INFORMATION AND CONTROL **37,** 302–320 (1978)

Complexity of Automaton Identification from Given Data*

E MARK GOLD†

Département d'Informatique, Université de Montréal, Montréal, Canada

The question of whether there is an automaton with n states which agrees with a finite set D of data is shown to be NP-complete, although identification-in-the-limit of finite automata is possible in polynomial time as a function of the size of D. Necessary and sufficient conditions are given for D to be realizable by an automaton whose states are reachable from the initial state by a given set T of input strings. Although this question is also NP-complete, these conditions suggest heuristic approaches. Even if a solution to this problem were available, it is shown that finding a minimal set T does not necessarily give the smallest possible T.

1. INTRODUCTION

NP-Complete

A predicate $P(x)$ which is NP-complete is difficult to compute: $P(x)$ can be compute in exponential time as a function of the size of the input x, but it is believed that polynomial time computation is not possible; see Karp (1975) and Aho *et al.* (1974).

* This work received support from the National Research Council of Canada.

† Present address: Computer Science Dept., University of Rochester, Rochester, N.Y. 14627.

0019-9958/78/0373–0302$02.00/0

Minimum Automaton Identification from Given Data is NP-Complete

Suppose that we are given data D consisting of a finite number of observations of the I/O behavior of an unknown black box with I/O function b. The principal objective of this paper is to show that the problem of finding a minimum state finite automaton which agrees with D is "NP-complete" in the following sense: Theorem 2 (transition assignment is NP-complete) implies that the question "Is there an automaton with n states which agrees with D?" is NP-complete.

Nerode Algorithm

The complexity results of this paper resulted from efforts to adapt the Nerode algorithm to the problem of automaton identification from given data. The Nerode (1958) algorithm for the problem of *automaton synthesis* yields the minimum finite state automaton which realizes a given black box function b. Automaton synthesis assumes that the entire function b is given. In particular, the Nerode algorithm assumes that means are available for determining if $b_{\bar{u}} = b_{\bar{v}}$ for any pair $\bar{u}$, $\bar{v}$ of input strings, where $b_{\bar{u}}$ is the black box which results if $\bar{u}$ is applied to b. The Nerode algorithm can be used with a finite amount of data if the number of states needed to realize the unknown black box is specified, and if the experiments which produce the data can be chosen.

The Ho algorithm adapts the Nerode algortihm to the synthesis of linear automata, see Zeiger (1967). Application of the Nerode approach to more general classes of automata is straightforward, e.g., see Arbib and Zeiger (1969). Arbib and Manes (1974) discuss further generalization to abstract machines in a category-theoretic framework.

Automaton Identification from Finite Data

The work referenced above is concerned with generalization of the Nerode algorithm to larger classes of machines. The work which led to this paper was concerned with the adaption of the Nerode algorithm to the problem of *automaton identification from finite data*: One wishes to identify a black box which is known to be realizable by a finite (state) automaton, but the necessary number of states is not known. Only a finite amount of data is available, so it is not possible to prove $b_{\bar{u}} = b_{\bar{v}}$.

There are 2 variations of this problem: automaton identification from requested data, and automaton identification from given data. In the case of *requested data*, any finite number of experiments, chosen at will, can be performed on the black box which can be reset to its initial state before each experiment. The identification algorithm must choose the experiments as well as use the results of these experiments to guess a finite automaton which, hopefully, realizes the I/O function b of the unknown black box. In the case of automaton identification

from *given data*, the identification algorithm has no choice about the data, it is given.

In Gold (1972) I discuss a straightforward adaption of the Nerode algorithm, which I call *state characterization*, to the problem of automaton identification from requested data. The results of that paper were not original, but I reference it because it introduces the notation and terminology used in this peper.

Concerning automaton identification from given data, one approach is discussed by Bierman and Feldman (1972). An obvious approach is ***minimum automaton identification***: Construct a finite automaton with the minimum number of states which agrees with the given data D. This approach has many desirable properties discussed in the next section, such as efficient use of data. However, it is the objective of this paper to show that the construction of a minimum state automaton which agrees with given data is, in general, computationally difficult.

2. Summary of Complexity Results

Attributes of Automaton Identification Rules

An *automaton identification rule* is a computable function g which, given data D about black box b, produces a finite automaton $g(D)$ (g for "guess"). In the case of requested data, an automaton identification rule also generates the experiments which produce a growing sequence of data D_1, D_2,... . The rule will be said to have the *identification-in-the-limit* property if it can be guaranteed that for every black box b realizable by a finite automaton there is an i such that $g(D_i)$, $g(D_{i+1})$,... are the same and realize b. In the case of given data, g will be said to have the *identification-in-the-limit* property if for every such b there is a data set D_b such that for all data sets D which include D_b the guesses $g(D)$ are the same and realize b.

An automaton identification rule is *feasible* if $g(D)$ always agrees with D (defined formally in next section). The stronger ***minimum automaton identification*** property requires that $g(D)$ have the minimum number of states. Since D is finite, there can be nonequivalent $g(D)$ with this property.

Suppose g has the identification-in-the-limit property. Its space, time, and data requirements are of interest. Space complexity is not discussed here. Time complexity refers to the time required to compute $g(D)$ as a function of the size of D if the fastest algorithm is used. Presumably, polynomial time is practical and *NP*-complete is impractical.

Concerning data requirements, in a somewhat different context (Gold, 1967) I introduced the following notion: g is *optimally data efficient* if there is no g' which, for all b, correctly identifies b from as small a data set as g and sometimes smaller. I will not try to formalize this notion in the present context because I only use it to motivate the interest in minimum automaton identification:

Suppose our guessing rule tries all finite automata in order of increasing number of states, and chooses $g(D)$ to be the first finite automaton which agrees with D. This is a minimum automaton identification rule. Also, it is an example of identification-by-enumeration. In Gold (1967) I showed that all identification-by-enumeration rules are optimally data efficient.

