Some Decidability Results on Grammatical Inference and Complexity

JEROME FELDMAN

Computer Science Department, Stanford University, Stanford, California 94305

The problem of grammatical inference is considered and a number of positive answers to decidability questions obtained. Conditions are prescribed under which it is possible for a machine to infer a grammar (or the best grammar) for even the general rewriting systems.

1. INTRODUCTION AND DEFINITIONS

The grammatical inference problem is easy to state: One must choose, on the basis of a finite set of symbol strings, a grammar for the language of which the given set is a sample. Precisely the same problem arises in trying to choose a model (theory, function, etc.) to explain a collection of sample data. This is one of the most important information processing problems known and it is surprising that there has been so little work on its formalization. The papers that have been written (e.g., Solomonoff, 1964) all discuss the importance of the problem so we will forego that discussion here.

Any attempt to formalize the grammatical inference problem must include precise formulations of several concepts left vague in the statement above. The four central notions are: the hypothesis space, the measure of adequacy, the rules by which samples are drawn, and the criterion for success in the limit of the inference process. For this paper, the hypothesis spaces will be subsets of the general rewriting systems expressed in a particular notation. A grammar will adequately represent a sample when it generates all of the known strings and none of the non-strings of the sample; in Section 3 we introduce an additional requirement of minimum complexity. For the presentation of sample strings, we require only that each string ultimately be included, (Horning, 1969) makes use of stronger requirements. The criterion for success in the limit of the grammatical inference process is one of the main subjects of this paper. In the remainder of this section we
present several distinct notions of success in the limit as well as further discussion of the other criteria mentioned above.

The results in this paper do not depend in any important way on the properties of grammars and could be developed in a more general setting. The presentation chosen here attempts to preserve the flavor of the original problem, while being sufficiently general to be easily adaptable to the problems of inferring functions, theories, etc. Within any specific domain there will be additional results attainable from the structure of the hypothesis space; Feldman et al. (1969) and Horning (1969, 1971) carry out such an extension for context-free grammars.

Our main interest is in efficient techniques for inferring grammars; the results of this paper are a foundation for the study of efficient inference methods. All of the theorems in this paper are for machines which enumerate all the possible grammars. Although there is a sense (Gold, 1967) in which no other method can be uniformly better than enumeration, we do not expect enumeration to compare favorably with other methods. Horning (1969) discusses several alternative inference techniques, including an enumerative technique which covers all feasible grammars but only looks at a small fraction of the total number of grammars in a class. Some preliminary results on heuristic methods of grammatical inference are described in Biermann and Feldman (1972).

The universal terminal alphabet $\mathcal{T}$ is the set of symbols $\{a, a_1, a_2, \ldots\}$. The universal variable alphabet $\mathcal{V}$ is the set of symbols $\{X = Z, Z_1, Z_2, \ldots\}$.

We will also use the following notational conventions. The string of zero symbols is denoted by $\epsilon$, the empty set by $\emptyset$. If $S$ is any set of symbols, $S^*$ is the set of finite strings of symbols from $S$, $S^+ = S^* - \epsilon$, and $S^n$ is the set of strings in $S^+$ of length $\leq n$.

A general rewriting system (grs, grammar) is a quadruple $G = (V, T, X, P)$ where $V, T$ are finite sets, $V \subseteq \mathcal{V} \cup \mathcal{T}$, $T = \mathcal{T} \cap V$, $X \in V \setminus T$ and $P$ is a finite set of productions (rules) of the form $u \rightarrow w$ with $u, w \in V^*$. We call $X$ the distinguished variable of the grs. If $G$ is a grs, $w, y \in V^*$ we write $w \Rightarrow y$ if there exists $z_1, z_2, u, v \in V^*$ such that $w = z_1 uz_2$, and $y = z_1 vz_2$ and the rule $u \rightarrow v$ appears in $P$. We will also say $y$ follows from $w$ in one production step. The transitive closure of $\Rightarrow$ is written $\Rightarrow^*$. The language $L(G)$ generated by $G$ is the set of strings, $y$, of $T^*$ for which $X \Rightarrow^* y$, i.e.,

$$L(G) = \{y \mid y \in T^* \text{ and } X \Rightarrow^* y\}.$$  

Since the $Z_i$ are dummy variables, one can assume that if $Z_i$ appears in $G$ then $Z_{i-1}$ does also without affecting the languages generable by a grs.
It is well known (Hopcroft and Ullman, 1969) that the grs generate exactly the recursively enumerable sets.

We will be interested in various decidability results concerning grs and some subclasses. A function will be said to be effective (computable) if there is an algorithm which carries out the mapping from its domain to its range. The notions of algorithm and decidability will not be made precise here; we will follow the practice of proof by Church's thesis (Rogers, 1968). This practice results from the widespread belief that any sufficiently well-specified process could be enunciated in terms of any of the several classes of formally equivalent machines. We will describe effective processes in terms of a finite-state control, unbounded memory, but otherwise unspecified machine, $M$.

A decidable rewriting system (drs), $G$, is a grs such that there is a machine $M$ capable of computing for any $y \in T^*$ the value of the predicate $y \in L(G)$. The primitive recursive functions, context-sensitive and context-free grammars (Hopcroft and Ullman, 1969) are all drs.

We define the length $l(G)$ of a grs $G$ to be the sum of the lengths of productions in our notation. There are, under our definitions, only a finite number of grs of a given length over a finite alphabet. A class $C$ of the grs is a subset which can be effectively enumerated in increasing order of $l(G)$.

Typical classes over a finite alphabet include all the grs, the context-free grammars, grammars in some standard form, and grammars with 3 variables, etc. Although it is not important here, one can think of a class of grammars as being the language generated by some grammar-grammar (Feldman et al., 1969).

