Theoretical Computer Science 23 (1983) 95-101
North-Holland Publishing Company

NOTE

ON THE FINITE-VALUEDNESS PROBLEM FOR SEQUENTIAL MACHINES

Tat-hung CHAN† and Oscar H. IBARRA*
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Communicated by R. Karp
Received December 1981
Revised July 1982

Abstract. For a set $A = \{A_1, \ldots, A_m\}$ of $n \times n$ matrices with nonnegative integer entries, let $P(A)$ be the set of all finite products of matices in A. We show that there is a square space algorithm to decide, given A, whether or not $P(A)$ is finite. As a corollary, we show that there is an exponential space algorithm to decide, given a nondeterministic Mealy sequential machine M with accepting states, whether there exists an integer $d \geq 0$ such that on every input, the number of accepting computations of M producing distinct outputs is at most d. If d exists, the smallest such d can be computed in space exponential in size$(M) + \log d$. The space bound reduces to polynomial for the analogous problem of ambiguity of nondeterministic finite acceptors.

1. Introduction

An acceptor M is said to be *d-ambiguous* for an integer $d \geq 0$ if on every input, M has at most d distinct accepting computations. M is *finitely ambiguous* if it is d-ambiguous for some d. The terms *d-valued* and *finite-valued* have analogous definitions for transducers if we count outputs on accepting computations instead of just accepting computations. The *finite-ambiguity problem* and the *finite-valuedness problem* refer to the corresponding decision problems for acceptors and transducers, respectively. For a finitely ambiguous acceptor M, the *degree of ambiguity* is the smallest d such that M is d-ambiguous.

It is known that there is an algorithm to decide for a nondeterministic finite automaton (nfa) M and a nonnegative integer d whether M is d-ambiguous [6]. In fact, there is an algorithm to decide given a nondeterministic finite transducer M – essentially a generalized sequential machine (gsm) with accepting states – and a nonnegative integer d whether M is d-valued [1]. The algorithms in [1, 6] are of polynomial-time complexity when d is fixed.

In this note, we consider the problem of deciding the existence of d. More precisely, we show that the finite ambiguity problem for nfa's and the finite-

† Current address: Department of Mathematics and Computer Science, State University College, Fredonia, NY 14063, U.S.A.

* This author was supported in part by NSF Grant MCS 81-02853.

valuedness problem for nondeterministic Mealy sequential machines with accepting states (i.e. nfa's without ε-moves that output a single symbol per move) are decidable. The proof involves a reduction to a decision problem concerning products of square matrices over nonnegative integers. The more general finite-valuedness problem for gsm's remains open.

2. Deciding whether $\boldsymbol{P}(\boldsymbol{A})$ is finite

In the following, A and B are $n \times n$ matrices of nonnegative integers, $A(i,j)$ is the (i,j)-entry of A, and $\|A\|$ is the sum of all entries of A, i.e. $\sum_{i,j} A(i,j)$. $\boldsymbol{A}$ denotes a finite set of $n \times n$ matrices of nonnegative integers, and $P(\boldsymbol{A})$ denotes the set of all finite products of matrices in $\boldsymbol{A}$.

Fact 1. *If A and B have no zero row and no zero column, then their product AB also has no zero row and no zero column.*

Fact 2. *If A and B have no zero row and no zero column, then $\|AB\| \geq \|A\|$ and $\|AB\| \geq \|B\|$. Furthermore $\|AB\| = \|A\|(=\|B\|)$ iff $B(A)$ is a permutation matrix.*

Proof. It suffices to prove the assertions relating AB and A; the others follow by transposition. Now

$$\|AB\| = \sum_{i,j}\sum_{k} A(i,k)B(k,j) = \sum_{i,k}\left[A(i,k)\sum_{j} B(k,j)\right]. \tag{1}$$

For each k, $\sum_j B(k,j) \geq 1$ because B has no zero row. Hence,

$$\|AB\| \geq \sum_{i,k} A(i,k) = \|A\|.$$

If B is a permutation matrix, then the colums of AB are just the columns of A permuted, so that $\|AB\| = \|A\|$. On the other hand, suppose B is not a permutation matrix. Then there is a k_0 such that $\sum_j B(k_0,j) \geq 2$. Since A has no zero column there is an i_0 such that $A(i_0,k_0) > 0$. From (1) we see that

$$\|AB\| \geq \|A\| + A(i_0,k_0) > \|A\|. \quad \square$$

Definition. A matrix A is said to have *Property U* if it has no zero row or zero column, and is not a permutation matrix.

