INFORMATION AND CONTROL 39, 337–350 (1978)

On the Complexity of Minimum Inference of Regular Sets*

DANA ANGLUIN

Computer Science Department, Edinburgh University, Edinburgh EH9 3JZ, Scotland

We prove results concerning the computational tractability of some problems related to determining minimum realizations of finite samples of regular sets by finite automata and regular expressions.

1. INTRODUCTION

In this paper we consider the computational problem of finding a smallest finite-state description, in some specified system of description, compatible with a given finite positive and negative sample of a regular set. A procedure which solves this problem may be used to perform identification in the limit of the regular sets. Work on the general topic of identification in the limit, or algorithmic inductive inference, may be found in Gold (1967), Feldman (1972), Blum and Blum (1975).

Algorithms hitherto proposed to solve problems of this kind are exhaustive search procedures, for example, Horning (1969), Biermann (1974), Wharton (1977). Gold (1974) has shown that in general the problem is unlikely to admit of a polynomial time algorithm, that is,

THEOREM 1 (Gold). *The problem of determining, for a given finite sample S and positive integer t, whether there exists a deterministic finite automaton of at most t states compatible with S, is NP-complete.*

On the other hand, if the sample is required to classify all strings not exceeding a given length, we have the following result of Trakhtenbrot and Barzdin (1973):

THEOREM 2 (Trakhtenbrot and Barzdin). *There is a polynomial time algorithm which for any uniform-complete sample S finds a deterministic finite automaton of the minimum possible number of states compatible with S.*

This suggests that constraints on the density of the sample might be used to guarantee computational tractibility of the problem. However, in Section 3

* These results appear in the author's Ph.D. thesis, submitted to the Electrical Engineering and Computer Science Department, University of California, Berkeley, March 1976. The research was supported by the National Science Foundation, Grant GJ35604X1.

0019-9958/78/0393-0337$02.00/0

643/39/3-8

below we demonstrate that the problem remains *NP*-hard even under rather strong constraints on the density of the sample. The construction given also shows that various restrictions on the type of the realizing automaton are similarly ineffective.

In Section 4 we consider another possible system of representation of regular sets, namely regular expressions. We prove a theorem analogous to Theorem 1 for this system of representation and various restrictions of it. The techniques required for dealing with smallest regular expressions are different from those concerning minimum state finite automata, and are possibly of independent interest.

In Levin (1972), Pfleeger (1973), Pudlak and Springsteel (1977) may be found related results on the complexity of finding minimum realizations of incompletely-specified Boolean functions, minimizing incompletely-specified deterministic automata, and finding hypotheses in specified forms in agreement with given observations, respectively.

2. Definitions

$U = \{0, 1\}$ is the alphabet throughout. If m and n are nonnegative integers with $m \leqslant n$ we use $U_m{}^n$ to denote strings of length at least m and at most n over U. The null string is denoted Λ.

If u and v are strings, then $| u |$ denotes the length of u, $u \cdot v$ and uv denote the concatenation of u and v, rev(u) denotes the reverse of u, and $u(i)$ denotes the ith letter of u.

If A and B are sets of strings then $A \cdot B$ denotes the set of all strings uv such that $u \in A$ and $v \in B$. A^n is defined inductively: $A^0 = \{\Lambda\}$ and $A^{i+1} = A^i \cdot A$ for all nonnegative integers i. A^* denotes the union of all A^n as n ranges over all nonnegative integers; A^+ is A^* minus the null string.

If S is any finite set, $| S |$ denotes the cardinality of S. $\log x$ means the base two logarithm of x. $\lceil x \rceil$ denotes the least integer not less than x.

A *sample* S is a finite subset of $U^+ \times U$ such that whenever $\langle u, a \rangle$ and $\langle v, b \rangle$ are members of S and $u = v$ then $a = b$. The *domain* of S, denoted domain(S), is the set of all strings u such that for some $a \in U$, $\langle u, a \rangle \in S$. (We assume that a sample is presented as input via a string which lists every pair in the sample, so that the length of the input is proportional to the sum of the lengths of the strings in the domain of the sample.)