There is an extension of the notion of ordered sequence which will be useful. A sequence $\langle y_1, y_2, \ldots \rangle$ is said to be approximately ordered by a function $f(y)$ iff there is a function $\tau_f(k)$ such that for each $k > 1$, $t > \tau_f(k)$ implies

$$f(y_t) \geq f(y_k).$$

If $\tau_f(k)$ is effectively computable then $\langle y_1, y_2, \ldots \rangle$ is said to be effectively approximately ordered (EAO) by $f$ and $f$ is said to be EAO by $\langle y_1, y_2, \ldots \rangle$. The EAO property is similar to one used in (Arbib, 1966) for proof measures and seems widely applicable. We will be content to present the basic property of EAO sequences which will be used in the sequel.

**Lemma 1.1.** For any sequence $\langle y_i \rangle$ which is EAO by a computable $f(y)$ there is a machine $M$ which will produce the sequence $\langle y_i \rangle$ in strict order of $f(y)$.

**Proof.** The machine $M$ proceeds as follows. It computes $f(y_1)$, finds
\(\tau(1)\) such that \(t > \tau(1)\) implies \(f(y_1) \geq f(y_t)\). It is then able to print all the \(y_t\) such that \(f(y_t) < f(y_1)\). \(M\) then computes \(f\) for all \(y_t, 2 \leq i \leq \tau(1) + 1\), finds the one \(y_j\), with lowest \(f\), not yet printed. It then computes \(\tau(j)\) such that \(t > \tau(j)\) implies \(f(y_1) \geq f(y_j)\), which enables it to print all \(y_i\) such that \(f(y_i) < f(y_j)\). \(M\) then finds the lowest, \(f(y_j)\), in the interval \(3 \leq i < \tau(2) + 1\) which has not yet been printed, and continues.

The EAO property is interesting because it is often much more natural and easy to establish than strict ordering, but for purposes of effective enumeration is just as powerful. We will be interested in enumerations of some class \(\mathcal{C}\) of grammars which will be ordered in particular ways.

Let \(\Sigma\) be the set of all finite sets contained in \(T^*\). We define a grammatical inference machine \(M_{q,\gamma} : \Sigma \rightarrow \mathcal{C}\) to be a function from sets in \(\Sigma\) to grammars in a class \(\mathcal{C}\), depending upon a complexity function \(\gamma\). We will sometimes suppress the subscripts on \(M\) when the class or complexity measure is understood. We will consider complexity measures in Section 3. The grammatical inference problem without complexity considerations is modelled as follows.

The machine \(M_q\) is presented, at each time \(t = 1, 2, \ldots\), with a string \(y_t \in T^*\) and guesses a grammar \(A_t = M_q(S_t)\) as a grammar for the set of strings of which \(S_t = \{y_1, \ldots, y_t\}\) is a sample. We will be interested in the machine’s guesses \(A_t\) and especially in the limiting behaviour as \(t \rightarrow \infty\).

More formally, an information sequence of a language \(L\), \(I(L)\) is a sequence of strings from the set

\[\{+y \mid y \in L\} \cup \{-y \mid y \in T^* - L\}.
\]

A positive information sequence \(I_+(L)\) is an information sequence of \(L\) containing only strings of the form \(+y\). An information sequence is complete if each string in \(T^*\) occurs in the sequence. A positive information sequence is complete if each string in \(L\) occurs in the sequence. We will be concerned only with complete information sequences. The set of all complete (positive) information sequences for all \(L \subseteq T^*\) is denoted \(\mathcal{J}(\mathcal{I}_+^\ast)\). Following Gold (1967), \(\mathcal{I}_+^\ast\) is called the set of text presentations and \(\mathcal{I}_+\) the set of informant presentations. Let \(I(L)\) be a (positive) information sequence of \(L\). We define a (positive) sample \(S_t(I)\) to be the unordered set: \(S_t(I) = \{\pm y_1, \ldots, \pm y_t\}\). A grammar \(G\) is said to be compatible with a sample \(S_t\) if \(+y \in S_t\) implies \(y \in L(G)\) and \(-y \in S_t\) implies \(y \in T^* - L(G)\). The set of strings of \(L(G)\) up to length \(j\), \(L(G) \cap T^j\), will be denoted \(L_j(G)\).

Consider a class \(\mathcal{C}\) of grammars and a machine \(M_q\). Suppose that for some \(G \in \mathcal{C}\), some \(I(L(G)) \in \mathcal{J}\) is chosen for presentation to the machine \(M\). We assume that \(M\) can form a guess \(A_t = M(S_t)\) at each presentation of
a string $y_t \in I(L(G))$ and define three successively weaker formalizations of the notion of the machine $M_\varphi$ learning the grammar $G$.

The machine $M$ is said to identify the grammar $G$ in the limit if there is a $\tau$ such that $t > \tau$ implies both $A_t = A_\tau$ and $L(A_\tau) = L(G)$. If $M$ is such that $t > \tau$ implies $L(A_t) = L(G)$ then $M$ is said to match $G$ in the limit.

Intuitively, $M$ identifies $G$ if it eventually guesses only one grammar and that grammar generates exactly $L(G)$. This does not imply that $M$ can effectively choose one grammar and stop considering new data. The machine $M$ will match the grammar $G$ if it eventually guesses only grammars of $L(G)$, albeit different ones. These notions are closely related and both require that $M$ find the correct language in finite time. We would also like to consider a weaker form of learning, in which the guesses $A_t$ would be ever better approximations to the grammar $G$. One idea is to put a metric on the space of grammars in $\mathcal{G}$ and use convergence as a criterion (Wharton, 1972). We use instead a notion of approachability, which is different from that used in our previous papers (Horning, 1969; Biermann and Feldman, 1972) and will require some discussion.

**Definition 1.2.** The machine $M$ is said to approach the grammar $G$ if the following two conditions hold:

(a) For any $y \in L(G)$ there is a time $\tau$ such that $t > \tau$ implies $y \in L(A_t)$.

(b) For any $H$ such that $L(H) - L(G) \neq \varnothing$ there is a time $\tau$ such that $t > \tau$ implies $A_t \neq H$.

