SIAM J. COMPUT.
Vol. 11, No. 3, August 1982

THE EQUIVALENCE PROBLEM FOR DETERMINISTIC TWO-WAY SEQUENTIAL TRANSDUCERS IS DECIDABLE*

EITAN M. GURARI†

Abstract. The equivalence problem for deterministic two-way sequential transducers is a long time open problem which is known to be decidable for some restricted cases. Here, the problem is shown to be decidable also for the general case. In fact, the result holds even when the devices are allowed to make some finite number of nondeterministic moves.

Key words. transducer, decidability, PSPACE-completeness

1. Introduction. The problem of deciding the equivalence of deterministic two-way sequential transducers was first posed in [2]. Since then the problem has been shown to be decidable for the restricted cases in which the input heads are allowed to make only some finite number of reversals [5] or when the input strings are over some bounded language (i.e., a language of the form $a_1^* \cdots a_k^*$, where $a_1, \cdots, a_k$ are distinct symbols) [6]. These results are known to hold even when the devices are allowed to make some finite number of nondeterministic moves [5]. In this paper the above decidable results are shown to hold also for the general case, i.e., for deterministic two-way sequential transducers that are allowed to make some finite number of nondeterministic moves. On the other hand, it should be noted that the equivalence problem is known to be undecidable for nondeterministic one-way sequential transducers [4], [9]. The reader is referred to, e.g., [1] for the applicability of transducers in defining translations.

The results of this paper are given in the next section. The remainder of this section is devoted to recalling the (informal) definitions of the devices used in this paper.

A *two-way sequential transducer*, e.g., [2] is a two-way finite-state automaton which is augmented by an output tape. At the start of each computation, the two-way sequential transducer is set to a specific initial state, the input head is on the leftmost character of the input string and the output tape contains blanks only. An *atomic move* consists of changing the state of the finite-state control, moving the input head -1, 0 or 1 positions to the right and writing 0 or 1 nonblank characters on the output tape. Two such devices are said to be *equivalent* if they agree on all their input-output relations defined by their corresponding accepting computations. (Accepting configurations are assumed to be halting configurations.) A two-way sequential transducer is said to be *deterministic* if in each of its configurations it has at most one choice of next move. A *reversal-bounded m-counters machine*, e.g., [8] is a one-way finite-state automaton which is augmented by m ($\geqq 1$) counters. Each of the counters is capable of storing any nonnegative integer. On each atomic move, at most one of the counters is incremented by -1 or $+1$, while in every computation, a counter can alternately increase and decrease its value by no more than some finite number of times.

2. The results. Theorem 1 below is the main result of this paper. The proof of Theorem 1 generalizes an idea in [5].

THEOREM 1. *The equivalence problem is decidable for deterministic two-way sequential transducers.*

* Received by the editors December 19, 1979 and in revised form May 13, 1981. This research was supported in part by the National Science Foundation under grant MCS 79-09967.

† Department of Computer Science, State University of New York at Buffalo, Amherst, New York 14226.

Proof. Let M_1 and M_2 be any two given deterministic two-way sequential transducers. Without loss of generality it is assumed that M_1 and M_2 have a (same) distinguished symbol that they output when and only when they enter a halting configuration. Then a (nondeterministic) reversal-bounded 2-counters machine M is constructed such that M halts on some input if and only if M_1 and M_2 are inequivalent. (Note that M can be simulated by a pushdown automaton whose pushdown store behaves like a counter.) The result then follows from the decidability of the emptiness problem for reversal-bounded m-counters machines [5], [8].

M, when introduced with an input string, starts its computation by nondeterministically determining some positive integer, say v, and putting it into its counters, say C_1 and C_2. Then M nondeterministically simulates (in parallel) the computations of M_1 and M_2 on such an input. However, whenever M_i, $i=1$ or 2, is to output a symbol, M instead decreases C_i by 1 or leaves C_i unchanged depending on whether C_i contains a nonzero or a zero value, respectively. In addition, M records in its finite-state control the vth symbols in the outputs of M_1 and M_2 if and when they are encountered. M halts (on the given input) if and only if M_1 halts in an accepting configuration while M_2 does not halt in an accepting configuration, M_2 halts in an accepting configuration while M_1 does not halt in an accepting configuration, or M_1 and M_2 halt in accepting configurations but the symbols recorded in the finite-state control of M are distinct. The simulation of an accepting computation of M_i, $i=1, 2$, is given below.