A *partially-specified machine* M is a quadruple $\langle Q, p, \delta, \lambda \rangle$ such that Q is a finite set, the set of *states* of M, $p \in Q$ is the *initial state* of M, δ maps a subset of $Q \times U$ into Q, and λ maps a subset of $Q \times U$ into U. We implicitly consider δ and λ as extended in the usual way to maps from a subset of $Q \times U^*$ into Q and a subset of $Q \times U^+$ into U, respectively. We define

$$\bar{\lambda}(q, a_1 a_2 \cdots a_n) = \lambda(q, a_1) \cdot \lambda(q, a_1 a_2) \cdots \lambda(q, a_1 a_2 \cdots a_n),$$

whenever the right hand side is defined. Thus $\bar{\lambda}$ maps a subset of $Q \times U^+$ into U^+. We abbreviate $\delta(p, u)$, $\lambda(p, u)$, $\bar{\lambda}(p, u)$ by $\delta(u)$, $\lambda(u)$, $\bar{\lambda}(u)$.

A *fully-specified machine* (or simply, a *machine*) is a partially-specified machine $\langle Q, p, \delta, \lambda \rangle$ such that δ and λ are defined on all of $Q \times U$.

A partially-specified machine $\langle Q, p, \delta, \lambda \rangle$ will be said to *agree with* (or be *compatible with*) a sample S iff for every $\langle u, a \rangle \in S$, $\lambda(u)$ is defined, and $\lambda(u) = a$.

Regular expressions and the sets they denote are defined inductively as follows: 0 and 1 are regular expressions, denoting the sets $\{0\}$ and $\{1\}$; if E and F are regular expressions denoting the sets S and T then $(E \cdot F)$, $(E \vee F)$, and $(E)^*$ are regular expressions denoting the sets $S \cdot T$, $S \cup T$, and S^*.

The set denoted by the regular expressions E will be denoted by $L(E)$. Two regular expressions E and F are *equivalent* iff $L(E) = L(F)$.

We will freely omit unnecessary parentheses and the concatenation symbol when informally designating regular expressions.

A regular expression E *agrees with* (or is *compatible with*) a sample S iff for each $\langle u, a \rangle \in S$, $u \in L(E)$ iff $a = 1$.

We use the definitions of deterministic and nondeterministic polynomial time computability and reducibility, of the classes P and NP, and of NP-completeness as found in Cook (1971) and Karp (1972). A set S is *NP-hard* iff every set in NP is polynomial time reducible to S.

3. On the Effect of Sample Density

We define the *size* of a partially-specified machine to be the cardinality of the set of states of the machine.

We define a sample S to be *uniform-complete* iff the domain of S consists of all strings not exceeding a given length and no others. In other words, there exists an integer k such that domain$(S) = U_1^k$.

We need also a quantification of the notion of a "nearly" uniform-complete sample. Thus, given a real-valued function $g(x)$ we say that a sample S is $g(x)$-*incomplete* iff the domain of S is $U_1^k - A$ for some positive integer k and some set A of cardinality less than $g(2^{k+1})$. For example, a sample whose domain consists of all strings of length not exceeding $2k + 1$ which do not have 1^k as a prefix is $x^{1/2}$-incomplete for any positive integer k.

We have the following easy corollary of Theorem 2:

Corollary 1. *For any positive number d there is a polynomial time algorithm which correctly decides for any $(d \log x)$-incomplete sample S and positive integer t whether there is a machine of size at most t which agrees with S.*

Proof. Suppose S is a $(d \log x)$-incomplete sample. If k is the length of the longest string in the domain of S and $n = 2^{k+1}$ then the domain of S is U_1^k

minus at most $d \log n$ strings. For each of the at most n^d possible ways of specifying outputs for the missing strings, we may apply the algorithm of Theorem 2 and take the smallest of the results. ▮

The remainder of this section is devoted to proving the following:

THEOREM 3. *For any positive number ϵ it is an NP-complete problem to decide, for an arbitrary x^ϵ-incomplete sample S and positive integer t, whether there is a machine of size at most t which agrees with S.*

The reason that Gold's construction for the proof of Theorem 1 does not suffice to prove Theorem 3 is that a propositional formula f of m clauses and n variables is transformed to a sample containing strings of length at least $r = \max(m, n)$. For such a sample to be x^ϵ-incomplete, it must contain $c2^r$ strings for some positive constant c, which would not be polynomial in the size of f as required for the *NP*-reduction. Thus, the primary purpose of the new construction is to keep the sample strings to length $O(\log r)$. We give a construction which achieves this, and then briefly indicate how the ϵ may be achieved.

Proof of Theorem 3. Let ϵ be a fixed positive number.

