Inductive Inference: Theory and Methods

DANA ANGLUIN

Department of Computer Science, Yale University, New Haven, Connecticut 06520

AND

CARL H. SMITH

Department of Computer Science, The University of Maryland, College Park, Maryland 20742

There has been a great deal of theoretical and experimental work in computer science on *inductive inference* systems, that is, systems that try to infer general rules from examples. However, a complete and applicable theory of such systems is still a distant goal. This survey highlights and explains the main ideas that have been developed in the study of inductive inference, with special emphasis on the relations between the general theory and the specific algorithms and implementations.

Categories and Subject Descriptors: F.1.0 [**Computation by Abstract Devices**]: General; I.2.2 [**Artificial Intelligence**]: Automatic Programming—program synthesis; I.2.6 [**Artificial Intelligence**]: Learning—*induction*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Inductive inference, learning by example

INTRODUCTION

The term "inductive inference" denotes the process of hypothesizing a general rule from examples. For example, given the strings

011, 000011, 00111, 0001, 0011,

one could conjecture that they are characterized by the rule

any number of 0*'s followed by any number of* 1*'s.*

This kind of process seems to be a fundamental and ubiquitous component of intelligent behavior. But a formal study of these processes raises a number of questions, among them:

- How can we evaluate inference methods?
- What are good criteria for comparing hypotheses?
- Can theoretical models of inference methods be useful?

Most of the work in inductive inference can be viewed as attempts to answer these questions.

The scope of this survey encompasses work done within the general paradigm of inductive inference established by Gold in his fundamental paper [Gold 1967], as described in subsequent sections. There are two branches in inductive inference research: one is concerned with the general theoretical properties of inference methods, and the other with finding specific, practical inference methods. We shall describe and compare the major ideas and results from both branches. Even within this limited scope, we are bound to omit significant ideas, issues, and references, for which we ask the reader's forbearance.

It is useful not to confuse inductive inference with learning, although induction may be used in learning. *Webster's Dictionary* indicates the spirit of this distinction

CONTENTS

in its definitions. “To learn” is “to gain knowledge, understanding, or skill by study, instruction, or experience,” or more broadly, “to come to be able to.” In contrast, “induction” is defined as “the act, process, or result of an instance of reasoning from a part to a whole, from particulars to generals, or from the individual to the universal.” Researchers in artificial intelligence are investigating many aspects of the broader topic of learning. The survey article by Dietterich et al. [1982] divides the topic of learning into four areas: rote learning, learning by being told, learning from examples, and learning by analogy. Of these, learning from examples overlaps with inductive inference, but there are some differences in emphasis. Often the work in artificial intelligence is more concerned with cognitive modeling than the work in inductive inference, and less concerned with formal properties such as convergence in the limit or computational efficiency. The collection of articles edited by Michalski et al. [1983] covers a number of issues and recent developments in machine learning, and provides an extensive bibliography on the subject.

Another area related to inductive inference is the formal study of language acquisition within linguistics. Wexler and Culicover [1980], for example, describe a set of constraints on transformational grammars which guarantee that they may be learned with probability 1 from pairs consisting of a deep-structure parse and the surface sentence derived from it, where the degree of sentence embedding is at most 2. They emphasize use of learnability to suggest detailed restrictions on the possible grammars of natural languages; most of the payoff seems to be in linguistics rather than inductive inference.

Other surveys covering some of the same material as the present one are Biermann and Feldman’s [1972a] survey of grammatical inference, Fu and Booth’s [1975] survey of the inference of deterministic and stochastic grammars and its use in structural pattern recognition, the survey by Klette and Wiehagen [1980] of East German work in inductive inference, D. Smith’s [1982] survey of inferring LISP programs from examples, and Bundy and Silver’s [1981] brief survey of rule learning programs. The comprehensive paper by Case and C. Smith [1983] gives a useful account of many of the theoretical results in inductive inference. See also the books by Fu [1975, 1977, 1982] and Gonzalez and Thomason [1978] on structural pattern recognition and the need for grammatical inference.

1. BASICS

1.1 An Example

We start with a concrete example intended to help the reader interpret the abstract concepts described later. Consider the typical IQ test problem of guessing the next term of the numerical sequence 3, 5, 7. A reasonable guess of the next term is 9, and we might be quite annoyed to find that the test maker had intended the sequence of odd prime numbers, so that the "correct" next term is 11 instead of 9. Yet what is the basis of this annoyance? There are infinitely many more or less plausible rules for generating different sequences that begin with the three terms 3, 5, 7. Is "the odd numbers starting with 3" somehow preferable to all the others on grounds of simplicity?

Suppose instead that we imagine a somewhat different setting. Before us stands a black box (or computer terminal) that will play a game with us. It will display the first element of a numerical sequence, and we will guess the second element and type it in. The black box will then display YES or NO, according to whether our guess was correct, and also display the correct first and second elements. We then guess the third element and type it in, and so on. A possible sequence of interactions is

It displays 2.
We guess 3 is next.
It displays *YES*: 2, 3.
We guess 4 is next.
It displays *NO*: 2, 3, 5.
We guess 7 is next.
It displays *YES*: 2, 3, 5, 7.
We guess 11 is next.
It displays *YES*: 2, 3, 5, 7, 11.

And the interaction continues similarly.

A plausible criterion of success is whether we eventually "catch on" to the rule being presented, that is, whether after a certain point our guesses of the next element are always correct. Of course, if the black box keeps changing the underlying rule, we may make infinitely many wrong guesses. Thus we restrict the box to select a rule at the start of play and not change it thereafter.

Can we always beat the box in this setting? That seems to depend on the class of rules the box is allowed to select from. Suppose that the box starts with the rule

The first term is 1 *and*
each subsequent term is 1 *greater*
than the guess of that term.

Then every single guess will be off by precisely -1. This may not seem like an acceptable rule for the box to select; we shall clarify subsequently in what sense it may be considered legitimate.

Let us now restrict the box still further so that it chooses a rule such that the term at position n of the sequence is expressible as a fixed polynomial in n with positive integer coefficients. One such sequence, specified by the polynomial $n^2 + 3$, begins

$$4, 7, 12, 19, 28, 39, \ldots.$$

Now can we beat the box? The answer is yes, and we shall demonstrate two different methods, which illustrate concepts presented later.

The first method is very general and powerful, and very crude. We begin to enumerate systematically all the polynomials of one variable with integer coefficients. We could proceed by enumerating for $d = 0, 1, 2, \ldots$ the finite number of $(d+1)$-tuples of integers in the range $[-d, d]$ and using them as the coefficients of $1, n, n^2, \ldots, n^d$. One such enumeration begins

$$0,\ 1+n,\ 1,\ 1-n,\ n,\ 0,\ -n,$$
$$-1+n,\ -1,\ -1-n,$$
$$2+2n+2n^2,\ 2+2n+n^2, \ldots.$$

Then, when required to guess the next term given the initial terms $y_1, y_2, \ldots, y_t$, we find the first polynomial, say $p(n)$, in this enumeration such that $p(n) = y_n$ for $n = 1, 2, \ldots, t$, and we guess $p(t+1)$.

To see that this strategy always succeeds, suppose that the box initially selects the polynomial $q(n)$, where

$$q(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_r n^r.$$

When d exceeds the maximum of the absolute values of all the a_i's and r, the polynomial $q(n)$ appears in the enumeration for

d. Consider the first occurrence of $q(n)$ in our enumeration. All of the polynomials preceding it are different from $q(n)$ for some values of n, so when t is large enough, $q(n)$ is the first polynomial in our enumeration that agrees with all the values $y_1, y_2, \ldots, y_t$. At this point and at every subsequent point we use $q(n)$ to predict the next element, and so our guesses will all be correct from this point on.

This enumeration method, which searches systematically through the whole space of possible rules eliminating those that disagree with the data until it finds one that agrees with the data, searches through more than d^d different polynomials to arrive at the correct degree d explanation. A much more expeditious method is to take the initial segment $y_1, y_2, \ldots, y_t$ of the sequence, interpolate a polynomial of degree at most $t - 1$ through these points, and use this polynomial to predict the next term. This approach will also succeed in the limit, with much less computational cost than the preceding method. In particular, this method uses a number of arithmetic operations polynomial in d to converge to a rule of degree d, whereas the enumeration method uses a number of operations that are exponential in d. This suggests that the size of the search space (typically exponential) is not the only factor determining the existence of efficient inference methods.

Although the second method is much more efficient, it is also special to the domain. If we were to add to the set of possible rules sequences that are sums of integer exponentials in n, for example, 2^n or $3^n + 5^n$, it is immediately clear how to modify the first method to enumerate functions of this type so that it continues to beat the box, but it is much less straightforward to modify the second method for this case.

Note also that with neither method can we conclude that we have reached the point where our guesses will continue to be correct. After seeing, say,

$$0, 0, 0, 0, 0, 0,$$

we might be tempted to conclude that this polynomial sequence is always 0, but instead it might be generated by

$$(n-1)(n-2)(n-3) \times (n-4)(n-5)(n-6),$$

so that the next term is 720 instead of 0. Without a bound on the degree of the polynomial being exemplified, there is no way to tell when we have successfully pinned it down.

Our example illustrates several of the concepts formalized in inductive inference, and we shall refer to it in the discussion that follows.

1.2 Two Concepts

Gold's [1967] theoretical study of language learning introduced two fundamental concepts that have been very important in subsequent theoretical work in inductive inference: *identification in the limit,* and *identification by enumeration.*

1.2.1 Identification in the Limit

Identification in the limit views inductive inference as an infinite process. Then the eventual or limiting behavior of an inductive inference method may be used as one criterion of its success. In our example, the game of guessing the next element of the sequence is prolonged indefinitely, and success is defined as always making the correct guess after some finite number of errors.

More abstractly, suppose that M is an inductive inference method that is attempting to describe correctly some unknown rule R. If M is run repeatedly on larger and larger collections of examples of R, an infinite sequence of M's conjectures is generated, say, $g_1, g_2, g_3, \ldots$. If there exists some number m such that g_m is a correct description of R and

$$g_m = g_{m+1} = g_{m+2} = \cdots,$$

then M is said to identify R correctly in the limit on this sequence of examples.

M may be viewed as learning more and more about the unknown rule R and successively modifying its conjecture about R. If after some finite time M stops modifying its conjecture and the final conjecture is a correct description of R, then M correctly identifies R in the limit on this sequence of examples. Note that M cannot determine whether it has converged to a correct hypothesis, since new data may or may not conflict with the current conjecture.

1.2.2 Identification by Enumeration

Identification by enumeration is an abstraction of our first method for guessing polynomial sequences, namely, systematically searching the space of possible rules until one is found that agrees with all the data so far. Suppose that a particular domain of rules is specified, and there is an enumeration of descriptions, say, d_1, d_2, $d_3, \ldots$, such that each rule in the domain has one or more descriptions in this enumeration. (This corresponds to the enumeration of polynomials in our example.) Given any collection of examples of a rule, the method of identification by enumeration goes down this list to find the first description, say d_i, that is compatible with the given examples and then conjectures d_i. (In our example, since the problem was to guess values, not rules, we used the polynomial to generate the next value.)

This method may or may not achieve correct identification in the limit, and may or may not be computable. If the presentation of examples and the compatibility relation satisfy the following two conditions, then the enumeration method is guaranteed to identify in the limit all the rules in the domain:

(1) A correct hypothesis is always compatible with the examples given.
(2) Any incorrect hypothesis is incompatible with some sufficiently large collection of examples and with all larger collections.

In order that the enumeration method be computable, the enumeration $d_1, d_2, d_3, \ldots$ must be computable, and it must be possible to compute whether a given description and a given collection of examples are compatible. These conditions were all satisfied in our polynomial example, giving another demonstration of its correctness in the limit.

The method of identification by enumeration is very general and powerful but also rather impractical because the size of the space that must be searched is typically exponential in the length of the description of the correct guess. Some improved variants of identification by enumeration are described in Section 7.

1.3 Specifying Inference Problems

Researchers have studied inductive inference problems for a variety of types of objects including LISP programs, finite-state machines, tree grammars, Turing machines, logical expressions, and stochastic context-free grammars, using various representations of sample inputs and inferred rules. To define an inductive inference problem, five items must be specified:

(1) the class of rules being considered, usually a class of functions or languages;
(2) the *hypothesis space*, that is, a set of descriptions such that each rule in the class has at least one description in the hypothesis space;
(3) for each rule, its set of *examples*, and the sequences of examples that constitute *admissible presentations* of the rule;
(4) the class of inference methods under consideration;
(5) the criteria for a successful inference.