Fact 3. *If A has Property U, then $P(\{A\}) = \{A^k\}$ is infinite.*

Proof. By Facts 1 and 2, $\|A\| < \|A^2\| < \|A^3\| < \cdots$. $\square$

The converse of Fact 3 is obviously false. Nevertheless, a suitable modification can be generalized to yield a necessary and sufficient condition for $P(A)$ to be infinite.

Definition. Let A be an $n \times n$ matrix, and let J be a nonempty subset of $\{1, \ldots, n\}$. Then $\pi_J^n(A)$ denotes the matrix obtained by deleting from A those columns and rows whose indices are *not* in J.

Theorem 1. *Let $A = \{A_1, \ldots, A_m\}$, each A_i an $n \times n$ matrix. $P(A)$ is infinite if and only if there is a B in $P(A)$ and a nonempty subset J of $\{1, \ldots, n\}$ such that $\pi_J^n(B)$ has Property U.*

Proof. 'If'. Suppose B and J exist as stated. By Fact 3, $\{(\pi_J^n(B))^k\}$ is infinite. Now it is easy to check that $\pi_J^n(B^k) \geqslant (\pi_J^n(B))^k$, where $\geqslant$ holds componentwise. This shows that $\{B^k\}$ and hence $P(A)$ are infinite.

'Only if'. We proceed by double induction on the size measure (n, m) of A, where $(n_1, m_1) < (n_2, m_2)$ iff $n_1 < n_2$ or ($n_1 = n_2$ and $m_1 < m_2$). The base case $n = 1$ of the outer induction is trivial whereas the base case $m = 0$ of the inner induction holds vacuously. For the inductive step of the inner induction, consider $A = \{A_1, \ldots, A_m\}$ with $n \geqslant 2$ and $m \geqslant 1$ such that $P(A)$ is infinite.

Case 1: None of $A_1, \ldots, A_m$ has a zero row or zero column. Then in fact one of them must have Property U, as otherwise they are all permutation matrices, implying that $|P(A)| \leqslant n!$, a contradiction.

Case 2: At least one A_i has a zero row or zero column. Without loss of generality, assume that A_1 has zero bottom row (row n). Let $A' = \{A_2, \ldots, A_m\}$. If $P(A')$ is infinite, then the desired B and J exist by the inner inductive hypothesis. Otherwise

(1) $P(A')$ is a finite set, say $\{C_1, C_2, \ldots, C_s\}$, where $C_1 = I$ (the identity matrix), and

(2) it must be the case that the set of products of matrices in A which contain A_1 is infinite.

Now these products can be written in the form

$$C_{i_0} A_1 C_{i_1} A_1 C_{i_2} \cdots A_1 C_{i_t}, \quad t \geqslant 1.$$

Because there can only be s choices for C_{i_0}, in fact the set of products of the form

$$A_1 C_{i_1} A_1 C_{i_2} \cdots A_1 C_{i_t}, \quad t \geqslant 1$$

must be infinite. Let $D_i = A_1 C_i$, and $D = \{D_1, \ldots, D_s\}$. Then $P(D)$ is infinite. But each D_i has zero bottom row and so can be written as

$$D_i = \left[\begin{array}{c|c} X_i & Y_i \\ \hline 0 & 0 \end{array}\right].$$

Hence

$$D_{i_1} \cdots D_{i_t} = \left[\begin{array}{c|c} X_{i_1} \cdots X_{i_t} & X_{i_1} \cdots X_{i_{t-1}} Y_{i_t} \\ \hline 0 & 0 \end{array}\right]$$

which implies that $P(\{X_1, \ldots, X_s\})$ is infinite. Since the X_i's have dimension $n-1$, by the outer inductive hypothesis there is a product $B' = X_{i_1} \cdots X_{i_t}$ and a nonempty $J \subseteq \{1, \ldots, n-1\}$ such that $\pi_J^{n-1}(B')$ has Property U. But then $\pi_J^{n-1}(B') = \pi_J^n(B)$ where $B = D_{i_1} \cdots D_{i_t}$. □

