q_n)$, $(n \geqslant 1)$, is a *path* from p_1 to q_n if and only if $q_i = p_{i+1}$ for all i $(1 \leqslant i < n)$. In this case, a string $x = u_1 u_2 \cdots u_n$ is called a *path string* of σ.

2.2. Polynomial-time identification in the limit from positive data

In this paper we adopt a slight modification of the original definition in [17] for the notion of polynomial-time identification in the limit from positive data. (In fact, the definition turns out to be a restricted version of the Pitt's definition where positive data is only provided.)

For any class of languages to be identified, let $\mathscr{R}$ be a *class of representations* for the class of languages. Given an r in $\mathscr{R}$, $L(r)$ denotes the language represented by r. 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 strings not in $L(r)$ appear in the sequence. Each element of $L(r)$ is called a *positive example* (or simply, *example*) of $L(r)$.

Let r be a representation in $\mathscr{R}$. An algorithm $\mathscr{A}$ is said to *identify r in the limit from positive data* if and only if $\mathscr{A}$ takes any positive presentation of $L(r)$ as input, and outputs an infinite sequence of r_is in $\mathscr{R}$ such that there exist r' in $\mathscr{R}$ and $j > 0$ so that for all $i \geqslant j$, the ith conjecture (representation) r_i is identical to r' and $L(r') = L(r)$. A class $\mathscr{R}$ is *identifiable in the limit from positive data* if and only if there exists an algorithm $\mathscr{A}$ that, given any r in $\mathscr{R}$, identifies r in the limit from positive data.

Let $\mathscr{A}$ be an algorithm for identifying $\mathscr{R}$ in the limit from positive data. Suppose that after examining i examples, the algorithm $\mathscr{A}$ conjectures some r_i. We say that $\mathscr{A}$ makes an *implicit error of prediction* at step i if r_i is not consistent with the $(i+1)$st example, i.e., if $L(r_i)$ fails to contain the $(i+1)$st example.

A class $\mathscr{R}$ is *polynomial-time identifiable in the limit from positive data* if and only if there exists an algorithm $\mathscr{A}$ for identifying $\mathscr{R}$ in the limit from positive data with the property that there exist polynomials p and q such that for any n, for any r of size n, and for any positive presentation of $L(r)$, the number of implicit errors of prediction made by $\mathscr{A}$ is at most $p(n)$, and the time used by $\mathscr{A}$ between receiving the ith example w_i and outputting the ith conjecture r_i is at most $q(n, m_1 + \cdots + m_i)$, where $m_j = len(w_j)$ and the size of r is the length of a description for r.

3. Very simple grammars and languages

Let $G = (V_N, \Sigma, P, S)$ be a context-free grammar (CFG) in Greibach normal form (GNF), i.e., each rule of P is of the form: $A \to a\alpha$, where $A \in V_N$, $a \in \Sigma$, $\alpha \in V_N^*$. If for

all $(a,A)\in\Sigma\times V_N$, there is at most one rule of the form $A\to a\alpha$ (for some $\alpha\in V_N^*$) in P, then G is said to be *simple deterministic*.

Let G be a *CFG* in *GNF*. 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 an *a-handle rule*. Then, G is said to be *very simple* if and only if for each a in Σ, there exists exactly one a-handle rule in P. A language L is *very simple* if and only if 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.)

Let $S\Rightarrow_L^* x\alpha$ be the left-most derivation in $G=(V_N,\Sigma,P,S)$, where $x\in\Sigma^*$, $\alpha\in V_N^*$. Then, define $L(\alpha)=\{w\in\Sigma^*\mid\alpha\Rightarrow_L^* w\}$. A CFG G is *reduced* if and only if for any α such that $S\Rightarrow_L^* x\alpha$ in G, it holds that $L(\alpha)\neq\emptyset$.

Example 1 (Laird and Gamble [11]). Consider a *CFG* $G=(\{S,C,L,T,R\},\ \{p,s,t,x,\wedge,(,),0,\bullet\},P,S)$, where P consists of the following:

$$\begin{array}{lll} S\to pLTCTCTR, & S\to,\wedge LSCSR, & S\to t,\\ T\to sLTR, & T\to x, & T\to 0,\\ C\to\bullet, & L\to(, & R\to). \end{array}$$

It is easily seen that the grammar G is very simple and that, for example, strings

"$p(s(0)\bullet x\bullet 0)$" and " $\wedge(p(0\bullet 0\bullet 0)\bullet p(s(0)\bullet 0\bullet s(0)))$"

are generated by G. If we take p, s and $\wedge$ as a predicate symbol "**plus**", a function symbol "*successor*", and a logical "*and*" in prefix notation, respectively, then these two strings are interpreted as logical formulas: **plus**$(s(0),x,0)$ and **plus**$(0,0,0)\wedge$**plus**$(s(0),0,s(0))$. Note that "t", "x", and "$\bullet$" denote a logical constant **true**, a variable, and comma, respectively. Thus, the language $L(G)$ defines a subset of formulas in first-order predicate logic which is not a regular language. Note that the very simple grammar G is reduced.

[**Convention**] In what follows, it is assumed that very simple grammars we consider in this paper are all *reduced*. Further, a notation $\Rightarrow$ (or $\underset{G}{\Rightarrow}$) always denotes the leftmost derivation (by G).

3.1. Basic properties

The next result immediately follows from the definition.

Lemma 1. *Let L be a very simple language.*

(i) [Prefix-free Property] *For each x in Σ^* and y in Σ^+, if $x\in L$, then $xy\notin L$.*

(ii) *For each w in L, if $len(w)\geqslant 2$, then the initial symbol of w must differ from the last symbol of w.*

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

Thus, the class of very simple languages forms a proper subclass of simple deterministic languages by Korenjak and Hopcroft [10]. Further, it is easy to see that the class 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 left Szilard language of G.*

Proof. For a given very simple grammar $G=(V_N,\Sigma,P,S)$, define a renaming f by: for each $\pi_a\in\mathrm{Label}(P)$, $f(\pi_a)=a$, where π_a is a label of the a-handle rule. Then, it is obvious that $S\Rightarrow^{\pi_{a_1}\cdots\pi_{a_n}}a_1\cdots a_n\in\Sigma^*$ if and only if $f(\pi_{a_1}\cdots\pi_{a_n})=a_1\cdots a_n\in\Sigma^*$. Thus, $L(G)=f(D_L(G))$ holds. □

Since left Szilard languages of *CFGs* of finite index (i.e., of nonterminal bounded *CFGs*) are regular [15], it is seen from Example 1 that there exists a very simple language L which is of infinite index (i.e., not nonterminal bounded).

3.2. Closure properties

In this subsection, we will present the closure properties of very simple languages.

Lemma 3. *$L_0=\{a^{3n}b|n\geqslant 0\}$ is not a very simple language.*

Proof. Suppose that $L_0=L(G)$ for some very simple grammar $G=(V_N,\{a,b\},P,S)$, where we may assume that $V_N=\{S,X\}$, $P=\{S\to a\alpha,X\to b\beta\}$ $(\alpha,\beta\in V_N^*)$. Since b is in L_0, there exists a rule: $S\to b$. Hence, the b-handle rule is $S\to b$ and $X=S$, $\beta=\lambda$. Further, since $aaab$ is in L_0, there must exist derivations $S\Rightarrow a\alpha$ and $\alpha\Rightarrow^* aab$. Let $\alpha=Y\alpha'$ $(Y\in V_N,\alpha'\in V_N^*)$, then there exist a rule $Y\to a\gamma$ and a derivation $\gamma\alpha'\Rightarrow^* ab$. From the rules $S\to a\alpha$ and $Y\to a\gamma$, $S=Y$ and $\alpha=\gamma$ follow, and hence the a-handle rule is $S\to aS\alpha'$. Since $\alpha(=S\alpha')\Rightarrow^* aab\alpha'^3$ and $\alpha\Rightarrow^* aab$, it must hold that $\alpha'=\lambda$. Thus, we have that $P=\{S\to aS,S\to b\}$ and hence, $L(G)=\{a^nb|n\geqslant 0\}$ contradicting to that $L_0=L(G)$. □

Theorem 4. *The class of very simple languages is closed under none of the following: union, concatenation, intersection, complement, Kleene closure $(+,*)$, (λ-free) homomorphim, inverse homomorphism, intersection with regular languages, or reversal.*

Proof. (*Union*:) Consider $L_1=\{ab\}$ and $L_2=\{abc\}$ which are clearly very simple. Since very simple (or simple deterministic) languages have the prefix-free property [8], $L_1\cup L_2=\{ab,abc\}$ is not very simple.
(*Concatenation*:) It is obvious that $L_3=\{ba\}$ is very simple, while $L_1L_2=\{abba\}$ is not, as seen in Example 2, which is shown from (ii) of Lemma 1.

(*Intersection*:) Let $L_4=\{ab^mcd^nef^ng|m,n\geqslant 0\}$ and $L_5=\{ab^mcd^mef^ng|m,n\geqslant 0\}$ be languages generated by very simple grammars G and G' with the sets of rules:

$$\begin{array}{llll} S\to aABC & A\to bA & S'\to aA'B'C' & A'\to c \\ A\to c & B\to dBD & A'\to bA'D' & C'\to fC' \\ B\to e & C\to g & B'\to e & C'\to g \\ D\to f & & D'\to d. & \end{array}$$

Then, $L_4\cap L_5=\{ab^ncd^nef^ng|n\geqslant 0\}$ is a language which is proven to be noncontext-free.

(*Complement*:) $L_6=\{a\}$ is clearly a very simple language over Σ (containing $\{a\}$), while its complement $L_6^c=\Sigma^*-L_6$ is not from (ii) of Lemma 1, because L_6^c contains $\{a^n|n\geqslant 2\}$.

(*Kleene closure*:) Consider again L_6. Then, L_6^*(or L_6^+) is not very simple, which is proved from (ii) of Lemma 1.

(*Homomorphism*:) Consider a very simple language $L_7=\{a^nb|n\geqslant 0\}$ and a homomorphism h_1 defined by $h_1(a)=a$, $h_1(b)=\lambda$. Then, $h_1(L_7)=\{a^n|n\geqslant 0\}$ is not very simple. Further, consider a λ-free homomorphism h_2 defined by $h_2(a)=a$, $h_1(b)=a$. Then, $h_2(L_7)=\{a^n|n\geqslant 1\}$ is not very simple.

