CS 139
STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AI-98

A STUDY OF GRAMMATICAL INFERENCE

by

JAMES JAY HORNING

TECHNICAL REPORT NO. CS 139
AUGUST 1969

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

A STUDY OF GRAMMATICAL INFERENCE

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE

AND THE COMMITTEE ON THE GRADUATE DIVISION

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

James Jay Horning

August 1969

STANFORD ARTIFICIAL INTELLIGENCE PROJECT
MEMO AI-98

AUGUST 1969

A STUDY OF GRAMMATICAL INFERENCE

by

James Jay Horning

ABSTRACT: The present study has been motivated by the twin goals of devising useful inference procedures and of demonstrating a sound formal basis for such procedures. The former has led to the rejection of formally simple solutions involving restrictions which are unreasonable in practice; the latter, to the rejection of heuristic "bags of tricks" whose performance is in general imponderable. Part I states the general grammatical inference problem for formal languages, reviews previous work, establishes definitions and notation, and states my position on evaluation measures. Part II is devoted to a solution for a particular class of grammatical inference problems, based on an assumed probabilistic structure. The fundamental results are contained in Chapter V; the remaining chapters discuss extensions and removal of restrictions. Part III covers a variety of related topics, none of which are treated in any depth.

The research reported here was supported in part by the Advanced Research Projects Agency of the Office of the Department of Defense (SD-183).

PREFACE

Grammatical inference is the process of discovering an acceptable grammar for a language, on the basis of finite samples from the language and is an interesting form of inductive inference. One of the principal tasks of linguists (and, it is widely believed, of children) is inferring grammars for natural languages. Yet remarkably little is known about either the actual methods used in grammatical inference or the possibilities and limitations of various inference techniques.

The grammatical inference problem was stated formally by Chomsky in 1957. Context-free grammars, which Chomsky introduced at the same time, were quickly adopted by computer scientists for the formal definition of programming languages, and by now their uses are legion. But the computing community (with the exception of Solomonoff) seems to have generally ignored grammatical inference for the next ten years, perhaps because of Chomsky's negative views on its solvability (reinforced by those of Shamir and Bar-Hillel). In 1967, Feldman and Gold proposed radically different solutions to the problem, and it is now starting to receive some of the attention that it deserves.

The present study has been motivated by the twin goals of devising useful inference procedures and of demonstrating a sound formal basis for such procedures. The former has led to the rejection of formally simple solutions involving restrictions which are unreasonable in practice; the latter, to the rejection of heuristic "bags of tricks" whose performance is in general imponderable. Part I states the general

grammatical inference problem for formal languages, reviews previous work, establishes definitions and notation, and states my position on evaluation measures. Part II is devoted to a solution for a particular class of grammatical inference problems, based on an assumed probabilistic structure. The fundamental results are contained in Chapter V; the remaining chapters discuss extensions and removal of restrictions. Part III covers a variety of related topics, none of which are treated in any depth.

I was originally introduced to the grammatical inference problem by Professor Jerome Feldman, and the present study was begun at his suggestion. Many of my results were derived to confirm or deny his conjectures, and I owe a great deal to his suggestions, his prodding, and his continuing interest. Thanks are also due to Professors David Huffman, William McKeeman, and William Miller for balanced doses of criticism and encouragement; to Rod Fredrickson and the Stanford Computation Center, Campus Facility, for financial support, computer time, and freedom to pursue the research reported here; to Stephen Reder for many stimulating discussions; to my wife for her patience; and especially to Phyllis Winkler for typing above and beyond the call of duty.

PART I

PRELIMINARIES

PART II

THE ENUMERATIVE BAYESEAN PROCEDURE FOR GRAMMATICAL INFERENCE

PART III

FURTHER CONSIDERATIONS

PART I

PRELIMINARIES

I. STATEMENT OF THE PROBLEM

Introduction

> The strongest requirement that could be placed on the relation between a theory of linguistic structure and particular grammars is that the theory must provide a practical and mechanical method for actually constructing the grammar, given a corpus of utterances. Let us say that such a theory provides us with a discovery procedure for grammars.
>
> [Chomsky 1957]

This study considers solutions to the problem of inferring a grammar for a language on the basis of finite samples from the language. For example, Feldman [1967], from the sample

b	baba
bb	abba
aa	bbaba
baa	bbaa
aba	aabb

inferred the grammar

S ::= b | bS | aA

A ::= a | bA | aS .

This grammar generates all the strings of the sample (and an infinite number of other strings), but it is not the only, nor even the shortest, grammar with this property. Finite samples do not uniquely determine particular grammars (or even particular languages) from the infinite

classes studied here; we cannot prove that any particular grammar is correct. Thus we are led from mathematical (demonstrative) reasoning to what has been called "plausible reasoning" [Polya 1954] or "non-demonstrative reasoning" [Nagel 1963]. Such methods will not yield answers which are certainly correct; all will sometimes infer grammars which further evidence might prove to be incorrect. Nevertheless, we are interested in developing and justifying particular methods as adequate and, under some conditions, optimal solutions to the problem of grammatical inference.

Since individual answers cannot be proven correct, it is important in each case to understand clearly and explicitly what problem is being solved, what is required of a solution to that problem, the assumptions under which a proposed solution is valid, and the relation of these assumptions to conditions which will obtain in potential applications. By making sufficiently strong assumptions we can make the grammatical inference problem formally trivial -- although perhaps still computationally laborious. Conversely, the problem can be formulated in such a fashion as to make the very existence of solutions doubtful. Potential applications lie at various points between these extremes, and no single solution is likely to satisfy all of them. Much of this study is devoted to identifying forms of the grammatical inference problem which are both solvable and useful; in a few cases we have "solutions looking for problems."

The classes of grammars treated in this study are subsets of the context-free grammars. In even the simplest case (finite-state grammars), however, we are dealing with an infinite set of grammars.

Furthermore, the set of possible samples is infinite. This means that formally simple operations which require the enumeration of all grammars or all samples must be excluded from computationally acceptable solutions. But it is not sufficient merely to show that each computation is finite. We cannot, for example, reasonably consider using algorithms[1] which require N^{N^N} or 10^{10^N} computations at the N-th step. Thus, an important part of this study is the consideration of practical bounds of applicability of the methods developed. In addition, we present the results of a computer implementation of one algorithm for grammatical inference.

Related Problems

> The grand aim of all science is to cover the greatest number of empirical facts by logical deduction from the smallest possible number of hypotheses or axioms.
>
> [Einstein]

Many aspects of intelligent behavior, whether "artificial" or "natural," involve plausible reasoning. One large class of problems stresses the recognition of regularities in data, and solutions are variously called "pattern classification," "concept formation," "data reduction," "learning," "explanation," or even "understanding." Another class is aimed at prediction of future observations or determination of the true state of nature, based on observational data; solutions may be called "extrapolation,"

---

[1] In accord with Church's thesis, we make no distinction among the terms algorithm, effective procedure, device, and computer program. Similarly, we use effective and computable as synonyms. [Cf. Davis 1958.]

"generalization," "statistical estimation," or "inductive inference." This division is based more upon divergence of viewpoint and emphasis than upon difference of technique -- we will later show that a certain inference technique can be justified either on the grounds of most efficient encodings of data or most probable estimates of the true grammar -- and there is no necessary conflict between them. They will lead, however, to rather different statements of the problem, and to rather different criteria for judging the results. In the first case, the concern is for the relation of the answer to the observed data: how well does it fit? how much can it explain? how efficient a representation does it provide? In the second, primary concern is for the expected relation of the answer to the true state; the observational data is important to the degree that it conveys information about that state.

Grammatical inference may be viewed either way. An inferred grammar is both a description of observed strings and a prediction of which other strings are of the same nature. This study incorporates approaches used by both schools of thought. Most of the borrowed results are not deep mathematically, since the principal difficulties in grammatical inference spring directly from the infinite, non-parametric hypothesis space, rather than from any mathematical complexities. We have most often used the following paradigm for insight into the inference problem: The inference device is confronted with a state of nature which is known (believed) to be represented by some one hypothesis out of a given set; the state of nature is not directly observable, but data which depends in some known fashion on the state

is available as an infinite sequence of observations; after each observation the device is to guess the state of nature.

We would certainly hope that our inference strategy would ultimately have an arbitrarily high probability of settling on the true state of nature. An optimal strategy is one that picks the true state more often than any other with the same information does. It never makes sense to infer at any time a hypothesis which can be deductively falsified by observations already made. We will later develop the viewpoint that an inductive inference device should generally be coupled to a deductive device which pre-screens the hypotheses for adequacy. Such tests could be built directly into the inductive device, but a separation appears desirable on two grounds: deductive inference, with corresponding search-limiting and pruning techniques is a well-developed field in itself, and as much as possible should be carried over intact; advanced deductive methods may be orders of magnitude more efficient than simple screening as an adjunct to inductive methods.

Why Infer Context-Free Grammars?

> Furthermore it [constituent-structure] is the only theory of grammar with any linguistic motivation which is sufficiently simple to permit serious abstract study.
>
> [Chomsky 1963]

There are two general justifications for studying the application of inductive methods to a particular problem: either the problem needs to be solved, i.e., some practical application is envisioned, or the solution appears formally interesting, i.e., some insight into either

the problem itself or into inductive methods is sought. In the case of grammatical inference, both apply. This study represents a step (certainly not the final one) in the development of procedures for currently known applications; these procedures appear to be applicable to more general problem areas.

Grammars are an interesting and powerful class of concepts. In the last dozen years context-free grammars have shown themselves useful in computer science, linguistics, and logic as definitions of formal languages, as hypotheses about -- or as generalizations of observations on -- natural languages, and as representations of complex structures. There are many situations in which grammars are known to be useful, but in which the relevant grammar is not known a priori, and consequently must be inferred from observations of the language. We present here two to which our study appears relevant.

Picture grammars: Two-dimensional pictures can be mapped into strings of picture elements, representing line segments, and picture operators, representing connectivity. Classes of pictures are thus mapped into sets of strings (i.e., languages) which are representable by grammars. Particular attention has been devoted to photographs of nuclear events recorded in bubble chambers; a direct correlation can be made between the reactions creating sets of pictures and the grammars representing them. A considerable amount of tedious analysis of these photographs could be avoided if there were a satisfactory program for inferring picture grammars. [Shaw 1968] [Miller and Shaw 1968] [George 1968].

Speech recognition: Some computerized speech recognition systems operate within the constraints of a fixed context-free grammar at the morpheme (or word) level. Nevertheless, they must adjust to the phonetic performance of individual speakers. Systems allowing sufficient a priori variation to handle all users generally do rather poorly for particular users. One reasonable representation for an individual speaker would be a set of low-level grammars representing his pronunciations of the words or phonemes occurring in the main grammar. For such a representation to be practical, there must be a means for the computer to learn these low-level grammars. [Reddy 1966] [Vicens 1969].

Grammatical inference also seems to be of about the right difficulty to be formally interesting.[1/] It is a non-trivial problem which forces us to realistically deal with infinite hypothesis and observation spaces for which we do not have any convenient finite parameterization. Yet we have enough structure to make the problem formally tractable. Grammars provide a well-defined characterization of sets of strings, and the notion of deductive consistency between hypothesis and data is precise and readily testable. A well-developed meta-theory is available, and many important properties of grammars have already been established. For experimental work we have at least

---

[1/] The task of the linguist attempting to discover grammars for natural languages is of course incredibly more complex than the one we have set ourselves. In the first place, the natural languages are much larger (richer) structures than those studied here; in the second, it is doubtful that context-free grammars (to which we limit ourselves), even of great size, are adequate models of natural language [Chomsky 1963]; finally, natural language learning is certainly complicated (although perhaps assisted) by questions of meaning (semantics) which we do not treat at all. It should be noted, however, that at least one important paper on which this study relies [Gold 1967] was motivated by an interest in the problems of a child (or a linguist) learning a language.

three controls on the magnitude of the problem: the class of grammars (e.g., from the hierarchy of finite state, linear, context-free), and the numbers of terminal and non-terminal symbols. We tend to feel that experience using grammars in a computer science context has led to a certain amount of insight into the subject matter which we could not bring to bear on an arbitrary form of inference. Humans perform reasonably well in simple forms of the grammatical inference problem[1] and some heuristic programs have been written [Feldman, et al. 1969]; thus, there are at least rough standards of comparison available.

Criteria for a Solution

> ...There are three main tasks in the kind of program for linguistic theory that we have suggested. First, it is necessary to state precisely (if possible, with operational, behavioral tests) the external criteria of adequacy for grammars. Second, we must characterize the form of grammars in a general and explicit way so that we can actually propose grammars of this form for particular languages. Third, we must analyze and define the notion of simplicity that we intend to use in choosing among grammars all of which are of the proper form.
>
> [Chomsky 1957]

So far we have left open the question of what constitutes a solution to the grammatical inference problem. In fact we have suggested that there is no single solution to the problem, since there are actually many forms of the problem. Most people probably approach the question with an implicit criterion something like

> The device always guesses the answer which I believe is the best answer based on its data.

Such a criterion poses what John McCarthy has called an "ill-formed problem," since it is difficult to determine when (or if) you have

---

[1] Cf. Chapter XI, pp. 156-160.

solved it. Artificial intelligence research often depends on finding well-formed approximations to ill-formed problems. In the present case, we have a well-formed problem when we have specified:

1) The hypothesis space, i.e., the class of grammars to be inferred.
2) The observation space, i.e., the form of the data and anything which is known about its structure.
3) The evaluation measure, i.e., an objective means of specifying the best hypothesis in a given situation.
4) The required performance, i.e., the criterion an acceptable solution must satisfy.

Having specified the first three items, we generally cannot set the fourth arbitrarily and still have a solution. Often we will have a fixed requirement, and vary some of the first items so that a solution is obtainable. Typical requirements are

a) The device must be able to infer a correct grammar for any language generated by a grammar in the class[1].
b) The device must (infer/approach) the correct grammar for (all/virtually all) valid sequences of observations and (stop at a finite time with the correct answer / settle on the correct answer at a finite time / reject each incorrect answer at a finite time / approach unit probability for the correct answer).

---

[1] It is generally neither possible nor desirable to strengthen this to the requirement that the same grammar from which the observations were derived be inferred.

c) The device must ultimately yield the best grammar for each language.

d) The device must (make a best guess / must minimize the probability of guessing an incorrect grammar) at each time.

Several investigations involving various combinations of the above alternatives have been reported and are summarized in the next chapter; we present results for other combinations in later chapters.

II. PREVIOUS WORK AND RESULTS

> One may arrive at a grammar by intuition, guess-work, all sorts of partial methodological hints, reliance on past experience, etc. It is no doubt possible to give an organized account of many useful procedures of analysis, but it is questionable whether these can be formulated rigorously, exhaustively and simply enough to qualify as a practical and mechanical discovery algorithm.
>
> [Chomsky 1957]

Constructive Methods

From the time that phrase structure grammars were proposed as models for natural languages there has been interest in the problem of discovering grammars for languages. Not surprisingly, the principal generalization that can be drawn from these studies is that the power of the inference device depends mostly on the assumptions that can be made about its environment.

Several methods have been proposed which can be classed as constructive, since sample strings are used systematically to construct the rules of a grammar. These present interesting heuristic approaches to grammatical inference, which may be useful in situations where a quick approximate (or reasonable) inference is more useful than a computationally laborious best inference. Some of them have been programmed with "subjectively reasonable" results [Feldman, et al. 1969]. However, they are not generally presented as solutions to well-formed problems, and it is not even clear that well-formed problems can be devised for which they would be optimal solutions. Since this approach

differs so substantially from our own, we will merely sketch two of the methods, without attempting proofs or a general development.

A method originally developed for finite-state grammars [Chomsky and Miller 1957] was generalized by Solomonoff [1959] for context-free grammars. It is based on the observation that most of the interesting structure of a grammar (and most of the difficulty in discovering the grammar) involves the recursive symbols[1] and the rules involved in the recursions. Consider some recursive symbol A, where

$$A \rightarrow^{+} \alpha A \beta \quad , \quad \text{with} \quad \alpha\beta \in V_t^{+}$$

Now if we have

$$S \rightarrow^{+} \delta A \gamma \rightarrow^{+} \delta\theta\gamma \quad , \quad \text{with} \quad \delta\theta\gamma \in V_t^{+}$$

we must also have

$$S \rightarrow^{+} \delta\alpha^{n}\theta\beta^{n}\gamma$$

for any $n \geq 0$. Solomonoff calls (α,β) a <u>cycle</u>, since the generation process can loop to place an arbitrary number of them in a sentence. His method depends on the determination of all the cycles in a language. Solomonoff states the problem as follows:

> Now suppose that we are given a large [enough to contain examples of all cycles] set of acceptable sentences from some ... language, and we are asked to discover <u>a</u> set of grammar rules that <u>could</u> generate the set. We are also

[1] Throughout this chapter we rely on the reader's prior knowledge of context-free grammars and related terminology. Some readers may wish to return to this chapter after reading our formal definitions in Chapter III.

> given a teacher. If we have any idea of what the grammar is, we are permitted to devise a trial string of symbols, and ask the teacher if it is an acceptable sentence. This is the only permissable type of question. [Emphasis added.]

This corresponds to a finite amount of text presentation followed by a responsive informant. Solomonoff does not mention a measure on the inferred grammar, but it appears that he wishes to infer a grammar which generates precisely the unknown language. For each partitioning of each string σ in the sample as $\sigma = \delta\alpha\theta\beta\gamma$ with $\alpha\beta \neq \lambda$ he questions the teacher about the validity of $\delta\theta\gamma$. If it is also an acceptable sentence, he hypothesizes (α,β) as a cycle, which he checks by asking about the validity of $\delta\alpha^2\theta\beta^2\gamma$, $\delta\alpha^3\theta\beta^3\gamma$, etc. "for enough cases of repetition to be reasonably certain that an arbitrary number of such repetitions would leave the sentence acceptable." Sentences without detectable cycles are basic, and are retained unchanged; cycle markers are inserted in the others at the points where cycles have been detected; the whole process is then repeated until no more cycles are detected. The cycles which have been obtained are analyzed in a similar manner.

> When, finally, cycles are found from which it is impossible to extract more cycles, we will have obtained what we call "basic cycles." We will be able to generate all the sentences in the language, when we have obtained an adequate set of basic sentences with all the cycle markers and basic cycles that may be attached to them, along with an adequate set of basic cycles with all of the cycle markers and basic cycles which may be attached to them.... It can ... be shown that there always exists at least one finite set of sentences with their cycle markers, and associated cycles with their cycle markers, etc., such that this finite set can generate the entire language.

As a last step, the basic cycle form can be converted to the more conventional form of context-free grammar, if it is desired.

There is no mention of a computer implementation of Solomonoff's method. A direct application of the procedure as described would encounter a number of problems. If the sample set were too small, not all recursions would be detected. However, large sample sets would result in an incredible barrage of questions to the teacher, many of which could have been answered by reference to strings already in the sample. Although Solomonoff does not touch on the issue of complexity in this paper, it appears that, in general, large, highly redundant grammars would be inferred; simplification procedures like those of Feldman and Gips (discussed below) may be in order. Finally, Shamir and Bar-Hillel [1962] challenge the adequacy of the method in principle for non-sequential context-free languages, or grammars with double cycles (e.g. $A \rightarrow^{+} \alpha A \beta$ and $A \rightarrow^{+} \delta A \gamma$).

A rather different method for inferring finite-state grammars is proposed by Feldman [1967] who states the problem as

> Suppose we are given the problem of finding a non-trivial finite-state grammar for [a set of strings] .... A non-trivial solution to a grammatical inference problem must generate an infinite number of strings and must not generate all the strings over the alphabet of the problem.

Feldman's method involves constructing an ad hoc non-recursive grammar which generates precisely the sample, then using residue analysis and merging to form a recursive grammar that generates the same strings plus an infinite number of others. The method is closely related to the covering grammars of Reynolds [1968]. Consider any grammar which generates all the strings in a finite sample. We can

construct a related non-recursive grammar, which also generates the sample, by replacing recursive occurrences of non-terminals by new non-terminals whose definitions are copies of the old, except that recursive occurrences are replaced by still other non-terminals, etc. Since all derivations are of finite length, this splitting process need only be repeated a finite number of times to produce a grammar which generates the complete sample. Furthermore, if any right part in the original grammar was used to derive the sample, either it or one of its copies will be used in the non-recursive grammar.

Reversing our viewpoint, any grammar which generates the sample covers an ad hoc grammar for the sample, and (if it has no unused right parts) can be constructed from the ad hoc grammar by repeatedly merging non-terminals, i.e., by replacing all occurrences of one non-terminal by the other and combining their definitions. But it is straightforward to construct the standard-form finite-state ad hoc grammar for any given sample. Feldman's strategy is to 1) construct the non-recursive grammar for the sample, 2) make the grammar recursive by merging each residue non-terminal (a non-terminal which produces only strings of length one or two) with the non-residue non-terminal which produces all the same strings "plus as few new (short) strings as possible," 3) simplify the grammar without further changing the language by merging equivalent non-terminals (non-terminals which have identical definitions if one is substituted for the other).

Gips has programmed Feldman's strategy, and sample computer runs are given by Feldman, et al. [1969]. The method does not generalize well

beyond finite-state grammars, because even linear languages are not uniquely deconcatenable -- thus, instead of one non-recursive grammar to be merged, there are many ad hoc grammars. Even if a deterministic merging algorithm is retained (and its rationale weakens as we get away from finite-state), the grammars derived from the various non-recursive grammars must be compared somehow. Feldman has proposed modifications of his strategy to meet these objections.

When Gips generalized his program to pivot grammars (a special form of operator grammars which lie between linear and general context-free grammars in power), he chose not to extend Feldman's method, but rather to use a simplification of Solomonoff's method, which would work without a teacher. He makes the following assumptions: all cycles are on a single non-terminal (which is the distinguished non-terminal unless all strings start or end with the same symbol); all cycles appear in the sample with both $n = 0$ and $n = 1$; and no fortuitous embeddings occur, i.e., $n = 0$ and $n = 1$ are sufficient evidence for a cycle. After cycle detection, he simplifies as before by merging equivalent non-terminals. Computer runs, a more complete discussion, and suggestions for improvement are given by Feldman, et al. [1969].

Enumerative Methods

The methods of the previous section are effective -- they can be programmed and they infer grammars. However, it is difficult to say in what (if any) sense they are optimal or produce a best grammar. Solomonoff's method is careful, by checking with the teacher, not to infer a grammar that generates too large a language, but if its sample

is too small it will not produce a grammar that generates the language that it is being taught. No weight is attached to the size of the grammar itself, and it will -- especially if the sample is large -- generally be very large and have many more rules than are necessary. Feldman's method, on the other hand, attempts to substantially reduce the number of rules without enlarging the language too much, but the trade-off is ill-defined. In neither case are we assured that their behavior will improve (i.e., that their answers will approach a grammar for the language, or that the probability of answering correctly grows) as the sample size increases.

Basically, the problem is that for a grammar to be best, it must be better than all other possible grammars. But the constructive methods have no way to compare the grammars they produce with all of those that they don't -- even if we were to supply them with a goodness measure. Thus there is no reason to believe that they can be easily modified to infer best grammars, by any reasonable criterion.

The most careful study related to grammatical inference which we have found in the literature is Gold's "Language Identification in the Limit" [1967]. Gold uses Turing machine programs rather than grammars as names for languages, and studies a somewhat different class of inference problems. The power of his results comes both from his precise statement of the problem and his use of enumerative methods which guarantee that no relevant answer will be missed. Even though we have studied different forms of the problem, we are indebted to him both as a model of clarity and a source of methods.

Gold is interested in languages, not grammars (or Turing machines). Consequently he uses a binary goodness measure: either a machine

generates the correct (although unknown) language or it does not. He also has a fixed requirement on performance, which he terms identifiability in the limit -- for any language [machine] in the hypothesis space, and any allowed presentation of that language in the observation space, there must be a finite time after which the inference device always yields the same answer, which is correct. Under these conditions, he is interested in the effect of varying the hypothesis space and the observation space on the solvability of the inference problem, i.e., determining the (language class, presentation method) pairs for which the inference problem is solvable.

Gold's principal result is the strong effect of text presentation (in which only valid strings are given) vs. informant presentation (in which both valid and invalid strings are identified) on the learnable classes of languages. With complete text presentation not even the finite-state languages are identifiable in the limit, while with complete informant presentation even the context-sensitive languages are identifiable.[1/] As an illustration that the order, as well as the form, of presentation is important, he proves that presentation by means of a primitive recursive function (anomalous text) is not only more powerful than complete text, but more powerful than any of the informant presentations considered.

Many of Gold's proofs involve a special form of guessing rule which he terms identification by enumeration, and defines as follows:

[1/] In the sequel when we prove identifiability in the limit with text presentation, it is with a different performance requirement, and a different condition on the presentation.

"Enumerate the class of objects [grammars] in any way, perhaps with repetitions.... At time t guess the unknown object to be the first object of the enumeration which agrees with the information received so far." He shows that for his definition of learnability there can be no other guessing rule which is uniformly better than any identification by enumeration rule. He does not discuss learning rate or the best choice at a particular time, except to note that this could only be meaningful if an a priori probability distribution were defined.

Several of Gold's results are extended in "Grammatical Complexity and Inference" [Feldman, et al. 1969]. The focus is on text presentation. By weakening the performance requirement from identifiability in the limit to approachability in the limit (each incorrect grammar is rejected at a finite time), a form of the grammatical inference problem is obtained which is solvable with text presentation for any admissable class of grammars (including context-sensitive, context-free, etc.).