State Characterization from Requested Data

In the case of requested data I showed in Gold (1972) that state characterization has the following properties:

1. Minimum automaton identification.
2. Identification-in-the-limit (implied by 1).
3. Computationally trivial.

State Characterization from Given Data

The complexity results of this paper are an outgrowth of attempts to adapt state characterization to the problem of automaton identification from given data. The proposed method of adaption is straightforward: In the case of requested data, one can request the data needed by the state characterization algorithm. In the case of given data the data which is needed by the algorithm and not provided is guessed. The problem is to guess the missing data in such a way that the constructed finite automaton will be small.

Theorem 1 (Data Matrix Agreement) is the fundamental theorem of the state characterization approach to automaton identification from finite data. It gives sufficient constraints on the use of the state characterization approach to guarantee that the constructed finite automaton will agree with the data from which it was constructed.

This theorem is of interest in itself. Indeed, it is necessary to show the validity of the earlier results on the application of state characterization to the requested data problem (Gold 1972).

Furthermore, Theorem 1 assures the validity of the *timid state characterization* algorithm for given data, in the proof of Theorem 4, which has the following properties:

1. Feasible automaton identification.
2. Identification-in-the-limit (not implied by 1).
3. Polynomial time computation.

However, the main reason for including the Data Matrix Agreement Theorem in this paper is that it serves as a lemma in the proof of the principal result, Theorem 2, which says that minimum automaton identification from given data is *NP*-complete.

Data–Time Tradeoff

Timid state characterization is very inefficient in its use of data. So, concerning the given data problem, the results of this paper suggest that a data–time tradeoff is necessary: Identification-in-the-limit can be achieved in polynomial time at the cost of additional data being required to correctly identify the unknown black box. The most obvious approach to obtaining optimal data efficiency is computationally impractical.

However, there is still the possibility that optimal data efficiency can be achieved in polynomial time.

The *timid state characterization* approach to given data is as follows: The given data D is searched for a subset D_0 such that the state characterization algorithm can be applied to D_0 without having to guess missing data. If the resulting finite automaton agrees with all of D then it is taken to be the guess $g(D)$. Otherwise, a feasible finite automaton $g_{\text{tabl}}(D)$ for D is constructed in the easiest way. $g_{\text{tabl}}(D)$ is easy to compute but doesn't have the identification-in-the-limit property.

The timid state characterization algorithm uses data efficiently in the following sense: If we are lucky, and we are given just the right type of data D, then timid state characterization will obtain a correct, minimum state realization for the unknown black box from a quantity of data which is a polynomial function of the required number of states. The timid state characterization algorithm uses data inefficiently if we are not lucky and are given data which is not directly usable by the state characterization algorithm. The timid state characterization algorithm essentially ignores data which it cannot use easily.

It is straightforward to adapt state characterization to minimum automaton identification from given data if we are not interested in computation time: A backtracking algorithm can be used to guess the missing data. Varying degrees of "timidity" can be introduced to give varying data–-time tradeoffs. Namely, the backtracking can be truncated at some prior time limit.

In summary, state characterization requires a certain type of data (the results of certain experiments) and so is well suited to the requested data problem: The required data is requested and a trivial computation is capable of correctly identifying the unknown black box from a small amount of data. State characterization can be adapted to the given data problem by the use of 2 types of timidity in order to reduce computation time: A tractable subset of the given data is selected and backtracking is truncated.

By means of an appropriate combination of these 2 types of timidity I believe that reasonably data efficient identification from given data should be possible with computation time asymptotically linear. Namely, let the unknown black box be fixed and suppose that we are given an enormous body of data which resulted from a not very bizarre set of experiments. It should be possible to select a small subset of the data, correctly identify the black box without much backtracking, then make one pass through the entire body of data to check the finite automaton which was constructed from the subset.

3. Terminology and Notation

Strings: Prefix- and Suffix-Complete

An *alphabet* U is a finite set. u denotes an element of U, $\bar{u}$ denotes a finite string of elements of U, and ϕ denotes the null string. U^* denotes the set of all $\bar{u}$ including ϕ, and U^+ denotes the set of all $\bar{u}$ excluding ϕ. A subset Σ of U^* will be called *prefix-complete* if $\bar{u} \in \Sigma$ implies all prefixes of $\bar{u}$ are elements of Σ *including* ϕ, *suffix-complete* if $\bar{u} \in \Sigma$ implies all suffixes of $\bar{u} \in \Sigma$ *excluding* ϕ.

Finite Automata and Black Boxes: Reachable States

A *finite automaton* will mean a Mealy model finite state automaton with initial state specified, namely, a 6-tuple

$$FA = \langle U, S, Y, f_{\text{tr}}, f_{\text{out}}, s_0 \rangle,$$

where U is the *input alphabet*, S is the *state alphabet*, Y is the *output alphabet*, f_{tr}: $S \times U \to S$ is the *state transition function*, f_{out}: $S \times U \to Y$ is the *output function*, and s_0 is the *initial state*. For any $\bar{u} \in U^*$, $s_{\bar{u}}$ denotes the *state reachable by* $\bar{u}$, namely the state which results if FA is started in state s_0 and input string $\bar{u}$ is applied. In particular, $s_\phi = s_0$. For any subset T of U^*, S_T denotes the set of states which are reachable by some $\bar{u} \in T$.

A *black box* is a triple $\langle U, Y, b \rangle$ where U is the input alphabet, Y is the output alphabet, and b: $U^+ \to Y$ is the *I/O function* of the black box. The black box will be denoted b. The intended interpretation is that $y = b(\bar{u})$ is the final output if $\bar{u}$ is applied to the black box. For any $\bar{u} \in U^*$, $b_{\bar{u}}$ denotes the *black box state* which results if $\bar{u}$ is applied to b. Namely, $b_{\bar{u}}$ is the I/O function defined by

$$b_{\bar{u}}(\bar{v}) = b(\bar{u}\bar{v}) \qquad \text{for all} \quad \bar{v} \in U^+.$$

In particular, $b_\phi = b$. The I/O function of a finite automaton FA is a black box which will be denoted $B(FA)$. FA will be said to *realize* b if $B(FA) = b$. The black box value $b(\phi)$ is undefined because finite automata are Mealy model.

