INFORMATION AND CONTROL 38, 310–329 (1978)

On the Inference of Stochastic Regular Grammars

A. VAN DER MUDE* AND ADRIAN WALKER†

Department of Computer Science, Rutgers University, New Brunswick, New Jersey 08903

The relevance of grammatical inference techniques to the semiautomatic construction from empirical data, of a model of human decision making, is outlined. A grammatical inference problem is presented in which the least complex stochastic regular grammar is sought which describes a given set of strings. An upper bound on the complexity of the best grammar for a given data set is found, and some properties of the grammars which are less complex than the bound are proved. The technique of splitting grammars is used to organize a search of these grammars. An initial grammar is defined, and it is established that any best grammar is obtainable by repeated splitting of the initial grammar. The performance of a program based on these results is described.

1. INTRODUCTION

In computer science, and particularly in artificial intelligence, we are often confronted by a large body of empirical data which we wish to summarize as a theory. In order to test the theory, we write it as a program. A number of programs both in the area of simulation of human cognitive processes and in the area of matching human expert performance have been written in the form of production systems (see Davis and King, 1977). We can classify the way in which such programs are constructed according to the amount of help which the computer provides during writing and debugging. At one extreme the program is written entirely by a person, and the computer just provides a faster and more reliable method than hand simulation to find out which features must be modified. At an intermediate level, production rules are acquired by a conversational interaction between a knowledge acquisition program and a human expert, see, e.g., Davis *et al.* (1977). The emphasis at this level is on computational assistance in modifying a set of production rules, which is initially written by a person, by specifically adding or deleting rules. At the most automated level, we are interested in placing as much as we can of the burden of rule writing on the computer itself. While this last level of activity raises some difficult problems, we believe that the general emphasis of the work which is being done in theory construction is moving gradually, along the continuum we have outlined, toward the more highly automated level. It therefore seems reasonable to pick

* Present address: 14701 SW 87 Ct., Miami, FL. 33176.

† Present address: Bell Laboratories, 600 Mountain Avenue, Murray Hill, N.J. 07974.

0019–9958/78/0383–0310$02.00/0

a specific problem which is outside the range of current computational practice and to see how much progress can be made in solving it.

Inductive inference is a paradigm for theory construction, and one particular inductive inference problem that has been studied is grammatical inference, see Biermann and Feldman (1973). In grammatical inference, the problem is to find an algorithm which, given a set of strings as data, produces a grammar whose language closely approximates the data. The grammar may be considered a theory that describes the strings, which are observations. A best grammar is one which is simple, which generates the given strings, and which generates as few other strings as possible.

In the literature on grammatical inference, the initial work established that certain problems were solvable in principle, while others were not. Gold (1967) considered the case when the data set is constantly increasing as new strings are added, and formalized the notion of "language identifiability in the limit." He studied different classes of grammars and found for which classes of grammars and types of data an algorithm existed, and for which classes and types of data no algorithm could exist. The problem of time complexity of the inference process was not addressed directly in this study. Horning (1969) considered the case in which a stochastic context-free grammar is to be constructed to be a good fit (in a Bayesian sense) to the data, and in which almost any reasonable measure of the simplicity of a grammar may be used. He showed that the required inference algorithm exists, in the sense that an enumeration of all candidate grammars in approximate order of decreasing simplicity will eventually yield a best grammar. So far, the known algorithms (see Fu and Booth, 1975) yield programs with computing times which increase rapidly with the size of the grammar required to model the data. However, Gaines (1977) has shown that it is possible to solve nontrivial grammatical inference problems, and he has described a general system theoretic framework for such problems.

In this paper we define a specific grammatical inference problem, we establish some properties of the problem which relate to its efficient algorithmic solution, and we describe an experimental program which will always produce a solution. The running time of the program depends on the initial information supplied. In the problem which we have chosen, a finite set of strings of symbols is given, together with information on the frequency of occurrence of each string, and the requirement is to construct a stochastic regular grammar which generates the strings. The grammar should, on the one hand, assign high probabilities to the more common strings, while, on the other hand, the grammar should be simple. Although we consider only regular grammars, we do consider an ambiguous grammar to be a possible output from a grammatical inference program. Our program includes an implementation of a splitting technique of Horning (1969). To that technique we add some results about a unique root for a tree of grammars, and about deductive methods for pruning such a tree.

Our reasons for choosing this particular problem are as follows. We are

interested in the process by which a theory is constructed from a number of empirical observations, in the case when the theory can be represented by a directed, labeled graph in which each arc has a normalized rational weight attached to it. We call such a graph a stochastic graph. To convert such a graph to a stochastic regular grammar, and vice versa, is little more than a change in notation. In an interpretation of a stochastic graph which forms our practical motive for this work, the graph is thought of as a model of a decision-making system in which events, represented by the nodes, tend with certain weights to cause other events, and in which causal chains and loops can be present, see Walker (1977). The importance of a choice of representation for a theory formation task is described by Amarel (1971). Our interpretation of a stochastic graph can be regarded as a representational hypothesis. Some other theory representations which share some features with stochastic graphs are, for instance, the causal networks used in the medical model CASNET (see Weiss *et al.*, 1977) and the production systems MYCIN (Davis *et al.*, 1977; see also Davis and King, 1977), DENDRAL (Buchanan *et al.*, 1972) and RITA (Waterman, 1977).

In Section 2 we describe the notation and definitions which we use, and we give a precise statement of our inference problem. In Section 3 we prove some results about some features of the problem which are relevant to its solution. Then, in Section 4, we describe a program based on the results in Section 3, and we give some examples of grammars which are constructed by the program. Finally, in Section 5, we indicate some directions for further work on this problem, and we describe some related problems which may be of practical interest.

2. Notation and Definitions

We follow the notation of Hopcroft and Ullman (1969), to which we add the following items.

Definition. A *stochastic regular grammar* (SRG) is a grammar $G = \langle V_N, V_T, P, A_1 \rangle$ in which V_N is an ordered set of symbols of size N, $V_N = \langle A_1, \ldots, A_i, \ldots, A_N \rangle$, V_T is an unordered set of terminal symbols, $A_1 \in V_N$ is the start symbol, and each production in the finite set P is in one of the forms

$$c_k/d_i\colon A_i \to aA_j,$$
$$c_k/d_i\colon A_i \to a,$$

where $c_k \leqslant d_i$ are positive integers. If $A_i \in V_N$ then there exists at least one production in P with A_i on the left. The productions in P with A_i on the left are of the form $c_r/d_i\colon A_i \to \beta_r$ for $r = 1, \ldots, t$ where (i) $\sum_{r=1}^{t} c_r = d_i$ and (ii) $\gcd(c_1, \ldots, c_t, d_i) = 1$. We call c_r/d_i a *probability* and we call d_i the *index* of A_i.

We say that the *complexity* of G is $C(G) = \sum_{i=1}^{N} d_i$, and that the *probability* of G is $P(G) = 2^{-C(G)}$; see the remark at the end of this section.

DEFINITION. A *base grammar* is a stochastic regular grammar $G = \langle V_N, V_T, P, A_1 \rangle$ in which each of the t productions with A_i on the left-hand side has probability $1/t$.