Feldman, et al., also consider the question of learning a best grammar for a language, and early forms of some of our results occur in that paper. Goodness is equated with "least complex" and measures of complexity are developed both for grammars, and for sets of strings, given grammars. It is argued that a reasonable measure of the best grammar in a situation must involve both of these complexities. Effective identification in the limit of the best grammar from any complete informant presentation is proved for a restricted form of goodness measure. In accordance with Gold's result (text presentation is

inadequate to learn a correct grammar) it is shown that complete text presentation is inadequate to learn a best correct grammar with the goodness measure used (which will rate some incorrect grammars as better than any correct grammar).

The success of all the enumerative methods depends on the fact that each class of grammars in which we are likely to be interested (e.g., finite-state, context-free) is denumerable; the procedures are constructed so that at any finite time only a finite number of grammars (forming a prefix of the enumeration) need to be considered.

Inductive Methods

As Gold notes, to judge among hypotheses which are all consistent with a given sample requires at least an a priori probability distribution on the hypotheses and, to make this judgement a plausible function of the sample, conditional probabilities as well. In this study we use Bayesean methods presented by Watanabe [1960] and Solomonoff [1964] for inductive inference without necessarily sharing either of their world views or endorsing all of their conclusions.

"Information-Theoretical Aspects of Inductive and Deductive Inference" [Watanabe 1960] is motivated by the belief that

> A practical need will be felt more and more acutely in the future for a well-founded mathematical method of executing as much as possible of what is called inductive inference, including hypothesis testing.

The presentation begins with

> ... ten important features of inductive inference which any adequate theory of inductive inference should incorporate in some way or another, and which the present mathematical model indeed does. Admittedly, these ten conditions may not be sufficient but they are certainly necessary.

The bulk of the paper is devoted to the demonstration that Bayes' theorem -- with a priori and a posteriori "credibility" substituted for "probability" and "deductive probability" substituted for "conditional probability" -- actually meets Watanabe's ten conditions.[1] Although he requires "(7) Existence of law with objective validity" he seem reluctant to assign an objective meaning to the "deductive probabilities." He is also rather casual about the assignment of "a priori credibilities" (which "can even be altered in the middle of a series of experiments") since Bayes' theorem guarantees that "the ultimate conclusion will be free from the subjective pre-judgement."

Watanabe's results cannot be directly applied to the problem of grammatical inference, since he explicitly limits himself to finite hypothesis and observation spaces, and these limitations are essential to his development. However, his detailed discussion of the reasonableness of using Bayes' theorem for the assignment of "credibilities" to hypotheses -- even when objective probabilities are not assumed -- lends some support to the usefulness of this rule in inductive inference generally.

"A Formal Theory of Inductive Inference" [Solomonoff 1964] is a more ambitious treatise, and includes a discussion of grammatical inference.

> The problem dealt with is the extrapolation of a very long sequence of symbols -- presumably containing all of the information to be used in the induction. Almost all, if not all, problems in induction can be put in this form.
> .....
> Few rigorous results are presented.
> .....
> The third application, using phrase structure grammars, is least exact of the three. First a formal solution is presented. Though it appears to have certain deficiencies, it is hoped that presentation of this admittedly inadequate model will suggest acceptable improvements in it.

---

[1] These are plausible conditions, but not important to our development.

Solomonoff's viewpoint -- that inductive inference is merely a form of sequence extrapolation, and that a priori probabilities of sequences are determined by their likelihood of generation by an arbitrary universal (Turing) machine -- are not shared by this author. The acceptability of his reasoning depends on the acceptability of his premises:

> Suppose that all of the sensory observations of a human being since his birth were coded in some sort of uniform digital notation and written down as a long sequence of symbols. Then a model that accounts in an optimum manner for the creation of this string, including the interaction of the man with his environment, can be formed by supposing that the string was created as the output of a universal machine of random input.
>
> Here "random input" means that the input sequence is a Markov chain with the probability of each symbol being a function of only previous symbols in the finite past. The input alphabet may be any finite alphabet.
>
> .....
>
> By "optimum manner" it is meant that the model we are discussing is at least as good as any other model of the universe in accounting for the sequence in question.
>
> [Emphasis Solomonoff's]

However, we do share his conviction that

> It is possible to devise a complete theory of inductive inference using Bayes' theorem, if we are able to assign an a priori probability to every conceivable sequence of symbols.

We are also indebted to him for a number of key ideas, including the notions that stochastic grammars provide the appropriate means for associating conditional probabilities with strings, that the derivations of a grammar have an essentially simpler structure than their strings (and provide a useful encoding of the strings), that a priori probabilities may be assigned to grammars by determining the probabilities of their

irredundant written forms, and that ordinary grammars can be considered as stochastic grammars with the probabilities left as free parameters.

Solomonoff equates the problem of finding a grammar which "best fits" a given set of strings with the problem of finding the grammar which provides the best encoding in the following sense: the total probability of strings consisting of the grammar followed by derivations of the given set is maximal for that grammar. The probabilities of (grammar, derivation) strings are evaluated by considering each string as a set of interleaved Bernoulli sequences and applying approximations developed on the basis of three-tape Turing machines.

> In the previous section we had shown how to obtain a probability from a given set of strings, a PSG [grammar] that could have produced those strings, and a set of legal derivations of the strings from the grammar rules.
>
> From a formal point of view, this solves the problem of obtaining a PSG of optimum fit (i.e., highest probability), since we can order all PSG's and their derivations (if any) of the set of strings. ...
>
> This is not, however, a practical solution. The problem of finding a set of PSG's that "fits well" cannot be solved in any reasonable length of time by ordering all PSG's and sets of derivations, and testing each PSG in turn.

To remedy this problem, Solomonoff proposes "a method of digital 'Hill climbing'" starting from an ad hoc grammar and proceeding by a set of "mutations."

> At the present time, a set of mutation types has been devised that makes it possible to discover certain grammar types, but the effectiveness of these mutation types for more general kinds of grammars is uncertain.

Without knowing Solomonoff's "mutations," of course, it is difficult to judge their effectiveness; we conjecture that the merging and splitting rules mentioned in the section on constructive methods

would provide an adequate base. Of more serious consequence is the fact that Solomonoff ignores the question of how likely his measure is to prefer a grammar which is objectively correct. He also omits any consideration of whether the sequence of grammars selected by his measure will converge, and if so, whether the limiting choice will be correct.

In many ways Gold's work and Solomonoff's are complementary. Gold states a precise problem, and judges a potential solution on its limiting behavior -- requiring some correct answer but not discriminating among the correct answers nor worrying about how soon they are found. Solomonoff, in an ill-defined problem space, is concerned with the best answer on the basis of the current evidence -- that is, the most probable explanation, including both the likelihood of the explanation and its fit to the sample -- but does not worry about correctness (in any absolute sense) or limiting behavior. In succeeding sections we shall attempt to combine the strengths of both these approaches. Unfortunately, this is somewhat at the cost of adopting the complexities involved in each.

III. DEFINITIONS AND NOTATION

Rewriting Systems and Context-Free Grammars

In this chapter we establish definitions and notation which will be used throughout the sequel. Since the significance of some definitions will become apparent only in later chapters, the reader is urged to refer back here as he reads on. Although we establish basic results from the literature, nothing new is developed in this chapter. Where notations vary in the literature, we generally follow McKeeman [1966] or McKeeman, Horning, and Wortman [1970].

We assume familiarity with basic set theory. We denote sets by one or more capital letters, possibly subscripted $(A,\ldots,Z,V_t,PR,\ldots)$, or by explicitly naming the elements within braces ({ }) with conditions following the vertical bar (|) ; $\emptyset$ denotes the empty set; $\in$ denotes set membership; $\subset$ denotes set containment; $\cup$ denotes set union; $\cap$ denotes set intersection; - denotes set difference; and $\times$ denotes set product.

We also use the notation of predicate calculus. We denote logical "and" by $\wedge$; logical "or" by $\vee$; logical negation by $\neg$; logical equivalence by $\equiv$; and logical implication by $\supset$. $(\forall x)$ denotes "for all x" and $(\exists x)$ denotes "there exists an x."

Def. III.1. A vocabulary (or alphabet) is a finite set of elements called symbols. We denote symbols by letters (relying on context to distinguish them from sets).

Def. III.2. A string is a finite sequence of symbols from a vocabulary. We denote strings either by lower case Greek letters $(\alpha,\beta,\ldots,\omega)$ or by the juxtaposition of their symbols (e.g., if b , C , and d are symbols, bbb , bCd , and bCbCbC are strings).

Def. III.3. The empty string, denoted λ , is the sequence containing no symbols.

Def. III.4. The operation of catenation, denoted by the juxtaposition of strings or symbols, forms the string which consists of the successive sequences on which it operates (e.g., if α = bcd and β = dcb, then $\beta\alpha\beta$ = dcbbcddcb ; and $\alpha\lambda = \lambda\alpha = \alpha$).[1] If $\tau = \varphi\psi$, then φ is a head of τ and ψ is a tail of τ . We use α^n to denote n-fold catenation of α . Thus $\alpha^0 = \lambda$, $\alpha^n = \alpha\alpha^{n-1} = \alpha^{n-1}\alpha$ for $n > 0$.

Def. III.5. The length of a string τ , denoted $|\tau|$, is the number of symbols in the sequence. For any symbol b

$$|\lambda| = 0 \qquad |b| = 1$$

$$|b\tau| = |\tau b| = |\tau|+1$$

and

$$|\varphi\psi| = |\varphi| + |\psi| \quad .$$

If φ is a head of τ and $|\varphi| = n$, then φ is the n-head of τ , denoted $h_n(\tau)$. Likewise, if ψ is a tail of τ and $|\psi| = m$, then ψ is the m-tail of τ , denoted $t_m(\tau)$.

---

[1] Note that catenation is associative (although not commutative). This is the justification for not requiring an explicit catenation symbol or scope delimiter.

Def. III.6. For each vocabulary V, the set of strings over V, is denoted by V^*,[1]

$$V^* = \{\varphi \mid \varphi = \lambda \vee (\exists X \in V)(\exists \psi \in V^*)\varphi = \psi X\} .$$

Def. III.7. The set of non-empty strings over V is denoted by V^+,[2]

$$V^+ = \{\varphi \mid \varphi \in V \vee (\exists X \in V)(\exists \psi \in V^+)\varphi = \psi X\} .$$

Def. III.8. A rewriting system is an ordered pair $(V,\rightarrow)$ where V is a vocabulary and $\rightarrow$ is a relation on $V^* \times V^*$. For $\sigma,\tau \in V^*$, $\sigma \rightarrow \tau$ is read as σ directly produces τ and τ directly reduces to σ.

Def. III.9. If there exist strings $\varphi_0,\ldots,\varphi_n$ such that

$$\begin{aligned} &\varphi_0 \rightarrow \varphi_1 \\ &\varphi_1 \rightarrow \varphi_2 \\ &\quad\vdots \\ &\varphi_{n-1} \rightarrow \varphi_n \end{aligned}$$

for $n \geq 1$, then φ_0 produces φ_n and φ_n reduces to φ_0. This relation, denoted $\rightarrow^+$, is the transitive completion of $\rightarrow$, and we write $\varphi_0 \rightarrow^+ \varphi_n$.

---

[1] V^* is the free monoid generated by elements of V under the operation of catenation with λ as the identity.

[2] Note that $V^* = V^+ \cup \{\lambda\}$. V^+ is the free semigroup without identity generated by the symbols of V under the operation of catenation.

Def. III.10. The reflexive transitive completion of $\rightarrow$ is denoted by $\overset{*}{\rightarrow}$.

$$[\sigma \overset{*}{\rightarrow} \tau] \equiv [\sigma \overset{+}{\rightarrow} \tau \vee \sigma = \tau] .$$

Def. III.11. A contextual rewriting system is a rewriting system in which the relation $\rightarrow$ can be applied to substrings without regard for other symbols in the string, i.e., one for which

$$(\forall \sigma \epsilon V^*)(\forall \tau \epsilon V^*)([\sigma \rightarrow \tau] \supset (\forall \varphi \epsilon V^*)(\forall \psi \epsilon V^*)[\varphi\sigma\psi \rightarrow \varphi\tau\psi]) .$$

This is a strong restriction on $\rightarrow$; each pair of strings for which it is true implies arbitrarily many other pairs for which it is true.

Def. III.12. A context-free rewriting system is a contextual rewriting system for which $\rightarrow$ is completely specified by its values with a single symbol on the left, i.e., one for which

$$(\forall \sigma \epsilon V^*)(\forall \tau \epsilon V^*)([\sigma \rightarrow \tau] \supset (\exists A \epsilon V)(\exists \omega \epsilon V^*)(\exists \varphi \epsilon V^*)(\exists \psi \epsilon V^*)$$
$$[\sigma = \varphi A \psi \wedge \tau = \varphi \omega \psi \wedge A \rightarrow \omega]) .$$

Note that this further restriction on $\rightarrow$ again makes the rewriting system easier to specify.

Def. III.13. A string σ is terminal when it does not directly produce any string

$$\neg(\exists \tau)[\sigma \rightarrow \tau] .$$

In a context-free rewriting system, this implies that no symbol in

the string directly produces anything. Thus $\rightarrow$ partitions V into two subsets: the non-terminal vocabulary, denoted V_n, consists of symbols which can be rewritten; the terminal vocabulary, denoted V_t, of those which cannot.

$$V_n = \{A \mid (\exists \omega \epsilon V^*)[A \rightarrow \omega]\}$$

$$V_t = V - V_n = \{a \mid a \epsilon V \wedge (\forall \omega \epsilon V^*) \neg[a \rightarrow \omega]\} \quad .$$

V_t^* is the set of terminal strings. Note that, by definition, $V_n \cap V_t = \emptyset$.

Def. III.14. A context-free grammar[1] is an ordered quadruple $G = (V_n, V_t, \rightarrow, S)$ [2] where $(V_n \cup V_t, \rightarrow)$ is a context-free rewriting system with V_n as its non-terminal vocabulary and V_t as its terminal vocabulary, and $S \epsilon V_n$. S is the sentence symbol or distinguished non-terminal, and is the symbol which produces the language described by the grammar.

The following table summarizes notational conventions which we will use (possibly with subscripts) when referring to grammars:

Items	Symbols
members of V_t	a,b,c,...,z
members of V_n	A,B,C,...

---

[1] We often abbreviate "context-free grammar" to "grammar," where no confusion can result.

[2] We will write $\vec{G}$ rather than just $\rightarrow$ when G may not be clear from the context.

Items (continued)	Symbols (continued)
sentence symbol	S
arbitrary members of V	...,X,Y,Z
members of V_t^*	$\alpha,\beta,\gamma,\ldots$
members of V*	$\ldots,\varphi,\psi,\omega$
empty string	λ

Def. III.15. The sentential set of G , denoted SS(G) is the set of strings (sentential forms) produced by the sentence symbol.

$$SS(G) = \{\omega \mid S \rightarrow^* \omega\}$$

Def. III.16. The language of G , denoted L(G) , is the set of its terminal sentential forms (sentences).

$$L(G) = SS(G) \cap V_t^*$$

Grammars are weakly equivalent if they have the same language. A language is context-free iff it is the language of some context-free grammar.

Def. III.17. A derivation for a sentential form σ is a finite sequence $<\tau_0,\ldots,\tau_n>$ such that $\tau_0 = S$, $\tau_n = \sigma$, and $\tau_i \rightarrow \tau_{i+1}$ for $i = 0,1,\ldots,n-1$. Each pair (τ_i,τ_{i+1}) is a derivation step. By definition, every sentential form has at least one derivation.

Def. III.18. A derivation step is canonical iff it is of the form $(\varphi A\alpha,\varphi\omega\alpha)$, where $A \epsilon V_n$ and $\alpha \epsilon V_t^*$ and $A \rightarrow \omega$. A derivation is canonical iff each of its steps is canonical.

Every sentence has at least one canonical derivation. In fact, we can, without loss of generality, restrict our attention to canonical derivations, since for every derivation there is an equivalent canonical derivation which involves rewriting the same non-terminal symbols in the same way, but possibly in a different order [Ginsburg 1966].

Def. III.19. A sentential form is ambiguous with respect to G if it has more than one canonical derivation, unambiguous otherwise. G is ambiguous iff some sentential form is ambiguous with respect to G .

Ambiguity is one of the most intensively studied properties of context-free grammars. In most applications it is an undesirable property: an ambiguous sentence is assigned two distinct structures by the grammar, making interpretation unsure;[1/] when the grammar is used to assign codes to strings, ambiguous strings do not have unique codes. But ambiguity is an undecidable property. There is no effective procedure for determining whether an arbitrary context-free[2/] grammar is ambiguous [Chomsky 1963] [Ginsburg 1966]. The ambiguity of any string (hence any finite set of strings) with respect to a grammar is, however, decidable. In the sequel, we assume -- except where specifically

---

1/ Note, however, that natural languages are inherently ambiguous, and an adequate grammar for a natural language must reflect this ambiguity.

2/ Or meta-linear, or linear, or any other "interesting" subset of context-free except finite-state.

noted -- that unambiguous grammars are desired, and will reject grammars which make any sample string ambiguous.

Def. III.20. The set of productions of G, denoted $PR(G)$, is the set of ordered pairs of strings related by $\rightarrow$ where the left member is a single symbol.

$$PR(G) = \{(A,\omega) \mid A \rightarrow \omega, A \in V_n, \omega \in V^*\} .$$

Alternatively, we may consider the set of productions as basic, and derive $G(PR,S) = (V_n(PR), V_t(PR), \vec{PR}, S)$ from PR and S by the following definitions:

Def. III.21. $V_n(PR) = \{A \mid (\exists\omega)(A,\omega) \in PR\}$.

Def. III.22. $V_t(PR) = \{a \mid (\exists A)(\exists\varphi)(\exists\psi)(A,\varphi a\psi) \in PR\} - V_n$.

Def. III.23. $[\sigma \vec{PR} \tau] \equiv (\exists\varphi)(\exists A)(\exists\psi)(\exists\omega)[\sigma = \varphi A\psi \wedge \tau = \varphi\omega\psi \wedge (A,\omega) \in PR]$.

Various well-known classes of grammars are defined by restrictions on the forms of productions:

Def. III.24. A production is terminating iff it is of the form (A,α) for $\alpha \in V_t^*$; it is erasing iff it is of the form (A,λ) . Grammars with no erasing productions are λ-free.

Any context-free language not containing λ is generated by a λ-free grammar [Chomsky 1963] [Ginsburg 1966]. λ-free grammars are generally more convenient to handle. We restrict ourselves to λ-free grammars in the sequel.

Def. III.25. A production is linear iff it is of the form $(A,\alpha B\beta)$ for $\alpha,\beta\epsilon V_t^*$ and $B\epsilon V_n$. A linear production is left-linear iff $\beta = \lambda$, right-linear iff $\alpha = \lambda$. A grammar is (linear / left-linear / right-linear) iff all its productions are terminating or (linear / left-linear / right-linear). It is finite-state (regular) iff it is either left- or right-linear. The languages generated by linear grammars are linear languages; those generated by finite-state grammars are finite-state languages (regular sets).

Def. III.26. A production is in (Greibach) standard (ℓ-) form iff it is of the form $(A,a\omega)$ for $a\epsilon V_t$, $\omega\epsilon V_n^*$ (and $|\omega| \leq \ell$). A grammar is (ℓ-) standard iff all its productions are in (ℓ-) standard form. A standard grammar is an S-grammar iff for each pair of productions $(A,a\varphi)$ $(A,a\psi)$ either $\varphi = \psi$ or $\varphi = \lambda$ or $\psi = \lambda$.

Def. III.27. A production is in (Chomsky) normal form iff it is of the form (A,a) for $a\epsilon V_t$, or (A,BC) for $B,C\epsilon V_n$. A grammar is normal iff all its productions are in normal form.

Def. III.28. The non-terminal symbol A is recursive iff $A \rightarrow^+ \varphi A\psi$ for some $\varphi,\psi\epsilon V^*$. A grammar is recursive iff at least one of its non-terminal symbols is recursive.

Def. III.29. A grammar is connected if each non-terminal symbol occurs in some sentential form

$$(\forall A\epsilon V_n)(\exists\varphi\epsilon V^*)(\exists\psi\epsilon V^*)\ S \rightarrow^* \varphi A\psi \quad .$$

A grammar is non-blocking if each non-terminal symbol produces

some terminal string

$$(\forall A \epsilon V_n)(\exists \alpha \epsilon V_t^*)\ A \rightarrow^+ \alpha\ .$$

A grammar is reduced if it is connected and non-blocking.

For every non-reduced grammar there is an equivalent (in the sense of producing the same sentences with the same derivations) reduced grammar which can be formed merely by removing all productions involving disconnected or blocking non-terminal symbols. Since the reduced grammar has all the generative power of the non-reduced grammar, and is simpler by any reasonable measure, we will further restrict our attention to reduced grammars.

Def. III.30. A meta-language is a language, each of whose sentences (written grammars) specifies a grammar.

We use a variant of the BNF meta-language [Naur 1960] for our written grammars. Each grammar takes the form of a sequence of rules, each of which consists of a non-terminal symbol (the left part), followed by the meta-symbol ::= followed by the right part, which is a sequence of strings (alternatives) separated by the meta-symbol | . Each rule indicates that the relation $\rightarrow$ holds between the left part and each of its alternatives,[1] e.g., we interpret

$$A ::= \varphi \mid \psi \mid \ldots \mid \omega$$

as

[1] It is implicit that $A \rightarrow \omega$ is true for those pairs indicated by the written grammar, and for no others.

$$(A,\varphi)\epsilon PR\ ,\ (A,\psi)\epsilon PR\ ,\ \ldots\ ,\ (A,\omega)\epsilon PR\ .$$

A written grammar directly defines PR and, if we impose the convention that the left part of the first rule is the sentence symbol, defines G(PR,S) . Thus each written grammar specifies a particular context-free grammar. However, a single grammar may be specified by many distinct written grammars, since PR is an unordered set, but the rules and alternatives are necessarily written in some order. We will use written grammars extensively, without further remark, but we regard the underlying rewriting system as the more basic entity.

As usually interpreted, the rewriting systems defined by the context-free grammars of this section are permissive. Each defines a set of valid derivations (and thus a set of valid strings) without making any distinction among them. The predicate $f_G(\alpha) \equiv (S \overset{*}{\underset{G}{\rightarrow}} \alpha) \equiv \alpha\epsilon L(G)$ is a characteristic function [Davis 1958] of its language. Such a system fails to provide a direct counterpart to the intuitive notion that some sentences are more likely than others. In the next section we will develop an extension of context-free grammars (called stochastic grammars) to meet this difficulty. When we wish to distinguish the customary, permissive grammars developed in this section from those of the next, we will refer to them as characteristic grammars.

Stochastic Grammars and Probabilities of Sentences

We will now define a class of grammars and rewriting systems which not only specify languages but also provide probability distributions over the strings in their languages. These definitions are natural extensions of those in the previous section, and we will use the

same notation and conventions. In particular, we restrict derivations to canonical derivations and we specify grammars in terms of their productions.

Informally, at each step in a canonical derivation the rightmost non-terminal symbol is rewritten as one of the alternatives in the rule of which it is the left part. We may specify a derivation (and hence a string) in terms of the sequence of alternatives selected at successive steps. If the alternatives of each rule are numbered, the sequence of alternatives used in its derivation serves as a convenient digital encoding of a string. For example, consider the grammar G_1 :

```
S ::= T | S + T
          0     1
T ::= P | T * P
          0     1
P ::= a | ( S )
          0     1
```

The string a*(a*a + a) has the binary code 0111000100000 , which may be seen as follows:

sentential form	rightmost non-terminal	code digit	alternative
S	S	0	T
T	T	1	T*P
T*P	P	1	(S)
T*(S)	S	1	S+T
T*(S+T)	T	0	P

sentential form	rightmost non-terminal	code digit	alternative
T*(S+P)	P	0	a
T*(S+a)	S	0	T
T*(T+a)	T	1	T*P
T*(T*P+a)	P	0	a
T*(T*a+a)	T	0	P
T*(P*a+a)	P	0	a
T*(a*a+a)	T	0	P
P*(a*a+a)	P	0	a
a*(a*a+a)	none[1]		

In this example we have used only 13 bits of information to encode a 9-symbol string over a 5-character alphabet, whereas the obvious technique of using a unique binary code for each character would require at least 19 bits (plus some means of indicating its length) to encode the same string. In either case, of course, additional information is required to specify the grammar or the character codes, respectively. This is a question of some importance, which we treat at length in the sequel.

When sets of strings (e.g., arithmetic expressions in programs) are collected from users of a language, the valid sentences do not all occur with the same frequency. Although this result is tautological

---

[1] Note that we can unambiguously run together (catenate) these digital codes for strings and later separate (deconcatenate) them, since the end of a code is signalled by the derivation of a terminal string. The strings themselves, however, will not generally have this property of unique deconcatenability.

for infinite languages and finite samples, it is instructive to consider the pattern of the frequencies. In general, although the number of sentences of a given length is an increasing function of length, their frequency of occurrence is a decreasing function of length (and of length of derivation). This behavior may be modelled by assuming that each alternative has a fixed probability of selection whenever its rule is applied. The probability of a derivation is then just the product of the probabilities associated with the sequence of alternatives selected. Individual long derivations will generally be less probable than short ones, since each factor in the probability is less than one, and they have more factors.