$M$ will be said to strongly approach $G$ if the following additional condition holds.

(c) There is an $A$ such that $L(A) = L(G)$ and for an infinite number of $t$, $A_t = A$.

**Lemma 1.2.** If $M$ approaches $G$ then for any $H$ such that $L(G) \neq L(H)$, $A_t = H$ for only finitely many $t$.

**Proof.** If $L(G) - L(H) \neq \varnothing$ then there is a $y \in L(G) - L(H)$ and a time $\tau$ after which $y$ must be in $L(A_t)$ by condition (a) of Definition 1.2. Therefore $t > \tau$ implies $A_t \neq H$. If $L(H) - L(G) \neq \varnothing$ condition (b) states directly that there is a $\tau$ such that $t > \tau$ implies $A_t \neq H$.

This definition of approachability is asymmetric with regard to $L(G)$ and its complement. This asymmetry arises from the fact that there is no procedure for enumerating the complement of $L(G)$ for a grs $G$. One could
In addition to the limiting requirement of approaching $G$, we would like to require that the $A_t$ bear some resemblance to $G$. This requirement is met nicely by the complexity measures of Section 3. In the absence of complexity considerations condition (c) of Definition 1.2 forces $M$ to choose $A_t$ with some care. Thus a machine which chose $A_t$ so that $I(A_t) = S_t$ (guessed exactly the sample) could approach, but not strongly approach, a grammar.

2. INFERRING GRAMMARS

We will be interested in the conditions under which a machine $M$ can be expected to learn a grammar $G \in \mathcal{C}$, from successive samples $S_t$. Much of the early work on this problem was done by Gold (1967) in connection with his work on limiting recursion (Gold, 1965).

The main results of (Gold, 1967) deal with the great difference in learnability effected by allowing information sequences with negative instances, $I \in \mathcal{I}$ (informant presentation), rather than just positive instances, $I \in \mathcal{I}^+$ (text presentation). We will informally outline certain key proofs and then extend the results in various ways.

THEOREM 2.1 (Gold). For any class $\mathcal{C}$ of the dts there is a machine $M$ such that for any $G \in \mathcal{C}$ and any $I(L(G)) \in \mathcal{I}$, $G$ is identifiable in the limit by $M$.

Proof. The machine $M$ sequences through an enumeration $\mathcal{G}$ of $\mathcal{C}$. At each time, $t$, there is a first $G \in \mathcal{G}$ which is compatible with $S_t(I)$, it is the guess $A_t$ of $M$ at time $t$. Since $I$ contains each $y \in T^*$, any $A$ such that $L(A) \neq L(G)$ will eventually be incompatible with some $S_t$, either by generating a $-y \in S_t$ or failing to generate a $y \in S_t$. At some time $\tau$, $A_\tau$ will be such that $L(A_\tau) = L(G)$. Then $A_\tau$ will be compatible with the remainder of the information and will be the constant result of $M$.

Thus with informant presentation, a very wide class of grammars can be learned in the limit. By restricting the information to only $I \in \mathcal{I}^+$, we give up identifiability in the limit almost entirely. Let everything be as before except that the set of information sequences $\mathcal{I}^+ = \{I\}$ contains only sequences of the form $\langle +y_1, +y_2, \ldots \rangle$. 
Theorem 2.2. If $I(L(G))$ is restricted to $\mathcal{J}_+$ then any class $\mathcal{C}$ generating all finite languages and any one infinite language $L_\infty$ is not matchable in the limit.

Proof. We show that for any $M$, there is a sequence $I$, which will make $M$ change its value $A_i$ an infinite number of times for $L_\infty$. Since $M$ must infer all finite languages there is a sample which causes it to yield some $G(L_1)$ such that $L_1 \subseteq L_\infty$. Now consider an information sequence which then presents some string $x \in L_\infty - L_1$, repeatedly. At some time $t$, $M(S_t)$ must be a grammar of $L_1 \cup \{x\} = L_2$ because all finite languages are inferred. This construction can be repeated indefinitely, yielding an information sequence $I(L_\infty)$ which will pick a new value of $M$ an infinite number of times. Since each $L_i$ is finite, the machine $M$ chooses an $A_i$ such that $L(A_i) \neq L_\infty$ an infinite number of times.

It is possible, however, to construct a machine which will strongly approach the drs using only positive information sequences. The proof of this theorem is somewhat more complex than is strictly necessary in order to simplify some subsequent statements.

Theorem 2.3. For any class $\mathcal{C}$ of drs, there is a machine $M_\mathcal{C}(S)$ such that for any $G \in \mathcal{C}$ and $I(L(G)) \in \mathcal{J}_+$, $G$ is strongly approachable through $I$.

Proof. Let $\mathcal{G} = \langle G_1, G_2, \ldots \rangle$ be an enumeration of $\mathcal{C}$. The machine $M_\mathcal{G}$ will associate with each $G_i$ an integer $n_i$. Initially $n_i = i$ for all $i$. As the machine $M_\mathcal{G}$ carries out the algorithm described below some of these numbers will be increased. The machine $M_\mathcal{G}$ will also compute the value of a rapidly increasing function $f(u)$ called the bounding function, which it uses to estimate how much of $L(A_i)$ to consider at time $t$. The choice of $f(u)$ will be considered in Remark 2.1, one can think of it here as factorial $(u)$.

Algorithm for $M_\mathcal{G}$

Stage 1: Set $A_1 = G_1$. Go to Stage 2.

Stage $t$: Find the smallest value, $u$, such that $f(u) > t$. For each $G_i$, $i = 1, \ldots, t$, compute $m_i = \min(n_i, u)$. Look for the first $i \leq t$ such that $L_{m_i}(G_i) = S_t \cap T^{m_i}$ and $S_t \subseteq L(G_i)$.