In every accepting computation of M_i, its input head visits each of the symbols of the input strings at most s_i times, where s_i is the number of states of M_i. This is because M_i is deterministic and a repetition of a state on a symbol of a given input string implies a nonhalting computation. In addition, every such halting computation of M_i can be described by a time-input graph which shows the sequence of the transition rules used during the computation (see Fig. 1(a)). A node is at coordinate (ζ, μ) in the graph if and only if it corresponds to the transition rule associated with the ζth move in the computation and just before this move the input head of M_i was at the μth symbol of the input string.

Now, consider any accepting computation of M_i. Then a linear tree, say T, which describes the computation can also be constructed (see Fig. 1(b)). Each node in T corresponds to an ordered set of transition rules with a separator dividing the set into two. The ith node in T is associated with the ith symbol of the input string, say a_i, where

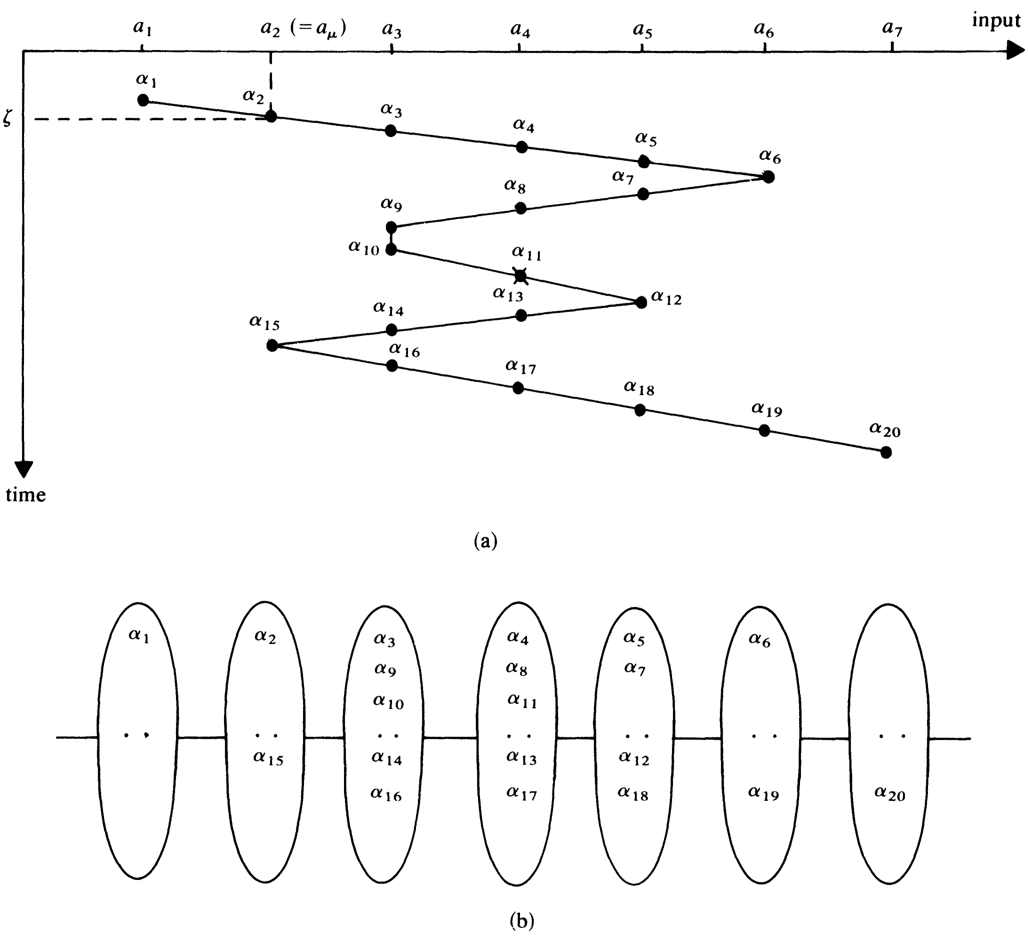
(i) the corresponding set of transition rules includes exactly those being used on the moves involving a_i and this in their order of being used; and