We may extend our meta-language to indicate these probabilities by following each n-alternative rule with the n-tuple of its alternative probabilities, e.g., G_2 :

S ::= T | S + T	(2/3 , 1/3)
T ::= P | T * P	(1/2 , 1/2)
P ::= a | (S)	(3/4 , 1/4)

The string a*(a*a+a) has probability

$$\frac{2}{3} \times \frac{1}{2} \times \frac{1}{4} \times \frac{1}{3} \times \frac{1}{2} \times \frac{3}{4} \times \frac{2}{3} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} = \frac{3}{2^{13}}$$

with respect to G_2 , and the string a+a has probability

$$\frac{1}{3} \times \frac{1}{2} \times \frac{3}{4} \times \frac{2}{3} \times \frac{1}{2} \times \frac{3}{4} = \frac{1}{32} \quad .$$

Def. III.31. A stochastic production is an ordered triple, (A,ω,p) , where A is a symbol, ω is a string and p is a number

$0 < p \leq 1$. A stochastic rule for A is a set of stochastic productions, $\{(A,\omega_i,p_i) | i = 1,\ldots\}$, all having A (the left part) as the first element, distinct second elements (alternatives), and with the property that the third elements (alternative probabilities sum to unity. A stochastic grammar is a union of stochastic rules with distinct left parts, together with a sentence symbol which is the left part of some rule.

Def. III.32. For each stochastic grammar G , the corresponding characteristic grammar, denoted $\bar{G}$, is the grammar formed by deleting the alternative probabilities from each production. Defs. III.15-19, 24-29 extend to stochastic grammars in terms of their corresponding characteristic grammars (e.g., G is ambiguous iff $\bar{G}$ is ambiguous, linear iff $\bar{G}$ is linear).

Note that $\bar{G}_2 = G_1$.

Def. III.33. If (A,ω,p) is a stochastic production of G and $\varphi \epsilon V^*$, $\alpha \epsilon V_t^*$ then $\varphi A\alpha$ directly produces $\varphi\omega\alpha$ with probability p , denoted $\varphi A\alpha \underset{p}{\rightarrow} \varphi\omega A$. Every string produces itself with probability unity $\psi \overset{*}{\underset{1}{\rightarrow}} \psi$. If φ_o produces φ_1 with probability p_1 and φ_1 produces φ_2 with probability p_2 , then φ_o produces φ_2 with probability $p_1 \cdot p_2$.

$$(\forall\varphi_o)(\forall\varphi_1)(\forall\varphi_2)[\varphi_o \underset{p_1}{\overset{*}{\rightarrow}} \varphi_1 \wedge \varphi_1 \underset{p_2}{\rightarrow} \varphi_2] \supset [\varphi_o \underset{p_1 \cdot p_2}{\overset{*}{\rightarrow}} \varphi_2] \quad .$$

If S produces τ with probability p , then the probability of τ with respect to G , denoted $P(\tau|G)$, is p ; if S does not produce τ , the probability of τ with respect to G is zero.

Since every sentence has a unique[1] canonical derivation, it is produced with a definite, non-zero probability. It should be clear that (for non-blocking grammars) the probabilities of its sentences sum to unity, since each non-terminal sentential form directly produces a set of sentential forms which has an aggregate probability equal to its own probability, and the sentence symbol has probability one.

Def. III.34. Stochastic grammars are stochastically equivalent if they are weakly equivalent and assign the same probability to every string in their common language.

In at least some applications, the normative properties of stochastic grammars are advantageous. The nuclear physicist wishes to know not only which events are observed in pictures, but how probable (frequent) the various alternatives are. Similarly, a speech recognition system attempting to resolve phonetic ambiguities requires discrimination between probable and improbable (but still possible) interpretations of a sound. In any case, there seems to be no objection to inferring stochastic (rather than characteristic) grammars, provided that this assists in the inference. It is trivial to drop the alternative probabilities and obtain the corresponding characteristic grammar as a final step, if probabilities are not desired.

It is not always necessary or desirable to allow complete generality in the values of alternative probabilities. If we allow

---

[1] Recall that we are restricting ourselves to unambiguous (over the sample) grammars.

them to be arbitrary real numbers, our hypothesis space is enlarged from the countably infinite set of characteristic grammars to an uncountably infinite set. We will see in the sequel that countability is important for our inference techniques. A more immediate problem is that we have no way to write arbitrary real numbers. However, the rational numbers are dense in the reals and we can write any rational number as a fraction. We lose nothing in practice by restricting alternative probabilities to rational numbers, and we shall do so in the sequel when we require a countable hypothesis space.

It is often convenient to establish a one-to-one correspondence between the characteristic grammars and a subset of the stochastic grammars. The simplest assumption leading to such a correspondence is that all alternative probabilities in a rule are equal. This assumption works well for finite-state or linear grammars, and is the one implicit in most calculations of limiting entropy. We might expect that it would generalize well to context-free grammars. Pohl [1967], however, has shown that this is false; pathological results are obtained for simple recursive grammars of the sort used for arithmetic expressions in programming languages. For example, consider $\hat{G}_1$ obtained from G_1 by application of this rule:

S ::= T | S + T	(1/2 , 1/2)
T ::= P | T * P	(1/2 , 1/2)
P ::= a | (S)	(1/2 , 1/2)

Let $\hat{A}$ denote the weighted average length of strings produced from A . We see

$$\hat{S} = \frac{1}{2}\hat{T} + \frac{1}{2}(\hat{S} + 1 + \hat{T}) = \hat{T} + \frac{1}{2}\hat{S} + \frac{1}{2}$$

$$\hat{T} = \frac{1}{2}\hat{P} + \frac{1}{2}(\hat{T} + 1 + \hat{P}) = \hat{P} + \frac{1}{2}\hat{T} + \frac{1}{2}$$

$$\hat{P} = \frac{1}{2}(1) + \frac{1}{2}(1 + \hat{S} + 1) = \frac{1}{2}\hat{S} + \frac{3}{2} \quad .$$

We can solve these equations successively

$$\hat{S} = \hat{T} + \frac{1}{2}\hat{S} + \frac{1}{2} = 2\hat{T} + 1$$

$$\hat{T} = \hat{P} + \frac{1}{2}\hat{T} + \frac{1}{2} = 2\hat{P} + 1$$

$$= \hat{S} + 4$$

or

$$\hat{S} = 2\hat{S} + 9$$

There are no non-infinite positive solutions for $\hat{S}$. This corresponds to the fact that (with these probabilities) unboundedly long derivations do not have vanishing probabilities. For G_2 , however,

$$\hat{S} = 23$$

$$\hat{T} = 15$$

$$\hat{P} = 7 \quad .$$

An analysis similar to Pohl's shows that no method of assigning probabilities based solely on the form of the rules will yield only well-behaved (i.e., with finite expected length) stochastic grammars. If this result were limited to obscure or isolated instances, we could perhaps live with it; after all, no restriction based solely on the form of rules will yield only reduced grammars. The power of Pohl's result

springs from the fact that it applies directly to the best understood application of context-free grammars: grammars for arithmetic expressions in programming languages. A correspondence which fails in this context must surely be suspect in general.

A quite different approach which retains a one-to-one correspondence is to assume that the probabilities are not supplied with the grammar at all, but are parameters which must be learned after (or as) we identify the correct grammar, i.e., that each characteristic grammar is just a stochastic grammar form with the alternative probabilities missing. Parameter estimation is, of course, a well-known topic in statistical inference. We will show in Chapter VII that an inference procedure based on this assumption is not substantially more difficult than one in which the probabilities are assumed to be known a priori (e.g., equi-probable alternatives in a rule).

Enumerations and Orderings

In the sequel it will often be necessary to (at least formally) list the elements (i.e., grammars) of our hypothesis spaces in some order. This is possible for all finite sets and for countably infinite sets.

Def. III.35. Let X be a set. The sequence $E = \langle e_1, e_2, e_3, \ldots \rangle$ is an enumeration of X iff every element of X occurs in E, i.e., iff

$$(\forall x \in X)(\exists k > 0) e_k = x .$$

X is denumerable (countable) iff there is an enumeration for X.

Def. III.36. Let $f(k)$ be a function of one integer argument. f is a monotonic function if, for r some one of the relations $<$, $>$, $\leq$, $\geq$, and for each j and k, $j > k$ implies $f(j)\ r\ f(k)$; if r is $<$ or $>$, f is strictly monotonic.

Def. III.37. Let E be an enumeration and $g(e)$ a function over its elements. E is (strictly) ordered by g iff $f(k) = g(e_k)$ is (strictly) monotonic.

Def. III.38. Let $f(k)$ be a function of one integer argument. f is effectively approximately monotonic if for r some one of the relations $<$, $>$, $\leq$, $\geq$, there is a computable function $T(k)$ such that for any k and any $j > T(k)$

$$f(j)\ r\ f(k) .$$

Def. III.39. Let E be an enumeration and g(e) a function over its elements. E is effectively approximately ordered (EAO) by g if $f(k) = g(e_k)$ is effectively approximately monotonic.

We will use the EAO property extensively. From an EAO enumeration we can always effectively construct an ordered enumeration [Feldman 1969], but in practice this conversion is not usually required. The EAO property is often much easier to establish than ordering.

As an example, suppose that we are given some λ-free grammar and wish to enumerate its language. We may proceed as follows:

1) Let $SS_o = \langle S \rangle$, and $k = 0$.
2) For $i = 0,1,2,\ldots$ do step 3).
3) Set SS_i to the empty sequence. For each successive $\tau \epsilon SS_{i-1}$ do step 4).
4) Let A be the rightmost non-terminal symbol of τ. For each alternative ω in the rule for A, do step 5).
5) Let σ be the result of substituting ω for A in τ. If σ is terminal do step 6), otherwise do step 7).
6) Enumerate σ, i.e., set k to k+1 and e_k to σ.
7) Add σ to SS_i.

The i-th repetition of step 3 will cause all sentences derivable through i derivation steps to be enumerated, and all other sentential forms resulting from i derivation steps to be placed in SS_i. Let N be the number of rules in the grammar, M, the maximum number of alternatives in a rule, and L, the length of

the longest alternative. Then a string enumerated on the i-th repetition cannot be longer than $i \cdot L$; if the grammar is λ-free, reduced, and unambiguous it cannot be shorter than i/N .[1/] At most M^i strings will be enumerated on the i-th repetition. This enumeration is effectively approximately ordered by length, for if $|e_k| = \ell_k$, all strings of that length must be enumerated on or before $i = \ell_k \cdot N$. Thus at most

$$\sum_{i=1}^{\ell_k \cdot N} M^i = \frac{M^{\ell_k \cdot N+1} - M}{M-1}$$

strings will be enumerated before the last string of length $\leq \ell_k$. But ℓ_k is computable, and M and N are constants, so we may set

$$T(k) = \frac{M^{\ell_k \cdot N+1} - M}{M-1} .$$

This is not generally the best $T(k)$ that could be computed, but it is adequate to establish that the enumeration is EAO by length.

[1/] For a standard grammar the length is precisely i , for a normal grammar $(i+1)/2$ for odd i only. In these cases, the procedure results in an enumeration ordered by length.

Presentations

We turn now from the hypothesis space to the observation space. It is necessary to define precisely the allowable classes of observations. In much of the sequel we also need probability distributions over our observation spaces. The raw data of grammatical inference are strings, either indicated to be part of the unknown language or not.

Def. III.40. A positive instance of $L(G)$ is an ordered pair $(+,\sigma)$, where $\sigma \epsilon L(G)$. A negative instance of $L(G)$ is an ordered pair $(-,\sigma)$, where $\sigma \epsilon V_t^+ - L(G)$. An instance is a positive or negative instance.

Def. III.41. An information sequence of a language is a sequence $I = \langle I_1, I_2, \ldots \rangle$ of instances of the language. If each I_k is a positive instance, I is a positive information sequence. If $I_k = (\underset{-}{+}, \sigma_k)$ and the sequence $\langle \sigma_1, \sigma_2, \ldots \rangle$ is an enumeration of V_t^+, I is a complete information sequence. If the sequence $\langle \sigma_1, \sigma_2, \ldots \rangle$ is an enumeration of $L(G)$, I is a complete positive information sequence. If no instance is repeated, I is irredundant.

Def. III.42. A presentation of a language is a set of information sequences of the language. A presentation method is a mapping from languages into their presentations. A text presentation is restricted to complete positive information sequences; an informant presentation to complete information sequences; and an irredundant presentation to irredundant information sequences.

We have previously remarked [Chapter II] on the striking difference between text and informant presentations which was found by Gold. In the absence of probabilistic information, the requirement of completeness seems to be necessary for reliable inference. Consider, for example, the problem of inferring a grammar when strings of even length are systematically excluded from the information sequence.

We can not, of course, deal directly with infinite information sequences. But we can consider limiting behavior as successively larger subsequences are used.

Def. III.43. If I is the information sequence $\langle I_1, I_2, \ldots \rangle$ then $S_k(I) = \langle I_1, I_2, \ldots, I_k \rangle$ is a sample of size k.

For stochastic grammars and languages, we can impose a probabilistic structure on presentations.

Def. III.44. The stochastic text presentation of G is the infinite sequence $X = \langle X_1, X_2, \ldots \rangle$ of independent and identically distributed random variables (iidrv) with the distribution given by G, i.e.,

$$(\forall \sigma \in V_t^+) P(x_i = \sigma | X) = P(\sigma | G) \quad .$$

A stochastic sample of size k from a presentation consists of values for the first k random variables and has probability equal to the product of their individual probabilities. If

$$S_k = \langle \sigma_1, \sigma_2, \ldots, \sigma_k \rangle$$

then

$$P(S_t | G) = \prod_{i=1}^{k} P(\sigma_i | G) \quad .$$

We denote the set of all stochastic samples of size k for G by $S_k(G)$.

Individual stochastic samples may of course vary widely in their properties. We can show, however, that as $k \to \infty$ the relative frequencies of strings converge to the same limit for almost all samples.

Def. III.45. The frequency of τ in S_k denoted $f(\tau,S_k)$ is the number of times which τ occurs in the sequence $\langle\sigma_1.\sigma_2,\ldots,\sigma_k\rangle$.[1/] The relative frequency is $f(\tau,S_k)/k$.

We have the formal identity

$$P(S_k|G) = \prod_{i=1}^{l_k} P(\sigma_i|G) = \prod_{\tau\epsilon V_t^+} P(\tau|G)^{f(\tau,S_k)} \quad .$$

For any finite k , the infinite product is effectively computable, since only a finite number of factors differ from one. The stochastic presentation also corresponds formally to the infinite multinomial expansion

$$\Big(\sum_{\tau\epsilon V_t^+} P(\tau|G) \times \tau\Big)^k$$

where the coefficient associated with each product of strings is the

---

[1/] Formally, $f(\tau,S_k) = f(\tau,S_k,k)$ where $f(\tau,S_k,0) = 0$ and

$$f(\tau,S_k,j+1) = f(\tau,S_k,j) + \begin{cases} 0 & \text{if } \sigma_{j+1} \neq \tau \\ 1 & \text{if } \sigma_{j+1} = \tau \end{cases}$$

for $k \geq 0$, $j < k$.

probability of that collection of strings as a stochastic sample. We make use of this form to compute expectations over a presentation. In particular, if we can distinguish some τ, and are concerned only with functions involving it, the multinomial reduces to a binomial. For example, the expected value of $f(\tau,S_k)$ is

$$E[f(\tau,S_k)|G] = \sum_{f=0}^{k} f\cdot\binom{k}{f}\cdot P(\tau|G)^f[1-P(\tau|G)]^{k-f} \quad .$$

We may evaluate this sum by formally differentiating the identity $\sum_{f=0}^{k} \binom{k}{f}p^f q^{k-f} = (p+q)^k$ with respect to p

$$\frac{\partial}{\partial p}\left(\sum_{f=0}^{k} \binom{k}{f}p^f q^{k-f}\right) = \frac{\partial}{\partial p}(p+q)^k$$

$$\sum_{f=0}^{k} f\cdot\binom{k}{f}p^{f-1}q^{k-f} = k\cdot(p+q)^{k-1}$$

$$\sum_{f=0}^{k} f\cdot\binom{k}{f}p^f q^{k-f} = p\cdot k\cdot(p+q)^{k-1} \quad .$$

Now setting $p = P(\tau|G)$, $q = 1-p$

$$E[f(\tau,S_k)|G] = P(\tau|G)\cdot k \quad .$$

Dividing by k we find the expected relative frequency

$$E[f(\tau,S_k)/k|G] = P(\tau|G) \quad .$$

This says that, on the average, each string occurs with relative frequency equal to its probability. We can sharpen this result somewhat.

Lemma III.46. As $k \to \infty$ the relative frequencies of strings in stochastic samples S_k converge to their probabilities, with probability one (wpl). More precisely, given any $\epsilon > 0$, $\delta > 0$, there is a $T(\delta,\epsilon)$ such that for any $k > T$ the total probability of samples of size k such that $|f(\tau,S_k)/k - P(\tau|G)| \geq \delta$ is less than ϵ.

Proof. We use a method which will be important in the sequel. The total probability of a set is simply the expectation of its characteristic function, and we may bound this probability by taking the expectation of any function which is uniformly greater than or equal to the characteristic function. In the present case define

$$C_\tau(\delta,S_k) = \begin{cases} 1 & \text{if } |f(\tau,S_k)/k - P(\tau|G)| \geq \delta \\ 0 & \text{otherwise} \end{cases}$$

$$D_\tau(\delta,k) = E[C_\tau(\delta,S_k)|G]$$

$$= \sum_{f=0}^{k} C_\tau(\delta,S_k)\binom{k}{f}P(\tau|G)^f[1-P(\tau|G)]^{k-f} \quad .$$

It is not easy to place this sum in closed form. However, we can readily bound C_τ:

$$C_\tau(\delta,S_k) \leq [f(\tau,S_k)/k-P(\tau|G)]^2/\delta^2 \quad .$$

Thus

$$D_\tau(\delta,k) \leq \frac{E[f(\tau,S_k)^2|G]}{k^2\cdot\delta^2} - \frac{2\cdot E[f(\tau,S_k)\cdot P(\tau|G)|G]}{k\cdot\delta^2} + \frac{E[P(\tau|G)^2|G]}{\delta^2}$$

$$= \frac{E[f(\tau,S_k)^2|G]/k^2 - P(\tau|G)^2}{\delta^2}$$

But, by a repetition of our formal differentiation, we find

$$E[f(\tau,S_k)^2|G] = (k^2-k)P(\tau|G)^2 + k\cdot P(\tau|G)$$

so

$$D_\tau(\delta,k) \leq \frac{[(k^2-k)\cdot P(\tau|G)^2 + k\cdot P(\tau|G)]/k^2 - P(\tau|G)^2}{\delta^2}$$

$$= \frac{P(\tau|G) - P(\tau|G)^2}{k\cdot\delta^2}$$

$$\leq \frac{P(\tau|G)}{k\cdot\delta^2} \quad .$$

Now the aggregate probability of a union of sets is bounded by the sum of their aggregate probabilities.

$$D(\delta,k) \leq \sum_{\tau\epsilon V_t^+} D_\tau(\delta,k) \leq \sum_{\tau\epsilon V_t^+} \frac{P(\tau|G)}{k\cdot\delta^2} = \frac{1}{k\cdot\delta^2} \quad .$$

Finally, $D(\delta,k)$ is less than ϵ whenever k is greater than $\frac{1}{\epsilon\cdot\delta^2}$. Thus we may set $T(\delta,\epsilon) = \frac{1}{\epsilon\cdot\delta^2}$.

Q.E.D.

Def. III.47. An information sequence I is convergent iff each string has a limiting relative frequency equal to its probability, i.e.,

$$(\forall \tau \in V_t^+) \lim_{k \to \infty} f(\tau, S_k(I))/k = P(\tau|G) \quad .$$

In view of Lemma III.46, "almost all" information sequences in a stochastic presentation are convergent. Thus we can form a fair prediction of the effects of stochastic presentation by studying those of convergent information sequences.

BLANK PAGE

IV. EVALUATION MEASURES

"The most simple relations are the most common, and this is the foundation upon which induction rests."

[Laplace]

"Nevertheless, it should be fairly clear that under any reasonable definition of 'simplicity of grammar,' most of the decisions about relative complexity that we reach below will stand."

[Chomsky 1957]

Complexity and Probability

We now turn to measures for determining the best grammar in a situation. We have previously noted that a deductively falsifiable hypothesis should never be inferred. In the case of grammatical inference this means that only grammars which generate (at least) all the positive instances and none of the negative instances in the current sample are reasonable answers. We call such grammars deductively acceptable (DA). For stochastic grammars and presentations this condition is equivalent to the requirement that the grammar assign a non-zero probability to the current sample.

We present two motivations for the class of measures we use, one based on complexity and the other on probability. Either seems to provide an adequate basis for our methods, and the sequel does not depend on their conjectured equivalence, but some readers will probably find one or the other interpretation more satisfying.

The classical measure used to evaluate an inductive inference or generalization is its "simplicity." Of all the hypotheses which are consistent with our observations, the simplest is the preferred explanation. Practical application of this principle requires some effective means of evaluating simplicity (or complexity). But the complexity of an object is not independent of the context in which it is viewed. A single computer-generated display might be seen either as a simple algebraic equation, or as a complex series of vectors and arcs produced by the display hardware. Similarly the complexity attributed to

<list op> ::= c <seq>

<seq> ::= a <seq> | d <seq> | r

will depend on whether one views it (as a typist might) as an arbitrary sequence of characters or (as the reader probably does) as a grammar. In the latter case, the complexity will also be a function of one's familiarity with (and attitude towards) the BNF meta-language, on the class of grammars with which this grammar is being compared, and on the use for which the grammar is intended. We do not believe that it is meaningful to define an absolute complexity measure for grammars independent of these factors. When the unqualified phrase "the complexity of ..." is used, the qualifier "in the context ..." should always be implicit.

The measure of the complexity of an object in a context which we have chosen for this study is the minimum amount of information required to specify (or select) that particular object in the given

context. For example, the complexity of a derivation (in the context of a particular stochastic grammar) is the total amount of information required to specify the alternatives selected at each step. When applied in various situations this measure yields plausible results (e.g., it meets the requirements set by Feldman [1969] for a complexity measure); we conjecture that it corresponds closely to the intuitive notion of complexity. In this thesis we formally equate complexity and amount of information for a very specific reason: Shannon's definition of a precise measure corresponding to the intuitive use of "information" has led to a well-developed information theory, whose insights and methods we wish to apply.

The selection of an object in a context corresponds to the information-theoretic operation of transmitting a message from an ensemble of possible messages. The amount of information used to transmit the message depends on the encoding which is used. In the abstract, without knowing the use to which it will be put, we have very few grounds for judging any particular encoding method. We can require that it be complete (provide a code for each message in the ensemble), distinct (provide unique codes for each object), and irredundant (no code can be shortened without lengthening some other). But these constraints do not determine the minimum information required to transmit a particular message.

How much information is required to encode "the number one"? Twelve letters? Perhaps "the number" can be understood from context and three letters will suffice. If our code includes digits, "1" would be more compact. In the IBM System/360 the message is variously

transmitted using 4, 8, 12, 16, 24, 32, or 64 bits, depending on whether it is to be used as a register number, an operation code, a displacement, a half-word integer, an address, a full word or a double word number. No one of these answers is absolutely correct, independent of context.

A basic result of information theory is that the optimum -- in the sense of requiring the transmission of the smallest average amount of information -- encoding method for an ensemble of independently selected messages depends only on the probabilities with which the individual messages are transmitted. In addition, the information involved in optimally coding a particular message is the negative of the logarithm[1] of its probability. Thus highly probable messages have short codes, improbable ones, long codes. Since we have equated complexity with information, we interpret this to mean that objects (hypotheses, grammars) which are probable in a context should be considered less complex (in that context) than ones which are improbable. This is a sort of converse to Occam's Razor, which says that simple hypotheses should be considered more probable than complex ones. In the sequel we will implicitly assume that this relationship is valid and that complexity measures and probability measures are interconvertable[2]. This is mostly a matter

---

1/ Strictly speaking, is proportional to the logarithm to some fixed base. However, both the constant of proportionality and the base are customarily absorbed into the unit of information (the bit).

2/ In many real-life situations, of course, we do not have any effective means for assigning either probabilities or complexities to hypotheses. Consider the hypotheses "There is life on Mars," "There is life after death," and "There is life after 30." How complex are they? How probable? "The credence that we place in a conjecture is bound to depend on our whole background, on the whole scientific atmosphere of our time." [Polya 1954]. One advantage of using grammatical inference to study induction is that our hypothesis space can be defined objectively and we can define and study inference where objectively correct results are known.

of convenience, allowing us to use whichever terminology seems more natural to describe a situation, and our development is not dependent on this relationship.

Bayes' Theorem and Inference

It might seem that, given a complexity measure for hypotheses, a general solution to the inference problem would be to order the hypotheses by complexity and then, at each step, pick the first (i.e., simplest) deductively acceptable hypothesis. This simple rule is inadequate for some forms of grammatical inference, in particular, when text presentation[1] is used [cf. Gold 1967, Feldman et al. 1969]. All the "interesting" infinite classes of grammars (including the finite-state grammars) generate all the finite languages as well some infinite languages. Let $L(G_o)$ be an infinite language and $\{L(G_i) | i = 1,2,...\}$ be the (infinite) set of finite subsets of $L(G_o)$. Now G_o must occur at some finite point in the enumeration, hence only a finite number of grammars can precede (i.e., be simpler than) it. Let j be such that $L(G_j)$ has no grammar before G_o in the enumeration (there must be an infinite number of such j's). Now $L(G_j) \subset L(G_o)$ so G_o is DA whenever G_j is. Since G_o is simpler than G_j, G_j can never be inferred, even when it is correct (i.e., $I = I(G_j)$) and G_o is incorrect.

The method of the previous paragraph fails because it assumes that the complexity measure is independent of the sample. For positive samples there are two types of trivial DA grammars which represent opposite extremes. An ad hoc grammar produces precisely the current sample; a universal grammar produces every string over the terminal vocabulary. In general, these grammars do not represent acceptable

[1] Although we do not limit ourselves to text presentation (positive samples), we are particularly interested in conditions under which it is adequate for inference.

generalizations or inductions, and we would expect a satisfactory measure to rule them out. Informally, we see that ad hoc grammars fit the data very well, but at the expense of considerable complexity in the grammar; universal grammars can be quite simple, but will generally be a poor fit to the sample. To exclude these (and other similarly inappropriate) grammars a measure must be a function of both the grammar and its degree of fit. If it neglects the former it will sometimes select overly complex grammars which produce languages that are "too small" (insufficient generalization); if the latter, it will sometimes infer overly simple grammars which produce languages that are "too large" (excessive generalization).