Experiments

For any $\bar{w} \in U^+$ the *experiment* $e_{\bar{w}}$ is the functional

$$e_{\bar{w}}(b) = b(\bar{w}) \qquad \text{for all} \quad b,$$

Therefore,

$$e_{\bar{w}}(b_{\bar{u}}) = b(\bar{u}\bar{w}).$$

The output $e_{\bar{w}}(b)$ is the *result* of experiment $e_{\bar{w}}$ performed on b. $e_{\bar{w}}(FA)$ means $e_{\bar{w}}(B(FA))$. The experiment e_ϕ is undefined because finite automata are Mealy model.

Data

Data (set of data, body of data) is a finite set of pairs

$$D = \{(\bar{u}_1, y_1), \ldots, (\bar{u}_n, y_n)\},$$

where $\bar{u}_i \in U^+$, $y_i \in Y$, and the $\bar{u}_i$ are all different. Each pair $(\bar{u}_i, y_i)$ is a ***datum.*** $D(\bar{u})$ will mean y if $(\bar{u}, y)$ is a datum of D, undefined otherwise. A black box B *agrees* with D if

$$b(\bar{u}_i) = y_i \qquad \text{for} \quad i = 1, \ldots, n.$$

A finite automaton FA *agrees* with D if $B(FA)$ agrees with D. I will use the following notation to specify D:

$$D = \begin{cases} 00 \to 2 \\ 01 \to 1. \\ 1 \to 2 \end{cases}$$

If D is *prefix-complete*, i.e., if its set of input strings is prefix-complete, then the abbreviated notation

$$D = \begin{cases} 10 \to 10 \\ 0110 \to 0111, \end{cases}$$

means

$$D = \begin{cases} 1 \to 1 \\ 10 \to 0 \\ 0 \to 0 \\ 01 \to 1 \\ 011 \to 1 \\ 0110 \to 1. \end{cases}$$

State Characterization Matrix

A *set of test states* is a finite set of input strings

$$T = \{\bar{u}_1, \ldots, \bar{u}_n\} \text{ with } \bar{u}_1 = \phi.$$

Define the *set of transition states* to be

$$X(T) = TU - T$$
$$= \{\bar{u}u : \bar{u} \in T, u \in U, \bar{u}u \notin T\}.$$

A *set of experiments* will ambiguously mean a set of non-null input strings

$$E = \{\bar{w}_1, \ldots, \bar{w}_r\} \qquad \text{with} \quad \bar{w}_i \neq \phi.$$

The elements of E will sometimes be considered to be the experiments $e_{\bar{w}_i}$ determined by the $\bar{w}_i$. A *state characterization matrix* is a triple $\langle T, E, M\rangle$ where M is a matrix with labeled rows and columns such that

1. The rows are labeled with the elements of $T \cup X(T)$.
2. The columns are labeled with the elements of E.
3. Each entry of M is either an element of Y or a "hole."
4. If $\bar{v}_i, \bar{v}_j \in T \cup X(T)$ and $\bar{w}_i, \bar{w}_j \in E$ and $\bar{v}_i\bar{w}_i = \bar{v}_j\bar{w}_j$ then the $(\bar{v}_i, \bar{w}_i)$ and $(\bar{v}_j, \bar{w}_j)$ positions in M will be called *tied.* Tied positions must have the same entry.

The *data contained in* M is
$D(M) = \{(\bar{v}\bar{w}, y)$: $\bar{v} \in T \cup X(T)$, $\bar{w} \in E$, the $(\bar{v}, \bar{w})$ entry of M is $y \in Y\}$.

4. State Characterization from Given Data

Transition Assignment Problem $P_{\text{TrAss}}(D, T)$

Given data D, to use state characterization a set T of test states is chosen and the hypothesis is made that D agrees with some finite automaton FA whose states are reachable by T, i.e., $s_T = S$. In order to construct FA, note that its input set U and output set Y are determined by D. For the present we can take its state set S to be T, although we may later find that some of these states can be identified. It is assumed that each state $t_{\bar{u}} \in T$ will be reachable by $\bar{u}$, i.e., $s_{\bar{u}} = t_{\bar{u}}$. So the initial state of FA must be $s_0 = t_\phi \in T$. So it only remains to construct f_{tr} and f_{out}.

The problem of constructing f_{tr} will be called the *transition assignment problem* $P_{\text{TrAss}}(D, T)$. Since $t_{\bar{u}}$ is to be reachable by $\bar{u}$, if $\bar{u}, \bar{u}u \in T$ then necessarily $f_{\text{tr}}(t_{\bar{u}}, u) = t_{\bar{u}u}$. So the transition assignment problem is to identify each transition state $x \in X(T)$ with some test state $t \in T$ in a way which is consistent with the data D.

If f_{tr} can be constructed it is easy to construct f_{out} so that FA has the desired property. This justifies the definition:

Transition assignment question $Q_{\text{TrAss}}(D, T)$. Given: data D, test states T. Question: Is there a finite automaton with states reachable by T which agrees with D?

Data Matrix $M(D, T, E)$; Obviously Different Rows

Given D and having chosen T we now choose a set E of experiments. The *data matrix* $M(D, T, E)$ is the state characterization matrix with rows $T \cup X(T)$ and columns E such that for $\bar{u} \in T \cup X(T)$, $\bar{w} \in E$ the $(\bar{u}, \bar{w})$ entry is $D(\bar{u}\bar{w})$ if defined by D, a hole otherwise (see Fig. 1). Given D, T it is always possible to choose E large enough so that all the data of D is contained in (the entries of) $M(D, T, E)$, but this is not required by the definition.

The $\bar{u}_1$- and $\bar{u}_2$-rows of $M(D, T, E)$, or of any state characterization matrix M, will be called *obviously different* if for some $e \in E$ the entries $(\bar{u}_1, e)$ and $(\bar{u}_2, e)$ of M are different and neither is a hole. In this case $\bar{u}_1$, $\bar{u}_2$ will be said to be *obviously different in* M. If M has no holes, then 2 rows are either identical or obviously different.

If we wish to construct FA in agreement with M such that the states of FA are the row labels of M, then each row of M is a vector of characteristics (attributes) of the state of FA which corresponds to its row label. The holes are "don't cares." So, 2 states of FA can't be identified if their rows of M are obviously different.

