TR-317

A Constructive Method for Grammatical Inference of Linear Languages based on Control Sets

by

Y. Takada

November, 1987

A Constructive Method for Grammatical Inference of Linear Languages Based on Control Sets

by

YUJI TAKADA

Research Associate, Fundamental Informatics Section,
International Institute for Advanced Study of
Social Information Science (IIAS-SIS),
FUJITSU LIMITED

Abstract

We provide a new constructive method for grammatical inference of linear languages. We first show that there exists a fixed linear grammar by which any linear language is generated with a regular control set. From this, the problem of identifying an unknown linear language is reduced to the problem of identifying a regular control set for the fixed linear grammar. With structural informations of a linear language L, given an oracle for L and some auxiliary information about the linear grammar which generates L, the algorithm we describe makes a polynomial number of queries and outputs a description of a finite automaton accepting the regular control set with which L is generated by the fixed linear grammar.

1. Introduction

A language is identified by specifying a grammar which generates it. The problem of specifying a grammar from finite number of strings in a language is known as grammatical inference. It is convenient to classify the methods for grammatical inference into two categories, constructive methods and enumerative methods. Constructive methods systematically use sample strings to construct the grammar. On the other hand, enumerative methods use them to select the grammar which generates the language among enumerated candidates. Constructive methods have one important advantage over enumerative methods, that is, they frequently require only modest amounts of computation. But most of known constructive methods can solve only grammatical inference for the class of regular languages. There is a few studies of constructive methods for more general classes.

Biermann (1971) developed a computer program which constructs grammars for a linear language from finite subsets of the language. Tanatsugu (1987) constructed a practical inference procedure for linear languages. Both of them find, using some given parameters for bounding computation, the self-embedding substrings in the linear language. But procedures for finding self-embedding substrings are not so efficient.

In this paper, we provide a new constructive method for the grammatical inference of linear languages. We first show that there exists a fixed linear grammar by which any linear language is generated with a regular control set (Theorem 3.2). In this case, self-embedding variables of a linear grammar correspond to the states on loops in the transition diagram of the finite state automaton which accepts the regular control set. From this, the problem of identifying an unknown linear language L is reduced to the problem of identifying a regular control set for the fixed linear grammar. It is well known (Angluin,1981; Biermann,1972) that any regular set is identified by some effective algorithms.

With structural informations of L, given an oracle to answer membership questions about L and some auxiliary information about a linear grammar which generates L, our algorithm makes a polynomial number of queries and outputs a description of a finite automaton accepting the regular control set with which L is generated by the fixed linear grammar (Algorithm LID in section 4.2).

Although the algorithm LID can only identify the class of linear languages, this method may be extended to cover significantly more general classes of context-free languages.

2. Preliminaries

Let Σ be a finite alphabet containing k symbols for some $k \geq 2$. Let Σ^* denote the set of all strings over Σ including the null string λ. Let u and v be two strings. Then, uv denotes the concatenation of u and v, and $|u|$ denotes the length of u. A string u is said to be a prefix of a string w if and only if there exists a string v such that w=uv. If W is a set of strings, we denote by Pr(W) the set of all prefixes of strings in W. We note that if W is not empty then Pr(W) contains λ.

Let X be a set. We denote by 2^X the power set of X, i.e., the set of all subsets of X. For two sets X and Y, $X \oplus Y$ denotes the symmetric difference of X and Y, i.e., $X \oplus Y = (X \cup Y) - (X \cap Y)$.

A *deterministic finite automaton* (abbreviated as DFA) M over Σ is a 5-tuple $<K, \Sigma, \delta, q_0, F>$, where K is a finite nonempty set of states, Σ is a finite input alphabet, δ is a transition function from $K \times \Sigma$ to K, $q_0 \in K$ is the initial state, and $F \subset K$ is the set of final states.

We extend δ to a function from $K \times \Sigma^*$ to K such that for all $q \in K$, $\delta(q, \lambda) = q$ and $\delta(q, uv) = \delta(\delta(q, u), v)$ for all strings u and v in Σ^*. The set accepted by M, denoted T(M), is the set of strings u such that $\delta(q_0, u)$ is in F, i.e., $T(M) = \{u \in \Sigma^* \mid \delta(q_0, u) \in F\}$.

A *nondeterministic finite automaton* (abbreviated as NFA) M' is a 5-tuple $<K, \Sigma, \delta', q_0, F>$, where K, Σ, q_0, and F have the same meaning as for a DFA, but δ'

is a function from $K\times\Sigma$ to 2^K. The transition function δ' can be extended to a function from $2^K\times\Sigma^*$ to 2^K such that for all strings u and v in Σ^*, $\delta'(q,\lambda)=\{q\}$ and $\delta'(q,uv)=\{p \mid \text{for some state } r \text{ in } \delta'(q,u),\ p \text{ is in } \delta'(r,v)\}$ for all $q\in K$, and $\delta'(Q,u)=\cup_{q\in Q}\delta'(q,u)$ for $Q\subset 2^K$. T(M') is the set $\{u \mid \delta'(q_0,u)\cap F\neq\emptyset\}$. A DFA is an NFA by definition.

It is well known that if a set R is accepted by an NFA then there exists a DFA that accepts R.

A subset R of Σ^* is called *regular* if and only if R is accepted by a DFA. If R is regular, then there is a minimum state DFA, unique up to isomorphism, which accepts R. This is called the *canonical* finite automaton for R.

Let $M=\langle K,\Sigma,\delta,q_0,F\rangle$ be an NFA. A state q is called *live* if and only if there exist strings u and v in Σ^* such that $uv\in T(M)$ and $q\in\delta(q_0,u)$. A state which is not live is called *dead*. It is easy to verify that the canonical finite automaton has at most one dead state.

A context-free grammar G over Σ is a 4-tuple $\langle N,\Sigma,\Pi,S\rangle$. N is a finite nonempty set and called *variables*. We assume that N and Σ are disjoint, and denote $N\cup\Sigma$ by V. Π is a finite nonempty set of *productions*; each production is of the form $A\to u$, where A is a variable and u is a string in V^*. We distinguish each production in Π by its label π_i. S is a special variable called the *start symbol*.