The problem in our example, the extrapolation of polynomial sequences, could be specified as follows. The rule space is the set of all functions f from the positive integers to the integers such that $f(n)$ is some fixed polynomial in n with integer coefficients. The hypothesis space is the set of polynomials in one variable with integer coefficients. (Since this is a problem of prediction rather than identification, the hypothesis space is immaterial, but see the next example.) For each rule f, its set of examples consists of pairs of the form $\langle n, f(n) \rangle$, where n is a positive integer. There is only one admissible presentation of f, namely, the infinite sequence

$$\langle 1, f(1)\rangle, \langle 2, f(2)\rangle, \langle 3, f(3)\rangle, \ldots,$$

which we abbreviate as

$$f(1), f(2), f(3), \ldots.$$

We were not very explicit about the class of inference methods in the example, but a reasonable restriction might be that any method be representable by a computer program taking as input a finite sequence of integers, always halting, and giving a single integer as output. The criterion we used was that a method M succeeds on a rule f, provided that there exists some integer N such that for all $n \geq N$,

$$M(f(1), f(2), \ldots, f(n)) = f(n + 1).$$

To further illustrate these ideas, let us formalize a problem of guessing regular expressions from positive and negative data. Suppose we are given the facts that the strings

$$0011, 000011, 0000, 011, 00, 000000,$$

are in the regular set L, and that

$$0010, 0, 00110, 111, 0001111, 00000,$$

are not in L. A plausible conjecture might be that L consists of strings that are either an even number of zeros or any number of zeros followed by two ones, defined by the regular expression

$$(00)^* + 0^*11.$$

We can formally define the inference problem as follows. The class of rules is all the regular sets over the alphabet $\{0, 1\}$. The hypothesis space is all regular expressions over the same alphabet. (There are various other hypothesis spaces that we could use, namely, deterministic finite-state acceptors, nondeterministic finite-state acceptors, or context-free grammars. Note that the hypothesis space must contain descriptions of all the rules, but it may contain descriptions of other things as well.) An example of a regular language L is a pair $\langle s, d \rangle$ such that d is Y or N according to whether the string s is in L or not. An admissible presentation of L is an infinite sequence of examples of L such that every string over the alphabet $\{0, 1\}$ appears in some example in the sequence. So an admissible presentation of the language of the expression 0^*11 might begin

$$\langle 00011, Y \rangle, \langle 00, N \rangle, \langle 11, Y \rangle, \langle 011, Y \rangle,$$
$$\langle 1, N \rangle, \langle 0111, N \rangle, \langle 1, N \rangle, \ldots.$$

(Note that repetitions are permitted.) We consider methods that can be represented by a computer program that takes a finite initial segment of a presentation of any regular language as input, always halts, and produces as output a regular expression. We choose identification in the limit as the criterion of success, that is, a method M identifies a language L if and only if for every admissible presentation

$$\langle s_1, d_1 \rangle, \langle s_2, d_2 \rangle, \langle s_3, d_3 \rangle, \ldots$$

of L, if E_n is the expression produced by M on the initial segment of the first n terms of this sequence, then there exists N such that $E_n = E_N$ for all $n \geq N$, and moreover, the language of E_N is L. The reader should easily be able to construct a variant of identification by enumeration that successfully identifies in the limit all the regular sets over the alphabet $\{0, 1\}$ (recalling that the question of whether a regular expression generates a given string is effectively decidable.) It would be nice to have a more efficient method, by analogy with the use of polynomial interpolation in our first example, but no similarly efficient method has been found for this problem, and there is some evidence that none may exist (see Section 5.1).

At this point the reader should have some idea of the basic Gold paradigm of inference in the limit, and some inkling of the variety of ways in which inference problems may be posed. In the next section we discuss the specification of rules, hypothesis spaces, and data presentations. In subsequent sections we examine various possible restrictions on inference methods, criteria for successful inference, general and practical inference methods, and applications.

2. DOMAINS, GUESSES, AND EXAMPLES

We now consider various settings for inductive inference problems, that is, types of rules, along with their hypothesis spaces and data presentations. The types of rules that we consider are functions, languages and predicates, and stochastic languages.

2.1 Functions, Sequences, and Traces

A class of functions is specified together with two sets, the domain of argument values, and the range of function values. The functions may be either *total* (defined on all argument values) or *partial* (not necessarily defined on all argument values). The functions considered are computable, and the typical hypothesis space is a class of programs, such as the class of all Turing machines or the class of all LISP programs satisfying a certain syntactic restriction.

If f is a function, the *graph* of f is the set of all ordered pairs $\langle x, f(x) \rangle$ such that f is defined on argument x. An *example* of a function is an element of its graph. An *admissible presentation* of a function is an infinite sequence containing all and only the elements of the graph of the function. For instance, if f is the function that extracts the last element of a list, one presentation of it begins

$$\langle (a\ b\ c), c \rangle, \langle (e\ d), d \rangle, \langle (a\ e\ i\ o), o \rangle, \ldots .$$

Some of the questions in function inference concern the effects of various assumptions about admissible presentations. A reasonable restriction is to limit presentations to those that are computable, or computable within some complexity bound. Another common restriction is to specify that the examples in a presentation be given in the sequence determined by a fixed total ordering of the domain. Unless we specify otherwise, we shall use the general definition of admissible presentation given above. We discuss and compare other presentations in Section 4.3.

An obvious way to present a total function f defined on the natural numbers is to identify it with the sequence of its result values,

$$f(0), f(1), f(2), \ldots,$$

and define this sequence to be the unique admissible presentation of f. The *sequence inference problem* is to infer classes of functions presented in this fashion. Another problem for sequences is to predict the next term (rather than identifying the rule) from finite initial segments of the sequence. Predicting the next value of a sequence is also called extrapolation and is the problem that we considered in the first example in Section 1. Some early systems for identifying or predicting sequences are described by Fredkin [1964], Persson [1966], Pivar and Finkelstein [1964], Pivar and Gord [1964], and Simon and Kotovsky [1963].

In the following discussion, we use the term "inference" to refer to both identification and prediction. We explore the issue of prediction versus identification in Section 4.2.1.

The phrase "inference from input/output behavior" focuses attention on the programs or automata computing a function rather than on the function itself. From this perspective, it is reasonable to consider inference problems that give additional information about the programs or automata, usually designated as "trace information." An example is the problem of inferring a Turing machine from a sequence of snapshots giving the tape contents and head positions before and after every step of execution on some input. Another type of trace information is the sequence of instructions executed by an unknown program running on a given input. Biermann has developed a general methodology for using trace information that has been applied to program synthesis in a variety of domains [Biermann, 1972, 1978; Biermann and Krishnaswamy 1976; Biermann and Smith 1977; Biermann et al. 1975]. Barzdin [1974a] has also considered the problem of synthesizing programs from traces.

In the case above of Turing machine traces, it may initially seem that the trace information is just the input/output behavior of a different function, namely, the transition function of the machine. However, it may happen that two instances of the same snapshot in the execution sequence have different successors because the unknown machine is in different control states in the two instances. Thus the sequence of snapshots may not be argument/value pairs for any function at all. In fact, since sequences of snapshots contain no state information, they constitute an incomplete specification of the input/output behavior of the Turing machine's transition function. On the other hand, the sequences of snapshots contain more information about the underlying machine than input/output pairs. Thus inference from trace information is not reducible to inference from input/output behavior.

2.2 Languages and Predicates

A class of languages is specified by a universal set S and a collection of subsets of S. As an example, consider the class of all regular languages over the alphabet $\{a, b\}$; in this case, S is the set of all strings of a's and b's, that is, $\{a, b\}^*$. The hypothesis space for a class of sets is typically a class of formal grammars (e.g., the right linear grammars) or a class of acceptors (e.g., the class of all deterministic finite-state automata), although other types of descriptions, such as the regular expressions, are possible. The term *grammatical inference* is often used to describe the inference of a class of languages described by grammars. The sets we consider can be enumerated by some effective process; that is, they are recursively enumerable.

In the case of inferring an unknown language L, there is an important distinction between giving only positive information (members of L) and giving both positive and negative information (both members and nonmembers of L). A *positive presentation of* L is an infinite sequence giving all and only the elements of L. A *complete presentation of* L is a sequence of ordered pairs $\langle s, d \rangle$ from $S \times \{0, 1\}$ such that $d = 1$ if and only if s is a member of L, and such that every element s of S appears as the first component of some pair in the sequence. A positive presentation eventually includes every member of L, whereas a complete presentation eventually classifies every element of S as to its membership in L. Clearly, a complete presentation can be used effectively to construct a positive presentation, but not vice versa. If the language being presented is all strings of balanced parentheses, a positive presentation might begin

$$(())(),\ ()(),\ ((()))(()),\ (())((()()())),$$
$$((())),\ ()(),\ \ldots$$

and a complete (positive and negative) presentation might begin

$$\langle ()(, 0\rangle, \langle (()), 1\rangle, \langle ((((, 0\rangle,$$
$$\langle)))(, 0\rangle, \langle ()()(), 1\rangle, \langle (), 1\rangle, \ldots.$$

Another method of presenting a language L is *presentation by informant*. An informant is an oracle that the inference method may query concerning the language L. The inference method proposes some string s, and the oracle answers "yes" or "no" according to whether s belongs to L. In a very abstract sense, presentation by informant is equivalent to complete presentation, but in practice algorithms using queries tend to be quite different from algorithms using given data and no queries. Gold [1967] defined positive and complete presentation and presentation by informant and studied their effects on inferability (described in Section 4.3).

Pao and Carr [1978] consider a mixture of given data and queries in identifying regular sets. Their inference algorithm is initially given a structurally complete sample and then makes queries to complete the identification. (For a regular set L, a structurally complete sample is one that exercises every transition in the minimum acceptor for L.) Angluin [1982a] demonstrates the improvement in efficiency that this form of mixed presentation permits. It would be interesting to see whether the ideas of mixed presentation and structurally complete samples generalize usefully to other settings, for example, the context-free grammars.

A language L may be associated with the predicate P that is true for all s in L and false for all s not in L. Consequently, inferring a language may be thought of as inferring a single unary predicate. The problem of inferring several predicates of arbitrary arity simultaneously may be defined by a straightforward generalization of the notions of examples and presentations. Shapiro's Model Inference system [Shapiro 1981a, 1981b], for instance, infers Prolog programs corresponding to the Peano axiomatizations of the predicates plus(x, y, z) and times(x, y, z) over the unary numbers.

2.3 Stochastic Languages

A stochastic language is a language together with a probability distribution on its elements. The hypothesis space is usually a class of stochastic grammars. A stochastic grammar has probabilities associated with alternative productions, which are used to define the probability that the grammar derives a given string. An *example* of a stochastic language is an element of the language, and a *presentation* is any infinite sequence of examples. The presentations of a given stochastic language are assumed to be generated by random selection from the language according to the probabilities associated with its elements. Under this assumption, the frequency of a given value in initial segments of a presentation may be used to estimate its probability in the language. A frequently used criterion of success is identification in the limit with probability 1, which filters out the effects of statistically aberrant presentations. Inference of stochastic languages has been studied by Cook et al. [1976], Gaines [1976], Horning [1969], Maryanski and Booth [1977], and Van der Mude and Walker [1978].

3. CRITERIA FOR EVALUATING INFERENCE METHODS

3.1 What Is an Inference Method?

Intuitively, an inference method is a computable process of some kind that reads in examples and outputs guesses from the hypothesis space. We can define inference methods formally by using a modification of some standard model of computation. Turing machines equipped with a special input instruction (used to request the next example) and a special output instruction (used to output the value of a guess) provide one such model. The Turing machine model, and others as well, may be restricted in a variety of natural ways, such as requiring that each guess be compatible with the examples read in up to the time that the guess is made (the *consistency* property). The effects of this and other restrictions have been the focus of considerable study. We survey some of this work in Section 4.4.

3.2 Criteria of Success: Scope, Power, and Inferability

An important component in specifying an inference problem is the particular criterion of success chosen. We have discussed identification in the limit, but many other criteria have been studied. We shall now describe a framework for comparing various inference criteria.

Let C be a particular criterion of correct inference, D a particular type of data presentation, and I a particular class of inference methods. For any method M from I, let the *scope* of M with respect to C and D be the class of rules that M correctly infers with respect to the criterion C, using the method of data presentation D. A method M_1 is *more powerful* than a method M_2 with respect to C and D if and only if the scope of M_1 properly contains the scope of M_2.

A set U of rules is *inferable* with respect to I, C, and D if and only if U is contained in the scope of some method M from I with respect to C and D. Denote the class of all such sets U by $\text{Inf}(I, C, D)$.

Note that the preceding definition permits the inference method M to succeed on more than just the rules in U. An alternative definition that eliminates this possibility is the following. A set U of rules is *exactly inferable* with respect to I, C, and D if and only if U is exactly equal to the scope of some method M from I with respect to C and D. Denote the class of all such sets U by $\text{XInf}(I, C, D)$.

One way to compare various inference criteria, methods of data presentation, and types of inference methods is to compare the corresponding classes $\text{Inf}(I, C, D)$ and $\text{XInf}(I, C, D)$, which are called *identification types*. We describe some results on comparing identification types in Section 4.

3.3 Quality of Hypotheses

Since the inference process is potentially infinite, a person using a given inference method to identify an unknown rule usually cannot tell whether the method has converged. Thus correct identification or prediction in the limit appears to be a rather weak criterion of success in practice, and it may be desirable to augment it with some guarantee of the quality of hypotheses produced at finite stages of the process. Consequently, there has been much work on measures of "goodness" of hypotheses with respect to samples, and on methods of finding hypotheses that are optimal with respect to these measures. Such measures have been designed to capture the "simplicity" of the hypothesis and its "goodness of fit" to the finite set of sample data seen so far. We examine measures of this kind in Section 5.

3.4 Data Efficiency

We can measure the efficiency of an inference method by the quantity of data it requires to converge to a correct hypothesis. If M is an inference method and

$$x = x_1, x_2, x_3, \ldots$$

is an admissible presentation of a rule that M correctly infers, we define the *convergence point* of M on x to be the least positive integer n such that M outputs a correct hypothesis before reading x_{n+1} and does not subsequently change its guess. If M_1 and M_2 are two inference methods, than M_1 is as *data efficient* as M_2 if and only if for any admissible presentation x of any rule that M_2 identifies, M_1 also identifies the rule and the convergence point of M_1 on x does not exceed the convergence point of M_2 on x. M_1 is *strictly more data efficient* than M_2 if and only if M_1 is as data efficient as M_2 and there exists a presentation x of a rule that M_2 identifies such that the convergence point of M_1 on x is strictly less than the convergence point of M_2 on x. Finally, a method M is *optimally data efficient* if and only if no other method is strictly more data efficient than M.

Gold [1967] has shown that identification by enumeration is an optimally data efficient method. Jantke and Beick [1981] have extended Gold's result to show that the sets of recursive functions identifiable by optimally data efficient methods are precisely those identified by consistent and class preserving identification methods. (Definitions of these terms can be found in Section 4.4.)

3.5 Numbers of Hypotheses, Mind Changes, Errors

Another measure of efficiency is the number of times an identification method changes its hypothesis en route to a successful inference. Two quantities have been studied in this connection. If x is an admissible presentation of some rule and M some identification method, we may define $\mathrm{DH}(M, x)$ as the number of *distinct hypotheses* output by M in the course of reading the sequence of examples x and $\mathrm{MC}(M, x)$ as the number of *mind changes*, that is, times when M outputs a hypothesis different from its previous one.

Note that $1 + \mathrm{MC}(M, x)$ is an upper bound for $\mathrm{DH}(M, x)$. Also, if M identifies in the limit the rule presented by x, then both $\mathrm{MC}(M, x)$ and $\mathrm{DH}(M, x)$ are finite. The converse is not always true, since M may converge to an incorrect hypothesis. (A "reliable" method, defined in Section 4.4, may not converge to an incorrect hypothesis.)