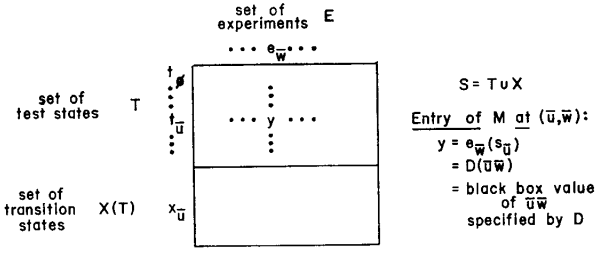

FIG. 1. Data matrix $M(D, T, E)$.

Automaton $FA(M)$ Constructed from State Characterization Matrix M

Let M be a state characterization matrix *with no holes* such that every x-row is identical to some t-row. The *constructed automata* $FA(M)$ are defined nondeterministically as follows. The definition is nondeterministic, but all the constructed automata are deterministic, and all of them satisfy Theorem 1.

The input, output alphabets U, Y of $FA(M)$ are specified by M. The state alphabet S of $FA(M)$ is any partitioning of T into equivalence classes such that if $\bar{u}$, $\bar{v} \in T$ are in the same equivalence class $[\bar{u}] \in S$ then (1) the $\bar{u}$- and $\bar{v}$-rows of M are identical, and (2) for all $u \in U$, if $\bar{u}u$, $\bar{v}u \in T$ then $[\bar{u}u] = [\bar{v}u]$, otherwise

the $\bar{u}u$- and $\bar{v}u$-rows of M are identical. The initial state s_0 of $FA(M)$ is $[\phi]$. For all $\bar{u} \in T$, $u \in U$, the dynamics of $FA(M)$ are defined by

$$f_{\text{out}}([\bar{u}], u) = \begin{cases} (\bar{u}, u)\text{-entry of } M \text{ if } u \in E \\ \text{arbitrary otherwise,} \end{cases}$$

$$f_{\text{tr}}([\bar{u}], u) = \begin{cases} [\bar{v}u] \text{ if there is a } \bar{v} \in [\bar{u}] \text{ such that } \bar{v}u \in T \\ \text{otherwise any } [\bar{v}] \text{ such that } \bar{v} \in T \text{ and the } \bar{v}\text{-row of } M \\ \text{is identical to the } \bar{u}u\text{-row.} \end{cases}$$

The Data Matrix Agreement Theorem says that if T is prefix-complete and E is suffix-complete then the states of $FA(M)$ are reachable by T and $FA(M)$ agrees with the data in M.

Hole Filling Problem $P_{\text{HolFil}}(D, T, E)$

Given data D, test states T, experiments E the *hole filling problem* $P_{\text{HolFil}}(D, T, E)$ is to fill the holes of the data matrix $M(D, T, E)$ to obtain a state characterization matrix M' such that every x-row is identical to some t-row. If some x-row was obviously different from every t-row in $M(D, T, E)$ then there is no solution to the hole filling problem. Otherwise, the hole filling question "Is there a solution to $P_{\text{HolFil}}(D, T, E)$?" may be difficult to decide because tied holes must be filled with the same $y \in Y$ for M' to be a state characterization matrix. An example is shown in Fig. 2.

Suppose we are given data D and a prefix-complete set T of test states. We can choose any set of experiments such that all the data of D is in $M(D, T, E)$ and augment it to obtain a set $E(D, T)$ of experiments with the additional property of being suffix-complete. If we can find a solution M' to the hole filling problem $P_{\text{HolFil}}(D, T, E(D, T))$ then $FA(M')$ is a finite automaton with states reachable by T which agrees with D.

The converse is easy to prove. So the question $Q_{\text{TrAss}}(D, T)$ "Is there an FA with states reachable by T which agrees with D?" is equivalent to "Is there a

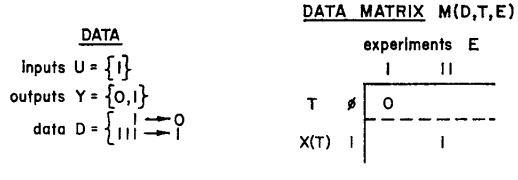

FIG. 2. Data matrix with rows which are not obviously different but can't be identical. Therefore, there is no finite automaton with 1 state which agrees with D. s_1 cannot be identified with s_ϕ because the 2 holes in $M(D, T, E)$ are tied. No matter how they are filled, s_1 will become obviously different from s_ϕ.

solution to $P_{\text{HolFil}}(D, T, E(D, T))$?'' This result will be used to prove that minimum automaton identification is *NP*-complete. Namely, Cook's prototype *NP*-complete problem will be reduced to the hole filling problem.

Augmentation of Test States

Still assuming that the test states T are prefix-complete, if there is no solution to the hole filling problem $P_{\text{HolFil}}(D, T, E(D, T))$ then there is no finite automaton with states reachable by T which agrees with D. So it is necessary to augment T.

For example, in the data matrix $M(D, T, E(D, T))$ suppose that $x \in X(T)$ was obviously different from all $t \in T$ before we tried filling holes. Then we would probably add x to T. If no x was obviously different from all $t \in T$ in the data matrix but every way of filling the holes (with tied holes being filled the same) gives M' in which some x is not identical to any of the $t \in T$, then it is difficult to decide how to augment T; see Sect. 7.

In any case, if we start with a prefix-complete T, say $T = \{\phi\}$, and always augment T with elements of $X(T)$, then T will remain prefix-complete.

5. Theorem 1. Data Matrix Agreement

Jean-Paul Brassard constructed the example of Fig. 3 to show me that it is possible that a state characterization matrix M has no holes, and every x-row is identical to some t-row so that $FA(M)$ can be constructed, but $FA(M)$ doesn't agree with $D(M)$. In this example $FA(M)$ is uniquely defined. It has 1 state s_0 and its only output is 1. Jean-Paul Brassard suggested that in this example $FA(M)$ does not agree with $D(M)$ because there are gaps in E, namely, $1 \in E$ and $111 \in E$, but $11 \notin E$. This observation led to the following theorem.