The idea of the priority function $n_i$ and its use in the proofs of Theorems 2.3 and 2.7 originated with Manuel Blum. This allows us to prove stronger theorems on strong approachability than those described in earlier papers, e.g., Biermann and Feldman, 1972.
i.e., such that the language generated by $G_i$ agrees exactly with the sample up to length $m_i$ and such that the entire sample is generated by $G_i$.

**Case 1.** If no such $i \leq t$ exists set $A_t = A_{t-1}$ and continue to Stage $t + 1$.

**Case 2.** If such an $i \leq t$ is found, set $A_t = G_i$, set $n_t = n_{t+1}$ and go to Stage $t + 1$.

We now show that the three conditions (a)-(c) of Definition 1.2 for $M_\mathcal{Q}$ to strongly approach $G$ are satisfied.

Let $G_i$ be the first grammar in $\mathcal{G}$ such that $L(G_i) = L(G)$. We will show that $G_i = A_t$ for infinitely many $t$. Assume to the contrary that there is a maximum value $n_i$ attained by $n_t$. Let $K_1$ be the stage at which $n_i$ is attained. Let $K_2$ be the first time such that the $u$ required for $f(u) \geq K_2$ also obeys $u \geq n_i$. Let $K_3$ be the least $t$ such that $S_t \cap T_{n_i} = L_{n_i}(G) = L_{n_i}(G_i)$. For all $t > \max(K_1, K_2, K_3)$, $G_i$ meets the two conditions for choice by $M_\mathcal{Q}$ as $A_t$ under Case 2. If $G_i$ is never $A_t$ then there is some $G_j, j < i$ which is chosen in Case 2 as $A_t$ infinitely often. This already shows without appealing to the proofs of conditions (a) and (b) that Case 2 of the algorithm for $M_\mathcal{Q}$ is entered infinitely often.

If we have, in addition that $M_\mathcal{Q}$ approaches $G$ [conditions (a) and (b) hold] then we have derived a contradiction to the assumption that a finite $n_i$ exists. For by Lemma 1.2 any $H$ such that $L(H) \neq L(G)$ can be $A_t$ only finitely often and $G_i$ was chosen to be the first grammar $H$ such that $L(H) = L(G)$. Thus $n_i$ is not finite and $G_i = A_t$ for infinitely many $t$, assuming that (a) and (b) hold.

Conditions (a) and (b) of Definition 1.1 can now be established using the fact that Case 2 of the algorithm is entered infinitely often. Let $y \in L(G)$ and $K_4$ be the first time such that $y \in S_t$. For any $t > K_4$ in which Case 2 holds, $M_\mathcal{Q}$ will have chosen $A_t$ such that $y \in S_t \cap L(A_t)$ and condition (a) holds. To establish condition (b) we let $G_j$ be such that $L(G_j) - L(G) \neq \varphi$. We must show that $A_t = G_j$ for at most finitely many $t$. Let $y$ be the shortest string of $L(G_j) - L(G)$. Let $K_5$ be the first value of $t$ such that the $u$ required for $f(u) \geq K_5$ also satisfies $u \geq l(y)$. Consider any time $t > K_5$ such that Case 2 of the algorithm applies. If $j \geq K_5$ then for $t \geq K_5$, $A_t \neq G_j$ because $A_t$ must be exact to length $m_t \geq \min(j, u) \geq u \geq l(y)$. If $j < K_5$ then for $t > K_5$, $m_j = \min(n_j, u)$ can be $n_j \leq l(y)$, but each time $G_j = A_t$ by Case 2 $M_\mathcal{Q}$ increases $n_j$ by one. Therefore $G_j$ can be $A_t$ only until $n_j \geq l(y)$ which will occur in finite time.

**Corollary 2.4.** The machine $M_\mathcal{Q}$ will identify $G \in \mathcal{C}$ through any
For which \( M \) uses a bounding function \( f \) such that for all \( j \)

\[
f(j) > t > f(j-1) \implies L_j(G) \subseteq S_t.
\]

**Proof.** Let \( G_i \) be the first grammar in \( G \) such that \( L(G_i) = L(G) \). The condition above guarantees that \( G_i \) will always be an acceptable \( A_t \) \((t \geq i)\) using Case 2 of the algorithm for Theorem 2.3. Since any \( G_j, j < i \), is such that \( L(G_j) \neq L(G) \), \( G_j \) will be unacceptable after a time \( t_j \) as shown above. For all \( t > \max t_j \) over \( 1 \leq j < i \), the machine \( M_e \) will always chose \( G_i \) as \( A_t \) and thus identify \( G \).

**Remark 2.1.** The machine \( M_e \) can use a very large bounding function and identify the grammar for any sequence ordered by a smaller function. The problem is that such a machine will reject overbroad grammars \([H \text{ such that } L(G) \subseteq L(H)]\) later in time. Further, if \( M \) has chosen a bounding function which is too small, it will eventually discover that fact. There will appear some \( y_t \) such that \( f(I(y_t)) < t \). At this time \( M \) can switch to a larger bounding function. There is, however, no way to construct an \( M \) which will identify \( G \) through arbitrary information sequences, because for any computable enumeration of computable bounding functions there is function eventually larger than any of them (Hartmanis and Hopcroft, 1971). The corresponding question for program inference is more interesting and is discussed in (Feldman and Shields, 1972).

The machine \( M \) used in the proof of Theorem 2.3 could make use of negative strings to require \( A_t \) be compatible with the \( \pm \) sample. One might conjecture that there is a machine for the drs that would use negative strings in an information sequence without knowing whether or not it was complete (that is, whether all or only some of the negative strings occur) and achieve the behavior of Theorem 2.1 for complete (+ and --) sequences and of 2.3 for incomplete ones. This conjecture is false even for the finite state grammars.

The finite state grammars are a very restricted class of the drs whose standard form can be specified as follows. Each rule in \( P \) is either of the form \( Z_i \rightarrow a_k \) or of the form \( Z_i \rightarrow a_kZ_j \) with \( Z_i, Z_j \in V - T, a_k \in T \).