A *linear grammar* G is a context-free grammar such that each production in Π is of the form

$$A\to uBv \quad\text{or}\quad A\to u$$

where $A,B\in N$ and $u,v\in\Sigma^*$.

Let $G=\langle N,\Sigma,\Pi,S\rangle$ be a linear grammar. We define the relation $\Rightarrow_G^{\pi_i}$ between strings in V^*. For $x,y\in V^*$, $x \Rightarrow_G^{\pi_i} y$ if and only if $x=vAw$, $y=vuw$ and $\pi_i{:}A\to u$ is a production of Π for some $v,w\in\Sigma^*$. We say that the production $\pi_i{:}A\to u$ is applied to the string x to obtain y.

Let $x_0, x_1, \ldots, x_k$ be strings in V^*, where $k\geq 1$. If

$$x_0 \Rightarrow_G^{\pi_1} x_1,\ x_1 \Rightarrow_G^{\pi_2} x_2,\ \ldots,\ x_{k-1} \Rightarrow_G^{\pi_k} x_k,$$

then we denote $x_0 \Rightarrow_G^{\alpha} x_k$, where $\alpha=\pi_1\pi_2...\pi_k$, which is called a *derivation* from x_0 to x_k with an *associate word* α in G.

Let L(A,G) denote the set $\{w\in\Sigma^* \mid A \Rightarrow_G^{\alpha} w, \alpha\in\Pi^*\}$. The language generated by G, denoted L(G), is the set L(S,G), i.e.,

$$L(G) = \{w\in\Sigma^* \mid S \Rightarrow_G^{\alpha} w, \alpha\in\Pi^*\}.$$

A language L is said to be *linear* if and only if there exists a linear grammar G such that L=L(G) holds.

A linear grammar $G=\langle N,\Sigma,\Pi,S\rangle$ is called in *linear normal form* if and only if each production is of the form

$$S\to\lambda,\ A\to a,\ A\to aB \text{ or } A\to Ba,$$

where $A,B\in N$, $a\in\Sigma$.

Proposition 2.1.

Any linear language is generated by a linear normal form grammar G.

Proof. Without loss of generality, we may assume that any linear language is generated by a linear grammar $G'=\langle N',\Sigma,\Pi',S\rangle$ which has no production of the form $A\to A$ or $B\to\lambda$, where $A\in N'$, $B\in N'-\{S\}$.

We construct a linear normal form grammar $G=\langle N,\Sigma,\Pi,S\rangle$ from G' as follows; Each production of the form $S\to\lambda$, $A\to aB$, $A\to Ba$ or $A\to a$ in Π' is in Π. If a production of the form $A\to a_1a_2...a_k$ ($k\geq2$) is in Π', then we introduce new variables $C_1,C_2, ... , C_{k-1}$ into N and new productions $A\to a_1C_1$, $C_1\to a_2C_2$, ... , $C_{k-1}\to a_k$ into Π. If a production of the form $A\to a_1a_2...a_iBb_1b_2...b_j$ is in Π', where $i\geq2$ and $j\geq0$, or $i\geq0$ and $j\geq2$, then we introduce new variables $C_1, C_2, ... , C_{i+j-1}$ into N and new productions $A\to a_1C_1$, $C_1\to a_2C_2$, ... , $C_{i-1}\to a_iC_i$, $C_i\to C_{i+1}b_1$, $C_{i+1}\to C_{i+2}b_2$, ... , $C_{i+j-1}\to Bb_j$ into Π. It is easily seen that $S \Rightarrow_{G'}^{\alpha'} w$ if and only if $S \Rightarrow_G^{\alpha} w$.

In what follows, we assume that G is in linear normal form, and also assume that for $w\in L(G)$, every variable A appears in some derivation $S \Rightarrow_G^{\alpha} w$, i.e., there exists a derivation $S \Rightarrow_G^{\beta} uAv \Rightarrow_G^{\gamma} w$, where $\alpha=\beta\gamma$.

3. Representation Theorem for Linear Languages

In this section, we show that there exists a fixed linear grammar G^0 by which any linear language L is generated with a regular control set C.

DEFINITION. A linear grammar $G^0=<\{S^0\},\Sigma,\Pi^0,S^0>$ is said to be *universal* if and only if Π^0 consists of the following productions,

$$\begin{aligned}\Pi^0 = \{\ & S^0\to a_1S^0,\ S^0\to a_2S^0,\ \dots,\ S^0\to a_nS^0,\\ & S^0\to S^0\,a_1,\ S^0\to S^0\,a_2,\ \dots,\ S^0\to S^0\,a_n,\\ & S^0\to a_1,\ S^0\to a_2,\ \dots,\ S^0\to a_n,\\ & S^0\to\lambda\ \},\end{aligned}$$

where $\Sigma=\{a_1,a_2,\dots,a_n\}$.

Note that for a given alphabet Σ, the universal linear grammar G^0 is uniquely determined.

DEFINITION. Let $G=<N,\Sigma,\Pi,S>$ be a linear grammar, and C be a subset of Π^*. Then

$$L_C(G) = \{w\in\Sigma^* \mid S \Rightarrow_G^{\alpha} w,\ \alpha\in C\}$$

is called *the language generated by G with control set C.*

Let $G=<N,\Sigma,\Pi,S>$ be a linear grammar, and $G^0=<\{S^0\},\Sigma,\Pi^0,S^0>$ be the universal linear grammar. We define a homomorphism h from Π^* to Π^{0*} such that

$$h(\pi)=\begin{cases}\pi^0_i \text{ where } \pi^0_i{:}S^0\to aS^0, \text{ if } \pi{:}A\to aB,\\ \pi^0_j \text{ where } \pi^0_j{:}S^0\to S^0a, \text{ if } \pi{:}A\to Ba,\\ \pi^0_k \text{ where } \pi^0{:}S^0\to a, \text{ if } \pi{:}A\to a,\\ \pi^0 \text{ where } \pi^0{:}S^0\to\lambda, \text{ if } \pi{:}S\to\lambda.\end{cases}$$

We construct the *corresponding* NFA $M=<K,\Pi^0,\delta,S,F>$ to G, where K, δ, F are defined as follows;

1. $K=N\cup\{q_F\}$ where $q_F \notin N$.

