INFORMATION AND CONTROL 45, 117–135 (1980)

Inductive Inference of Formal Languages from Positive Data*

DANA ANGLUIN

Department of Mathematics, University of California, Santa Barbara, California 93106

We consider inductive inference of formal languages, as defined by Gold (1967), in the case of positive data, i.e., when the examples of a given formal language are successive elements of some arbitrary enumeration of the elements of the language. We prove a theorem characterizing when an indexed family of nonempty recursive formal languages is inferrable from positive data. From this theorem we obtain other useful conditions for inference from positive data, and give several examples of their application. We give counterexamples to two variants of the characterizing condition, and investigate conditions for inference from positive data that avoids "overgeneralization."

1. INTRODUCTION

The theory of inductive inference may be viewed as an attempt to use our understanding of computation and of effective and efficient computability to model and investigate processes that perform inductive generalizations. One example of such a process is a child who, upon hearing and attempting to produce utterances in some natural language, gradually comes to possess (implicitly) a complicated and substantially "correct" grammar for the language. Another less dramatic case is the frequent use in human communication of illustrative examples to convey all or part of the specification of some algorithmic procedure. Solomonoff (1964) and Gold (1967) discuss these and other motivations more fully.

Gold (1967) gives a definition of correct inductive inference in which, informally speaking, the inferring process is presented successively with a larger and larger corpus of examples, eventually including any particular example, and simultaneously makes a sequence of guesses of the underlying rule being exemplified. If the sequence of guesses eventually converges to a single value which is a correct description of the underlying rule, the inference is correct. If the sequence of guesses oscillates indefinitely, or converges to an incorrect description, the inference is incorrect. This definition and variations of it, general types of algorithmic processes to perform inference, and characteriza-

* This research was funded by the National Science Foundation under Grant No. MCS77-11360. A preliminary version of these results appeared in the Proceedings of the 11th Annual ACM Symposium on Theory of Computing, Atlanta, Ga., May 1979.

tions of classes of sets and functions that are inferrable have been studied by Barzdin and Freivald (1972), Biermann (1972), Blum and Blum (1975), Case and Smith (1978), Feldman (1972), and Kugel (1977), among others.

This paper is concerned with inductive inference of classes of formal languages. A formal language is simply a particular set of strings over some fixed finite alphabet of symbols. We distinguish two fundamentally different ways of presenting a formal language by examples. One way is to give both examples and counterexamples, i.e., a collection of strings that are members of the language, and are so marked, together with a collection of strings that are not members of the language, so marked. The second way is to give only examples, i.e., a collection of strings that are members of the language. The first method of presentation we call "by positive and negative data," the second "by positive data."

Intuitively, an added difficulty in trying to do inference from positive rather than positive and negative data is the problem of "overgeneralization." If in the course of making guesses the inferring process makes a guess that is overly general, i.e., specifies a language that is a proper superset of the true answer, then with positive and negative data there will eventually be a counterexample to the guess, i.e., a string that is contained in the guessed language but is not a member of the true language. No such specific conflict with the examples will occur in the case of inference from positive data.

This difficulty in knowing when to generalize means that inference from positive data is strictly less powerful than inference from positive and negative data. Gold (1967, Theorem I.8) shows that any class of formal languages over an alphabet Σ that contains every finite language together with at least one infinite language over Σ cannot be correctly inferred from positive data. Since this applies even to the class of finite-state languages over Σ, this theorem has been interpreted as showing that inference from positive data is too weak to be of much theoretical interest.

However, the "pattern languages" of Angluin (1979) appear to provide an example of a nontrivial and interesting class of languages for which correct inference from positive data is possible. In fact, it seems that searching for specific classes of languages which are inferrable from positive data may help to generate new approaches to concrete problems of inductive inference that avoid the difficulties of the apparent computational intractability of some previous approaches. (See Gold (1978) and Angluin (1978) on the computational complexity of some concrete inference problems.)

The main result of the present paper is a characterization of those classes of nonempty recursive languages for which correct inference from positive data is possible. This characterization is used to derive other, more easily applied, conditions for the possibility of inference from positive data. We discuss several specific examples of the application of these conditions. Counter-examples are given to two possible variations of the main characterizing condition. We consider the special case of inference from positive data that avoids "overgeneralization," and give sufficient conditions for it.

2. Definitions

Let Σ be a fixed finite alphabet of symbols. The set of all finite strings of symbols from Σ is denoted by Σ^*. The length of a string w is denoted by $|w|$. The concatenation of two strings u and v is denoted by uv or $u \cdot v$.

A *language* is any set of strings over Σ, i.e., any subset of Σ^*. Let L be a nonempty language. A *positive presentation of* L is an infinite sequence $\sigma = s_1, s_2, s_3, \ldots,$ of strings such that the set $\{s_1, s_2, s_3, \ldots\}$ is precisely L. An *indexed family of nonempty languages* is an infinite sequence $L_1, L_2, L_3, \ldots,$ where each L_i is a nonempty language. An *indexed family of nonempty recursive languages* is an indexed family of nonempty languages $L_1, L_2, L_3, \ldots,$ such that there exists an effective procedure to compute the membership function

$$\begin{aligned} f(i, w) &= 1 \qquad \text{if} \quad w \in L_i \\ &= 0 \qquad \text{otherwise.} \end{aligned}$$

An *inference machine* is defined to be a certain type of Turing machine, though clearly an equivalent definition may be given for other formal models of computation. We consider a deterministic Turing machine with input alphabet Σ, a finite tape alphabet Δ, and several one-way infinite tapes: a read-only *sample tape*, a write-only *guess tape*, and a finite number of read–write *scratch tapes*. Each tape is equipped with one head, which is initially positioned at the first square of the tape. The machine has a finite number of *states*, of which there are four distinct distinguished states: the *initial state*, the *request state*, the *answer state*, and the *guess state*. The *finite control* of the machine consists of a finite function that specifies for each tuple consisting of a state of the machine (other than the request state) and the symbols currently scanned by the heads on the sample and scratch tapes, a *move*, consisting of a state of the machine, symbols from Δ to write at the currently scanned squares on the scratch and guess tapes, and a specification for each of the tapes whether its head should be shifted one square to the right, or left, or not at all. We stipulate that no move of the machine may specify that the guess tape head be shifted left, and any move of the machine that writes a nonblank symbol on the guess tape must also shift the guess tape head right.