Notation. We shall sometimes write $A_i \to \beta$ instead of c_k/d_i: $A_i \to \beta$ when the values of c_k and d_i need not be specified.

Notation. If the nonterminal set of a grammar is of size $N + M$ then we write $V_{N+M} = \langle A_1, \ldots, A_N, A_{N+1}, \ldots, A_{N+M} \rangle$.

DEFINITION. Let G be a SRG and let $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_m$ be a derivation d in G in which a production having probability p_t is used in the tth step, $1 \leqslant t \leqslant m$. Then we say that the *probability of the derivation d in G* is $\prod_{t=1}^{m} p_t$. If $\alpha \in L(G)$ has k distinct derivations, each having probability p_j in G, we say that the *probability of* α *given* G is $P(\alpha \mid G) = \sum_{j=1}^{k} p_j$, while if $\alpha \notin L(G)$ we define $P(\alpha \mid G) = 0$.

DEFINITION. A *data set* is a set of the form $D = \{\langle \alpha_i, n_i \rangle \mid 1 \leqslant i \leqslant m\}$ where $m \geqslant 1$, α_i is a nonempty string over a finite alphabet, and n_i is a positive integer. The *language* of D, $L(D)$ is the set $\{\alpha_i \mid 1 \leqslant i \leqslant m\}$.

If D is a data set and G is a SRG we define the *probability of D given G* as $P(D \mid G) = \prod_{i=1}^{m} (P(\alpha_i \mid G))^{n_i}$.

DEFINITION. Let G be a SRG and let $D = \{\langle \alpha_i, n_i \rangle \mid 1 \leqslant i \leqslant m\}$. We say that G is *reduced wrt D* if for each production q of G there exists an $\alpha_i \in L(D)$ and a derivation of α_i in G which makes use of q.

We say that G is *reduced* if for each production q of G there is an $\alpha \in L(G)$ and a derivation of α in G which makes use of q.

DEFINITION. If V_T is the smallest alphabet such that $L(D) \subseteq V_T^+$ then the *final alphabet* of D is defined as $F_D = \{a \mid \exists \alpha \in L(D), \exists \beta \in V_T^*, \alpha = \beta a\}$. We also define the *embedded alphabet* of D as $E_D = \{a \mid \exists \alpha \in L(D), \exists \beta \in V_T^*, \exists \gamma \in V_T^+, \alpha = \beta a \gamma\}$. Note that $F_D \cap E_D$ is not necessarily empty.

DEFINITION. Given a data set D, we define the *universal grammar* for D as $G_u = \langle \langle A_1 \rangle, V_T, P, A_1 \rangle$ where G_u is a base grammar with P defined as follows:

$$A_1 \to aA_1 \qquad \text{is in } P \text{ iff } a \in E_D,$$
$$A_1 \to a \qquad \text{is in } P \text{ iff } a \in F_D.$$

Note that $C(G_u) = |E_D| + |F_D|$, where $|E|$ denotes the cardinality of E.

DEFINITION. Given a data set $D = \{\langle \alpha_i , n_i \rangle \mid 1 \leqslant i \leqslant m\}$ where $\alpha_i = a_{i1} \cdots a_{il_i}$, $l_i \geqslant 1$, we define the *tree grammar* for D as $G_D = \langle V_N , V_T , P, A_1 \rangle$ where the cardinality of V_N is $N = 1 + \sum_{i=1}^{m} (l_i - 1)$, and $V_T = \{a_{ij} \mid 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant l_i\}$.

Let B_{ij} be distinct symbols different from A_1 and let $n = \sum_{i=1}^{m} n_i$. Then P is defined as follows:

$$
\begin{aligned}
P = {} & \{n_i'/n' : A_1 \to a_{i1} \mid 1 \leqslant i \leqslant m, l_i = 1\} \\
& \cup \{n_i'/n' : A_1 \to a_{i1}B_{i1} \mid 1 \leqslant i \leqslant m, l_i > 1\} \\
& \cup \{1/1 : B_{ij} \to a_{i(j+1)}B_{i(j+1)} \mid 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant l_i - 2\} \\
& \cup \{1/1 : B_{i(l_i-1)} \to a_{il_i} \mid 1 \leqslant i \leqslant m\},
\end{aligned}
$$

where $n_1'/n' = n_i/n$ for $1 \leqslant i \leqslant m$, and $\gcd(n_1', \ldots, n_m', n') = 1$.

Notation. A_0 denotes the empty string λ.

DEFINITION. Let $G = \langle V_N , V_T , P, A_1 \rangle$ be a base grammar. For any production p in P the *split set* of p, $Q(p)$, is defined as follows:

(1) $Q(p) = \{A_1 \to aA_1 , A_1 \to aA_{N+1} , A_{N+1} \to aA_1 , A_{N+1} \to aA_{N+1}\}$ iff $p = A_1 \to aA_1$.

(2) $Q(p) = \{A_1 \to aA_i , A_{N+1} \to aA_i\}$ iff $p = A_1 \to aA_i$, $i \neq 1$, where i may equal 0, (i.e., $p = A_1 \to a$).

(3) $Q(p) = \{A_i \to aA_1 , A_i \to aA_{N+1}\}$ iff $p = A_i \to aA_1$, $i \neq 1$.

(4) $Q(p) = \{A_i \to aA_j\}$ iff $p = A_i \to aA_j$, $i \neq 1$ and $j \neq 1$, where j may equal 0.

A base grammar $G' = \langle V_{N+1} , V_T , P', A_1 \rangle$ is a *split* of G iff for all $p \in P$ $Q(p) \cap P' \neq \varnothing$, and for all $q \in P'$ there exists a $p \in P$ such that $q \in Q(p)$. We also define G to be a *merge* of G'.

Given a base grammar $G' = \langle V_{N+M} , V_T , P', A_1 \rangle$, G' is in the set *splits* (G) if there exist base grammars $G_i = \langle V_{N+i} , V_T , P_i , A_1 \rangle$, $0 \leqslant i \leqslant M$, where $G_0 = G$, $G_M = G'$, and G_{i+1} is a split of G_i.

DEFINITION. A base grammar $G_M = \langle V_{N+1} , V_T , P_M , A_1 \rangle$ is the *mainsplit* of $G = \langle V_N , V_T , P, A_1 \rangle$ iff $P_M = \bigcup_{p \in P} Q(p)$.

DEFINITION. A base grammar $G' = \langle V_N , V_T , P', A_1 \rangle$ is a *subgrammar* of $G = \langle V_N , V_T , P, A_1 \rangle$ iff $P' \subseteq P$. If G and G' are both splits of the same grammar then G' is a *true subgrammar* of G.

Next we define the terms best grammar, deductively acceptable, and search limit.

DEFINITION. For a data set D and SRG G, G is *deductively acceptable* (DA) if $L(D) \subseteq L(G)$ and G is reduced wrt D.

DEFINITION. If D is a data set and G is a SRG, we define the *search limit* for G to be $C(G) + \lfloor \log_2(P(D \mid G_D)/P(D \mid G)) \rfloor$ where G_D is the tree grammar for D. Note that the search limit is an integer.

Finally, we define the problem which we wish to solve.