**Lemma 2.5.** If \( M \) is a machine which will approach any finite state grammar \( G \) for any \( I(L(G)) \in \mathcal{F}_+ \), then there is a finite state grammar \( H \) and an information sequence \( I(L(G)) \in \mathcal{F}_- \) which will cause \( M \) to guess incorrectly an infinite number of times, so that \( M \) will not be able to match the finite state grammars in the limit.
Proof. We form a subclass of the finite state grammars for which $M$ will change its guess, $A_t$, an infinite number of times. Let this class $\mathcal{C} = \{H_i\}$ be defined as follows:

$$L(H_0) = a^*b^* \quad \text{(any sequence of a's followed by any sequences of b's)}$$

and

$$\text{for } i > 0, \quad L(H_i) = \bigcup_{j=0}^{i} a^j b^*.$$

The languages $H_i$, $i \geq 0$ all have finite state grammars. We will show that for any $M$ which will approach all the $H_i$, $i > 0$ there is a complete $I(L(H_0)) \in \mathcal{F}$ which will cause $M$ to guess an $A$ such that $L(A) \neq L(H_0)$ an infinite number of times.

Suppose the positive strings of $L(H_0)$ are arranged as follows. I starts with enough strings in $L(H_1)$ to cause $M$ to guess some $A_t$ such that $L(A_t) = L(H_1)$. This must be possible because $M$ is assumed to approach $H_1$. Then I contains enough strings in $H_2$ to cause $L(A_t) = L(H_2)$, etc. The negative strings may appear arbitrarily in I subject to the restriction that all $-y \in T^* - L(H_i)$ appear after $H_j = A_t$ for some $t$. Since $M$ will guess incorrectly an infinite number of times, it will fail to match $H_0$.

This proof makes it clear that a machine (like that of 2.1) which attempts to identify grammars from $I \in \mathcal{F}$ will fail to even approach them if not all negative strings are present. Intuitively, the machine of Theorem 2.1 adopts a very conservative strategy; it chooses the first grammar which is compatible with the sample. It succeeds because the negative strings in a complete sample guarantee that any incorrect grammar will ultimately be incompatible. The machine of Theorem 2.3 does not have this guarantee, so it must constantly look for more suitable guesses.

Lemma 2.5 and Corollary 2.4 together show that an information sequence of known approximate order is better for inference than a complete positive and negative sequence.

The proof of Theorem 2.3 depends on the computability of the predicate "$y \in L(G)$" for every $G \in \mathcal{C}$ and thus cannot apply to the grs. By placing an additional restriction on $\mathcal{C}$ we can establish an alternative size measure which will enable a machine $M$ to strongly approach a class $\mathcal{C}$ of grs.

Definition 2.6. A class $\mathcal{C}$ is continuous if for every $I(L(G))$, $G \in \mathcal{C}$ and every sample $S_i$ the following condition holds. Let $m_i$ be the
length of the longest string in $S_t$, then there is a grammar $H \in \mathcal{C}$ such that

$$L_{m_t}(H) = S_t.$$ 

A continuous class is one which contains for each sample $S_t$, a grammar whose language up to $m_t$ is exactly that sample. Any class generating all the finite languages is trivially continuous.

**Theorem 2.7.** Let $\mathcal{C}$ be a continuous\(^2\) class of grs. Then there is a machine $M_\mathcal{C}$ such that for any $G \in \mathcal{C}$, $I(L(G)) \in \mathcal{A}_+$, $G$ is strongly approachable through $I$.

**Proof.** This will be similar to the proof of Theorem 2.3. Let $\mathcal{G} = \langle G_1, \ldots, G_n, \ldots \rangle$ be an enumeration of $\mathcal{C}$ and let $n_i$ be an index function as before. The machine $M_\mathcal{C}$ will employ a procedure called dovetailing: perform one derivation step with $G_1$, then one step each with $G_1$ and $G_2$ and so on. $K_n(G_i)$ will mean the set of terminal strings of length $\leq n$ derived so far from $G_i$.

**Algorithm for $M_\mathcal{C}$ at Stage $t$**

Continue dovetailing at least one more step and until finding the first $i$ such that

$$K_{n_i}(G_i) = S_t \cap T^{n_i}.$$ 

Set $A_t = G_i$ and increase $n_i$ by 1. Go to Stage $t + 1$.

The grammar $G_i$ sought at each stage in the algorithm will be found in finite time because of the hypothesized continuity of $\mathcal{C}$. As in the algorithm of Theorem 2.3, any grammar $G_j$ such that $L(G_j) \not= L(G)$ will be chosen as $A_t$ at most finitely often. This follows because any

$$y \in L(G_j) - L(G) \cup L(G) - L(G_j)$$

will eliminate $G_j$ after $n_j$ exceeds $l(y)$. Condition (b) of Definition 1.2 follows immediately. To establish (a), let $y \in L(G)$ and $K_1$ be the first $t$ such that $y \in S_t$. For any $G_j$, $j \geq l(y)$ we have $n_j \geq l(y)$ and (a) is satisfied whenever $G_j = A_t$. For $G_j$, $j < l(y)$ we still will have $n_j < l(y)$ and $G_j = A_t$ only finitely often.

Condition (c) follows from (a) and (b) as in Theorem 2.3. Let $G_t$ be

---

\(^2\) This condition can be dropped (Blum, 1972). However, the analogue to Corollary 2.4 is much weaker for the grs.
the first grammar in $\mathcal{G}$ such that $L(G_i) = L(G)$. If $n_i$ is unbounded, the proof is complete. If not, there is a finite maximum $n_i$ and a time after which

$$K_{\tilde{A}_i}(G_i) = S_i \cap T^{\tilde{A}_i}$$

always holds. If $G_i \neq A_i$ it is because some $G_j$, $j < i$ also meets this condition. But any $G_j$ such that $L(G_j) \neq L(G)$ can be guessed only finitely often (Lemma 1.2) and $G_i$ is the first grammar such that $L(G_i) = L(G)$.