To see that the indicated problem is in *NP*, we note that if S and t are given and t exceeds the length of the string presenting S then there must necessarily be a machine of size t which agrees with S. Otherwise, we may nondeterministically guess a machine of size t and check that it agrees with S.

The proof that the problem is *NP*-hard is a polynomial time reduction of a known *NP*-complete problem to it. First we assume that k is a fixed positive integer (the value will be specified later) and define a particular incompletely-specified machine M to be used in the proof.

We set

$$L = 4k + 3,$$

$$A = \{u \in U^* : |u| \leqslant 2k + 1 \text{ and if } |u| > k \text{ then } u(k+1) = 1\},$$

$$h(u) = u \cdot 1^s, \text{ where } s = L - 2|u|.$$

(Note that h is injective, with range disjoint from A.)

$$B = \{h(u) : u \in A\},$$

$$Q = A \cup B.$$

For each $w \in Q$ and $a \in U$,

(i) if $|w| < k$ or $k < |w| < 2k + 1$ then

$$\delta(w, a) = w \cdot a \quad \text{and} \quad \lambda(w, a) = 1;$$

(ii) if $|w| = k$ or $|w| = 2k + 1$ then

$$\delta(w, 1) = w \cdot 1 \quad \text{and} \quad \lambda(w, 1) = 1;$$

(iii) if $2k+1 < |w| < L$ then $h^{-1}(w) = v \cdot b$ for some $b \in U$, and we let

$$\delta(w, a) = h(v) \quad \text{and} \quad \lambda(w, 0) = b \quad \text{and} \quad \lambda(w, 1) = 1;$$

(iv) if $|w| = L$ then

$$\delta(w, a) = w \quad \text{and} \quad \lambda(w, a) = 0.$$

And, finally,

$$M = \langle Q, \Lambda, \delta, \lambda \rangle.$$

Figure 1 gives another view of the definition of M, indicating the inductive definition of the partially-specified machine T_{n+1} (R_{n+1}) from two copies of T_n (R_n), and the construction of M from $2^k + 1$ copies of each of T_k and R_k. If w is a state in the left half of M, then $h(w)$ is its mirror image in the right half.

If we define the *level* of a state w of M to be $|w|$, we note that the only unspecified values of δ and λ are the 0-transitions and 0-outputs from states at levels k and $2k+1$. Define

$$\begin{aligned} g(r, s) &= 0 \quad \text{if} \quad r + s > L \\ &= 1 \quad \text{if} \quad r + s \leqslant L \end{aligned}$$

The following facts may be verified of M:

(a) $|Q| = 2^{2k+2} - 2^{k+1} - 2$;
(b) $\delta(w) = w$ for all $w \in Q$;
(c) $\lambda(w \cdot 1^s) = g(|w|, s)$ for all $w \in Q$;
(d) $\bar{\lambda}(h(w) \cdot 0^s) = 1^t \cdot \mathrm{rev}(w)$ for all $w \in A$, where $s = |w|$ and $t = L - s$;
(e) $\bar{\lambda}(w \cdot 0^s) = 1^t \cdot \mathrm{rev}(h^{-1}(w))$ for all $w \in B$, where $t = |w|$ and $s = L - t$.

Item (c) may be used to distinguish states at different levels of M, items (d) and (e) to distinguish two states at the same level. We accordingly define a sample,

$S = \{\langle u, \lambda(u) \rangle$: either $u = w \cdot 1^s$ for $w \in Q$ and $s \leqslant L + 1 - |w|$ or $u = w \cdot 1^s \cdot 0^t$ for $w \in A$, $s = L - 2|w|$ and $t \leqslant |w|\}$.

LEMMA 1. *If $M' = \langle Q', p', \delta', \lambda' \rangle$ is any machine which agrees with this sample S then $\delta'(v) \neq \delta'(w)$ for all v and w in Q with $v \neq w$. Consequently $|Q'| \geqslant |Q|$.*

Proof. Fix v and w from Q with $v \neq w$. There are three possible cases:

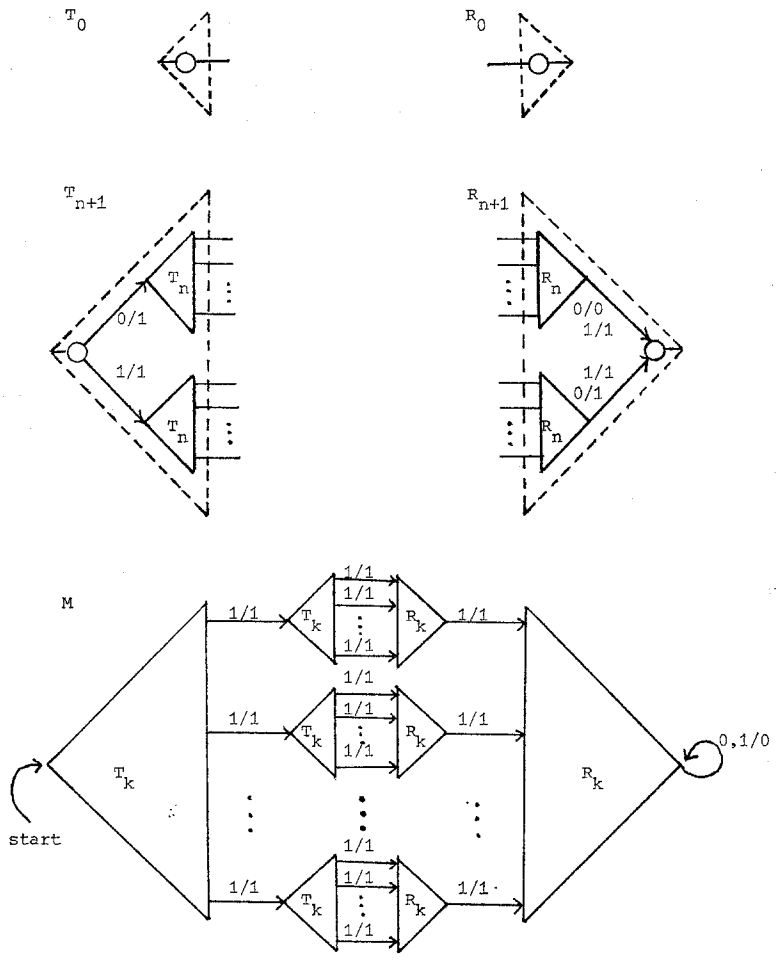

FIG. 1. Construction of M.

(i) $|v| \neq |w|$. Without loss of generality, assume that $|v| > |w|$. Let $s = L + 1 - |v|$. Since M' agrees with S,

$$\lambda'(v \cdot 1^s) = \lambda(v \cdot 1^s) = g(|v|, s) = 0,$$
$$\lambda'(w \cdot 1^s) = g(|w|, s) = 1,$$

so $\delta'(v) \neq \delta'(w)$.

(ii) $|v| = |w|$ and $|v| \leqslant 2k + 1$. Let $t = |v|$ and $s = L - 2t$. Then

$$\bar{\lambda}'(v \cdot 1^s \cdot 0^t) = 1^{s+t} \cdot \mathrm{rev}(v),$$
$$\bar{\lambda}'(w \cdot 1^s \cdot 0^t) = 1^{s+t} \cdot \mathrm{rev}(w).$$

(iii) $|v| = |w|$ and $|v| > 2k+1$. Let $v_1 = h^{-1}(v)$, $w_1 = h^{-1}(w)$, $t = |v|$, $s = L - t$. Then $v_1 \neq w_1$ and

$$\bar{\lambda}'(v \cdot 0^s) = 1^t \cdot \mathrm{rev}(v_1),$$
$$\bar{\lambda}'(w \cdot 0^s) = 1^t \cdot \mathrm{rev}(w_1). \quad \blacksquare$$

Now we give the reduction. Let $SF = \{f: f$ is a propositional formula in conjunctive normal form with each clause containing either only positive or only negative literals$\}$. Let $SSAT = \{f \in SF: f$ is satisfiable$\}$. Then $SSAT$ is an NP-complete problem, see Gold(1974).