Let $\sigma = s_1, s_2, s_3, \ldots,$ be an infinite sequence of elements of Σ^*. Let M be an inference machine, as defined above. We define $M[\sigma]$, the sequence of guesses produced by M on input σ as follows. Let the strings of nonblank symbols from Δ be enumerated effectively and take the ith string in this enumeration to represent the positive integer i. Start M in the initial state, with all tapes blank and all tape heads positioned to the first squares of their respective tapes. Initialize the sequence τ to be null. As long as M is not in the request state or the guess state, perform the uniquely determined move to transform the present configuration to its successor. When M enters the request state for the ith time, place s_i followed by blanks on the sample tape, position the sample tape head at the first square, put M in the answer state and continue the computation. When M enters the guess state for the ith time, append the positive integer represented by the current nonblank initial portion of the guess tape as the ith term of the sequence τ, clear the guess tape to all blanks, position the guess tape head to the first square, and then perform whatever move of M applies. Let $M[\sigma]$ be the null, finite, or infinite sequence τ generated by this process carried on indefinitely. (Intuitively, of course, we think of M as from time to time requesting an input and being given the next element of σ, and from time to time producing a guess, which is appended to τ.)

The sequence $M[\sigma]$ is said to *converge to* the positive integer i if and only if either $M[\sigma]$ is infinite and all but finitely many terms of it are equal to i, or $M[\sigma]$ is nonnull and finite, and the last term is equal to i.

Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty languages. An inference machine M is said to *infer the language* L_i *from positive data* if and only if for every positive presentation σ of L_i, there exists a positive integer j such that $M[\sigma]$ converges to j and $L_j = L_i$. M is said to *infer the family* $L_1, L_2, L_3, \ldots,$ *from positive data* if and only if for every $i \geqslant 1$ M infers the language L_i from positive data. An indexed family of nonempty languages $L_1, L_2, L_3, \ldots,$ is *inferrable from positive data* if and only if there exists an inference machine M that infers $L_1, L_2, L_3, \ldots,$ from positive data.

We refer the reader to Rogers (1967) for definitions concerning computability and acceptable Gödel numberings.

3. Characterizing Condition for Positive Inference

In this section we state and prove a condition characterizing when an indexed family of nonempty recursive languages is inferrable from positive data. Informally, this condition requires that for every language L of the family, there exists a "telltale" finite subset T of L such that no language of the family that also contains T is a proper subset of L. Moreover, it must be possible to enumerate effectively some such telltale finite set from any index for L. The point of the telltale subset is that once the strings of that subset have appeared among the sample strings, we need not fear "overgeneralization" in guessing L. This is because the true answer, even if it is not L, cannot be a proper subset of L. In this case, we will eventually see a conflict between the data and L, which will force us to change our guess.

Condition 1. An indexed family of nonempty languages *satisfies Condition* 1 if and only if there exists an effective procedure which on any input $i \geqslant 1$ enumerates a set of strings T_i such that

(i) T_i is finite,

(ii) $T_i \subseteq L_i$, and

(iii) for all $j \geqslant 1$, if $T_i \subseteq L_j$ then L_j is not a proper subset of L_i.

THEOREM 1. *An indexed family of nonempty recursive languages is inferrable from positive data if and only if it satisfies Condition* 1.

Proof. We first assume that $L_1, L_2, L_3, \ldots,$ is an indexed family of nonempty recursive languages that satisfies Condition 1 and describe an inference machine that infer $L_1, L_2, L_3, \ldots,$ from positive data. The machine is defined in stages, beginning with stage 1:

Stage n. Request another sample string. Let S_n denote the set of sample strings read in thus far. Let g be the least positive integer (if any) such that $g \leqslant n$, $S_n \subseteq L_g$, and $T_g^{(n)} \subseteq S_n$, where $T_g^{(n)}$ denotes the set of strings produced in the first n steps of the enumeration of T_g from g. If no such g exists, then go to stage $n + 1$, otherwise output the guess g and go to stage $n + 1$.

This procedure is effective because S_n is always an explicitly given finite set, so $S_n \subseteq L_i$ is decidable by the decidability of "$s \in L_i$?." It should thus be evident that the procedure may be implemented by an inference machine. To see that this procedure infers $L_1, L_2, L_3, \ldots,$ from positive data, let $k \geqslant 1$ be arbitrary and assume that $\sigma = s_1, s_2, s_3, \ldots,$ is any positive presentation of L_k. At each stage n, $S_n = \{s_1, s_2, \ldots, s_n\}$. Let m be the least positive integer such that $L_m = L_k$. For each positive integer $i < m$ either

(i) $L_m - L_i \neq \varnothing$, in which case we choose n_i sufficiently large that $S_n \cap (L_m - L_i) \neq \varnothing$ for all $n \geqslant n_i$, or

(ii) $L_m \subsetneq L_i$, in which case we choose n_i sufficiently large that $T_i^{(n)} = T_i$ for all $n \geqslant n_i$.

Finally, let n_m be sufficiently large that $T_m \subseteq S_n$ for all $n \geqslant n_m$. If $N = \max\{n_1, n_2, \ldots, n_m, m\}$ then for all $n \geqslant N$ we have $m \leqslant n$, $S_n \subseteq L_m$, and $T_m^{(n)} \subseteq T_m \subseteq S_n$. Further, for each positive integer $i < m$, if $L_m - L_i \neq \varnothing$ so $S_n \nsubseteq L_i$, and if $L_m \subsetneq L_i$ then $T_i \nsubseteq L_m$ so since $T_i^{(n)} = T_i$, $T_i^{(n)} \nsubseteq L_m$. Thus m will be the guess of the procedure at stage n. Since $n \geqslant N$ was arbitrary, we see that the sequence of guesses converges to m, an index for L_k. Since k and σ were arbitrary, we see that this procedure infers $L_1, L_2, L_3, \ldots,$ from positive data.

For the converse, suppose M is an inference machine that infers $L_1, L_2, L_3, \ldots$ from positive data. We construct an effective procedure to enumerate for each i a finite set T_i satisfying Condition 1. It is clear that there exists an effective procedure which given i enumerates a positive presentation of i. (Such a procedure might effectively enumerate all strings of Σ^* testing each one for membership in L_i and outputting it if it is a member of L_i. To guarantee that an infinite sequence of strings is enumerated, the procedure can output s_1, the first string it finds in L_i, infinitely often, say after a fixed number of failed membership trials. Recall that each L_i is nonempty by hypothesis.) Given such a procedure, there is another effective procedure to enumerate all finite sequences of elements of L_i (repetitions permitted) in an obvious fashion. The procedure to enumerate T_i from i is defined as follows:

Given i, let $s_1, s_2, s_3, \ldots,$ be an effective enumeration of a positive presentation of L_i. Go to stage 0.

