AUTOMATA, TABLEAUX, AND TEMPORAL LOGICS
(Extended Abstract)

E. Allen EMERSON[1]

Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712

1. Introduction

There has recently been a resurgence of interest in the theory of finite automata on infinite trees (cf. [RA69], [RA70]) because of the intimate relationship between such automata and temporal logic (cf. [ST81], [WVS83]). For example, in testing satisfiability of a formula p_o in the branching time logic CTL, a directed graph labelled with appropriate subformulae of p_o, a *tableau*, is constructed (cf. [EH82], [EC82]). This tableau may be viewed as defining a finite automaton on infinite trees which accepts input trees that define models of p_o. The satisfiability problem for CTL is thus reduced to the nonemptiness problem for finite automata on infinite trees. A general framework for applying such automata-theoretic techniques is described in [VW84] where it is shown that for a number of temporal logics, satisfiability can be reduced to testing nonemptiness of Buchi automata on infinite trees resulting in a deterministic exponential time decision procedures. For more expressive logics such as CTL* ([EH83]), more elaborate reductions to tree automata with more complicated acceptance conditions yield superexponential, but still elementary time, decision procedures.

In this paper, we study tree automata in depth, and provide new, more efficient algorithms for testing nonemptiness for various classes of tree automata. Our work should therefore lead to more efficient decision procedures for testing satisfiability of temporal logics. For example, one consequence of our findings is that CTL* can be decided in nondeterministic double exponential time and double exponential space. (The best previous known upper bound was deterministic triple exponential time (cf. [ES84].) and triple exponential space.) Another is that the logic ECTL ([EH83]) is decidable in deterministic exponential time.

We begin by defining the notion of the transition diagram of a tree automaton. It is derived from the "AND/OR graph" formulation of the tableau for CTL as described in [EC82]. (Roughly, the OR-nodes correspond to states of the automaton, while the AND-nodes give the input symbols labelling the transitions between states.) Surprisingly, the transition diagram idea for tree automata has never before appeared in the literature. We believe that it is important because it makes tree automata "easier to think about" and the graph-theoretic

[1]Work supported in part by NSF Grant MCS8302878

nature of the algorithms for testing nonemptiness more apparent. Moreover, in the context of temporal logic, we see that while the tableau can be viewed as an automaton, the converse relationship also holds: we can view a tree automaton as a tableau and apply tableau-theoretic techniques to automata. As we shall see, this can result in improved algorithms for testing nonemptiness of autotmata which, in turn, yield improved decision procedures for temporal logics.

We then define the notion of a tree automaton running on (essentially) an arbitrary directed graph. If the tree automaton accepts a graph, we may think of the graph as a "model" of the (temporal modality) defined by the automaton. We are then able to prove a Linear Size Model Theorem for pairs tree automata: if a pairs tree automata accepts some tree, then in fact it accepts some graph with number of nodes linear in the size of the transition diagram of the tree automaton. The crucial property that makes this possible is that the family of sets of states appearing infinitely often along a path meeting the pairs condition is closed under union. This closure under union was used in [ST81] to get a finite tree-model theorem for complemented pairs automata; we use [HR72] to get a finite tree-model and then use closure under union to collapse down to the linear size model. Our linear size model theorem also applies to automata where acceptance is defined by means of a single complemented pair.

We go on to show that testing nonemptiness for pairs automata is in NP (cf. [RA70], [HR72]) For pairs automata with a single pair, or for complemented pairs automata with a single pair we can do much better: nonemptiness for these automata can be tested in deterministic polynomial time. We believe that this later result will have particularly significant applications in obtaining decision procedures.

Finally, we consider the limits of the "expressive power" of finite automata on infinite trees. We exhibit a natural correctness property, *uniform inevitability*, which is not definable by any type of finite automata on infinite trees. It is, however, definable by a pushdown automaton on infinite tress. This suggests that these automata also merit study (cf. [KP83]).