**Remark 2.3.** The machines of 2.3, 2.4, 2.7 always chose the first grammar in $\mathcal{G}$ which met their conditions. One could choose an enumeration $\mathcal{G}$ of $\mathcal{G}$ which was ordered by size or intrinsic complexity (cf. Section 3) and have the machines of Theorems 2.3, 2.4, 2.7 choose the “best” grammars. We consider the question of inferring good grammars more carefully in the next section.

### 3. Inferring Good Grammars

The idea of requiring a grammatical inference machine to find a “best” grammar is a natural one and was studied informally by Solomonoff (1959, 1964). Once again, finding a best grammar is completely analogous to finding a best model, theory, function, etc. Any attempt to formalize the notion of “best” encounters a host of practical and philosophical problems; cf. Biermann and Feldman (1972). The purpose of this paper is to study, for a very general class of complexity functions, the conditions under which a machine can be expected to infer the best grammar. All results will hold for the general rewriting systems.

Let $\mathbb{R}$ be the non-negative rational numbers. We define a general grammatical complexity function $\gamma : \Sigma \times \mathcal{C} \rightarrow \mathbb{R}$ as follows:

**Definition 3.1.** A grammatical complexity function $\gamma_\mathcal{C}(S, G)$ satisfies the following conditions:

- **C1.** The function $\gamma$ is expressible as $\gamma(c(G, \mathcal{C}), d(S, G))$ and is a computable unbounded increasing function of each of its two arguments.

- **C2.** The intrinsic complexity $c(G, \mathcal{C})$ is a positive computable unbounded function EAO by $l(G)$.

- **C3.** The derivational complexity $d(S, G)$ is a positive function and is defined iff $S \subseteq L(G)$. We further require the existence of a computable function $D(S, G, m)$ which $= 0$ iff $d(S, G) \leq m$ and $= 1$ otherwise.
Remark 3.1. As an example, let the intrinsic complexity of $G$ be $l(G)$. Let the complexity $d(y, G)$ of a string $y \in S$, be the minimum number of steps needed to derive $y$ from $G$, and the complexity of a sample $S$ be the average over $y \in S$ of $d(y, G)/l(y)$. The complexity $\gamma$ could be the sum of the intrinsic and derivational complexity measures. This measure is similar to the ones used in (Feldman et al., 1969), but is somewhat too simple-minded, as the discussion there points out.

Remark 3.2. It is surprising that there is so little literature on the definition of a function for combining intrinsic and derivational complexity (Pager, 1970). The condition C1 seems to be a natural one to place on such a combined measure.

The conditions C2 and C3 are essentially the same as those used in the literature on complexity of computation; cf. Blum (1967a, 1967b). Instead of C2, Blum requires that there be an algorithm for computing the finite number of grammars with any fixed complexity. The condition C2 implies this for any class (as defined here) of grammars and is somewhat more specific about the algorithm. The recent survey (Hartmanis and Hopcroft, 1971) uses a condition slightly stronger than C2.

The notation used in defining $c, d$ emphasizes the relationship between complexity and probability, e.g., $c(G, \mathcal{C})$ is the complexity of $G$ given the class $\mathcal{C}$. Horning (Horning, 1969, 1971) bases his approach to grammatical inference on the probability measure $\pi\gamma$ in a Bayesian inference scheme.

We recall that the problem is for a machine $M$ to infer the best grammar $A$ of a class $\mathcal{C}$ from an information sequence $I(L(G))$, $G \in \mathcal{C}$. The machine $M_{\mathcal{C},\gamma}(S)$ will, as before, use an enumeration of the class $\mathcal{C}$, but will use more care in choosing the enumeration.

**Definition 3.2.** An Occam's enumeration $\mathcal{E}$ of a class $\mathcal{C}$ of grammars relative to an intrinsic complexity measure $c(G, \mathcal{C})$ is one which is strictly in order of intrinsic complexity, i.e., if $G_i, G_j \in \mathcal{E}$, $i > j$ then $c(G_i, \mathcal{C}) \geq c(G_j, \mathcal{C})$.

**Lemma 3.3.** For any class $\mathcal{C}$ of grammars there is an effective procedure for producing an Occam's enumeration of $\mathcal{C}$, relative to any intrinsic complexity satisfying Condition C2 of 3.1.

**Proof.** Immediate from the definition of a class $\mathcal{C}$, Lemma 1.1 and the requirement that $c(G, \mathcal{C})$ be an EAO function of the length of $G$.

All of usual classes of grammars over a finite terminal alphabet (grs,
grammars in some standard form, etc.) have effective Occam's enumerations. The following lemma and theorem show that the best grammar in a class for a finite sample can always be found.

**Lemma 3.4.** Let $\mathcal{G} = \langle G_1, G_2, \ldots \rangle$ be an Occam's enumeration of a class $\mathcal{C}$ and let $S$ be a sample of some $I(L(G))$, $G \in \mathcal{C}$. Let $i$ be the index of some $G_i \in \mathcal{G}$ for which $\gamma_i = \gamma(c(G_i, \mathcal{C}), d(S, G_i))$ is computable. Then there is a computable index $k$ such that $j > k$ implies $\gamma_j > \gamma_i$ or $\gamma_j$ is undefined.

**Proof.** If $\mathcal{C}$ is finite the result is trivial, assume $\mathcal{C}$ is infinite. Then $c(G_j, \mathcal{C})$ grows without bound as $j$ increases. Further, by C1 of 3.1, $\gamma(c(G_j, \mathcal{C}), 0)$ also grows without bound as $j$ increases. Let $k$ be the first index for which $\gamma(c(G_k, \mathcal{C}), 0) > \gamma_i$. Since $\mathcal{G}$ is an Occam's enumeration and $\gamma$ is an increasing function of $c(G_j, \mathcal{C})$, any $j > k$ has $\gamma(c(G_j, \mathcal{C}), 0) > \gamma_i$. Now $\gamma$ is also an increasing function of its second argument $d(S, G_j)$ which is $\geq 0$ if it is defined. Therefore $k$ satisfies the condition of the lemma.