Theorem 1. *Let $\langle T, E, M\rangle$ be a state characterization matrix without holes such that every x-row is identical to some t-row, T is prefix-complete, and E is suffix-complete. Then $FA(M)$ agrees with the data $D(M)$ in M. Furthermore, if $\bar{u} \in T$ then starting $FA(M)$ in state s_0 and applying $\bar{u}$ puts $FA(M)$ in state $[\bar{u}]$.*

Corollary. *Let data D and test states T be given such that T is prefix-complete. Choose any suffix-complete set E of experiments such that $M(D, T, E)$ contains all the data in D. Then the question $Q_{\text{TrAss}}(D, T)$ "Is there a finite automaton with states reachable by T which agrees with D?" is equivalent to the question $Q_{\text{HolFil}}(D, T, E)$ "Can the holes of $M(D, T, E)$ be filled in such a way that every x-row will be identical with some t-row?" Given D, T a suitable E can be found in polynomial time.*

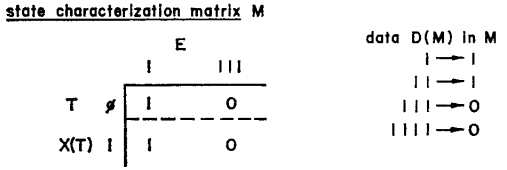

Fig. 3. Example of M such that $FA(M)$ does not agree with $D(M)$.

Proof. The proof of the corollary is straightforward and will be omitted.

The second conclusion of the theorem, that $\bar{u} \in T$ implies state $[\bar{u}]$ of $FA(M)$ is reachable by $\bar{u}$, follows immediately from the definition of f_{tr} in $FA(M)$, given that T is assumed to be prefix-complete. Namely, induction can be used on the prefixes of $\bar{u}$.

Let $\bar{u} \in T \cup X(T)$, $\bar{w} \in E$. It is to be proved that if $FA(M)$ is started in s_0 and $\bar{u}\bar{w}$ is applied, then its final output will be $y =$ the $(\bar{u}, \bar{w})$-entry of M. This is equivalent to saying that if $FA(M)$ is started in $s_{\bar{u}}$ and $\bar{w}$ is applied, then the final output will by y.

Case 1. $\bar{u} \in T$. Then $s_{\bar{u}} = [\bar{u}]$ by the second conclusion of the theorem. Induction will be used on the length of $\bar{w}$. If $\bar{w} = u \in U$ then the output produced by applying u to $[\bar{u}]$ of $FA(M)$ is $f_{\text{out}}([\bar{u}], u)$. This is the $(\bar{u}, u)$-entry of M by the definition of f_{out}. So, assume that the first conclusion is proved for some $\bar{w}$ and all $\bar{u} \in T$. It only remains to prove it for $u\bar{w}$ and all $\bar{u} \in T$, assuming $u\bar{w} \in E$. (The hypothesis that E is suffix-complete implies that if $u\bar{w} \in E$ then $\bar{w} \in E$, so this is a valid statement of the inductive step.) u, $\bar{w}$ are now fixed. For all $\bar{u} \in T$ it is to be shown that if $FA(M)$ is started in $s_{\bar{u}u}$ and $\bar{w}$ is applied, then the final output is the $(\bar{u}, u\bar{w})$-entry of M which is the same as the $(\bar{u}u, \bar{w})$-entry since these 2 positions are tied. If $\bar{u}u \in T$ this follows from the inductive hypothesis. If $\bar{u}u \in X(T)$ let $[\bar{v}] = s_{\bar{u}u} = f_{\text{tr}}([\bar{u}], u)$. By definition of f_{tr}, $\bar{v} \in T$ and the $\bar{v}$-row of M is identical to the $\bar{u}u$-row. So it remains to prove that starting $FA(M)$ in $s_{\bar{u}u} = [\bar{v}]$ and applying $\bar{w}$ gives the $(\bar{u}u, \bar{w})$-entry $=$ the $(\bar{v}, \bar{w})$-entry of M as final output. Since $\bar{v} \in T$ this follows from the inductive hypothesis.

Case 2. $\bar{u} \in X(T)$. It is to be shown that if $FA(M)$ is started in $s_{\bar{u}}$ and $\bar{w}$ is applied the final output will be the $(\bar{u}, \bar{w})$-entry of M. Let $\bar{u} = \bar{u}'u$ where $\bar{u}' \in T$. By the second conclusion, $s_{\bar{u}'} = [\bar{u}']$. So, by the definition of f_{tr}, $s_{\bar{u}} = f_{\text{tr}}([\bar{u}'], u) = [\bar{v}]$ where $\bar{v} \in T$ and the $\bar{u}$- and $\bar{v}$-rows of M are identical. By the second conclusion $s_{\bar{u}} = [\bar{v}] = s_{\bar{v}}$. So it is to be shown that if $FA(M)$ is started in $s_{\bar{v}}$ and $\bar{w}$ is applied then the final output will be the $(\bar{u}, \bar{w})$-entry $=$ the $(\bar{v}, \bar{w})$-entry of M. Since $\bar{v} \in T$ this has been proved in Case 1.

6. Theorem 2. Transition Assignment is *NP*-Complete

Two input strings $\bar{u}$, $\bar{v}$ will be called *obviously different w.r.t. D* if there is an experiment $e_{\bar{w}}$ such that $e_{\bar{w}}(s_{\bar{u}}) = D(\bar{u}\bar{w})$ and $e_{\bar{w}}(s_{\bar{u}}) = D(\bar{v}\bar{w})$ are specified by D

and $e_{\bar{w}}(s_{\bar{u}}) \neq e_{\bar{w}}(s_{\bar{v}})$. Given D, $\bar{u}$, $\bar{v}$ it can be determined in polynomial time whether or not $\bar{u}$ and $\bar{v}$ are obviously different w.r.t. D.

THEOREM 2. *Let D, T range over pairs such that* $\mathrm{Card}(U) = \mathrm{Card}(Y) = 2$, $T = \{\phi, 1, \ldots, 1^{n-1}\}$ *for any n, and the test states are obviously different from each other* w.r.t. D. *Even with D, T restricted to these pairs, $Q_{\mathrm{TrAss}}(D, T)$ "Is there a finite automaton with states reachable by T which agrees with D?" is NP-complete.*

Minimum Automaton Identification Question $Q_{\min}(D, n)$. Given: data D, positive integer n. Question: Is there a finite automaton with n states which agrees with D?