2.
$$\delta(S,\pi^0)=\begin{cases}\{q_F\} \text{ if } \pi:S\to\lambda,\ \pi^0:S^0\to\lambda, \text{ and } \pi\in h^{-1}(\pi^0),\\ \\ \emptyset \text{ if } \pi^0:S^0\to\lambda \text{ and } h^{-1}(\pi^0)=\emptyset.\end{cases}$$
$$\delta(A,\pi^0{}_j)=\begin{cases}\{B \mid \pi_i:A\to aB\in\Pi,\ \pi_i\in h^{-1}(\pi^0{}_j)\} \text{ if } \pi^0{}_j:S^0\to aS^0,\\ \\ \{B \mid \pi_i:A\to Ba\in\Pi,\ \pi_i\in h^{-1}(\pi^0{}_j)\} \text{ if } \pi^0{}_j:S^0\to S^0a.\end{cases}$$
$$\delta(A,\pi^0{}_l)=\begin{cases}\{q_F\} \text{ if } \pi_k:A\to a\in\Pi,\ \pi^0{}_l:S^0\to a \text{ and } \pi_k\in h^{-1}(\pi^0{}_l),\\ \\ \emptyset \text{ if } \pi^0{}_l:S^0\to a \text{ and } h^{-1}(\pi^0{}_l)=\emptyset.\end{cases}$$

3. $F=\{q_F\}$.

Now we have the following Lemma.

Lemma 3.1.

For any $w\in\Sigma^*$, $A\in N$ and $\alpha\in\Pi^*$, $A \Rightarrow_G{}^{\alpha} w$ if and only if $S^0 \Rightarrow_{G^0}{}^{\alpha^0} w$ and $\delta(A,\alpha^0)\ni q_F$, where $\alpha^0=h(\alpha)$.

Proof. The argument is an induction on the length of associate words α and α^0. If $A \Rightarrow_G{}^{\pi} a$, then $\pi:A\to a$ is in Π, so $S^0 \Rightarrow_{G^0}{}^{h(\pi)} a$. By definition of δ, $\delta(A,h(\pi))\ni q_F$. Conversely, if $S^0 \Rightarrow_{G^0}{}^{\pi^0} a$ and $\delta(A,\pi^0)\ni q_F$, then by definition of δ, there exists a $\pi\in h^{-1}(\pi^0):A\to a$ and $A \Rightarrow_G{}^{\pi} a$.

Inductively suppose that for any α in Π^* and any α^0 in Π^{0*} such that $|\alpha|\le n$ and $|\alpha^0|\le n$ the assertion holds. If $A \Rightarrow_G{}^{\pi} aB \Rightarrow_G{}^{\alpha} aw$, then $S^0 \Rightarrow_{G^0}{}^{h(\alpha)} w$ and $\delta(B,h(\alpha))\ni q_F$ by the inductive hypothesis. Since $\pi:A\to aB$ is in Π, we have $\delta(A,h(\pi))\ni B$ by definition of δ. Therefore, $S^0 \Rightarrow_{G^0}{}^{h(\pi)} aS^0 \Rightarrow_{G^0}{}^{h(\alpha)} aw$ and

$$\delta(\delta(A,h(\pi)),h(\alpha))=\delta(A,h(\pi)h(\alpha))=\delta(A,h(\pi\alpha))\ni q_F.$$

Conversely, suppose that $S^0 \Rightarrow_{G^0}^{\pi^0} aS^0 \Rightarrow_{G^0}^{\alpha^0} aw$ and $\delta(A,\pi^0) \ni B$ and $\delta(B,\alpha^0) \ni q_F$. Then $B \Rightarrow_G^{\alpha} w$ where $\alpha \in h^{-1}(\alpha^0)$ by the inductive hypothesis. By definition of δ, for $\pi^0: S^0 \to aS^0$ in Π^0 there exists $\pi \in h^{-1}(\pi^0): A \to aB$ and $A \Rightarrow_G^{\pi} aB$, therefore, $A \Rightarrow_G^{\pi\alpha} aw$.

The next theorem follows immediately from this Lemma by putting C=T(M).

Theorem 3.2.

For any linear language L, there exists a regular control set C such that for the universal linear grammar G^0, $L=L_C(G^0)$ holds.

We can also prove the converse case.

Theorem 3.3.

Let G^0 be the universal linear grammar over Σ and C be a regular control set for G^0. Then, $L=L_C(G^0)$ is a linear language.

Proof. Let $M=\langle K,\Pi^0,\delta,S,F\rangle$ be a DFA such that C=T(M) holds. We define a linear grammar $G=\langle K,\Sigma,\Pi,S\rangle$, to which M is corresponding, and a homomorphism h from Π^* to Π^{0*} as follows:

1. If $\delta(A,\pi^0_i)=B$ and π^0_i is $S^0 \to aS^0$, then $\pi_i: A \to aB$ is in Π and $h(\pi_i)=\pi^0_i$.
2. If $\delta(A,\pi^0_j)=B$ and π^0_j is $S^0 \to S^0a$, then $\pi_j: A \to Ba$ is in Π and $h(\pi_j)=\pi^0_j$.
3. If $\delta(A,\pi^0_k) \in F$ and π^0_k is $S^0 \to a$, then $\pi_k: A \to a$ is in Π and $h(\pi_k)=\pi^0_k$.
4. If $\delta(A,\pi^0) \in F$ and π^0 is $S^0 \to \lambda$, then $\pi: A \to \lambda$ is in Π and $h(\pi)=\pi^0$.

By Lemma 3.1, for any $w \in \Sigma^*$, $S^0 \Rightarrow_{G^0}^{\alpha^0} w$ and $\delta(S,\alpha^0) \in F$ if and only if $S \Rightarrow_G^{\alpha} w$, where $\alpha \in h^{-1}(\alpha^0)$.

Thus, we may reduce the problem of identifying a linear language L to the problem of identifying a regular control set C such that $L=L_C(G^0)$ holds for the universal linear grammar G^0. To identify C, we construct a DFA M over Π^0

corresponding to a linear grammar G which generates L. However, we have further difficulties in identifying a regular control set from examples of L because of the universal property of G^0. That is, for any linear grammar G which generates L there exists a regular control set C such that $L(G)=L_C(G^0)$ holds. Therefore, in general, it is unknown whether the set

$$C'=\{\alpha^0 \mid S^0 \Rightarrow_{G^0}{}^{\alpha^0} w,\ w\in L\} = \cup_{L(G)=L}\, h(\{\alpha \mid S \Rightarrow_G{}^{\alpha} w,\ w\in L\})$$

is regular or not because there exist infinite number of linear grammars which generate L. If the procedure for identifying C from given examples of L might consider all derivations of them in G^0, then it should construct infinite number of candidates of C. Hence, we give some sufficient condition for the procedure to identify some C and halt.