(*Inverse homomorphism*:) For the very simple language L_6, consider a homomorphism h defined by $h(b)=a, h(c)=\lambda$. Then, $h^{-1}(L_6)=\{c^mbc^n|m,n\geqslant 0\}$ is not very simple, as seen in Example 2.

(*Intersection with regular languages*:) For the very simple language L_7, consider $L_7\cap(a^3)^*b=\{a^{3n}b|n\geqslant 0\}$ which is not very simple from Lemma 1.

(*Reversal*:) For the very simple language L_7, the reversal $L_7^R=\{ba^n|n\geqslant 0\}$ is not very simple, because it does not have the prefix-free property of (i) of Lemma 1. □

3.3. Other properties of very simple grammars

Let $G=(V_N,\Sigma,P,S)$ be any very simple grammar. Then, a symbol a in Σ is said to be *terminating* (in G) if the a-handle rule is of the form $A\to a$. A symbol a in Σ is said to be *starting* if and only if there is a string $w\in L(G)$ whose first symbol is a. A symbol a in Σ is said to be *final* if there is a string $w\in L(G)$ whose last symbol is a. Note that the notion of "terminating" depends on G, while those of "starting" and "final" do not.

The next lemma almost immediately follows from definitions.

Lemma 5. *Let w,w_1,w_2 be in a very simple language L. Then, for any $G=(V_N,\Sigma,P,S)$ generating L, and 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: $S\to a\alpha$ for some $\alpha\in V_N^*$,*

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

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

(4) *if* $w_1 = xby_1$ *and* $w_2 = xcy_2$ *for some* $x, y_i \in \Sigma^*$, *then there exist* $X \in V_N$ *and* α, $\beta \in V_N^*$ *such that* $X \to b\alpha$ *and* $X \to c\beta$ *are in* P.

The following are useful for our later discussion.

Lemma 6. *Let* $G = (V_N, \Sigma, P, S)$ *be any very simple grammar and let* α *and* α' *be in* V_N^+. *Further, let* $x \in \Sigma^+, u, u_1, u_2 \in \Sigma^*$. *Then,*
(1) *if* $S \Rightarrow^* x\alpha$ *and* $S \Rightarrow^* x\alpha'$, *then* $\alpha = \alpha'$ (Forward determinism),
(2) *if* $\alpha \Rightarrow^* x$ *and* $\alpha' \Rightarrow^* x$, *then* $\alpha = \alpha'$ (Backward determinism),
(3) *if* $S \Rightarrow^* u_1\alpha \Rightarrow^* u_1 x$ *and* $S \Rightarrow^* u_2\alpha' \Rightarrow^* u_2 x$, *then* $u_1 \backslash L = u_2 \backslash L$.

Lemma 7 (Iteration Lemma). *Let* L *be a very simple language. If* $w = ux^n v$ *and* $w' = uv$ *are in* L *for some* $n \geqslant 1$, $u \in \Sigma^*$, $x, v \in \Sigma^+$, *then for each* $i \geqslant 0$, $ux^i v$ *is in* L.

Finally, we can easily make a characterization on the relationships among the classes of regular, zero-reversible, and very simple languages.

A regular language L is *zero-reversible* if and only if whenever $u_1 x$ and $u_2 x$ are in L, $u_1 \backslash L = u_2 \backslash L$ holds [2].

From the definitions together with Lemma 6, the next theorem almost immediately follows.

Theorem 8. *For a language* L *over* Σ, *the following are equivalent*:
(1) L *is the left Szilard language of a linear grammar*,
(2) L *is very simple and regular*,
(3) L *is very simple and zero-reversible*.

Thus, the intersection of the class of very simple languages and that of regular languages is exactly the class of (left) Szilard languages of linear grammars. (See Fig. 1.)

3.4. Schema representations of very simple grammars

We introduce a new representation method for very simple grammars called *grammar schemata*, which enables us to deal with very simple grammars in a uniform manner. Any very simple grammar can be analysed and decomposed into two constructs: a grammar schema and its interpretation. Informally, the former provides a skeletal structure of the original grammar, while the latter specifies the details of it.

Given a finite alphabet Σ, let $V_{N,S} = \{X_a | a \in \Sigma\} \cup \{S\}$ 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 over all elements from $V_{N,S}^*$. Let $\Gamma = (V_{N,S} \cup PAR)$. Then, a construct $X \to ax$, where $X \in V_{N,S}$, $a \in \Sigma$ and $x \in \Gamma^*$, is called a *rule form*. We call a quadruple $\mathcal{G} = (V_{N,S}, \Sigma, P, S)$ a *grammar schema* if P is a finite set of rule forms.

An *interpretation* $I = (f_n, f_p)$ over Σ is an ordered pair of mappings, where f_n is a coding defined on $V_{N,S}$ such that $f_n(S) = S$, and f_p is a homomorphism defined on PAR

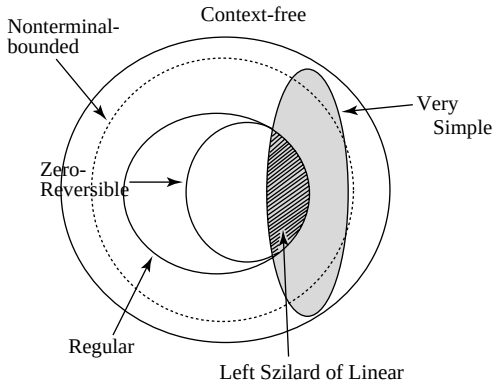

Fig. 1. Language class relations.

such that for all $x \in PAR$, $f_p(x)$ is in Γ^*. Then, given a grammar schema $\mathcal{G}$, let $I(\mathcal{G})$ be a quadruple defined by $(f_n(V_{N,S}), \Sigma, I(P), S)$, where $I(P) = \{f_n(X) \to a f_p(x) | X \to ax \in P\}$. Thus, I induces a mapping on the set of all grammar schemata. An interpretation I is said to be *ground* if and only if for all $x \in PAR$, $f_p(x) \in V_{N,S}^*$. Note that any *CFG* in Greibach normal form is taken as a grammar schema of special type.

3.5. Finding very simple grammars consistent with positive data

In this subsection, we consider the following problem for very simple grammars. Suppose that we are given a finite set of strings $R = \{w_1, w_2, \ldots, w_t\}$ from an unknown very simple language $L = L(G_*)$ for some very simple grammar G_*. Starting with constructing the initial grammar schema $\mathcal{G}_0 = (V_{N,S}, \Sigma, \{X_a \to a x_a | a \in \Sigma\}, S)$, our goal here is to find (identify) a ground interpretation I such that $I(\mathcal{G}_0)$ is consistent with R, i.e., $L(I(\mathcal{G}_0))$ contains R, where $\Sigma = \text{alph}(R)$.

In order to identify such a ground interpretation $I = (f_n, f_p)$ satisfying the requirements, we have to determine f_n and f_p. First, we describe the manner of determining f_n.

[Computation of f_n]

We will demonstrate the way how to determine a mapping f_n from a graph called the *structure graph of R* denoted by T_R.

Let

$$\Sigma_s(R) = \{a \in \Sigma | \exists w \in R, \exists x \in \Sigma^*(w = ax)\}.$$

$$\Sigma_f(R) = \{a \in \Sigma | \exists w \in R, \exists x \in \Sigma^*(w = xa)\}.$$

Then, we may define that for all $a \in \Sigma_s(R)$, $f_n(X_a) = S$. In order to determine $f_n(X)$ for other $X \in V_{N,S} - (\{X_a | a \in \Sigma_s(R)\} \cup \{S\})$, we construct a directed graph called the *structure graph* T_R of R as follows.

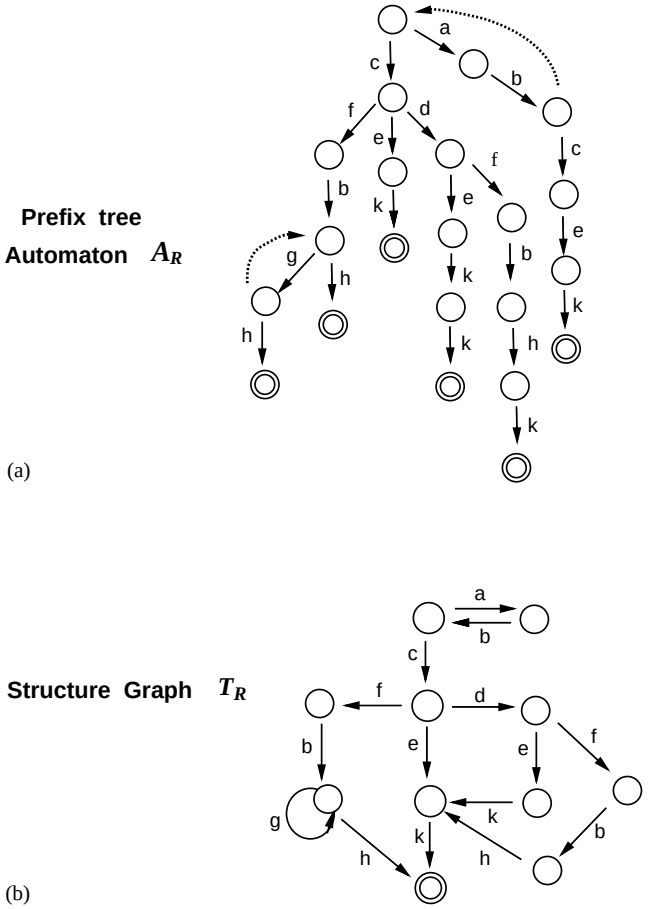

Fig. 2. Prefix tree automaton A_R and structure graph T_R. In (a) two nodes connected by a dotted arrow are to be identified.

From a given R, we first construct the prefix tree automaton A_R for R in the sense of the paper [2]. Then, using the properties of very simple languages presented in Lemmas 6 and 7, from A_R we construct the structure graph T_R. Once the graph T_R is obtained, a mapping f_n is immediately determined from T_R. Let us illustrate this procedure using an example.