(ii) the separator divides the set of the transition rules into those used till (and including) the instant in which the vth output symbol is written (e.g., a subset of $\alpha_1, \cdots, \alpha_{11}$ in Fig. 1) and those used after the vth output symbol is written.

Thus, in simulating an accepting computation of M_i the device M needs just nondeterministically determine a sequence of ordered sets of distinct transition rules with a separator dividing each such set into two, where the sequence of these sets corresponds to the linear tree which describes the desired computation.

From the discussion above, the following algorithm can be given to describe the computation of the counter machine M on a given input.

```
$v \leftarrow$ (nondeterministically determine a positive integer); $C_1 \leftarrow v$; $C_2 \leftarrow v$;
$M_1$accept $\leftarrow$ 'false'; $M_2$accept $\leftarrow$ 'false';
**initialize** $V_1$; **initialize** $V_2$;
**while** both $V_1$ and $V_2$ contain at least one transition rule **do**
**begin** $V_1 \leftarrow$ **next** $V_1$; $V_2 \leftarrow$ **next** $V_2$; **end**;
```



a_1
a_2 $(=a_\mu)$
a_3
a_4
a_5
a_6
a_7
input
ζ
time
α_1 α_2 α_3 α_4 α_5 α_6 α_7 α_8 α_9 α_{10} α_{11} α_{12} α_{13} α_{14} α_{15} α_{16} α_{17} α_{18} α_{19} α_{20}
(a)
α_1
α_2 α_{15}
α_3 α_9 α_{10} α_{14} α_{16}
α_4 α_8 α_{11} α_{13} α_{17}
α_5 α_7 α_{12} α_{18}
α_6 α_{19}
α_{20}
(b)

FIG. 1. *A description of an accepting computation of a deterministic two-way sequential transducer by* (a) *a graph and* (b) *a linear tree.*

```
if $M_i$accept and $V_i$ contains at least one transition
            rule then reject; (*improper simulation*)          }
if not $M_i$accept and $V_i$ contains more than one transition } $i = 1, 2$
            rule then reject; (*improper simulation*)          }
if $M_1$accept $\neq$ $M_2$accept then accept; (*exactly one of $M_1$ and
                                    $M_2$ accept the input*)
if $M_1$accept and $M_2$accept then
    if $C_1 = C_2 = 0$ then
        if $sym_1 \neq sym_2$ then accept; (*both $M_1$ and $M_2$ accept the
                                    input but their $v$th output
                                    symbols are distinct*)
reject
```

At any given instant of the computation, V_i holds the ordered set of the transition rules used by M_i in the moves from the input symbol under consideration, $i = 1, 2$. However, if the ordered set consists of more than s_i elements (i.e. M_i enters an infinite loop), then only the (at most $s_i + 1$) transition rules till and including the first repetition of transition rule are held in V_i. In addition, V_i holds a separator which divides the set into two. Moreover, associated with each transition rule in V_i is the direction of the input head movement (i.e. $-1, 0$ or 1) just before the transition rule is reached. (In Fig. 1 the corresponding values of V_i are $((\alpha_1, 0), *)$, $((\alpha_2, 1), *, (\alpha_{15}, -1))$, $((\alpha_3, 1), (\alpha_9, -1), (\alpha_{10}, 0), *, (\alpha_{14}, -1), (\alpha_{16}, 1)), \cdots, (*, (\alpha_{20}, 1))$.) Whenever M (nondeterministically) determines a new value for V_i it also:

(a) reads an input symbol and checks that all the transition rules in the previous value of V_i are on such a symbol. This step, however, is not performed when V_i is initialized;