The relation between complexity and probability suggests a method based on Bayes' theorem to refine probability estimates on the basis of observations. This "theorem" is actually an elementary consistency requirement on conditional probabilities [Savage 1962]. Suppose we have an exhaustive set of mutually exclusive hypotheses H_i, $i = 1,2,3,\ldots$ and a similar set of observable samples S_j, $j = 1,2,3,\ldots$. Denote the probability that the i-th hypothesis is true in context C by $P(H_i|C)$, the probability that the j-th sample is observed in context C by $P(S_j|C)$, and their joint probability by $P(H_i,S_j|C)$. Also denote the conditional probability of the j-th sample given that the i-th hypothesis is true by $P(S_j|H_i,C)$ and the converse by $P(H_i|S_j,C)$. For these measures to make sense we require

$$P(H_i,S_j|C) = P(H_i|C)\cdot P(S_j|H_i,C)$$

$$P(H_i,S_j|C) = P(S_j|C)\cdot P(H_i|S_j,C) \quad .$$

We can eliminate $P(H_i,S_j|C)$ and solve for one of the conditional probabilities, e.g.,

$$P(H_i|S_j,C) = \frac{P(H_i|C)\cdot P(S_j|H_i,C)}{P(S_j|C)} \quad .$$

But it must also be the case that exactly one hypothesis is true so

$$P(S_j|C) = \sum_i P(H_i,S_j|C) = \sum_i P(H_i|C)\cdot P(S_j|H_i,C) \quad .$$

Thus we have

$$P(H_i|S_j,C) = \frac{P(H_i|C)\cdot P(S_j|H_i,C)}{\sum_\ell P(H_\ell|C)\cdot P(S_j|H_\ell,C)} \quad .$$

This rule may be used to compute the a posteriori conditional probability of H_i when S_j is observed. It requires only the a priori probabilities of the hypotheses[1/] and the conditional probabilities of

---

1/ Bayesean techniques are sometimes criticized on the grounds that the required a priori probabilities are, in general, unknown, and the reader may feel that our suggestions for determination of a priori probabilities of grammars (given in the next section) are somewhat artificial. These criticisms have some merit, but we feel that the Bayesean viewpoint permits the most direct understanding of our methods, by providing intuitive meaning to what would otherwise be arbitrary formal operations. Thus much of the presentation will have a Bayesean flavor. The reader who finds this distasteful may take comfort in one or more of the following rationalizations:

1. We can obviously construct, for test purposes, situations in which both the a priori and conditional probabilities are precisely controlled.
2. In some of the envisioned applications (e.g., speech recognition, bubble chamber pictures) the a priori probabilities may actually be known from prior experience.
3. Each inference procedure will actually be presented with some (frequency) distribution. If the Bayesean procedure incorporates all the advance knowledge that we have about this distribution, then no other procedure can be constructed which will uniformly (or even on average) do better.
4. We show in Part II that our procedure will ultimately learn a correct grammar independent of the particular (non-zero) a priori probability assigned to the grammar. The worst that an incorrect assignment can do is delay the learning. However, without <u>some</u> assignment our procedure is not effective.

the observed sample, given the various hypotheses. If we must select one hypothesis on the basis of the sample S_j, we minimize our risk of error by choosing an H_k such that $P(H_k|S_j,C)$ is a maximum.[1]

For purposes of maximization, the value of the denominator $P(S_j|C)$ is irrelevant. What is important is to maximize the product $P(H_i|C)\cdot P(S_j|H_i,C)$. The minimum risk requirement, plus a knowledge of the probabilities involved leads to a precise identification[2] of the best hypothesis -- in our case the best grammar -- to be guessed on the basis of a sample.

We may interpret this solution in terms of complexity measures by taking logarithms. If $M(H_i|C)$ is the complexity of H_i in the context C and $M(S_j|H_i,C)$ is the complexity of the sample in the further context of H_i, then we are to minimize the sum

$$M(H_i|S_j,C) = M(H_i|C) + M(S_j|H_i,C) \quad .$$

This indicates that complexity of explanation has two components, which we may call the intrinsic complexity of the hypothesis and the relative complexity of the sample, given the hypothesis. In information-theoretic terms, we may express this as looking for that representation of the sample (i.e., hypothesis) which minimizes the combined information requirement of the representation and the encoded sample. The "representation problem" is a classical problem of artificial intelligence;

1/ This risk, precisely $1-P(H_k|S_j,C)$, is known as the Bayes' risk in statistical decision theory.

2/ Except, of course, when there is no unique maximum. We than have a set of equally good guesses.

here we give it a precise operational meaning by specifying the class of hypotheses over which we are to minimize.

Going one step further, we can treat this result as suggestive, even for complexity measures which are not motivated by information-theoretic or probabistic considerations. Feldman [1969] requires a complexity measure to be an unboundedly increasing function of two measures, one which indicates how complex the hypothesis is in the context (independent of the sample), and another which indicates how complex the sample is in terms of the hypothesis. He shows that these restrictions are sufficient for a number of decidability results. We do not further pursue that approach here.

Grammar-Grammars and A Priori Measures

It should now be clear why we have devoted so much attention to stochastic grammars and stochastic presentations. We propose to use Bayes' theorem to incorporate information from the sample. Stochastic grammars provide the conditional probabilities which Bayes' theorem requires. Part II is entirely devoted to procedures which use such measures. No particular form is assumed for the a priori measure on the grammars -- if an enumeration of the hypothesis space (EAO by probability or complexity) is provided in the problem statement, the procedure will work effectively. It remains to be shown that problems can be stated reasonably, i.e., that there is a finite means for specifying the probability distribution of an infinite class of grammars.

Before treating any specific measure we note that the complexity attributed to a grammar by any reasonable measure should increase when

we unboundedly increase any of

-- the number of rules (non-terminals) in the grammar,

-- the number of alternatives in any particular rule, or

-- the length of any particular alternative,

while holding all other factors fixed. Disagreement about complexity measures will involve the form and weights of these elements, not their existence.

Written grammars are strings; any particular class of written grammars is a subset of all the strings over a vocabulary, i.e., is a language. In fact, it is easy to write grammar-grammars [Schorre 1964] which generate only written finite-state, or linear, or context-free, or even context-sensitive grammars. We can include restrictions on the form of productions (e.g., standard form or normal form) if we wish. We could have formally defined written grammars in terms of grammar-grammars, but we wished to avoid an appearance of circularity.

A class of grammars may be specified to an inference procedure by means of a grammar-grammar. If the grammar-grammar is a stochastic grammar, it also imposes a probability distribution over its sentences (grammars). We may take as the a priori probability of a grammar its probability with respect to the grammar-grammar.

For example, consider the simple grammar-grammar $\bar{G}_3$ [1]:

S ::= R | RR	(1/2, 1/2)
R ::= N "::=" P	(1)
P ::= A | P "|" A	(1/2, 1/2)

[1] We have an immediate problem in distinguishing symbols of the grammar-grammar from those of the grammars it is to generate, which we resolve by "quoting" all symbols of the terminal vocabulary of the grammar-grammar.

A ::= T | TN (1/2, 1/2)

T ::= "a" | "b" (1/2, 1/2)

N ::= "S" | "A" (1/2, 1/2)

$\bar{G}_3$ generates finite-state grammars with one or two rules and a terminal vocabulary of (at most) "a" and "b". To the universal grammar G_4

S ::= a | b | aS | bS

it assigns probability $P(G_4|\bar{G}_3) = 2^{-15}$, or complexity $M(G_4|\bar{G}_3) = 15$. To the grammar G_5

S ::= b | bS | aA

A ::= a | bA | aS

which appeared at the start of Chapter I, it assigns probability $P(G_5|\bar{G}_3) = 2^{-23}$, or complexity $M(G_5|\bar{G}_3) = 23$. Now we may compare G_4 and G_5 on the basis of Feldman's sample S_{10}

b	baba
bb	abba
aa	bbaba
baa	bbaa
aba	aabb

which they both generate. Under the assumption of equal probabilities for all the alternatives in a rule[1/] the complexity of each derivation step with respect to G_4 is $-\log_2(1/4) = 2$, and with respect to G_5

1/ Which, we have previously noted, is reasonable (and customary) for finite-state grammars.

is $-\log_2(1/3) = \log_2(3) \approx 1.58$. Since each derivation step adds precisely one terminal symbol, the length of a string is equal to the number of steps in which it was derived. Thus, for either grammar, 32 steps are required to derive S_{10}

$$M(S_{10}|G_4) = 32 \times 2 = 64$$
$$M(S_{10}|G_5) = 32 \times 1.58 = 50.56 \quad .$$

Finally,

$$M(G_4|S_{10},\bar{G}_3) = M(G_4|\bar{G}_3) + M(S_{10}|G_4)$$
$$= 15 + 64 = 79$$
$$M(G_5|S_{10},\bar{G}_3) = M(G_5|\bar{G}_3) + M(S_{10}|G_5)$$
$$= 23 + 50.56 = 73.56 \quad .$$

Or, in terms of probability,

$$\frac{P(G_5|S_{10},\bar{G}_3)}{P(G_4|S_{10},\bar{G}_3)} = \frac{2^{-73.56}}{2^{-79}} = 2^{5.46} \approx 44 \quad .$$

That is, in the context of the grammar-grammar $\bar{G}_3$ and our assumption of equi-probable alternatives, the grammar which Feldman inferred, G_5 , is about 44 times as probable as the universal grammar, G_4 . We leave it as an exercise for the reader to show that under these conditions G_5 is in fact more probable than any other grammar generated by $\bar{G}_3$.

The point of this example has not been that $\bar{G}_3$ is a particularly good grammar-grammar (better ones are available) nor that $M(G|\bar{G}_3)$ is the correct complexity measure, nor even that Feldman

inferred the right grammar. Rather, it is that stochastic grammar-grammars provide an adequate means for specifying a priori probabilities of grammars. By varying the probabilities associated with the alternatives of $\bar{G}_3$ we can control the weight attached to various components of the complexity measure (e.g., by increasing the second probability in the third rule we decrease the bias against rules with many alternatives, by increasing the first probability of the fourth rule we indicate that terminating rules are less complex, etc.) Several grammar-grammars and their associated complexity measures are given by Feldman, et al. [1969].

There are still two minor points to be cleared up. First, we have restricted ourselves (cf. Chapter III) to reduced, unambiguous grammars. But our grammar-grammars cannot, in general, enforce this restriction. Thus only a subset of the language of the grammar-grammar will be allowable, and the probabilities of the allowable grammars will not sum to unity. They will sum to a finite value, however, which (if known) could be used to normalize their probabilities. But when we compare the probabilities of grammars this normalization constant cancels, so we do not really need to know its value.

Second, we require our grammar-grammars (like all grammars) to be finite. Yet they must generate grammars with an unbounded number of non-terminal symbols.[1] There are two ways to resolve this conflict: in Feldman, et al. [1969] a collection (roughly equivalent to the language of a grammar-grammar-grammar) is defined as an infinite set of grammar-grammars differing only in the number of non-terminal

[1] We assume that the terminal vocabulary is known for any given application.

symbols allowed in the grammars they generate; alternatively, we may note that only a finite set of characters will be used in written grammars, and non-terminals may be denoted by certain strings of these characters (i.e., that non-terminal names themselves form a language). We may then add rules to the grammar-grammar to generate, e.g., BNF-like, non-terminal names:

```
N ::= "<" M ">"
M ::= L | ML
L ::= "A" | "B" | "C" | ...
```

Either means of handling an infinite non-terminal vocabulary is formally adequate. The two are not equivalent, however, and the method chosen will have some effect on the a priori distribution obtained.

PART II

THE ENUMERATIVE BAYESEAN PROCEDURE FOR GRAMMATICAL INFERENCE

BLANK PAGE

V. BASIC PROPERTIES OF THE PROCEDURE EB

> It should be noted of course that proving the existence of a finite algorithm is only a first step toward finding a practical algorithm. In particular, we look at the "finite" algorithm for testing structural equivalence as presented here as such a first step. Nevertheless, in an area replete with theorems beginning 'there is no finite procedure for ... ,' it is gratifying to be able to present some more encouraging results.
>
> [Paull and Unger 1968]

Assumptions

In this chapter we state a restricted form of the grammatical inference problem (we assume that all relevant probability distributions are known), present a solution, and show that this solution has desirable properties. Later chapters discuss means for improving efficiency and for relaxing various restrictions.

We state the problem as follows:

1) The hypothesis space is a denumerable class of stochastic grammars; each grammar has a known (computable) a priori probability of being correct (i.e., of being the source of the observations); an enumeration effectively approximately ordered by probability is available.
2) The observation space is the stochastic text presentation of a grammar in the hypothesis space.

3) The best hypothesis is the (a priori) most probable[1] of the grammars which are stochastically equivalent to the grammar which is the source of the observations.

4) At each step of any presentation, an acceptable procedure must minimize the probability of guessing other than the best hypothesis.

These assumptions provide a well-defined problem in the sense of Chapter I. In the next section we show that it is solvable, under the assumption that the given a priori probabilities are objectively correct.[2] We do not specify the form of the a priori probability function. If it is derived from a grammar-grammar, as suggested in Chapter IV, any enumeration which is EA0 by length will also be EA0 by probability. To incorporate other probability or complexity measures we would have to produce EA0 enumerations.

Assumption 2) is the really strong condition, since it requires that successive strings in the samples be independent and identically distributed random variables. For some applications (e.g., bubble chamber pictures) this is almost certainly a valid assumption; for others (e.g., speech recognition) it may be a good approximation. But there are certainly applications [cf. Feldman 1967] for which

---

[1] For definiteness, if there is no unique maximum, arbitrarily define the first occurrence of the maximum to be the best.

[2] If they are not, our procedure will not be optimal in the sense of requirement 4), but will still be effective, and will still identify the best grammar in the limit.

it does not hold at all -- a topic to which we return in Chapter XI.

The third assumption should not be controversial. We would certainly require that the best grammar at least be weakly equivalent to the grammar which generated the sample. We have no means for discriminating among stochastically equivalent grammars except a priori probability, and we in general prefer the most probable (or simplest) grammar.

The final requirement is also natural (although we will suggest in Chapter XI that it might perhaps be improved). It merely repeats our informal suggestion that the optimal procedure is the one which guesses right most often. Minimizing the probability of guessing wrong is equivalent to maximizing the probability of guessing right. The only information (other than the probability distributions themselves) available is the current sample, so we wish to determine the grammar with maximum a posteriori probability, given the sample. From Chapter IV we recall that we are to maximize

$$P(G_i|S_k,C) = \frac{P(G_i|C)\cdot P(S_k|G_i,C)}{P(S_k|C)}$$

or, equivalently

$$P'(G_i|S_k,C) = P(G_i|C)\cdot P(S_k|G_i,C)$$

which, by our results in Chapter III, may be re-written

$$P'(G_i|S_k,C) = P(G_i|C) \cdot \prod_{j=1}^{k} P(\sigma_j|G_i)$$

$$= P(G_i|C) \cdot \prod_{\tau\in V_t^+} P(\tau|G_i)^{f(\tau,S_k)} \quad .$$

The Procedure EB

Def. V.1. The enumerative Bayesean procedure. Let the hypothesis space be $\langle G_1, G_2, \ldots \rangle$ with $T(\delta)$ a computable function such that $i > T(\delta)$ implies $P(G_i|C) < \delta$, and let S_k be the current sample. The procedure EB consists of the following four steps:

1) Let t_k be the least integer such that G_{t_k} is DA with respect to S_k.
2) Let $\delta_k = P'(G_{t_k}|S_k, C)$.
3) Let $T_k = T(\delta_k)$.
4) For each G_i, $t_k \leq i \leq T_k$ compute $P'(G_i|S_k, C)$. Let EB(k) be the first i in this range for which $P'(G_i|S_k, C)$ is maximum. Guess $G_{EB(k)}$.

Lemma V.2. The procedure EB is effective.

Proof. Each step is effective:

1) By assumption the sample is generated by some grammar in the hypothesis space, which has a finite index. Therefore the first DA grammar has a finite index.
2) $P(G|C)$ and $P(S_k|G)$ are computable and non-zero; so is their product.
3) By hypothesis T is computable, and since $\delta_k > 0$, T_k is finite.
4) The maximization is over a finite set of computable values.

Q.E.D.

Theorem V.3. Of all the grammars in $\langle G_1, G_2, \ldots \rangle$, the procedure EB guesses a grammar with maximum a posteriori probability, given the sample S_k.

Proof. Assume not. Then there must be some G_i such that

$$P'(G_i|S_k,C) > P'(G_{EB(k)}|S_k,C) \quad .$$

We show that assuming any value for i leads to a contradiction:

1) $i < t_k$. This contradicts the fact that t_k is the least index of a DA grammar.

2) $t_k \leq i \leq T_k$. This contradicts the fact that $P'(G_{EB(k)}|S_k,C)$ is the maximum value of P' over this range.

3) $i > T_k$. From

$$P'(G_i|S_k,C) > P'(G_{EB(k)}|S_k,C)$$

we have a fortiori

$$P'(G_i|S_k,C) > \delta_k$$

$$P(G_i|C)\cdot P(S_k|G_i) > \delta_k$$

and, since $0 \leq P(S_k|G_i) \leq 1$,

$$P(G_i|C) > \delta_k \quad .$$

But this contradicts the hypothesis that $i > T(\delta_k)$ implies $P(G_i|C) < \delta_k$.

Q.E.D.

Taken together, Lemma V.2 and Theorem V.3 show that the procedure EB is an effective solution to the grammatical inference problem stated in the previous section. We now turn to the question of its limiting behavior. We would like to show that in the limit it always identifies the best grammar, but this is not possible. A stochastic presentation contains all sorts of perverse information sequences; we content ourselves with a proof that their aggregate probability is infinitesimal.

Lemma V.4. If $\{\hat{G}_j | j = 1,2,...\}$ is a set of stochastically equivalent grammars, the procedure EB will guess at most one of them.

Proof. Any sample will have the same conditional probability with respect to all the $\hat{G}_j$. The P' are thus proportional to the a priori probabilities of the grammars. Thus, at most, the first $\hat{G}_j$ with maximum a priori probability can ever be guessed by EB.

Corollary V.5. The procedure EB will never guess any grammar that is stochastically equivalent to the best grammar, except the best grammar itself.

Proof. This is immediate from our definition of the best grammar and Lemma V.4.

This result does not require that stochastic equivalence be decidable, or that EB make any special tests for equivalence. Its practical advantage comes from the fact that we must exclude stochastically equivalent grammars from the next theorem.

For a variety of reasons (including the fact that it computes P' rather than P) the procedure EB need not approach a limiting probability of one for the best grammar. However, a weaker condition is sufficient to assure that the best grammar will be guessed consistently.

Def. V.6. A grammar G is N-preferred over the set $\{G_i | i = 1,2,\ldots\}$ with the sample S_k iff its a posteriori probability is at least N times that of any element in the set, i.e.,

$$(\forall i) P'(G|S_k,C) \geq N \cdot P'(G_i|S_k,C) \quad .$$

Theorem V.7. Let G_B be the best grammar for the stochastic presentation of G. Let $\hat{G} = \{G_j | G_j$ is stochastically inequivalent to $G\}$. For any $N > 0$, $\epsilon > 0$ there is a $T(G_B,N,\epsilon)$ such that the total probability of samples S_k for which G_B is not N-preferred over $\hat{G}$ is less than ϵ, whenever $k > T(G_B,N,\epsilon)$.

Proof. For each j, let $C_j(N,S_k)$ be the characteristic function of samples S_k with the property that G_B is not N-preferred over $\{G_j\}$. We may readily bound C_j:

$$C_j(N,S_k) < \left[\frac{N \cdot P'(G_j|S_k,C)}{P'(G_B|S_k,C)} \right]^{\alpha} , \quad \alpha > 0 ,$$

and thus its expectation D_j

$$D_j(N,k) = E[C_j(N,S_k)|G]$$

$$< E\left[\left[\frac{N \cdot P'(G_j|S_k,C)}{P'(G_B|S_k,C)} \right]^{\alpha} |G \right]$$

$$= \sum_{S_k(G)} P(S_k|G,C) \cdot \left[\frac{N \cdot P'(G_j|S_k,C)}{P'(G_B|S_k,C)}\right]^{\alpha}$$

$$= \left[\frac{N \cdot P(G_j|C)}{P(G_B|C)}\right]^{\alpha} \cdot \sum_{S_k(G)} P(S_k|G) \cdot \left[\frac{P(S_k|G_j)}{P(S_k|G_B)}\right]^{\alpha}$$

$$= \left[\frac{N \cdot P(G_j|C)}{P(G_B|C)}\right]^{\alpha} \cdot \sum_{S_k(G)} P(S_k|G_B)^{1-\alpha} \, P(S_k|G_j)^{\alpha} \quad .$$

Now we partition $\hat{G}$ into two sets: a finite set of "probable" grammars, and an infinite set with low total a priori probability, and show that the contribution due to each can be bounded.

1) Improbable grammars: There is a finite $M = M(G_B, \epsilon/2N)$ such that

$$\sum_{i=1}^{M} P(G_i|C) > 1 - \frac{\epsilon}{2N} \cdot P(G_B|C)$$

or

$$\sum_{i=M+1}^{\infty} P(G_i|C) < \frac{\epsilon}{2N} \cdot P(G_B|C) \quad .$$

Setting $\alpha = 1$, we have

$$D_j(N,k) < \frac{N \cdot P(G_j|C)}{P(G_B|C)} \cdot \sum_{S_k(G)} P(S_k|G_i)$$

$$\leq \frac{N \cdot P(G_j|C)}{P(G_B|C)} \quad .$$

Now we can define $D_{Im}(N,k)$ as the aggregate probability

over the improbable grammars.

$$D_{Im} \leq \sum_{i=M+1}^{\infty} D_i(N,k)$$

$$< \sum_{i=M+1}^{\infty} \frac{N \cdot P(G_i|C)}{P(G_B|C)}$$

$$= \frac{N}{P(G_B|C)} \cdot \sum_{i=M+1}^{\infty} P(G_i|C)$$

$$< \frac{N}{P(G_B|C)} \cdot \frac{\epsilon}{2N} \cdot P(G_B|C)$$

$$= \frac{\epsilon}{2} \quad .$$

2) Probable grammars: Set $\alpha = 1/2$

$$D_j(N,k) < \left[\frac{N \cdot P(G_j|C)}{P(G_B|C)}\right]^{1/2} \cdot \sum_{S_k(G)} [P(S_k|G_B) \cdot P(S_k|G_j)]^{1/2}$$

$$= \left[\frac{N \cdot P(G_j|C)}{P(G_B|C)}\right]^{1/2} \cdot \left[\sum_{\tau \epsilon V_t^+} (P(\tau|G_B) \cdot P(\tau|G_j))^{1/2}\right]^k$$

$$= \left[\frac{N \cdot P(G_j|C)}{P(G_B|C)}\right]^{1/2} \cdot R_j^k$$

where

$$R_j = \sum_{\tau \epsilon V_t^+} (P(\tau|G_B) \cdot P(\tau|G_j))^{1/2} < 1 \quad .\underline{1/}$$

1/ The inequality follows from stochastic inequivalence, and is the reason we have treated separately the case of grammars which are stochastically equivalent to G_B .

Now let

$$T_j(G_B,N,\epsilon) = \frac{\log\left[\frac{\epsilon}{2\cdot M}\cdot\left(\frac{P(G_B|C)}{N\cdot F(G_j|C)}\right)^{1/2}\right]}{-\log\,(R_j)}$$

If $k > T_j(G_B,N,\epsilon)$ then

$$D_j(N,k) < \frac{\epsilon}{2M}$$

and if

$$T(G_B,N,\epsilon) = \max_{\substack{i=1,\ldots,M \\ G_i\epsilon\hat{G}}} [T_i(G_B,N,\epsilon)]$$

then for $k > T(G_B,N,\epsilon)$

$$D_{Pr} \leq \sum_{\substack{i=1 \\ G_i\epsilon\hat{G}}}^{M} D_i(N,k)$$

$$\leq \sum_{i=1}^{M} D_i(N,k)$$

$$< \sum_{i=1}^{M} \frac{\epsilon}{2M}$$

$$= \frac{\epsilon}{2} \quad .$$

Finally, combining the two sets

$$D \leq D_{Im} + D_{Pr} < \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \quad .$$

Q.E.D.

Def. V.8. An inference procedure converges to a grammar G under a stochastic presentation iff for any $\epsilon > 0$ there is a $T(\epsilon)$ such that for any $k > T(\epsilon)$ the total probability of samples S_k such that the procedure does not guess G is less than ϵ.

Corollary V.9. The procedure EB converges to the best grammar under any stochastic presentation.

Proof. By Corollary V.5 it never guesses a non-best grammar stochastically equivalent to the best grammar, so we need only consider inequivalent grammars. But, by Theorem V.7, the best grammar will ultimately be N-preferred over all the inequivalent grammars, with probability greater than $1-\epsilon$. We can, for example, set $T(\epsilon) = T(G_B, 2, \epsilon)$.

Corollary V.9 assures us that with stochastic presentation the procedure EB ultimately has arbitarily small probability of guessing other than the best grammar. Its proof does not involve any properties of the a priori probability measure other than that the enumeration is EAO by probability. However, the procedure is not effective without some probability measure -- it cannot be certain when it is safe to stop enumerating. If we are given an enumeration but no a priori probabilities, we can assign them almost arbitrarily (e.g., let $P(G_i|C) = 2^{-i}$) without jeopardizing limiting behavior.