The remainder of the paper is organized as follows: Section 2 gives preliminary definitions including the important notions of the transition diagram for a tree automaton and runs of a tree automaton on graphs. Section 3 describes how the tree automaton can be viewed as a tableau. The linear size model theorem for pairs and single complemented pairs tree automata are proved in Section 4 while our new algorithms for testing nonemptiness are given in section 5. Section 6 defines the property of uniform inevitability and shows that it not definable by finite state tree automata. Finally, some concluding remarks are presented in Section 7.

2. Preliminaries

2.1. Automata. To simplify the exposition, we only consider automata on infinite binary trees here; the extension to infinite n-ary trees is routine. The set $\{0,1\}^*$ of all infinite strings over alphabet $\{0,1\}$, may be viewed as an *infinite binary tree* where the *root node* is the empty string λ and each *node* $v \in \{0,1\}^*$ has as its *successors* the nodes v0 and v1. A finite (infinite) *path* through the tree is a finite (resp., infinite) sequence $x = v_0,v_1,v_2,$ of nodes such that for all i, v_{i+1} is a successor of v_i. Let Σ be a finite alphabet of infinite symbols. An infinite binary Σ-tree is a labelling T which maps $\{0,1\}^* \dashrightarrow \Sigma$.

A *finite automaton* $\mathcal{A}$ *on infinite binary* Σ*-trees* consists of a tuple (Σ,S,δ,s_0) plus an *acceptance condition* (which is described subsequently) where

Σ is the *input alphabet* labelling the nodes of the input tree,

S is the set of *states* of the automaton,

$\delta : S \times \Sigma \dashrightarrow$ PowerSet($S \times S$ is the (nondeterministic) *transition function*, and

$s_0 \in S$ is the *start state* of the automaton.

A *run* of $\mathcal{A}$ on the input Σ-tree T is a function $\rho : \{0,1\}^* \dashrightarrow S$ such that for all $v \in \{0,1\}^*$ $(\rho(v0), \rho(v1)) \in \delta(\rho(v), T(v))$ and $\rho(\lambda) = s_0$. We say that $\mathcal{A}$ *accepts* input Σ-tree T iff $\exists$ a run ρ of $\mathcal{A}$ on T such that $\forall$ path x starting at the root of $\{0,1\}^*$ if $r = \rho|x$, the sequence of states $\mathcal{A}$ goes through along path x, then the acceptance condition (as below) holds along r.

For a *Buchi* automaton acceptance is defined in terms of a distinguished set of states, GREEN $\subseteq$ S. A sequence $r = s_1,s_2,s_3,...$ of states meets the Buchi condition specified by GREEN iff there exist infinitely many i such that $s_i \in$ GREEN. If we think of a green light flashing upon entering any state of GREEN then r meets the condition iff $\overset{\infty}{\exists}$ GREEN flashes along r.[2] For a *pairs* automaton we have a finite list $((RED_1,GREEN_1),...,(RED_k,GREEN_k))$ of pairs of sets of states (think of them as pairs of colored lights where $\mathcal{A}$ flashes the red light of the 1st pair upon entering any state of set RED_1, etc.): r meets the pairs condition iff $\exists$ pair i %el [1:k] ($\neg\overset{\infty}{\exists}RED_i$ flashes and $\overset{\infty}{\exists}GREEN_i$ flashes). Finally, a complemented pairs automaton is defined by the above pairs condition being false, i.e., $\forall$ pairs $i \in [1:k]$ ($\overset{\infty}{\exists}$ $GREEN_i$ flashes implies $\overset{\infty}{\exists}$ RED_i flashes).

The *transition diagram* of $\mathcal{A}$ is an AND/OR-graph where the set S of states of $\mathcal{A}$ comprises the the set of OR-nodes while the AND-nodes define the allowable moves of the

---

[2]We use $\overset{\infty}{\exists}$ to mean "there exist infinitely many" or "infinitely often" and $\overset{\infty}{\forall}$ to mean "for all but a finite number of instances" or "almost everywhere".

automaton: Suppose that for $\mathcal{A}$, $\delta(s,a) = \{(t_1,u_1),\ldots, (t_m,u_m)\}$ and $\delta(s,a) = \{(v_1,w_1),\ldots,(v_n,w_n)\}$ then the transition diagram contains the portion shown in figure 1. We merge OR-nodes representing the same of $\mathcal{A}$. See also figure 2. Intuitively, the OR-nodes indicate that a non-deterministic choice of move must be made while the AND-nodes indicate that the automaton must continue down both branches of the input tree.