DEFINITION. The *free inference problem for SRG's* is the following. Give a compurationally feasible algorithm which takes as input a data set D and produces as output a SRG $\hat{G}$ such that $P(D \mid \hat{G})\, P(\hat{G})$ is greatest. This $\hat{G}$ is said to be a *best grammar* for D.

Remark. It may be useful to view $C(G)$ as an extension of the notion of a count of the productions of G. In fact if G is a base grammar, then $C(G)$ is equal to the number of productions. Our $P(G)$ and $P(D \mid G)$ are based on those of Horning (1969), who gives a Bayesian argument for maximizing their product. His $P(G)$ is assigned to G by a stochastic grammar-grammar, i.e. a grammar which generates grammars. Our $P(G)$, though unnormalized, seems, with $P(D \mid G)$, to capture the intuitive idea of a good grammar for a data set. The reader may prefer to view our free inference problem as that of finding a $\hat{G}$ for which $C(D \mid \hat{G}) + C(\hat{G})$ is least where $C(D \mid G) = -\log_2 P(D \mid G)$. For discussions of optimality criteria, see e.g. Gaines (1977) or Feldman and Shields (1977).

3. RESULTS ABOUT BEST GRAMMARS

The tree grammar of a data set has some properties which are useful in searching for a grammar which maximizes $P(D \mid G)\, P(G)$. To illustrate the construction of G_D from D, consider the following simple example.

EXAMPLE 1. Let $D = \{\langle a, 3\rangle, \langle aa, 4\rangle, \langle aaa, 2\rangle\}$. Then $G_D = \langle\langle A_1, B_{21}, B_{31}, B_{32}\rangle, \{a\}, P, A_1\rangle$ where P consists of the productions

$$\begin{aligned} &3/9\colon A_1 \to a, && 1/1\colon B_{21} \to a,\\ &4/9\colon A_1 \to aB_{21}, && 1/1\colon B_{31} \to aB_{32},\\ &2/9\colon A_1 \to aB_{31}, && 1/1\colon B_{32} \to a. \end{aligned}$$

So, $C(G_D) = 12$, $P(G_D) = 2^{-12}$ and $P(D \mid G_D) = (3/9)^3\,(4/9)^4\,(2/9)^2 = 2^{10}/3^{15}$. ∎

Note that in general if $D = \{\langle \alpha_i, n_i\rangle \mid 1 \leqslant i \leqslant m\}$ and $n = \sum_{i=1}^{m} n_i$ then $P(D \mid G_D) = \prod_{i=1}^{m} (n_i/n)^{n_i}$.

We first wish to show that if D is a data set and G is any SRG, then $P(D \mid G) \leqslant P(D \mid G_D)$.

We shall need the following Lemma.

LEMMA 1. *Let $m \geqslant 1$ and let $n_1, \ldots, n_m$ be positive integers with $\sum_{i=1}^{m} n_i = n$. If $p_1, \ldots, p_m$ are nonnegative rationals such that $\sum_{i=1}^{m} p_i = 1$, then $\prod_{i=1}^{m} (p_i)^{n_i}$ takes on its greatest value when $p_i = n_i/n$ for $1 \leqslant i \leqslant m$.*

Proof. Immediate from Gibbs' theorem (see, e.g., Watanabe, 1969). ∎

We use the Lemma to show that the tree grammar G_D maximizes $P(D \mid G)$.

THEOREM 1. *Let D be a data set. Then for any* SRG G, $P(D \mid G) \leqslant P(D \mid G_D)$.

Proof. Let $D = \{\langle \alpha_i, n_i \rangle \mid 1 \leqslant i \leqslant m\}$, $n = \sum_{i=1}^{m} n_i$, and let $G = \langle V_N, V_T, P, A_1 \rangle$ be a SRG. Let us write $p_i = P(\alpha_i \mid G)$. We consider the following cases.

1. $p_i = n_i/n$ for $1 \leqslant i \leqslant m$. In this case, it follows from the construction of G_D from D that $p_i = P(\alpha_i \mid G_D)$ for each i, so $P(D \mid G) = P(D \mid G_D)$.

2. $p_j \neq n_j/n$ for some j. In this case we proceed as follows. If G is a reduced grammar then, by Theorem 1 of Ellis (1969), $\sum_{\alpha \in L(G)} P(\alpha \mid G) = 1$. Hence it follows easily that if G is any SRG, not necessarily reduced, then $\sum_{\alpha \in L(G)} P(\alpha \mid G) \leqslant 1$. So $\sum_{i=1}^{m} p_i = t$ for some rational $t \leqslant 1$. We have the following subcases:

2.1. $t = 1$. It follows from Lemma 1 that $P(D \mid G) = \prod_{i=1}^{m} p_i^{n_i} \leqslant \prod_{i=1}^{m} (n_i/n)^{n_i} = P(D \mid G_D)$.

2.2. $t < 1$. Without loss of generality, let $p_i = a_i/b$ for $1 \leqslant i \leqslant m$, and let $(1 - t) = c/t$.

Let $D' = \{\langle \alpha_i, n_i' \rangle \mid 1 \leqslant i \leqslant m, n_1' = a_1 + c$, and $n_i' = a_i$ for $2 \leqslant i \leqslant m\}$, and let G' be the tree grammar for D'. Since $\sum_{i=1}^{m} n_i' = b$, we have $P(\alpha_1 \mid G') = p_1 + c/b$ and $P(\alpha_i \mid G') = p_i$ for $2 \leqslant i \leqslant m$. So $P(D \mid G) = \prod_{i=1}^{m} p_i^{n_i} < (r_1 + c/p_1) \prod_{i=2}^{m} p_i^{n_i} = P(D \mid G')$. Since, by construction of G', $\sum_{i=1}^{m} P(\alpha_i \mid G') = 1$, it follows from cases 1 and 2.1 that $P(D \mid G') < P(D \mid G_D)$. Hence $P(D \mid G) < P(D \mid G_D)$. ∎

Thus no SRG G has $P(D \mid G) > P(D \mid G_D)$. We recall that the statement of our free inference problem requires that we find a $\hat{G}$ which maximizes $P(D \mid G)\, P(G)$.

We show next that $\hat{G}$ lies among the grammars G having $P(G) \geqslant P(G_D)$.

COROLLARY 1. *If $\hat{G}$ is a grammar which maximizes $P(D \mid G)\, P(G)$, then $P(\hat{G}) \geqslant P(G_D)$.*

Proof. By Theorem 1, $P(D \mid \hat{G}) \leqslant P(D \mid G_D)$. Since $P(D \mid \hat{G})\, P(\hat{G}) \geqslant P(D \mid G_D)\, P(G_D)$, it follows that $P(\hat{G}) \geqslant P(G_D)$. ∎

Therefore in searching for a best grammar for a data set D we need only look at grammars with a complexity $C(G) \leqslant C(G_D)$. This gives us an upper bound

for the search. But since $P(D \mid G_D)$ has been shown to be maximal we can reduce this upper bound as better grammars are found. This comes as a result of the following Lemma.