Let $f \in SF$ be given. Suppose f has m clauses and n variables. Fix $k = \lceil \log(m+n) \rceil$. Consider the machine $M = \langle Q, \Lambda, \delta, \lambda \rangle$ defined above for this value of k. Choose two disjoint sets C and V contained in U^k with $|C| = m$ and $|V| = n$, to represent the clauses and variables of f, respectively. Define

$$\begin{aligned} \mathrm{in}(v, c) &= 1 \quad \text{if } v \in V,\ c \in C, \text{ and variable } v \text{ appears in clause } c, \\ &= 0 \quad \text{otherwise,} \\ \mathrm{sense}(c) &= 1 \quad \text{if } c \in C \text{ and clause } c \text{ contains only positive literals,} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

$$T_1 = \{\langle v1w0, \mathrm{in}(v, w)\rangle: v, w \in U^k\},$$
$$T_2 = \{\langle c01^s, g(s, k)\rangle: c \in C,\ s \leqslant L + 1 - k\},$$
$$T_3 = \{\langle c01c0, 1\rangle: c \in C\},$$
$$T_4 = \{\langle c00, \mathrm{sense}(c)\rangle: c \in C\}.$$

And finally,

$$S_f = S \cup T_1 \cup T_2 \cup T_3 \cup T_4,$$

where S is the sample defined above for the value of k chosen.

Since the domains of the components of the union defining S_f are pairwise disjoint, S_f is a sample.

LEMMA 2. *There is a machine of size at most $|Q|$ agreeing with S_f iff $f \in SSAT$.*

Proof. Suppose that $M' = \langle Q', p', \delta', \lambda' \rangle$ is a machine of size at most $|Q|$ which agrees with S_f. Define $\tau(v) = \lambda'(v0)$ for all $v \in V$. We shall show that τ is an assignment which satisfies f. Note that M' agrees with S, so from Lemma 1 we conclude that Q' consists precisely of those elements $\delta'(w)$ such that $w \in Q$ and that these are all distinct. Let $c \in C$ be any clause. Let v_c be the unique element of Q such that $\delta'(v_c) = \delta'(c0)$. Since M' agrees with T_2, $\lambda'(v_c 1^s) = \lambda'(c01^s) = g(s, k)$ for all positive integers s not exceeding $L + 1 - k$. This

shows that v_c must be of length (and level) k. By agreement with T_3 we have $\lambda'(v_c 1 c0) = \lambda'(c01c0) = 1$, so by agreement with T_1, $\text{in}(v_c, c) = 1$, so $v_c \in V$ and the variable v_c appears in the clause c. Finally, since M' agrees with T_4, $\tau(v_c) = \lambda'(v_c 0) = \lambda'(c00) = \text{sense}(c)$, so the value assigned by τ to v_c satisfies c. Since c was an arbitrary clause of f, τ satisfies f.

Conversely, suppose f is satisfiable and let τ be an assignment of 0 and 1 to elements of V which satisfies f. For each $c \in C$ let v_c be such that $\text{in}(v_c, c) = 1$ and $\text{sense}(c) = \tau(v_c)$. We define a machine $M' = \langle Q', \Lambda, \delta', \lambda' \rangle$ as follows:

$$Q' = Q.$$

For all $w \in Q'$ and $a \in U$,

$$\begin{aligned}
\delta'(w, a) &= \delta(w, a) \text{ if this is defined,} \\
&= v_w \text{ if } w \in C \text{ and } a = 0, \\
&= w1 \text{ otherwise,}
\end{aligned}$$

$$\begin{aligned}
\lambda'(w, a) &= \lambda(w, a) \text{ if this is defined,} \\
&= \text{in}(u, v) \text{ if } w = u1v, \text{ where } u, v \in U^k \text{ and } a = 0, \\
&= \tau(w) \text{ if } w \in V \text{ and } a = 0, \\
&= 1 \text{ otherwise.}
\end{aligned}$$

Then M' is a fully-specified machine with $|Q|$ states. To see that it agrees with S_f,

(i) M' extends M and consequently agrees with S;

(ii) M' agrees with T_1 by explicit construction;

(iii) if $c \in C$ and $1 \leqslant s \leqslant L + 1 - k$ then

$$\begin{aligned}
\lambda'(c0) &= 1 \text{ by the default case for } \lambda', \\
\lambda'(c01^s) &= \lambda'(v_c 1^s) = \lambda(v_c 1^s) = g(s, k),
\end{aligned}$$

so M' agrees with T_2;

(iv) for all $c \in C$,

$$\lambda'(c01c0) = \lambda'(v_c 1c0) = \text{in}(v_c, c) = 1,$$

so M' agrees with T_3;

(v) for all $c \in C$,

$$\lambda'(c00) = \lambda'(v_c 0) = \tau(v_c) = \text{sense}(c),$$

so M' agrees with T_4. ∎

To complete the proof of Theorem 3 we must show how to achieve x^ϵ-incompleteness of the sample. (Note that the strings of S_f are of length $O(\log(m+n))$.) We therefore pad S_f as follows.