We now define what it means for a tree automaton to run on a graph. A *Σ-labelled, binary directed graph* $G = (V,A_0,A_1,L)$ consists of

V - a set of underlying *nodes*,

A_0 - a total function mapping V --> V which assigns to each node v a unique successor node called the *0-successor* of v,

A_1 - defined similarly to A_0,

L - a *labelling* function V --> Σ which assigns to each node a symbol from Σ

A *run* of $\mathcal{A}$ on G is a mapping ρ:V --> S such that $\forall$ v $\in$ V, $\rho(A_0(v), A_1(v)) \in \delta(\rho(v), L(v))$ and $\rho(v_0) = s_0$. Intuitively, a run is a labelling of G with states of $\mathcal{A}$ consistent with the local structure of $\mathcal{A}$'s transition diagram.

2.2. Temporal Logic. CTL* is the branching time temporal logic with basic modalities of the form, A ("for all paths) or E ("for some path), followed by an arbitrary linear time formula (involving nestings and boolean connectives as desired) over the linear time operators Fp ("sometimes p"), Gp ("always p"), Xp ("nexttime p"), and [p U q] ("q sometime holds and p holds up until then"). In the restricted logic CTL, the basic modalities are of the form: A or E followed by a single F, G, X, or U. We use $\overset{\infty}{F}p$ to abbreviate GFp ("infinitely often p") and $\overset{\infty}{G}p$ to abbreviate FGp ("almost everywhere p"). ECTL is the logic with basic modalities of the form A or E followed by a single F, G, X, U, $\overset{\infty}{F}$, or $\overset{\infty}{G}$.

3. Automata and Temporal Tableau

In the full paper we will show that tree automata, the tableau as defined in [EC82] or [EH82] or [BMP81], and the maximal model construction of [WVS83], [VW84] are all equivalent formalisms.

4. Linear Size Model Theorem

Note that in testing nonemptiness we can, without loss of generality, restrict our attention to automata over a 1 symbol alphabet (cf. [Hr72]). In the remainder of the paper we restrict our attention to such automata in order to simplify the exposition. Our results generalize in a routine way to automata over multi-symbol alphabets.

4.1. Theorem. Suppose $\mathcal{A}$ is a pairs tree automaton over a single letter alphabet. If $\mathcal{A}$ accepts some tree T then $\mathcal{A}$ accepts some graph G with number of nodes linear in the size of $\mathcal{A}$'s transition diagram.

proof sketch. By [HR72] we know that $\mathcal{A}$ accepts some finitely generated tree with a finite number of nodes. This may be viewed as a finite graph G accepted by $\mathcal{A}$. The accepting run induces a labelling of the nodes of G with the states of $\mathcal{A}$ consistent with $\mathcal{A}$'s transition table. We claim that we can chop out duplicate nodes, i.e., nodes with the same label. Suppose nodes u and v have the same label. If we redirect all the arcs coming into v so that they point instead to u, and then delete all nodes (and incident arcs) that are no longer reachable from the start node v_0, then the resulting graph G' will still be labelled consistently with respect to the transition diagram of $\mathcal{A}$. It will moreover have at least one less pair of duplicate nodes than G since at the very least node u was chopped out. If the run of $\mathcal{A}$ defined on G' is accepting then we have successfully obtained a strictly smaller accepted graph. However, the run of $\mathcal{A}$ defined on G' may not be accepting because we may have introduced a path violating the pairs condition. This path which was not present in G must end in a cycle of the form: w --> u ------> w where w --> v was an arc in G redirected to be the arc w --> u in G', and u ------> w is a path from node u to node w that was present in G and is still present in G'. Similarly, we can try to replace u by v. This might yield an accepted smaller graph G" or it might introduce a cycle violating the pairs condition of the form y --> v ------> y where y --> u was an arc in G changed to y --> v in G" and v ------> y is a path present both in G and G".