Consider the problem of identifying a recursively enumerable class of recursive functions. Identification by enumeration is applicable; however, to identify the nth function in the enumeration, n distinct hypotheses and $n - 1$ mind changes may be necessary. Barzdin and Freivald [1972] have shown that there is an identification method that requires only $\log n + o(\log n)$ distinct hypotheses, and that $\log n - O(1)$ is a lower bound for this problem. Their method constructs a hypothesis that does not necessarily belong to the given enumeration. If a hypothesis from the enumeration is desired, then n is a lower bound on the number of distinct hypotheses, although there is a randomized strategy that achieves an expected value of $O(\log n)$ distinct hypotheses. Podnieks [1975a] obtains an asymptotically exact estimate of the number of mind changes for a probabilistic strategy inferring effectively enumerable classes of total recursive functions. Jantke [1979] proves analogous general results about the number of mind changes for selective and constructive inference operators. Barzdin [1972, 1974a] has also considered the question of the number of distinct hypotheses for identifying finite automata and programs in the presence of additional information about a program, such as the number of instructions or a complete or partial trace of its execution.

The analogous measure for the prediction problem is the number of errors of prediction for a given sequence. This quantity is finite if and only if the prediction method succeeds in the limit. Barzdin and Freivald [1972] obtain upper and lower bounds of $\log n + o(\log n)$ and $\log n - O(1)$ for the problem of predicting a recursively enumerable set of recursive functions.

4. COMPARING INFERENCE CRITERIA

We now turn our attention to studies that compare various inference criteria with one another. Comparisons of inference problems with other computational problems and classes may be found in Adleman and Blum [1975], Brandt [1981], Jung [1977], Kugel [1977], Thiele [1973, 1975], and Wiehagen [1978].

4.1 Identification in the Limit

Consider first some very general definitions of identification in the limit for functions and languages. Let $\phi_1, \phi_2, \phi_3, \ldots$ be any acceptable programming system [Machtey and Young 1978]. (For concreteness the reader may think of this simply as an enumeration of all Turing machines, LISP programs, or whatever.) Then each ϕ_i denotes a partial recursive function, and each partial recursive function is denoted by at least one ϕ_i. For each i define W_i to be the domain of definition of the function ϕ_i; $W_1, W_2, W_3, \ldots$ is then an enumeration of all the recursively enumerable sets.

Let M be an identification method. If

$$x = x_1, x_2, x_3, \ldots$$

is an infinite sequence of inputs, denote by $M[x]$ the empty, finite, or infinite sequence of outputs produced by M with input sequence x. M is defined to *converge* to i on input x if and only if either $M[x]$ is finite and its last element is i or $M[x]$ is infinite and equal to i except at finitely many points. The convergence of $M[x]$ to i is denoted by $M[x]\downarrow i$.

Assume that a specific definition D of admissible presentations of functions and languages has been chosen. If f is a partial recursive function, let $D(f)$ denote the set of admissible presentations of the function f according to the definition D. Similarly, if L is a recursively enumerable set, let $D(L)$ denote the set of admissible presentations of L.

If f is a partial recursive function, M *identifies f in the limit* if and only if for every x in $D(f)$, there exists an i such that $M[x]\downarrow i$ and f is a subfunction of ϕ_i. (Note that if f is a total function, then the second condition reduces to $f = \phi_i$.) If U is a set of functions, then M *identifies U in the limit* if and only if M identifies in the limit every f in U.

If L is a recursively enumerable language, *M identifies L in the limit* if and only if for every presentation x from $D(L)$, there exists an i such that $M[x]\downarrow i$ and $L = W_i$. If U is a set of recursively enumerable languages, then M is said to *identify U in the limit* if and only if M identifies in the limit every L in U.

Note that in these definitions the hypothesis spaces are universal for the partial recursive functions and the recursively enumerable sets. Various restrictions and modifications of these definitions are described in subsequent sections.

Let EX denote the class of all sets U of total recursive functions such that some identification method identifies U in the limit. (EX abbreviates "explanatory.") A fundamental result, due to Gold [1967], is that the set R of all total recursive functions cannot be identified in the limit by any identification method. This result can be proved by taking any sufficiently powerful identification method and defining from it a total recursive function that it does not identify in the limit. (The intuition behind the proof is similar to the faintly suspect rule "one more than the guess" described in Section 1.1.) In some sense, this means that there is no "most powerful" method of identifying the total recursive functions. One important focus of research has been the attempt to characterize just which sets of functions and languages *are* identifiable in the limit.

Gold's result was claimed earlier by Putnam in his 1963 Voice of America lecture [Putnam 1975]. However, Putnam's proof implicitly assumes that the inference machines are restricted to output only programs for total functions. Machines restricted in this fashion are discussed in Section 4.4.

4.2 Other Inference Criteria

4.2.1 Prediction versus Identification

The sequence prediction problem is to predict the next value in the range of a total recursive function of the natural numbers, given some initial segment of the range, or, in other words, to predict the nth element of the sequence given the first $n - 1$ terms.

A slightly different type of inference method, called a *prediction* method, is used for these problems. A prediction method is simply an algorithmic device that accepts as input a finite (possibly empty) sequence of values and may output some value and halt, or may diverge. If M is defined on the sequence $x_1, x_2, \ldots, x_n$, let $M(x_1, x_2, \ldots, x_n)$ denote the value output. Three definitions of successful prediction have been considered, differing in the restrictions they impose on the divergence of the prediction method, M.

The first, and most restricted, is NV-prediction, considered by Barzdin [1972] and Blum and Blum [1975]. (NV abbreviates "next value.") *M NV-predicts* a total recursive function f if and only if $M(x_1, x_2, \ldots, x_n)$ is defined for all finite sequences of values $x_1, x_2, \ldots, x_n$ and

$$M(f(0), f(1), \ldots, f(n-1)) = f(n)$$

for all but finitely many n. (This is the type of convergence that we implicitly used in Section 1.1.) If U is a set of total recursive functions, then *M NV-predicts U* if and only if M NV-predicts every f in U. Let NV denote the class of all sets of total recursive functions that are NV-predicted by some prediction method.

M NV′-predicts the function f if and only if $M(f(0), f(1), \ldots, f(n-1))$ is defined for all n, and

$$M(f(0), f(1), \ldots, f(n-1)) = f(n)$$

for all but finitely many n. The notion of NV′-prediction is due to Barzdin and Freivald [1972]. NV′-prediction of a set of functions and the class NV′ are defined analogously.

The third and least restricted definition is due to Podnieks [1974]. *M NV″-predicts* f if and only if for all but finitely many n, $M(f(0), f(1), \ldots, f(n-1))$ is defined and equal to $f(n)$. NV″-prediction of a set of functions and the class NV″ are defined analogously.

Note that NV requires that M converge on all finite sequences, and NV′ requires that M converge on all initial segments of correctly predicted functions. However, NV″ allows M to diverge on finitely many initial segments of correctly predicted functions. These apparent differences in strength of restriction are real: NV is a proper subclass of NV′, which in turn is a proper subclass of NV″ [Barzdin and Freivald 1972; Case and Smith 1983; Podnieks 1974]. Also, NV and NV′ are proper subclasses of EX, whereas NV″ is equal to BC, a proper superclass of EX defined in Section 4.2.2 below [Case and Smith 1983; Podnieks 1975b]. Further comparisons with other identification types are given in subsequent sections.

Naive intuition suggests that prediction is possible whenever identification is, since the hypothesized rule may be used to predict the next term. However, this is false unless all guesses are restricted to be total recursive functions. In a limited sense, the reverse is true.

To see that a method of NV′-prediction may be used to give a method of identification, suppose that A is an algorithm that takes an initial segment of a sequence and predicts the next term. Given an initial segment of a sequence,

$$x_1, x_2, \ldots, x_n,$$

define a rule f as follows:

$$\begin{aligned} f(i) &= x_i \quad \text{for} \quad i = 1, 2, \ldots, n, \\ f(i+1) &= A(f(1), f(2), \ldots, f(i)) \\ &\qquad \text{for} \quad i \geq n. \end{aligned}$$

Clearly, a strategy that uses f as its new guess every time the current conjecture turns out to be false will correctly identify in the limit any sequence that A correctly predicts in the limit.

4.2.2 Behaviorally Correct Identification

Behaviorally correct (*BC*) *identification* requires that the semantics (rather than the syntax) of the guesses converge correctly in the limit. An inference process is allowed to continue changing its guess indefinitely, as long as after some finite initial segment the guesses are all correct descriptions of the underlying rule.

If M is an identification method and f is a total recursive function, then *M BC-identifies f* if and only if for every admissible presentation x of f, $M[x]$ is an infinite sequence: $y_1, y_2, y_3, \ldots$ such that for all but finitely many n,

$$\phi_{y_n} = f.$$

M BC-identifies a set U of total recursive functions if and only if M BC-identifies every f in U. BC denotes the class of all sets U such that for some inference method M, M BC-identifies U.

BC-identification is a less restrictive criterion than identification in the limit; in fact, EX is a proper subclass of BC [Barzdin 1974b; Case and Smith 1983]. The class BC is equal to the prediction class NV″ [Case and Smith 1983; Podnieks 1974]. Feldman [1972] introduced the concept of behaviorally correct identification of languages, which he termed "matching in the limit." He also defined notions of approaching and strong approaching, and compared these with identification in the limit.

4.2.3 Approximate Identification and Anomalies

We can weaken the definition of identification in the limit, and thus allow more classes of rules to be effectively identifiable, by permitting the final guess to be "nearly correct" as a representation of the rule being presented. It is probably rare that a scientific theory or large computer program is completely "bug free," that is, gives the correct intended answers on all possible inputs [Laudan 1977]. Thus it seems realistic to consider hypotheses that are approximately correct in some sense.

Wharton [1974] proposes a general notion of approximation of one language by another one. In particular, each string s over the alphabet is assigned a fixed positive weight $w(s)$ such that the sum of the weights of all the strings is 1. Then the measure of the distance $d(L_1, L_2)$ between two languaged L_1 and L_2 is the sum of the weights of the strings in the symmetric difference of L_1 and L_2. This defines a metric on the space of languages. For any positive number ϵ, an inference method ϵ-identifies a language L if and only if it converges to a grammar G such that

$$d(L(G), L) < \epsilon.$$

One result of this theory is that for any ϵ, the class of all languages over a given alphabet is ϵ-identifiable in the limit from positive presentations by a method that outputs only grammars for finite languages. Consequently, ϵ-identification allows more classes of languages to be inferred from weaker types of data presentation than the standard notion of identification in the limit, but perhaps the criterion of approximation is too liberal.

As an alternative, we might permit the final guess to have some finite number of "bugs" or "anomalies," that is, places where the guess and the target rule disagree. These anomalies are not weighted, as in Wharton's scheme, but an upper bound may be placed on their number.

If k is a natural number and f and g are partial recursive functions, define $f =^k g$ if and only if $f(n) = g(n)$ for all but at most k values of n. Thus $f =^0 g$ is equivalent to $f = g$. If f is a total recursive function and M is an identification method, then *M identifies f in the limit with at most k anomalies* if and only if for every admissible presentation x of f, $M[x]$ converges to a value i such that $\phi_i =^k f$. Let EX^k denote the class of all sets U of recursive functions such that some identification method M identifies all f in U in the limit with at most k anomalies. Thus EX^0 is just EX. Define $f =^* g$ to mean that $f(n) = g(n)$ for all but finitely many values of n. The class EX^* is defined analogously. EX^*-identification is called *a.e. identification* by Blum and Blum [1975] and *subidentification* by Minicozzi [1976].

Case and Smith [1978, 1983] show that $\mathrm{EX}^0, \mathrm{EX}^1, \mathrm{EX}^2, \ldots$ is a proper hierarchy of classes, and that their union is properly contained in EX*. Barzdin and Podnieks [1973] study a relaxation of the convergence criteria that allows identification by a method M that ultimately outputs only hypotheses from some finite set of correct hypotheses. They show that such a relaxation does not alter in any way the set of identifiable functions. Case and Smith [1978, 1983] extend this result to cases of identification with anomalies. Anomalies may also be permitted with respect to the behaviorally correct identification criterion, where the classes BC^k, for $k = 0, 1, 2, \ldots$ and $k = *$ are analogously defined. Case and Smith also show that this is a proper hierarchy. Harrington has constructed a "most powerful" inference method that BC*-identifies all the total recursive functions (see Case and Smith [1983]). However, it is necessarily the case that for Harrington's machine each successive correct hypothesis has more bugs than the previous one [Chen 1982].

4.2.4 Identification by Teams of Machines

The class EX is not closed under union [Barzdin 1974b; Blum and Blum 1975]; that is, there are two classes U and V of total recursive functions such that each class is identifiable in the limit, but their union is not. Blum and Blum [1975] give the following example. Let U contain all 0, 1-valued total recursive functions that are 0 on all but finitely many points. Let V contain all 0, 1-valued total recursive functions f such that the least value of n for which $f(n) = 1$ has the property that $\phi_n = f$. This slightly paradoxical set is called the self-describing functions for obvious reasons. Clearly, each of U and V is separately identifiable, but it may be shown that their union is not. ("Reliability," discussed in Section 4.4, is a restriction that guarantees closure under union and other operations.)

The nonunion property of identification leads to the definition of identification by "teams" of inductive inference machines. We consider some fixed finite collection (a team) of inductive inference machines working in parallel on some sequence of examples, and define the rule as correctly identified if one or more of the machines succeeds in identifying the rule according to some particular criterion of inference. Then there is a hierarchy with respect to the number of machines in the team for the basic criterion of identification in the limit, and others. In particular, for each n, n inductive inference machines can EX-identify sets of functions that cannot be BC-identified by any smaller team of inference machines [C. Smith 1982].

Studies of the trade-offs between the number of inductive inference machines in a team and the accuracy of the hypothesis (as measured by the number of anomalies) indicate that this relationship is more fundamental than the distinction between EX and BC identification. C. Smith [1982] has shown that m EX^a machines can be simulated by n EX^b machines if and only if

$$n \le m \times \left(1 + \left\lfloor \frac{b}{(a+1)} \right\rfloor\right).$$

Daley [1983] has shown that the same formula characterizes the trade-off between accuracy of the hypothesis and the number of machines for BC-identification.