LEMMA 2. *Let D be a data set and let G and G' be SRG's. If G is such that $P(D \mid G)\,P(G) \geqslant P(D \mid G')\,P(G')$, then*

$$C(G) \leqslant C(G') + \lfloor \log_2(P(D \mid G_D)/P(D \mid G')) \rfloor.$$

Proof. Assume the opposite, i.e., there is a G, such that

$$P(D \mid G)\,P(G) \geqslant P(D \mid G')\,P(G') \tag{1}$$

and $C(G) > M + N$ where $M = C(G')$ and

$$N = \lfloor \log_2(P(D \mid G_D)/P(D \mid G')) \rfloor. \tag{2}$$

Since $C(G)$ takes on integer values, $C(G) \geqslant M + N + 1$, so

$$P(G) \leqslant 2^{-(M+N+1)}. \tag{3}$$

By Theorem 1,

$$P(D \mid G) \leqslant P(D \mid G_D). \tag{4}$$

So from (3) and (4)

$$P(D \mid G)\,P(G) \leqslant 2^{-(M+N+1)} P(D \mid G_D). \tag{5}$$

Now we can write $P(D \mid G')\,P(G')$ as $(P(D \mid G_D)(P(D \mid G')/P(D \mid G_D))2^{-M})$. From (2) $P(D \mid G')/P(D \mid G_D) > 2^{-(N+1)}$ so

$$P(D \mid G')\,P(G') > P(D \mid G_D)\,2^{-(M+N+1)}. \tag{6}$$

Hence from (1), (5), and (6) we have $2^{-(M+N+1)} > 2^{-(M+N+1)}$, which is a contradiction. ∎

In Section 4 we use the Lemma to bound the complexity of the grammars of interest during a search, based on the grammars already found. In this section, we shall use the Lemma in showing that G_D is not, in general, a best grammar for D. It is easy to see that, for any data set D, $P(D \mid G_D)\,P(G_D) > 0$, and that G_D is reduced wrt D. We shall show that any grammar G for which $P(D \mid G)\,P(G)$ is a greatest is reduced wrt D. We make use of the following result.

LEMMA 3. *There exists an algorithm which takes as input any data set D and any SRG G not reduced wrt D, and produces as output a SRG G' reduced wrt D such that $P(D \mid G)\,P(G) < P(D \mid G')\,P(G')$. Moreover, if G is unambiguous then so is G'.*

Proof. Let $D = \{\langle \alpha_i , n_i \rangle \mid 1 \leqslant i \leqslant m\}$ and $G = \langle V_N , V_T , P, A_1 \rangle$ be as in the statement of the Lemma. If $P(D \mid G) = 0$, then the Lemma holds with $G' = G_D$. Suppose $P(D \mid G) > 0$, and consider the following construction.

1. $Q = \{q \in P \mid$ there is an i, $1 \leqslant i \leqslant m$, such that q is used in a derivation of α_i in $G\}$.

2. $V'_T = \{X \in V_T \mid X$ appears in $Q\}$.

3. $V'_N = \langle X \in V_N \mid X$ appears in $Q \rangle$ where A_1 is the first element of V'_N and the other elements appear in any order.

4. For each $B \in V_N$ let P_B and Q_B be the productions in P and Q, respectively, having B on the left, denoted by $P_B = \{i_r/j_B \colon B \to \beta_r \mid 1 \leqslant r \leqslant t_B\}$, where t_B is the number of productions of P with B on the left and $Q_B = \{i_r/j_B \colon B \to \beta_r \mid 1 \leqslant r \leqslant s_B\}$, where s_B is the number of productions of Q with B on the left. Without loss of generality suppose that if $r > s_B$ then $i_r/j_B \colon B \to \beta_r$ is in P_B but not in Q_B.

5. $k_B = \sum_{r=1}^{s_B} i_r$.

6. Let i'_r, $1 \leqslant r \leqslant s_B$, j'_B be integers satisfying the following conditions:

   (i) There exists a $k \geqslant 1$ such that $k \cdot i'_r = i_r$, $1 \leqslant r \leqslant s_B$ and $k \cdot j'_B = j_B$.

   (ii) $\gcd(i'_1, \ldots, i'_{s_B}, j'_B) = 1$. Clearly, $i'_r/j'_B = i_r/k_B$.

7. $P'_B = \{i'_r/j'_B \colon B \to \beta_r \mid 1 \leqslant r \leqslant s_B\}$ is obtained from Q_B by replacing j_B by j'_B and i_r by i'_r. Clearly $\sum_{r=1}^{s_B} i'_r/j'_B = 1$.

8. $P' = \bigcup_{B \in V_N} P'_B$.

9. $G' = \langle V'_N , V'_T , P', A_1 \rangle$.

We claim that G' is a SRG, reduced wrt D, such that $P(D \mid G)\, P(G) < P(D \mid G')\, P(G')$.

To see that G' is an SRG, note that by steps 4–8 of the construction, the productions are in the required form, and for each $B \in V_N$

$$\sum_{r=1}^{s_B} i'_r/j'_B = 1.$$

Clearly $A_1 \in V'_N$.

It follows from steps 1, 4, 7, and 8 that G' is reduced wrt D. From steps 5 and 6 we have that for each $B \in V'_N$,

$$k_B = \sum_{r=1}^{s_B} i_r \leqslant \sum_{r=1}^{t_B} i_r = j_B \qquad \text{and} \qquad j'_B \leqslant k_B , \quad \text{so } j'_B \leqslant j_B .$$

Hence if $V'_N \subsetneqq V_N$ then $P(G) < P(G')$. If $V'_N = V_N$ then, since G is not reduced wrt D, it follows from steps 1 and 4 that there is a $B \in V'_N$ such that $s_B < t_B$,

hence $j'_B < j_B$. So again $P(G) < P(G')$. Since $i'_r/j'_B = i_r/k_B \geqslant i_r/j_B$, and every derivation in G is also a derivation in G', we have $P(D \mid G) \leqslant P(D \mid G')$. Hence $P(D \mid G)\,P(G) < P(D \mid G')\,P(G')$.

It is easy to see from steps 4–7 that if G is unambiguous, so is G'. ▮

We can use Lemma 3 to show that the best grammars for a data set are deductively acceptable.

THEOREM 2. *If D is a data set and G is a best grammar for D, then G is* DA.

Proof. Let G be a best grammar for D. Since $P(D \mid G)\,P(G) > 0$ it is easy to see that $L(D) \subseteq L(G)$. It follows from Lemma 3 that G is reduced wrt D. Hence G is DA. ▮

Although G_D generates each string in D and assigns an optimum probability to it, G_D is in general a complex grammar with a low value of $P(G_D)$, hence it may fail to maximize the product $P(D \mid G)\,P(G)$. We shall show that the algorithm for constructing G_D from D is not a solution to our free inference problem for SRG's.

THEOREM 3. *There exists a data set D such that if $\hat{G}$ is a SRG which maximizes $P(D \mid G)\,P(G)$ then $P(D \mid \hat{G}) < P(D \mid G_D)$.*

Proof. Let $D = \{\langle a, 2\rangle, \langle aa, 1\rangle\}$. Then the productions of G_D are

$$2/3\colon A_1 \to a, \qquad 1/1\colon B_{21} \to a,$$
$$1/3\colon A_1 \to aB_{21},$$

so $P(G_D) = 2^{-4}$ and $P(D \mid G_D) = 2^2/3^3$. Let G' be a grammar whose productions are

$$1/2\colon A_1 \to a, \qquad 1/2\colon A_1 \to aA_1.$$

Then $P(G') = 2^{-2}$ and $P(D \mid G') = 2^{-4} < P(D \mid G_D)$, but $P(D \mid G')P(G') > P(D \mid G_D)\,P(G_D)$.