It must be possible to either replace v by u or replace u by v. If both replacements were to lead to a nonaccepting graph, then, as above, we would have present in the original graph G the following arcs and paths: w --> v, u ------> w, y --> u, v ------> y. These form the following cycle in the original G: w --> v ------> y --> u --> ··· --> w which itself violates the pairs condition because it is the union of two violating cycles (those above). This contradiction shows that one of the replacements must be possible.

Hence, given any two duplicate nodes, we can always eliminate one of them. We continue chopping out duplicates until none remain. The final graph contains at most a single node labelled with any given state of $\mathcal{A}$. □

We can similarly establish

4.2. Theorem. Suppose $\mathcal{A}$ is a complemented pairs tree automaton with a single pair over a single letter alphabet. If $\mathcal{A}$ accepts some tree T then $\mathcal{A}$ accepts some graph G with number of nodes linear in the size of $\mathcal{A}$'s transition diagram.

5. Testing Nonemptiness

5.1. Theorem. Testing nonemptiness for pairs type automata $\mathcal{A}$ is in NP.

proof sketch. By the Linear Size Model Theorem we know that there is a graph G accepted by $\mathcal{A}$ of size linear in $\mathcal{A}$. Guess such a G. We must now check that along every path starting in the start node of G, the pairs condition holds, i.e., $\exists$ pair $i \in [1{:}k]$ ($\neg\overset{\infty}{\exists}RED_i$ and $\overset{\infty}{\exists}GREEN_i$). But this just amounts to a model checking problem in the logic FCTL of [EL84] which can be solved in polynomial (in fact, linear) time. □

Remark. Note that the proof of the Linear Size Model Theorem really tells us more than the theorem asserts. If the automaton $\mathcal{A}$ accepts some graph then it accepts a graph G contained in the transition diagram of $\mathcal{A}$ in this sense: G is of the form (T, A_0, A_1, L, s_0) where

(1) $T \subseteq S$, the state set of $\mathcal{A}$,
(2) for each $s \in T$, $(A_0(s),A_1(s)) \in \delta(s,\sigma)$
(3) for each $s \in T$, $L(s) = \sigma$, and
(4) $s_0 \in T$.

5.2. Corollary. Testing satisfiability for CTL* is in nondeterministic double exponential time and double exponential space.

Proof sketch. Redo [ES84] using pairs automata instead of complemented pairs automata. The pairs automata constructed will be of size double exponential in the length of the input formula. □

However, we can do much better for pairs automata with a single pair.

5.3. Theorem. There is an algorithm for testing nonemptiness of an input pairs automaton $\mathcal{A}$ with a single pair which runs in deterministic polynomial time.

Proof sketch. By the above remark, if $\mathcal{A}$ accepts any graph then it accepts some graph G contained in (the transition diagram of) $\mathcal{A}$. We must construct G so that each node in G has sufficient successors in G In tableau terms if OR-node s is in G we say that s has sufficient successors in G if there exists an AND-node son d of s in $\mathcal{A}$ which in turn has two OR-node sons t,u in G. We will try to pick sufficient successors for each OR-node s to get a model of the CTL* formula $A[\overset{\infty}{G}P \vee \overset{\infty}{F}Q]$ where $P \equiv \neg RED$ and $Q \equiv GREEN$. In other words, we must for each OR-node we include in G, we must strike a link to 2 successor OR-nodes inter-

mediated by an AND-node. (The model we will try to construct is built out of OR-nodes so our approach here is "dual" to that of [EC82] which built the model out of AND-nodes; we could formulate things in the dual approach but the present one seems more direct.)