DEFINITION. A linear grammar $G=\langle N,\Sigma,\Pi,S\rangle$ is *canonical* if and only if it satisfies the following conditions:

1. There is no pair of productions $A\to aB$, $A\to aC$ or $A\to Ba$, $A\to Ca$, where $B\neq C$.
2. For any distinct variables $A,B\in N$, $L(A,G)\neq L(B,G)$.

Since the equivalence problem for the class of linear grammars is not decidable, the condition 2 in the definition of the canonical linear grammar is not effective.

We show that, given a linear language L, there exists a canonical grammar which generates L. But it is not unique in general, so we use the term "canonical" for a linear grammar in a different way from a canonical finite automaton.

Lemma 3.4.

Given a linear grammar G, one can get a linear grammar G' such that L(G')=L(G) and G' has no pair of productions $A\to aB$, $A\to aC$ or $A\to Ba$, $A\to Ca$, where $B\neq C$.

Proof. Given a linear grammar G, by Theorem 3.2, we can get an NFA M corresponding to G. Let M' be the DFA such that T(M)=T(M') holds. By Theorem 3.3, we can get a linear grammar G' from M'. Then, clearly, $L(G)=L_C(G^0)=L(G')$ holds. The construction of G' ensures that G' has no pair of productions $A \to aB$, $A \to aC$ or $A \to Ba$, $A \to Ca$.

Lemma 3.5.

For any linear language L, there exists a linear grammar G such that L=L(G) and for any distinct variables A and B of G, $L(A,G) \neq L(B,G)$.

Proof. Let $H=<N,\Sigma,\Pi,S>$ be a linear grammar such that L=L(H) holds. For any variables A, $B \in N$, if L(A,H)=L(B,H), then we remove B from N and replace all occurrences of B in each production of Π by A. Let N' be the new set of variables and Π' be the new set of productions. Then $H'=<N',\Sigma,\Pi',S>$ is a linear grammar. Clearly, $S \Rightarrow_H{}^{\alpha} w$ if and only if $S \Rightarrow_{H'}{}^{\alpha'} w$. Therefore, L(H)=L(H'). By repeating this procedure the proof is completed.

From Lemmas 3.4 and 3.5, we get the following result.

Proposition 3.6.

For any linear language L, there exists a canonical linear grammar G such that L=L(G) holds.

Proposition 3.7.

Let G is a linear grammar and M is a finite automaton corresponding to G. Then, if G is canonical, M is canonical.

Proof. Since G is canonical, M is a DFA. Since for any distinct variables A and B of G, there exists a string w in Σ^* such that $w \in L(A,G)$ but $w \notin L(B,G)$, for any distinct states p and q of M, there exists an associate word α such that $\delta(p,\alpha)\in F$ and $\delta(q,\alpha)\notin F$, or vice versa.

Unfortunately, the converse case does not hold in general. For example, suppose that $G=<N,\Sigma,\Pi,S>$ is a linear grammar, where $N=\{S,A,B,C,D\}$, $\Sigma=\{a,b\}$, $\Pi=\{S\to b, S\to bA, S\to Bb, A\to aC, C\to Sa, B\to Da, D\to aS\}$. Then, G is not canonical because $L(A,G)=L(B,G)$. The universal linear grammar over Σ is $G^0=<\{S^0\},\Sigma,\Pi^0,S^0>$, where $\Pi^0=\{\pi^0_1{:}S^0\to aS^0, \pi^0_2{:}S^0\to bS^0, \pi^0_3{:}S^0\to S^0a, \pi^0_4{:}S^0\to S^0b, \pi^0_5{:}S^0\to a, \pi^0_6{:}S^0\to b, \pi^0_7{:}S^0\to \lambda\}$. The DFA corresponding to G is $M=<N,\Pi^0,\delta,S,\{q_F\}>$, where the transition function δ is defined as $\delta(S,\pi^0_6)=q_F$, $\delta(S,\pi^0_2)=A$, $\delta(S,\pi^0_4)=B$, $\delta(A,\pi^0_1)=C$, $\delta(C,\pi^0_3)=S$, $\delta(B,\pi^0_3)=D$, $\delta(D,\pi^0_1)=S$. It is easy to verify that M is canonical.

The next theorem follows immediately from Propositions 3.6 and 3.7.

Theorem 3.8.

For any linear language L, there exists a canonical finite automaton M such that $C=T(M)$ and $L=L_C(G^0)$ hold.

Thus, to identify a linear language L, we may construct a canonical finite automaton M corresponding to a canonical linear grammar which generates L.

4. Inference of Linear Languages

In this section, based on the results described in above sections, we provide the algorithm LID which efficiently identifies a linear language L.

4.1 Consistency for the membership oracle

Let L be any linear language. Let G be a canonical linear grammar such that $L=L(G)$ holds and $G^0=<\{S^0\},\Sigma,\Pi^0,S^0>$ be the universal linear grammar over Σ. Let

C be a regular control set such that $L=L_C(G^0)$ holds. In order to determine whether an associate word α^0 is in C or not, LID asks the oracle whether w is in L(G) or not for the string w such that $S^0 \Rightarrow_{G^0} \alpha^0$ w. But in general, even if $w \in L(G)$, α^0 is not always in C.

DEFINITION. A regular control set C is said to be *consistent* for G if and only if for any $\alpha^0 \in \Pi^{0*}$ and the string w such that $S^0 \Rightarrow_{G^0} \alpha^0$ w, $\alpha^0 \in C$ whenever $w \in L(G)$.

If any string in L(G) has the unique derivation, then G is said to be *unambiguous*. The next Proposition immediately follows from the definition.

Proposition 4.1.

If G is unambiguous, then any C is consistent for G.

In order to make G unambiguous, we introduce a parenthesis grammar which displays the derivations of the corresponding strings in L(G) by the nesting brackets.