Stage 0. Put s_1 in T_i and let τ_1 be the finite sequence $\langle s_1 \rangle$. Go to stage 1.

Stage n ($n \geqslant 1$). Begin enumerating all finite sequences of elements of L_i, say $\rho_1, \rho_2, \rho_3, \ldots$. Dovetail the computations of M on the input sequences $\tau_n\rho_1, \tau_n\rho_2, \tau_n\rho_3, \ldots,$ (where juxtaposition denotes concatenation of the finite sequences of strings). If we find a sequence $\tau_n\rho_j$ such that the most recent guess of M when it requests the next input beyond τ_n differs from the most recent guess of M when it requests the next input beyond $\tau_n\rho_j$, then we define $\tau_{n+1} = \tau_n\rho_j\langle s_{n+1}\rangle$, add all the elements of τ_{n+1} to T_i, and go to stage $n + 1$.

It is clear that the elements of T_i are (uniformly) effectively enumerable from i. We must see that T_i satisfies Condition 1.

First, assume that this procedure executes an infinite number of stages. Then the limit τ of $\tau_1, \tau_2, \tau_3, \ldots,$ will be an infinite sequence containing all and only the elements of L_i, i.e., is a positive presentation of L_i. (Note that we add s_{n+1} to τ when we go from stage n to stage $n + 1$.) However, M on input τ must change its guess after reading in all of τ_n and before reading in all of τ_{n+1}, for all $n \geqslant 1$. Hence $M[\tau]$ does not converge to any value, so M fails to infer L_i from positive data, a contradiction.

Hence there exists a positive integer n such that this procedure enters stage n and never enters stage $n + 1$. Then T_i is the finite set of elements in τ_n, because we only enlarge T_i when leaving a stage. Furthermore, if ρ is *any* infinite sequence of elements of L_i then M on input $\tau_n\rho$ (that is, the finite sequence τ_n followed by the infinite sequence ρ) never changes its guess after reading in all of τ_n (because if it did, the procedure would eventually test a sufficiently long initial segment ρ_j of ρ to detect this, and go on to stage $n + 1$), and consequently must converge to a guess of an index of L_i for any such $\tau_n\rho$.

Suppose that there exists a positive integer j such that $T_i \subseteq L_j$ and $L_j \subsetneq L_i$. Let ρ be any positive presentation of L_j. Since the elements of τ_n are contained in T_i, $\tau_n\rho$ is also a positive presentation of L_j. By the argument in the preceding paragraph, $M[\tau_n\rho]$ converges to an index of $L_i \neq L_j$, so M fails to infer L_j from positive data, a contradiction.

Thus T_i satisfies Condition 1. ▮

4. Other Conditions

In this section we derive as corollaries of Theorem 1 other (possibly more easily used) conditions for the possibility of correct inference from positive data. The application of these to several examples is given in the next section. Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty recursive languages.

Condition 2. *We say* $L_1, L_2, L_3, \ldots,$ satisfies Condition 2 *provided that, for every* $i \geqslant 1$, *there exists a finite set* $T_i \subseteq L_i$ *such that for every* $j \geqslant 1$, *if* $T_i \subseteq L_j$ *then* L_j *is not a proper subset of* L_i.

Note that Condition 2 is simply Condition 1 with the requirement of effective enumerability of T_i dropped. Also note that the satisfaction of Condition 2 does not depend at all on the indexing of the family of languages, but simply on the set of languages that appear in the sequence.

Corollary 1. *If* $L_1, L_2, L_3, \ldots,$ *is an indexed family of recursive languages that is inferrable from positive data, then it satisfies Condition* 2.

Proof. From Theorem 1, since Condition 1 is stronger than Condition 2. ∎

In Section 6 we give an example to show that Condition 2 is not sufficient for inferrability of $L_1, L_2, L_3, \ldots,$ from positive data. Corollary 1 strengthens the related Theorem I.8 of Gold (1967) in a useful way.

Condition 3. *We say* $L_1, L_2, L_3, \ldots,$ satisfies Condition 3 *if for each nonempty finite set* $S \subseteq \Sigma^*$, *the set* $C(S) = \{L\colon S \subseteq L \text{ and } L = L_i \text{ for some } i\}$ *is of finite cardinality.*

Note that Condition 3 also depends only on the set $\{L_1, L_2, L_3, \ldots\}$ and not on the indexing of it.

Corollary 2. *If* $L_1, L_2, L_3, \ldots,$ *is an indexed family of nonempty recursive languages such that Condition* 3 *is satisfied, then it is inferrable from positive data.*

Proof. Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty recursive languages that satisfies Condition 3. We show that it satisfies Condition 1, and apply Theorem 1. Let $s_1, s_2, s_3, \ldots,$ be an effective enumeration of the elements of Σ^*. For each i and n, $L_i^{(n)}$ will denote the computable finite set $L_i \cap \{s_1, s_2, \ldots, s_n\}$. We define a procedure to enumerate a set T_i from i in stages, beginning with stage 0:

Stage 0. Let t_1 be the least element of L_i. Set $A_1 = \{t_1\}$, put t_1 into T_i and go to stage 1.

Stage n $(n \geqslant 1)$. Search for a pair (j, m) such that $A_n \subseteq L_j$ and $L_j^{(m)} \subsetneq L_i^{(m)}$.

If one is found, set $A_{n+1} = L_i^{(m)}$, put the elements of A_{n+1} into T_i, and go to stage $n + 1$.

Clearly T_i may be (uniformly) effectively enumerated from i. Also, each A_n constructed is an initial segment of L_i and properly contains A_{n-1} if $n > 1$. There are finitely many distinct languages from the given family that contain the string t_1, by Condition 3. Each stage of the procedure must find and "cancel" (by enlarging A_{n+1} so that it is not contained in the language) at least one such language containing t_1. Further, no such language need be cancelled more than once (by the monotonicity of $A_1 \subseteq A_2 \subseteq \cdots$) so there are only finitely many stages in the execution of this procedure. That is, there exists some positive integer n such that this procedure enters stage n and never enters stage $n + 1$. Thus $T_i = A_n$ is of finite cardinality, and clearly $T_i \subseteq L_i$. Assume that for some $j \geqslant 1$, $T_i \subseteq L_j$ and $L_j \subsetneq L_i$. Then for some m, $L_j^{(m)} \subsetneq L_i^{(m)}$, and the procedure in stage n must eventually find this (or some other) pair (j, m) such that $A_n \subseteq L_j^{(m)}$ and $L_j^{(m)} \subsetneq L_i^{(m)}$, and go on to stage $n + 1$, contradicting our choice of n. Thus T_i satisfies Condition 1. ▮

We note that the procedure described above is actually computing in the limit whether $L_j \subseteq L_i$. We are thus led to formulate a condition directly concerning the decidability of inclusion.