**Remark 3.3.** The results of Section 2 were developed for arbitrary enumerations of the grammars in a class. These results could be strengthened in an obvious way for Occam's enumerations. For example, the machine of Theorem 2.1 could be made to identify in the limit the grammar of lowest intrinsic complexity.

Lemma 3.4 shows that a machine $M$ need only consider a finite number of grammars as candidates for the best match for some fixed sample, $S$. The proof is based on the fact that intrinsic complexity alone will rule out all grammars beyond some point in an Occam's enumeration. The next result shows that $M$ can always find the best grammar for a given finite sample.

**Theorem 3.5.** For any ordered class $\mathcal{C}$, there is a machine $M_\mathcal{C}$, which will find the best $A \in \mathcal{C}$ for any sample $S \subseteq I(L(G))$, $G \in \mathcal{C}$.

**Proof.** The machine $M$ will use an Occam's enumeration $\mathcal{G} = \langle G_1, G_2, \ldots \rangle$ of $\mathcal{C}$. It proceeds as follows to compute $A_i$. At each stage, $t$, $M$ adds at least one new grammar $G_j$ to $\mathcal{A}_t$ [the set of $G_i$ considered at stage $t$] proceeds to generate strings by dovetailing as in the proof of Theorem 2.7. After a derivational step is carried out for each grammar in $\mathcal{A}_t$, $G_{j+1}$ is added to $\mathcal{A}_t$ and dovetailing continues. It will occur, after a finite number of dovetailing steps, that a first $G_i \in \mathcal{A}_t$ is such that the sample $S$ is known to be contained in $L(G_i)$, i.e.,

$$S \subseteq K(G_i).$$
The grammar $G$ used as a base for $I(L(G))$ is one such grammar, but may not be the first.

The machine $M$ then computes $\gamma_i = \gamma(c(G_1, \mathcal{C}), d(S, G_j))$. It then uses the construction of Lemma 3.4 to find the $k$ such that $j > k$ implies $\gamma_j > \gamma_i$. $M$ must now choose the best grammar from $\mathcal{G}_k = \langle G_1, \ldots, G_k \rangle$. It next computes an integer $m$ such that $\gamma(c(G_1, \mathcal{C}), m) > \gamma_i$. Such an $m$ must exist and no grammar $G_j$ with $d(S, G_j) > m$ can have $\gamma_j < \gamma_i$.

$M$ now computes the function $D(S, G_j, m)$ of Definition 3.1 for each $1 \leq j \leq k$. If $D(S, G_j, m)$ is 1 then $G_j$ is not a candidate for the best grammar for $S$. However, $D(S, G_j, m) = 0$ for $G_j$ implies $d(S, G_j)$ is computable by C3 of 3.1. Now $M$ need only compute the value $\gamma_j$ for the finite number of $G_j$ for which $D(S, G_j, m) = 0$ and choose one with lowest $\gamma_j$. For specificity, $M$ will choose the first $G_j$ in $\mathcal{G}$ with minimum $\gamma(S, G_j)$.

Intuitively, the device $M$ tries to derive $S$ from the $G_j$ until it succeeds with some $G_j$. $M$ now has a lower bound $\gamma(c(G_1, \mathcal{C})$ on the intrinsic complexity of grammars in $\mathcal{C}$ and an upper bound $\gamma_i$ on the total complexity of its best guess so far. Since $\gamma$ increases without bound, there is some value, $m$, of derivational complexity which forces $\gamma$ to be greater than $\gamma_i$ for any $G \in \mathcal{C}$. $M$ now has an upper bound for $\gamma$ and needs to consider only a finite number of $G_j$ as contenders. $M$ cannot know, for grs, whether or not $G_j$ derives $S$. It can, however, establish a complexity bound beyond which $G_j$ will not be chosen in any case. Thus it can find the best grammar in a finite amount of time.

Remark 3.4. Theorem 3.5 was proved assuming that the information sequence was positive. If negative strings are permitted the machine $M$ should be made to reject a grammar (remove it from $\mathcal{G}_i$) when it is known to derive a negative string. Call this new machine $M^-$. We cannot require that the guesses $A_i$ made by $M^-$ be compatible with the sample $S_i$ and to be simultaneously of the lowest complexity. It is this limitation which motivated the assymmetry in the definition (1.2) of approachability. We will now show that $M^-$ can guess the best grammar at each step and still approach any grs through an informant presentation.

Theorem 3.6. For any ordered class $\mathcal{C}$ and any grammatical complexity $\gamma$, the machine $M^-$ has the following behaviour for any $I(L(G)) \in \mathcal{I}$, $G \in \mathcal{C}$. $M^-$ will for each sample $S_i$ choose a grammar $A_i \in \mathcal{C}$ which minimizes $\gamma(S_i, G)$ and will also approach $G$ as $t \to \infty$.

Proof. Theorem 3.5 establishes the first half of the condition on $M^-$. We must show that the guesses $A_i$ meet conditions (a) and (b) of Defini-
tion 1.2. For (a) we note that $A_t$ must generate all positive strings in $S_t$. If $L(H) - L(G) \neq \varphi$ then there is some $y \in L(H) - L(G)$ which will appear as $-y$ in $I(L(G))$. After the first time $\tau$ at which $H$ is known by $M^-$ to derive $y$, $H$ will never be considered as a candidate for $A_t$ and (b) is also satisfied.

**Remark 3.5.** Notice that the construction of Theorem 3.5 was not altered in proving the subsequent theorem. Thus the machine $M^-$ carrying out the strategy outlined there will do as well as conditions allow without knowing exactly which conditions hold. It is not possible to strengthen Theorem 3.5 to guarantee strong approachability. One can construct quite natural $\tau$, e.g., the $\gamma$ of Remark 3.1, and simple information sequences $I(L(G))$ for which the best grammar $A_t$ never has $L(A_t) = L(G)$. 