4.3 The Effects of Data Presentation

Gold [1967] considers the effects of different types of data presentation on the inference of languages. He shows that no set of languages containing all the finite languages and at least one infinite language can be identified in the limit from positive presentations. This result applies to many important classes of languages (e.g., the regular languages and the context-free languages), which presents something of a problem to linguists attempting to account formally for the fact that children learn their native languages from what appears to be positive rather than positive and negative data. Researchers have made various suggestions for getting around this difficulty. One is to add a semantic component to the model of the input; another is to consider stochastic presentations of the data. For example, Horning [1969] shows that the stochastic context-free languages can be identified in the limit with probability 1 from stochastic presentations (which contain no explicit negative data). Both semantics and stochastic presentation are employed by Wexler and Culicover [1980] to obtain their results.

However, it is not the case that no interesting classes of languages can be identified from positive presentations, and it could be argued that classes in which *any* finite language could be the target language are inappropriate for inference. Angluin gives a characterization of those sets of recursive languages that can be identified in the limit from positive data [Angluin 1980a], and two concrete examples of such sets [Angluin 1980b, 1982b], described further in Section 7.5. Osherson and Weinstein [1982] consider inference from positive presentations with respect to EX, EX*, BC, BC*, and other criteria. Case and Lynes [1982] extend their results to include all the EX- and BC-type inference criteria for positive presentations.

Gold also considers restricting positive presentations to those that may be enumerated by primitive recursive programs and shows that there exists an identification method that will identify in the limit every recursively enumerable set from such presentations. Wiehagen [1977] describes further work on characterizing the effects of restrictions of this type.

To infer total recursive functions, we may assume without loss of generality that the data presentation is in the order

$$f(0), f(1), f(2), \ldots .$$

The reason is that an identification method may simply read and store inputs until it has some complete initial segment, simulate a method that expects values in the above order, and then repeat this cycle. However, for some types of restricted methods, this difference in presentations makes a difference in identifiability, as shown by Jantke and Beick [1981].

Blum and Blum [1975] consider the effects of restrictions of presentations on the identification of partial recursive functions. They show that identification types using arbitrary and effective presentations are equivalent, and identification methods may be made order independent without loss of generality. An order-independence result for positive presentations is described by Osherson and Weinstein [1982].

4.4 Restrictions on Inference Methods

We shall now describe some of the restrictions on inference methods that have been studied. In general, the effect of a restriction is to make fewer sets of rules identifiable, since some otherwise acceptable inference methods may not satisfy the restriction.

One restriction is *consistency*, which requires that an identification method output as guesses only hypotheses that are compatible (or consistent) with the examples read in up to the time that the guess is made. Clearly, identification by enumeration is a consistent method. To be a little more precise, there is a distinction between requiring that the guesses of a method be consistent on every finite sequence of examples or only on those finite sequences of examples that are initial segments of presentations of successfully identified rules.

If CONS denotes the class of sets of recursive functions that are identifiable by a consistent inference method (in the second sense), then CONS is a proper subset of EX [Wiehagen 1976]; that is, there exist sets of recursive functions that are identifiable in the limit, but not by any consistent method. One such set is that of the self-describing functions, defined in Section 4.2.4. Consistency in the first sense yields weaker inference methods than does consistency in the second sense [Wiehagen and Liepe 1976]. Additional studies of consistency may be found in Freivald and Wiehagen [1979], Jantke and Beick [1981], and Kugel [1977].

One of the salient properties of identification in the limit is that the inference method generally has no effective way of testing for convergence, so processing continues indefinitely. Finite identification removes this uncertainty. One definition (essentially that of Freivald and Wiehagen [1979]) is that a finite or infinite sequence $y_1, y_2, y_3, \ldots$ *finitely converges* to a value t at element n if and only if $y_n = t$ and n is the least natural number such that

$y_n = y_{n+1}$. So as soon as a value is repeated in two successive positions, the sequence is defined to have converged.

Let f be a function and M an identification method. If for every admissible presentation x of f there exists an i such that $M[x]$ finitely converges to i and $\phi_i = f$, then *M finitely identifies f*. The definition for languages is analogous.

Finite identification is harder to achieve than identification in the limit, and so fewer classes of functions are finitely identifiable. In particular, if FIN denotes the class of sets of total recursive functions that are finitely identified by some identification method, then FIN is a proper subclass of EX [Lindner 1972]. Freivald [1979] defines finite identification by a probabilistic inference strategy of a class of functions with a certain probability; he shows that identification with probability greater than $\frac{2}{3}$ is the same as finite identification, and that there is a hierarchy of classes based on the probabilities $\frac{2}{3}, \frac{3}{5}, \frac{4}{7}, \frac{5}{9}, \ldots$. Flanagan [1981] describes an interesting variant of a finite identification problem for finite functions, in connection with inferring the semantics of a language with known syntax. The problem is to identify an unknown function f defined on and taking values in the set $\{1, 2, \ldots, n\}$, given a collection of pairs of sets $\langle X, Y\rangle$ such that $f(X) = Y$.

Another restriction is *reliability*, which requires that if an identification method converges on an input sequence, it must converge to a correct hypothesis. If a reliable method fails to identify some rule, it must do so by changing its hypothesis infinitely often. For a reliable method, successful identification occurs exactly when the number of mind changes, MC(M, x), is finite.

To be precise, the class of inputs over which a method must be reliable must be specified. An identification method M is *reliable* on a set of functions U if and only if for every f in U and every presentation x of f, if $M[x]$ converges, then it converges to an index for a superfunction of f [Blum and Blum 1975].

Minicozzi [1976] exhibits a variety of effective closure properties for the class of sets of total recursive functions identifiable by an identification method that is reliable on all the total recursive functions. (She uses the term "strong" instead of "reliable.") For example, this class is closed under infinite union, whereas the standard type EX is not even closed under finite union. Wiehagen and Liepe [1976] characterize when the union of two sets in EX is also in EX. Blum and Blum [1975] give complexity-theoretic characterizations of the sets of partial recursive functions identified by methods reliable on all the partial recursive functions, and just on the total recursive functions. (These results are described further in Section 6.)

A restriction termed *Popperian* by Case and Ngo-Manguelle [1979] is that all the hypotheses output as guesses by an identification method must be total recursive functions. (This restriction reflects Popper's methodological requirement that hypotheses in science be refutable.) The classes of total recursive functions identifiable by Popperian machines are precisely the prediction type NV [Case and Smith 1983]. Intuitively, this equivalence exists because the total recursive guesses of a Popperian identification machine may be used to predict without fear that it will not converge, and a predictive method, may have its answers fed back into it to define a total recursive function to use as a guess. (See the description in Section 4.2.1.)

An identification method is *conservative* if and only if it outputs a hypothesis different from its previous guess only when the previous guess is incompatible with the examples read in so far. A conservative method will stick to a hypothesis as long as it is consistent with the data read in so far. Clearly, identification by enumeration may be implemented as a conservative method. The effects of restricting identification methods to be conservative are considered by Angluin [1980a] and Kugel [1977].

Given a class U of total recursive functions and an identification method M, Jantke and Beick [1981] define M to be *class preserving* on U if and only if for every presentation x of a function from U, every element of the sequence $M[x]$ is from U. Intuitively, M is forced to make its guesses from the class U. Identification by enumeration is a class-preserving method on the class of functions in the given enumeration. For an arbitrary inference method, the class U is usually taken to be the class of functions correctly identified by the method. Jantke and Beick study the effect of this restriction, as well as a related restriction called "weakly class preserving." The class-preserving property is related to what Jantke calls *selective* in his study of inference operators [Jantke 1979], which is discussed further in Section 6. Jantke and Beick [1981] also study the effects of various combinations of these restrictions and others.

Another restriction requires that an identification method produce the "smallest" possible description as its ultimate guess of a rule. For a hypothesis space consisting of the programs from some acceptable programming system, "smallest" is defined with respect to an abstract measure of program size, defined by Blum [1967a]. Freivald [1975a] considers the identification in the limit of absolutely minimal programs. He shows that whether or not minimal programs for some sets of functions can be identified in the limit depends on the choice of an acceptable programming system used as the hypothesis space. In fact, the only classes of functions that can be inferred in all such hypothesis spaces are finite [Freivald 1974]. In contrast, Chen [1982] shows that the inference of nearly minimal-size programs (programs minimal with respect to a recursive factor) is independent of the hypothesis space. He also considers variations of identification in the limit, including identification with anomalies and identification by Popperian inference machines, and shows that identification of nearly minimal-size programs is independent of the hypothesis space for these variations. However, there is a loss of inferring power if a nearly minimal program is required, unless a finite unbounded number of anomalies is tolerated [Chen 1982].

In their study of the inference of minimal-size programs, Freivald et al. [1982] consider inference performed with respect to a particular class of restricted hypothesis spaced called Friedberg numberings. A *Friedberg numbering* is a list of all and only the partial recursive functions ϕ_1, ϕ_2, $\phi_3, \ldots$ such that each function appears only once in the list. Friedberg [1958] has shown that such enumerations exist. Freivald et al. show that for every class of recursive functions U in EX, there is a Friedberg numbering that can be used as a hypothesis space to identify in the limit all the functions in U. They also prove that if U is in FIN, then U is finitely identifiable within the constructed Friedberg numbering.

5. COMPARING HYPOTHESES

There are at least two obvious criteria to consider in evaluating hypotheses: the simplicity of the hypothesis and its relationship to the sample data. These may be given formal significance in several ways, some of which are considered below.

Each of the measures we consider can be represented abstractly as a partial ordering of the hypotheses compatible with a given sample. The ordering is permitted to depend on the sample under consideration. The relation *g is less than h with respect to* S is the definition of what it means for g to be a "better" explanation of the sample S than h. When two guesses are incomparable in the ordering, the interpretation is that neither is a better explanation of the sample than the other. The set of *best guesses* for a given sample are the minimal elements in this ordering. We call an ordering of this type a *goodness ordering*. We now consider some general properties of such orderings, and describe particular orderings that have been studied.

A goodness ordering is *computable* if there is an algorithm that takes any finite sample and returns an element of the set of best guesses for the sample. It is *sample independent* if and only if whenever g is less than h with respect to some sample S, then g is less than h with respect to every sample S with which they are both compatible. Orderings based only on some property of the hypotheses (e.g., size) are sample independent.

Consider any computable sample-independent goodness ordering, and let A denote an algorithm for finding an element of the set of best guesses for a given sample. The algorithm A may be used in the following scheme P to produce an identification method:

(1) A is used to generate a best guess for the initial sample. Then the method continues to read data and check the current guess for compatibility with it.
(2) As long as the current guess remains compatible, it is not changed, but if it becomes incompatible with the data, A is used to generate a new best guess for the current sample set.

The method obtained from P is guaranteed to output a best guess for the given sample at every finite stage, and it is *conservative* in the sense that it does not change a previous hypothesis unless it is incompatible with the data. Whether this process correctly identifies in the limit all the rules in the domain depends on properties of the specific domain and ordering.

The rest of Section 5 describes specific orderings based on simplicity of hypotheses, goodness of fit to the data, and mixtures of these.

5.1 Simplicity of Hypotheses

If there is some measure of "simplicity" on the set of all possible hypotheses, we may consider the ordering in which g is less than h if g is simpler than h. This is a sample independent goodness ordering. The set of best guesses for a sample is the set of simplest hypotheses that are compatible with the sample. To illustrate, define the complexity of a polynomial to be its degree. Then the classical interpolation algorithm produces the least complex polynomial compatible with each initial segment of a given sequence.

Consider also the problem of inferring finite-state acceptors from complete presentations, and define the complexity of a finite-state acceptor to be the number of states that it contains. The problem of computing the best guesses for a given sample is as follows. Given two disjoint finite sets of strings, S_0 and S_1, find a finite-state acceptor with the fewest possible states that accepts all the strings in S_1 and none of the strings in S_0. It is easy to see that this is computable: just enumerate acceptors in order of increasing numbers of states and take the first compatible one. Moreover, if this algorithm is used in the inference scheme P, the resulting method correctly identifies in the limit all the regular sets.

However, Gold [1978] has shown that the computational problem of finding a minimum finite-state acceptor compatible with given data is NP-hard. Consequently, all known algorithms for this problem require exponential time on some inputs. An analogous result holds also for regular expressions. Define the size of a regular expression to be the number of occurrences of alphabet symbols it contains, so that the size of the expression $(00)^* + 0^*11$ is 5. Then the computational problem of finding a regular expression of smallest size compatible with given positive and negative examples is NP-hard [Angluin 1978].

In general, identification by enumeration produces the "simplest" compatible hypothesis if the enumeration order is consistent with the intended simplicity ordering of hypotheses. However, the complexity results suggest that there may be no analog of polynomial interpolation in general; that is, in some settings there may be no efficient algorithm for the computational problem of finding the simplest compatible hypothesis.

5.2 A Set-Theoretic Goodness of Fit Measure

When languages are inferred from positive data, one way to define the goodness of fit of hypotheses to samples is to use the ordering of set inclusion among all the candidate languages that contain the given positive examples. Let H be the hypothesis space and $L(h)$ the language generated by hypothesis h. Consider the ordering in which g is less than or equal to h if and only if $L(g)$ is a subset of $L(h)$. Then given a sample S, the set of best guesses for S are the minimal elements in this ordering that are compatible with S, that is, such that S is a subset of $L(h)$. Roughly speaking, a best guess for S generates the sample S and as few additional strings as possible. This ordering is sample independent, and so the inference scheme P is applicable.

Despite the difficulties of identification from positive data, there are efficient algorithms for this criterion that lead to correct identification in the limit in several different domains [Angluin 1980b, 1982b; Crespi-Reghizzi 1972; Crespi-Reghizzi et al. 1978; Shinohara 1982a, 1982b]. These results are described in Section 7.

