The Equivalence Problem for Deterministic Two-Way Sequential Transducers is Decidable*

Eitan M. Gurari
Department of Computer Science
SUNY at Buffalo
4226 Ridge Lea Road
Amherst, N.Y. 14226

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. This even when the devices are allowed to make some finite number of nondeterministic moves.

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 [6] or when the input strings are over some bounded language (i.e., a language of the form $a_1^*\dots a_k^*$, where $a_1,\dots,a_k$ are distinct symbols) [7]. These results are known to hold even when the devices are allowed to make some finite number of nondeterministic moves [6]. 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 [5,10]. The reader is refered to, e.g., [1,4] for the applicability of transducers in defining translations.

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 left most 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 non blank 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 possible configurations it can use at most one transition rule. A reversal-bounded m-counters machine, e.g., [9] is a one-way finite-state automaton which is augmented by m (≥ 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 each 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 to Theorem 1 generalizes an idea in [6].

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

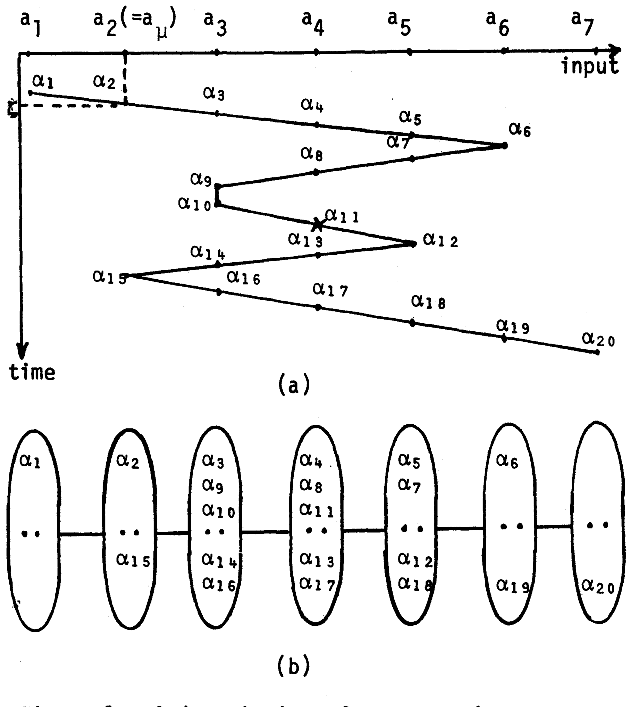
Proof. Let M_1 and M_2 be any two given deterministic two-way sequential transducers. Without loss of generality it is assumed that M_1 and M_2 have a (same) distinguished symbol that they output when, and only when, they enter to a halting configuration. Then a (nondeterministic) reversal-bounded 2-counters 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 an unbounded-reversal counter.) The result then follows from the decidability of the emptiness problem for reversal-bounded m-counters machines [6,9].

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 v'th 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

* Research supported in part by NSF Grant MCS79-09967.

accepting configuration while M_2 does not halt in an accepting configuration, or 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 string 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 Figure 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 ζ'th move in the computation and just before this move the input head of M was at the μ'th symbol of the input string.



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)

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

Now, consider any accepting computation of M_i. Then a linear tree, say, T, which describes the computation, can also be constructed (see Figure 1(b)). Each node in T corresponds to an ordered set of transition rules with a separator dividing the set into two. The i'th node in T is associated with the i'th 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 v'th output symbol is written (e.g., a subset of $\alpha_1,\ldots,\alpha_{11}$ in Figure 1) and those used after the v'th 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. ■

For every two given deterministic sequential transducers M_1 and M_2 which are allowed to make some finite number of nondeterministic moves two corresponding equivalent two-way sequential tranducers $\hat{M}_1$ and $\hat{M}_2$ can be constructed with each of them being a union of some finite number of deterministic two-way transducers. 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 makes $\hat{M}_i$ but which is defined by none of the accepting computations of the deterministic two-way sequential transducers making $\hat{M}_{3-i}$, 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 [8]. Thus, from [8] 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. Aho, A. and Ullman, J., "The theory of parsing, translation, and compiling," Vol. 1, Prentice-Hall, Englewood Cliffs, N.J., 1972.

2. Ehrich, R. and Yau, S., Two-way sequential transductions and stack automata, Information

and Control 18 (1971), 404-446.

3. Garey, M. and Johnson, D., "Computers and intractability: A guide to the theory of NP-completeness," H. Freeman, San Francisco, 1978.

4. Gonzalez, R. and Thomason, M., "Syntactic pattern recognition: An introduction," Addison-Wesley, Reading, Mass., 1978.

5. Griffiths, T., The unsolvability of the equivalence problem for ε-free nondeterministic generalized machines, Journal of the Association for Computing Machinery 15 (1968), 409-413.

6. Gurari, E., Transducers with decidable equivalence problem, TR-CS-79-4, University of Wisconsin-Milwaukee (1979).

7. Gurari, E. and Ibarra, O., Some decision problems concerning sequential transducers and checking automata, Journal of Computer and System Sciences 18 (1979), 18-34.

8. Hunt, H., On the time and tape complexity of languages I, Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (1973), 10-19.

9. Ibarra, O., Reversal-bounded multicounter machines and their decision problems, Journal of the Association for Computing Machinery 25 (1978), 116-133.

10. Ibarra, O., The unsolvability of the equivalence problem for ε-free NGSM's with unary input (output) alphabet and applications, SIAM Journal on Computing 4 (1978), 524-532.