COROLLARY. $Q_{\min}(D, n)$ *is NP-complete for* $\mathrm{Card}(U) = \mathrm{Card}(Y) = 2$.

The proof of the corollary is omitted. Dana Angluin has found a reduction of the type used below to prove Theorem 2, but more complicated, which proves Theorem 2 to be valid even if D is restricted to prefix-complete data sets. Therefore, $Q_{\min}(D, n)$ is NP-complete for prefix-complete D.

Satisfiability Question Which Will Be Reduced to Q_{TrAss}

The satisfiability question is known to be NP-complete for conjunctive normal form expressions, which are of the form

$$F' = C_1 \,\&\, \cdots \,\&\, C_n,$$

where the clauses C_i are of the form

$$C_i = c_{i1} \vee \cdots \vee c_{in_i}$$

where the c_{ij} are literals z_k or $\neg z_k$ where the z_k are Boolean variables. This question will be reduced to $Q_{\mathrm{TrAss}}(D, T)$ with (D, T) satisfying the constraints of Theorem 2. The first step is to reduce the question "Is F' satisfiable?" to the question "Is F satisfiable?" where F is a conjunctive normal form expression which satisfies the constraint that in each clause either none of the literals are complemented or all are. This reduction can be performed by replacing each clause,

$$C_i = z_{k_1} \vee \cdots \vee z_{k_a} \vee \neg z_{k_{a+1}} \vee \cdots \vee \neg z_{k_b},$$

by 2 clauses

$$C_{2i-1} = z_i' \vee z_{k_1} \vee \cdots \vee z_{k_a},$$

$$C_{2i} = \neg z_i' \vee \neg z_{k_{a+1}} \vee \cdots \vee \neg z_{k_b},$$

where $z_1', \ldots, z_n'$ are n new Boolean variables. It is easy to show that if specific values of $z_1, \ldots, z_1', \ldots$ satisfy F then the same values of $z_1, \ldots$ satisfy F'; and if specific values of $z_1, \ldots$ satisfy F' then it's easy to choose values of $z_1', \ldots$ in order to satisfy F.

Furthermore, it will be assumed that F has the same number n of variables and clauses. This can be accomplished by adding any number $\geqslant 1$ of new clauses

$$C_i'' = z_1'' \vee \cdots \vee z_r'',$$

where the z_j'' are any number $\geqslant 1$ of new variables.

For any such F define the 2 characteristic functions

$$I_F(i, j) = \begin{cases} \text{"hole"} & \text{if } z_j \in C_i \text{ or } \neg z_j \in C_i \\ 0 & \text{otherwise,} \end{cases}$$

$$\tau_F(i) = \begin{cases} 1 & \text{if } C_i \text{ contains uncomplemented variables} \\ 0 & \text{if } C_i \text{ contains complemented variables.} \end{cases}$$

Proof of Theorem 2

For any F of the above form, with n variables and clauses, let D_F consist of the data in the state characterization matrix M_F which is defined in Fig. 4.

$M_F = M(D_F, T_n, E_n)$ contains the data in D_F, T_n is prefix-complete, and E_n is suffix-complete. So, by the corollary to the Data Matrix Agreement Theorem it only remains to show that $Q_{\mathrm{HolFil}}(D_T, T_n, E_n)$ "Can the holes of M_F be filled such that each x-row is the same as some t-row?" is equivalent to "Is F satisfiable?"

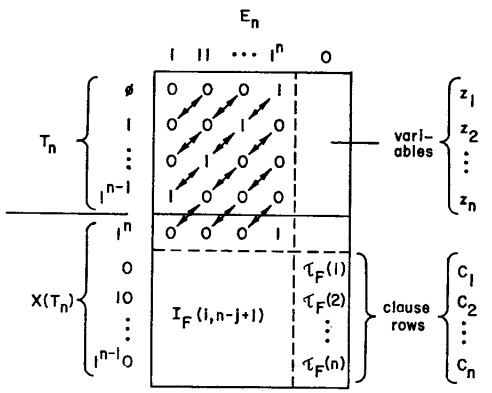

FIG. 4. State characterization matrix M_F such that the hole filling question is equivalent to "Is F satisfiable?" Double arrow denotes tied positions.

To see this, note that none of the holes of M_F are tied. The first x-row can certainly be made identical to the first t-row. In order to make the C_i-row identical to the z_j-row it is necessary to assign $\tau_F(i)$ to the rightmost position of the z_j-row. $\tau_F(i)$ is defined to be the value that a variable must be assigned in F in order to satisfy the C_i-clause. However, not every z_j can be assigned $\tau_F(i)$ to satisfy C_i in F. This is simulated in M_F by the $I_F(i, n-j+1)$ entry in the j-th position of the C_i-row. If this entry is 0 then the C_i-row cannot be made identical to the z_j-row which has a 1 in this position. By definition of I_F, the C_i-row has a hole in this position, and can be made identical to the z_j-row, iff in F the clause C_i can be satisfied by setting $z_j = \tau_F(i)$.

7. Theorem 3. Minimal Set of Test States May Not Be Minimum

Feasible set of test states. A set T of test states will be called feasible for data D if there is a finite automaton with states reachable by T which agrees with D.

State Selection Problem. Given: data D and an oracle for Q_{TrAss}. Find: a set T of test states of minimum cardinality which is feasible for D.

One approach to the problem of constructing a minimum state automaton in agreement with D is to use a heuristic algorithm for P_{TrAss} such as state characterization with some "timidity" as discussed at the end of Sect. 2. We are still left with the state selection problem. I don't know if the state selection problem is NP-complete, but the theorem of this section shows that the obvious approach does not work:

Construct any feasible finite automaton FA for D, e.g., $g_{\text{tabl}}(D)$ defined in the next section, which can be constructed in polynomial time but has lots of states. Find a prefix-complete set T of input strings such that the states of FA are reachable by T. Now try removing the test states of T one at a time, in an appropriate order so that T remains prefix-complete, using the oracle to determine if T remains feasible for D. This will yield a minimal T feasible for D. But the following theorem says that a minimal feasible T is not necessarily of minimum cardinality.