Suppose there is a grammar G, different from G', such that $P(D \mid G)\,P(G) \geqslant P(D \mid G')\,P(G')$. Then, by Lemma 2, $C(G) \leqslant C(G') + [\log_2(P(D \mid G_D)/P(D \mid G'))]$ that is $C(G) \leqslant 3$. Since $P(D \mid G) > 0$, each of the strings a, aa is derivable in G. So $A_1 \to a$ and $A_1 \to aX$ are productions of G, where $X = A_1$ or $X = A_2$. Hence, since $C(G) \leqslant 3$, it is easy to see that G has no more than two nonterminals.

It follows from Lemma 3 that we may assume without loss of generality that G is reduced wrt D. Hence if G has two nonterminals, it follows easily from $C(G) \leqslant 3$ that the productions of G are:

$$1/2\colon A_1 \to a, \qquad 1/1\colon A_2 \to a,$$
$$1/2\colon A_1 \to aA_2.$$

In this case $P(D \mid G)\,P(G) = P(D \mid G')\,P(G')$ and $P(D \mid G) < P(D \mid G_D)$. On the other hand, if G has one nonterminal then its productions are clearly either

$$1/3\colon A_1 \to a, \qquad 2/3\colon A_1 \to aA_1,$$

or

$$2/3\colon A_1 \to a, \qquad 1/3\colon A_1 \to aA_1.$$

In either case $P(D \mid G)\,P(G) < P(D \mid G')\,P(G')$. Hence for the two best grammars $\hat{G}$ for D, $P(D \mid \hat{G}) < P(D \mid G_D)$, which completes the proof. ∎

Theorem 3 tells us that a best grammar $\hat{G}$ for a data set D does not necessarily have $P(D \mid \hat{G})$ equal to the maximum possible value $P(D \mid G_D)$. Hence the algorithm which constructs G_D from D is not a solution to our free inference problem for SRG's.

Next we investigate whether $\hat{G}$ is necessarily unambiguous, that is, whether each $\alpha \in L(\hat{G})$ has a unique derivation.

THEOREM 4. *There exists a data set D such that if $\hat{G}$ is a SRG which maximizes $P(D \mid G)\,P(G)$ then $\hat{G}$ is ambiguous.*

Proof. Let D and G_D be as in Example 1. We shall show that if $\hat{G}$ is a best grammar for D, then $\hat{G}$ is ambiguous.

Let G' be a grammar whose productions are

$$1/3\colon A_1 \to a, \qquad 1/3\colon A_1 \to aA_2,$$
$$1/3\colon A_1 \to aA_1, \qquad 1/1\colon A_2 \to a.$$

Note that G' is ambiguous. Then $C(G') = 4$, and $P(D \mid G') = 2^{12}/3^{17}$. Hence, by Lemma 2, if G is a SRG such that $P(D \mid G)\,P(G) \geqslant P(D \mid G')\,P(G')$ then $C(G) \leqslant 4 + \lfloor \log_2 9/4 \rfloor$, that is $C(G) \leqslant 5$. Let us suppose that there is such a G, and that it is not ambiguous. By Lemma 3 we may assume that G is reduced wrt D, that is, each production of G is used in deriving some string in $\{a, aa, aaa\}$.

Clearly $A_1 \to a$ is a production of G. Suppose both $A_1 \to aA_1$ and $A_1 \to aA_2$ are productions of G. Since G is reduced wrt D, $A_2 \Rightarrow a$ or $A_2 \overset{*}{\Rightarrow} aa$. But then G is ambiguous. Hence, either $A_1 \to aA_1$ is in G, or $A_1 \to aA_2$ is in G, but not both.

So, if $A_1 \to aA_1$ is in G, then G is one of the grammars $i/j\colon A_1 \to a$, $(j-i)/j\colon A_1 \to aA_1$, where $1 \leqslant i < j \leqslant 5$. It is straightforward to check that, for each such G, $P(D \mid G)\,P(G) < P(D \mid G')\,P(G')$.

Hence, $A_1 \to aA_2$ is in G and $A_1 \to aA_1$ is not in G. We recall that $A_1 \to a$ is in G. Suppose G has only A_1 and A_2 as nonterminals. Then since $aa \in L(G)$, $A_2 \to a$ is in G. Since $aaa \in L(G)$ and G is unambiguous, it follows that either

$A_2 \to aA_1$, or $A_2 \to aA_2$ is in G, but not both. So the productions of G are obtained by assigning probabilities to one of the following sets:

$$\begin{array}{ll} \{A_1 \to a, & \{A_1 \to a, \\ A_1 \to aA_2, & A_1 \to aA_2, \\ A_2 \to a, & A_2 \to a, \\ A_2 \to aA_1\} & A_2 \to aA_2\}. \end{array}$$

There are five possible ways to assign probabilities to each set so that $C(G) \leqslant 5$, and it is again straightforward to check that, for each G so obtained, $P(D \mid G)\, P(G) < P(D \mid G')\, P(G')$.

Suppose G has nonterminals A_1, A_2, A_3. Then, up to the renaming of A_2 and A_3, G is one of the grammars:

$$\begin{array}{lll} \{1/2: A_1 \to a & \{1/3: A_1 \to a & \{1/3: A_1 \to a, \\ 1/2: A_1 \to aA_2, & 1/3: A_1 \to aA_2, & 1/3: A_1 \to aA_2, \\ 1/2: A_2 \to a, & 1/1: A_2 \to a, & 1/1: A_2 \to a, \\ 1/2: A_2 \to aA_3, & 1/3: A_1 \to aA_3, & 1/3: A_1 \to aA_3, \\ 1/1: A_3 \to a\} & 1/1: A_3 \to aA_1\} & 1/1: A_3 \to aA_2\}. \end{array}$$

Again, $P(D \mid G)\, P(G) < P(D \mid G')\, P(G')$ for each G. Clearly G has no more than three nonterminals, so each best grammar for D is ambiguous, which completes our proof. ∎

Thus if we require a grammar which maximizes $P(D \mid G)\, P(G)$, then there is at least one data set D which leads to optimum grammars $\hat{G}$ which yield more than one derivation of some string in D.

We shall now establish some properties of grammars that are obtained by splitting. These properties, along with the concept of the search limit, are used in the algorithm described in Section 4. To illustrate the splitting of a grammar consider the following example.