Example 3. Suppose that $R=\{abcek, cdek^2, cek, cdfbhk, cfbh, cfbgh\}$ $(=\{w_1,\ldots,w_6\})$ is given as a sample set of $L(G_*)$ for some unknown very simple grammar G_*. Then, we have the prefix tree automaton A_R for R shown in (a) of Fig. 2. Note that $\Sigma=\mathrm{alph}(R)=\{a,b,c,d,e,f,g,h,k\}$.

Then, making use of properties of very simple grammars in Lemma 6, we merge all the nodes (nonterminals) which must be identical in any correct grammar for $L(G_*)$.

By Lemma 7, we know that ab and g are loopings. Hence, we eventually obtain the *structure graph of* R, T_R, shown in (b) of Fig. 2.

Now, from 4 of Lemma 5, we know that if terminal symbols s and t are the labels of edges that have an identical starting node in T_R, then X_s and X_t must be identical. Note that, for our example, $\Sigma_s(R)=\{a,c\}$ and $\Sigma_f(R)=\{h,k\}$. Hence, we have that $X_a=X_c(=S)$, $X_d=X_e=X_f$, and $X_g=X_h$. In this example, we define f_n as follows:

$$f_n(X_a)=f_n(X_c)=S,\quad f_n(X_e)=f_n(X_f)=X_d,\quad f_n(X_h)=X_g,$$

$$\text{and for other } X\in\{X_b,X_d,X_g,X_k\},\quad f_n(X)=X.$$

Thus, in what follows, we assume that f_n is defined so that for each X in $V_{N,S}-(\{X_a|a\in\Sigma_s\}\cup\{S\})$ the image $f_n(X)$ is the one with the first index in alphabetical order, among those to be indentified.

Note 1. As described above, f_n is determined from the structure graph T_R whose construction is solely due to the properties proved valid in Lemmas 5, 6 and 7, which assures that any very simple grammar consistent with R must satisfy those properties.

[Computation of f_p]

Recall that

$$\Sigma_f(R)=\{a\in\Sigma|\exists w\in R,\exists x\in\Sigma^*(w=xa)\}.$$

We say that a parameter x_a is *empty* if it is λ. Obviously, for all $a\in\Sigma_f(R)$, x_a must be empty. Also, for all $a\in\Sigma_s(R)-\Sigma_f(R)$, x_a is *not* empty (i.e., a is not terminating). These facts are used for determining f_p below.

In what follows, first we are going to present procedures and definitions, and then we will show a sample run to illustrate them.

Determining unknown values of f_p is performed in the following three steps.

Step 1: *Constructing Length Equation Set* Lg(R). Let $\mathrm{alph}(R)=\Sigma=\{a_1,\ldots,a_r\}$ be an ordered alphabet. For $w\in\Sigma^*$ and $a\in\Sigma$, $\#_a(w)$ denotes the number of occurrences of a in w.

For all $a\in\Sigma$, let $n_a=len(f_p(x_a))-1$, where $f_p(x_a)$ is the value (a string of nonterminals) to be determined for the unknown parameter x_a. (Note that n_a indicates the length increase of a sentential form induced by one application of the a-handle rule. Hence, it holds that $n_a=-1$ if and only if $f_p(x_a)=\lambda$.) Further, let $R_a=\{w\in R|w \text{ contains } a\}$. Then, we have that for any $a\in\Sigma$,

(C1) $-1\leqslant n_a\leqslant B(a,w)$ (where $B(a,w)=len(w)/\#_a(w)$ and w is a string of the minimum length in R_a),

(C2) if $a\in\Sigma_s(R)-\Sigma_f(R)$, then $n_a\geqslant 0$.

It is easily seen that for each $w_i\in R$,

$$\#_{a_1}(w_i)n_{a_1}+\cdots+\#_{a_r}(w_i)n_{a_r}=-1\quad\cdots\quad(\ell.w_i).$$

By constructing the equation $(\ell.w_i)$ for each w_i in R in the manner described above, we have a set of equations $\mathrm{Lg}(R)=\{(\ell.w_1),\ldots,(\ell.w_t)\}$. Since for $a\in\Sigma_f(R)$, $f_p(x_a)=\lambda$, we know that $n_a=-1$.

Let $\mathrm{Lg}(R')=\{(\ell.w_i)'|1\leqslant i\leqslant t\}$ be the set of equations $(\ell.w_i)'$ obtained from $(\ell.w_i)$ by substituting $n_a=-1$ for each $a\in\Sigma_f(R)$.

In order to find a concrete very simple grammar G which is consistent with the given positive data R, first we have to solve the set of equations $\mathrm{Lg}(R')$, so that we may obtain a possible ground interpretation $I=(f_n,f_p)$ satisfying the requirements.

Step 2: *Solving Equation Set* Lg(R′). Let $\Sigma'(=\{a_1,\ldots,a_m\})$ be the ordered set $\mathrm{alph}(R)-\Sigma_f(R)$ and let $LPAR=\{n_a|a\in\Sigma'\}$ be the set of length parameters to be solved. Further, let $\mathrm{vec}(w_i)$ be a row vector defined by

$$\mathrm{vec}(w_i)=(\#_{a_1}(w_i),\ldots,\#_{a_m}(w_i)),$$

where, each $\#_{a_j}(w_i)$ is a coefficient of n_{a_j} in the left-hand side of $(\ell,w_i)'$.

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

$$(i,j)\text{-entry of } M_R \overset{\text{def}}{\Leftrightarrow} \text{the coefficient of the } j\text{th parameter } n_{a_j} \text{ in the } i\text{th equation } (\ell.w_i)',$$

that is,

$$M_R=\begin{pmatrix}\mathrm{vec}(w_1)\\ \vdots\\ \mathrm{vec}(w_t)\end{pmatrix}.$$

(Note that $t=|R|$ and $m=|LPAR|$.) Then, the set of equations $\{(\ell.w_i)'|1\leqslant i\leqslant t\}$ is reformulated by a matrix equation:

$$M_R\mathscr{X}^{\mathrm{T}}=\mathscr{C}^{\mathrm{T}}\quad\cdots\quad(\mathrm{E})$$

where $\mathscr{X}=(n_{a_1},\ldots,n_{a_m})$ is a vector of length parameters, $\mathscr{C}=(k'_1,\ldots,k'_t)$ is a constant vector and each k'_j is the constant in the right-hand side of $(\ell.w_i)'$.

Solving the matrix equation (E) is performed in a usual manner in linear algebra. In some case, we may have indeterminate parameters in the solution. The next lemma is one of the well-known results in linear algebra.

Lemma 9. *The equation* (E) *has a solution if and only if the rank of* M_R *is equal to that of* $M_R\mathscr{C}^{\mathrm{T}}$.

Let m and t be the numbers of parameters and equations in (E), respectively. Then, we also know the following.

Lemma 10. *Suppose that* $m=t$ *holds. Then,* M_R *is nonsingular if and only if* (E) *has a unique solution.*

Thus, if $m = \mathrm{rank}(M_R)$, then the equation (E) has a unique solution. And, if $m > t$ or $m \neq \mathrm{rank}(M_R)$, then (E) is solved in part. That is, let $\mathrm{Sol}(\mathrm{Lg}(R'))$ be the set of solutions of $\mathrm{Lg}(R')$. Then, we have that

$$\mathrm{Sol}(\mathrm{Lg}(R')) = \text{P-Sol}(\mathrm{Lg}(R')) \cup \mathrm{Uns}(\mathrm{Lg}(R')),$$

where P-Sol$(\mathrm{Lg}(R'))$ is the set of solutions partly solved, and $\mathrm{Uns}(\mathrm{Lg}(R'))$ is the set of equations involved in indeterminate parameters left unsolved.

Then, by assigning an appropriate value to each indeterminate parameter in Uns(Lg (R')), it is always possible to have a complete solution for $\mathrm{Lg}(R')$.

Step 3: *Determining the values of* f_p.

(3-1) First, we may define f_p in part by

$$\text{for all } a \text{ in } \Sigma_f(R), \quad f_p(x_a) = \lambda.$$

(3-2) Determining the values of f_p for other parameters is performed by *simulating* the derivation for each w in R as follows:

Let $\mathscr{X} = (\ell_{a_1}, \ldots, \ell_{a_m})$ be a solution vector obtained from (E) in (Step 2). Then, for $a \in \Sigma'$ such that $\ell_a \geqslant 0$, we may assume that $f_p(x_a) = Z_{a,1} \cdots Z_{a,\ell_a+1}$. For $a \in \Sigma'$ such that $\ell_a = -1$, we may trivially define $f_p(x_a) = \lambda$. The set $\{f_n(X_a) \to a f_p(x_a) | a \in \Sigma'\}$ is called the *set of candidate rules* obtained from $\mathscr{X}$.

In order to identify each unknown nonterminal $Z_{a,j}$ introduced above, for each $w = b_1 \cdots b_k \in R$, we simulate the derivation for w using each candidate b_i-handle rule: $f_n(X_{b_i}) \to b_i Z_{b_i,1} \cdots Z_{b_i,\ell_{b_i}+1}$ or $f_n(X_{b_i}) \to b_i$ (for $i = 1, \ldots, k$).

Starting with $S \Rightarrow b_1 Z_{b_1,1} \cdots Z_{b_1,\ell_{b_1}+1}$, since $Z_{b_1,1}$ must produce $b_2 u$ (for some $u \in \Sigma^*$), it must hold that $Z_{b_1,1} = f_n(X_{b_2})$ and that $f_n(X_{b_2}) \to b_2 Z_{b_2,1} \cdots Z_{b_2,\ell_{b_2}+1}$. Thus, we have that

$$\begin{aligned} S &\Rightarrow b_1 Z_{b_1,1} \cdots Z_{b_1,\ell_{b_1}+1} \\ &\Rightarrow b_1 b_2 Z_{b_2,1} \cdots Z_{b_2,\ell_{b_2}+1} Z_{b_1,2} \cdots Z_{b_1,\ell_{b_1}+1} \\ &\quad (= b_1 b_2 Z_{b_{i,3}} \cdots Z_{b_{i,p}}). \end{aligned}$$

In the same manner, for each j $(3 \leqslant j \leqslant p)$, we can determine the value of $Z_{b_{i,j}}$. That is, for each $i = 1, \ldots, k$, $f_p(x_{b_i})$ is determined. Thus, a *simulation of the derivation* for w is completed.