Strictly speaking, we do not need the existence of a stochastic presentation. A virtually identical proof of Theorem V.7 can be based on convergence of the information sequence rather than on stochastic presentation. In the limit, therefore, the successive strings of the

sample need not be independent, so long as the relative frequencies converge properly.

Improvements

The procedure EB is deliberately simple, to make its essential characteristics apparent. Our statement of the problem has uniquely defined (except for ties) the guesses it must make, but the method by which it obtains them is not unique. In this section we consider a variety of changes to EB which leave its guesses unchanged, but improve its efficiency.

Lemma V.10. The guess EB(k) is unchanged if, at each time k, EB considers only grammars which are DA with respect to S_k.

Proof. If a grammar G_i is non-DA, $P(S_k|G_i) = 0$ and $P'(G_i|S_k,C) = 0$. But G_{t_k} (at least) has a posteriori probability $\delta_k > 0$, so G_i does not have the maximum a posteriori probability. But by Lemma V.3, EB guesses a grammar which does have maximum a posteriori probability. Therefore G_i is not the guess EB(k) and dropping it from consideration will not change EB's guess.

This lemma is important in the next chapter, where we consider efficient enumerations of DA grammars.

Theorem V.11. The guess EB(k) is unchanged if in step 2) "$\delta_k = P'(G_{t_k}|S_k,C)$" is replaced by "$\delta_k = P'(G_{t_k}|S_k,C)/\Delta_k$, where $\Delta_k = \prod_{\tau \in V_t^+} [f(\tau,S_k)/k]^{f(\tau,S_k)}$."

Proof. We first show that Δ_k is the maximum value of $P(S_k|G_i,C)$ for any i. Let $\{p_\tau|\tau\epsilon V_t^+\}$ be the set of probabilities $P(\tau|G_i,C)$ which maximize $P(S_k|G_i,C)$. We use the method of LaGrange to maximize

$$\prod_{\tau\epsilon V_t^+} p_\tau^{f(\tau,S_k)}$$

subject to the constraint

$$\sum_{\tau\epsilon V_t^+} p_\tau = 1 \quad .$$

Let

$$L = \lambda \cdot \sum_{\tau\epsilon V_t^+} p_\tau + \prod_{\tau\epsilon V_t^+} p_\tau^{f(\tau,S_k)} \quad .$$

$$0 = \frac{\partial L}{\partial p_\tau} = \lambda + \frac{f(\tau,S_k)}{p_\tau} \prod_{\tau\epsilon V_t^+} p_\tau^{f(\tau,S_k)}$$

$$= \lambda + \frac{f(\tau,S_k)}{p_\tau} \cdot (L-\lambda)$$

$$p_\tau = f(\tau,S_k) \cdot \frac{\lambda-L}{\lambda} \quad .$$

We can determine λ from

$$\sum_{\tau\epsilon V_t^+} p_\tau = 1 \quad , \quad \sum_{\tau\epsilon V_t^+} f(\tau,S_k) = k \quad ,$$

yielding

$$1 = k \cdot \frac{\lambda-L}{\lambda} \quad ,$$

$$p_\tau = f(\tau,S_k)/k \quad .$$

Substituting this value into $P(S_k|G_i,C)$ we have

$$\max_i [P(S_k|G_i,C)] \leq \prod_{\tau \epsilon V_t^+} p_\tau^{f(\tau,S_k)}$$

$$= \prod_{\tau \epsilon V_t^+} [f(\tau,S_k)/k]^{f(\tau,S_k)}$$

$$= \Delta_k \quad .$$

But this implies

$$P'(G_i|S_k,C) = P(G_i|C) \cdot P(S_k|G_i,C)$$

$$\leq P(G_i|C) \cdot \Delta_k \quad .$$

Thus if $P(G_i|C) < P'(G_{t_k}|S_k,C)/\Delta_k$

$$P'(G_i|S_k,C) < P'(G_{t_k}|S_k,C)$$

and G_i cannot be the guess $EB(k)$.

Q.E.D.

This theorem permits a significant reduction in the number of grammars considered by EB at each step. For example, if the information sequence is convergent

$$f(\tau,S_k)/k \rightarrow P(\tau|G)$$

and

$$\Delta_k = \prod_{\tau \epsilon V_t^+} [f(\tau,S_k)/k]^{f(\tau,S_k)}$$

$$\rightarrow \prod_{\tau \epsilon V_t^+} P(\tau|G)^{P(\tau|G) \cdot k}$$

$$= [\prod_{\tau \epsilon V_t^+} P(\tau|G)^{P(\tau|G)}]^k \quad .$$

A typical value for the bracketed factor is $1/2$, indicating that a factor of 2^k in the cutoff point is likely.

Lemma V.12. The guess EB(k) is unchanged if, in step 4) of EB, whenever a G_i is found such that $P'(G_i|S_k,C) > P'(G_{t_k}|S_k,C)$ t_k is replaced by i and steps 2) through 4) are repeated.

Proof. Obvious. No grammar better than G_i can occur after $T(P'(G_i|S_k,C))$, so the enumeration on out to the old T_k is superfluous.

Lemma V.13. The grammars up to G_{T_k} need not be re-enumerated at time $k+1$ if $P'(G_i|S_k,C)$ has been recorded for each DA grammar.

Proof. If a grammar is DA at time $k+1$, it must have been DA at time k . The new conditional probabilities can be computed by the recursive relation

$$P'(G_i|S_{k+1},C) = P'(G_i|S_k,C)\cdot P(\sigma_{k+1}|G_i) \quad .$$

Bounding A Posteriori Probabilities

In some situations it would be desirable for the procedure EB to not only guess a grammar, but to estimate the a posteriori probability that it is the best grammar. To compute $P(G_i|S_k,C)$ rather than $P'(G_i|S_k,C)$ we require the denominator $P(S_k|C)$ which is defined by an infinite sum

$$P(S_k|C) = \sum_i P(G_i|C)\cdot P(S_k|G_i) = \sum_i P'(G_i|S_k,C) \quad .$$

This sum can be bounded in terms of quantities which EB computes anyhow.

Lemma V.13. In the procedure EB

$$\sum_{i=1}^{T_k} P'(G_i|S_k,C) \leq P(S_k|C) \leq \sum_{i=1}^{T_k} P'(G_i|S_k,C) + \Delta_k \cdot (1 - \sum_{i=1}^{T_k} P(G_i|C)) .$$

Proof. All terms of the sum $P(S_k|C) = \sum_i P'(G_i|S_k,C)$ are non-negative. The lower bound is simply the first T_k terms of the sum. The upper bound is attainable iff all the remaining terms have $P(S_k|G_i) = \Delta_k$, which Theorem V.11 demonstrated to be maximal.

Both of these bounds will be fairly loose until the best grammar is enumerated, and fairly tight thereafter.

VI. DEDUCTIVE CONSIDERATIONS

> To imagine that an adequate grammar could be selected from the infinitude of conceivable alternatives by some process of pure induction on a finite corpus of utterances is to misjudge completely the magnitude of the problem.
>
> . . . . .
>
> Too much faith should not be put in the powers of induction, even when aided by intelligent heuristics, to discover the right grammar. After all, stupid people learn to talk, but even the brightest apes do not.
>
> [Chomsky 1963]

Reasons for Deductive Preprocessing

We have shown in Lemma V.10 that the procedure EB will make the same guess independent of whether the enumeration contains all grammars or merely the deductively adequate (DA) grammars, since non-DA grammars are assigned a posteriori probabilities of zero. Thus, formally, there is no need to augment the inductive procedure with a deductive procedure: all grammars which can be ruled out deductively are automatically rejected by Bayes' theorem. In practice, however, there may be substantial advantage to a procedure that eliminates as many grammars as it can deductively, using the procedure EB only to discriminate among DA grammars.

The need for deductive preprocessing arises from the large number of grammars with similar complexities. Lemma V.2 shows that only a

finite number of grammars are considered at each step, but does not bound that number. Informally, we can see that this number is the number of grammars "not too much" more complex than the grammar selected at that step. Before the correct grammar can be guessed, the procedure must at least have considered all other grammars of equal or lower complexity. We can use the number of such grammars as a lower bound on the number considered in the inference. But this number grows exponentially with complexity.

We have repeatedly used the grammar G_4 :

$$S ::= b \mid bS \mid aA$$
$$A ::= a \mid bA \mid aS$$

as an example. Depending on the precise complexity measure used, there are from several hundred to a few thousand finite-state grammars with two terminal symbols which are no more complex than this one, and therefore must be considered by the procedure. Only about 20 of these are DA, however, given the sample S_{10} :

b	baba
bb	abba
aa	bbaba
baa	bbaa
aba	aabb

Only the DA grammars actually contribute in any way to the solution; the others merely absorb computation, and (ideally) should be rejected as soon as possible. The reason that the constructive methods of Chapter II

involve less computation than enumerative methods is that only DA grammars are ever considered. Although this relatively small sample has yielded an orders of magnitude reduction in the hypothesis space, it might not appear necessary, since a thousand grammars may be managed directly (but slowly); however, the million or billion grammars involved in inferring grammars only two or three times as complex are clearly unmanageable.[1/]

The principal goal of this study has been to develop adequate and effective methods for evaluating the DA grammars, once they are enumerated, leaving the development of deductive techniques and tree-searching procedures to others [cf. Pohl 1969, Sandewall 1969, Feldman, et al. 1969]. For at least some interesting applications (e.g., bubble chamber picture grammars), the enumeration can be greatly restricted by application-dependent, extra-grammatical criteria. However, in the course of programming and testing the general procedure we have found that some deductive preprocessing is required to make grammars that we consider at least marginally "interesting," (three or more non-terminals) inferrable with computational effort which we consider reasonable (a few minutes of 360/67 CPU time in LISP/360). The balance of this chapter is devoted to such methods.

1/ In any case, given any finite amount of computation it is easy to construct a grammar which will not be enumerated directly with that amount of computation. Our further results will show that this need not even be a particularly large grammar.

Restricting Productions

One method of restricting our hypothesis space (reducing our enumeration) is to consider only grammars whose productions have some particular form, e.g., the standard grammars or normal grammars of Defs. III.26 and III.27. The following theorems assure us that such restrictions do not reduce the power of our inference procedure.

Theorem [Greibach 1965]. Each context-free language is generated by a 2-standard grammar.

Theorem [Chomsky 1963]. Each context-free language is generated by a normal grammar.

The latter theorem can be extended to show that every stochastic grammar is stochastically equivalent to a normal stochastic grammar. For each grammar which is not normal (or standard) we can construct an equivalent grammar which is. Unfortunately, these transformations do not preserve complexity of grammars. These methods will not generally yield the simplest equivalent grammar in the given form. In fact, since weak equivalence is undecidable, it is not decidable which is the simplest weakly equivalent grammar in a given form; the problem is general, and not a fault of any particular construction used.

Consider the grammar

$$S ::= a \mid bSS$$

which is 2-standard. By Chomsky's construction (which we do not detail here) this grammar can be transformed to a normal grammar by adding two

new non-terminal symbols.

S ::= a | AB

A ::= b

B ::= SS

The new rules have only one alternative;[1] all derivations involving them are forced. Thus the original probabilistic structure on its language can be preserved by using the same alternative probabilities in the first rule. The complexity (or a priori probability) of this grammar -- by any reasonable measure -- has clearly not been preserved by the transformation. Finally, Greibach's construction can be applied to re-transform this grammar to a 2-standard grammar, yielding

S ::= a | bB

A ::= b

B ::= aS | bBS

Some languages have very simple grammars in one form, but only more complex grammars in a second; for other languages the situation is reversed. Thus the behavior of the inference procedure may be strongly influenced by the form selected. If a grammar in a particular form is required by the application, it is generally inefficient to use another form for the inference and transform afterwards. Although it is thus inappropriate to pick a single form for the general inference

---

[1] This is generally true for the Chomsky construction.

procedure, we can reward (through improved performance) problem statements which contain a restriction to a particular form.

A similar restriction which is more generally applicable involves the use of canonical written grammars. Grammars which differ only by the systematic substitution of names for the non-terminals are completely equivalent and cannot conceivably be distinguished by an inference procedure. We have implicitly utilized this equivalence by considering only grammars involving a standard set of non-terminal names (e.g., in our grammar-grammars). However, we can further restrict our class of written grammars without loss of generality.[1/]

Although the productions of a grammar are an unordered set, they are of necessity placed in some order in the written grammar. Written grammars which differ only in the order of their productions represent precisely the same grammar. We are thus free to order the productions for our convenience, e.g., by collecting all productions for a given non-terminal into a single rule. We can reduce to one the number of written grammars which represent any grammar by introducing a canonical ordering on productions, and requiring that productions be written in canonical order. For example, we might order our vocabulary by placing the distinguished non-terminal first, followed by the other non-terminals in alphabetical order, and finally by the terminal symbols in alphabetical order. Strings could be ordered by placing short strings before long, and ordering strings of equal length by the

---

1/ The ideal would be to establish a canonical written form such that each set of completely equivalent grammars corresponds to precisely one written grammar in canonical form, but we have been unable to achieve this goal.

first symbol in which they differ. Finally, productions could be ordered by left part, and by right part for equal left parts. Thus, the productions of the grammar G_4 would be ordered

(S,b)(S,aA)(S,bS)(A,a)(A,aS)(A,bA)

and the canonical form of the written grammar would be

S ::= b | aA | bS

A ::= a | aS | bA

Note that if we have restricted ourselves to particular names (e.g., S, A, B...) for the non-terminals (to reduce the number of completely equivalent grammars that we consider) the left parts of the rules are now redundant in the written form, and can be omitted without ambiguity

::= b | aA | bS

::= a | aS | bA

(although with some loss in readability). This <u>irredundant canonical written form</u> is the internal form adopted for use in our inference program; grammars are converted to more conventional form for output. Also note that a finite context-free grammar-grammar cannot enforce canonical ordering of the productions within a rule. Thus if we measure the probability (or complexity) of a canonical rule with k alternatives relative to a stochastic grammar-grammar we should multiply the value by k! (or subtract log(k!) from the complexity) to account for the k!-1 other equi-probable written rules which it also represents.

Our restrictions on written grammars have reduced the number of completely equivalent grammars which will be considered, without eliminating all representatives of any equivalence class.[1] We can now calculate precisely the number of different canonical written grammars in a standard form with given numbers (N and T) of terminal and non-terminal symbols. Let R_F denote the number of distinct right parts in form F, then the number of distinct rules is

$$RR_F = 2^{R_F}$$

and the number of distinct grammars in form F is

$$G_F = (RR_F)^N = 2^{N \cdot R_F} .$$

Now for standard finite-state grammars

$$R_{FS} = T + N \cdot T$$

$$G_{FS} = 2^{N \cdot T \cdot (1+N)} > 2^{N^2 \cdot T} .$$

For (Chomsky) normal grammars

$$R_{C2} = T + N^2$$

$$G_{C2} = 2^{N \cdot (T+N^2)} > 2^{N^3} .$$

For (Greibach) 2-standard grammars

---

[1] Each class will now have (N-1)! canonical written grammars, where N is the number of non-terminals.

$$R_{G2} = T(1 + N + N^2)$$

$$G_{G2} = 2^{N \cdot T(1+N+N^2)} > 2^{N^3 \cdot T} \quad .$$

However, if we restrict ourselves to G2 grammars which are also S-grammars

$$RR_{G2S} = 2^T \cdot T^{(1+N+N^2)}$$

$$G_{G2S} = 2^{N \cdot T} \cdot T^{(N+N^2+N^3)} > T^{N^3} \quad .$$

Since $T < 2^T$, this represents a substantial reduction.

Simple Restrictions

Our restriction (imposed in Chapter III) to reduced grammars may be thought of as a sample-independent reduction of the space of grammars. There are other restrictions using information from the sample which further reduce the space.

Up to this point we have tacitly assumed that the appropriate terminal alphabet for the grammar is known a priori. We can relax this requirement by the following observations: (1) If a grammar contains a terminal symbol which has not yet appeared in the sample, there is a better grammar for the sample[1] not containing that symbol; (2) If a grammar does not contain some terminal symbol which has appeared in the sample, it is not DA for that sample. From this we conclude that at each step it is only necessary to consider grammars whose terminal vocabulary is precisely that of the current sample.

[1] We know that the grammar obtained by deleting all productions involving the unused symbol will be simpler. It will also have simpler (or, at least, no more complex) derivations.

It is often possible to rule out whole classes of grammars as non-DA on the form of individual rules, i.e., to restrict ourselves to grammars containing particular rules. The simplest case of this sort involves strings of length 1, which (in either a standard or a normal grammar) must be produced by the rule for the distinguished non-terminal. Thus if there are k distinct strings of length 1 in the sample, the number of grammars enumerated can be reduced by a factor of 2^k merely by considering only grammars of the form

$$S ::= a_1|a_2|\ldots|a_k|\ldots$$
$$\vdots$$

Similarly, in standard grammars, each symbol a_i which is the first symbol of a sentence of length greater than one must occur in a production of the form $(S,a_i\varphi)$ for some φ. This analysis can be extended to rules beyond the distinguished rule, but at substantial complication, and with marginal utility. However, in its simple form it can contribute substantially to the effectiveness of the method discussed in the next section, for which the distinguished rule is of particular concern.

Splitting Grammars

As we showed in an earlier section, the number of grammars with N non-terminals becomes huge for fairly small values of N. Merely enumerating them (preparatory to determining which few of them are deductively adequate) is a sizable computational task. It is desirable to eliminate as many as possible "by class" (e.g., by requiring certain

productions, as in the previous section) before enumerating them. In this section we develop a method of enumerating the grammars with N+1 non-terminals by "splitting" the grammars with N non-terminals and show that all DA grammars with N+1 non-terminals result from splits of DA grammars with N non-terminals. Thus we can exclude (without enumerating them) the grammars resulting from splits of non-DA grammars.

Def. VI.1. A grammar G is a split of G' on A_i and G' is a merge of G on A_i and A_j iff replacing every occurrence of A_j in PR(G) by A_i yields PR(G') .[1/]

Lemma VI.2. Any merge of a DA grammar is DA.

Proof. If G is DA, every string in the sample has a derivation in G . But every derivation in G has an image in G' (obtained by substituting A_i for each occurrence of A_j) which produces the same terminal string. Therefore G' produces every string in the sample and is DA.

Corollary VI.3. Every DA grammar is the split of a DA grammar.

Corollary VI.4. Every split of a non-DA grammar is non-DA.

Remark VI.5. Merging preserves the form of productions. Thus any (standard/normal) DA grammar is the split of a (standard/normal) DA grammar. Similarly for reduced grammars. However, the

---

1/ This is a special case of grammatical covering [Reynolds 1968].

S-grammar property is not preserved; we will have to split non-S-grammars to obtain some S-grammars.

Def. VI.6. A split or merge is canonical if $A_i = S$ and A_j is the canonically last non-terminal of G, i.e., if the distinguished non-terminal is split into the distinguished and last non-terminals.

Lemma VI.7. Each (*) grammar with N+1 non-terminals, $N > 0$, is the canonical split of precisely one (*) grammar with N non-terminals.

Proof. The requirement that the distinguished and last non-terminals be combined uniquely determines the canonical merge of any given grammar; this is the unique grammar which can be canonically split to form the given grammar.

Corollary VI.8. The (*) grammars with N+1 non-terminals can be enumerated without repetition by performing all possible canonical splits of (*) grammars with N non-terminals.

Remark VI.9. By Remark VI.5 "DA," "standard," "normal," and/or "reduced" can be substituted for (*) in Lemma VI.7 and Corollary VI.8.

We have shown that by splitting the (comparatively few) DA grammars with N non-terminals we obtain all of (but not only) the DA grammars with N+1 non-terminals. The number of splits can be quite large (although it will not entail a complete enumeration) and

many of the resulting grammars will not be DA. We wish to rule out classes of non-DA grammars as efficiently as possible. For this purpose the required production test on the distinguished rule, given in the previous section, is fairly effective.

Def. VI.10. The canonical grammar splitting procedure. Let PR_S be the subset of $PR(G)$ with S as left part, let PR_R be the required subset of PR_S, and let N be the non-terminal to be added. The procedure CGS consists of the following steps.

1) For each $T \subset PR_S - PR_R$ do step 2).

2) Let $D = T \cup PR_R$ and $L = PR_S - T$. For each subset $E \subset D$ do step 3).

3) Let M be the result of replacing S as a left part by N in $E \cup L$, and let $PR_M = M \cup (PR - PR_S)$. If P is a production let $\hat{P}$ be any production obtained from P by substituting N for zero or more occurrences of S in the right part of P. Enumerate each $G(\hat{PR}_M)$, where $\hat{PR}_M$ is obtained from PR_M by replacing each $P \in PR_M$ by one or more $\hat{P}$'s.

If the grammar to be split has a distinguished rule with k necessary and ℓ "unnecessary" productions and the distinguished non-terminal occurs m times, there will be at least

$$s(k,\ell,m) = 2^k \cdot 3^{\ell+m}$$

canonical splits of the grammar; if the distinguished non-terminal occurs in the distinguished rule, or more than once in a single production, there will be somewhat more splits.

Finally, the notion of splitting permits us to organize the space of grammars as a tree, where the branches from each node represent splits, and each level represents an additional non-terminal. When a node is eliminated (non-DA), so are all its dependent nodes. Branches can be "grown" independently; we need not enumerate all grammars with N non-terminals before starting on those with $N+1$. In particular, there is a minimum "cost" in complexity involved in any split, so at any given time we need only perform splits which could result in grammars which are not more complex than the current bound, δ_k.

VII. INFERRING PARAMETERIZED GRAMMARS

> Those who believe that "probability" refers to a state of things, a property of a system (as proposed by von Mises), might well consider how they would prove to a skeptic that just because a coin comes up heads twice in a row he should not believe that it is loaded. If he persists in this belief, will you insist that he is irrational? Stupid? If you take this point of view, you must admit we are talking about states of knowledge. (The coin could be loaded, you know!)
>
> [Savage 1962]

> Succinctly, how does one go about stating the distribution of the parameter from available information? And, what if no a priori information is available, then what?
>
> [Aigner 1968]

> When the probability of a simple event is unknown, we may suppose all values of this probability between 0 and 1 as equally likely.
>
> [Laplace]

Estimation of Parameters

The deductive methods of Chapter VI (e.g., grammar splitting) were stated in terms of characteristic grammars rather than stochastic grammars. A one-to-one correspondence between the two kinds of grammars is required to combine these methods with those of Chapter V, which assume stochastic grammars. Chapter III mentioned two means for establishing this correspondence: all alternatives of a rule may be considered equi-probable, or the probabilities of alternatives may be treated as free parameters which must also be learned. There we noted

inadequacies of the former approach; now we turn to development of the latter. The proofs in Chapter V in general have analogs for parameterized grammars. The notation becomes substantially more cumbersome and we do not believe that a formal development is particularly enlightening. Therefore, the presentation in this chapter is mostly informal.

In this section we treat the case where the grammar is fixed and the probabilities are to be learned. The next section addresses the more general problem of simultaneously learning both the grammar and the parameters. Although the application is new, the statistical methodology is not particularly novel [Tribus 1962] [Savage 1962] [Aigner 1968]; we sketch it here because we have not found it presented anywhere in quite this form.

We may treat the hypothesis $H_i(\theta)$ with the free parameter θ as a set of compound hypotheses $H_i(\theta) = \{(H_i,\theta_j) | j = 1,2,3,\ldots\}$. If $P(\theta_j|H_i,C)$ and $P(S_k|\theta_j,H_i,C)$ are known for each j, Bayes' theorem can be used just as before to compute a posteriori probabilities

$$P(\theta_j|S_k,H_i,C) = P(\theta_j|H_i,C) \cdot \frac{P(S_k|\theta_j,H_i,C)}{\sum_\ell P(\theta_\ell|H_i,C) \cdot P(S_k|\theta_\ell,H_i,C)} .$$

In the continuous case, θ becomes a real variable, $P(\theta_j|H_i,C)$ is replaced by $\rho(\theta|H_i,C)$ where ρ is a probability density function (pdf), $\int\rho(\theta|H_i,C)d\theta = 1$, and the summation over ℓ turns intc integration over θ.

$$\rho(\theta|S_k,H_i,C) = \rho(\theta|H_i,C) \cdot \frac{P(S_k|\theta,H_i,C)}{\int \rho(\theta'|H_i,C) \cdot P(S_k|\theta',H_i,C)d\theta'} .$$

This is the general rule for determining the a posteriori density function from the a priori density and the sample. In the case of a parameterized stochastic grammar, with which we are concerned, θ corresponds to the probability of a particular alternative (or to a vector of such probabilities), let it be the n-th production. Now P must have the form

$$P(S_k|\theta_n,G_i,C) = \theta_n^{F_i(p_n,S_k)} \cdot g(S_k,G_i,C)$$

where

$$F_i(p_n,S_k) = \sum_{\tau\epsilon V_t^+} f(\tau,S_k) \cdot u_i(p_n,\tau)$$

and $u_i(p_n,\tau)$ is the number of uses of p_n in the derivation of τ through G_i.

Since g is independent of θ_n, it comes out of the integral and then cancels; leaving

$$\rho(\theta_n|S_k,G_i,C) = \rho(\theta_n|G_i,C) \cdot \frac{\theta^{F_i(p_n,S_k)}}{\int \rho(\theta'_n|G_i,C) \cdot \theta'^{F_i(p_n,S_k)} d\theta'_n} .$$

As usual, the Bayesean denominator is merely a normalizing constant, and the form of the result is determined by the numerator. Hence the effect of the sample S_k on the form of the pdf is to multiply it by $\theta_n^{F_i(p_n,S_k)}$. Similarly, if we have not one free parameter, but a vector of free parameters, the multi-dimensional pdf will be multiplied

by $\theta_1^{F_i(p_1,S_k)} \ldots \theta_m^{F_i(p_m,S_k)}$. In the absence of any knowledge of the form of ρ , this is about all we can say.