EXAMPLE 2. Let $G = \langle\langle A_1, A_2\rangle, \{a, b\}, P, A_1\rangle$ where P consists of the productions

$$A_1 \to bA_1, \quad A_1 \to aA_2, \quad A_1 \to a, \quad A_2 \to bA_2, \quad A_2 \to aA_1.$$

Then

$$\begin{aligned} Q(A_1 \to bA_1) &= \{A_1 \to bA_1, A_1 \to bA_3, A_3 \to bA_1, A_3 \to bA_3\}, \\ Q(A_1 \to aA_2) &= \{A_1 \to aA_2, A_3 \to aA_2\}, \\ Q(A_1 \to a) &= \{A_1 \to a, A_3 \to a\}, \\ Q(A_2 \to bA_2) &= \{A_2 \to bA_2\}, \\ Q(A_2 \to aA_1) &= \{A_2 \to aA_1, A_2 \to aA_3\}, \end{aligned}$$

and G has as one of its splits

$$G' = \langle\langle A_1, A_2, A_3\rangle, \{a, b\}, P', A_1\rangle$$

where P' consists of the productions

$$A_1 \to bA_3, \quad A_1 \to aA_2, \quad A_2 \to bA_2, \quad A_2 \to aA_1,$$
$$A_3 \to bA_3, \quad A_3 \to aA_2, \quad A_3 \to a.$$

It can be seen that for all $p \in P$ $Q(p) \cap P' \neq \varnothing$ and for all $q \in P'$ there exists a $p \in P$ such that $q \in Q(p)$. Note also that the language of G is $(b^*ab^*a)^*b^*a$, and the language of G' is $(b^*ab^*a)^*b^+a$, which is a subset of the language of G. ∎

First we note that the split of a grammar generates a subset of the language of the grammar.

LEMMA 4. *If G and G' are base grammars such that G' is a split of G, then $L(G') \subseteq L(G)$.*

Proof. This is straightforward, and is omitted. ∎

Next we show that if a reduced grammar is a split of some grammar, then that grammar is reduced also.

LEMMA 5. *Let D be a data set and let $G = \langle V_N, V_T, P, A_1\rangle$ and $G' = \langle V_{N+1}, V_T, P', A_1\rangle$ be base grammars. If G' is a split of G and G' is reduced wrt D, then G is reduced wrt D.*

Proof. Assume G' is reduced wrt D. Let p be any production in P. By the definition of a split, $Q(p) \cap P' \neq \varnothing$. Let $q \in Q(p) \cap P'$ and $A_1 \Rightarrow a_1A_{i_1} \overset{*}{\Rightarrow} a_1 \cdots a_jA_{i_j} \overset{*}{\Rightarrow} a_i \cdots a_l = \alpha$ be a derivation of α in G' which makes use of q. Then there is a derivation of α in G

$$A_1 \Rightarrow a_1B_{i_1} \overset{*}{\Rightarrow} a_1 \cdots a_j B_{i_j} \overset{*}{\Rightarrow} a_1 \cdots a_l = \alpha \quad \text{where } B_{i_k} = A_{i_k} \text{ if } i_k \neq N+1$$

and $B_{i_k} = A_1$ if $i_k = N+1$. If q is the production $A_{i_{j-1}} \to a_jA_{i_j}$ then production p is $B_{i_{j-1}} \to a_jB_{i_j}$ and there exists a derivation of a string $\alpha \in L(D)$ which makes use of p. So G is reduced wrt D. ∎

We can now prove a stronger statement about a grammar and its split.

LEMMA 6. *If G' is a split of G and G' is deductively acceptable (DA) then G is DA.*

Proof. Immediate from Lemmas 4 and 5, and the definition of DA. ∎

Next we extend the result to a grammar and all of its splits.

LEMMA 7. *If $G' \in splits\ (G)$ and G' is DA then G is DA.*

Proof. A straightforward induction, using Lemma 6 on the number of nonterminals in G'. It is omitted. ∎

We shall now show that any grammar with more than one nonterminal is the split of exactly one grammar. Note that we have defined a SRG to have an ordered set of nonterminals. So two grammars that differ only in the renaming of their nonterminals are considered to be different.

LEMMA 8. *If $G' = \langle V_{N+1}, V_T, P', A_1\rangle$ is a base grammar, then there exists exactly one base grammar $G = \langle V_N, V_T, P, A_1\rangle$ such that G' is a split of G.*

Proof. By the definition of split, it is clear that, for any grammar that has more than one nonterminal, there is at least one merge. Suppose that $G_1 = \langle V_N, V_T, P_1, A_1\rangle$, $G_2 = \langle V_N, V_T, P_2, A_1\rangle$, $P_1 \neq P_2$, and that G' is a split of both G_1 and G_2. Since $P_1 \neq P_2$ we may assume without loss of generality that there is a production $p \in P_1 - P_2$. By the definition of split, since G' is a split of G_1 and $p \in P_1$ we have $Q(p) \cap P' \neq \varnothing$. Let $q \in Q(p) \cap P'$. By the definition of split if $q \in P'$ then there exists a $p' \in P_2$ such that $q \in Q(p')$. $Q(p)$ is such that if $q \in Q(p')$ and $q \in Q(p)$ then $p = p'$. So $p \in P_2$, which is a contradiction of the assumption. ∎

Next we extend this result to the case that G' is obtained by repeated splitting of a grammar with one nonterminal.

LEMMA 9. *If $G' = \langle V_N, V_T, P', A_1\rangle$ is a base grammar with $N > 1$, then there is exactly one grammar of the form $G = \langle\langle A_1\rangle, V_T, P, A_1\rangle$ such that $G' \in Splits\ (G)$.*

Proof. This is by a straightforward induction on the number of nonterminals of G', using Lemma 8, and is omitted. ∎

We shall use Lemmas 7 and 9 to show that all the grammars for D which we need to consider can be obtained by splitting G_u, the universal grammar for D. We shall also need the following result.

LEMMA 10. *Let D be a data set. Let $G_u = \langle\langle A_1\rangle, V_T, P_u, A_1\rangle$ be the universal grammar for D and let $G = \langle\langle A_1\rangle, V_T, P, A_1\rangle$ be any 1-nonterminal grammar such that $P_u \neq P$. Then G is not DA.*

Proof. By the definition of the universal grammar for D we have $E_D = \{a \mid A_1 \to aA_1 \in P_u\}$ and $F_D = \{a \mid A_1 \to a \in P_u\}$. Define $E_G = \{a \mid A_1 \to aA_1 \in P\}$ and $F_G = \{a \mid A_1 \to a \in P\}$. Since $P \neq P_u$ then either $E_D \neq E_G$ or $F_D \neq F_G$. If $F_D \neq F_G$ then there exists an $a \in V_T$ such that $a \in F_D$, $a \notin F_G$ or $a \notin F_D$, $a \in F_G$.

If $a \in F_D$ and $a \notin F_G$ then there exists an $\alpha \in L(D)$ and $a\beta \in V_T^*$ such that $\alpha = \beta a$. Since $a \notin F_G$ there is no production $A_1 \to a$ in P so α cannot be derived in G. So G is not DA.

If $a \notin F_D$ and $a \in F_G$ then the production $A \to a$ in P is not used in any derivation of a string in $L(D)$ since $a \notin F_D$ and no string in $L(D)$ ends in a. So G is not reduced wrt D and G is not DA.

If $E_D \neq E_G$ then the proof is similar to the case where $F_D \neq F_G$, so G is not DA. ▮

THEOREM 5. *Let D be a data set, and let G_u be the universal grammar for D. If $G = \langle V_N, V_T, P, A_1 \rangle$ where $N > 1$, and G is* DA, *then $G \in$ splits (G_u).*

Proof. By Lemma 9 there exists exactly one 1-nonterminal grammar $G' = \langle \langle A_1 \rangle, V_T, P', A_1 \rangle$ such that $G \in$ splits (G'). Assume $P' \neq P_u$, the production set of G_u. Then by Lemma 10 G' is not DA. But by Lemma 7 since G is DA and $G \in$ splits (G'), then G' is DA, which is a contradiction. So $P' = P_u$ and G splits (G_u). ▮

In Section 4 we describe a grammatical inference algorithm which is based on these results and we give some of its properties.