Determining complete values of f_p is performed by combining the results of simulation of the derivations for all $w \in R$.

From Lemma 9, a corollary immediately follows.

Corollary 11. *It is decidable given any finite subset* R *of* Σ^* *whether or not there exists a very simple grammar consistent with* R.

(Recall that a finite set is not necessarily very simple.)

[Procedure **Consistent**(R)*]*

By Int(R) we denote the set of all ground interpretations $I=(f_n, f_p)$ *obtained from R in the manner described above*. Let CR$(\mathscr{X})$ be the set of candidate rules determined from a solution vector $\mathscr{X}$ of (E).

There are, in general, two cases for the simulation process of R via CR$(\mathscr{X})$: the simulation finishes without any *additional* nonterminal identification (i.e., merging two nonterminals) on f_n, or it forces us to make a new nonterminal identification. In the former case, CR$(\mathscr{X})$ is said to be *good*.

In order to obtain the least general grammar $I(\mathscr{G}_0)$ consistent with R, we want to *avoid* choosing an interpretation $I=(f_n, f_p)$ that causes the latter case, as much as we possibly can.

We say that a ground interpretation $I=(f_n, f_p)$ in Int(R) is *admissible to* R if and only if either there exists a good CR$(\mathscr{X})$ by which f_p is determined or there exists no good set of candidate rules.

Our goal here is to find a ground interpretation I admissible to R.

Now, let us define a *procedure* **Consistent**(R) to obtain I admissible to R from a given R. That is, **Consistent**(R) is defined as follows:

Input: R (a finite subset of the target language L_*);
Output: $I=(f_n, f_p)$ admissible to R;
Procedure:
  initialize $F_v=\emptyset$;
determine f_n by constructing the structure graph of R;
construct Lg(R) and solve it, where for each $a\in\Sigma_f(R)$, $n_a=-1$;
let S_v be the set of all solution vectors $\mathscr{X}$ satisfying (C1) and (C2);
while $S_v\neq\emptyset$ **do**
  begin
    choose any solution $\mathscr{X}$ from S_v to obtain the values for the rest
      of unspecified parameters of f_p;
    let $S_v:=S_v-\{\mathscr{X}\}$ and $F_v:=F_v\cup\{\mathscr{X}\}$;
    determine CR$(\mathscr{X})$ (the set of candidate rules) from $\mathscr{X}$;
    simulate R with CR$(\mathscr{X})$ to determine f_p;
    if CR$(\mathscr{X})$ is good, **then** output $I=(f_n, f_p)$ and halt;
  end
take any solution $\mathscr{X}$ from F_v;
let f_p be the one obtained by simulating R via CR$(\mathscr{X})$ (where additional
  nonterminal identifications are allowed);
renew f_n by making additional identifications obtained through
  the simulation of R via CR$(\mathscr{X})$;
output $I=(f_n, f_p)$ and halt.

We write the input–output relation as $I=$ **Consistent**(R). (Note that from the manner of construction, $I(=$ **Consistent**$(R))$ can obviously provide a very simple grammar $I(\mathscr{G}_0)$ that is consistent with R.)

Note 2. In the actual implementation of the procedure above, one can make **Consistent** (R) more efficient in an incremental fashion so that it may produce and try a new solution vector $\mathscr{X}$ only after the current CR($\mathscr{X}$) turned out to be not good. We chose the current presentation for **Consistent**(R), because of its simplicity and understandability.

Example 4. Returning to the previous example, let R be the sample set from Example 3. Let $\mathscr{G}_0 = (V_{N,S}, \Sigma, P_0, S)$, where $V_{N,S} = \{S, X_a, X_b, X_c, X_d, X_e, X_f, X_g, X_h, X_k\}$. Since we have already obtained f_n, all we have to do is to construct Lg(R) and solve it for finding f_p which satisfies the requirements.

Firstly, since $\Sigma_f(R) = \{h, k\}$, we have already obtained that

$$f_p(x_h) = f_p(x_k) = \lambda, \quad \text{(i.e., } n_h = n_k = -1\text{)}.$$

From R, we have that $\mathrm{Lg}(R) = \{(\ell.w_1), (\ell.w_2), \ldots, (\ell.w_6)\}$, where

$$\begin{aligned}
n_a + n_b + n_c + n_e + n_k &= -1 \quad \cdots \quad (\ell.w_1)\\
n_c + n_d + n_e + 2n_k &= -1 \quad \cdots \quad (\ell.w_2)\\
n_c + n_e + n_k &= -1 \quad \cdots \quad (\ell.w_3)\\
n_b + n_c + n_d + n_f + n_h + n_k &= -1 \quad \cdots \quad (\ell.w_4)\\
n_b + n_c + n_f + n_h &= -1 \quad \cdots \quad (\ell.w_5)\\
n_b + n_c + n_f + n_g + n_h &= -1 \quad \cdots \quad (\ell.w_6).
\end{aligned}$$

That is, the set of equations Lg(R') comprising

$$\begin{aligned}
n_a + n_b + n_c + n_e &= 0 \quad \cdots \quad (\ell.w_1)'\\
n_c + n_d + n_e &= 1 \quad \cdots \quad (\ell.w_2)'\\
n_c + n_e &= 0 \quad \cdots \quad (\ell.w_3)'\\
n_b + n_c + n_d + n_f &= 1 \quad \cdots \quad (\ell.w_4)'\\
n_b + n_c + n_f &= 0 \quad \cdots \quad (\ell.w_5)'\\
n_b + n_c + n_f + n_g &= 0 \quad \cdots \quad (\ell.w_6)'
\end{aligned}$$

is obtained, where $\Sigma'(= \mathrm{alph}(R) - \Sigma_f(R)) = \{a, b, c, d, e, f, g\}$. Then, construct a matrix equation

$$M_R \mathscr{X}^{\mathrm{T}} = \mathscr{C}^{\mathrm{T}} \cdots \quad \text{(E)}$$

where $\mathscr{X} = (n_a, n_b, n_c, n_d, n_e, n_f, n_g)$, $\mathscr{C} = (0, 1, 0, 1, 0, 0)$, and

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

Computing (E) is performed in a usual manner:

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

Thus, we have that $\mathrm{Sol}(\mathrm{Lg}(R')) = \{n_d = 1, n_g = 0\} \cup \mathrm{Uns}(\mathrm{Lg}(R'))$, where $\mathrm{Uns}(\mathrm{Lg}(R')) = \{n_a + n_b = 0, n_c + n_e = 0, n_b + n_c + n_f = 0\}$.

By taking (C1) and (C2) into consideration, we may choose a set of solutions $\{n_a = 1, n_b = -1, n_c = 0, n_e = 0, n_f = 1\}$ that produces a solution vector:

$$\mathscr{X} = (1, -1, 0, 1, 0, 1, 0)\ (= (n_a, n_b, n_c, n_d, n_e, n_f, n_g))$$

In order to determine the values of f_p, the set of candidate rules $\mathrm{CR}(\mathscr{X})$ is constructed:

$S \to aZ_{a,1}Z_{a,2}$, $\quad X_b \to b$
$S \to cZ_{c,1}$, $\quad X_{td} \to dZ_{d,1}Z_{d,2}$
$X_d \to eZ_{e,1}$, $\quad X_d \to fZ_{f,1}Z_{f,2}$
$X_g \to gZ_{g,1}$.
(Note that $f_n(X_a) = f_n(X_c) = S$, $f_n(X_e) = f_n(X_f) = X_d$.)

In simulating the derivation for $w_1 = abcek$, since $S \Rightarrow aZ_{a,1}Z_{a,2} \Rightarrow^* abcek$, it must hold that $Z_{a,1} = X_b$ and that $Z_{a,2} = X_c (= S)$. Further, from $S \Rightarrow cZ_{c,1} \Rightarrow cek$, it must hold that $Z_{c,1} = X_e (= X_d)$ and that $Z_{e,1} = X_k$. Thus, we have that

$$f_p(x_a) = Z_{a,1}Z_{a,2} = X_bS, \quad f_p(x_c) = X_d \text{ and } f_p(x_e) = X_k.$$

(With these assignments for length parameters, we succeed in simulating the derivations for w_2 and w_3.)

In a similar manner, from the simulation of w_4,

$$f_p(x_d) = X_dX_k, \quad f_p(x_f) = X_bX_g \quad \text{and } f_p(x_g) = X_g$$

are obtained. Thus, we complete a solution for f_p using $\mathrm{CR}(\mathscr{X})$. (Note that $\mathrm{CR}(\mathscr{X})$ is good.) Further, we can see that any other solution fails in simulating R without any additional nonterminal identification on f_n in this sample run.

This eventually provides the unique interpretation $I=(f_n, f_p)$ admissible to R, where

$$\begin{aligned}
&f_n(X_a)=S, && f_p(x_a)=X_bS\\
&f_n(X_b)=X_b, && f_p(x_b)=\lambda\\
&f_n(X_c)=S, && f_p(x_c)=X_d\\
&f_n(X_d)=X_d, && f_p(x_d)=X_dX_k\\
&f_n(X_e)=X_d, && f_p(x_e)=X_k\\
&f_n(X_f)=X_d, && f_p(x_f)=X_bX_g\\
&f_n(X_g)=X_g, && f_p(x_g)=X_g\\
&f_n(X_h)=X_g, && f_p(x_h)=\lambda\\
&f_n(X_k)=X_k, && f_p(x_k)=\lambda.
\end{aligned}$$

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

$$\begin{aligned}
&S\to aX_bS, && X_b\to b, && S\to cX_d,\\
&X_d\to dX_dX_k, && X_d\to eX_k, && X_d\to fX_bX_g,\\
&X_g\to gX_g, && X_g\to h, && X_k\to k.
\end{aligned}$$

4. Identifying very simple grammars from positive data

4.1. Characterizing the structure of $\mathscr{I}_\Sigma$

Throughout this section, we assume that $G_*=(V_{N*}, \Sigma, P_*, S_*)$ is a target very simple grammar and that R is a finite subset of $L(G_*)$. Further, let $\mathscr{I}_\Sigma=\{I=(f_n, f_p)|I$: ground interpretation over $\Sigma\}$, where $|\Sigma|\geqslant 2$. (It is obvious that any (nonempty) very simple language over the unary alphabet trivially becomes $\{a\}$.)