DEFINITION. Let $G=\langle N,\Sigma,\Pi,S\rangle$ be a linear grammar. The *parenthesis grammar* of G is denoted by $[G]=\langle N,\Sigma\cup\{[,]\},\Pi',S\rangle$, where "[" and "]" are special symbols not in Σ and Π' is obtained from Π by replacing every production $A\rightarrow u$ by $A\rightarrow[u]$.

A parenthesis grammar [G] is *backwards-deterministic* if and only if no two productions in [G] have the same right side. Note that a backwards-deterministic parenthesis grammar [G] is unambiguous. Clearly, the parenthesis grammar $[G^0]$ of any universal linear grammar is backwards-deterministic. Therefore, the following Proposition holds.

Proposition 4.2.

Any regular control set is consistent for [G].

4.2 Inference Algorithm LID

Let L be any linear language. Let G be a canonical linear grammar such that $L=L(G)$ holds and G^0 be the universal linear grammar over Σ. Let $[G]=<N,\Sigma,\Pi,S>$ be the parenthesis grammar of G and $[G^0]=<\{S^0\},\Sigma,\Pi^0,S^0>$ be the parenthesis grammar of G^0. Let $M=<K,\Pi^0,\delta,S,F>$ be the canonical finite automaton corresponding to $[G^0]$ and C be the set T(M).

DEFINITION. The *representative sample* of L([G]) is a finite subset R_a of L([G]) such that for every production π of [G] there exists a string w in R_a such that π appears in the derivation $S \Rightarrow_{[G]}{}^{\alpha} w$.

A finite subset R_c of Π^{0*} is said to be the *associate representative sample* of C with respect to R_a if and only if it satisfies

$$R_c = \{ \alpha^0 \mid S^0 \Rightarrow_{[G^0]}{}^{\alpha^0} w,\ w \in R_a \}.$$

We note that for every variable A of [G], A appears in a derivation $S \Rightarrow_{[G]}{}^{\alpha} w$ for some $w \in R_a$.

Let P_c denote the set $Pr(R_c)$. Let d_0 be a new symbol which is not contained in P_c. We denote the augmented set $P_c \cup \{d_0\}$ by P_c'.

We define a function f on $P_c' \times \Pi^0$ by for all $\pi^0 \in \Pi^0$

$$f(d_0,\pi^0)= d_0$$

$$f(\mu,\pi^0)= \begin{cases} d_0 \text{ if } \mu \in R_c, \\ \\ \mu\pi^0 \ \text{ if } \mu \in P_c - R_c. \end{cases}$$

Let T_c' denote the set $P_c' \cup \{f(\mu,\pi^0) \mid \mu \in P_c \text{ and } \pi^0 \in \Pi^0\}$, and T_c denote $T_c' - \{d_0\}$. Let Σ' denote $\Sigma \cup \{[,]\}$.

To account for the relation between associate words of $[G^0]$ and elements of $L([G])$, we define a partial function g from Π^{0*} to Σ'^* such that if $S^0 \Rightarrow_{[G^0]}{}^{\alpha^0} w$ for $\alpha^0 \in \Pi^{0*}$ and $w \in \Sigma'^*$ then $g(\alpha^0)=w$, otherwise $g(\alpha^0)$ is undefined. If $g(\alpha^0)$ is undefined, then clearly w is not in $L([G])$, where $S^0 \Rightarrow_{[G^0]}{}^{\alpha^0} w$.

Next we define a series of equivalence relations $\equiv_i$ on T_c'. Let $N_1=\{\lambda\}$. we define a function E_1 from T_c' to 2^{N_1} such that $E_1(d_0)=\emptyset$ and for $\mu \in T_c$,

$$E_1(\mu)= \begin{cases} \{\lambda\} \text{ if } g(\mu) \in L([G]), \\ \\ \emptyset \text{ otherwise.} \end{cases}$$

Let $\equiv_1$ be a relation on T_c' such that for any μ and μ' $\mu \equiv_1 \mu'$ if and only if $E_1(\mu)=E_1(\mu')$.

Assume that for $i \geq 1$, N_i, E_i and $\equiv_i$ are defined. For any pair of $\mu,\mu' \in P_c'$, if $\mu \equiv_i \mu'$ but $f(\mu,\pi^0) \not\equiv_i f(\mu',\pi^0)$ for some production $\pi^0 \in \Pi^0$, then let $N_{i+1}=N_i \cup \{\pi^0 v\}$, where $v \in E_i(f(\mu,\pi^0)) \oplus E_i(f(\mu',\pi^0))$. We define a function E_{i+1} from T_c' to $2^{N_{i+1}}$ such that $E_{i+1}(d_0)=\emptyset$ and $E_{i+1}(\mu)= \{v \in N_{i+1} \mid g(\mu v) \in L([G])\}$ for $\mu \in T_c$. Let $\equiv_{i+1}$ be a relation on T_c' such that for any μ and μ' $\mu \equiv_{i+1} \mu'$ if and only if $E_{i+1}(\mu)=E_{i+1}(\mu')$. Clearly, $\equiv_{i+1}$ is an equivalence relation. We denote by $[\mu]_{\equiv_{i+1}}$ an equivalence class of T_c' which contains μ. Since T_c' is finite, there exists a positive integer m such that $\equiv_m = \equiv_{m+1} = \ldots$ holds.

Lemma 4.4.

For any $\mu \in T_c$, if $E_m(\mu)$ is not empty then there exists a $\mu' \in P_c$ such that $\mu \equiv_m \mu'$ holds.

Proof. Clearly, if $\mu \in P_c$ then the assertion holds. Suppose that $\mu \in T_c - P_c$ and $E_m(\mu) \neq \emptyset$. Then, for any $v \in E_m(\mu)$, since $g(\mu v) \in L([G])$, there exists a variable A of [G] such that $S \Rightarrow_{[G]}{}^{\alpha} uAv \Rightarrow_{[G]}{}^{\beta} w$, where $w=g(\mu v)$, $h(\alpha)=\mu$, $h(\beta)=v$ and $u,v \in \Sigma^*$. By definition of R_a, there exists a string $w' \in R_a$ such that $S \Rightarrow_{[G]}{}^{\alpha'} u'Av' \Rightarrow_{[G]}{}^{\beta'} w'$, where $u',v',w' \in \Sigma^*$. Therefore, there exists a $\mu'=h(\alpha') \in P_c$. Hence, $\mu \equiv_m \mu'$.