4. A GRAMMAR CONSTRUCTION PROGRAM

Our algorithm, which we call COGRAM, consists of a main routine and a subroutine that splits grammars. The main routine begins by reading in a data set D, computing $P(G)$ and $P(D \mid G)$ for the tree grammar and for the universal grammar for D. The best of the two grammars is saved and the search limit for the universal grammar is computed. The search limit for the program is set to the minimum of the search limit for the universal grammar and the complexity of the tree grammar. The program next splits the universal grammar and stores all its deductively acceptable splits. Then in order of increasing complexity C, the program splits all base grammars it has stored that are of complexity C. As it searches its store for these grammars, all base grammars of complexity less than C have weights assigned to the productions of the grammar to construct all SRG's with complexity equal to C that have the same productions as the base grammar. $P(G)\,P(D \mid G)$ is computed for each of these grammars and is compared to the current best grammar. If any one of these grammars is better than the current best grammar, then the search limit for the grammar is computed, and if it is lower than the current search limit, then the search limit is reset. This process continues until all DA base grammars of complexity less than the current search limit have been split and all SRG's with complexity less than or equal to this search limit are constructed and compared to the best grammar.

When the subroutine which splits grammars is called with the parameter G, the mainsplit of G is computed. The subroutine then finds the splits of G by enumerating the subgrammars of the mainsplit of G (see Property 1 below). If G' is a deductively acceptable split of G then $P(G')\,P(D \mid G')$ is computed and compared to the current best value. If $C(G') = C(G)$ then the subroutine is recursively called with parameter G' (see Property 2 below). If G' is not a deductively acceptable split of G then either G' is not a true subgrammar, or $L(D) \not\subseteq L(G')$, or G' is not reduced wrt D. If G' is not a true subgrammar then all of its subgrammars are skipped in the enumeration (see Property 3 below). If $L(D) \not\subseteq L(G')$ then all of the subgrammars of G' are skipped (see Property 4). If G' is not reduced wrt D, then the only subgrammars of G' that are enumerated are those grammars that do not have any of the productions in G' that are not used in any derivation in G' of a string in $L(D)$ (See Property 5). The subroutine continues until all the subgrammars of the mainsplit of G that were not skipped have been enumerated and stored.

In support of this algorithm we present the following five properties, whose proofs are straightfoward and are omitted.

Property 1. Let G be a grammar, and let G_M be the mainsplit of G. If G' is a split of G, then G' is a subgrammar of G_M.

Property 2. Given G, there are a finite number of grammars G' such that $G' \in \text{splits}\ (G)$ and $C(G') = C(G)$.

Property 3. If G, G', and G'' are such that G'' is a subgrammar of G' and G' is not a split of G, then G'' is not a split of G.

Property 4. If G' is a subgrammar of G then $L(G') \subseteq L(G)$.

Property 5. If G' is a subgrammar of G such that G' is reduced wrt D and G is not reduced wrt D, then any production in G that is not used in a derivation in G of some string in $L(D)$, does not occur in the production set of G'.

The operation of the successive bounding process in COGRAM is illustrated in Fig. 1, in which the ith SRG tested is plotted against $C(G_i)$ as the solid line, and the bound provided by the search limit is shown as a dotted line. The split of a given base grammar corresponds to a slanting line, while the assignment of weights to a base grammar corresponds to a horizontal segment.

COGRAM has been programmed in the language SITBOL, Gimpel (1972), on a PDP-10 computer, and results have been obtained for the following data sets:

Data Set 1. The data set from the proof of Theorem 3 was used, and the grammar G' in the proof of Theorem 3 was found. Total CPU time was 77 sec.

Data Set 2. The data set of Example 1 was used, and the ambiguous grammar G' in the proof of Theorem 4 was found. G' was found within 10 sec of CPU

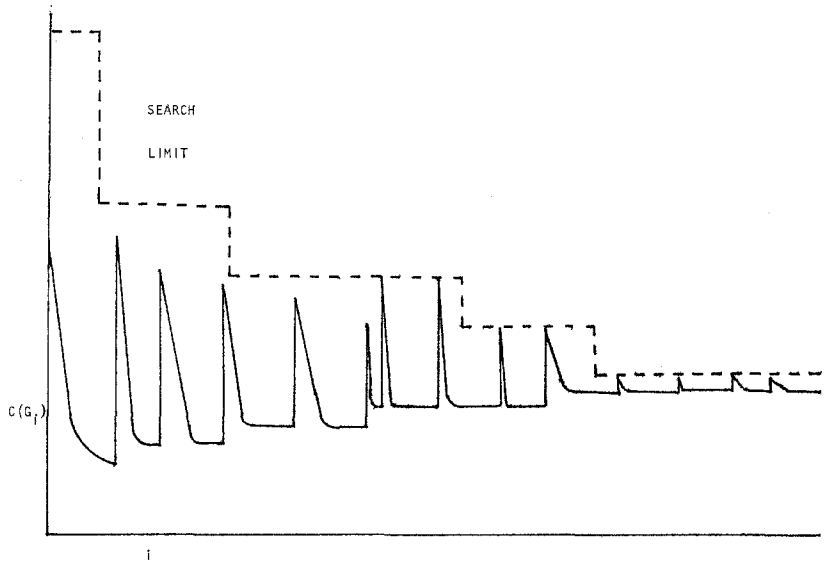

FIG. 1. Operation of the bounding process.

time, but since the search limit remained at a value of 5, a further 247 sec was needed to ensure that G' was indeed the best grammar.

Data Set 3. The three shortest strings in the set $\{ab\}^+$ were given as follows: $\{\langle ab, 4\rangle, \langle abab, 2\rangle, \langle ababab, 1\rangle\}$. The grammar $\{1/1: A_1 \to aA_2,\ 1/2: A_2 \to b,\ 1/2: A_2 \to bA_1\}$ was found within 11 sec and the search limit, which was initially 16 was reset to 4. Checking that this grammar was indeed the best grammar took a further 284 sec.

Data Set 4. The data set $\{\langle a, 7\rangle, \langle aaa, 4\rangle, \langle aaaaa, 2\rangle, \langle aaaaaaa, 1\rangle\}$ was a sample run in Horning (1969). COGRAM found the best grammar in the first 13 sec, and used another 106 sec to ensure that it was indeed the best. The best grammar had the production set $\{1/2: A_1 \to a,\ 1/2: A_1 \to aA_2,\ 1/1\ A_2 \to aA_1\}$.