5.3 Mixed Measures: Derivational Complexity

Feldman [1972] gives a reasonable set of general axioms for orderings that combine grammatical complexity with derivational complexity for inferring grammars. Any such ordering is called a *grammatical complexity measure.* He proves that every ordering that satisfies the axioms is computable; that is, there is an algorithm for finding a grammar compatible with given data and minimal in the ordering. (The Bayesian measure of Horning, discussed below, is one instance of a grammatical complexity measure.) Some results are given relating such measures to various criteria of inference in the limit. Analogous results for measures that combine program size and resource complexity for inference of programs are given by Feldman and Shields [1977].

5.4 Mixed Measures: Bayesian Approaches

One approach to combining the measures of simplicity of the hypothesis and goodness of fit to the data, based on Bayes' Theorem, defines certain probability measures and then looks for a hypothesis that maximizes the probability of the hypothesis, given that a particular sample is observed. We define a probability measure $\Pr(h)$ on the hypothesis space, a probability measure $\Pr(S)$ on the sample space, and $\Pr(S \mid h)$, the probability of a given hypothesis h generating a given sample S. Then we define the ordering to be g is less than h with respect to S if and only if $\Pr(g \mid S)$ is greater than $\Pr(h \mid S)$.

For a given sample S, the best guesses are those h that maximize $\Pr(h \mid S)$. Since by Bayes' Theorem,

$$\Pr(h \mid S) = \frac{\Pr(h)\Pr(S \mid h)}{\Pr(S)},$$

it suffices to maximize $\Pr(h)\Pr(S \mid h)$.

If we think of $\Pr(h)$ as a measure of the simplicity of the hypothesis h (higher probability assigned to simpler hypotheses) and $\Pr(S \mid h)$ as a measure of the goodness of fit of h to the sample S (higher probability assigned to a better fit), then this measure does trade off simplicity against goodness of fit. Note that this ordering is usually not sample independent, and so the scheme P described above is not applicable.

To use this approach, we must decide how to assign probabilities to the elements of the hypothesis space. This represents an opportunity to incorporate prior knowledge of the domain into the inference strategy. The approach is practically useful only insofar as the probability measures are computable or approximately computable in the appropriate sense. Watanabe [1960] gives arguments for the use of Bayesian methods in inductive inference and investigates the rate at which evidence diminishes the effect of the prior probabilities in the limit.

An early and important paper by Solomonoff [1964] advocates use of the Bayesian approach with a priori probabilities based on the theory of program size complexity. Work on the general theory of program size complexity and its relation to randomness and information content of specific objects has been done by Kolmogorov, Chaitin, and Martin-Lof, among others [Chaitin 1974, 1975, 1976; Daley 1973, 1977; Gacs 1983; Kolmogorov 1965; Martin-Lof 1966, 1971; Zvonkin and Levin 1970]. More recent work by Solomonoff on this topic may be found in Solomonoff [1975, 1978].

Horning [1969] has used this approach in the context of inferring stochastic context-free grammars. The probability of a sample given a grammar is defined using the production probabilities of the grammar, and the probability measure on grammars is defined by a stochastic grammar generator. Horning describes an enumerative algorithm for finding h to maximize $\Pr(h)\Pr(S|h)$, given a sample S of strings as input. He proves that if the input is stochastically generated, this algorithm leads to correct identification in the limit with probability 1. Note that even though the context-free languages are not identifiable in the limit from positive presentations, they are identifiable in the limit with probability 1 in this stochastic setting, which includes no explicit negative examples. Van der Mude and Walker [1978] have also used this approach for the inference of stochastic regular grammars.

Cook et al. [1976] define a measure that is rather similar to Horning's for the problem of inferring stochastic context free grammars from stochastically generated positive samples. They define a measure of grammatical complexity, $C(G)$, and a measure of the discrepancy between the language of a grammar and given sample, $D(L(G), S)$. Their combined measure is a positive linear combination of these two measures. One interesting property of their measure of grammatical complexity is that it assigns a lower complexity to the grammar

$$\{X \rightarrow aaaaa, X \rightarrow bbbbb\}$$

than to the grammar

$$\{X \rightarrow aabab, X \rightarrow babba\},$$

thus taking into account somewhat the randomness of the strings composing the grammar. This may be viewed as an approximation to a priori probabilities defined using program size complexity. They describe an algorithm that takes a sample and finds a grammar whose measure is a local minimum with respect to certain transformation steps. (Their algorithm is described in Section 7.3.)

5.5 Other Mixed Measures

Maryanski and Booth [1977] consider the problem of inferring stochastic finite automata from stochastically generated data. The measure of goodness of fit of a given stochastic finite automaton to a given set of strings is defined using a chi-square test of the expected frequencies of the sample strings against their actual frequencies in the sample. The measure of simplicity of an automaton is the number of states that it contains. The user specifies a threshold for the chi-square test, and then the desired answer is an automaton with the minimum possible number of states that meets the threshold value for goodness of fit. Maryanski and Booth describe an enumerative algorithm for solving this problem and study its implementation. (See also Gaines [1978] for certain corrections.)

Gaines [1976] also considers the problem of mixed measures in the context of inferring stochastic finite automata from stochastically generated data. His suggestion is not to trade off simplicity and goodness of fit, but to consider them jointly and independently. The measure of simplicity is the number of states in an automaton, and the measure of goodness of fit is defined using the transition probabilities of the automaton. The ordering that he considers is simply the product of these two total orders; that is, an automaton is a best guess for a given sample if and only if no other automaton has both fewer states and a better fit to the sample. An algorithm is described that enumerates the best guesses for a given sample. Examples are given to illustrate the properties of this ordering. Gaines also discusses the use of stochastic methods in deterministic cases to filter out errors in the data.

6. GENERAL INFERENCE METHODS

Identification by enumeration, sketched in the introductory section of this paper, is a consistent, class preserving, conservative, and optimally data efficient method. We may think of it as a general scheme parameterized by an enumeration of a class of recursive languages or total recursive functions to yield an identification method for the enumerated class, or, equivalently, as a general search method that can be adapted to different domains by giving it different generators for the spaces to be searched. We shall now consider general inference methods related to identification by enumeration. Particular inference methods, some based on enumeration and others not, are described in the next section.

6.1 Identification of Resource-Bounded Classes

Blum and Blum [1975] describe three different types of inference methods that use resource bounds as part of the criterion for deciding when to "give up" on the current hypothesis. For concreteness, imagine that $p_1, p_2, p_3, \ldots$ is an enumeration of programs in some acceptable programming system. As a resource, consider the number of steps executed by a program, although any complexity measure would do [Blum 1967b].

Let h be a total recursive function of one argument. A function f is called *h-easy* if and only if there exists a program p_i that computes f such that for all but finitely many inputs x, p_i on input x runs for at most $h(x)$ steps. Given h, the following method correctly predicts all the h-easy functions in the limit. On input $y_1, y_2, \ldots, y_n$, search in some fixed order through all pairs $\langle i, k\rangle$ until the first one is found such that

$$p_i(x) = y_x,$$

for all $x \leq n$, and p_i runs for at most $\max\{k, h(x)\}$ steps on input x, for $x \leq n + 1$. When the first such pair $\langle i, k\rangle$ is found, then output the guess $p_i(n + 1)$. This method gives up on a hypothesis when it runs too slowly (as measured by h), although it may come back to the hypothesis later when k is sufficiently increased. Adleman (see Blum and Blum [1975]) and Barzdin and Freivald [1972] have shown independently that the sets of functions in the class NV are all the subsets of the h-easy sets of functions, as h ranges over all recursive functions.

Blum and Blum also consider two identification methods that are, respectively, reliable on all the partial recursive functions (P) and reliable on the total recursive functions (R). The first of these uses an *a priori* bound $h(x, y)$ that is a total recursive function of two arguments. A function f is said to be *h-honest* if and only if there exists a program p_i to compute f such that for all but finitely many inputs x, if $y = f(x)$, then p_i runs for at most $h(x, y)$ steps on input x. They show that the sets of partial recursive functions identifiable by methods that are reliable on P are all subsets of h-honest sets of functions, as h ranges over all total recursive functions of two variables. The method constructed using a given h is similar to the prediction method described above, except that for the input/output pair $\langle x, y\rangle$, the bound $\max\{k, h(x, y)\}$ is used as the upper bound on the running time of a viable hypothesis when checking consistency with the data.

The last type of inference method that Blum and Blum consider is parameterized by a recursive operator instead of a recursive function and is called an *a posteriori* method. We shall not give formal definitions for this case, but intuitively the method gives up on the current hypothesis only when it has found another hypothesis that correctly computes the exemplified function on "many more inputs" and "much faster" than the current hypothesis. The quantification of "more inputs" and "faster" is given by the particular recursive operator used. Blum and Blum characterize the sets of partial recursive functions identified by methods that are reliable on R as all the subsets of sets identified by this general scheme for all possible recursive operators.

6.2 Methods Uniformly Adaptable to Different Domains

We can also generalize identification by enumeration by studying properties of what might be termed effectively adaptable inference methods. Consider, for example, a general inference scheme that is given a program p to enumerate a class U of functions, which then "adapts itself" to be an identification method for the class U. Identification by enumeration is one such scheme. Jantke [1979] studies general properties of such schemes, which he terms *strategic operators*. He defines a *selective* strategic operator as one that uses as hypotheses only elements of the class U enumerated by the given program p. A *constructive* strategic operator constructs hypotheses by a recursive selection of elements of U depending on the input. Jantke shows that constructive strategic operators may be more powerful than selective ones.

Even when there is no difference in power, constructive strategic operators may be exponentially more efficient in the number of mind changes than selective ones for particular enumerations.

6.3 A Complexity Theory for Inference Methods

Daley and Smith [1983], building on the preliminary work of Freivald [1975b], have developed an axiomatic treatment of the complexity of inference in the spirit of Blum's axiomatic treatment of the complexity of computation [Blum 1967b]. An *inference complexity measure* for a collection of inference procedures is a sequence of functionals, one for each procedure, satisfying the following conditions:

- The complexity of an inference is defined if and only if the inference procedure converges.
- The complexity functional is defined precisely on the input sequences on which the inference procedure is defined.
- There is a limiting recursive procedure for determining whether a given inference procedure has a particular complexity on a certain input.
- The complexity functional requires precisely the same input to converge as the inference procedure.
- The number of mind changes is an absolute lower bound on the complexity of any inference.

Using a rigorous version of the above axioms, Daley and Smith [1983] show the existence of arbitrarily difficult to infer sets of functions and sets of inferable functions for which there is no most efficient inference procedure. A trade-off between complexity and accuracy (as in the number of anomalies) is also exhibited.

7. PRACTICAL INFERENCE METHODS

The general inference methods just described are not directly feasible in practical domains. Our goal now is to describe some of the ideas used in the large variety of specific inference methods that have been proposed and studied.

7.1 Search Spaces

A useful approach is to organize the space of hypotheses into something more complex than the simple nonadaptive linear list used in identification by enumeration. A richer organization of the hypothesis space may allow the elimination of more hypotheses than just the current one when an incompatibility with the examples is detected. Furthermore, the structure itself may suggest a plausible replacement for a failed hypothesis.

To illustrate this approach, consider the problem of finding a deterministic finite-state acceptor of minimum size compatible with a given positive sample S_1 and negative sample S_0. Although this problem is NP-hard [Angluin 1978; Gold 1978], the hypothesis space has a useful structure. In particular, if M is the canonical tree acceptor of the set S_1, then every reduced deterministic acceptor compatible with S_1 and S_0 may be obtained by partitioning the states of M and merging the states within each block of the partition. Consequently, it is sufficient to search the space of partitions of the states of M. Moreover, if M_1 and M_2 are obtained from the two partitions π_1 and π_2 of the states of M, and π_1 is a refinement of π_2, then $L(M_1)$ is contained in $L(M_2)$. Thus, if M_1 incorrectly accepts some element of S_0, then M_2 does also and need not be examined. Also, any partition of the states of M that does not correspond to a deterministic acceptor may be collapsed until it does. This structure and related ones have been used in many contexts [Angluin 1982a, 1982b; Biermann and Feldman 1972b; Feldman et al. 1969; Gold 1972, 1978; Miclet 1980; Pao and Carr 1978; Veelenturf 1978].

Another useful structure is provided by the subsumption relation in predicate logic. If A and B are conjunctions of atoms, A *subsumes* B if and only if there exists a substitution σ such that the atoms of $\sigma(A)$ are a subset of those of B. For example, $U(x)$ & $E(x)$ & $T(y)$ subsumes $U(a)$ & $E(a)$ & $T(f(a))$ & $I(f(a))$. If A subsumes B, then A is *more general* than B.

Consider as the set of hypotheses all finite conjunctions of atoms from some logical language. A hypothesis denotes the set of ground atoms that it subsumes. Inferring such hypotheses from positive or complete presentations of conjunctions of ground atoms is a language inference problem. Subsumption gives a partial ordering of the hypothesis space according to generality/specificity. If a hypothesis A fails to subsume some positive example, then no hypothesis more specific than A need be considered, and, conversely, when A subsumes some negative example, no hypothesis more general than A need be considered. The field generally called *concept learning* includes many elaborations and extensions of this simple setting (see, e.g., Dietterich and Michalski [1979], Hayes-Roth and McDermott [1978], Michalski [1980], Vere [1980, 1981], and Winston [1975]).

Mitchell [1982] describes an abstraction of this setting for concept learning, and gives an algorithm called *version spaces* that maintains the boundaries of the subspace of compatible hypotheses. That is, it maintains the set G of most general hypotheses that do not subsume any negative example, as well as the set S of most specific hypotheses that subsume every positive example. As positive and negative examples accumulate, the subspace of compatible hypotheses shrinks accordingly, and if it is reduced to one hypothesis, the unknown concept is said to be unambiguously learned.

The subsumption relation and the computability of the least general generalization have been formally studied by Reynolds [1970] and Plotkin [1970, 1971a, 1971b]. Shapiro [1981a, 1981b] has made use of subsumption to order search spaces of axioms in Horn form. When an axiom is found to be false, its immediate descendants in the subsumption relation are taken to be plausible replacements.

7.2 Pruning: Forbidden Features

A more complex hypothesis space may be used to eliminate a whole group of hypotheses when the current hypothesis is found to be incorrect. One approach uses a diagnosis routine that takes a hypothesis and a collection of examples with which it is incompatible and attempts to find the reason for the incompatibility. The reason is often a forbidden feature, that is, some part of the discredited hypothesis that will cause any hypothesis in which it is included to be incompatible with the data. Subsequent searching is structured to avoid examining hypotheses that incorporate the forbidden feature.