To simplify the sequel, we will assume that ρ itself takes the particularly simple form of a constant times a power of θ .[1/] This assumption admits a large class of pdf's (the multi-variate Beta distributions), which includes some important special cases to be discussed later, and has two convenient properties: ρ is completely specified by the exponent for each θ_n (the constant multiplier is determined by the requirement that ρ be a pdf); and the class is closed under Bayesean inference -- the new exponents again determine the function. If we write F_n for $F_i(p_n,S_k)$ we find

$$\rho(\theta_1,\ldots,\theta_m|S_k,G_i,C) = K \cdot \theta_1^{\beta_1}\ldots\theta_m^{\beta_m} \cdot \frac{\theta_1^{F_1}\ldots\theta_m^{F_m}}{\int K\cdot\theta_1'^{\beta_1}\ldots\theta_m'^{\beta_m}\cdot\theta_1'^{F_1}\ldots\theta_m'^{F_m}\cdot d\theta_1'\ldots d\theta_m'}$$

$$= \frac{\theta_1^{\beta_1+F_1}\ldots\theta_m^{\beta_m+F_m}}{\int \theta_1'^{\beta_1+F_1}\ldots\theta_m'^{\beta_m+F_m}\, d\theta_1'\ldots d\theta_m'} \quad .$$

To obtain the value of the normalizing constant, we must supply limits for the integration. If $\theta_1\ldots\theta_m$ were all independent, we could factor the integral into the product of m integrals, each involving only one θ_n . Because of the constraint that for each rule the sum

---

1/ The generalization to linear combinations of such functions (and hence to analytic functions) is straightforward; we do not pursue it here.

of the probabilities of the alternatives must be unity, we obtain instead one factor for each rule

$$\int_{0,\ldots,0}^{1,\ldots,1} \theta_1^{\beta_1+F_1} \ldots \theta_r^{\beta_r+F_r} \delta(\theta_1+\ldots+\theta_r-1)d\theta_1\ldots d\theta_r \quad .$$

In the simple case $r = 2$ this is the familiar Beta integral

$$\int_{0,0}^{1,1} \theta_1^{\beta_1+F_1} \theta_2^{\beta_2+F_2} \delta(\theta_1+\theta_2-1)d\theta_1 d\theta_2$$

$$= \int_0^1 \theta_1^{\beta_1+F_1} (1-\theta_1)^{\beta_2+F_2} d\theta_1$$

$$= B(\beta_1+F_1+1,\beta_2+F_2+1) = \frac{(\beta_1+F_1)!(\beta_2+F_2)!}{(\beta_1+\beta_2+F_1+F_2+1)!} \quad .$$

In the general case we have

$$\rho(\theta_1,\ldots,\theta_r|S_k,G_i,C) = \frac{1}{B(\beta_1+F_1+1,\ldots,\beta_r+F_r+1)} \cdot \prod_{n=1}^{r} \theta_n^{\beta_n+F_n}$$

$$= [(\sum_{n=1}^{r}(\beta_n+F_n+1))-1]! \cdot \prod_{n=1}^{r} \frac{\theta_n^{\beta_n+F_n}}{(\beta_n+F_n)!}$$

for each rule, independently.

It might appear that we have strayed somewhat from our original quest, which was to estimate the θ's. But now that we have a convenient form for the pdf, we can easily estimate θ_n by taking its expected value under the a posteriori distribution,

$$E[\theta_n|S_k] = \int \theta_n \cdot \rho(\theta_1,\ldots,\theta_r|S_k,G_i,C)d\theta_1\ldots d\theta_r$$

$$= \int \frac{\theta_1^{\beta_1+F_1}\ldots\theta_n^{\beta_n+F_n+1}\ldots\theta_r^{\beta_r+F_r}\,\delta(\theta_1+\ldots+\theta_r-1)d\theta_1\ldots d\theta_r}{B(\beta_1+F_1,\ldots,\beta_r+F_r)}$$

$$= \frac{B(\beta_1+F_1,\ldots,\beta_n+F_n+1,\ldots,\beta_r+F_r)}{B(\beta_1+F_1,\ldots,\beta_n+F_n,\ldots,\beta_r+F_r)}$$

$$= \frac{\beta_n+F_n+1}{\sum_{m=1}^{r}(\beta_m+F_m+1)} = \frac{\beta_n+F_n+1}{\beta+F+r}$$

where

$$\beta = \sum_{m=1}^{r}\beta_m \qquad F = \sum_{m=1}^{r}F_m \quad .$$

Similarly we can compute the variance of the density function

$$E[\theta_n^2|S_k]-E[\theta_n|S_k]^2 = \frac{(\beta_n+F_n+1)(\beta_n+F_n+2)}{(\beta+F+r)(\beta+F+r+1)} - \left[\frac{\beta_n+F_n+1}{\beta+F+r}\right]^2$$

$$= \frac{(\beta_n+F_n+1)(\beta+F+r) - (\beta_n+F_n+1)^2}{(\beta+F+r)^2(\beta+F+r+1)}$$

$$\lim_{k\to\infty} E[\theta_n^2|S_k] - E[\theta_n|S_k]^2 = \lim_{k\to\infty}\frac{F\cdot F_n - F_n^2}{F^3} = 0 \quad .$$

Thus the estimated values for the probability parameters have a very simple form in terms of the exponents in the a priori density

function (the β's) and the observed frequencies of the alternatives (the F's). Furthermore, the variance becomes small as the frequencies increase (i.e., the distribution peaks ever more sharply around the estimate). Note that the result involves only the sums β_n+F_n , not their individual values. We may interpret this as saying that a priori bias (β's) and observations (F's) affect the answer in precisely analogous fashions; at any time we can move observations into the bias and work from a "new" a priori distribution, without affecting later results.

We have not yet discussed how the β's are to be selected. In general, we might expect "by symmetry" that the β_n should all have the same value within a rule (although this might not be the case if we had some reason for distinguishing among alternatives, e.g., by length). Two particular choices are popular in the statistical literature: If each $\beta_n = 0$, corresponding to a uniform (independent of the θ_n) a priori density, we obtain the famous Laplace rule of succession

$$E[\theta_n|S_k] = \frac{F_n+1}{F+r} \quad .$$

Most "subjectivist" statisticians support this view. If, however, each $\beta_m = -1$, we obtain

$$E[\theta_n|S_k] = \frac{F_n}{F}$$

the "maximum likelihood" estimate; most "frequentist" statisticians

would support this result (since it is an unbiased estimate of θ_n) but deny the validity of our derivation.

In the limit, of course, any choice of finite values for the β's will have only infinitesimal effect on the estimate; our earlier proof that relative frequencies of strings in a stochastic presentation converge wpl to their probabilities can be carried over directly to show that $E[\theta_n|S_k]$ converges wpl to the probability of its alternative, independent of the β's .[1] Thus we shall not be dogmatic in insisting that any particular values must be used for the β's . We might hope that in each application, experience would indicate appropriate values; we conjecture that small positive values are generally best and that 0 will not usually be far wrong.

These results can be made more concrete by means of an example. Consider the grammar

$$S ::= t|hS$$

with probabilities θ_t and θ_h, where $\theta_t+\theta_h = 1$.[2] Assume that we have no a priori reason for preferring either alternative, so

---

[1] This is what Bayeseans mean by the statement that "you can always overwhelm a poor choice of the prior by sufficient evidence."

[2] The reader who easily relates statistics to coin flipping may interpret t as "tails," h as "heads," a string $h^n t$ as a "trial" which obtained a run of n "heads" and then terminated on appearance of "tails." The expected length of a string is the expected length of a trial, θ_t and θ_h are the probabilities of "tails" and "heads" on each toss.

$\beta_t = \beta_h$. We will compare the answers resulting from three different choices for the β's :

(a) the "frequentist" -- $\beta_t = \beta_h = -1$;

(b) the "indifferentist" -- $\beta_t = \beta_h = 0$;

(c) the "experienced" person who from earlier evidence thinks a "fair" division is highly probable -- $\beta_t = \beta_h = 50$.

At the start: With $S_o = \emptyset$, before seeing any strings, each choice yields $E[\theta_t|S_o] = E[\theta_n|S_o] = \frac{1}{2}$ (although for (a) we require L'Hospital's rule to compute the value).

After one t : If $S_1 = \langle t \rangle$, $F_t = 1$, $F_h = 0$, $F = 1$:

(a) $E[\theta_t|S_1] = \frac{F_t}{F} = \frac{1}{1} = 1$

(b) $E[\theta_t|S_1] = \frac{F_t+1}{F+2} = \frac{2}{3}$

(c) $E[\theta_t|S_1] = \frac{F_t+50}{F+100} = \frac{51}{101}$.

On the evidence of a single string, the "frequentist" estimates the probability of t at unity, the "indifferentist" estimates a 2:1 bias for t and the "experienced" estimates only a slight shift.

After another t : If $S_2 = \langle t,t \rangle$, $F_t = 2$, $F_h = 0$, $F = 2$:

(a) $E[\theta_t|S_2] = \frac{F_t}{F} = \frac{2}{2} = 1$

(b) $E[\theta_t|S_2] = \frac{F_t+1}{F+2} = \frac{3}{4}$

(c) $E[\theta_t|S_2] = \frac{F_t+50}{F+100} = \frac{52}{102} = \frac{26}{51}$.

After some h's : If $S_3 = \langle t,t,hhht \rangle$, $F_t = 3$, $F_h = 3$, $F = 6$:

(a)

$$E[\theta_t|S_3] = \frac{F_t}{F} = \frac{3}{6} = \frac{1}{2}$$

(b)

$$E[\theta_t|S_3] = \frac{F_t+1}{F+2} = \frac{4}{8} = \frac{1}{2}$$

(c)

$$E[\theta_t|S_3] = \frac{F_t+50}{F+100} = \frac{53}{106} = \frac{1}{2} .$$

Everyone agrees at this point.

After 1000 observations: If $F_t = 600$, $F_h = 400$, $F = 1000$:

(a)

$$E[\theta_t|S_{1000}] = \frac{F_t}{F} = \frac{600}{1000} = .6000$$

(b)

$$E[\theta_t|S_{1000}] = \frac{F_t+1}{F+2} = \frac{601}{1002} = .5998$$

(c)

$$E[\theta_t|S_{1000}] = \frac{F_t+50}{F+100} = \frac{650}{1100} = .5909$$

All three pretty well agree on the amount of bias for t .

Evaluation of Hypotheses with Free Parameters

We have shown how to estimate the values of the parameters for a fixed hypothesis. We turn now to the question of picking the best hypothesis when the parameters are not yet fixed. The general approach is based on the observation that for the hypothesis to be correct, it must be correct for some values of its free parameters; in the last section we developed estimates for the pdf of the values.

If Θ is a free real parameter of $H_i(\Theta)$

$$P(H_i(\Theta)|C) = \int P(H_i|C) \cdot \rho(\Theta|H_i,C)d\Theta$$

$$= P(H_i|C)$$

and

$$P(S_k|H_i(\Theta),C) = \int P(S_k|\Theta,H_i,C) \cdot \rho(\Theta|H_i,C) \cdot d\Theta$$

so, by Bayes' theorem

$$P(H_i(\Theta)|S_k,C) = P(H_i(\Theta)|C) \cdot \frac{P(S_k|H_i(\Theta),C)}{P(S_k|C)} \quad .$$

Dropping (as usual) the normalizing denominator

$$P'(H_i(\Theta)|S_k,C) = P(H_i(\Theta)|C) \cdot P(S_k|H_i(\Theta),C)$$

$$= P(H_i|C) \cdot \int P(S_k|\Theta,H_i,C) \cdot \rho(\Theta|H_i,C)d\Theta \quad .$$

Now specializing to stochastic grammars, recall

$$P(S_k|\Theta_1,\ldots,\Theta_m,G_i,C) = \prod_{n=1}^{m} \Theta_n^{F_i(p_n,S_k)} \quad .$$

The integral again can be factored into independent integrals for each rule. Recalling

$$\rho(\Theta_1,\ldots,\Theta_r|G_i,C) = \delta(\Theta_1+\ldots+\Theta_r-1) \frac{\prod_{n=1}^{r} \Theta_n^{\beta_n}}{B(\beta_1+1,\ldots,\beta_r+1)}$$

we have for each rule

$$J_R(\theta_1,\ldots,\theta_r|G_i,C) = \frac{\int \prod_{n=1}^{r} \theta_n^{F_i(p_n,S_k)+\beta_n}\, \delta(\theta_1+\ldots+\theta_r-1)d\theta_1\ldots d\theta_r}{B(\beta_1+1,\ldots,\beta_r+1)}$$

$$= \frac{B(\beta_1+F_i(p_1,S_k)+1,\ldots,\beta_r+F_i(p_r,S_k)+1)}{B(\beta_1+1,\ldots,\beta_r+1)}$$

$$= \frac{(\beta_1+F_i(p_1,S_k))!\ \ldots(\beta_r+F_i(p_r,S_k))!\cdot(\beta+r-1)!}{\beta_1!\ldots\beta_r!\cdot(\beta+F+r-1)!}$$

where (as before)

$$\beta = \sum_{n=1}^{r}\beta_n \qquad F = \sum_{n=1}^{r}F_i(p_n,S_k) \quad .$$

This result (with all β's set to 0) was derived by Solomonoff [1964] using a completely different method based on substantially different assumptions. His method suggests both an information-theoretic interpretation for the result and an incremental method for its computation. Suppose that with each alternative we store the current value of $\gamma_n = \beta_n+F_i(p_n,S_k)+1$ and that every time that alternative is used in a derivation we use the current estimate of θ_n

$$E[\theta_n|S_k] = \frac{\gamma_n}{\gamma}$$

to compute the minimum information (complexity) involved in that step

$$I = -\log\left(\frac{\gamma_n}{\gamma}\right) = \log\gamma - \log\gamma_n$$

and then increment both γ_n and γ by one. Now, after S_k has been completely derived, we find that (for each rule) γ has taken on each value from $\beta+r$ to $\beta+F+r-1$ precisely once, and each γ_n has taken on the values β_n+1 to $\beta_n+F_i(p_n,S_k)$. The sum will be independent of the order in which alternatives were used, depending only on their final frequencies, and is the negative of the logarithm of the integral computed previously

$$-\log(J_R) = \sum_{j=\beta+r}^{\beta+F+r-1} \log(j) - \sum_{n=1}^{r} \sum_{j=\beta_n+1}^{\beta_n+F_i(p_n,S_k)} \log(j) \quad .$$

We note that negative values for the β's (e.g., the "frequentist" $\beta_n = -1$) will in general cause this value to become infinite. Intuitively, we may see why as follows: if any γ_n is zero (or negative) when γ is positive, then $E[\theta_n|S_k]$ is zero (negative) and an infinite amount of information is required to specify that alternative. We note this in case (a) of the example in the previous section, where, on the basis of one observation $E[\theta_t|S_1] = 1$, $E[\theta_h|S_1] = 0$. No additional complexity is involved in further t's, but there is infinite complexity in the first h following these t's. The a priori probability density function has equal poles at $\theta_t = 0$ and $\theta_t = 1$, (recall L'Hospital's rule was required to evaluate $E[\theta_t|S_o]$ before the first observation). The first observation cancels one of these poles with a zero, leaving the other to completely dominate.

Limiting Behavior

In Chapter V we proved that the procedure EB was not only optimal at each step, but also converged to the best grammar. The proof involved the fact that each stochastic grammar in the enumeration had fixed values for each of its alternative probabilities. It can be shown that the evaluation measure of the previous section also converges to a correct grammar. But we require a different measure of best for parameterized grammars.

A characteristic (parameterized) grammar is stochastically compatible with a stochastic grammar if there is some assignment of its alternative probabilities $\hat{\theta}$ which will make it stochastically equivalent.[1] The degree of a grammar is the number of productions minus the number of rules, i.e., the number of alternative probabilities which can be adjusted independently. We state the following result, and then sketch its derivation in the case of convergent information sequences: Using the evaluation measure of the previous section, the procedure EB will converge to a stochastically compatible grammar of minimum degree.

For a convergent information sequence of the stochastic grammar G

$$\lim_{k \to \infty} f(\tau, S_k)/k = P(\tau|G) \quad .$$

Let

[1] Recall that we showed in a previous section that θ will converge to $\hat{\theta}$ in the limit. Thus any grammar stochastically compatible with the true grammar will approach stochastic equivalence with the true grammar.

$$\alpha_n = \lim_{n\to\infty} F_i(p_n, S_k)/k$$

$$= \lim_{k\to\infty} \sum_{\tau\epsilon V_t^+} f(\tau, S_k) \cdot u_i(p_n, \tau)/k$$

$$= \sum_{\tau\epsilon V_t^+} P(\tau|G) \cdot u_i(p_n, \tau)$$

and

$$\alpha = \sum_{n=1}^{m} \alpha_n \quad .$$

Now consider[1]

$$J_R(\theta_1, \ldots, \theta_r | G_i, C) = \frac{(\beta_1 + F_i(p_1, S_k))! \ \ldots (\beta_r + F_i(p_r, S_k))! (\beta - r + 1)!}{\beta_1! \ \ldots \beta_r! (\beta + F + r - 1)!} \quad .$$

Let

$$K_R = \frac{(\beta + r - 1)!}{\beta_1! \ldots \beta_r!} \quad .$$

Now let

---

[1] Recall that J_R is the contribution to the a posteriori probability made by a single rule.

$$J_R^{\infty}(\Theta_1,\ldots,\Theta_r|G_i,C) = \lim_{k\to\infty} J_R(\Theta_1,\ldots,\Theta_r|G_i,C)$$

$$= K_R \cdot \lim_{k\to\infty} \frac{\prod_{n=1}^{r}(\beta_n+F_i(p_n,S_k))!}{(\beta+F+r-1)!}$$

$$= K_R \cdot \frac{\prod_{n=1}^{r}(\beta_n+\alpha_n\cdot k)!}{(\beta+\alpha\cdot k+r-1)!} \quad .$$

By Stirling's approximation

$$x! \approx \sqrt{2\pi x}\ (\frac{x}{e})^x$$

$$J_R^{\infty}(\Theta_1,\ldots,\Theta_r|G_i,C)$$

$$\approx K_R \cdot \frac{\prod_{n=1}^{r}\left(\sqrt{2\pi\cdot(\beta_n+\alpha_n\cdot k)}\cdot\left[\frac{\beta_n+\alpha_n\cdot k}{e}\right]^{\beta_n+\alpha_n\cdot k}\right)}{\sqrt{2\pi\cdot(\beta+\alpha\cdot k+r-1)}\cdot\left[\frac{\beta+\alpha\cdot k+r-1}{e}\right]^{\beta+\alpha\cdot k+r-1}}$$

After considerable simplification, this reduces to

$$J_R^{\infty}(\Theta_1,\ldots,\Theta_r|G_i,C) \approx C_R \cdot D_R(k) \cdot P_R^k$$

where

$$C_R = (2\pi)^{\frac{r-1}{2}} \cdot \frac{(\beta+r-1)!}{\beta_1!\ldots\beta_r!} \cdot \frac{\alpha_1^{\beta_1+\frac{1}{2}} \ldots \alpha_r^{\beta_r+\frac{1}{2}}}{\alpha^{\beta+r-\frac{1}{2}}}$$

$$D_R(k) = k^{\frac{1-r}{2}}$$

$$P_R = \frac{\alpha_1^{\alpha_1} \cdots \alpha_r^{\alpha_r}}{\alpha^{\alpha}} \quad .$$

The factor P_R^k is precisely the contribution that this rule would make to the derivational probability if the a priori alternative probabilities were fixed at their optimum values, e.g., if the grammar is stochastically compatible then $\prod_R P_R^k$ will be the derivational probability of the stochastically equivalent form. As $k \to \infty$ this factor dominates, and in analogy with Theorem V.7 any grammar which is stochastically compatible to the true grammar will be preferred over any grammar which is not.

Among stochastically equivalent grammars, the contributions of the P_R^k factors will be equal, so the $D_R(k)$ factors will dominate: $\prod_R D_r(k) = k^{-\sum_R (1-r)/2}$. But $d = \sum_R (r-1)$ is precisely the degree of the grammar. $k^{-d/2}$ is maximum for minimum degree, so, out of a set of stochastically equivalent grammars, one of lowest degree will ultimately be preferred, in fact, the one with maximum

$$P(G_i|C) \cdot \prod_R C_R \quad .$$

The constants C_R are sensitive to the β's , and are a function of the "distance" between the initial approximations β_n/β and the limiting values α_n/α .

It may seem surprising that the evaluation method of the previous section ultimately prefers grammars of minimum degree, independent of the apparent complexity of their rules. This may perhaps seem more

reasonable in light of the remark[1] that the rules are finite and discrete, and therefore have bounded complexity, whereas the parameters may converge to any real number in the open interval zero to one. To specify any one of these parameters exactly would require an infinite amount of information. We do not do this, but rather use the samples to refine the estimates ever more closely. The more independent estimates which are made (i.e., the higher the degree) the more complexity involved, and this consideration ultimately outweighs any fixed complexity of rules.

---

[1] This argument is admittedly a posteriori, as the author was surprised by the result when he first derived it.

VIII. IMPLEMENTATION AND RESULTS

> We cannot live and we cannot solve problems without a modicum of optimism.
>
> [Polya 1954]

> Of course, we would have to supply the language-learning device with some sort of heuristic principles that would enable it, given its input data and a range of possible grammars, to make a rapid selection of a few promising alternatives, which could then be submitted to a process of evaluation, or that would enable it to evaluate certain characteristics of the grammar before others. The necessary heuristic procedures could be simplified, however, by providing in advance a narrower specification of the class of potential grammars. The proper division of labor between heuristic methods and specification of form remains to be decided.
>
> [Chomsky 1963]

Reasons for Implementation

For a variety of reasons, several of the procedures discussed in the last three chapters have been implemented as running computer programs. The discipline involved in actually stating these procedures as programs has led to greater precision of definition in a number of cases. The insights gained through running the procedures have strongly influenced the direction of our research -- the methods of Chapter VI were developed only after preliminary computer runs indicated

the essential role of deductive preprocessing. Finally, we felt that it was desirable to demonstrate that our formally optimal methods were practically implementable.

The programs have been successful in the sense that they have verified our methods and contributed to our understanding of the problem. They have, however, been disappointing in terms of computational efficiency, and it is not claimed that in their present form they are economically justifiable for practical applications. It seems clear that various heuristics could greatly speed the inference process, probably at small cost in terms of optimality. It is also evident that in many applications the restrictions on the hypothesis space are quite stringent, much more so than any general restrictions that we have proposed for our methods. Economics would probably dictate the use of both heuristics and extra-grammatical constraints to prune the hypothesis space in any real application. These topics are beyond the scope of what was attempted here.

All the programs were written in LISP/360 and run under the ORVYL time-sharing monitor on the Stanford University Computation Center, Campus Facility, IBM System 360/67 computer. LISP was chosen as the only sufficiently powerful language available under that monitor, rather than for any appropriateness to the problem. Due to the slowness of the programs, the original goal of extensive interaction with running programs was never fully realized, but the interactive capabilities greatly facilitated debugging. The programs were slow due to a combination of unfavorable circumstances: as

indicated in Chapter VI, the enumerative problem is immense, even with our improvements; LISP is not the optimal language for this application; the LISP compiler was not available in the time-shared system, so all programs were run interpretively; the problem of garbage-collecting list structures on secondary storage greatly slowed the system.[1] In short, the programs pushed the ORVYL-LISP/360 system far beyond its appropriate operating range.

The important procedures used in the programs have been given in the preceding three chapters; we do not repeat them here. Neither do we give program listings, although they are available from the author on request. The programs exhibit the characteristic incomprehensibility of large LISP programs, and little point would be served by repeating them here. We would advise anyone planning another implementation to work directly from the algorithms previously given.

Effects of Deductive Preprocessing

The initial portion of the enumerative Bayesean procedure selected for implementation was the enumeration itself. Although it was straightforward to write a program which enumerated the grammars in a given form, it soon became apparent that the enumeration process represented a serious problem. The first program quickly consumed the available memory for list structure and then began interminable

---

[1] This final problem was further complicated by the necessity of utilizing the structure-modifying pseudo-functions RPLACA and RPLACD which defeated the LISP system's attempts to concentrate lists on single pages.

garbage collections. It had to be abandoned in favor of a more complicated program which enumerated grammars one at a time and then immediately tested them, releasing the storage assigned to unsatisfactory grammars.

Even when restricted to reduced grammars, however, the procedure was rather slow, due to the voluminous nature of the enumeration. It was tested on two-non-terminal standard finite-state grammars with a two-symbol terminal alphabet.[1] By our results of Chapter VI, there are 2^{12} such grammars, and a large number of them are reduced. To progress, it was necessary to eliminate even more grammars. This was done by introducing the DA test based on a fixed sample. In a typical run using two minutes of CPU time about a tenth of the grammars (440) were enumerated.[2] Of these, 363 were either not reduced or not DA with respect to the one-string sample <b> . None of the 77 remaining grammars would have been DA with respect to the two-string sample <b,bb> .

This procedure (GRAMMARLIST) was transferred to the batch-processing system which provided more memory and a compiler, resulting in at least a factor of six speedup for short enumerations (and presumably more for longer ones). In the longest run, using the eleven-string sample

---

[1] With an eye to inferring Feldman's [1967] example grammar with which we began Chapter I.