Given an oracle to answer membership questions about L([G]) and a representative sample Ra of L([G]), the inference algorithm LID constructs the canonical finite automaton M' such that C=T(M') holds.

LID constructs a finite automaton $M'=<K',\Pi^0,\delta',q_0',F'>$ as follows; The set of states K' is $T_c'/\equiv_m$, the initial state q_0' is $[\lambda]_{\equiv_m}$, and the set of final states F' is $\{[\mu]_{\equiv_m} \mid E_m(\mu)\ni\lambda\}$. The transition function δ' is defined as follows; if $E_m(\mu)$ is empty, then $\delta'([\mu]_{\equiv_m},\pi^0)=[\mu]_{\equiv_m}$ for all $\pi^0\in\Pi^0$. If $E_m(\mu)$ is not empty, then since by Lemma 4.4 there exists a $\mu'\in P_c$ such that $\mu\equiv_m\mu'$, $\delta'([\mu]_{\equiv_m},\pi^0)=[f(\mu',\pi^0)]_{\equiv_m}$ for all $\pi^0\in\Pi^0$.

If $\mu,\mu'\in P_c$ are such that $\mu\equiv_m\mu'$, then $f(\mu,\pi^0)\equiv_m f(\mu',\pi^0)$ for all $\pi^0\in\Pi^0$. Hence, the transition function δ' is well defined and M' is a well defined DFA. We note that if there exists $\mu\in T_c$ such that $\mu\equiv_m d_0$, then the state $[\mu]_{\equiv_m}$ is the dead state of M. Having constructed M', LID outputs a description of it and halts.

```
Algorithm LID
  input:    oracle for L([G]) and a representative sample $R_a$ of L([G]).
  output:   description of the canonical finite automaton M' such that C=T(M')
            and $L([G])=L_C([G^0])$ hold.
  procedure:
  begin
    /* construct the following sets from input $R_a$ */
    $R_c := \{ \alpha^0 \mid S^0 \Rightarrow_{[G^0]}\alpha^0\ w, w\in R_a \}$ ;
    $P_c := Pr(R_c)$ ;
    $P_c' := P_c \cup \{d_0\}$ ;
    $T_c' := P_c' \cup \{ f(\mu,\pi^0) \mid \mu\in P_c$ and $\pi^0\in\Pi^0 \}$ ;
    $T_c := T_c' - \{d_0\}$ ;

    /* initialization */
    for all $\mu\in T_c'$ do  $E_0(\mu) := \emptyset$ ;
    $v_1 := \lambda$ ;

    /* MAIN LOOP */
    for i:=1 to infinite do
      begin
        $E_i(d_0) := \emptyset$ ;
        /* construct $E_i(\mu)$ by querying the oracle for L([G]) about the string $g(\mu v_i)$
             for each $\mu\in T_c$ */
    for all $\mu \in T_c$ do
      if      $g(\mu v_i) \in L([G])$
      then    $E_i(\mu) := E_{i-1}(\mu) \cup \{v_i\}$
      else    $E_i(\mu) := E_{i-1}(\mu)$ ;
    if
          found a pair of $\mu, \mu' \in P_c'$ and a production $\pi^0 \in \Pi^0$ such that $\mu \equiv_i \mu'$
          but $f(\mu, \pi^0) \not\equiv_i f(\mu', \pi^0)$
    then
        begin
          choose some associate word $v \in E_i(f(\mu,\pi^0)) \oplus E_i(f(\mu',\pi^0))$ ;
          $v_{i+1} := \pi^0 v$
        end
    else
        begin
          m := i ;
          construct a finite automaton M and halt
        end
  end
end.
```

Example.

Consider a linear language $L=\{ a^n b a^n \mid n \geq 0\}$. Let $G=<N,\Sigma,\Pi,S>$ be a linear grammar, where $N=\{S,A\}$, $\Sigma=\{a,b\}$, $\Pi=\{S \to aA, S \to b, A \to Sa\}$. Then, G is canonical and generates L. Let $[G]=<N,\Sigma,\Pi',S>$ be the parenthesis grammar of G, where $\Pi'=\{S \to [aA], S \to [b], A \to [Sa]\}$. The parenthesis grammar of the universal linear grammar over Σ is $[G^0]=<\{S^0\},\Sigma,\Pi^0,S^0>$, where $\Pi^0=\{\pi^0{}_1{:}S^0 \to [aS^0], \pi^0{}_2{:}S^0 \to [bS^0], \pi^0{}_3{:}S^0 \to [S^0a], \pi^0{}_4{:}S^0 \to [S^0b], \pi^0{}_5{:}S^0 \to [a], \pi^0{}_6{:}S^0 \to [b]\}$. A representative sample R_a for $L([G])$ is $\{[a[[b]a]]\}$. Then, the associate representative sample R_c with respect to R_a is $\{\pi^0{}_1\pi^0{}_3\pi^0{}_6\}$. P_c is $\{\lambda, \pi^0{}_1, \pi^0{}_1\pi^0{}_3, \pi^0{}_1\pi^0{}_3\pi^0{}_6\}$ and T_c is $\{\lambda, \pi^0{}_1, \pi^0{}_2, \pi^0{}_3, \pi^0{}_4, \pi^0{}_5, \pi^0{}_6, \pi^0{}_1\pi^0{}_1, \pi^0{}_1\pi^0{}_2, \pi^0{}_1\pi^0{}_3, \pi^0{}_1\pi^0{}_4, \pi^0{}_1\pi^0{}_5, \pi^0{}_1\pi^0{}_6, \pi^0{}_1\pi^0{}_3\pi^0{}_1, \pi^0{}_1\pi^0{}_3\pi^0{}_2, \pi^0{}_1\pi^0{}_3\pi^0{}_3, \pi^0{}_1\pi^0{}_3\pi^0{}_4, \pi^0{}_1\pi^0{}_3\pi^0{}_5, \pi^0{}_1\pi^0{}_3\pi^0{}_6\}$.