We will compute the set of states of $\mathcal{A}$ at which $A[\overset{\infty}{G}P \wedge \overset{\infty}{F}Q]$ is satisfiable inductively based on the following fixpoint characterization: $A[\overset{\infty}{G}P \wedge \overset{\infty}{F}Q] = \mu Y.AFAG((P \vee Y) \wedge AF(Q \vee Y))$. (To see that the fixpoint characterization is valid, consider the dual property $E[\overset{\infty}{F}P \vee \overset{\infty}{G}Q]$ which, pretty obviously, has the dual fixpoint characterization $\nu Y.EGEF((P \wedge Y) \vee EG(Q \wedge Y))$.) Let $\tau[Y] = AFAG((P \vee Y) \wedge AF(Q \vee Y))$. Then we compute $Y^1 = \tau[\text{false}]$, $Y^2 = \tau[Y^1]$, etc. As we add add states to Y^i we choose sufficient successors so that if $s \in Y^i$, there really is a model of Y^i at s selected out of the tableau.

Details are provided in the full paper □.

Similarly we can show that

5.4. Theorem. There is an algorithm for testing nonemptiness of an input complemented pairs automaton $\mathcal{A}$ with a single pair which runs in deterministic polynomial time. □

We can reduce the emptiness problem for ECTL to pairs automata with a single pair to establish that

5.5. Corollary. ECTL is decidable in deterministic exponential time. □

6. Limits of Expressive Power of Tree Automata

The correctness property known as *inevitability* of predicate P is expressible in CTL* as, simply, AFQ - meaning that along every future computation path x, there exists a time i along x at which Q is true. The time i in general depends on the particular path x followed. In other words, it has the pattern ∀ path ∃ time P. The property defined by swapping the quantifiers to get the pattern ∃ time ∀ path P - meaning that there exists a time i such that along every future computation path x, P holds at time i - we call *uniform inevitability*. The time i is uniform over all paths x. We have the following

6.1. Theorem. Uniform inevitability is not definable by any finite state tree automaton with any acceptance condition; hence, this property is not expressible in any of the temporal logics shown to be decidable using finite state tree automata.

The proof uses a sort of "pumping lemma" type argument. However, it turns out that a tree automaton with a pushdown store can be used to define this property. (It guesses i by pushing i symbols on its stack initially and decrements the stack as it advances from level to level.)

7. Conclusion

We have studied the problem of testing nonemptiness for tree automata. We have shown that, by viewing the automata as tableaux, it is possible to give better algorithms for testing nonemptiness than were previously known, which, in many cases, also yield improved temporal decision procedures. We have also investigated the limits of the expressive power of tree automata.

Finally, among related work, we mention that some related results regarding the complexity of tree automata and of temporal logics were obtained by Vardi and Stockmeyer ([VS85]).

8. References

[AB80] Abrahamson, K., Decidability and Expressiveness of Logics of Processes, PhD Thesis, Univ. of Washington, 1980.

[BMP81] Ben-Ari, M., Manna, Z., and Pnueli, A., The Temporal Logic of Branching Time. 8th Annual ACM Symp. on Principles of Programming Languages, 1981.

[CE81] Clarke, E. M., and Emerson, E. A., Design and Synthesis of Synchronization Skeletons using Branching Time Temporal Logic, Proceedings of the IBM Workshop on Logics of Programs, Springer-Verlag Lecture Notes in Computer Science #131, 1981.

[CES83] Clarke, E. M., Emerson, E. A., and Sistla, A. P., Automatic Verification of Finite State Concurrent Programs: A Practical Approach, POPL83.

[EC80] Emerson, E. A., and Clarke, E. M., *Characterizing Correctness Properties of Parallel Programs Using Fixpoints*, Proc. ICALP 80, LNCS Vol. 85, Springer Verlag, 1980, pp. 169-181.

[EC82] Emerson, E. A., and Clarke, E. M., Using Branching Time Logic to Synthesize Synchronization Skeletons, Science of Computer Programming, vol. 2, pp. 241-266, 1982.

[EH82] Emerson, E. A., and Halpern, J. Y., Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. 14th Annual ACM Symp. on Theory of Computing, 1982.

[EH83] Emerson, E. A., and Halpern, J. Y., 'Sometimes' and 'Not Never' Revisited: On Branching versus Linear Time. POPL83.

[EL84] Emerson, E. A., and Lei, C. L., *Modalities for Model Checking: Branching Time Strikes Back*, to be presented at the 12th Annual ACM Symposium on Principles of Programming Languages.