Wharton [1977] describes one version of this idea in his work on speeding up the enumeration of grammars. His method, which is a form of backtracking, enumerates context-free grammars in increasing order of complexity and tests them against a given sample. If there is an incompatibility, a diagnostic routine returns the first production in the grammar that must be changed to avoid the incompatibility detected. (All the productions up through the indicated one constitute a forbidden feature of the grammar.) The enumerator then skips over the intervening grammars until it finds the first one for which the indicated production is different. Unfortunately, this method does not avoid a forbidden set of productions that occurs in a different position or order in a grammar.

Shapiro [1981a] describes a method of searching the hypothesis space of all finite sets of axioms of the form $H \rightarrow P$, where H is a conjunction of atoms and P is a single atom. The data consist of a set of ground atoms marked as *true* and another set marked as *false* in an unknown model. When the current hypothesis is found to be incompatible with the data, the diagnostic procedure queries an oracle (the user) about certain other ground atoms and returns an axiom $H \rightarrow P$ of the current hypothesis that is false in the unknown model. The offending axiom is then discarded and never again appears in any hypothesis. This approach is more accurate in pinpointing forbidden features, and more successful in avoiding them, than a straightforward backtracking approach.

7.3 Hill Climbing

The hill climbing method finds a locally optimal hypothesis by exploring a neighborhood of the current hypothesis and moving if a better hypothesis is found. The exploration is iterated until the current hypothesis is optimal within its neighborhood.

Cook et al. [1976] propose a hill climbing method for inferring stochastic context-free grammars from stochastically generated positive data. They define a particular measure $M(G, S)$ that combines the complexity of the grammar G with its discrepancy from the sample S. They also define several grammar transformations (substitution, disjunction, simplification) designed to reduce the complexity of a grammar without changing the language that it generates by very much. Their procedure begins with a grammar that generates precisely the observed sample (high complexity and low discrepancy) and searches the neighborhood that consists of all the grammars obtained by applying a single transformation to the current grammar. If any of these transformed grammars is better (according to the measure M) than the current one, the procedure moves to the best grammar in the neighborhood and iterates the search. Otherwise, it stops and outputs the locally optimal grammar that it has found. The behavior of their method on several inference tasks suggests that methods of this kind deserve further study.

7.4 Jumping to Conclusions: Plausible Features

Methods based on systematic search and pruning of hypothesis spaces are essentially cautious. They are designed not to skip over any compatible hypothesis, and so it is usually easy to prove that they produce a simplest compatible hypothesis if the simplicity ordering is consistent with the order of search. For the same reason, they tend to be very time consuming, with the search time growing exponentially in the size of the hypothesis to be discovered.

Methods of another type, called "jumping to conclusions," are generally based on a set of criteria that suggest, on the basis of the data, features that are probably or plausibly present in the desired hypothesis. By composing such features into a full or partial hypothesis, such methods greatly reduce the number of hypotheses examined, sometimes to just one. We use the term *constructive* for methods that jump to conclusions. Their efficiency is often obtained at the expense of any simple characterization of what hypotheses will be produced in a given situation. There has been some success in finding efficient constructive inference algorithms with simply characterized results; these are described in the next section. Many other methods should be described as heuristic; we consider these first.

Biermann and Feldman [1972b] describe one heuristic method, called the method of *k-tails*, for inferring nondeterministic finite-state transducers from input/output data. The method requires a user-supplied parameter k, a positive integer that controls when similar states will be identified. States are identified if they have the same behavior on strings of length at most k, that is, their k-tails are the same.

The method starts with a tree automaton M for the given sample and forms a partition π_k of its states. Two states are in the same block of π_k if and only if they are not distinguishable by strings derived from the sample of length less than or equal to k. The method then identifies the states within each block of π_k and ouputs the resulting (not necessarily deterministic) transducer. In the space of all possible partitions of the states of M, indistinguishability by short strings is taken as evidence that two states should be identified in the final result. If k is chosen sufficiently large and there are enough data, this method produces a correct deterministic transducer. The user's control of k allows tuning of the method to some extent.

The method of k-tails has been generalized for use in inferring tree grammars by Brayer and Fu [1977] and Levine [1981, 1982]. Another heuristic method for inferring tree grammars is described by Gonzalez et al. [1976].

Miclet [1980] describes another approach based on state partitioning, but with a more flexible criterion of similarity than identity of k-tails. He defines a distance measure between the behaviors associated with two states. His method looks for a pair of states that are at most some fixed distance apart, identifies the closest such pair, recomputes the distance measures, and iterates until no pair of states is sufficiently close to be identified.

7.5 Efficient and Characterizable Methods

We shall now briefly describe several constructive inference methods that are provably efficient and have simply characterizable results. These methods all correctly identify in the limit a nontrivial class of formal languages from positive data. Each uses a polynomial-time algorithm to produce a smallest (in the sense of set containment) language in the class containing a given positive sample.

Crespi-Reghizzi [1972] describes a method for inferring context-free grammars from bracketed samples (i.e., positive samples from a bracketed grammar). Given a bracketed sample, in effect one has a set of parse trees from the unknown grammar with all internal labels erased, and the problem is to reconstruct the internal labels. Crespi-Reghizzi's approach is to apply a fixed function which can depend on the subtree rooted at a node and/or the environment of the subtree in the parse tree, and to use the result of this function to label the node. A general theory of functions of this type, called "abstract profiles," is developed by Crespi-Reghizzi and Mandrioli [1980a, 1980b]. One type of function yields a method for the free operator precedence grammars [Crespi-Reghizzi 1972], another, the free homogeneous k-profile grammars [Crespi-Reghizzi et al. 1978]. Berger and Pair [1978] have studied this general approach in an abstract setting using bimorphisms. The method may be viewed as a clustering method similar to the heuristics described in the preceding section.

A method described by Angluin [1980b] infers the *pattern languages*. A *pattern* is a concatenation of variables and constants, such as $30xx$ or $14xy1x55y$. The language $L(p)$ generated by a pattern p is the set of strings obtainable by substituting constant strings for the variables of the pattern, and so $L(30xx4)$ contains the strings 30114, 30004, 301231234, but not the strings 30564 or 114. She presents an algorithm that finds a smallest pattern language containing a given positive sample. "Pattern automata" are used to represent concisely the set of all the patterns that could have generated a given string or set of strings. For the special case of patterns that contain occurrences of only one variable, the algorithm is shown to run in polynomial time.

Shinohara [1982a, 1982b] has considered other types of pattern languages, including the *extended regular pattern languages*, in which all the variables occurring in a pattern are distinct and the null string may be substituted for any variable, and the *non-cross-pattern languages*, in which all of the occurrences of each variable are either to the left or to the right of all occurrences of any other variable. He gives polynomial-time algorithms for finding minimal extended regular and non-cross-pattern languages from positive data. These algorithms start with a "most general" pattern and specialize it as much as possible using local optimization. They lead to correct identification in the limit of the extended regular and non-cross-pattern languages. Shinohara's application of the regular pattern languages to a data entry system is described in Section 8.

Another method described by Angluin [1982b] infers the *k-reversible languages*, which form a proper subclass of the regular languages. Let k be a nonnegative integer and L a regular language. Then L is *k-reversible* if and only if whenever u_1vw and u_2vw are in L and $|v| = k$, then for every z, u_1vz is in L if and only if u_2vz is in L. An example of a zero-reversible language is the set of strings of 0's and 1's with an even number of 0's and an even number of 1's. For any fixed k there is a polynomial-time algorithm for finding the smallest k-reversible language containing a given positive sample. The algorithm, which is based on a heuristic originally proposed by Feldman [1967], constructs the canonical tree acceptor for the sample and then successively merges any pair of states that satisfy a particular criterion of similarity (dependent on k). The method is thus another clustering method.

7.6 LISP Program Inference

A number of researchers have studied the inference of LISP programs from their input/output behavior, which constitutes a significant subfield of inductive inference. D. R. Smith [1982] gives a survey of this work. We shall present some of the main ideas.

The systems of Hardy [1975] and Shaw et al. [1975] are heuristic; they are based on matching a schema for a LISP program to the given data and filling in the schema with concrete values in a plausible way. The system of Siklossy and Sykes [1975] has a similar general approach for inferring a LISP-like language. Biermann [1978] applies his general methodology of synthesizing programs from traces to a class that he calls the regular LISP programs. Biermann and Smith [1977] combine this method with a hierarchical decomposition scheme to achieve greater efficiency in the synthesis of "scanning" LISP programs.

Summers [1976, 1977] describes a general approach to LISP program inference based on simple recursive schemata and an algorithmic matching procedure. Each output y is expressed as a term composed from the base functions (*car*, *cdr*, and *cons*) applied to the corresponding input x, and then a recurrence relation is sought among these terms and used to synthesize a recursive program. For example, the input/output pairs

$$\langle (A), A\rangle, \langle (A\ B), B\rangle, \langle (A\ B\ C), C\rangle$$

give rise to the terms

$$t_1 = car(x), \qquad t_2 = car(cdr(x)),$$
$$t_3 = car(cdr(cdr(x))),$$

and the recurrence

$$t_{i+1} = t_i(cdr(x)/x)$$

is detected. A separate, somewhat similar procedure is used to discover the predicates and synthesize the recursive function

$$f(x) = \text{if atom}(cdr(x))$$
$$\text{then } car(x) \text{ else } f(cdr(x)),$$

which extracts the last element of an input list.

Summers also describes a method of introducing auxiliary variables to synthesize more complex functions. Considerable work exploring and expanding his approach has subsequently been done by Guiho, Jouannaud, Kodratoff, and others [Jouannaud and Guiho 1979; Jouannaud and Kodratoff 1979, 1980; Jouannaud et al. 1977; Kodratoff 1979; Kodratoff and Fargues 1978]. One promising feature of this work is that the underlying matching algorithms are quite efficient. Another is that there has been reasonable success in characterizing the inferences made by this and related methods.

8. SKETCH OF SOME APPLICATIONS

Inductive inference may be used to provide abstract models of the process of scientific inquiry or the process by which a child acquires its native language [Gold 1967; Putnam 1975; Wexler and Culicover 1980]. In the area of L-systems, which uses tools from formal language theory to describe biological organisms as they change in time, the problem of identifying an organism, given a sequence of descriptions of it, has been addressed by Doucet [1974] and Feliciangeli and Herman [1977]. There have also been a number of proposals for applying inductive inference in practical systems, which we now sketch.

In structural pattern recognition, a grammar or other formal device describing all the patterns assigned to a particular class is specified. To decide whether a given input pattern belongs to the specified class, it is parsed according to the given grammar. Stochastic grammars are particularly suitable for this application. It is often quite difficult to devise grammars to fit large bodies of data, and it would be useful to automate the task at least partially. For an excellent introduction to the area of structural pattern recognition, including its relationship to inference, see the books by Fu [1975, 1977, 1982] and Gonzalez and Thomason [1978].

In the Meta-Dendral project at Stanford [Buchanan and Feigenbaum 1978], inductive inference of "breakage rules" for chemical molecules from mass spectrograph data has been very successfully automated. This project was undertaken partly to alleviate the bottleneck of acquiring a large number of such rules in an explicit form from chemical experts.

Inductive inference is also potentially applicable in automatic program synthesis. Biermann and Krishnaswamy [1976] describe a synthesis system that uses trace information provided by the user. Shaw et al. [1975] present an interactive system for synthesizing LISP programs. Shapiro [1982a, 1982b] suggests that examples and specifications may be combined to provide a useful, partially automated debugging tool for Prolog programs.

A slightly different emphasis is given by Siklossy and Sykes [1975], who suggest that an induction mechanism may be useful to a robot in generalizing solutions it has found for specific problems. Stolfo and Harrison [Stolfo 1979; Stolfo and Harrison 1979] describe a system that infers heuristics for puzzle solving from examples of solutions.

Crespi-Reghizzi et al. [1973], as well as Knobe and Knobe [1976], suggest that inference may be useful in specifying programming languages. Certainly no expert language designer would trust any existing inference system to construct a reasonable grammar for a realistic language solely from examples, but inference may be of use in some stages of the process of language design. Coulon and Kayser [1979] describe a system that infers a limited database query language from examples.

Another application allows nonexpert users to specify what they want a computer to do at least partly by examples. Systems based on this idea include Zloof's [1977] Query by Example system and the RITA system of Waterman et al. [1980]. Neither of these systems has a true induction component; instead, both use examples to specify variables or parameters for procedures and to provide an informal user interface.

Shinohara [1982b] describes an application of the regular pattern languages (see Section 7.5) to a system for data entry. For example, if the user is entering a series of bibliography entries with common keywords: Author:, Title:, Journal:, and Year:, then the system infers the regular pattern

Author: u Title: v Journal: w Year: x,

where the u, v, w, x represent variables in the pattern, and the rest are constants. The system then prints out the next constant portion of the pattern (e.g., Author:) and waits for the user to type in the next variable portion, and so on, effectively prompting the user for the fields of the record, and removing the burden of typing the keywords.

The Editing by Example system of Nix [1983, 1984] also contains a nontrivial application of inductive inference. In this system, which is implemented in the context of a general-purpose screen editor, the user may specify input/output pairs exemplifying a text transformation, and the system will attempt to infer the pattern common to the pairs and synthesize a program to perform the transformation in general. The user may then execute this program to perform further transformations, or may give further examples to the system. A liberal "undo" facility allows graceful recovery from errors arising from the inference process.

The text transformations synthesizable by the Editing by Example system are *gap programs*, which consist of a *gap pattern* that matches a portion of the text and parses it into fields, and a *gap replacement* that copies, rearranges, or deletes the fields (and may introduce new constant fields). The resulting string replaces the matched string, and the matching and replacement process continues with the remainder of the file. Nix's thesis contains many results related to the inference of gap programs.