In the final partition, $E_m(\mu)=\{\lambda\}$ for $\mu=\pi^0{}_6, \pi^0{}_1\pi^0{}_3\pi^0{}_6$, $E_m(\mu)=\{\pi^0{}_6\}$ for $\mu=\lambda, \pi^0{}_1\pi^0{}_3$, $E_m(\mu)=\{\pi^0{}_3\pi^0{}_6\}$ for $\mu=\pi^0{}_1, \pi^0{}_1\pi^0{}_3\pi^0{}_1$, and $E_m(\mu)=\emptyset$ for all other μ from T_c'. The output of LID is the DFA $M'=<K,\Pi^0,\delta,q_0,F>$ as follows;

1. $K=\{q_0, q_1, q_2, q_3\}$, where

   $q_0=\{\lambda, \pi^0{}_1\pi^0{}_3\}$,

$q_1=\{\pi^0{}_1, \pi^0{}_1\pi^0{}_3\pi^0{}_1\}$,

$q_2=\{\pi^0{}_6, \pi^0{}_1\pi^0{}_3\pi^0{}_6\}$,

$q_3=\{d_0, \pi^0{}_2, \pi^0{}_3, \pi^0{}_4, \pi^0{}_5, \pi^0{}_1\pi^0{}_1, \pi^0{}_1\pi^0{}_2, \pi^0{}_1\pi^0{}_4, \pi^0{}_1\pi^0{}_5, \pi^0{}_1\pi^0{}_6, \pi^0{}_1\pi^0{}_3\pi^0{}_2, \pi^0{}_1\pi^0{}_3\pi^0{}_3, \pi^0{}_1\pi^0{}_3\pi^0{}_4, \pi^0{}_1\pi^0{}_3\pi^0{}_5\}$.

2. $F=\{q_2\}$,

3. the transition function $\delta:K\times\Pi^0\to K$ is defined as follows

$\delta(q_0,\pi^0{}_1)=q_1$,

$\delta(q_0,\pi^0{}_6)=q_2$,

$\delta(q_1,\pi^0{}_3)=q_0$,

$\delta(q,\pi^0)=q_3$ for all other $\pi^0\in\Pi^0$ and $q\in K$

Regarding states of M' as variables, we get a grammar $G'=\langle N',\Sigma,\Pi',S'\rangle$, where N' is $\{S',A'\}$ and $S'=q_0$, $A'=q_1$, and Π' is $\{S'\to[aA'], S'\to[b], A'\to[S'a]\}$. Clearly, $L(G)=L(G')$.

5. Correctness and Effectiveness for LID

In this section, we prove the correctness of the algorithm LID and give an upper bound on the number of queries it must make.

Let L be a linear language over Σ. Let G be a canonical linear grammar which generates L, and $[G]=\langle N,\Sigma,\Pi,S\rangle$ be the parenthesis grammar of G. Let G^0 be the universal linear grammar over Σ, and $[G^0]=\langle\{S^0\},\Sigma,\Pi^0,S^0\rangle$ be the parenthesis grammar of G^0.

Let $M=\langle K,\Pi^0,\delta,q_0,F\rangle$ be the canonical finite automaton corresponding to [G], and $M'=\langle K',\Pi^0,\delta',q_0',F'\rangle$ be the canonical finite automaton which LID has constructed.

First we prove that LID always halts.

Lemma 5.1.

Let k be the size of the alphabet Σ and n is the number of prefixes of elements of R_c. LID executes the MAIN LOOP at most $(3k+1)n$ times and halts.

Proof. Clearly, the parenthesis grammar $[G^0]$ of the universal linear grammar over Σ has $3k+1$ productions. Since the cardinality of P_c is n, T_c contains no more than $(3k+1)n$ elements. The final partition of T_c' contains at most $(3k+1)n$ equivalence classes, so it is achieved after at most $(3k+1)n$ executions.

Next we define a function ϑ from the set of states of M' to the set of states of M by mapping $[\mu]_{\equiv_m}$ to $\delta(q_0,\mu)$ for all $\mu \in T_c$. The following lemma ensures that ϑ is well defined.

Lemma 5.2.

For all distinct $\mu_1,\mu_2 \in T_c$, if $\delta(q_0,\mu_1) \neq \delta(q_0,\mu_2)$, then $\mu_1 \not\equiv_m \mu_2$.

Proof. Let C be a regular control set T(M). For distinct $\mu_1,\mu_2 \in T_c$, if $\delta(q_0,\mu_1) \neq \delta(q_0,\mu_2)$, then there exists an associate word v such that $\mu_1 v \in C$ and $\mu_2 v \notin C$, or vice versa because M is canonical. We will prove that $\mu_1 \not\equiv_m \mu_2$ by an induction on the length of v. Clearly, if $v=\lambda$, then $\mu_1 \in C$ and $\mu_2 \notin C$, therefore, $g(\mu_1) \in L([G])$ and $g(\mu_2) \notin L([G])$, or vice versa. Hence, $v \in E_m(\mu_1) \oplus E_m(\mu_2)$ and $\mu_1 \not\equiv_m \mu_2$.

Inductively suppose that for some $n \geq 0$, whenever $\mu_1, \mu_2 \in T_c$ are such that $\mu_1 v \in C$ and $\mu_2 v \notin C$ for some v of length at most n, then $\mu_1 \not\equiv_m \mu_2$ holds. Suppose that for $\mu_1, \mu_2 \in T_c$ and some associate word $\pi^0 v$, $\mu_1 \pi^0 v \in C$ and $\mu_2 \pi^0 v \notin C$ hold. Since $\delta(q_0,\mu_1)$ is a live state of M, there exists an associate word $\mu_1' \in P_c$ such that $\delta(q_0,\mu_1)=\delta(q_0,\mu_1')$, so $\mu_1'\pi^0 v \in C$. If $\delta(q_0,\mu_2)$ is a live state of M, then there exists an associate word $\mu_2' \in P_c$ such that $\delta(q^0,\mu_2)=\delta(q^0,\mu_2')$, therefore, $\mu_2'\pi^0 v \notin C$. Since $\mu_1'\pi^0 v \in C$ and $|v| \leq n$, $\mu_1'\pi^0 \not\equiv_m \mu_2'\pi^0$ by the inductive hypothesis. But $f(\mu_1',\pi^0)=\mu_1'\pi^0$ and $f(\mu_2',\pi^0)=\mu_2'\pi^0$, so we have $\mu_1' \not\equiv_m \mu_2'$. Since $\mu_1 \equiv_m \mu_1'$ and $\mu_2 \equiv_m \mu_2'$, we have $\mu_1 \not\equiv_m \mu_2$. If $\delta(q_0,\mu_2)$ is the dead state of M, then $E_m(\mu_2)=\emptyset$ and $g(\mu_2 v) \notin L([G])$. Since