Recall that $\mathscr{G}_0=(V_{N,S}, \Sigma, \{X_a\to ax_a|a\in\Sigma\}, S)$ is the initial grammar schema. Now, for any $I=(f_n, f_p)$, $I'=(f'_n, f'_p)\in\mathscr{I}_\Sigma$, we define a binary relation $\preccurlyeq$ as follows: $f_n\preccurlyeq f'_n$ if and only if for every $X_a, X_b\in V_{N,S}(a\neq b)$, $f_n(X_a)=f_n(X_b)$ implies that $f'_n(X_a)=f'_n(X_b)$. When $f_n\preccurlyeq f'_n$ and $f_n\neq f'_n$, we denote it by $f_n\prec f'_n$. Further, we say that f_n is *incomparable* to f'_n if and only if $f_n\not\preccurlyeq f'_n$ and $f'_n\not\preccurlyeq f_n$ holds, where the symbol $\not\preccurlyeq$ denotes the negation of $\preccurlyeq$. (Recall that we assume all very simple grammars to be reduced.)

We now need to show the following series of technical lemmas.

Lemma 12. *Let $I=(f_n, f_p)$ and $I'=(f'_n, f'_p)$ be in $\mathscr{I}_\Sigma$ such that $L(I(\mathscr{G}_0))\subseteq L(I'(\mathscr{G}_0))$. Then, it holds that $f_n\preccurlyeq f'_n$.*

Proof. Let $I(\mathscr{G}_0)=G=(V_N, \Sigma, P, S)$ and $I'(\mathscr{G}_0)=G'=(V'_N, \Sigma, P', S')$ be very simple grammars such that $L(G)\subseteq L(G')$. Suppose that $f_n\not\preccurlyeq f'_n$, then we may assume that there

exist X_a, X_b $(a \neq b)$ in $V_{N,S}$ of $\mathscr{G}_0$ such that $f_n(X_a) = f_n(X_b)$ and $f'_n(X_a) \neq f'_n(X_b)$. There exists a string $w = xay$ in $L(G)$ such that

$$S \Rightarrow^* xA\alpha,\ A \Rightarrow a\gamma \quad \text{and} \quad \gamma\alpha \Rightarrow^* y \quad \text{for some } \alpha, \gamma \in V_N^*, \quad \text{where } A = f_n(X_a).$$

Further, since $f_n(X_a) = f_n(X_b)$, there also exists a string $w' = xbz$ in $L(G)$ such that

$$S \Rightarrow^* xA\alpha,\ A \Rightarrow b\beta \quad \text{and} \quad \beta\alpha \Rightarrow^* z \quad \text{for some } \beta \in V_N^*, \quad \text{where } A = f_n(X_b).$$

Since $L(G) \subseteq L(G')$, there must exist derivations in G' such that

$$S' \Rightarrow^* x\alpha',\ \alpha' \Rightarrow^* ay \quad \text{and} \quad \alpha' \Rightarrow^* bz \quad \text{for some } \alpha' \in V_N'^+.$$

Let X be the left-most nonterminal of α', then there must exist rules $X \to a\beta'$ and $X \to b\gamma'$ in P' of G', which leads to a contradiction that $X = f'_n(X_a) \neq f'_n(X_b) = X$ in V'_N of G'. □

Corollary 13. *Let $I = (f_n, f_p)$ and $I' = (f'_n, f'_p)$ be in $\mathscr{I}_\Sigma$. Then, $L(I(\mathscr{G}_0)) = L(I'(\mathscr{G}_0))$ implies $f_n = f'_n$.*

Lemma 14. *Let $I = (f_n, f_p)$ and $I' = (f'_n, f'_p)$ be in $\mathscr{I}_\Sigma$. Then,*
(i) *if $L(I(\mathscr{G}_0)) \subset L(I'(\mathscr{G}_0))$, then $f_n \prec f'_n$,*
(ii) *if $f_n = f'_n$ and $L(I(\mathscr{G}_0)) \neq L(I'(\mathscr{G}_0))$, then $L(I(\mathscr{G}_0))$ and $L(I'(\mathscr{G}_0))$ are incomparable.*

Proof. Let $G = I(\mathscr{G}_0) = (V_N, \Sigma, P, S)$ and $G' = I'(\mathscr{G}_0) = (V'_N, \Sigma, P', S')$. (i) Assume that $L(G) \subset L(G')$. Then, by Lemma 12, it holds that $f_n \preccurlyeq f'_n$. Now, by assuming that $f_n = f'_n$, we will derive a contradiction.

From the assumption that $L(G) \subset L(G')$, there exists a string $w_0 = a_1 a_2 \cdots a_k \in L(G') - L(G)$. Let

$$S \Rightarrow a_1\alpha_1 \Rightarrow^* a_1 \cdots a_j\alpha_j \Rightarrow^* a_1 \cdots a_{k-1}\alpha_{k-1} \Rightarrow a_1 \cdots a_k \quad \cdots \quad (*)$$

be a derivation for w_0 in G', where each $\alpha_j \in V_N'^+$. Since G fails to generate w_0, there are three possible cases in G.

Case 1. There exists j $(1 \leqslant j \leqslant k)$ such that $S \Rightarrow^* a_1 \cdots a_{j-1} f_n(X_b)\beta$ and $f_n(X_b) \neq f_n(X_{a_j})(b \neq a_j)$, for some $b \in \Sigma, \beta \in V_N^*$ in G: Let $f_n(X_b) \to b\gamma$ be in P and $\gamma\beta \Rightarrow^* u(\in \Sigma^*)$. (Recall our initial assumption that very simple grammars considered are all reduced.) Then, $S \Rightarrow^* a_1 \cdots a_{j-1} bu(= w) \in L(G)$. On the other hand, since $L(G) \subset L(G')$, there also exists a derivation for w in G'. Let $S' \Rightarrow^* a_1 \cdots a_{j-1}\alpha'$ and $\alpha' \Rightarrow^* bu$ in G', where $\alpha' \in V_N'^+$. Then, this together with $(*)$ implies that $\alpha' = \alpha_{j-1}$, $\alpha' \Rightarrow^* bu$ and $\alpha' \Rightarrow^* a_j \cdots a_k$. Therefore, we have that $f'_n(X_b) = f'_n(X_{a_j})$ (and $b \neq a_j$). Since $f_n = f'_n$, this also holds for G, i.e., $f_n(X_b) = f_n(X_{a_j})$, which is a contradiction.

Case 2. There exists $\beta \in V_N^+$ such that $S \Rightarrow^* w_0\beta$ and $\beta \Rightarrow^* x(\in \Sigma^+)$ in G. Then, $w_0 x$ is in $L(G)$. Since $L(G) \subset L(G')$, it holds that $w_0 x$ is in $L(G')$. This contradicts the prefix-free property of a very simple language $L(G')$.

Case 3. There exists j $(1 \leqslant j < k)$ such that $S \Rightarrow^* a_1 \cdots a_j (= y)$ in G: Since $L(G) \subset (G')$, it holds that y is in $L(G')$. This together with the fact that $ya_{j+1} \cdots a_k (= w_0)$ is in $L(G')$ contradicts the prefix-free property. Thus, we have that $f_n \prec f'_n$.

(ii) Suppose that $L(G) \subset L(G')$. Then, from (i) we have that $f_n \prec f'_n$, contradicting $f_n = f'_n$. Hence, it holds that $L(G) \not\subset L(G')$. Similarly, we also have that $L(G') \not\subset L(G)$. These, together with $L(G) \neq L(G')$, imply that $L(G)$ and $L(G')$ are incomparable. Thus, the proof is completed. □

Lemma 15. *For a finite subset R of $L(G_*)$, let $I = (f_n, f_p) =$ **Consistent**(R) and $G = I(\mathscr{G}_0)$. Then, either $L(G_*) = L(G)$ or $L(G_*) - L(G) \neq \emptyset$.*

Proof. Suppose otherwise, that is, $L(G_*) \subset L(G)$. (We prove by contradiction.) Then, from (i) of Lemma 14, we have that $f_{n*} \prec f_n$. (Note that $R \subseteq L(G_*)$ and $I = (f_n, f_p) =$ **Consistent**(R).) On the other hand, from the manner of constructing **Consistent**(R), there are two cases to be considered.

Case 1. A good CR($\mathscr{X}$) was found in the **while** loop: f_n is determined from the structure graph T_R whose construction is solely due to the properties common to all very simple grammars consistent with R and proved valid in Lemmas 5–7 (see Note 1 immediately after Example 3). This implies that $f_n \preccurlyeq f_{n*}$.

Case 2. Otherwise: In this case, since no good CR($\mathscr{X}$) was found for determining f_p, there must occur additional nonterminal identifications in simulating R. Let $\tilde{f}_n$ be the one, constructed from the structure graph T_R alone, from which the renewed version f_n is obtained by making such additions. Then, it holds that $\tilde{f}_n \prec f_{n*}$ and $\tilde{f}_n \prec f_n$, and we see that $\tilde{f}_n$ is the greatest lower bound of $\{f_n, f_{n*}\}$, i.e., $\tilde{f}_n = \inf\{f_n, f_{n*}\}$ in the lattice with $\preccurlyeq$ consisting of all possible f_ns with the maximum $\bar{f}_n$(where for any $X \in V_{N,S}$, $\bar{f}_n(X) = S$) and the minimum $\underline{f}_n$ (where $\underline{f}_n$ is an identity on $V_{N,S}$). Since $f_{n*} \prec f_n$, it holds that $\tilde{f}_n \prec f_{n*} \prec f_n$, which contradicts that $\tilde{f}_n$ is the greatest lower bound of $\{f_n\}$. □

4.2. Identification algorithm: IA

We now present an identification algorithm *IA* which is consistent, responsive and conservative [1], that is, intuitively, it always produces a guess consistent with the sample set R and makes at least one guess in response to each input before requesting another input, and only changes its guess when it conflicts with the sample set read in so far. The algorithm *IA* is given in Fig. 3, in which $\mathscr{G}_{0,\Sigma}$ denotes $(\{S\} \cup V_{N,\Sigma}, \Sigma, P_\Sigma, S)$, where $V_{N,\Sigma} = \{X_a | a \in \Sigma\}$, $P_\Sigma = \{X_a \to a x_a | a \in \Sigma\}$.