Let $r = \lceil (4k+5)/\epsilon \rceil$ and add a "preamble" of r states to the machine M to obtain the machine $M' = \langle Q', p', \delta', \lambda' \rangle$ as indicated in Fig. 2. Each of the additional r states may be distinguished from the original states of M by its output under either input 0 or 1, and from the other states of the preamble by its outputs under the input string 1^{r+1}. We define $s = 4k+4$, $t = r+s$,

$$V_1 = \{\langle u, \lambda'(u)\rangle : u \neq 1^r v \text{ for all strings } v, \text{ and } u \in U_1^t\},$$

$$V_2 = \{\langle 1^r u, b\rangle : \langle u, b\rangle \in S_f\},$$

$$S_f' = V_1 \cup V_2 .$$

Note that the domain of S_f' is U_1^t less at most 2^s strings, and since $2^s \leqslant (2^t)^\epsilon$ the sample S_f' is x^ϵ-incomplete. It is then straightforward but tedious to verify that Lemmas 1 and 2 may be strengthened to give

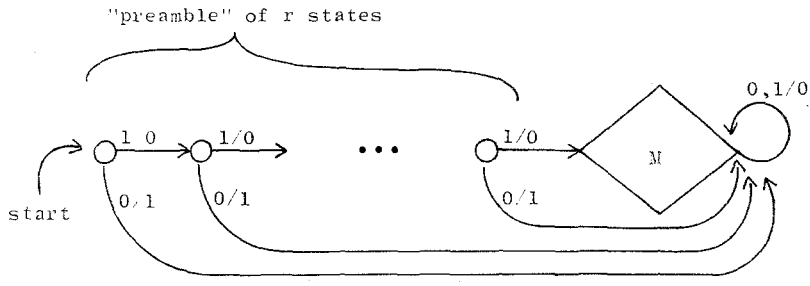

FIG. 2. The machine M'.

LEMMA 3. *There is a machine of size at most* $|Q'|$ *which agrees with* S_f' *iff* $f \in SSAT$.

It is clear that the indicated reduction may be carried out in polynomial time in the length of f, which concludes the proof of Theorem 3. ∎

We note that the machine M' constructed in the second half of the proof of Lemma 2 is "finite-language" (i.e., accepts a finite set of strings), so we have

COROLLARY 2. *If C is any class of machines which contains all the finite-language machines then it is an NP-hard problem to decide for a sample S and positive integer t whether there is a machine from class C which is compatible with S and of size at most t.*

4. Inferring Regular Expressions

The notion of size we shall use for regular expressions has been chosen primarily to simplify the proofs in this section. The *size* of a regular expression will be the number of occurrences of the symbols 0 and 1 in it. (The results that follow can be shown to hold for other definitions of size.) Define $R = \{\langle S, t\rangle$: S is a sample, and there is a regular expression of size at most t which agrees with $S\}$.

THEOREM 4. *R is an NP-complete problem.*

Proof. To see that R is in NP, we note first that for any expression of size m there is an equivalent expression of length (as a string) at most $10m$, using the fact that $(E^*)^*$ is equivalent to E^*. If t exceeds the sum of the lengths of the strings in the domain of S then there will necessarily be an expression of size at most t which agrees with S (namely, the disjunction of all the strings in the positive part of S). Otherwise, we nondeterministically guess a regular expression of length at most $10t$ and check that it agrees with S. (That the agreement may be checked in polynomial time is proved, for example, in Aho, Hopcroft, and Ullman (1974).)

The proof that R is NP-hard is a polynomial time reduction to it of the following problem: $SAT = \{f: f$ is a propositional formula in conjunctive normal form which is satisfiable$\}$. SAT is NP-complete, see Karp (1972). Let f be a propositional formula in conjunctive normal form with clauses numbered 1 to m and variables numbered 1 to n. Define

$$\begin{aligned}
\text{cont}(i, j) &= 0 \quad \text{if variable } j \text{ does not appear in clause } i,\\
&= 1 \quad \text{if variable } j \text{ appears positively in clause } i,\\
&= -1 \quad \text{if variable } j \text{ appears negatively in clause } i,
\end{aligned}$$

$$q = (1100)^n,$$

$$\begin{aligned}
F_{ij} &= 1100 \quad \text{if } \text{cont}(i, j) = 0,\\
&= 110 \quad \text{if } \text{cont}(i, j) = 1,\\
&= 100 \quad \text{if } \text{cont}(i, j) = -1,
\end{aligned}$$