**Corollary 3.7.** If $\gamma(\mathbb{S}_1, A)$ is bounded as $t \to \infty$ for at least one $A \in \mathcal{C}$ such that $L(A) = L(G)$, then the machine $M^-$ will consider only a finite number of grammars in the entire process of inferring a grammar from $I(L(G))$ and will match $G$.

**Proof.** The construction of Theorem 3.5 allows $M^-$ to consider only a finite number of grammars past the first grammar which generates $S_t$. Let $A$ be the first grammar satisfying the hypothesis and let $b$ be the upper bound on $\gamma(S, A)$. For $t$ exceeding the first $\tau$ for which $A \in \mathcal{A}_\tau$, we know that $\gamma(S_t, A) \leq b$. For all but a finite number of grammars, $\gamma(c(G, \mathcal{C}), 0) > b$ so that intrinsic complexity alone will assure that $A_t \neq G$ for $t > \tau$. For each one of this finite number of grammars $G_j$ such that $L(G_j) \neq L(G)$ we have from the proof of Theorem 3.6 a finite time $\tau_j$ such that $t > \tau_j$ implies $A_t \neq G_j$. For $t > \max(\tau_j)$ we have $L(A_t) = L(G)$ and thus $M^-$ matches $G$.

**Corollary 3.8.** If $\gamma(\mathbb{S}_1, A)$ converges for all $A \in \mathcal{C}$ such that $L(A) = L(G)$ for each $I(L(G))$ then the machine $M^-$ will match $G$ and will eventually guess only $A$ which minimize the limit of $\gamma(\mathbb{S}_1, A)$.

**Proof.** From the results above we know that $M^-$ will match $G$. Any $A_t$ which converges to a limit $\gamma_t$ greater than the minimum $\gamma_m$ must eventually have $\gamma(\mathbb{S}_1, A_t) > \gamma_m + \epsilon$ for $\epsilon > 0$. The machine $M^-$ cannot identify a grammar $A$ because several grammars may cause $\gamma$ to converge to the same value at different rates.

There are a number of reasonable complexity measures which are bounded (Feldman et al., 1969). For example, the measure described in Remark 3.1 is bounded for any completely reduced context free grammar.
The problem of showing that a particular complexity measure converges is often quite difficult (Feldman et al., 1969). The problem can be made simpler by placing additional requirements on the information sequence. This seems to be an excellent area for the application of summability methods for generalized convergence.

The limiting behaviour of the machine $M^-$ was worked out in terms of complete negative information sequences, $I \in \mathcal{I}$. What can one say in the limit about inference from positive sequences, $I \in \mathcal{I}^+$? A machine which chose the best grammar (according to $\gamma$) at each step could not be assured of even approaching the correct grammar in the limit. A complete discussion would be beyond the scope of this paper (cf. Feldman et al., 1969) but the following example should be sufficient.

Suppose $G$ generated all strings of a's except the string of sixty-nine a's. From a positive information sequence, any reasonable device would prefer a grammar which generated all strings of a's. The machines of Theorems 2.3 and 2.7 were able to strongly approach $G$ through $I(L(G)) \in \mathcal{I}^+$ by using a bounding function. Any grammar which generated even one extra string would eventually be rejected. We can combine the results of Theorems 2.7 and 3.5 to get a machine $M^+$ which has good characteristics.

The machine $M^+$ will approach any $G \in \mathcal{C}$, for $\mathcal{C}$ a continuous class of grs, through any $I(L(G)) \in \mathcal{I}^+$. The guesses $A_t$ of $M^+$ may not be optimal by $\gamma$ but will generally be quite good. At each time $t$, $M^+$ uses the construction of Theorem 3.5 to find the grammar $B_t$ of lowest $\gamma(S_t, G)$. If $B_t$ also satisfies the condition of Theorem 2.7 ($B_t$ is known to generate no $y \in T^* - S_t$ of length $\leq m_t$) then $B_t$ is the guess $A_t$. If not, the machine $M^+$ tries the grammar of next lowest $\gamma(S_t, G)$. Since the class $\mathcal{C}$ is continuous, $A_t$ can be computed in a finite amount of time. The fact that $M^+$ approaches $G$ is immediate from the proof of Theorem 2.7. Thus this mixed strategy machine $M^+$ will approach the grs through a positive information sequence, making much better guesses than the machine of Theorem 2.7 but can not be shown to strongly approach $G$ or to always guess the best $A_t$ by $\gamma$. Nor are there analogies for Corollaries 3.6 and 3.7 because the correct grammar may fail to satisfy the cut-off condition of Theorem 2.7 infinitely often.

One could also define a mixed-strategy machines for the drs, choosing the best grammar by $\gamma$ meeting the conditions of Theorem 2.3. This machine will only approach (not strongly approach) the drs and the analogue to Corollary 2.4 establishes only that the machine will match (not identify) from bounded information sequences.

The totality of these results delimits the range of possibilities for grammatical inference with or without complexity, with complete or only positive
information and for general or decidable systems. One could sharpen these results by considering more specific cases, as has been done in (Feldman et al., 1969; Horning, 1969) for context-free grammars. The application of the general decidability results to program inference is presented in (Feldman and Shields, 1972).

Our main interest, however, is in practical methods of inference and these results provide a sufficient framework. Among the more interesting remaining theoretical questions are: inference in the presence of noise, general strategies for interactive presentation and the inference of systems with semantics.

Received: October 13, 1969

References


Blum, M. “Some Results in Learning and Inductive Inference,” Computer Science Department, University of California, Berkeley, May 1972.


PERRISON, S. (1966), Some sequence extrapolating programs, Stanford A.I. Memo No. 46.


WHARTON, R. W., Dissertation, Computer Science Department, University of Toronto, to be published.