Corollary 1. *There is a nondeterministic linear space algorithm, and hence a deterministic square space algorithm, to decide, for an arbitrary set of matrices* $\boldsymbol{A}$, *whether* $P(\boldsymbol{A})$ *is infinite.*

Proof. The nondeterministic algorithm is

$B := I$;
do forever
  guess some A_i in $\boldsymbol{A}$;
  $B := B \otimes A_i$;
  guess a nonempty $J \subseteq \{1, \ldots, n\}$;
  if $\pi_J^n(B)$ has Property U **then** [**output**('accept'); **halt**]
od

$\otimes$ means that in the matrix multiplication the algorithm only keeps track of whether an entry is 0, 1, or ≥ 2. This eliminates element growth so that the algorithm runs in linear space. The correctness of the algorithm follows from Theorem 1. The deterministic version is obtained by Savitch's simulation [5]. □

3. The finite-ambiguity and finite-valuedness problems

We can use Corollary 1 to decide the finite-ambiguity problem for nondeterministic finite automata (nfa's). We shall consider only nfa's without ε-moves; extension to nfa's with ε-moves is straightforward. An nfa is a 5-tuple $M = \langle S, \Sigma, \delta, s_1, F\rangle$, where S, Σ, and F are the sets of states, input symbols, and accepting states, respectively, s_1 is the start state, and δ is a mapping from $S \times \Sigma$ into the set of all subsets of S. Given a string x, let $c_M(x)$ be the number of distinct accepting computations of M on input x. Note that $c_M(x) = 0$ if and only if x is not accepted.

We now describe an algorithm to decide if there exists an integer d such that $c_M(x) \leq d$ for all x in Σ^*.

Let $S = \{s_1, \ldots, s_n\}$. For each a in Σ construct an $n \times n$ matrix T_a such that

$$T_a(i,j) = \begin{cases} 1 & \text{if } s_i \xrightarrow{a} s_j \text{ is an arc in } M, \\ 0 & \text{otherwise.} \end{cases}$$

Let $\boldsymbol{A} = \{T_a \mid a \in \Sigma\}$. For each $w = a_1 \cdots a_m$ in Σ^*, let T_w denote $T_{a_1} \cdots T_{a_m}$. It is easily shown by induction that $T_w(i,j)$ is the number of distinct computations of M that change state s_i to s_j on input w.

Now we can assume that each state s_i of M

(1) can be reached from the start state s_1, and

(2) can each some accepting state,

since any states violating either condition can be deleted without changing the set of accepting computations. Clearly, if $P(A)$ is finite, then M is finitely ambiguous. Conversely, suppose $P(A)$ is infinite. Then there exists (i, j) such that for all k, there is some $w_k \in \Sigma^*$ such that $T_w(i, j) \geq k$. Let u, v be strings such that on input u (respectively v) M can change state from s_1 to s_i (respectively from s_j to an accepting state). Then for all k, uw_kv has k distinct accepting computations so M is not finitely ambiguous. In fact, the proof of Theorem 1 shows that there is a string w such that $w_k = w^k$ for $k \geq 1$.

If M is finitely ambiguous, we can compute its degree of ambiguity by testing for d-ambiguity for $d = 0, 1, 2, \ldots$. The following nondeterministic algorithm accepts (M, d) such that M is not d-ambiguous by working with the set of matrices $A = \{T_a \mid a \text{ in } \Sigma\}$ as in Corollary 1:

> $B := I$;
> **do forever**
>   **guess** some A_i in A;
> $B := B \otimes A_i$; (//the $\otimes$ operation identifies all integers $> d$//)
>   **if** $\Sigma_{s_j \in F} B(1, j) > d$ **then** [**output** ('accept'); **halt**]
> **od**

The deterministic version of this algorithm (cf. Corollary 1) decides whether M is d-ambiguous in space polynomial in size$(M) + \log d$. Hence

Theorem 2. *There is a polynomial space algorithm to decide, given an nfa M, whether it is finitely ambiguous. If M is finitely ambiguous, then its degree of ambiguity d can be computed in space polynomial in* size$(M) + \log d$.

The following proposition shows that it is unlikely that 'polynomial space' in Theorem 2 can be reduced to 'polynomial time'.