COROLLARY 4. *An indexed family of nonempty recursive languages $L_1, L_2, L_3, \ldots$ is said to* satisfy Condition 4 *provided there exists an effective procedure to compute the inclusion function*:

$$\begin{aligned} h(i,j) &= 1 \quad \text{if } L_i \subseteq L_j \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

We note that this condition, like Condition 1, depends essentially upon the indexing of the family.

COROLLARY 3. *Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty recursive languages that satisfies Conditions 2 and 4. Then $L_1, L_2, L_3, \ldots,$ is inferrable from positive data.*

Proof. We show that Condition 1 is satisfied, and apply Theorem 1. A procedure to enumerate a set T_i on input $i \geqslant 1$ is as follows, beginning with stage 0.

Stage 0. Let $A_1 = \varnothing$ and go to stage 1.

Stage n ($n \geqslant 1$). Search for an integer $j \geqslant 1$ such that $A_n \subseteq L_j$ and $L_j \subsetneq L_i$. If such a j is found, let m be the least positive integer such that $L_j^{(m)} \subsetneq L_i^{(m)}$ (where $L_k^{(m)}$ is as in the proof of Corollary 2). Set $A_{n+1} = L_i^{(m)}$, put the elements A_{n+1} into T_i, and go to stage $n + 1$.

Clearly the elements of $T_i = A_1 \cup A_2 \cup \cdots$ may be (uniformly) effectively enumerated from i. Each constructed A_n is an intial segment of L_i and properly contains A_{n-1} if $n > 1$. Thus $T_i \subseteq L_i$. Suppose this procedure executes infinitely many stages. By Condition 2 there exists a finite set $R_i \subseteq L_i$ such that for no $j \geqslant 1$ do we have $R_i \subseteq L_j$ and $L_j \subsetneq L_i$. Hence for some sufficiently large n, at stage n, $R_i \subseteq A_n$, so the search in stage n must fail to find any $j \geqslant 1$ such that $A_n \subseteq L_j$ and $L_j \subsetneq L_i$, so the procedure never reaches stage $n + 1$, a contradiction. Hence there exists some integer $n \geqslant 1$ such that this procedure enters stage n and never enters stage $n + 1$. Hence $T_i = A_n$ is finite, and for every $j \geqslant 1$ such that $T_i \subseteq L_j$, it must be the case that L_j is not a proper subset of L_i. Thus T_i satisfies Condition 1. ∎

5. Examples

In this section we give several examples of the application of the conditions for inference from positive data that were derived above.

Example 1. (This is the example that originally motivated this investigation.) Let $X = \{x_1, x_2, x_3, \ldots\}$ be an infinite set of symbols disjoint from Σ. By a *pattern* we shall mean a nonnull finite string over the alphabet $\Sigma \cup X$. If p is a pattern, then the *language of* p, denoted $L(p)$, is the set of elements of Σ^* that can be obtained from p by substituting a nonnull element $s_i \in \Sigma^*$ for each occurrence of the symbol x_i in p, for all $i \geqslant 1$. For example, if $\Sigma = \{0, 1, 2, \ldots, 9\}$ and p is $3x_6x_642x_{37}$ then some elements of $L(p)$ are 3004213, 31221224255, 311421. The *pattern languages* are the sets $L(p)$ where p is any pattern. We choose some straightforward concrete representation $\bar{p}$ of the pattern p as a string of symbols over some finite alphabet, choose an effective enumeration of the concrete representations: $\bar{p}_1, \bar{p}_2, \bar{p}_3, \ldots$, and index the pattern languages as $L(p_1), L(p_2), L(p_3), \ldots$. It is not difficult to see that this will be an indexed family of nonempty recursive languages. Furthermore, this family satisfies Condition 3 because for any finite nonempty set of strings $S \subseteq \Sigma^*$, if l is the minimum length of any string in S, then any pattern p such that $S \subseteq L(p)$ must have length $\leqslant l$, and there are only finitely many distinct pattern languages generated by patterns of length $\leqslant l$. Hence by Corollary 2, the pattern languages (with this indexing) are inferrable from positive data.

For more details of these and other results concerning the pattern languages, see Angluin (1979).

For the next two examples, we assume that the reader is familiar with the definitions and elementary properties of regular expressions; see, for example, Aho, Hopcroft, and Ullman (1974).

Example 2. *Let* $e_1, e_2, e_3, \ldots$, be an effective enumeration of all regular expressions over the alphabet $\{0, 1\}$ containing no operators other than $\cdot$ (concatenation) and $*$ (Kleene star). Then $L(e_1), L(e_2), L(e_3), \ldots,$ is an indexed family of nonempty recursive languages that is not inferrable from positive data. To see this, we shall prove that Condition 2 is violated. Consider the language $L' = L((0^*1^*)^*)$. Clearly $L' = \Sigma^*$. If $T = \{t_1, t_2, \ldots, t_k\}$ is any nonempty finite subset of L', then consider the language $L_T = L((t_1)^*(t_2)^* \cdots (t_k)^*)$. Clearly $T \subseteq L_T$. Also, if $m = 1 + \sum_{i=1}^{k} | t_i |$ and s is the string $0^m 1^m 0^{m+1} \cdots 0^{2m} 1^{2m}$, then it is not difficult to see that $s \notin L_T$, so $L_T \subsetneq L'$. Thus no "telltale" finite set exists for L', and Condition 2 fails (We note that Theorem I.8 of Gold (1967) does not apply to this example.)

EXAMPLE 3. Let $e_1, e_2, e_3, \ldots,$ be an effective enumeration regular expressions over the alphabet $\{0, 1\}$ that contain no operators other than $\cdot$ (concatenation) and $^+$ (Kleene plus, i.e., repetition an arbitrary *positive* number of times). Then $L(e_1), L(e_2), L(e_3), \ldots,$ is an indexed family of nonempty recursive languages that is inferrable from positive data. We show that this family satisfies Condition 3, and apply Corollary 2. For any expression e_i from this enumeration let $l(e_i)$ denote the length of e_i (as a string of symbols from $\{0, 1, \cdot, ^+, (,)\}$) and let $c(e_i)$ denote the number of occurrences of symbols from $\{0, 1\}$ in e_i. For any such e_i, if $s \in L(e_i)$ then $| s | \geqslant c(e_i)$, and also there exists another expression e_j from the enumeration such that $l(e_j) \leqslant 10c(e_i)$ and $L(e_j) = L(e_i)$. (This uses the fact that $L(((f)^+)^+) = L((f)^+)$ for any regular expression f.) Let S be an arbitrary finite nonempty set of strings over $\{0, 1\}$ and let m be the minimum length of any string in S. For any e_i such that $S \subseteq L(e_i)$, $c(e_i) \leqslant m$ and there exists some e_j with $l(e_j) \leqslant 10c(e_i) \leqslant 10m$ such that $L(e_j) = L(e_i)$. Since there are only finitely many expressions e_j from the enumeration such that $l(e_j) \leqslant 10m$, there are only finitely many distinct languages in the indexed family that contain S as a subset, so Condition 3 is satisfied.