$$S_1 = \{\langle q, 1\rangle\},$$

$$S_2 = \{\langle wxxy, 0\rangle: q = wxy \text{ and } x \text{ contains both 0 and 1}\},$$

$$S_3 = \{\langle (1100)^r\, 10(1100)^s, 0\rangle: r + s = n - 1\},$$

$$S_4 = \{\langle F_{i1}F_{i2} \cdots F_{in}, 0\rangle: 1 \leqslant i \leqslant m\},$$

$$S_f = S_1 \cup S_2 \cup S_3 \cup S_4.$$

LEMMA 4. *$f \in SAT$ iff $\langle S_f, 3n\rangle \in R$.*

Proof. Suppose $f \in SAT$. Let τ be an assignment of 0 and 1 to the variables of f which satisfies f. For each $j = 1, 2, \ldots, n$ let

$$\begin{aligned} E_j &= (1)^*00 \quad \text{if} \quad \tau(j) = 1, \\ &= 11(0)^* \quad \text{if} \quad \tau(j) = 0. \end{aligned}$$

Let $E = E_1E_2 \cdots E_n$. Then E is a regular expression of size $3n$. To see that E agrees with S_f note first that $L(E)$ is contained in $L((1^*0^*)^n)$ and if $w_1w_2 \cdots w_n \in L(E)$, where each $w_j \in L(1(1)^* 0(0)^*)$, then each $w_j \in L(E_j)$. Then

(i) $q \in L(E)$, so E agrees with S_1;

(ii) if $q = wxy$ and x contains both 0 and 1 then $wxxy$ is not in $L((1^*0^*)^n)$ and so is not in $L(E)$, so E agrees with S_2;

(iii) if $r + s = n - 1$ and $(1100)^r\, 10(1100)^s \in L(E)$ then $10 \in L(E_j)$ for some j, which is a contradiction, so E agrees with S_3;

(iv) if E does not agree with S_4 then for some i between 1 and m, $F_{i1}F_{i2} \cdots F_{in} \in L(E)$. Hence $F_{ij} \in L(E_j)$ for $j = 1, 2, \ldots, n$. Let k be any variable appearing in clause i. If k appears positively in i then $F_{ik} = 110$ so E_k must be $11(0)^*$ and $\tau(k) = 0$. If k appears negatively in clause i then similarly $\tau(k) = 1$. In either case, we find that τ does not satisfy clause i, contradicting our choice of τ. Hence E must agree with S_4.

Conversely suppose that there exists a regular expression of size at most $3n$ which agrees with S_f. We shall show that a minimum such expression must have essentially the form of E and derive from it an assignment which satisfies f. Let E be a regular expression of minimum possible size compatible with S_f. By hypothesis the size of E is at most $3n$. We use the associativity of concatenation to rewrite E as an equivalent expression of the same size: $F = F_1F_2 \cdots F_k$, where each F_i is not itself a concatenation. Since $q \in L(E)$ we may choose $q_1, q_2, \ldots, q_k$ such that $q = q_1q_2 \cdots q_k$ and each $q_i \in L(F_i)$. For each i, F_i cannot be of the form $(G \vee H)$. For suppose to the contrary that $F_i = (G \vee H)$. Since $q_i \in L(F_i)$ we have $q_i \in L(G)$ or $q_i \in L(H)$. If $q_i \in L(G)$ then by replacing F_i by G in F we get an expression of strictly smaller size which is still compatible with S_f, contradicting our choice of E. Similarly for the case of $q_i \in L(H)$.

Thus the only possibilities for F_i are 0 or 1 or $(G)^*$ for some regular expression G. In this last case, $q_1q_2 \cdots (q_i)^2 \cdots q_k$ is also in $L(E)$, so by agreement with S_2, q_i cannot contain both 0 and 1. Hence we may again reassociate the concatenations in E to obtain an expression $G_1H_1G_2H_2 \cdots G_nH_n$, where for each $j = 1, 2, \ldots, n$ we have $11 \in L(G_j)$ and $00 \in L(H_j)$.