Theorem 3. *It is possible that set T of test states is feasible for data D, no proper subset of T is feasible for D, but there exists a T' of lower cardinality than T which is feasible for D.*

Proof. Consider the (prefix-complete) data

$$D = \begin{cases} 0110 \to 0111 \\ 10 \to 10. \end{cases}$$

The following 2 sets of test states are feasible for D:

$$T_1 = \{\phi, 0\}$$
$$T_2 = \{\phi, 1, 11\}.$$

It is shown below that they are minimal feasible. So T_2 is obviously not minimum.

Consider experiment e_0 performed on states s_ϕ, s_1, s_{011} (of any finite automaton which agrees with D). D includes the following data:

$$e_0(s_\phi) = 0; \qquad e_0(s_1) = 0; \qquad e_0(s_{011}) = 1.$$

So at least 2 test states are needed for a feasible T, namely ϕ, $\bar{u}$ where $s_{\bar{u}} = s_{011}$. This data implies $s_1 \neq s_{011}$ so $T = \{\phi, 1\}$ is not feasible.

Figure 6 shows that T_1, T_2 are feasible. To show that T_2 is minimal it is necessary to show that no pair of the 3 states of T_2 can be equivalent in a feasible finite automaton for D with states reachable by T_2. If $s_\phi = s_{11}$ or $s_1 = s_{11}$ then $\{\phi, 1\}$ would be a feasible set of test states for D. If $s_\phi = s_1$, then input 1 would take s_ϕ to s_ϕ, so $s_{11} = s_\phi$.

In order to see the difficulties involved in the state selection problem, let's follow this example a little further. Suppose we start with the hypothesis $T = \{\phi\}$ and construct the data matrix $M(D, T, E)$ with suffix-complete E shown in Fig. 5, which contains all the data in D. Neither of the x-rows is obviously different from the t-row. So, if we were able to determine that the hole filling problem has no solution, how would we know that we should add 0 to T, rather than 1, in order to achieve a minimum feasible T for D?

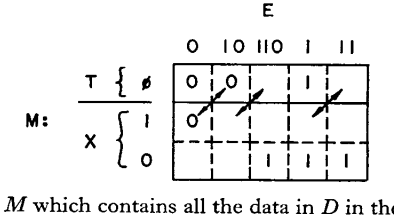

Fig. 5. Data matrix M which contains all the data in D in the proof of Theorem 3.

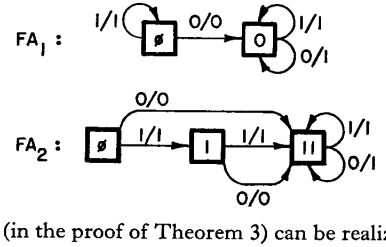

Fig. 6. Proof that D (in the proof of Theorem 3) can be realized by a finite automaton whose states are reachable by T_1 or T_2.

8. Theorem 4. Automaton Identification-in-the-Limit with Polynomial Time and Data

THEOREM 4. *There is an automaton identification rule $g(D)$ with the following properties*: (1) *feasible* ($g(D)$ *always agrees with* D), (2) *identification-in-the-limit*, (3) $g(D)$ *can be computed in polynomial time as a function of the size of* D, *and* (4) *for any black box* b *with* p *inputs and* n *state realization there is a data set* D_b *of size* $2n^2(p+1)$ *such that* $g(D)$ *is the minimum state realization of* b *for all* $D \supset D_b$. (4 *implies* 2.)

Proof. An automaton identification rule $g(D)$ will be demonstrated which has these 4 properties. First, an example $g_{\text{tim}}(D)$ of timid state characterization will be defined which lacks only the first property, feasibility. That is, $g_{\text{tim}}(D)$ will quickly and correctly identify b given D which contains the appropriate information. For other D the time to compute $g_{\text{tim}}(D)$ is reasonable, but $g_{\text{tim}}(D)$ may not even agree with D, much less realize b.

$g(D)$ will be obtained by changing $g_{\text{tim}}(D)$ to $g_{\text{tabl}}(D)$ if $g_{\text{tim}}(D)$ doesn't agree with D. $g_{\text{tabl}}(D)$ is easy to compute and agrees with D, but is unlikely to realize b.

Timid State Characterization

The following data D_b has a minimum state realization (finite automaton which agrees with D_b) which is easy to determine and is the minimum state realization for b: Let $T_b = \{\phi, \bar{u}_2, \ldots, \bar{u}_n\}$ be a minimum prefix-complete set of test states which reach all the states $b_{\bar{u}}$ of b. There are at most np transition states $\bar{v}_{iu} = \bar{u}_i u \in X(T_b)$. D_b will be constructed so that every pair $\bar{u}_j$, $\bar{u}_k$ of test states are obviously different w.r.t. D_b, and each transition state $\bar{v}_{iu}$ is obviously different from every test state $\bar{u}_k$ except the correct one $\bar{u}_{j_i}$. Let experiment $e_{\bar{w}_{jk}}$ distinguish $b_{\bar{u}_j}$ and $b_{\bar{u}_k}$, i.e.,

$$e_{\bar{w}_{jk}}(b_{\bar{u}_j}) \neq e_{\bar{w}_{jk}}(b_{\bar{u}_k}).$$

Then, in order to distinguish the test states $\bar{u}_j$, $\bar{u}_k$ it is sufficient for D_b to contain at most $n(n-1)$ datums

$$\left.\begin{aligned} \bar{u}_j\bar{w}_{jk} &\to b(\bar{u}_j\bar{w}_{jk}) \\ \bar{u}_k\bar{w}_{jk} &\to b(\bar{u}_k\bar{w}_{jk}) \end{aligned}\right\} j, k = 1, \ldots, n;\ j < k.$$

In order to distinguish each transition state $\bar{v}_{iu}$ from the test states $\bar{u}_k$ with $k \neq j_i$ which are distinguishable from it, it suffices for D_b to include at most $2n(n-1)p$ datums

$$\left.\begin{aligned} \bar{v}_{iu}\bar{w}_{j_ik} &\to b(\bar{v}_{iu}\bar{w}_{j_ik}) \\ \bar{u}_{j_i}\bar{w}_{j_ik} &\to b(\bar{u}_{j_i}\bar{w}_{j_ik}) \end{aligned}\right\} \begin{aligned} &u \in U \\ &i, k = 1, \ldots, n \\ &k \neq j_i. \end{aligned}$$