EXAMPLE 4. In this example we assume some computable encoding of positive integers as strings over Σ and discuss sets of positive integers. If n is any positive integer let $I(n)$ be the set of all positive integral multiples of n. Let the collection of all finite nonempty sets of prime positive integers be $T_1, T_2, T_3, \ldots,$ indexed, for example, in order of increasing $\prod_{p \in T_i} p$. Let $R_i = \bigcup_{p \in T_i} I(p)$. Then $R_1, R_2, R_3, \ldots,$ is an indexed family of nonempty recursive sets. For any i, $T_i \subseteq R_i$ and for any j, if $T_i \subseteq R_j$ then $R_i \subseteq R_j$, so the sets T_i satisfy Condition 2. Also, T_i is recursively enumerable from i, so Condition 1 is satisfied, and $R_1, R_2, R_3, \ldots,$ is inferrable from positive data by Theorem 1. Alternatively we can see that $R_i \subseteq R_j$ if and only if $T_i \subseteq T_j$, so Condition 4 is satisfied and Corollary 3 applies. We note that Condition 3 does not apply; for example, if $S = \{2\}$ then for any prime p, $I(2) \cup I(p)$ contains S, and there are infinitely many distinct such sets in the family $R_1, R_2, R_3, \ldots$.

6. Variations of Condition 1

In this section we show that the requirement in Condition 1 that T_i be recursively enumerable from i cannot be dropped entirely or strengthened to recursive without losing the sufficiency or necessity of the condition for inference from positive data.

Theorem 2. *There exists an indexed family of nonempty recursive languages satisfying Condition 2 that is not inferrable from positive data.*

Proof. Let M be an arbitrary inference machine. We first show how to construct an indexed family of nonempty recursive languages that M does not infer from positive data. This will be the basis of a construction that "cancels" all inference machines. For convenience, we will use sets of positive integers; we assume some straightforward encoding of positive integers as strings over Σ to obtain the correspondence to languages over Σ.

We describe a procedure to enumerate the elements of sets R_1, R_2, R_3,..., each of which is a subset of the set of powers of 2, $\{2, 4, 8, 16, 32,...\}$. The set R_1 is special; it consists simply of the even power of 2, that is, $R_1 = \{2^{2m}: m \geqslant 1\}$. Each of the other sets is, during the course of the procedure, in one of three states: live, terminating, or terminated. Only live or terminating sets may have new elements added.

During the procedure, we maintain three variables: σ which is a nonempty finite sequence of positive integers, b, which is either a positive integer or $+\infty$, and s which may take on the values "following 1" and "terminating i" for every positive integer i. The instruction to "augment σ" will mean the following. Let x be the current last element of σ; if $4x \leqslant b$ then append $4x$ to the end of σ, otherwise append (another) x to the end of σ. The procedure is defined in stages, beginning with stage 0.

Stage 0. Initialize all sets R_i for $i \geqslant 2$ to be live and contain the element 4. Set $\sigma = \langle 4 \rangle$, $b = +\infty$, $s =$ "following 1," and $g_0 =$ null. Go to stage 1.

Stage n ($n \geqslant 1$). Run M on input σ for n steps. If M is in the guess state after n steps, then set $g_n =$ the index on the guess tape at this point. Otherwise, set $g_n = g_{n-1}$.

(a) If $s =$ "following 1" and $g_n = 1$ then let m be the minimum index of any live set, set $b = 2^{2n}$, set the state of R_m to terminating, add 2^{2n+2} to all remaining live sets, augment σ, set $s =$ "terminating m," and go to stage $n + 1$.

(b) If $s =$ "following 1" and $g_n \neq 1$ then if g_n is either null or the index of a terminated set then add 2^{2n+2} to all live sets, augment σ, and go to stage $n + 1$. However, if g_n is the index of a live set, then add 2^{2n+1} to the set with index g_n, mark it terminated, add 2^{2n+2} to all live sets, augment σ, and go to stage $n + 1$.

(c) If $s =$ "terminating i" and $g_n = i$ then add 2^{2n+1} to R_i and mark it terminated, set $b = +\infty$, add 2^{2n+2} to all live sets, augment σ, set $s =$ "following 1" and go to stage $n + 1$.

(d) If $s =$ "terminating i" and $g_n \neq i$ then add 2^{2n+2} to all live sets, augment σ, and go to stage $n + 1$.

The underlying idea of this construction is to enumerate elements of R_1 until M switches its guess to 1, then to enumerate elements of some finite subset R_i of R_1 until M switches its guess to i at which time we add an element to R_i so that it is *not* a subset of R_1 and go back to enumerating elements of R_1 again. The values of s are the states of this process: "following 1" means that we are enumerating elements of R_1 waiting for M to switch its guess to 1; "terminating i" means that we are enumerating elements of R_i, waiting for M to switch its guess to i. If ever M fails to switch its guess to 1, the value of σ will be an enumeration of R_1 for which M fails to infer R_1. If M fails to switch its guess to the finite subset R_i, then σ will be an enumeration of R_i for which M fails to identify R_i.

Clearly $R_i \neq \varnothing$ for all $i \geqslant 1$. For each $i \geqslant 2$, R_i contains only powers of 2, and 2^{2n+1} or 2^{2n+2} is an element of R_i if and only if it is added to R_i at stage n of this procedure. Thus there exists an effective procedure to decide whether $m \in R_i$ given positive integers m and i. Thus R_1, R_2, R_3,... is an indexed family of nonempty recursive sets. To see that Condition 2 is satisfied we consider the two possible cases:

(i) Every set that is marked "terminating" is eventually marked "terminated." In the limit, the family consists of a collection of copies of R_1 (those sets that are never marked "terminating") and a collection of terminated finite sets. When a set is marked "terminated," an odd power of 2 is added as its last element, so it cannot be a proper subset of R_1. If for each terminated finite set R_i we take $T_i = R_i$ and for each $R_i = R_1$ we take $T_i = \{4\}$, these sets will satisfy Condition 2 (since no set in the family is a proper subset of R_1).