To see how the system operates, suppose that the user specifies the following two input/output examples:

```
George Washington, (202) 357-1243
     => George: 357-1243
Abe Lincoln, (202) 356-6636
     => Abe: 356-6636
```

The EBE system synthesizes a gap program that has a gap pattern

```
-1- ' ' -2- ',(202)' -3- eol
```

and a gap rearrangement

```
-1- ':' -3- eol
```

This gap program matches any line that consists of two words separated by a blank, followed by a comma and the area code (202), followed by a telephone number, and replaces each such line with the first of the two words, followed by a colon, followed by the telephone number (minus the area code).

The ideas and systems we have described represent just the beginning of the application of inductive inference. We expect that inductive inference will play a significant role in practical computing in the future.

9. FUTURE DIRECTIONS

As this survey documents, there has been considerable progress in inductive inference research under Gold's paradigm. However, as the reader is surely aware, we have not provided definitive answers to the most basic questions in inductive inference: Given a particular domain, what is the "best" possible inference method in that domain? What are the relevant measures of behavior? What are the trade-offs among them (memory, time, background knowledge, accuracy, errors, data, etc.)?

The most significant open problem in the field is perhaps not any specific technical question, but the gap between abstract and concrete results. It would be unfortunate if the abstract results proliferated fruitlessly, while the concrete results produced little or nothing of significance beyond their very narrow domains.

Paradoxically, part of the problem might be the original Gold paradigm itself. While it captures well certain gross features of the problem of natural language learning—essentially, one "lifelong" problem with data exclusively in the form of examples—it may not be wise to stretch it to try to include more and more of the "microstructure" of inference problems, or domains that consist of several related problems or that present data in a variety of forms. As useful as the Gold paradigm has been, we should not let it blind us to other important questions about the phenomena of inductive inference.

Another difficulty is the paucity of practical applications to guide the formulation of appropriate theoretical questions. Applications do not simply occur; they must be invented and developed, sometimes in advance of the theory that may later explain them.

Notwithstanding these problems, we are confident that the basic questions in inductive inference are sufficiently interesting and important that good work will continue to unravel them, and future generations will have a good laugh at our present ignorance. If this survey has stimulated any part of that future good work, it will have served its purpose.

ACKNOWLEDGMENTS

The initial stages of this project were completed while the second author was on the faculty of Purdue University. The Yale, Purdue, and University of Maryland Computer Science Departments provided computer services. The writing of this paper was greatly facilitated by CSNET. We wish to thank the National Science Foundation for its direct and indirect support of this project. We also thank all the people who read and commented on earlier drafts of this paper; their comments substantially improved its content and readability.

This work was supported by the National Science Foundation under Grants MCS-8301536, MCS-8105214, MCS-8002447, and MCS-7903912.

REFERENCES

ADELMAN, L., AND BLUM, M. 1975. Inductive inference and unsolvability. Tech. Rep., Electrical Engineering and Computer Science Dept., Univ. of Calif., Berkeley.

ANGLUIN, D. 1978. On the complexity of minimum inference of regular sets. *Inf. Control 39,* 337–350.

ANGLUIN, D. 1980a. Inductive inference of formal languages from positive data. *Inf. Control 45,* 117–135.

ANGLUIN, D. 1980b. Finding patterns common to a set of strings. *J. Comput. Syst. Sci. 21,* 46–62.

ANGLUIN, D. 1982a. A note on the number of queries needed to identify regular languages. *Inf. Control 51,* 76–87.

ANGLUIN, D. 1982b. Inference of reversible languages. *J. ACM 29,* 3 (July), 741–765.

BARZDIN, J. M. 1972. Prognostication of automata and functions. In *Information Processing 71,* B. Gilchrist, Ed. Elsevier North-Holland, New York, pp. 81–84.

BARZDIN, J. M. 1974a. On synthesizing programs given by examples. In *Lecture Notes in Computer Science, vol. 5.* Springer-Verlag, New York, pp. 53–63.

BARZDIN, J. M. 1974b. Two theorems on the limiting synthesis of functions. *Latv. Gos. Univ. Uch. Zapiski 210*, 82–88 (in Russian).

BARZDIN, J. M., AND FREIVALD, R. V. 1972. On the prediction of general recursive functions. *Sov. Math. Dokl. 13*, 1224–1228.

BARZDIN, J. M., AND PODNIEKS, K. M. 1973. The theory of inductive inference (in Russian). In *Proceedings of the 1st Conference on the Mathematical Foundations of Computer Science* (High Tetras, Czechoslovakia). Math. Inst. of the Slovak Academy of Sciences, pp. 9–15.

BERGER, J., AND PAIR, C. 1978. Inference for regular bilanguages. *J. Comput. Sys. Sci. 16*, 100–122.

BIERMANN, A. W. 1972. On the inference of Turing machines from sample computations. *Artif. Intell. 3*, 181–198.

BIERMANN, A. W. 1978. The inference of regular LISP programs from examples. *IEEE Trans. Syst. Man Cybern. SMC-8*, 585–600.

BIERMANN, A. W., AND FELDMAN, J. A. 1972a. A survey of results in grammatical inference. In *Frontiers of Pattern Recognition.* Academic Press, New York.

BIERMANN, A. W., AND FELDMAN, J. A. 1972b. On the synthesis of finite-state machines from samples of their behavior. *IEEE Trans. Comput. C-21*, 592–597.

BIERMANN, A. W., AND KRISHNASWAMY, R. 1976. Constructing programs from example computations. *IEEE Trans. Softw. Eng. SE-2*, 141–153.

BIERMANN, A. W., AND SMITH, D. R. 1977. The hierarchical synthesis of LISP scanning functions. In *Information Processing 77*, B. Gilchrist, Ed. Elsevier North-Holland, New York, pp. 41–45.

BIERMANN, A. W., BAUM, R. I., AND PETRY, F. E. 1975. Speeding up the synthesis of programs from traces. *IEEE Trans. Comput. C-24*, 122–136.

BLUM, L., AND BLUM, M. 1975. Toward a mathematical theory of inductive inference. *Inf. Control 28*, 125–155.

BLUM, M. 1967a. On the size of machines. *Inf. Control 11*, 257–265.

BLUM, M. 1967b. A machine-independent theory of the complexity of recursive functions. *J. ACM 14*, 2 (Apr.) 322–336.

BRANDT, U. 1981. A characterization of identifiable sets. Preprint, Technische Hochschule Darmstadt, F.R.G.

BRAYER, J. M., AND FU, K. S. 1977. A note on the k-tail method of tree grammar inference. *IEEE Trans. Syst. Man Cybern. SMC-7*, 293–300.

BUCHANAN, B. G., AND FEIGENBAUM, E. A. 1978. Dendral and Meta-Dendral: Their applications dimension. *Artif. Intell. 11*, 5–24.

BUNDY, A., AND SILVER, B. 1981. A critical survey of rule learning programs. Tech. Rep. 169, Dept. of Artificial Intelligence, Edinburgh Univ.

CASE, J., AND LYNES, C. 1982. Inductive inference and language identification. In *Proceedings of the International Colloquium on Algorithms, Languages, and Programming (ICALP) 82*, (Barcelona, Spain; June). Springer-Verlag, New York, pp. 107–115.

CASE, J., AND NGO-MANGUELLE, S. 1979. Refinements of inductive inference by Popperian machines. Tech. Rep., Dept. of Computer Science, State Univ. of New York, Buffalo.

CASE, J., AND SMITH, C. 1978. Anomaly hierarchies of mechanized inductive inference. In *Proceedings of the 10th ACM Symposium on Theory of Computing* (San Diego, Calif., May 1–3). ACM, New York, pp. 314–319.

CASE, J., AND SMITH, C. 1983. Comparison of identification criteria for machine inductive inference. *Theor. Comput. Sci. 25*, 193–220.

CHAITIN, G. J. 1974. Information-theoretic limitations of formal systems. *J. ACM 21*, 3 (July), 403–424.

CHAITIN, G. J. 1975. A theory of program size formally identical to information theory. *J. ACM 22*, 3 (July) 329–340.

CHAITIN, G. J. 1976. Information-theoretic characterizations of recursive infinite strings. *Theor. Comput. Sci. 2*, 45–48.

CHEN, K. J. 1982. Tradeoffs in the inductive inference of nearly minimal size programs. *Inf. Control 52*, 68–86.

COOK, C. M., ROSENFELD, A., AND ARONSON, A. R. 1976. Grammatical inference by hill-climbing. *Inf. Sci. 10*, 59–80.

COULON, D., AND KAYSER, D. 1979. Construction of natural language sentence acceptors by a supervised learning technique. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-1*, 94–99.

CRESPI-REGHIZZI, S. 1972. An effective model for grammar inference. In *Information Processing 71*, B. Gilchrist, Ed. Elsevier North-Holland, New York, pp. 524–529.

CRESPI-REGHIZZI, S., AND MANDRIOLI, D. 1980a. Abstract profiles for context-free languages. Tech. Rep. 80-6, Ist. di Elettrotechnica ed Elettronica del Politechnico di Milano, Milan, Italy.

CRESPI-REGHIZZI, S., AND MANDRIOLI, D. 1980b. Inferring grammars by means of profiles: a unifying view. Internal report, Ist. di Elettrotechnica ed Elettronica del Politechnico di Milano, Milan, Italy.

CRESPI-REGHIZZI, S., MELKANOFF, M. A., AND LICHTEN, L. 1973. The use of grammatical inference for designing programming languages. *Commun. ACM 16*, 2 (Feb.), 83–90.

CRESPI-REGHIZZI, S., GUIDA, G., AND MANDRIOLI, D. 1978. Noncounting context-free languages. *J. ACM 25*, 4 (Oct.), 571–580.

DALEY, R. 1973. An example of information and computation resource trade-off. *J. ACM 20*, 4 (Oct.), 687–695.

DALEY, R. 1977. On the inference of optimal descriptions. *Theor. Comput. Sci. 4*, 301–319.

DALEY, R. 1983. On the error correcting power of pluralism in inductive inference. *Theor. Comput. Sci. 24*, 95–104.

DALEY, R., AND SMITH, C. 1983. On the complexity of inductive inference. Tech. Rep. 83-4, Dept. of Computer Science, Univ. of Pittsburgh.

DIETTERICH, T. G., AND MICHALSKI, R. S. 1979. Learning and generalization of characteristic descriptions: Evaluation criteria and comparative review of selected methods. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (Tokyo, Aug.). International Joint Council on Artificial Intelligence, pp. 223–231.

DIETTERICH, T. G., LONDON, R., CLARKSON, K., AND DROMEY, R. 1982. Learning and inductive inference. In *The Handbook of Artificial Intelligence*, P. Cohen and E. Feigenbaum Eds. Kaufman, Los Altos, Calif., pp. 323–512.

DOUCET, P. G. 1974. The syntactic inference problem for D0L sequences. In *L-Systems*, G. Rozenberg and A. Salomaa, Eds. Springer Lecture Notes on Computer Science, vol. 15. Springer-Verlag, New York, pp. 146–161.

FELDMAN, J. A. 1967. First thoughts on grammatical inference. Tech. Rep., Computer Science Dept., Artificial Intelligence Memo 55, Stanford University, Stanford, Calif.

FELDMAN, J. A. 1972. Some decidability results in grammatical inference. *Inf. Control 20,* 244–262.

FELDMAN, J. A., AND SHIELDS, P. 1977. Total complexity and the inference of best programs. *Math. Syst. Theory. 10,* 181–191.

FELDMAN, J. A., GIPS, J., HORNING, J. J., AND REDER, S. 1969. Grammatical complexity and inference. Tech. Rep., Computer Science Dept., Stanford Univ., Stanford, Calif.

FELICIANGELI, H., AND HERMAN, G. 1977. Algorithms for producing grammars from sample derivations: A common problem of formal language theory and developmental biology. *J. Comput. Syst. Sci. 7,* 97–118.

FLANAGAN, P. A. 1981. A discrete model of semantic learning. Tech. Rep. CS-79, Computer Science Dept., Brown University, Providence, R. I.

FREDKIN, E. 1964. Techniques using LISP for automatically discovering interesting relations in data. In *The Programming Language LISP*, Berkey, Ed. M.I.T. Press, Cambridge, Mass., pp. 108–124.

FREIVALD, R. V. 1974. On the limit synthesis of numbers of general recursive functions in various computable numerations. *Sov. Math. Dokl. 15,* 1681–1683.

FREIVALD, R. V. 1975a. Minimal Godel numbers and their identification in the limit. In *Lecture Notes in Computer Science, vol. 32.* Springer-Verlag, New York, pp. 219–225.

FREIVALD, R. V. 1975b. On the complexity and optimality of computation in the limit. *Latv. Gos. Univ. Uch. Zapiski 233,* 155–173 (in Russian).

FREIVALD, R. V. 1979. Finite identification of general recursive functions by probabilistic strategies. In *Proceedings of the Conference on Algebraic, Arithmetic, and Categorial Methods in Computation Theory* (Berlin, Sept. 1979). L. Budach, Ed. Akademie-Verlag, Berlin, DDR, pp. 138–145.

FREIVALD, R. V., AND WIEHAGEN, R. 1979. Inductive inference with additional information. *Electron. Informationsverarb. Kybern. (EIK) 15,* 179–185.

FREIVALD, R. V., KINBER, E. B., AND WIEHAGEN, R. 1982. Inductive inference and computable one-one numberings. *Z. Logik Grundlag. Math. 23,* 463–479.

FRIEDBERG, R. 1958. Three theorems on recursive enumeration. *J. Symb. Logic 23,* 309–316.

FU, K. S. 1975. *Syntactic Methods in Pattern Recognition.* Academic Press, New York.

FU, K. S. 1977. *Syntactic Pattern Recognition, Applications.* Springer-Verlag, New York.

FU, K. S. 1982. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, New York.

FU, K. S., AND BOOTH, T. L. 1975. Grammatical inference: introduction and survey, parts 1 and 2. *IEEE Trans. Syst. Man Cybern. SMC-5,* 95–111, 409–423.

GACS, P. 1983. On the relationship between descriptional complexity and algorithmic probability. *Theor. Comput. Sci., 22,* 1, 2, 71–93.

GAINES, B. R. 1976. Behavior/structure transformations under uncertainty. *Int. J. Man-Mach. Stud. 8,* 337–365.

GAINES, B. R. 1978. Maryanski's grammatical inferencer. *IEEE Trans. Comput. C-28,* 62–64.