Now D_b contains information which makes it easy to construct f_{tr}. Finally, in order to specify f_{out}, at most np datums are required:

$$\bar{u}_i u \to b(\bar{u}_i u) \left\{ \begin{aligned} &i = 1, \ldots, n \\ &u \in U. \end{aligned} \right.$$

The following *timid state characterization* rule for automaton identification will correctly identify b if presented $D \supset D_b$: Given any D, in order to construct $g_{\text{tim}}(D)$ initialize $T = \{\phi\}$. If some $x \in X(T)$ is obviously different from all $t \in T$, then add x to T. Continue until T_D is obtained such that for each $x \in X(T_D)$ there is a $t \in T_D$ such that t is not obviously different from x. $g_{\text{tim}}(D)$ is now defined by

$$S = T_D,$$

$$\text{for } t \in T_D,\ u \in U: f_{\text{tr}}(t, u) = \begin{cases} tu \text{ if } tu \in T_D \\ \text{otherwise any } t' \in T_D \text{ such that } t' \text{ is not} \\ \text{obviously different from } tu \text{ w.r.t. } D, \end{cases}$$

$$\text{for } t \in T_D,\ u \in U: f_{\text{out}}(t, u) = \begin{cases} D(tu) \text{ if specified by } D \\ \text{arbitrary otherwise.} \end{cases}$$

Actually, in the construction of D_b it is necessary to use some care in choosing T_b. Otherwise it is possible that $D \supset D_b$ but the set T_D of obviously different test states found by timid state characterization is not T_b. T_D will certainly be feasible for b, but D_b may not contain the appropriate information to uniquely determine the correct f_{tr}.

Let us constrain the construction of T_b as follows. First specify the rule $g_{\text{tim}}(D)$ precisely. In particular, in the construction of T_D it is necessary to state in what order the transition states $X(T)$ will be checked to see if they should be added to T. Now choose T_b by this same rule: Start with $T = \{\phi\}$. Check each $x \in X(T)$ in the same order as used in the definition of g_{tim} to see if b_x is different from all b_t with $t \in T$. If so, add x to T. The resulting T_b can be used as above to define D_b, and g_{tim} will have the stated properties.

Feasible Automaton Identification by Table Construction

For any data D let b^D_{tabl} be the black box

$$b^D_{\text{tabl}}(\bar{u}) = \begin{cases} D(\bar{u}) & \text{if specified by } D \\ y_0 & \text{otherwise,} \end{cases}$$

where $y_0 \in Y$ is any output element. Define $g_{\text{tabl}}(D)$ to be the minimum state realization of b^D_{tabl}.

Feasible Modification of Timid State Characterization

$$g(D) = \begin{cases} g_{\text{tim}}(D) & \text{if } g_{\text{tim}}(D) \text{ agrees with } D \\ g_{\text{tabl}}(D) & \text{otherwise.} \end{cases}$$

RECEIVED: January 30, 1976; REVISED: October 8, 1976

REFERENCES

AHO, A. V., HOPCROFT, J. E., AND ULLMAN, J. D. (1974), "The Design and Analysis of Computer Algorithms," Addison-Wesley, Reading, Mass.

ARBIB, M. A., AND MANES, E. G. (1974), Machines in a category, *SIAM Review* **16**, 163–192.

ARBIB, M. A., AND ZEIGER, H. P. (1969), On the relevance of abstract algebra to control theory, *Automatica* **5**, 539–606.

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

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

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

KARP, R. M. (1975), On the computational complexity of combinatorial problems, *Networks* **5**, 45–68.

NERODE, A. (1958), Linear automaton transformations, *Proc. Amer. Math. Soc.* **9**, 541–544.

ZEIGER, H. P. (1967), Ho's algorithm, commutative diagrams and the uniqueness of minimal linear systems, *Inform. Control* **11**, 71–79.

INFORMATION AND CONTROL **37**, 321–333 (1978)

Tape Bounds for Some Subclasses of Deterministic Context-Free Languages

Y. IGARASHI

Centre for Computer Studies, University of Leeds, Leeds, LS2 9JT, England

The tape complexity of context-free languages is investigated. It is shown that all the members of two distinct subclasses of deterministic context-free languages are recognizable in $O(\log n)$ tape complexity on off-line deterministic Turing machines.

1. INTRODUCTION

There are several interesting observations to be made concerning the tape complexity of context-free languages. An early result given by Lewis *et al.* (1965) is that every context-free language can be recognized by an off-line deterministic Turing machine of $O((\log n)^2)$ tape complexity. This is still the best result known. Sudborough (1975) shows that if all linear context-free languages can be recognized by off-line deterministic Turing machines of $O(\log n)$ tape complexity, then the nondeterministic and deterministic context-sensitive languages are identical. He also discusses a deterministic context-free language (abbreviated *DCFL*) which is $\log n$ tape complete for the family of DCFL's (Sudborough, 1976a). Some closure properties on the class of $O(\log n)$ tape complexity languages (Ritchie and Springsteel, 1972) and on the class of $O(\log n)$ tape complexity functions (Lind, 1974) are known. It is also known that the class of $O(\log n)$ tape complexity context-free languages is closed under the star operation if and only if the deterministic and nondeterministic $O(\log n)$ tape complexity classes are identical (Flajolet and Steyaert, 1974; Monien, 1975). These results focus attention on the class of $O(\log n)$ tape complexity languages. In particular, it is natural to ask whether large subclasses of the deterministic context-free languages are recognizable by off-line deterministic Turing machines of $O(\log n)$ tape complexity. The class of languages recognizable by deterministic one-counter automata (Valiant, 1973; Valiant and Paterson, 1975) is a trivial example of such a subclass. Ritchie and Springstael (1972) show that Dyck languages, standard languages, structured context-free languages, and bounded context-free languages are recognizable by deterministic two-way marking automata. Hence they are all in the class of deterministic $O(\log n)$ tape complexity languages (Ritchie and Springstael, 1972; Hartmanis, 1972). It is also known that any parenthesis language (Lynch, 1975; Mehlhorn, 1975), any two-sided Dyck

0019-9958/78/0373-0321$02.00/0