(ii) Exactly one set, say R_t, is marked "terminating" but never marked "terminated." In the limit the family consists of a collection of copies of R_1 (those sets never marked "terminating"), a collection of terminated finite sets, and R_t, which is a finite subset of R_1. Let T be a finite subset of R_1 such that $T - R_t \neq \varnothing$. For each finite R_i, let $T_i = R_i$, and for those $R_i = R_1$, take $T_i = T$. Then since no terminated finite set is a subset of R_1 and $T \not\subseteq R_t$, these sets T_i will satisfy Condition 2.

To see that M fails to infer R_1, R_2, R_3,..., we consider the above two cases again.

(i) Every set marked "terminating" is eventually marked "terminated." Then it is easy to see that, as the stage number n tends to infinity, the value of b is unbounded and σ approaches an infinite sequence $\hat{\sigma}$ that is a positive presentation of L_1. Also, for infinitely many stages, s = "following 1." Thus $M[\hat{\sigma}]$ cannot converge to 1, for then some R_i would be marked "terminating" and never marked "terminated". If $M[\hat{\sigma}]$ converges to some $i \neq 1$ then the set R_i must be marked either "terminating" or "terminated" at some point and is therefore finite and not equal to R_1. Thus $M[\hat{\sigma}]$ must either fail to converge, or else converge to some i such that $R_i \neq R_1$, so M does not infer R_1 from positive data.

(ii) Exactly one set, say R_t, is marked "terminating" and never marked "terminated." When R_t is marked "terminating", the guess of M is 1. Subsequently, all remaining live sets are extended to be equal to R_1 in the limit. All terminated sets contain an odd power of 2. When it is marked "terminating," R_t contains a finite initial segment of R_1 ; b is set to the largest element of R_t and is never subsequently changed, so in the limit σ approaches an infinite sequence $\hat{\sigma}$ that is a positive presentation of R_t. But M never subsequently guesses t (otherwise R_t would be marked "terminated"), and no other positive integer is an index of R_t, so M does not infer R_t from positive data.

In either case, we see that M fails to infer $R_1, R_2, R_3, \ldots,$ from positive data. For the general construction, it is easy to see that there exists an enumeration $M_1, M_2, M_3, \ldots,$ of all inference machines such that the above construction to cancel M_i is uniformly effective given i. Thus we may simply use this construction, with powers of the ith prime number p_i in place of powers of 2 to cancel the machine M_i, for all $i \geqslant 1$. If $\langle m, n\rangle$ is a computable pairing function we define $R_{\langle i,1\rangle}, R_{\langle i,2\rangle}, R_{\langle i,3\rangle}, \ldots,$ to be the sets constructed to cancel M_i. (In this construction, if the guess of M_i at stage n is $g = \langle s, t\rangle$, where $s \neq i$ then we set g_n = null.) The resulting $R_1, R_2, R_3, \ldots,$ is an indexed family of nonempty recursive languages that satisfies Condition 2 and is not inferrable from positive data. ∎

We next define a version of Condition 1 in which the sets T_i are required to be recursive rather than just recursively enumerable.

CONDITION 1′. *An indexed family of nonempty recursive sets $L_1, L_2, L_3, \ldots,$ is said to* satisfy Condition 1′ *provided there exists an effective procedure which on input i outputs the elements of a finite set T_i and then halts, where $T_i \subseteq L_i$ and for all $j \geqslant 1$, if $T_i \subseteq L_j$ then L_j is not a proper subset of L_i.*

THEOREM 3. *There exists an indexed family of nonempty recursive languages that is inferrable from positive data but does not satisfy Condition* 1′.

Proof. As in the proof of Theorem 2 we consider sets of positive integers and assume some straightforward correspondence to sets of strings over Σ. Let $M_1, M_2, M_3, \ldots,$ be an enumeration of Turing machines giving an acceptable Gödel numbering of the partial recursive functions (see Rogers (1967)).

Define for each $i \geqslant 1$

$$\begin{aligned} f(i) &= n && \text{if } M_i \text{ on input } i \text{ halts in exactly } n \text{ steps} \\ &= +\infty && \text{if } M_i \text{ on input } i \text{ doesn't halt} \\ R_{2i-1} &= \{p_i^n : n \geqslant 1\} \\ R_{2i} &= \{p_i^n : n \leqslant f(i)\}, \end{aligned}$$

where p_i denotes the ith prime number. Clearly for each $i \geqslant 1$, either $R_{2i} = R_{2i-1}$ (if $f(i) = +\infty$), or R_{2i} is a finite subset of R_{2i-1}.

There is an effective procedure to determine whether $m \in R_i$ given m and i, as follows. If $i = 2j - 1$ and $m = p_j^n$ for $n \geqslant 1$, then $m \in R_i$. If $i = 2j$ and $m = p_j^n$ for some $n \geqslant 1$ then run M_j on input j for $n - 1$ steps; if M has not halted in that time then $m \in R_i$. In all other cases, $m \notin R_i$. Thus $R_1, R_2, R_3, \ldots,$ is an indexed family of nonempty recursive sets.

To see that it is inferrable from positive data, we define the following procedure, beginning with stage 1.

Stage n. Request another input. Let S_n denote the set of inputs read so far. If the elements of S_n are powers of p_i and $S_n \subseteq R_{2i}$ then output the guess $2i$; otherwise, output the guess $2i - 1$.

This procedure may be implemented by an inference machine (using the decidability of "$m \in R_i$?" from m and i), and is easily seen to infer $R_1, R_2, R_3, \ldots,$ in the limit from positive data.

Finally, assume that $R_1, R_2, R_3, \ldots,$ satisfies Condition 1′. Consider the following procedure.

On input i, compute the finite set T_{2i-1} and output 0 if $T_{2i-1} \subseteq R_{2i}$. Output 1 otherwise.

Since Condition 1′ allows us to compute all of the elements of T_{2i-1} by a procedure that halts, the procedure above is effective. However, for every $i \geqslant 1$ $T_{2i-1} \subseteq R_{2i}$ if and only if $R_{2i} = R_{2i-1}$ if and only if M_i does not halt on input i. Hence this procedure solves the halting problem, a contradiction. Thus $R_1, R_2, R_3, \ldots,$ does not satisfy Condition 1′. ▮

7. Avoiding Overgeneral Inferences

In this section we consider a further restriction on an indexed family of languages that guarantees that there exists some inference machine that infers the family from positive data and, intuitively speaking, never changes its guess unless a change is required to maintain consistency with the input data. Thus the machine never makes an overgeneralization that cannot be detected by conflict with the sample. First we consider certain properties an inference machine may have.

Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty languages. Let M be an inference machine. We say M is *consistent on* $L_1, L_2, L_3, \ldots,$ provided that for every $i \geqslant 1$ and every positive presentation σ of L_i, whenever M on input σ enters the guess state, the language guessed contains all the input strings read in up to that point in the computation. We say M is *responsive on* $L_1, L_2, L_3, \ldots,$ provided that for every $i \geqslant 1$ and every positive presentation σ of L_i, between any two distinct steps of the computation of M on input σ in which M is in the request state, there exists a step in which M is in the guess state. We say M is *conservative on* $L_1, L_2, L_3, \ldots,$ provided that for every $i \geqslant 1$ and every positive presentation σ of L_i, if M on input σ makes the guess i at some step and then makes the guess $j \neq i$ at some subsequent step, then L_i must fail to contain some input string read prior to the later step.

Intuitively, "consistent" means whenever M makes a guess it is consistent with the sample read in so far, "resposive" means that M makes at least one guess in response to each input before requesting another input, and "conservative" means that M only changes a guess when it conflicts with the sample read in so far.

Remark. The inference machine M constructed in the proof of Theorem 1 is consistent on the given family of languages by construction. We may easily modify it to be responsive on the family as well; if M is about to go to stage $n + 1$ without making a guess during stage n, it instead searches for the least i such that $S_n \subseteq L_i$ and guesses i. This search can only fail if the input is not a positive presentation of any language of the family, and gives a consistent guess when it succeeds. Thus the inference machines of Corollaries 2 and 3 may also be taken to be consistent and responsive on the respective families of languages.

THEOREM 4. *There exists an indexed family of nonempty recursive languages* $L_1, L_2, L_3, \ldots,$ *that is inferrable from positive data but is such that no inference machine* M *that infers* $L_1, L_2, L_3, \ldots,$ *from positive data can be conservative on* $L_1, L_2, L_3, \ldots$.

Proof. We first show how to "cancel" a single inference machine M. (As in the proofs of Theorems 2 and 3 we use sets of positive integers assuming some straightforward correspondence to languages over Σ.)

Define $R_1 = \{2^m : m \geqslant 1\}$. Let σ be the sequence 2, 4, 8, 16, 32,..., of positive powers of 2. Define R_i for $i > 1$ as follows. Run the computation of M on input σ for i steps. If during this computation, M guesses 1, then let U_i be the set of input strings read by M up to the first time it guesses 1. If M does not guess 1 during the first i steps of its computation on σ then let $U_i = \varnothing$. Define $R_i = U_i \cup \{2\}$.

Clearly $R_1, R_2, R_3, \ldots,$ is an indexed family of nonempty recursive languages. Consider the computation of $M[\sigma]$. There are two cases to consider. If M on σ never guesses 1 then $R_i = \{2\}$ for all $i > 1$ and M fails to infer R_1 from positive data. If M on σ eventually guesses 1, then let i be the first step at which M on σ guesses 1. Let $\hat{\sigma}$ be the finite initial segment of σ read by M up to step i, followed by an infinite sequence of 2's. The sequence $\hat{\sigma}$ is a positive presentation of the finite set R_i. Consider M on input $\hat{\sigma}$: at step i we know that it guesses 1. If M never subsequently changes its guess, it fails to infer R_i from positive data. On the other hand, if M subsequently changes its guess, it fails to be conservative on $R_1, R_2, R_3, \ldots$, because R_1 is consistent with every initial segment of $\hat{\sigma}$.

Thus in either case M must either fail to infer $R_1, R_2, R_3, \ldots$, from positive data or fail to be conservation on $R_1, R_2, R_3, \ldots$. To cancel all machines, we take a standard enumeration $M_1, M_2, M_3, \ldots$, of inference machines and some computable pairing function $\langle i, j\rangle$, and define $R_{\langle i,1\rangle}, R_{\langle i,2\rangle}, R_{\langle i,3\rangle}, \ldots$, as above with $\langle i, 1\rangle$ in place of 1 and powers of the ith prime p_i in place of powers of 2, in order to cancel M_i. The resulting $R_1, R_2, R_3, \ldots$, is an indexed family of nonempty recursive languages such that no inference machine can both infer the family from positive data and be conservative on it. To see that this family is inferrable from positive data, we consider the following procedure to enumerate a "telltale" set T_i from i:

On input $i = \langle j, k\rangle$, if $k > 1$ then simply compute and output all the (finitely many) elements of R_i. If $k = 1$ then initially output p_j and p_j^2 and begin running M_j on the input $\sigma_j = p_j, p_j^2, p_j^3, \ldots$. If M_j ever guesses the output $\langle j, 1\rangle$ then let r be the least positive integer such that M on σ_j has not read the input p_j^r when it first guesses $\langle j, 1\rangle$. Output p_j^r and halt.

Clearly T_i will be a finite set, recursively enumerable from i. For each finite R_i, $T_i = R_i$ and so satisfies Condition 1. If R_i is infinite, i.e., $i = \langle j, 1\rangle$ for some j, then T_i contains at least two powers of p_j and so is not contained in any $R_{\langle s,t\rangle}$ with $s \neq j$. There are two cases. If M_j on $\sigma_j = p_j, p_j^2, p_j^3, \ldots$, never guesses $\langle j, 1\rangle$, then $R_{\langle j,k\rangle} = \{p_j\}$ for all $k > 1$, so $T_i \nsubseteq R_{\langle j,k\rangle}$ for all $k > 1$. If M_j on σ_j eventually guesses $\langle j, 1\rangle$, then p_j^r is chosen so that $p_j^r \notin R_{\langle j,k\rangle}$ for all $k > 1$, and placed in T_i. Hence in any case the sets T_i satisfy Condition 1, and we apply Theorem 1 to conclude that $R_1, R_2, R_3, \ldots$, is inferrable from positive data. ▮

We note that the family $R_1, R_2, R_3, \ldots$, constructed in the preceding proof also satisfies Condition 3. Thus, without some additional restriction, we cannot guarantee correct conservative inference from positive data. Let $L_1, L_2, L_3, \ldots$, be an indexed family of nonempty languages. If $S \subseteq \Sigma^*$ and $i \geqslant 1$ then we say i is *descriptive of S* provided $S \subseteq L_i$ and for all $j \geqslant 1$ such that $S \subseteq L_j$, L_j is not a proper subset of L_i.