(b) decreases C_i by the value which equals the number of symbols contributed to the output of M_i by the transition rules that precede the separator in V_i;

(c) sets $M_i$accept to equal *true* if the last transition rule in V_i corresponds to a move to an accepting state;

(d) checks that the (new) value of V_i is compatible with its previous value;

(e) searches the simulated portion of the computation (given by the (new) value of V_i and the previous value of V_i) for a pair of consecutive transition rules, where the first transition rule immediately precedes a separator while the second transition rule immediately succeeds a separator. If such a pair is found, then M sets sym$_i$ to equal the output symbol of the first transition rule in this pair. □

For every two given deterministic sequential transducers M_1 and M_2 which are allowed to make some finite number, say k, of nondeterministic moves, two corresponding equivalent two-way sequential transducers $\hat{M}_1$ and $\hat{M}_2$ can be constructed with each of them being a union of 2^k deterministic two-way transducers. (At most two choices of next moves are assumed in each configuration of M_i, $i = 1, 2$). On a given input, $\hat{M}_i$, $i = 1, 2$, simulates its jth deterministic two-way sequential transducer. (The choice of j, $1 \leqq j \leqq 2^k$, is made nondeterministically.) The deterministic two-way sequential transducer, in turn, simulates the computation of M_i on the given input. However, whenever M_i is to make its lth, $1 \leqq l \leqq k$, nondeterministic move, the next move of the deterministic two-way sequential transducer is determined according to the value of the lth bit in the binary representation of j (zero corresponds to one choice and one corresponds to the other choice). Thus, M_1 and M_2 are inequivalent if and only if there exists an input-output relation definable by an accepting computation of a deterministic two-way sequential transducer which constitutes $\hat{M}_i$ but which is defined by none of the accepting computations of the deterministic two-way sequential transducers constituting $\hat{M}_{3-i}$, $i = 1$ or 2. Hence, the proof to Theorem 1 can be generalized to show also the next result.

THEOREM 2. *The equivalence problem is decidable for deterministic two-way sequential transducers which are allowed to make some finite number of nondeterministic moves.*

The nonemptiness problem for deterministic two-way finite-state automata is known to be PSPACE-complete [7]. On the other hand, the problem is solvable in t^{cm} time for reversal-bounded m-counters machines whose number of transition rules is t [5]. Thus, from [5], [7] and the proofs to Theorems 1 and 2, it follows that the inequivalence problem is PSPACE-complete for the transducers considered in Theorems 1 and 2. (See, e.g., [3] for the definition of PSPACE-complete.)

REFERENCES

[1] A. AHO AND J. ULLMAN, *The Theory of Parsing, Translation and Compiling*, vol. 1, Prentice-Hall, Englewood Cliffs, NJ, 1972.

[2] R. EHRICH AND S. YAU, *Two-way sequential transductions and stack automata*, Inform. and Control, 18 (1971), pp. 404–446.

[3] M. GAREY AND D. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1978.

[4] T. GRIFFITHS, *The unsolvability of the equivalence problem for ε-free nondeterministic generalized machines*, J. Assoc. Comput. Mach., 15 (1968), pp. 409–413.

[5] E. GURARI, *Transducers with decidable equivalence problem*, TR-CS-79-4, Univ. of Wisconsin Milwaukee, 1979; revised, TR-CS-81-182, State University of New York at Buffalo (1981).

[6] E. GURARI AND O. IBARRA, *Some decision problems concerning sequential transducers and checking automata*, J. Comput. and System Sci., 18 (1979), pp. 18–34.

[7] H. HUNT, *On the time and tape complexity of languages* I, Proc. of the Fifth Annual ACM Symposium on Theory of Computing, 1973, pp. 10–19.

[8] O. IBARRA, *Reversal-bounded multicounter machines and their decision problems*, J. Assoc. Comput. Mach., 25 (1978), pp. 116–133.

[9] ———, *The unsolvability of the equivalence problem for ε-free NGSM*'s *with unary input (output) alphabet and applications*, this Journal, 4 (1978), pp. 524–532.