$\mu_1'\pi^0 v \in C$ and $|v| \leq n$, $\mu_1'\pi^0 \not\equiv_m \mu_2$ by the inductive hypothesis. Thus, $E_m(\mu_1') \neq \emptyset$, for otherwise we must have $\mu_1' \equiv_m d_0$, $\mu_1'\pi^0 \equiv_m f(\mu_1', \pi^0) \equiv_m f(d_0, \pi^0)$ and $E_m(\mu_1'\pi^0) = \emptyset$, a contradiction. Hence, $E_m(\mu_1)$ and $E_m(\mu_1')$ is not empty and $\mu_1 \equiv_m \mu_1' \not\equiv_m \mu_2$. This completes the proof of the Lemma.

Lemma 5.3.

$\mathcal{J}$ is bijective.

Proof. By Lemma 5.2, $\mathcal{J}$ is well-defined. Since for all $\mu_1, \mu_2 \in T_c$, $\delta(q_0, \mu_1) = \delta(q_0, \mu_2)$ implies $\mu_1 \equiv_m \mu_2$, $\mathcal{J}$ is injective. If q is a state of M, then $\delta(q_0, \mu) = q$ for some $\mu \in T_c$, so $E_m(\mu)$ is mapped to q, and $\mathcal{J}$ is surjective.

THEOREM 5.4.

Given an oracle for a linear language $L([G])$ and a representative sample R_s of $L([G])$, LID outputs a description of the finite automaton which accepts the regular control set C such that $L([G])$ is generated by $[G^0]$ with C.

Proof. By Lemma 5.1, LID halts and outputs a description of the finite automaton M'. Let C be a regular control set which M accepts.

By Lemma 5.3, $\mathcal{J}$ is bijective. The initial state $[\lambda]_{\equiv_m}$ of M' is mapped to the initial state $\delta(q_0, \lambda) = q_0$ of M. If $[\mu]_{\equiv_m}$ is a final state of M', then since $E_m(\mu) \ni \lambda$ and $g(\mu) \in L(G)$, $\mu \in C$. Therefore, $\delta(q_0, \mu)$ is a final state of M. For $\mu \in T_c$ and $\pi^0 \in \Pi^0$, if $E_m(\mu) = \emptyset$, then $[\mu]_{\equiv_m}$ is mapped to the dead state of M, and these two states map to themselves under the input π^C. If $E_m(\mu) \neq \emptyset$, the transition on input π^0 from $[\mu]_{\equiv_m}$ is to state $[f(\mu', \pi^0)]_{\equiv_m}$, which is mapped to $\delta(q_0, \mu'\pi^0) = \delta(\delta(q_0, \mu'), \pi^0)$. Thus, the following diagram commutes;

$$\begin{tikzcd}
{[\mu]_{\equiv_m}} \arrow[r, "\mathcal{J}"] \arrow[d, "\delta'"'] & \delta(q_0, \mu) \arrow[d, "\delta"] \\
\delta'([\mu]_{\equiv_m}, \pi^0) \arrow[r, "\mathcal{J}"] & \delta(q_0, \mu\pi^0)
\end{tikzcd}$$

Therefore, M' is isomorphic to M. This completes the proof.

THEOREM 5.5.

Given an oracle for L([G]) and a representative sample R_a of L([G]), LID makes no more than $((3k+1)n)^2$ queries, where k is the size of the alphabet Σ, and n is the number of prefixes of elements of R_c.

Proof. By Lemma 5.1, after at most (3k+1)n executions, LID halts and outputs a description of the finite automaton M'. Each execution requires at most one query for each element of T_c, so the total number of queries is no more than $((3k+1)n)^2$.

THEOREM 5.6.

LID can be implemented to run in time polynomial in the size of the alphabet Σ and the input R_a.

Proof. Constructing R_c from R_a takes time polynomial in the length of element of R_a by Earley's algorithm (Earley, 1970). Searching for a pair to refine the current partition in the obvious way takes time polynomial in the representation of T_c'. Therefore, the running time of LID will be polynomial in k and the size of the input R_a.

ACKNOWLEDGEMENTS

The author is indebted to Dr. Tosio Kitagawa, the president of IIAS-SIS, Dr. Hajime Enomoto, the director of IIAS-SIS, for their useful suggestion and warm encouragement. He is also grateful to his colleagues, Dr. Takashi Yokomori and Dr. Taishin Nishida who worked through an earlier draft of the paper and gave many suggestions.

This is a part of the work in the major R&D of the Fifth Generation Computer Project, conducted under program set up by MITI.

REFERENCES

Angluin, D. (1981), "A Note on the Number of Queries Needed to Identify Regular Languages", *Information and Control*, 51, 76-87.

Biermann, A. W. (1971), "A Grammatical Inference Program for Linear Languages", *Forth Hawaii International Conference on System Sciences*, Honolulu, Hawaii, 12-14.

Biermann, A. W. (1972), "An Interactive Finite-state Language Learner", *Proceedings of First USA-JAPAN Computer Conference*, 13-20.

Earley, J. (1970), "An Efficient Context-Free Parsing Algorithm", *Comm. of the ACM*, 13, 94-102.

Ginsburg, S., and Spanier, E. H. (1968), "Control Sets on Grammars", *Math. Systems Theory*, 2, 159-177.

Harrison, M. A. (1978), "Introduction to Formal Language Theory", Addison-Wesley, Reading, Mass.

Hopcroft, J. E., and Ullman, J. D., (1979), "Introduction to Automata Theory, Languages and Computation", Addison-Wesley, Reading, Mass.

McNaughton, R. (1967), "Parenthesis Grammars", *J. of the ACM*, 14, 490-500.

Tanatsugu, K. (1987), "A Grammatical Inference for Context-free Languages based on Self-embedding", *Bulletin of Informatics and Cybernetics*, 22, 149-163.