CONDITION 5. *An indexed family of nonempty recursive languages $L_1, L_2, L_3, \ldots$, is said to satisfy Condition 5 provided there exists a partial recursive function f such that $f(S)$ is some i that is descriptive of S whenever S is a nonempty finite set of strings such that there exists some j that is descriptive of S.*

THEOREM 5. *Let $L_1, L_2, L_3, \ldots,$ be an indexed family of nonempty recursive languages that satisfies Conditions* 3 *and* 5. *Then there exists an inference machine M that infers $L_1, L_2, L_3, \ldots,$ from positive data and is consistent, responsive, and conservative on $L_1, L_2, L_3, \ldots$.*

Proof. Let f be a partial recursive function as specified in Condition 5 for $L_1, L_2, L_3, \ldots$. We define an inference procedure starting with stage 0.

Stage 0. Set $g_1 =$ "none" and go to stage 1.

Stage n ($n \geqslant 1$). Request another input. Let S_n denote the set of inputs read so far. If for some integer i, $g_n = i$ and $S_n \subseteq L_i$ then set $g_{n+1} = g_n$, output g_{n+1} and go to stage $n + 1$. Otherwise, let $g_{n+1} = f(S_n)$, output g_{n+1} and go to stage $n + 1$.

This may be implemented by an inference machine M using a subprocedure to compute f; if the computation of $f(S_n)$ diverges then M simply never gets to stage $n + 1$. By construction M is responsive and conservative on $L_1, L_2, L_3, \ldots$.

Let $i \geqslant 1$ be arbitrary, and let $\sigma = s_1, s_2, s_3, \ldots,$ be a positive presentation of L_i. For each n let $C_n = \{L: \{s_1, s_2, \ldots, s_n\} \subseteq L$ and for some $j \geqslant 1$, $L_j = L\}$. Since $L_i \in C_n$, C_n is nonempty, and by Condition 3, C_n has finite cardinality. Thus it must contain a minimal element (with respect to the set-containment ordering), and $f(\{s_1, s_2, \ldots, s_n\})$ must be defined and equal to an integer j such that $L_j \in C_n$. Hence for every $n \geqslant 1$, M eventually reaches stage n with $S_n = \{s_1, s_2, \ldots, s_n\}$, and guesses $g_{n+1} = j$, where $S_n \subseteq L_j$. Thus M is consistent on $L_1, L_2, L_3, \ldots$.

Furthermore, for each $L \in C_1$ either

(i) $L_i \subseteq L$, in which case let $n(L) = 1$, or

(ii) $L_i - L \neq \varnothing$, in which case let $n(L)$ be sufficiently large that $S_n - L \neq \varnothing$ for all $n \geqslant n(L)$.

Let $N = \max\{n(L): L \in C_1\}$. By construction, once M guesses an index of L_i it never subsequently changes its guess. At any stage $n \geqslant N$, if $S_n \subseteq L_j$ then $L_j \in C_1$ so by the choice of N either $L_j = L_i$ or $L_i \subsetneq L_j$, so an index of L_i must be guessed. Hence M infers $L_1, L_2, L_3, \ldots,$ from positive data. ▌

Referring to Example 3 in Section 5, we saw that Condition 3 is satisfied for the class of languages described by regular expressions using only the operations of concatenation and Kleene plus. The argument given there enables us to compute expressions representing the finitely many languages from the class containing a given finite nonempty sample set S, and because containment is decidable for languages represented by regular expressions, we may effectively find a minimal such language. Thus Condition 5 is also satisfied for this family, and Theorem 5 applies.

A somewhat more elaborate argument (see Angluin (1979)) is required to see that the family of pattern languages (Example 1) satisfies Condition 5, and thus Theorem 5 applies in this case as well.

We note that although the family of sets in Example 4 does not satisfy Condition 3, we can directly construct an inference machine that infers the family from positive data and is consistent, responsive, and conservative on it. There exists a recursive function f that with input S, a nonempty finite set of positive integers not containing 1, finds a set of primes Q of minimum cardinality such that every element of S is a positive multiple of an element of Q. (We remark that this problem is NP-hard if the integers are encoded in base $b > 1$, by a straightforward reduction from set-covering.) Using this f in the procedure described in Theorem 5 gives the desired result. Thus the conditions in Theorem 5 are sufficient but not necessary.

8. Remarks

We have not investigated the question of when an indexed family of nonempty recursively enumerable languages is inferrable from positive data, since the motivating applications all seem to be families of recursive languages. It is not clear how useful the conditions in Corollary 3 will prove in practice. Perhaps as in Example 4, Corollary 3 is generally no easier to apply than Theorem 1 itself.

Acknowledgment

It is a pleasure to acknowledge the contribution, in the form of useful discussions and a key suggestion, of Professor Manuel Blum to the work reported herein.

Received: June 29, 1979; revised: January 29, 1980

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison–Wesley, Reading, Mass.

Angluin, D. (1978), On the complexity of minimum inference of regular sets, *Inform Contr.* **39**, 337–350.

Angluin, D. (1979), Finding patterns common to a set of strings, *in* "Proceedings, 11th Annual ACM Symposium on Theory of Computing," pp. 130–141.

Barzdin, Ja.M. and Freivald, R. V. (1972), On the prediction of general recursive functions, *Soviet Math. Dokl.* **13**, 1224–1228.

Biermann, A. W. (1972), On the inference of Turing machines from sample computations, *Artificial Intelligence* **3**, 181–198.

Blum, L. and Blum, M. (1975), Toward a mathematical theory of inductive inference, *Inform. Contr.* **28**, 125–155.

CASE, J. AND SMITH, C. (1978), Anomaly hierarchies of mechanized inductive inference, *in* "Proceedings, 10th Annual ACM Symposium on Theory of Computing," pp. 314–319.

FELDMAN, J. A. (1972), Some decidability results on grammatical inference and complexity, *Inform. Contr.* **20**, 244–262.

GOLD, E. M. (1967), Language identification in the limit, *Inform. Contr.* **10**, 447–474.

GOLD, E. M. (1978), Complexity of automaton identification from given data, *Inform. Contr.* **37**, 302–320.

KUGEL, P. (1977), Induction, pure and simple, *Inform. Contr.* **35**, 276–336.

ROGERS, H., JR. (1967), "Theory of Recursive Functions and Effective Computability," McGraw–Hill, New York.

SOLOMONOFF, R. (1964), A formal theory of inductive inference, *Inform. Contr.* **7**, 1–22, 234–254.