Data Set 5. The data set $\{\langle b, 108\rangle, \langle bb, 18\rangle, \langle aa, 18\rangle, \langle baa, 5\rangle, \langle aba, 7\rangle, \langle baba, 1\rangle, \langle abba, 1\rangle\ \langle bbaba, 1\rangle\ \langle bbaa, 1\rangle\ \langle aabb, 1\rangle\}$ is based on an example of Horning (1969).COGRAM analyzed all 2-nonterminal grammars. These grammars were produced by splitting the universal grammar $\{1/4: A_1 \to a,\ 1/4: A_1 \to b,\ 1/4: A_1 \to aA_1,\ 1/4: A_1 \to bA_1\}$. The program was stopped after this split. The initial search limit was 183. The program produced the grammar $\{1/3: A_1 \to aA_2,\ 1/3: A_1 \to b,\ 1/3: A_1 \to bA_1,\ 1/4: A_2 \to a,\ 1/4: A_2 \to aA_2,\ 1/4: A_2 \to bA_1,\ 1/4: A_2 \to bA_2\}$ after 36 sec of run time, reducing the search limit to 132. The grammar $\{1/3: A_1 \to aA_2,\ 1/3: A_1 \to b,\ 1/3: A_1 \to bA_1,\ 1/3: A_2 \to a,\ 1/3: A_2 \to aA_1,\ 1/3: A_2 \to bA_2\}$ was found 42 sec later, reducing the search limit to 112. The total time to complete the analysis of all 2-non-

terminal grammars was 2151 sec, at which time the program was stopped. Thus it is possible that a best grammar with more than two nonterminals exists for this data set.

5. Conclusions

In this paper we have given some results about the inference of stochastic regular grammars. However, it is easy to see that some of the results in Section 3 that do not apply to splitting will also hold stochastic context-free grammars. The proofs of Lemmas 2 and 3, Theorems 1 and 2, and Corollary 1 can be easily adapted to the context-free case. We conjecture that Theorems 3 and 4 hold in the context-free case too.

We have found our inference procedure COGRAM useful for checking hypotheses. For instance, an earlier version of the program was used to assist in finding proofs of Theorems 3 and 4, and for testing the effect of Lemma 2. Apart from reprogramming in Fortran, which could be expected to yield a speed up factor of at least 10, there are three known techniques which could be incorporated into COGRAM to decrease its running time. First, we could structure the subgrammars of a grammar in a lattice, rather than a tree as at present, thus avoiding some checking of non-DA grammars. Second, we could seek a technique similar to that of Wharton (1977) to avoid checking grammars which are equivalent up to a renaming of nonterminals. Third, we could seek a direct way of computing a set of probabilities which, assigned to a given set of productions would yield the greatest possible value of $P(D \mid G)\, P(G)$ for those productions. Although these techniques would increase the experimental usefulness of COGRAM, it is unlikely that they alone would turn it into an applications program.

We see two main ways, within the present problem statement, of working toward the short running times required for applications. The first is to construct a good initial grammar algorithmically from the features (e.g., common substrings) of the data set. The second is to use heuristic methods for the same initialization task. An example of the first approach would be to seek a simple, fast algorithm which would produce a grammar whose value of $P(G)\, P(D \mid G)$ differs from that of a best grammar by at most a known amount.

However, there are different, but closely related, problem statements, in which extra information is given, which can result in shorter computations. Human interaction may be used to supply intermediate grammars, or to terminate the search when it is judged that a good enough grammar has been found. We can strengthen the data by providing instead of pairs $\langle \alpha_i , n_i \rangle$, quadruples $\langle A, \alpha_i , n_i , B \rangle$ where A and B are names of nonterminals. We can impose constraints on the shape of the graph corresponding to the SRG which is to be found. An application of these techniques in which such quadruples are available, and in which constraints are specified, is described in Walker (1977).

Clearly, it is always possible to strengthen the available information enough to obtain practical algorithms. However, progress in this area can be viewed as the process of requiring less information without requiring significantly more computing time. In this view, our free inference problem is a worst case in which very little information is available. Whether or not this worst case problem statement is inherently exponential in computing time requirement is an open question. But even if this is the case, the results in this paper should be useful in dealing with problems in which more information is available.

ACKNOWLEDGMENTS

We wish to thank T. Arbuckle and S. W. Ng for helpful discussions. This research was supported by Grant RR-643 to the Rutgers Research Resource on Computers in Biomedicine from the BRP, Division of Research Resources, National Institutes of Health.

RECEIVED: April 29, 1977; REVISED: September 12, 1977

REFERENCES

AMAREL, S. (1971), Representations and modelling in problems of program formation, *in* "Machine Intelligence 6" (Michie and Meltzer, Eds.), pp. 411–466, Univ. of Edinburgh Press.

BIERMANN, A. W., AND FELDMAN, J. A. (1973), A survey of results in grammatical inference, *in* "Frontiers of Pattern Recognition" (M. S. Watanabe, Ed.), pp. 31–54, Academic Press, New York.

BUCHANAN, B. G., FEIGENBAUM, E. A., AND SRIDHARAN, N. S. (1972), Heuristic theory formation: data interpretation and rule formation, *in* "Machine Intelligence 7" (Meltzer and Michie, Eds.), pp. 267–292, Univ. of Edinburgh Press.

DAVIS, R., BUCHANAN, B., AND SHORTLIFFE, E. (1977), Production rules as a representation for a knowledge-based consultation program, *Artificial Intelligence* **8**, 15.

DAVIS, R., AND KING, J. (1977), An overview of production systems, *in* "Machine Intelligence 8: Machine Representations of Knowledge" (Elcock and Michie, Eds.), pp. 300–331, Wiley, New York.

ELLIS, C. (1969), "Probabilistic Languages and Automata," Report No. 355, Dept. of Computer Sci., Univ. Illinois, Urbana.

FELDMAN, J. A., AND SHIELDS, P. C. (1977), Total complexity and the inference of best programs, *Math. Syst. Theor.*, **10**, 181.

FU, K. S., AND BOOTH, T. L. (1975), Grammatical inference: introduction and survey, parts I & II, *IEEE Trans. Systems, Man, and Cybernetics* **SMC-5**, 95.

GAINES, B. R. (1977), System identification, approximation, and complexity, *Internat. J. General Systems* **3**, 145.

GIMPEL, J. F. (1972), "SITBOL Version 1.0," Report S4D30, Bell Telephone Lab., Holmdel, N. J.

GOLD, M. (1967), Language identification in the limit, *Inform. Contr.* **10**, 447.

HOPCROFT, J. E., AND ULLMAN, J. D. (1969), "Formal Languages and Their Relation to Automata," Addison–Wesley, Reading, Mass.

HORNING, J. (1969), "A Study of Grammatical Inference," Report CS 139, Dept. Computer Sci., Stanford Univ.

SHORTLIFFE, E. (1976), "Computer Based Medical Consultations: MYCIN," American Elsevier, New York.

WALKER, A. (1977), "On the Induction of a Decision-Making System from a Data Base," Technical Report 80, Dept. Computer Sci., Rutgers University.

WATANABE, S. (1969), "Knowing and Guessing," Wiley, New York.

WATERMAN, D. A. (1977), Exemplary programming in RITA, *in* "Pattern-Directed Inference Systems" (Waterman and Hayes-Roth, Eds.), Academic Press, New York.

WEISS, S., KULIKOWSKI, C. A., AND SAFIR, A. (1978), Glaucoma consultation by computer, *Comput. Biol. Med.* **8**, 25.

WHARTON, R. M. (1977), Grammar enumeration and inference, *Inform. Contr.* **33**, 253.