Lemma 16. *Let $G_{R_0}, G_{R_1}, \ldots, G_{R_i}, \ldots$ be the sequence of conjectured grammars produced by IA, where $G_{R_i} = I_i(\mathscr{G}_{0,\Sigma})$. Then, there exists $r \geqslant 0$ such that for all $i \geqslant 0$, $G_{R_r} = G_{R_{r+i}}$ and $L(G_{R_r}) = L(G_*)$.*

Proof. From the property of *IA*, in particular, of the procedure **consistent**(R), there is an upper bound B (depending on the size of G_*) for which for each $i \geqslant 1$, the number of candidate interpretations I_i(=**consistent**(R_i)) for the i-th conjecture G_{R_i} is no more than B. (Note that for each $i \geqslant 1$, and interpretation I_* for G_* is potentially included in the set of those candidate interpretations I_i.) Thus, there exists $r \geqslant 0$ such that for

Input: a positive presentation of a very simple language $L(G_*)$
Output: a sequence of very simple grammars $G_{R_0}, G_{R_1}, \ldots$
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 $\mathrm{alph}(R_i) = \mathrm{alph}(R_{i-1}) \cup \{\mathrm{alph}(w_i)\}$;
    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 $\Sigma = \mathrm{alph}(R_i)$;
      let $I_i = \mathbf{Consistent}(R_i)$;
      output $G_{R_i} = I_i(\mathcal{G}_{0,\Sigma})$;
    let $i := i + 1$;

Fig. 3. The identification algorithm *IA*.

all $i \geqslant 0$, $G_{R_r} = G_{R_{r+i}}$. Suppose that $L(G_*) \neq L(G_{R_r})$, then by Lemma 15, there exists a string $w \in L(G_*) - L(G_{R_r})$ such that w is not yet provided as a positive example. This implies that *IA* produces a conjecture distinct from G_{R_r}, a contradiction. □

Thus, we have the following.

Theorem 17. *The class of very simple grammars is identifiable in the limit from positive data.*

4.3. Time complexity analysis

We analyse the time complexity of *IA* in two respects: time for updating a conjecture and a bound on the number of implicit errors of prediction.

We analyse the time complexity of *IA* as follows:

(1) *Time for updating a conjecture*. Let $N = \sum_{w_j \in R} len(w_j)$ and ℓ be the maximum length of positive data in R. The time for updating a conjecture is obviously dominated by the time for the procedure **Consistent**(R).

In performing **Consistent**(R), determining f_n requires at most $O(N)$ time. It takes at most $O(N)$ time to construct $\mathrm{Lg}(R)$. Solving $\mathrm{Lg}(R)$ requires at most $O(|\Sigma|^3)$ time, because it is reduced to the computation of an inverse matrix with at most $|\Sigma|$-dimension. From (C1) of (Step 1) for the computation of f_p in Section 3.5, the value n_a is bounded by ℓ, the number of all solution vectors of $\mathrm{Lg}(R)$ is bounded

by $\ell^{|\Sigma|}$. Hence, **while** loops are repeatedly performed at most $\ell^{|\Sigma|}$ times. Each **while** loop requires at most $O(N)$ time.

Thus, time for updating a conjecture is bounded by

$$O(|\Sigma|^3) + O(\ell^{|\Sigma|}) \times O(N) \leqslant O(\mathrm{Max}\{N^{|\Sigma|+1}, |\Sigma|^3\}).$$

Note 3. The time complexity result above gives only an upper bound and, as mentioned in Note 2, an actual implementation could involve more efficient techniques.

(2) *A bound on the number of implicit errors of prediction*. Given a target very simple grammar G_*, let R be the current sample set of positive examples of $L(G_*)$ over Σ. Further, let $\mathrm{Vec}(R) = \{\mathrm{vec}(w) | w \in R\}$, where $\mathrm{vec}(w) = (\#_{a_1}(w), \ldots, \#_{a_m}(w))$, and $\Sigma' = \{a_1, \ldots, a_m\}$ is an ordered set $\mathrm{alph}(R) - \Sigma_f(R)$ (see Section 3.5).

The number of implicit errors of prediction *IA* makes is analysed as follows.

Lemma 18. *The number of implicit errors of prediction IA makes is bounded by* $O(|\Sigma|)$.

Proof. Let $I = (f_n, f_p)$ be the output of Consistent(R) such that $I(\mathscr{G}_{0,\Sigma})$ is the current conjectured grammar from *IA*. Each time the conjecture $I(\mathscr{G}_{0,\Sigma})(= G_R)$ is refuted by the next new example w, we examine what changes happen on the current conjecture, that is, on $I = (f_n, f_p)$.

Each time $I(\mathscr{G}_{0,\Sigma})$ fails to generate w, *IA* produces a new conjecture $I'(\mathscr{G}_{0,\Sigma})$ which can generate w. Let $I' = (f'_n, f'_p)$ and $R' = R \cup \{w\}$, where $I' = \mathrm{Consistent}(R')$. We analyse the following cases:

Case (*i*). f'_n is distinct from f_n: From the definition of **Consistent**(R) in *IA*, this change entails either at least one new nonterminal identification in $\mathscr{G}_{0,\Sigma}$ or $\mathrm{alph}(R) \subset \mathrm{alph}(R')$, because of the monotonic incremental nature of nonterminal identification. Therefore, this case may happen in total at most $2|\Sigma|$ times.

Case (*ii*). f'_p is distinct from f_p: Let $\mathrm{Vec}(R)$ and $\mathrm{Vec}(R')$ be the sets of vectors for R and R', respectively. We note that since $R \subset R'$, $\mathrm{Vec}(R) \subseteq \mathrm{Vec}(R')$. There are two possible cases: (1) a vector $\mathrm{vec}(w)$ is a linear combination of $\mathrm{Vec}(R)$ and (2) $\mathrm{Vec}(R')(= \mathrm{Vec}(R) \cup \{\mathrm{vec}(w)\})$ is linearly independent.

In case (1), $\mathrm{Sol}(\mathrm{Lg}(R))$ remains unchanged. If $\mathrm{alph}(R) \subset \mathrm{alph}(R')$, then Case (i) must occur. Hence, suppose that $\mathrm{alph}(R) = \mathrm{alph}(R')$, and therefore, it must hold that for each $a \in \mathrm{alph}(R')$, $len(f_p(x_a)) = len(f'_p(x_a))$. However, since f'_p is distinct from f_p, $f_p(x_a) \neq f'_p(x_a)$ for some $a \in \mathrm{alph}(R')$. Suppose that $f_n = f'_n$. Then, since the simulation of R by the set of candidate rules for I' must succeed using only the set of candidate rules for I, at that moment, for each $a \in \mathrm{alph}(R')$, the value of $f'_p(x_a)$ is completely determined through the simulation of $R(= R' - \{w\})$. Hence, without additional nonterminal identifications, we cannot have a situation where $f_p(x_a) \neq f'_p(x_a)$ for some $a \in \mathrm{alph}(R')$ by simulating the derivation for w, because $f_n = f'_n$ and $\mathrm{alph}(R) = \mathrm{alph}(R')$. Thus, as far as $f_p \neq f'_p$, we conclude that $f_n \neq f'_n$, that is, Case (i) must occur.

In case (2), for each R the number of vectors in $\mathrm{Vec}(R)$ linearly independent is bounded by the number of parameters to be solved. Hence, this can happen in total at most $|\Sigma|$ times.

Each time the conjecture fails, and hence, I' is distinct from I, at least one of the above cases must occur. Therefore, the number of implicit errors of prediction is bounded by $O(|\Sigma|)$. □

One problem to be discussed here is how we should define the size of a grammar $|G|$ for a very simple grammar G with the terminal alphabet Σ. In a usual setting, the size of a grammar may often be defined as the number of rules. However, for a fixed Σ, the number of rules of any very simple grammar is equal to the alphabet size, which implies that every very simple grammar with Σ has the same size of $|\Sigma|$.

We may define the size of a very simple grammar $G=(V_N,\Sigma,P,S)$ as the length of its description for P. That is, let us define $size(G)$ by $\sum_{A\to a\alpha\in P}(len(\alpha)+3)$. Then, it is obvious that $|\Sigma|\leqslant size(G)$.

Summing up the above observations together with Lemma 18, we have the following theorem.

Theorem 19. *The algorithm IA may be implemented to run in time polynomial in* $O(m)$ *for updating a conjecture, and the number of implicit errors of prediction is bounded by* $O(n)$, *where* $n=size(G_*)$, *and* $m=\mathrm{Max}\{N^{|\Sigma|+1},|\Sigma|^3\}$, $N=\sum_{w_j\in R} len(w_j)$, *and* R *is the set of positive data provided.*

In a similar manner discussed in [4,17], under the condition that only positive counterexamples are supplied in the identification process, the polynomial-time identifiability in the limit from positive data implies the polynomial-time identifiability via equivalence queries. That is, we have the following corollary.

Corollary 20. *The class of very simple grammars is polynomial-time identifiable via only equivalence queries, provided that only positive counterexamples are supplied in the identification process.*

5. An example run

Example 5. Consider $G_*=(\{S,A,B,C\},\{a,b,c,d,e,k\},P,S)$ as the target grammar, where $P=\{S\to aAS, S\to cBC, A\to b, B\to dBC, B\to e, C\to k\}$. After all initial procedures, *IA* is ready to read the input example.

(1) Let $w_1=cd^2ek^3$ be the first example string from $L(G_*)$. $R_1=\{w_1\}$ and $\mathrm{alph}(R_1)=\{c,d,e,k\}$. Then, since w_1 is trivially not in $L(G_{R_0})$, the algorithm *IA* constructs the augmented grammar schema $\mathscr{G}_{0,\Sigma}$ whose set of rule forms is $\{X_c\to cx_c, X_d\to dx_d, X_e\to ex_e, X_k\to kx_k\}$. Let $V_N=\{X_c,X_d,X_e,X_k,S\}$. Since a symbol k is final, the k-handle rule is $X_k\to k$ and $f_p(x_k)=\lambda(n_k=-1)$. Further, $f_n(X_c)=S$ is immediately obtained together with $f_p(x_c)\neq\lambda(n_c\geqslant 0)$. The length equation for w_1 is

$$n_c+2n_d+n_e+3n_k=-1 \quad \cdots \quad (\ell.w_1), \text{ and hence,}$$
$$n_c+2n_d+n_e=2 \quad \cdots (\ell.w_1)'.$$

Thus, we have that $\Sigma_s(R_1)=\{c\}, \Sigma_f(R_1)=\{k\}$, and $\mathrm{Sol}(\mathrm{Lg}((R_1)))=\{n_c+2n_d+n_e=2\}$, where $\Sigma'=\{c,d,e\}$. From the structure graph of R_1, it follows that $f_n(X_c)=S,\ f_n(X)=X(\forall X\in\{X_d,X_e,X_k\})$. (See (a) of Fig. 4 for the structure graph T_{R_1}.)