[2] This figure cannot be linearly extrapolated to the full 4000 grammars, however, since the enumeration slowed down noticeably as more memory was consumed by the list of acceptable grammars and as lists became increasingly "scrambled" across page boundaries.

b	bbb
bb	bbbb
aa	abab
baa	baba
aba	abba
aab	

it determined in one minute's computation that there were 304 reduced DA grammars. Feldman's grammar, G_4 :

S ::= b | bS | aA

A ::= a | bA | aS

was not among the first 1000 grammars enumerated, but was the first reduced DA grammar to be enumerated -- the only such among the first 1800 grammars. Four grammars which were simpler than G_4 were reduced and DA, but they were all ambiguous. In this case, the requirements that a grammar be both reduced and DA cut the hypothesis space by more than an order of magnitude. Had the additional requirement of unambiguity over the sample (implemented later) been in effect, it would have provided another order of magnitude.

A similar test was performed with (Chomsky) normal grammars rather than finite-state grammars. The run was terminated after one minute of CPU time, during which 3100 grammars were enumerated and

360 found to be reduced and DA.[1] The larger number of DA grammars can be largely attributed to a greater density of universal grammars among normal grammars than among finite-state grammars.

By this time, the author was convinced that the computational cost of enumerating and testing all grammars was excessive. This led directly to the development of the splitting methods of Chapter VI, which were incorporated into the inference procedure discussed in the next section.

Although the importance (and effectiveness) of the restrictions to reduced and to DA grammars grows with the size of the grammars being enumerated, the following simple example is illustrative: Consider the standard finite-state grammars with one non-terminal symbol and two terminal symbols. Eliminating the null grammar, we have a set of 15 grammars over these vocabularies:

1) S ::= a

2) S ::= b

3) S ::= a | b

4) S ::= aS

5) S ::= a | aS

---

[1] As an interesting sidelight, it is worth mentioning that the sample was intended to represent the language with an even number of a's in each string. The author had inferred the following simple (but ambiguous) normal grammar

S ::= b | SS | AA

A ::= a | AS | SA

but the enumeration procedure quickly found two simpler grammars for the same language.

6) S ::= b | aS

7) S ::= a | b | aS

8) S ::= bS

9) S ::= a | bS

10) S ::= b | bS

11) S ::= a | b | bS

12) S ::= aS | bS

13) S ::= a | aS | bS

14) S ::= b | aS | bS

15) S ::= a | b | aS | bS

We can immediately eliminate 1), 2), 4), 5), 8), and 10) because they do not have the correct terminal vocabulary, $V_t = \{a,b\}$. Nine grammars remain; of these, 12) is not reduced. If the information sequence is <b,bb,aa,baa,...> , the first sample eliminates 9) and 13); the second, 3), 6), and 7); and the third, 11) and 14), leaving only 15) (the universal grammar) as DA. Three strings reduced the hypothesis space from eight grammars to one. Further strings can not lead to further deductive learning. Our general observation is that the restriction to DA grammars is most effective when (for N non-terminal grammars) the sample contains the strings of the language up to length 2N .

A Complete Inference Procedure

GRAMMARLIST was followed by EVALUATER, a program which would evaluate any fixed list of grammars on the basis of samples input from the terminal. The results were as predicted, and the only

particular interest of the program is that it ran very fast (a few seconds for the largest test case), confirming our belief that the Bayesean portion of our procedure is not a limiting factor; if the problem of efficiently enumerating only the DA grammars is solved our method should be quite practical.

The final implementation was a complete inference procedure, INFER, which incorporated grammar splitting, tests for ambiguity, and parameter learning. It is a rather large (550 lines of LISP) program, and thus consumes a substantial fraction of the available free storage itself, further slowing the system. We were able to verify correct operation of all its components on small grammars, but each attempt to infer more interesting (i.e., larger) grammars had to be terminated because of excessive time.

INFER is given an initial hypothesis space (generally the one non-terminal universal grammar) as a parameter. It accepts sample strings from the terminal, evaluates the active grammars, determines which (if any) of them should be split, splits them, and finally prints out the number of active grammars, the minimum value of complexity and the value of δ_k (the split level). If its guess has changed, it also prints its new guess. We present below a typical (3 minute) computer run. Input from the terminal is lower case, and follows the computer prompts (!), computer output is upper case[1/] and should be largely self-explanatory. The language consisted of odd-length strings of A's , the information sequence being

---

1/ Due to LISP restrictions we were forced to modify our convention that terminal symbols are lower case, non-terminal, upper. For this run, $V_t = \{A\}$, $V_n = \{S,X,Y,...\}$.

<A,AAA,AAAAA,A,A,AAA,AAAAAAA,A,A,A,AAA,A,AAA,AAAAA,...> .

```
! a
    (A)
    1 ACTIVE GRAMMARS.  BEST VALUE: 5.0. SPLIT LEVEL: 5.0
    TOTAL COMPLEXITY: 5.0, INTRINSIC: 4.0 UNSPLIT 6.0 1/
    S  ::= A | A S    (3 : 2 1)
    ----------
! aaa
    (A A A)
    SPLITTING
       TOTAL COMPLEXITY: 8.9, INTRINSIC: 4.0 UNSPLIT 6.0
       S  ::= A | A S    (6 : 3 3)
       ----------
          TOTAL COMPLEXITY: 15.1, INTRINSIC: 10.0 UNSPLIT 12.2
          S  ::= A | A X    (4 : 2 2)
          X  ::= A | A X    (4 : 2 2)
          ----------
          TOTAL COMPLEXITY: 16.9, INTRINSIC: 10.3 UNSPLIT 13.2
          S  ::= A | A X | A S    (7 : 3 1 3)
          X  ::= A X    (1 : 1)
          ----------
          TOTAL COMPLEXITY: 12.6, INTRINSIC: 9.0 UNSPLIT 11.2
          S  ::= A | A X    (5 : 3 2)
          X  ::= A S    (2 : 2)
          ----------
          TOTAL COMPLEXITY: 14.6, INTRINSIC: 10.0 UNSPLIT 12.2
          S  ::= A | A X    (5 : 3 2)
          X  ::= A | A S    (3 : 1 2)
          ----------
          TOTAL COMPLEXITY: 15.6, INTRINSIC: 11.0 UNSPLIT 13.9
          S  ::= A | A X    (5 : 3 2)
          X  ::= A X | A X    (3 : 1 2)
```

1/ The number following UNSPLIT is the minimum intrinsic complexity of a split of this grammar, i.e., the value the SPLIT LEVEL must exceed to justify splitting this grammar.

```
    (5 . NEW ACTIVES)
    6 ACTIVE GRAMMARS.  BEST VALUE: 8.9. SPLIT LEVEL: 6.9
! aaaaa
    (A A A A A)
    5 ACTIVE GRAMMARS.  BEST VALUE: 13.7. SPLIT LEVEL: 8.7
! a a aaa aaaaaaa
    (A)
    5 ACTIVE GRAMMARS.  BEST VALUE: 15.2. SPLIT LEVEL: 9.2
    (A)
    5 ACTIVE GRAMMARS.  BEST VALUE: 16.5. SPLIT LEVEL: 9.7
    (A A A)
    5 ACTIVE GRAMMARS.  BEST VALUE: 19.5. SPLIT LEVEL: 10.6
    (A A A A A A A)
    SPLITTING
      TOTAL COMPLEXITY: 24.8, INTRINSIC: 9.0 UNSPLIT 11.2
      S  ::= A | A X    (16 : 8 8)
      X  ::= A S    (8 : 8)
      ----------
        TOTAL COMPLEXITY: 32.2, INTRINSIC: 15.9 UNSPLIT 18.0
        S  ::= A | A X    (9 : 4 5)
        X  ::= A Y    (8 : 8)
        Y  ::= A | A X    (9 : 5 4)
        ----------
    (1 . NEW ACTIVES)
    SPLITTING
      TOTAL COMPLEXITY: 9.3, [1] INTRINSIC: 9.3 UNSPLIT 11.6
      S  ::= A | A X | A S    (3 : 1 1 1)
      X  ::= A    (1 : 1)
      ----------
        TOTAL COMPLEXITY: 40.6, INTRINSIC: 16.2 UNSPLIT 18.3
        S  ::= A | A Y | A X    (10 : 4 5 1)
        X  ::= A    (1 : 1)
        Y  ::= A | A Y    (16 : 5 11)
        ----------
```

[1] The TOTAL COMPLEXITY of an ambiguous grammar is invalid.

```
        TOTAL COMPLEXITY: 35.4, INTRINSIC: 15.9 UNSPLIT 18.0
        S  ::= A | A Y    (9 : 4 5)
        X  ::= A    (5 : 5)
        Y  ::= A Y | A X    (12 : 7 5)
        ----------
        TOTAL COMPLEXITY: 39.5, INTRINSIC: 17.9 UNSPLIT 20.3
        S  ::= A | A Y | A X    (10 : 4 5 1)
        X  ::= A    (5 : 5)
        Y  ::= A Y | A X    (12 : 7 5)
        ----------
        TOTAL COMPLEXITY: 34.0, INTRINSIC: 15.2 UNSPLIT 17.3
        S  ::= A | A Y | A X    (17 : 8 8 1)
        X  ::= A    (1 : 1)
        Y  ::= A S    (8 : 8)
        ----------
        TOTAL COMPLEXITY: 38.0, INTRINSIC: 16.2 UNSPLIT 18.3
        S  ::= A | A Y | A X    (17 : 8 8 1)
        X  ::= A    (1 : 1)
        Y  ::= A | A S    (9 : 1 8)
    (5 . NEW ACTIVES)
    11 ACTIVE GRAMMARS.  BEST VALUE: 24.8. SPLIT LEVEL: 12.0
    TOTAL COMPLEXITY: 24.8, INTRINSIC: 9.0 SPLIT
    S  ::= A | A X    (16 : 8 8)
    X  ::= A S    (8 : 8)
    ----------
! a a a aaa a aaa aaaaa
    (A)
    11 ACTIVE GRAMMARS.  BEST VALUE: 25.8. SPLIT LEVEL: 11.8
    (A)
    11 ACTIVE GRAMMARS.  BEST VALUE: 26.8. SPLIT LEVEL: 11.9
    (A)
    11 ACTIVE GRAMMARS.  BEST VALUE: 27.6. SPLIT LEVEL: 12.0
    (A A A)
    11 ACTIVE GRAMMARS.  BEST VALUE: 29.7. SPLIT LEVEL: 11.7
```

```
(A)
11 ACTIVE GRAMMARS.  BEST VALUE: 30.5. SPLIT LEVEL: 11.5
(A A A)
11 ACTIVE GRAMMARS.  BEST VALUE: 32.7. SPLIT LEVEL: 11.9
(A A A A A)
SPLITTING
   TOTAL COMPLEXITY: 47.9, INTRINSIC: 10.0 UNSPLIT 12.2
   S  ::= A | A X    (16 : 8 8)
   X  ::= A | A X    (24 : 8 16)
   ----------
(0 . NEW ACTIVES)
SPLITTING
   TOTAL COMPLEXITY: 40.7, INTRINSIC: 10.0 UNSPLIT 12.2
   S  ::= A | A X    (27 : 15 12)
   X  ::= A | A S    (13 : 1 12)
   ----------
      TOTAL COMPLEXITY: 48.2, INTRINSIC: 16.9 UNSPLIT 19.0
      S  ::= A | A X    (16 : 8 8)
      X  ::= A | A Y    (13 : 1 12)
      Y  ::= A | A X    (13 : 8 5)
      ----------
(1 . NEW ACTIVES)
12 ACTIVE GRAMMARS.  BEST VALUE: 36.0 SPLIT LEVEL: 12.3
```

The longest computer run (45 minutes) with INFER involved an attempt to "force" it to infer a finite-state grammar with three non-terminals, using an information sequence of the language $\{A^{3n+1} | n \geq 0\}$. On the basis of the sample <A,AAAA,A,AAAAAAA,A,A,A> the procedure enumerated the desired grammar

S ::= A | AX

X ::= AY

Y ::= AS

but found that it was still 3.7 bits more complex than the universal grammar. About two more strings would have reversed this evaluation, but the next string (AAAA) initiated further splitting which consumed the last two thirds of the run without further evaluation of the desired grammar.

There are two major flaws in the current splitting algorithm. First, whenever a grammar is split, all of its splits are immediately enumerated; if only the probable ones were initially enumerated the procedure would be much faster. Second, grammars are split "too soon." By splitting whenever it is possible that a split could be the best grammar, we frequently split when the sample is too small for the DA test to be fully effective. In practice, we have observed that it is always some time after a grammar is added to the hypothesis space before it becomes the guess. We conjecture that a good splitting heuristic would markedly improve performance.

PART III

FURTHER CONSIDERATIONS

IX. LEARNING RATES AND OPTIMAL PRESENTATION

External and Internal Measures

There are many reasons why we might wish to measure how much an inference procedure has learned at a particular time: so we could measure the effect of various modes of presentation or various hypothesis spaces; so a procedure could evaluate its performance; or so that we could demonstrate that one procedure was better than another. A measure based solely on the guesses made by the procedure (its external behavior) is of necessity gross -- it cannot reflect how close the procedure is to guessing the right answer; nor how sure the procedure is of its guess, once it has made the right guess. With Bayesean procedures, at least, we can do better by basing the measure on the a posteriori probabilities within the procedure.

Watanabe [1960] has proposed that an "entropy" measure be applied to hypothesis spaces, and indicates that the expected decrease of this measure results from learning by the inference procedure. We regard the suggestion as valid, and present here a generalization which seems to provide an adequate measure for learning. Its general utility remains to be evaluated in terms of practical results following from its use.

Def. IX.1. The ignorance of a Bayesean procedure after seeing the sample S_k, denoted $IG(S_k,C)$, is the amount of information which would be required to specify the best hypothesis to the procedure.

$$IG(S_k,C) = -\log[P(H_B|S_k,C)] \quad .$$

Def. IX.2. The effective learning of a Bayesean procedure on the basis of the sample S_k, denoted $EL(S_k,C)$, is the reduction in its ignorance.

$$\begin{aligned} EL(S_k,C) &= IG(\emptyset,C) - IG(S_k,C) \\ &= \log[P(H_B|S_k,C)/P(H_B|C)] \\ &= \log[P(S_k|H_B,C)/P(S_k|C)] \quad . \end{aligned}$$

The procedure cannot, of course, evaluate its own ignorance or effective learning without knowing the correct answer, so this measure must be evaluated externally.

Effective learning is additive, i.e., if $S_k = S_iS_j$ then

$$EL(S_k,C) = EL(S_i,C) + EL(S_j,S_iC) \quad .$$

However, effective learning need not have a positive value. This corresponds to the common sense notion that a valid observation may be misleading.[1/]

1/ Consider, for example, a "fair" coin that comes up "heads" on the first three flips, or an information sequence for $\{h^nt|n \geq 0\}$ which begins with the string hhht .

A procedure may introspectively estimate the values of its ignorance and effective learning, on the basis of the a posteriori probabilities of hypotheses.

Def. IX.3. The uncertainty[1] of a Bayesean procedure after seeing the sample S_k, denoted $U(S_k,C)$, is the expected value of its ignorance.

$$U(S_k,C) = -\sum_i P(H_i|S_k,C) \cdot \log[P(H_i|S_k,C)]$$

Def. IX.4. The apparent learning of a Bayesean procedure on the basis of the sample S_k, denoted $AL(S_k,C)$, is the expected value of effective learning.

$$AL(S_k,C) = \sum_i P(H_i|S_k,C)\cdot\log[P(H_i|S_k,C)] - \sum_i P(H_i|S_k,C)\cdot\log[P(H_i|C)] \quad .$$

In contrast to effective learning, apparent learning is always non-negative. Regardless of what its observations are, the procedure will never estimate that those observations have been misleading. Apparent learning is also non-additive and not even monotonic. The

[1] This is Watanabe's [1960] "entropy" measure.

value of apparent learning can return to zero after having been positive.[1]

The values of ignorance and uncertainty depend on whether we "lump" together stochastically equivalent hypotheses or treat them as separate. Fortunately, however, the values of both effective and apparent learning are independent of this choice. Such partitioning affects only the origin of the ignorance and uncertainty scales, and the learning measures are based on differences.

For any procedure which converges to the correct answer in the limit, both effective and apparent learning will converge to the initial value of ignorance (with stochastically equivalent hypotheses lumped).

Not all hypothesis spaces have finite uncertainties. We have, of course,

$$\sum_i P(H_i|C) = 1$$

and

$$\lim_{x \to 0} x \cdot \log x = 0$$

but these are not sufficient to guarantee that

$$\sum_i P(H_i|C) \cdot \log[P(H_i|C)]$$

---

[1] A simple example of this occurs when testing a coin which is known a priori to be biased 2/3 to 1/3. The hypothesis space consists of H_h ("heads" favored) and H_t ("tails" favored). A run of n "heads" causes apparent learning. A run of n "tails" causes apparent learning. But a run of n "heads" followed by a run of n "tails" causes no apparent learning.

converges.[1] Convergence will, in general, depend on the particular probability distribution. In some important cases, (e.g., a distribution defined by an ordered (or a linear) stochastic grammar-grammar) convergence is assured. The conditions for finite uncertainty of the language of a stochastic grammar and finite expected length of strings in the language are closely related [cf. Pohl 1967]. For an unambiguous stochastic grammar, the uncertainty of the language of any non-terminal symbol is the sum of the uncertainty of its alternatives and the weighted sum of the uncertainties of their languages.

Consider the grammar-grammar $\bar{G}_3$ from Chapter IV:

S ::= R | RR	(1/2, 1/2)
R ::= N "::=" P	(1)
P ::= A | P "|" A	(1/2, 1/2)
A ::= T | TN	(1/2, 1/2)
T ::= "a" | "b"	(1/2, 1/2)
N ::= "S" | "A"	(1/2, 1/2)

---

[1] Consider the case where for each n there are precisely $2^{(2^n-n)}$ hypotheses with probability $2^{-(2^n)}$. Now

$$\sum_{n=1}^{\infty} 2^{(2^n-n)} \cdot 2^{-(2^n)} = \sum_{n=1}^{\infty} 2^{-n} = 1 \quad ,$$

so this is a valid distribution, but

$$-\sum_{n=1}^{\infty} 2^{(2^n-n)} \cdot [2^{-(2^n)} \cdot \log_2(2^{-(2^n)})] = \sum_{n=1}^{\infty} 2^{-n} \cdot 2^n = \sum_{n=1}^{\infty} 1$$

which diverges rapidly.

If we use $\hat{X}$ to denote $U(L(X),\bar{G}_3)$, we have the system of equations

$$\hat{S} = -\frac{1}{2}\cdot\log[1/2] - \frac{1}{2}\cdot\log[1/2] + \frac{1}{2}\cdot\hat{R} + \frac{1}{2}\cdot(\hat{R}+\hat{R})$$

$$= 1 + \frac{3}{2}\hat{R}$$

$$\hat{R} = \hat{N} + \hat{P}$$

$$\hat{P} = 1 + \hat{A} + \frac{1}{2}\hat{P} \quad .$$

$$= 2 + 2\hat{A}$$

$$\hat{A} = 1 + \hat{T} + \frac{1}{2}\hat{N}$$

$$\hat{T} = 1$$

$$\hat{N} = 1$$

In $\bar{G}_3$ the non-terminal symbols are <u>ordered</u> in the sense that no non-terminal symbol produces any non-terminal symbol defined by an earlier rule. For any such grammar[1] the matrix of coefficients is essentially triangular and the (positive) solution of the equations is readily obtained by back substitution, e.g.,

---

[1] We need another condition which almost always holds in practice. If $p_n = (A,\omega,\theta_n)$ let $R(p_n)$ be the number of times A occurs in ω. Now we require that, for each rule, $\sum_{n=1}^{r} R(p_n)\cdot\theta_n < 1$. If this condition does not hold, each A is expected to produce at least one more A, and there will not be non-infinite positive solutions for $\hat{A}$.

$$\hat{N} = 1$$

$$\hat{T} = 1$$

$$\hat{A} = \frac{5}{2}$$

$$\hat{P} = 7$$

$$\hat{R} = 8$$

$$\hat{S} = 13$$

The initial uncertainty of the hypothesis space generated by $\bar{G}_3$ is 13 , and the average amount of information which an inference procedure must obtain from a sample in order to infer the correct grammar is 13 bits.[1/]

Optimum Samples and Learning Rates

There are several potential applications for the measures introduced in the previous section, none of which have been explored to any depth.

An optimum sample of size k for a best grammar G_B may be defined as the S_k which maximizes effective learning

$$EL_{opt}(k,C) = \max_{S_k \epsilon S_k(G_B)} [EL(S_k,C)]$$

$$= \log \left[\max_{S_k \epsilon S_k(G_B)} [P(S_k|G_B,C)/P(S_k|C)] \right] .$$

[1/] This may be compared with the values for G_4 and G_5 , that is, 15 and 23 bits respectively. Both these grammars are somewhat more complex than average. Note, however, that this average is computed on the basis of $L(\bar{G}_3)$ which contains many simple non-reduced grammars that are not really part of the hypothesis space.

Informally, this indicates that the optimum sample contains those strings which the best grammar is much more likely to produce in combination than is the average grammar in the hypothesis space. There are actually two components in this computation: deductive learning springs from the elimination of non-DA grammars from the hypothesis space

$$DL(S_k,C) = -\log \left[\sum_{G_i \text{ is DA}} P(G_i|C) \right]$$

and pure inductive learning is the balance of the effective learning, i.e., the learning due to the use of frequency information in Bayes' theorem to refine a posteriori probabilities

$$PIL(S_k,C) = EL(S_k,C) - DL(S_k,C) \quad .$$

Initially, we would expect that most of the probable grammars will not generate $L(G_B)$ and deductive learning will be the predominant element. Later, however, when all the remaining probable DA grammars generate $L(G_B)$, most of the learning will of necessity be pure inductive learning.[1/]

Any of our learning measures can be divided by k to obtain learning rate per string. Perhaps of more interest would be the rate per symbol, obtaining by dividing by the total length of the sample. It seems likely that for the former measure, optimum samples will contain mostly long strings, while for the latter they will almost certainly contain mostly short strings.

1/ Universal grammars, for example, are only rejected on the basis of pure inductive learning, since they are always DA.

An interesting open question is whether there exists an effective procedure for constructing an optimum sample. Another open question is whether an optimum sample of size $k+1$ always contains an optimum sample of size k. If so, we are assured of the existence of optimum information sequences. If not, the construction of a sequence will be affected by the size of sample which we wish to be optimum.

We are also interested in the expected value of learning, given a grammar (or class of grammars) and stochastic presentation. These expected values are easy to define, but cannot generally be placed in a closed form for easy evaluation.

$$E[EL(S_k,C)|G_B] = \sum_{S_k(G_B)} P(S_k|G_B,C) \cdot EL(S_k,C)$$

$$= \sum_{S_k(G_B)} P(S_k|G_B,C) \cdot \log\left[\frac{P(G_B|S_k,C)}{P(G_B|C)}\right]$$

$$= \sum_{S_k(G_B)} P(S_k|G_B,C) \cdot \log\left[\frac{P(S_k|G_B,C)}{\sum_{\ell} P(G_\ell|C) \cdot P(S_k|G_\ell,C)}\right]$$

$$= \sum_{S_k(G_B)} P(S_k|G_B,C) \cdot \log[P(S_k|G_B,C)]$$

$$- \sum_{S_k(G_B)} P(S_k|G_B,C) \cdot \log[\sum_{\ell} P(G_\ell|C) \cdot P(S_k|G_\ell,C)]$$

$$E[EL(S_k,C)|C] = \sum_i P(G_i|C) \cdot E[EL(S_k,C)|G_i]$$

$$= \sum_i P(G_i|C) \sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \log\left[\frac{P(S_k|G_i,C)}{\sum_\ell P(G_\ell|C)\cdot P(S_k|G_\ell,C)}\right]$$

$$= \sum_i P(G_i|C) \sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \log[P(S_k|G_i,C)]$$

$$- \sum_i P(G_i|C) \sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \log[\sum_\ell P(G_\ell|C)\cdot P(S_k|G_\ell,C)] \ .$$

In both cases, it is the $\log[\sum_\ell]$ term that is intractable. It depends in a very delicate fashion on how close the languages of the various grammars are, and not much can be said in general. The other term may be interpreted as follows

$$\sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \log[P(S_k|G_i,C)]$$

$$= \sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \sum_{\tau\epsilon V_t^+} \log[P(\tau|G_i,C)^{f(\tau,S_k)}]$$

$$= \sum_{S_k(G_i)} P(S_k|G_i,C) \cdot \sum_{\tau\epsilon V_t^+} f(\tau,S_k) \cdot \log[P(\tau|G_i,C)]$$

$$= \sum_{\tau\epsilon V_t^+} \log[P(\tau|G_i,C)] \cdot \sum_{S_k(G_i)} f(\tau,S_k) \cdot P(S_k|G_i,C)$$

$$= \sum_{\tau \epsilon V_t^+} \log[P(\tau|G_i,C)] \cdot P(\tau|G_i,C) \cdot k$$

$$= k \cdot U(L(G_i),C)$$

where

$$U(L(G_i),C) = \sum_{\tau \epsilon V_t^+} P(\tau|G_i,C) \cdot \log[P(\tau|G_i,C)]$$

which we recognize as the uncertainty of G_i's language. This term in the expected value of learning is therefore a constant per string.

BLANK PAGE

X. NOISE

Simple Stochastic Noise

To this point, we have assumed (in common with previous studies of grammatical inference) that the information sequence is completely reliable. This is a highly suspect assumption for real applications, where noise of some sort is almost always present: keypunchers make errors, microphones pick up background, stray tracks show up in bubble chambers. Humans can perform at least some inference in the presence of such noise, rejecting some strings as special cases and inferring a hypothesis which covers the others. This chapter is devoted to conditions under which our procedures can function correctly, even with an information sequence containing errors.