GOLD, E. M. 1967. Language identification in the limit. *Inf. Control 10,* 447–474.

GOLD, E. M. 1972. System identification via state characterization. *Automatica 8,* 621–636.

GOLD, E. M. 1978. Complexity of automaton identification from given data. *Inf. Control. 37,* 302–320.

GONZALEZ, R. C., AND THOMASON, M. G. 1978. *Syntactic Pattern Recognition, An Introduction.* Addison-Wesley, Reading, Mass.

GONZALEZ, R. C., EDWARDS, J. J., AND THOMASON, M. G. 1976. An algorithm for the inference of tree grammars. *Int. J. Comput. Inf. Sci. 5,* 145–164.

HARDY, S. 1975. Synthesis of LISP programs from examples. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (Tibilsi, Georgia, USSR, Sept.). International Joint Council on Artificial Intelligence, pp. 268–273.

HAYES-ROTH, F., AND MCDERMOTT, J. 1978. An interference matching technique for inducing abstractions. *Commun. ACM 21,* 5 (May), 401–411.

HORNING, J. J. 1969. A study of grammatical inference. Ph.D. dissertation, Computer Science Dept., Stanford Univ., Stanford, Calif.

JANTKE, K. P. 1979. Natural properties of strategies identifying recursive functions. *Elektron. Informationsverarb. Kybern. (EIK) 15,* 487–496.

JANTKE, K. P., AND BEICK, H. R. 1981. Combining postulates of naturalness in inductive inference.

Elektron. Informationsverarb. Kybern. (EIK) 17, 465–484.

JOUANNAUD, J. P., AND GUIHO, G. 1979. Inference of functions with an interactive system. In *Machine Intelligence 9*, J. E. Hayes, D. Michie, and L. I. Mikulich, Eds. Wiley, New York, pp. 227–250.

JOUANNAUD, J. P., AND KODRATOFF, Y. 1979. Characterization of a class of functions synthesized by a Summers-like method using a B.M.W. matching technique. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence* (Tokyo, Aug.). International Joint Council on Artificial Intelligence, pp. 440–447.

JOUANNAUD, J. P., AND KODRATOFF, Y. 1980. An automatic construction of LISP programs by transformations of functions synthesized from their input–output behavior. *Int. J. Policy Analy. Inf. Syst. 4,* 331–358.

JOUANNAUD, J. P., GUIHO, G., AND TREUIL, T. P. 1977. SISP/1, an interactive system able to synthesize functions from examples. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence* (Cambridge, Mass., Aug.). International Joint Council on Artificial Intelligence, pp. 412–418.

JUNG, H. 1977. Zur Untersuchung von abstrakten interaktiven Erkennungssystemen. Ph.D. dissertation, Humbolt University, Berlin, GDR.

KLETTE, R., AND WIEHAGEN, R. 1980. Research in the theory of inductive inference by GDR mathematicians—A survey. *Inf. Sci. 22,* 149–169.

KNOBE, B., AND KNOBE, K. 1976. A method for inferring context-free grammars. *Inf. Control 31,* 129–146.

KODRATOFF, Y. 1979. A class of functions synthesized from a finite number of examples and a LISP program scheme. *Int. J. Comput. Inf. Sci. 8,* 489–521.

KODRATOFF, Y., AND FARGUES, J. 1978. A sane algorithm for the synthesis of LISP functions from example problems. In *Proceedings of the AISB/GI Conference on Artificial Intelligence* (Hamburg, July). Society for the Study of Artificial Intelligence and Simulation of Behavior/Gesellschaft für Informatik, pp. 169–175.

KOLMOGOROV, A. N. 1965. Three approaches to the quantitative definition of information. *Probl. Inf. Transm. 1,* 1–7.

KUGEL, P. 1977. Induction, pure and simple. *Inf. Control 35,* 276–336.

LAUDAN, L. 1977. *Progress and Its Problems.* University of California Press, Berkeley, Calif.

LEVINE, B. 1981. Derivatives of tree sets with applications to grammatical inference. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3,* 285–293.

LEVINE, B. 1982. The use of tree derivatives and a sample support parameter for inferring tree systems. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-4,* 25–34.

LINDNER, R. 1972. Algorithmische Erkennung. Ph.D. dissertation, Friedrich-Schiller-Universität, Jena, G.D.R.

MACHTEY, M., AND YOUNG, P. 1978. *An Introduction to the General Theory of Algorithms.* Elsevier North-Holland, New York.

MARTIN-LOF, P. 1966. The definition of random sequences. *Inf. Control 9,* 602–619.

MARTIN-LOF, P. 1971. Complexity oscillations in infinite binary strings. *Z. Wahrscheinlichkeitstheor. Verw. Geb. 19,* 225–230.

MARYANSKI, F. J., AND BOOTH, T. L. 1977. Inference of finite-state probabilistic grammars. *IEEE Trans. Comput. C-26,* 521–536.

MICHALSKI, R. 1980. Pattern recognition as rule guided inductive inference. *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-2,* 349–361.

MICHALSKI, R., CARBONELL, J., AND MITCHELL, T., Eds. 1983. *Machine Learning.* Tioga Publ., Palo Alto, Calif.

MICLET, L. 1980. Regular inference with a tail-clustering method. *IEEE Trans. Syst. Man Cybern. SMC-10,* 737–743.

MINICOZZI, E. 1976. Some natural properties of strong identification in inductive inference. *Theor. Comput. Sci. 2,* 345–360.

MITCHELL, T. M. 1982. Generalization as search. *Artif. Intell. 18,* 203–226.

NIX, R. 1983. Editing by example. Ph.D. dissertation, Computer Science Dept., Yale University, New Haven, Conn.

NIX, R. 1984. Editing by example. In *Proceedings of the 11th ACM Symposium on Principles of Programming Languages,* (Salt Lake City, Utah, Jan. 15–18). ACM, New York, pp. 186–195.

OSHERSON, D. N., AND WEINSTEIN, S. 1982. Criteria of language learning. *Inf. Control 52,* 123–138.

PAO, T. W., AND CARR, J. W., III. 1978. A solution of the syntactical induction-inference problem for regular languages. *Comput. Lang. 3,* 53–64.

PERSSON, S. 1966. Some sequence extrapolating programs: A study of representation and modeling in inquiring systems. Ph.D. dissertation, Computer Science Dept., Stanford University, Stanford, Calif.

PIVAR, M., AND FINKELSTEIN, M. 1964. Automation, using LISP, of inductive inference on sequences. In *The Programming Language LISP,* Berkey, Ed. M.I.T. Press, Cambridge, Mass., pp. 125–136.

PIVAR, M., AND GORD, E. 1964. The LISP program for inductive inference on sequences. In *The Programming Language LISP,* Berkey, Ed. M.I.T. Press, Cambridge, Mass., pp. 260–289.

PLOTKIN, G. D. 1970. A note on inductive generalization. In *Machine Intelligence 5.* Elsevier North-Holland, New York, pp. 153–163.

PLOTKIN, G. D. 1971a. Automatic methods of inductive inference. Ph.D. dissertation, Computer Science Dept., Edinburgh University.

PLOTKIN, G. D. 1971b. A further note on inductive generalization. In *Machine Intelligence 6.* Elsevier North-Holland, New York, pp. 101–124.

PODNIEKS, K. M. 1974. Comparing various concepts of function prediction, Part I. *Latv. Gosudarst. Univ. Uch. Zapiski 210,* 68–81 (in Russian).

PODNIEKS, K. M. 1975a. Probabilistic synthesis of enumerated classes of functions. *Sov. Math. Dokl. 16,* 1042–1045.

PODNIEKS, K. M. 1975b. Comparing various concepts of function prediction, Part II. *Latv. Gosudarst. Univ. Uch. Zapiski 233,* 33–44. (in Russian).

PUTNAM, H. 1975. Probability and confirmation. In *Mathematics, Matter and Method.* Cambridge Univ. Press, Cambridge, England.

REYNOLDS, J. C. 1970. Transformational systems and the algebraic structure of atomic formulas. In *Machine Intelligence 5.* Elsevier North-Holland, New York, pp. 135–151.

SHAPIRO, E. 1981a. Inductive inference of theories from facts. Tech. Rep. 192, Dept. of Computer Science, Yale University, New Haven, Conn.

SHAPIRO, E. 1981b. A general incremental algorithm that infers theories from facts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence* (Vancouver, B.C., Canada, Aug.). International Joint Council on Artificial Intelligence, pp. 446–451.

SHAPIRO, E. Y. 1982a. Algorithmic program diagnosis. In *Proceedings of the 9th ACM Symposium on Principles of Programming Languages* (Albuquerque, N. Mex.). ACM, New York, pp. 299–308.

SHAPIRO, E. 1982b. Algorithmic program debugging. Ph.D. dissertation, Computer Science Dept., Yale University, 1982. Published by M.I.T. Press, 1983.

SHAW, D., SWARTOUT, W., AND GREEN, C. 1975. Inferring LISP programs from example problems. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (Tibilsi, Georgia, USSR, Sept.). International Joint Council on Artificial Intelligence, pp. 260–267.

SHINOHARA, T. 1982a. Polynomial time inference of extended regular pattern languages. In *Proceedings, Software Science and Engineering,* Kyoto, Japan.

SHINOHARA, T. 1982b. Polynomial time inference of pattern languages and its applications. In *Proceedings of the 7th IBM Symposium on Mathematical Foundations of Computer Science.*

SIKLOSSY, L., AND SYKES, D. 1975. Automatic program synthesis for example problems. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence* (Tibilsi, Georgia, USSR, Sept.). International Joint Council on Artificial Intelligence, pp. 268–273.

SIMON, H. A., AND KOTOVSKY, K. 1963. Human acquisition of concepts for sequential patterns. *Psych. Rev. 70,* 534–546.

SMITH, C. H. 1982. The power of pluralism for automatic program synthesis. *J. ACM 29,* 4 (Oct.), 1144–1165.

SMITH, D. R. 1982. A survey of the synthesis of LISP programs from examples. In *Automatic Program Construction Techinques,* A. W. Biermann, G. Guiho, and Y. Kodratoff, Eds. Macmillan, New York.

SOLOMONOFF, R. J. 1964. A formal theory of inductive inference. *Inf. Control 7,* 1–22, 224–254.

SOLOMONOFF, R. J. 1975. Inductive inference theory—A unified approach to problems in pattern recognition and artificial intelligence. In *Proceedings of the 4th International Conference Artificial Intelligence* (Tibilsi, Georgia, USSR, Sept.). International Joint Council on Artificial Intelligence, pp. 274–280.

SOLOMONOFF, R. J. 1978. Complexity-based induction systems: comparisons and convergence theorems. *IEEE Trans. Inf. Theor. IT-24,* 422–432.

STOLFO, S. J. 1979. Automatic discovery heuristics for nondeterministic programs from sample execution traces. Ph.D. dissertation, Computer Science Dept., New York University.

STOLFO, S. J., AND HARRISON, M. C. 1979. Automatic discovery of heuristics for nondeterministic programs. Tech. Rep. 7, Computer Science Dept., New York University.

SUMMERS, P. D. 1976. Program construction from examples. Ph.D. dissertation, Computer Science Dept., Yale University, New Haven, Conn.

SUMMERS, P. D. 1977. A methodology for LISP program construction from examples. *J. ACM 24,* 1 (Jan.), 161–175.

THIELE, H. 1973. Lernverfahren zur Erkennung formaler Sprachen. *Kybern. Forsch. 3,* 11–93.

THIELE, H. 1975. Zur Charakterisierung von Erkennungssytemen mit einbettendem Konvergenzbegriff. *Kompliziertheit Lern Erkennungsprozessen 2,* 188–207. Jena, G.D.R.

VAN DER MUDE, A., AND WALKER, A. 1978. On the inference of stochastic regular grammars. *Inf. Control 38,* 310–329.

VEELENTURF, L. P. J. 1978. Inference of sequential machines from sample computations. *IEEE Trans. Comput. C-27,* 167–170.

VERE, S. A. 1980. Multilevel counterfactuals for generalizations of relational concepts and productions. *Artif. Intell. 14,* 139–164.

VERE, S. A. 1981. Constrained *N*-to-1 generalization. Preprint, Jet Propulsion Laboratory, Pasadena, Calif.

WATANABE, S. 1960. Information-theoretical aspects of inductive and deductive inference. *IBM J. Res. Devel. 4,* 208–231.

WATERMAN, D. A., FAUGHT, W. S., KLAHR, P., ROSENSCHEIN, S. J., AND WESSON, R. 1980. Design issues for exemplary programming. Tech. Rep. N-1484-RC, Rand Corporation, Santa Monica, Calif.

WEXLER, K., AND CULICOVER, P. 1980. *Formal Principles of Language Acquisition.* M.I.T. Press, Cambridge, Mass.

WHARTON, R. M. 1974. Approximate language identification. *Inf. Control 26,* 236–255.

WHARTON, R. M. 1977. Grammar enumeration and inference. *Inf. Control 33,* 253–272.

WIEHAGEN, R. 1976. Limes-erkennung rekursiver Funktionen durch spezielle Stratigen. *Elektron. Informationsverarbeit. Kybern. (EIK) 12,* 93–99.

WIEHAGEN, R. 1977. Identification of formal languages. In *Lecture Notes in Computer Science, vol. 53.* Springer-Verlag, New York, pp. 571–579.

WIEHAGEN, R. 1978. Characterization problems in the theory of inductive inference. In *Proceedings of the 5th Colloquium on Automata, Languages, and Programming* (Udine, Italy, July), Lecture Notes on Computer Science, vol. 62. Springer-Verlag, New York, pp. 494–508.

WIEHAGEN, R., AND LIEPE, W. 1976. Charakteristche Eigenschaften von erkennbaren Klassen rekursiver Funktionen. *Elektron. Informationsverarbeit. Kybern. (EIK) 12,* 421–438.

WINSTON, P. 1975. *Learning Structural Descriptions from Examples.* McGraw-Hill, New York.

ZLOOF, M. 1977. Query-by-Example: A data base language. *IBM Syst. J. 16,* 324–343.

ZVONKIN, A. K., AND LEVIN, L. A. 1970. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Rev. 25,* 83–124.

Received November 1982; revised October 1983; final revision accepted February 1984.