Proposition. *It is* PSPACE*-complete to decide, given an nfa M and an integer $d \geq 0$, whether M is d-ambiguous.*

Proof. By Theorem 2, we need only show PSPACE-hardness. In [4] it was shown that it is PSPACE-hard to decide, given a set $\{M_1, \ldots, M_k\}$ of deterministic finite automata (dfa's), whether there exists a string accepted by all the M_i's. Clearly, we can construct an nfa M from $M_1, \ldots, M_k$ such that M is $(k-1)$ ambiguous if and only if $M_1, \ldots, M_k$ do not accept a common string. The result follows. □

A nondeterministic Mealy sequential machine (or simply, nsm) is a 6-tuple $M = \langle S, \Sigma, \Delta, \delta, s_1, F\rangle$. S, Σ, s_1, and F are as in an nfa, and δ is a mapping from $S \times \Sigma$ into the set of all subsets of $S \times \Delta$. Thus, an nsm is an nfa without ε-moves that can output a single symbol per move. Given a string x in Σ^*, let $h_M(x)$ be the number of distinct outputs corresponding to accepting computations of M on input

x. Note that $h_M(x) = 0$ if and only if x is not accepted. M is *finite-valued* if there exists a nonnegative integer d such that $h_M(x) \leq d$ for all x in Σ^*.

Theorem 3. *There is an exponential space algorithm to decide, given an nsm M, whether it is finite-valued. If M is finite-valued, then the smallest integer d such that M is d-valued can be computed in space exponential in* size(M) + log d.

Proof. The algorithm consists of the following steps:

(1) Let $\Gamma = \Sigma \times \Delta$. Clearly, the set $L_M = \{(a_1, b_1) \cdots (a_n, b_n) \mid n \geq 0$, M on input $a_1 \cdots a_n$ has an accepting computation with output $b_1 \cdots b_n\}$ is a regular subset of Γ^*. Construct a dfa $M_1 = \langle S_1, \Gamma, \delta_1, s_1, F_1 \rangle$ accepting L_M.

(2) Next construct from M_1 an nfa M_2 with state set $S_1 \times \Delta$, and start state (s_1, b_1), where b_1 is any fixed element in Δ. The transitions in M_2 are defined as follows: If there is an arc

$$s \xrightarrow{(a,b)} t \text{ in } M_1$$

then define, for each c in Δ, the following arc in M_2:

$$(s, c) \xrightarrow{a} (t, b).$$

The set of accepting states of M_2 is $F_1 \times \Delta$.

It is easily verified by induction that $c_{M_2}(x) = h_M(x)$ for each x in Σ^*. Hence, M_2 is finitely ambiguous if and only if M is finite-valued. The result follows from Theorem 2. $\square$

Remark. The finite-ambiguity problem can be shown undecidable for 1-turn push-down automata without ε-moves by a reduction of the Post Correspondence Problem, and for ε-free counter machines by using the fact that it is undecidable for two arbitrary ε-free deterministic counter machines M_1, M_2 whether $L(M_1) \cap L(M_2)$ is empty [2]. The same techniques can be used to show that it is undecidable, for an arbitrary integer $d \geq 1$ and an arbitrary machine M *known* to be finitely ambiguous, whether M is d-ambiguous. For finite-turn counter machines, the d-ambiguity problem is decidable for each $d \geq 0$ [3], but the finite-ambiguity problem is open.

References

[1] E.M. Gurari and O.H. Ibarra, A note on finite-valued and finitely ambiguous transducers, *Math. Systems Theory*, to appear.

[2] J. Hartmanis and J.E. Hopcroft, What makes some language theory problems undecidable, *J. Comput. System Sci.* **4** (1970) 368–376.

[3] O.H. Ibarra, On some decision questions concerning pushdown machines, *Theoret. Comput. Sci.*, to appear.

[4] D. Kozen, Lower bounds for natural proof systems, *Proc. 18th Annual Symposium on Foundations of Computer Science* (1977) 254–266.
[5] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4** (1970) 177–192.
[6] R.E. Stearns and H.B. Hunt III, On the equivalence and containment problem for unambiguous regular expressions, grammars, and automata, *Proc. 22nd Annual Symposium on Foundations of Computer Science* (1981) 74–81.