Among possible solution vectors of $\{n_c+2n_d+n_e=2\}$, suppose we choose a solution $\mathscr{X}_1=(4,-1,0)(=(n_c,n_d,n_e))$. That is, the set of candidate rules $\mathrm{CR}(\mathscr{X}_1)$ is

$$\{S\to cZ_{c,1}Z_{c,2}Z_{c,3}Z_{c,4}Z_{c,5},\ X_d\to d,\ X_e\to eZ_{e,1}\}.$$

By simulating the derivation for w_1 via these rules, we have that: $Z_{c,1}=X_d,\ Z_{c,2}=X_d,\ Z_{c,3}=X_e,\ Z_{c,4}=Z_{c,5}=Z_{e,1}=X_k$. We see that $\mathrm{CR}(\mathscr{X}_1)$ is good and, as a result, a ground interpretation $I_1=(f_n,f_p)$ admissible to R_1 is obtained, where

$$\begin{array}{ll} f_n(X_c)=S, & f_n(X)=X\ (X\text{: otherwise}),\\ f_p(x_c)=X_d^2X_eX_k^2, & f_p(x_d)=\lambda,\\ f_p(x_e)=X_k, & f_p(x_k)=\lambda. \end{array}$$

The first conjectured grammar $G_{R_1}=I_1(\mathscr{G}_{0,\Sigma})$ is $(\{S,X_d,X_e,X_k\},\{c,d,e,k\},P_1,S)$, where P_1 is

$$\begin{array}{ll} S\to cX_d^2X_eX_k^2, & X_d\to d\\ X_e\to eX_k, & X_k\to k. \end{array}$$

(2) Suppose that $w_2=cd^4ek^5$ is given as the second example string. Then, $R_2=\{w_1,w_2\}$ and $\mathrm{alph}(R_2)=\mathrm{alph}(R_1)=\{c,d,e,k\}$. Since w_2 is not in $L(G_{R_1})$, $\mathscr{G}_{0,\Sigma}$ is to be augmented but, in fact, unchanged.

First, by constructing the structure graph T_{R_2} pictured in (b) of Fig. 4, we observe that $f_n(X_e)=X_d$.

Then, *IA* computes a length equation for w_2 and constructs $\mathrm{Lg}(R_2)=\{(\ell.w_1),\ (\ell.w_2)\}$, where

$$\begin{array}{ll} n_c+4n_d+n_e+5n_k=-1 & \cdots\ (\ell.w_2),\ \text{and hence,}\\ n_c+4n_d+n_e=4 & \cdots\ (\ell.w_2)'. \end{array}$$

To solve $\mathrm{Lg}(R_2')=\{(\ell.w_1)',(\ell.w_2)'\}$, we construct an associated matrix M_{R_2} and a matrix equation:

$$M_{R_2}=\begin{pmatrix}1&2&1\\1&4&1\end{pmatrix}\quad M_{R_2}\mathscr{X}_2^{\mathrm{T}}=(2,4)^{\mathrm{T}}\\ \mathscr{X}_2=(n_c,n_d,n_e).$$

The matrix computation is done as follows:

$$\left(\begin{array}{ccc|c}1&2&1&2\\1&4&1&4\end{array}\right)\Rightarrow^*\left(\begin{array}{ccc|c}1&0&1&0\\0&1&0&1\end{array}\right).$$

Thus, $\mathrm{Sol}(\mathrm{Lg}(R_2'))=\{n_d=1\}\cup\{n_c+n_e=0\}$. From $\mathrm{Sol}(\mathrm{Lg}(R_2'))$, for example, we may choose a solution vector $\mathscr{X}_2=(0,1,0)(=(n_c,n_d,n_e))$. Hence, the set of candidate rules

$$\mathrm{CR}(\mathscr{X}_2)=\{S\to cZ_{c,1},\ X_d\to dZ_{d,1}Z_{d,2},\ X_d\to eZ_{e,1}\}$$

is obtained. Simulating R_2 via $\mathrm{CR}(\mathscr{X}_2)$ succeeds without any additional nonterminal identification, and produces the identification results that $Z_{c,1}=X_d$, $Z_{d,1}=X_d$, $Z_{d,2}=X_k$, $Z_{e,1}=X_k$. From this, a ground interpretation $I_2=(f_n,f_p)$ is obtained, where

$$\begin{array}{ll} f_n(X_c)=S, & f_n(X_e)=X_d, \\ f_n(X)=X & (\forall X\in V_N-\{X_c,X_e\}), \\ f_p(x_c)=X_d, & f_p(x_d)=X_dX_k, \\ f_p(x_e)=X_k, & f_p(x_k)=\lambda. \end{array}$$

Thus, the second conjectured grammar $G_{R_2}=I_2(\mathscr{G}_{0,\Sigma})$ is the following: $(\{S,X_d,X_k\}, \{c,d,e,k\},P_2,S)$, where P_2 is

$$\begin{array}{ll} S\to cX_d, & X_d\to dX_dX_k, \\ X_d\to eX_k, & X_k\to k. \end{array}$$

(3) Suppose that $w_3=abababcek$ is given as the third example string. Then, $R_3=\{w_1,w_2,w_3\}$ and $\mathrm{alph}(R_3)=\mathrm{alph}(R_2)\cup\{a,b\}$. Since w_3 is not in $L(G_{R_2})$, in order to augment $\mathscr{G}_{0,\Sigma}$, two rule forms $X_a\to ax_a$ and $X_b\to bx_b$ are added to the set of rule forms of $\mathscr{G}_{0,\Sigma}$. Note that $V_N=\{X_a,X_b,X_d,X_k,S\}$, $\Sigma_s(R_3)=\{a,c\}$. The structure graph T_{R_3} is shown in (c) of Fig. 4. Hence, we have that $f_n(X_a)=S$ and $f_p(x_a)\neq\lambda(n_a\geqslant 0)$.

IA computes a length equation for w_3 and constructs $\mathrm{Lg}(R_3')=\{(\ell.w_1)',(\ell.w_2)',(\ell.w_3)'\}$, where

$$\begin{aligned} 3n_a+3n_b+n_c+n_e+n_k &= -1 \cdots (\ell.w_3), \text{ and hence,} \\ 3n_a+3n_b+n_c+n_e &= \ \ 0 \cdots (\ell.w_3)'. \end{aligned}$$

Then, an associated matrix M_{R_3} and its matrix equation are:

$$M_{R_3}=\begin{pmatrix} 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 4 & 1 \\ 3 & 3 & 1 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} M_{R_3}\mathscr{X}_3^{\mathrm{T}}=(2,4,0)^{\mathrm{T}} \\ \mathscr{X}_3=(n_a,n_b,n_c,n_d,n_e). \end{array}$$

The matrix computation is done as follows:

$$\left(\begin{array}{ccccc|c} 0 & 0 & 1 & 2 & 1 & 2 \\ 0 & 0 & 1 & 4 & 1 & 4 \\ 3 & 3 & 1 & 0 & 1 & 0 \end{array}\right) \Rightarrow^* \left(\begin{array}{ccccc|c} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right).$$

Thus, $\mathrm{Sol}(\mathrm{Lg}(R_3'))=\{n_d=1\}\cup\{n_a+n_b=0,n_c+n_e=0\}$ is obtained from which, for example, we may choose a solution vector $\mathscr{X}_3=(0,0,0,1,0)(=(n_a,n_b,n_c,n_d,n_e))$. Hence, the set of candidate rules $\mathrm{CR}(\mathscr{X}_3)$ is

$$\begin{array}{lll} S\to aZ_{a,1}, & X_b\to bZ_{b,1}, & S\to cZ_{c,1}, \\ X_d\to dZ_{d,1}Z_{d,2}, & X_d\to eZ_{e,1}. & \end{array}$$

Then, without any additional nonterminal identification, simulating R_3 via $\mathrm{CR}(\mathscr{X}_3)$ leads to the identification results that $Z_{a,1}=X_b$, $Z_{b,1}=S$, $Z_{c,1}=X_d$, $Z_{d,1}=X_d$, $Z_{d,2}=X_k$,

$Z_{e,1} = X_k$. From this, a ground interpretation $I_3 = (f_n, f_p)$ is obtained, where

$$\begin{array}{ll} f_n(X_a) = f_n(X_c) = S, & f_n(X_e) = X_d, \\ f_n(X_b) = X_b, & f_n(X_k) = X_k, \\ f_p(x_a) = X_b, & f_p(x_b) = S, \\ f_p(x_c) = X_d, & f_p(x_d) = X_d X_k, \\ f_p(x_e) = X_k, & f_p(x_k) = \lambda. \end{array}$$

Thus, the third conjectured grammar $G_{R_3} = I_3(\mathscr{G}_{0,\Sigma})$ is the following one: $(\{S, X_b, X_d, X_k\}, \{a, b, c, d, e, k\}, P_3, S)$, where P_3 consists of

$$\begin{array}{lll} S \to aX_b, & X_b \to bS, & S \to cX_d, \\ X_d \to dX_dX_k, & X_d \to eX_k, & X_k \to k. \end{array}$$

The conjecture G_{R_3} is equivalent (but not isomorphic) to G_*. Hence, G_{R_3} is always output as a conjecture for all input data afterwards.

6. Discussion

6.1. Related works

Mäkinen [13] discusses the problem of learning Szilard languages of linear grammars and gives a linear update-time algorithm for solving the problem, however he gives no consideration on the polynomial-time identifiability of the class in the sense discussed in this paper. 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]. (In fact, the class of very simple languages coincides with the left Szilard languages of *context-free* grammars [2].) Further, the class of very simple languages is incomparable to the class of zero-reversible languages. (See Fig. 1.) Thus, the main result in the present paper is in contrast with the fact that the class of regular languages is not polynomial-time identifiable in the limit using *DFA*s in the sense of Pitt [17].