Now the size of G_j is at most two for all j, for otherwise we could replace G_j by 11 and obtain an expression of strictly smaller size compatible with S_f. Similarly, the size of H_j is at most two. It may be verified that the only expressions

of size one which generate the string 11 are of the form $(J)^*$, where $1 \in L(J)$, and so must also generate the string 1, and similarly for expressions of size one which generate 00. Thus for each j we cannot have both G_j and H_j of size one, for otherwise $10 \in L(G_jH_j)$, contradicting agreement with S_3. To attain size at most (and exactly) $3n$ for E we must have for each j either the size of G_j is one or the size of H_j is one, but not both. Hence we define

$$\begin{aligned} \tau(j) &= 1 \quad \text{if the size of } G_j \text{ is one,} \\ &= 0 \quad \text{otherwise.} \end{aligned}$$

To see that τ satisfies f we suppose to the contrary that it falsifies clause i. Then for each variable j,

(i) if j does not appear in clause i then $F_{ij} = 1100 \in L(G_jH_j)$;

(ii) if j appears positively in clause i then $\tau(j) = 0$ and

$$F_{ij} = 110 \in L(G_jH_j);$$

(iii) if j appears negatively in clause i then $\tau(j) = 1$ and

$$F_{ij} = 100 \in L(G_jH_j).$$

Thus $F_{i1}F_{i2} \cdots F_{in} \in L(E)$, contradicting agreement with S_4. Hence τ must satisfy f and $f \in SAT$. ∎

The indicated construction of S_f from f may be carried out in polynomial time in the length of f, so we conclude that the problem R is NP-hard. ∎

We note that the expression constructed in the first half of the proof of Lemma 4 is of a special form in that it contains no occurrences of the symbol $\vee$. Thus we have

COROLLARY 3. *If C is any set of regular expressions containing all those expressions in which $\vee$ does not appear, then the problem of deciding for a sample S and positive integer t whether there exists an expression from C which is compatible with S and of size at most t is NP-hard.*

A separate construction is given to prove the analog of Corollary 3 for *-free regular expressions in Angluin (1976).

5. REMARKS AND CONCLUSIONS

In particular cases it might be more economical to represent a uniform-complete sample S as the list of strings u such that $\langle u, 1\rangle \in S$. The algorithm of Trakhtenbrot and Barzdin of Theorem 2 can be adapted to run in time

polynomial in the length of this form of presentation of the input, which may be of some practical interest. The question of whether Theorem 4 holds when regular expressions are allowed to contain the negation operator is open. In general, the identity of the regular set inferred for a given sample depends on the system of representation and definition of size chosen. Angluin (1976) gives an example of this phenomenon for deterministic versus nondeterministic automata.

It is hoped that these largely negative results will be of use in guiding the search for appropriate formulations of problems in concrete inductive inference, and in the evaluation of proposed algorithmic solutions.

Acknowledgments

The author would like to thank Mark Gold, Manuel Blum, and the referee for their help with this work.

Received: August 26, 1977; revised: April 21, 1978

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. (1976), "An Application of the Theory of Computational Complexity to the Study of Inductive Inference," Ph.D. Thesis, Electrical Engineering and Computer Science Department, University of California, Berkeley.

Biermann, A. W. (1974), "Constructing Programs from Example Computations," Report Number TR-74-5, Ohio State University.

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

Cook, S. A. (1971), The complexity of theorem-proving procedures, *in* "Proceedings of the Third Annual ACM Symposium on Theory of Computing," 151–158.

Feldman, J. A. (1972), Some decidability results on grammatical inference and complexity, *Inform. Contr.* **20**, 244–263.

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

Gold, E. M. (1974), Complexity of automaton identification from given data, *Inform. Contr.* **37** (1978), 302–320.

Horning, J. J. (1969), "A Study of Grammatical Inference," Ph.D. Thesis, Computer Science Department, Stanford University.

Karp, R. M. (1972), Reducibilities among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–104, Plenum, New York.

Levin, L. A. (1972), Universal combinatorial problems, *Problemy Peredači Informacii* **9**, 115–116.

Pfleeger, C. P. (1973), State reduction in incompletely specified finite-state machines, *IEEE Trans. Computers* **C-22**, 1099–1102.

PUDLAK, P., AND SPRINGSTEEL, F. M. (1977), Complexity of mechanized hypothesis formation, Preprint, Czechoslovak Academy of Sciences.

TRAKHTENBROT, B. A., AND BARZDIN, YA. M. (1973), "Finite Automata," pp. 98–99, North–Holland, Amsterdam.

WHARTON, R. M. (1977), Grammar enumeration and inference, *Inform. Contr.* **33**, 253–272.