Probably the simplest assumption about noise which models a realistic situation is that each string in the information sequence has a known fixed probability of being incorrect, and that the distribution of noise strings is known a priori (perhaps itself specified by a stochastic grammar). If P_N denotes the probability of noise and $P(\sigma_k|G_N,C)$ is the probability of σ_k as a noise string, then

$$P(\sigma_k|G_i,P_N,C) = (1-P_N) \cdot P(\sigma_k|G_i,C) + P_N \cdot P(\sigma_k|G_N,C) \quad .$$

When $P(\sigma_k|G_i,P_N,C)$ is substituted for $P(\sigma_k|G_i,C)$, stochastically inequivalent grammars remain stochastically inequivalent if $P_N < 1$. All of our previous results about the procedure EB (Chapters V and VII) still hold (e.g., the procedure is still effective, still converges to the best grammar, etc.). As P_N increases, the conditional probabilities of strings for the various grammars become progressively more similar (identical when $P_N = 1$) and the learning rate will be correspondingly reduced.

If all strings can occur as errors -- and this is generally a realistic assumption -- the noise grammar will be universal. Furthermore, we might expect that the probability of noise strings would decrease with length at least as rapidly as for grammatical strings (e.g., geometrically). The simple one non-terminal finite-state universal grammar may provide an adequate model for many situations.

If P_N is sufficiently small, there will be very little error introduced by the approximation

$$P(\sigma_k|G_i,P_N,C) \approx \begin{cases} (1-P_N) \cdot P(\sigma_k|G_i,C) & \text{if } P(\sigma_k|G_i,C) > 0 \\ P_N \cdot P(\sigma_k|G_N,C) & \text{if } P(\sigma_k|G_i,C) = 0 \end{cases} .$$

This corresponds to encoding a string by means of the grammar being considered if that is possible, and using the universal grammar only for the exceptional strings.

Allowance for noise strings almost completely defeats the deductive preprocessing of Chapter VI, since we can no longer require that a grammar be DA with respect to the sample. In general, different

grammars will identify different subsets of the sample as the error strings; without knowing the correct grammar the deductive preprocessor cannot determine which subset to use in testing deductive adequacy. It might perhaps reject grammars which failed to generate some fraction of the sample, but this is no longer a sharp criterion: it will not reject grammars as quickly (and not nearly as safely) as the DA requirement.

The efficiency lost by failure of the DA criterion arises as soon as we admit the possibility of noise, independent of the magnitude of P_N. In other respects, however, our procedure will function nearly as well as before, if $P_N \ll 1$. It will learn at nearly the same rate, converge to the best grammar almost as quickly, etc. This is probably typical of the noisy situations we envision as applications. But what of the extreme situations with low signal-to-noise ratios (i.e., $P_N \to 1$)? In the limit (provided the noise distribution is known precisely) the relatively small deviations from the noise distribution due to the valid strings will become significant; by the results of Chapter V, the correct grammar will ultimately be chosen. However, it would be futile to place confidence in guesses based on a small sample. It may be more efficient to first collect a statistically significant sample, and then merely infer a grammar for those strings which occur significantly more often than explained by the noise distribution.

Estimating the Error Rate

In the previous section we required both the frequency and distribution of incorrect strings. This is a strong requirement, which may be difficult to meet in practice. For small samples and low error rates, of course, the precise value used for P_N will have little effect on our computation; we may choose to replace it with a bound (e.g., "less than one percent error strings").

As the sample size grows large, an incorrect value for P_N will introduce a systematic stochastic inequivalence between our estimated distribution for the best grammar and the sample, ultimately significant enough to cause the selection of another grammar. But the large samples also provide us with the means to refine our estimate for P_N, much as we estimated parameters in Chapter VII. An exact method can be constructed. However, here we merely sketch an approximation.

Assume that the noise grammar G_N is known and universal, and that it is known that the best grammar is not universal. Then for any allowable candidate grammar there will be some strings generated by the noise grammar and not by the candidate. If P_N is non-zero, these will occur in sufficiently large samples. If m of these have occurred by time k, denote them by $\sigma_{N_1} \ldots \sigma_{N_m}$, their joint frequency by $F_N(k) = \sum_{j=1}^{m} f(\sigma_{N_j}, S_k)$ and their joint probability by $P(N,k) = \sum_{j=1}^{m} P(\sigma_{N_j} | G_N, C)$. Now the expected value of $F_N(k)$ is $P_N \cdot k \cdot P(N,k)$, and we could use $F_N(k)/(k \cdot P(N,k))$ as a maximum likelihood estimator for P_N. However, we do have some a priori

knowledge about P_N -- we know it should be small and wish to reject grammars which require it to be large (e.g., .99) -- so we instead use $F_N(k)/P(N,k)$ as an estimator for the total number of error strings in the current sample, and use this value in the generalized Laplace rule developed in Chapter VII

$$E[P_N|C] = \frac{\beta_1 + 1 + F_N(k)/P(N,k)}{\beta_1 + \beta_2 + 2 + k} \quad .$$

β_1 and β_2 will generally be chosen on the basis of prior experience in a particular application: $\beta_1 = 0$, $\beta_2 = 100$ is a somewhat more precise formulation of "less than one percent error strings." As $k \to \infty$ this estimate will (for the best grammar) converge to P_N ; by the assumption that no candidate grammar is universal, all stochastically inequivalent grammars remain inequivalent, independent of their particular estimates of P_N ; therefore the results of Chapters V and VII still hold and the inference procedure will converge to the best grammar.

In any given application, statistics can be kept on the final value of $E[P_N|C]$ for the selected grammar in each run and used to refine the values of β_1 and β_2 . Finally, the strings identified as error strings over a large number of runs, together with their frequencies, can be themselves the subject of grammatical inference, to determine if there is a better noise grammar than the one previously assumed.

Implications of Noise

We have seen that noise of known distribution presents no formal difficulty to the enumerative Bayesean procedure. On the other hand, by eliminating the DA criterion, it represents a substantial practical obstacle. We believe that this is a reflection of the difficulty of the problem, rather than of any particular inadequacy of the procedure. In fact, we would argue that the ease with which this procedure may be extended to handle noise is evidence for its general validity. Neither the constructive methods of Solomonoff [1959] and Feldman [1967] nor the enumerative method of Gold [1967] will work in the presence of noise. Furthermore, since they do not incorporate frequency information or probabilistic estimates, it is doubtful that they could be generalized to handle noise realistically. Noise inescapably complicates inference, but it seems to complicate our procedure less than most.

XI. CONCLUSIONS AND DIRECTIONS FOR FURTHER RESEARCH

> It is probably no accident that a theory of grammatical structure can be so readily and naturally generalized as a schema for theories of other kinds of complicated human behavior. An organism that is intricate and highly structured enough to perform the operations that we have seen to be involved in linguistic communication does not suddenly lose its intricacy and structure when it turns to nonlinguistic activities.
>
> [Miller and Chomsky, 1963]

Summary

> The burden of proof falls squarely on those who champion a quantitative application of the calculus of probability to plausible reasoning. All that they have to do is to produce a class of non-trivial conjectures A for which the credibility Pr{A} can be computed by a clear method that leads to acceptable results in at least some cases.
>
> [Polya 1954]

> There are few areas of science in which one would seriously consider the possibility of developing a general, practical, mechanical method for choosing among several theories, each compatible with the available data.
>
> [Chomsky 1957]

We have stated the grammatical inference problem in a very general form (Chapter I). By specializing to stochastic grammars and

presentations we have established a number of new effectiveness and decidability results (Part II). The procedure EB provides a complete formal solution to that grammatical inference problem: it is effective, it makes the optimal choice at each step, converges to the best grammar, and tolerates noise of known distribution. The procedure has been implemented and operates as predicted, although it is computationally expensive, even with the improvements we have developed.

The principal obstacle to practical utilization of our results is the amount of computation involved in enumerating and rejecting the vast number of grammars which are not deductively acceptable. If an efficient deductive preprocessor can be devised, evaluation of the DA grammars by the procedure EB should not prove unduly expensive. Additionally, the enumerative Bayesean procedure serves as a yardstick against which heuristic procedures should be judged -- such a measure being notably absent from previous proposals for heuristic inference procedures.

The procedure in its present form (as a LISP program) is too slow for all but very small grammars; without further search limiting techniques and recoding into a more efficient language it is probably not economic for any of the envisioned applications. Yet some (e.g., grammars for spoken words, or for bubble chamber pictures) are not completely beyond its reach. Inference of grammars on the order of ALGOL 60 will require substantially different techniques, probably involving the learning of subgrammars for sublanguages.

Our results not only provide a sound basis for the solution of an interesting class of grammatical inference problems, but are more generally applicable. Our general procedure does not rely on unique properties of grammars. Rather, it is based on the probability structure which they impose on the observations. Its requirements are merely an enumeration of the hypothesis space which is effectively approximately ordered by a priori probability, conditional probabilities of observations with respect to hypotheses, and convergence of the samples with probability one. Any inference problem which meets these conditions can be solved by such a procedure, and our proof that optimal choices from infinite, non-parameterized hypothesis spaces can be made effectively should make our procedure attractive in a wide variety of applications.

This study has treated only a few of the many valid formulations of the grammatical inference problem, and has not exhausted the interesting questions there. In the balance of this chapter we discuss a variety of extensions, conjectures, and open questions which we believe deserve further attention.

Strategy and Efficiency Considerations

> We cannot seriously propose that a child learns the values of 10^9 parameters in a childhood lasting only 10^8 seconds.
>
> [Miller and Chomsky, 1963]

Although the enumerative Bayesean procedure presented in Chapter V and refined in later chapters is formally optimal, its Achilles' heel

is efficiency. As we indicate in Chapter VI, the enumerative problem is immense; our implementation in Chapter VIII can infer only grammars of extremely modest proportions within reasonable bounds on computation. A more realistic theory of inference should include computational cost in its definition of optimality, reflecting the fact that in most applications there are trade-offs among the cost of computation, the cost of further sampling, and the cost of guessing incorrectly.

Short of developing a general theory of the cost of inference, one might test various heuristics which lead to nearly optimal solutions at substantially lower cost. The constructive methods of Chapter II, for example, may provide a starting point. The residue analysis method could certainly be refined to base its merging decisions on a goodness measure, and -- where two choices have similar measures -- to be non-deterministic and produce a small set of grammars for evaluation. The question of how often these heuristic methods yield optimal grammars, and how close they average, should be carefully studied.

Heuristic methods may be supplemented by enumerative methods if they work well often enough to be attractive, but fail too often to be completely acceptable. This could be achieved either by enumerating and testing a few grammars at each step, or by using some criterion to determine when the heuristic method is not working well enough. We conjecture that the χ^2 test provides an adequate criterion: if the presentation is stochastically equivalent to the candidate grammar, the expected value of χ^2 is the number of distinct strings in the sample,

minus one; however, if it is inequivalent, the expected value of χ^2 is proportional to the total number of strings in the sample (k), so, for any confidence level, an incorrect grammar would finally be rejected by this test.

More work is needed on efficient enumeration techniques, particularly on techniques for enumerating only DA grammars. Additionally, restrictions of the classes of grammars appropriate to particular applications should be determined, so that grammars of practical importance may be inferred.

Hypotheses can be arranged in hierarchies. Rather than constructing an inference procedure strictly for finite-state grammars, or for linear grammars, etc. (or for normal grammars, 1-standard grammars, 2-standard grammars, etc.), it is reasonable to think of constructing an inference procedure for context-free grammars, whose meta-hypotheses are finite-state, linear, etc. (or normal, 1-standard, 2-standard), or even a general inference procedure whose meta-meta-hypotheses are context-free grammar, Turing machine, etc. This generalization can be easily incorporated into our formal structure if we are given a priori probabilities for each of the meta-hypotheses (and if their dependent hypotheses all meet our general conditions). But this involves searching all the hypothesis spaces in parallel to find the best hypothesis in any of them. It would be of interest to develop a procedure which searched only the most probable space (e.g., finite-state) until it concluded that it wasn't doing very well (e.g., if the sample contained only balanced parentheses) and then switched to another space.

Again, such a procedure would not generally locate the optimum answer at a given time, but might well be preferable on a total cost basis.

Variations of Presentation

From Chapter V onward, we have considered only stochastic text presentation. Most of the other forms of presentation discussed in Chapter III can be justified for some applications, and results analogous to those for stochastic text would be interesting. Both Gold [1967] and Feldman, et al. [1969] discuss presentations without an underlying probability structure. Although our results are not precisely comparable, the situation is roughly that complete text is much weaker than stochastic text, which is about as powerful as complete informant presentation.

A stochastic informant presentation could be incorporated directly into our enumerative Bayesean scheme. Negative instances reduce the class of DA grammars, positive instances are treated as before. There does not seem to be any reasonable way to utilize the frequencies of negative instances, except in the case of noise (Chapter X), where frequencies play an important role.

Gold [1967] discusses a reactive informant, which classifies strings proposed by the inference procedure as either sentences or non-sentences. Such an informant does not impose a frequency distribution on strings, and it seems hard to improve much on Gold's results for guessing by enumeration; ordering the enumeration of grammars by a priori probability (Feldman [1969] terms this occam's enumeration) should improve average behavior, however. An unreliable

(noisy) informant would seem to indicate a Bayesean analysis. Either form of reactive informant raises the question of an optimum inquiry strategy. Probably the best strategy is to maximize at each step the expected value of apparent learning (Chapter IX) but we have not found an effective method for this maximization; simplifications or approximations are called for.

The discovery procedure of Solomonoff [1959] involves both text and reactive informant presentation. Many different combinations might be studied. One which we conjecture would be of practical interest allows the alternation of text-like and reactive informant presentations, i.e., the informant gives a string and its category, then the procedure proposes a string which the informant categorizes, etc.

The heuristic constructive inference techniques which have been proposed all assume -- explicitly or implicitly -- that their samples are in some sense representative, e.g., that they contain all (or most) of the shortest strings in the language. This intuitively reasonable assumption can be made precise in terms of an underlying probability distribution (i.e., a stochastic grammar). However, in cases where the sample is deliberately constructed, rather than resulting from a random process, a human is more likely to create a (nearly) irredundant text, probable strings first, than to repeat strings with frequency proportional to their probability. This is somewhat analogous to an urn problem without replacement -- though of a funny sort, since when a string is drawn, all copies of that string are removed. The statistics for irredundant samples are formally identical to the Fermi-Dirac

statistics of statistical mechanics which apply to particles obeying the Pauli exclusion principle.[1] Thus, although the formulae are more complex, they do relate to an intensively studied class, and it may be that the relevant transformations or approximations have already been developed. We present the following conjectures: (1) a modified enumerative Bayesean procedure will converge to the best grammar for irredundant (Fermi-Dirac) text, (2) the expected learning rate per string will be higher for irredundant text than for stochastic text, (3) an approximate solution, which will converge to the best grammar, can be obtained by estimating the frequencies for strings on the basis of their position in the information sequence and/or their length, and then applying the method of Chapter VII (grammars with free parameters).

Learning Partial Grammars

> Students of infants and of language have long wondered over the fact that a structure of such enormous formal complexity as language is so readily learned by organisms whose available intellectual resources appear in other respects quite limited.
>
> [Braine 1963]

---

[1] It is tempting, but probably fruitless, to pursue such analogies: stochastic presentation obeys Maxwell-Boltzmann statistics (can we find Bose-Einstein presentations?), complexity is analogous to E/kT (how do we interpret the "temperature" of a presentation?), we have already discussed the entropy of hypothesis spaces and presentations (should we, with Watanabe [1960], regard the decrease in entropy of the hypothesis space during inference as a profound violation of the Second Law of Thermodynamics, or should we relate it to the increased entropy of the sample?), etc.

We noted in Chapter VI that the number of N non-terminal grammars grows roughly as 2^{N^3}. We have hopes that plausible improvements in deductive methods, restrictions on the form of grammars considered, etc., will make grammars with several non-terminals (and several terminal symbols, for that matter) inferrable with reasonable effort. It is clear, however, that grammars as large as the ALGOL 60 grammar (i.e., grammars involving scores of non-terminals and scores of terminal symbols) will not be made attainable simply by improving the deductive processing. There are too many plausible (and deductively acceptable) grammars which are simpler, to expect that the ALGOL 60 grammar would ever be reached in the enumeration; the sample size required to make that particular grammar preferable is staggering.

On the other side of the coin, there is no reason to believe that any human ever has (or ever will) successfully inferred a complete correct grammar for ALGOL 60 solely on the basis of a set of ALGOL 60 programs. Unless we wish to join the White Queen in "believing impossible things before breakfast," we need not set such difficult goals for our inference procedures. Miller [1963, 1966] has given some evidence that people do rather poorly at inferring even rather simple grammars. The reader may wish to verify this by seeing how long it takes him to infer a context-free grammar for the following sample (numbers in parentheses indicate frequency of occurrence of the corresponding strings):

ab	(256)
babaa	(64)
ababbb	(16)
bbabaaaa	(16)
abababba	(8)
bababbbaa	(4)
abbabaabb	(4)
babaaabbb	(4)
abababbbbb	(1)
ababbbabbb	(1)
babaaababba	(2)
abbabaaabba	(2)
ababbabaaba	(2)
bbbabaaaaaa	(4)
bababbbaaa	(2)

For definiteness we provide the following hint, which would not normally be given to an inference procedure: There is an unambiguous one non-terminal stochastic grammar whose language up to strings of length 11 is precisely this sample, with probabilities proportional to the given frequencies. It is instructive to consider how much the hint simplifies the problem (in particular, it provides a large number of negative strings), how one would procede in the absence of the information in the hint, and how large a sample is needed to justify selection of this grammar.

Nevertheless, children do acquire natural languages, and it is widely assumed that they do so by inferring a grammar [Chomsky 1957] [Gold 1967]. But adequate grammars (be they context-free or transformational) for natural languages are certainly more complex than

the ALGOL 60 grammar, and the range of observed natural languages is sufficiently large to require a rather rich hypothesis space -- probably much richer than anything we have considered in this study.

How are we to account for language acquisition by children? Or, more to the point, does language acquisition by children suggest means for improving our grammatical inference procedures? We believe that it does, and we conjecture that an important distinction between the child's experience and that we have assumed for our procedures is this: The child is not initially presented the full adult language he is ultimately expected to learn. Rather, he is confronted with a very limited subset, both in syntax and vocabulary, which is gradually (albeit haphazardly) expanded as his competence grows. There is evidence that children's early utterances are representable by very simple context-free grammars with few non-terminal symbols [Braine 1963] and a small terminal vocabulary.

Foreign languages are not normally taught by confronting the beginning student with the work of a great prose stylist. Nor is the student introduced to a programming language by presenting him with a representative sample of the programs for which that language is used. On the contrary, particularly simple constructs are introduced first. A portion of the vocabulary is established and then used in simple sentences (or program fragments). After these are firmly acquired, both vocabulary and syntax are enlarged, hopefully until the student is fluent in the entire language.

The point of this discourse is that we should not expect our inference procedures to perform well when confronted directly with complex

languages. We do not know how humans acquire languages, but human performance suggests the desirability of progressing from simple subsets to more complete presentations of a language. There are at least three ways in which we might simplify the task of the inference procedure: we could give it some rules which must be incorporated in its grammar (e.g., subgrammars for <identifier> and <arithmetic expression> which are common to many programming languages, or for <electron track> and <proton track> which are common to many bubble chamber events); we could expose it to sublanguages (e.g., spoken repetitions of a particular word), let it build grammars for them separately and then incorporate them into a larger grammar for the whole language; or we could incorporate non-terminals into sample strings (e.g., "He hit <direct object>"). Similarly, a responsive informant could answer questions involving non-terminals, or instead of responding "no" could give the closest valid string.

REFERENCES

[Aigner 1968] Dennis J. Aigner. Principles of Statistical Decision Making. New York: Macmillan Co., 1968.

[Braine 1963] Martin D. S. Braine. "The Ontogeny of English Phrase Structure: The First Phrase," Language, 39, 1 (1963).

[Chomsky 1957] Noam Chomsky. Syntactic Structures. The Hague: Mouton and Co., 1957.

[Chomsky and Miller 1957] N. Chomsky and G. A. Miller. Pattern Conception. Report No. AFRC-TN-57-57, August 7, 1957.

[Chomsky 1963] Noam Chomsky. "Formal Properties of Grammars," in Handbook of Mathematical Psychology. Vol. II. New York: John Wiley and Sons, Inc. 1963.

[David 1958] Martin Davis. Computability and Unsolvability. New York: McGraw-Hill, Inc., 1958.

[Feldman 1967] Jerome Feldman. First Thoughts on Grammatical Inference. Artificial Intelligence Memorandum No. 55, Computer Science Dept., Stanford University, August 1967.

[Feldman 1969] Jerome A. Feldman. Some Decidability Results on Grammatical Inference and Complexity. Artificial Intelligence Memorandum, Computer Science Dept., Stanford University, September 1969.

[Feldman, et al. 1969] Jerome A. Feldman, James Gips, James J. Horning, Stephen Reder. Grammatical Complexity and Inference. Technical Report No. CS 125, Computer Science Dept., Stanford University, June 1969.

[George 1968] James E. George. CALGEN -- An Interactive Picture Calculus Generation System. Technical Report No. CS 114, Computer Science Dept., Stanford University, December 1968.

[Ginsburg 1966] Seymour Ginsburg. The Mathematical Theory of Context-Free Languages. New York: McGraw-Hill, Inc., 1966.

[Gold 1967] E. Mark Gold. "Language Identification in the Limit," Information and Control, 10, pp. 447-474 (1967).

[Greibach 1965] Sheila A. Greibach. "A New Normal-Form Theorem for Context-Free Phrase Structure Grammars," Journal of the ACM, 12, 1 (January 1965), pp. 42-52.

[McKeeman 1966] W. M. McKeeman. An Approach to Computer Language Design. Technical Report No. CS 48, Computer Science Dept., Stanford University, August 1966.

[McKeeman, Horning, and Wortman 1970] W. M. McKeeman, J. J. Horning, and D. B. Wortman. A Compiler Generator. Englewood Cliffs: Prentice-Hall, Inc., 1970.

[Machol and Gray 1962] Robert E. Machol and Paul Gray (eds.). Recent Developments in Information and Decision Processes. New York: Macmillan Co., 1962.

[Miller and Chomsky 1963] George A. Miller and Noam Chomsky. "Finitary Models of Language Users," in Handbook of Mathematical Psychology. Vol. II. New York: John Wiley and Sons, Inc. 1963.

[Miller 1963, 1966] G. Miller and M. Stein. "Grammarama Memos," Unpublished internal memos, Harvard Center for Cognitive Studies, December 1963 and August 1966.

[Miller and Shaw 1968] W. Miller and A. Shaw. "Linguistic Methods in Picture Processing," Proceedings of AFIPS FJCC 1968, pp. 279-291.

[Nagel 1963] Ernest Nagel. "Introduction," in Henry E. Kyburg, Jr., and Ernest Nagel (eds.). INDUCTION: Some Current Issues. Middletown: Wesleyan University Press, 1963.

[Naur 1960] Peter Naur (ed.). Report on the Algorithmic Language ALGOL 60. Copenhagen: Regnecentralen, 1960.

[Paull and Unger 1968] Marvin C. Paull and Stephen H. Unger. "Structural Equivalence of Context-Free Grammars," Journal of Computer and System Sciences, 2. New York: Academic Press, 1968.

[Pohl 1967] Ira Pohl. Letter to the editor, Communications of the ACM, 10, 12 (December 1967), p. 757.

[Pohl, 1969] Ira Pohl. Bi-Directional and Heuristic Search in Path Problems. SLAC Report No. 104, Stanford Linear Accelerator Center, May 1969.

[Polya 1954] G. Polya. Mathematics and Plausible Reasoning. Vol. I: Induction and Analogy. Vol. II: Patterns of Plausible Inference. Princeton: Princeton University Press, 1954.

[Reddy 1966] D. R. Reddy. An Approach to Computer Speech Recognition by Direct Analysis of Speech Wave. Technical Report No. CS 49, Computer Science Dept., Stanford University, September 1966.

[Reynolds 1968] John C. Reynolds. Grammatical Covering. Technical Memorandum No. 96, Argonne National Laboratory, March 1968.

[Sandewall 1969] Eric J. Sandewall. "A Planning Problem Solver Based on Lookahead in Stochastic Game Trees," Journal of the ACM, 16, 3 (July 1969) pp. 364-382.

[Savage 1962] Leonard J. Savage. "Bayesean Statistics," in [Machol and Gray 1962].

[Schorre 1964] D. V. Schorre. "META II, A Syntax-Oriented Compiler Writing Language," Proceedings of the 19th National ACM Conference, 1964.

[Shamir and Bar-Hillel 1962] E. Shamir and Y. Bar-Hillel. Review 2476 in puting Reviews, 3, 5 (May 1962).

[Shaw 1968] Alan C. Shaw. The Formal Description and Parsing of Pictures. Technical Report No. CS 94, Computer Science Dept., Stanford University, April 1968.

[Solomonoff 1959] R. Solomonoff. "A New Method for Discovering the Grammars of Phrase Structure Languages," Information Processing, June 1959.

[Solomonoff 1964] R. J. Solomonoff. "A Formal Theory of Inductive Inference," Information and Control, 7, pp. 1-22, 224-254 (1967).

[Tribus 1962] Myron Tribus. "The Use of the Maximum Entropy Estimate in the Estimation of Reliability," in [Machol and Gray 1962].

[Vicens 1969] P. Vicens. Aspects of Speech Recognition by Computer. Technical Report No. CS 127, Computer Science Dept., Stanford University, April 1969.

[Watanabe 1960] Satosi Watanabe. "Information-Theoretical Aspects of Inductive and Deductive Inference," IBM Journal, April 1960.