Wakatsuki and Tomita [21] study the inclusion problem for the class of very simple grammars and show the problem is decidable by giving an efficient algorithm.

6.2. Conclusions

By adopting a slight modification of Pitt's definition, 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 new definition. One of the interesting features of the learning algorithm developed in this paper is the property of "reliability" in the sense that the algorithm *IA* has the ability of "self-diagnosing" which can detect input data inconsistent with any very simple grammar (see Corollary 11). Hence, besides its ability to identify every very simple language, the algorithm *IA* is additionally able to *refute* (in the sense of Mukouchi

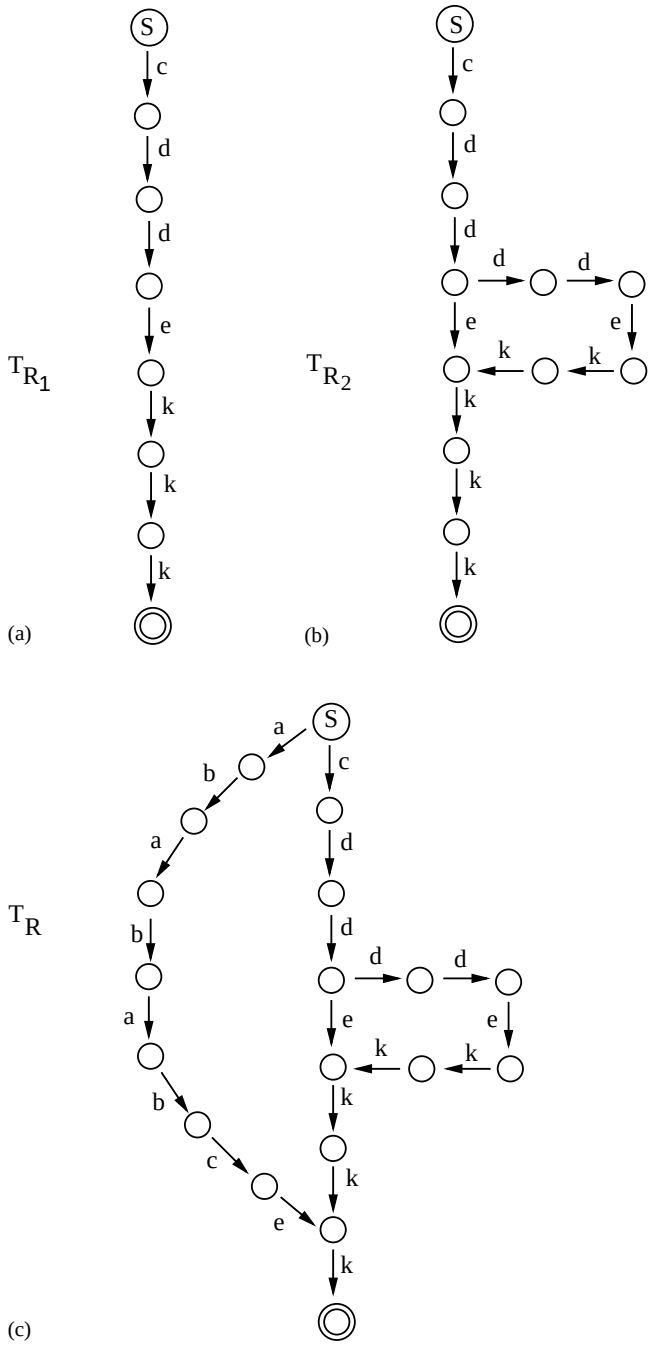
T_{R_1}
S
c
d
d
e
k
k
k
(a)
T_{R_2}
S
c
d
d
d
d
e
e
k
k
k
k
k
(b)
T_R
S
a
b
a
b
a
b
c
e
c
d
d
d
d
e
e
k
k
k
k
k
k
(c)

Fig. 4.

and Arikawa [16]) every positive presentation that does *not* describe a very simple language.

As we have shown in Section 3, the class of very simple grammars is moderately powerful in language generative capability, and it bears some of the essential properties of *CFG*s except for the nondeterministic features.

One of the results by Shinohara [19] shows that the class of context-sensitive 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. In fact, for a fixed alphabet Σ, one may argue a simple and enumerative algorithm for identifying the class in polynomial time, because the number of rules in a grammar G is bounded by $|\Sigma|$ and hence there are no more than $O(n^{|\Sigma|})$ feasible hypotheses of what the grammar G could be, where n is the size of G. Compared to such a naive enumerative algorithm, our algorithm has some advantages. First, from the viewpoint of an upper-bound of the time complexity, the latter is only *potentially* exponential in $|\Sigma|$ in the sense that we do not know at present whether or not the worst case really occurs, while the former must require exponential time if the target grammar is the last one up to equivalence in the enumeration process due to the "size" order of very simple grammars. Second, the number of updating conjectures is crucially important for a learning algorithm, and our algorithm requires at most $O(|\Sigma|)$ times, while the other needs at the worst case $O(n^{|\Sigma|})$ times. (Thus, these two algorithms make a great difference as $|\Sigma|$ grows up.)

For future study, it seems useful to pursue applications of an inference algorithm for very simple grammars to other domains of research interests. In fact, making use of the fact that the left Szilard language of any *CFG* is very simple, Mäkinen [14] discusses a theoretical application of the inference algorithm for very simple grammars to the inference problem for *CFG*s from the structural data in which a Szilard word together with its corresponding terminal string is given as an example. In this problem setting, he shows the structural inference problem for *CFG*s is efficiently solvable. In relation to this topic, it would be a very interesting open problem to know whether or not there is an algorithm for identifying the class of very simple grammars with Σ in polynomial time in n, N (total length of data provided) *and* $|\Sigma|$.

Acknowledgements

The author would like to express his gratitude to E. Tomita and M. Wakatsuki who not only worked through an earlier draft of the paper but gave invaluable comments and suggestions. In particular the author is very much grateful to M. Kanazawa and R. Yoshinaka for their crucial comments which pointed out flaws in argument concerning the characteristic samples in the original version of this paper. Also, he would like to thank E. Mäkinen for many useful reference papers. Last but not least, the author is deeply indebted to referees for their many useful comments.

This work is supported in part by Grants-in-Aid for Scientific Research Nos. 03245104 and 04229105 from the Ministry of Education, Science, Sports and Culture, Japan.

References

[1] D. Angluin, Inductive inference of formal languages from positive data, Inform. and Control 45 (1980) 117–135.
[2] D. Angluin, Inference of reversible languages, J. ACM 29 (1982) 741–765.
[3] D. Angluin, Learning regular sets from queries and counterexamples, Inform. and Comput. 75 (1987) 87–106.
[4] D. Angluin, Negative results for equivalence queries, Mach. Learning 5 (1990) 121–150.
[5] P. Butzbach, Une famille de congruences de thue pour lesquelles le probleme de l'equivalence est decidable. application a l'equivalence des grammaires separees, in: M. Nivat (Ed.), Automata, Languages and Programming, North-Holland/American Elsevier, Amsterdam, 1973, pp. 3–12.
[6] P. Garcia, E. Vidal, Inference of k-testable languages in the strict sense and application to syntactic pattern recognition, IEEE Trans. Pattern Anal. Mach. Intell. 12 (9) (1990) 920–925.
[7] E.M. Gold, Language identification in the limit, Inform. and Control 10 (1967) 447–474.
[8] M. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, MA, 1978.
[9] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, Reading, MA, 1979.
[10] A. Korenjak, J.E. Hopcroft, Simple deterministic languages, Proc. 7th Annu. IEEE Conf. on Switching and Automata Theory, 1966, pp. 36–46.
[11] P. Laird, E. Gamble, Analytical learning and term-rewriting systems, Report RIA-90-06-17-7, Ames Research Center, NASA, 1990.
[12] E. Mäkinen, On context-free derivations, Ser A 198, Act Universitatis Tamperensis, 1985.
[13] E. Mäkinen, The grammatical inference problem for the szilard languages of linear grammars, Inform. Process. Lett. 36 (1990) 203–206.
[14] E. Mäkinen, Remarks on the structural grammatical inference problem for context-free grammars, Inform. Process. Lett. 44 (1992) 125–127.
[15] E. Moriya, The associate language and the derivation complexity of formal grammars, Inform. and Control 22 (1973) 139–162.
[16] Y. Mukouchi, S. Arikawa, Towards a mathematical theory of machine discovery from facts, Theoret. Comput. Sci. 137 (1995) 53–84.
[17] L. Pitt, Inductive inference, DFAs, and computational complexity, Proc. 2nd Workshop on Analogical and Inductive Inference, Lecture Notes in Artificial Intelligence, Vol. 397, Springer, Berlin, 1989, pp. 18–44.
[18] M. Sato, K. Umayahara, Inductive inferability for formal languages from positive data, IEICE Trans. Inform. Systems E 75-D (4) (1992) 84–92.
[19] T. Shinohara, Rich classes inferable from positive data: length bounded elementary formal systems, Inform. Comput. 108 (1994) 175–186.
[20] N. Tanida, T. Yokomori, Polynomial-time identification of strictly regular languages in the limit, IEICE Trans. Inform. Systems E 75-D (1) (1992) 125–132.
[21] M. Wakatsuki, E. Tomita, A fast algorithm for checking the inclusion for very simple deterministic pushdown automata, IEICE Trans. Inform. Systems E 76-D (10) (1993) 1224–1233.
[22] K. Wright, Identification of unions of languages drawn from an identifiable class, Proc. 2nd Workshop on Computational Learning Theory, 1989, pp. 328–333, T. Motoki, T. Shinohara, K. Wright, The correct definition of finite elasticity: corrigendum to identification of unions, Proc. 4th Workshop on Computational Learning Theory, 1991, pp. 375.
[23] T. Yokomori, On polynomial-time learnability in the limit of strictly deterministic automata, Machine Learning 19 (2) (1995) 153–179.
[24] T. Yokomori, S. Kobayashi, Learning local languages and their application to DNA sequence analysis, IEEE Trans. Pattern Anal. Mach. Intell. 20 (10) (1998) 1067–1079.