[ES84] Emerson, E. A., and Sistla, A. P., *Deciding Branching Time Logic*, 16 Annual ACM Symp. on Theory of Computing, 1984.

[HR72] Hossley, R., and Rackoff, C, The Emptiness Problem For Automata on Infinite Trees, Proc. 13th IEEE Symp. Switching and Automata Theory, pp. 121-124, 1972.

[KP83] Koren, T. and Pnueli, A., There exist decidable context-free propositional dynamic logics, CMU Workshop on Logics of Programs, Springer LNCS #164, pp. 313-325, 1983.

[McN66] McNaughton, R., Testing and Generating Infinite Sequences by a Finite Automaton, Information and Control, vol. 9, 1966.

[MP79] Manna, Z., and Pnueli, A., The modal logic of programs, Proc. 6th Int. Colloquium on Automata, Languages, and Programming, Springer-Verlag Lecture Notes in Computer Science #71, pp. 385-410, 1979.

[ME74] Meyer, A. R., Weak Monadic Second Order Theory of Successor is Not Elementary Recursive, Boston Logic Colloquium, Springer-Verlag Lecture Notes in Mathematics #453, 1974.

[PN77] Pnueli, A., The Temporal Logic of Programs, 19th Annual Symp. on Foundations of Computer Science, 1977.

[RA69] Rabin, M., Decidability of Second order Theories and Automata on Infinite Trees, Trans. Amer. Math. Society, vol. 141, pp. 1-35, 1969.

[RA70] Rabin, M., Automata on Infinite Trees and the Synthesis Problem, Hebrew Univ., Tech. Report no. 37, 1970.

[ST81] Streett, R., Propositional Dynamic Logic of Looping and Converse (PhD Thesis), MIT Lab for Computer Science, TR-263, 1981. (a revised version appears in Information and Control, vol. 54, pp. 121-141, 1982.)

[Wo82] Wolper, P., A Translation from Full Branching Time Temporal Logic to One Letter Propositional Dynamic Logic with Looping, unpublished manuscript, 1982.

[VW83] Vardi, M., and Wolper, P., Yet Another Process Logic, CMU Workshop on Logics of Programs, Springer-Verlag, 1983.

[WVS83] Wolper, P., Vardi, M., and Sistla, A., Reasoning about Infinite Computations, 24th FOCS, 1983.

[VW84] Vardi, M. and Wolper, P., *Automata Theoretic Techniques for Modal Logics of Programs*, pp. 446-455, STOC84.

[VS85] Vardi, M. and Stockmeyer, L., *Improved Upper and Lower Bounds for Modal Logics of Programs*, to be presented at STOC85.

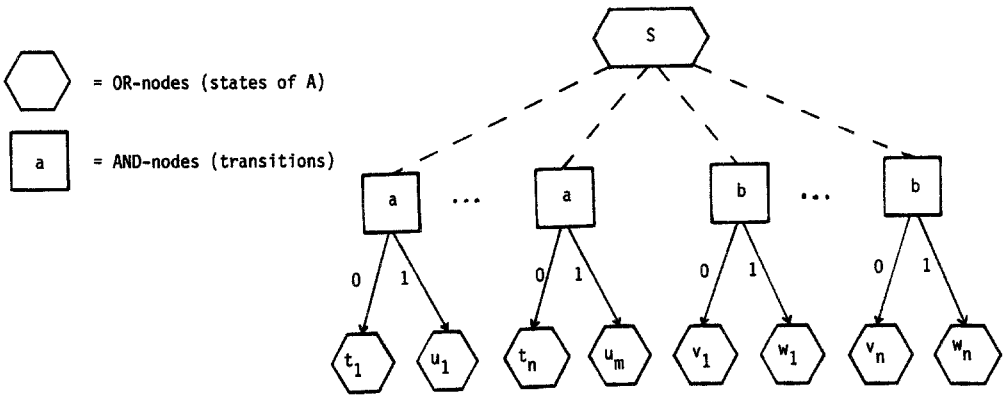
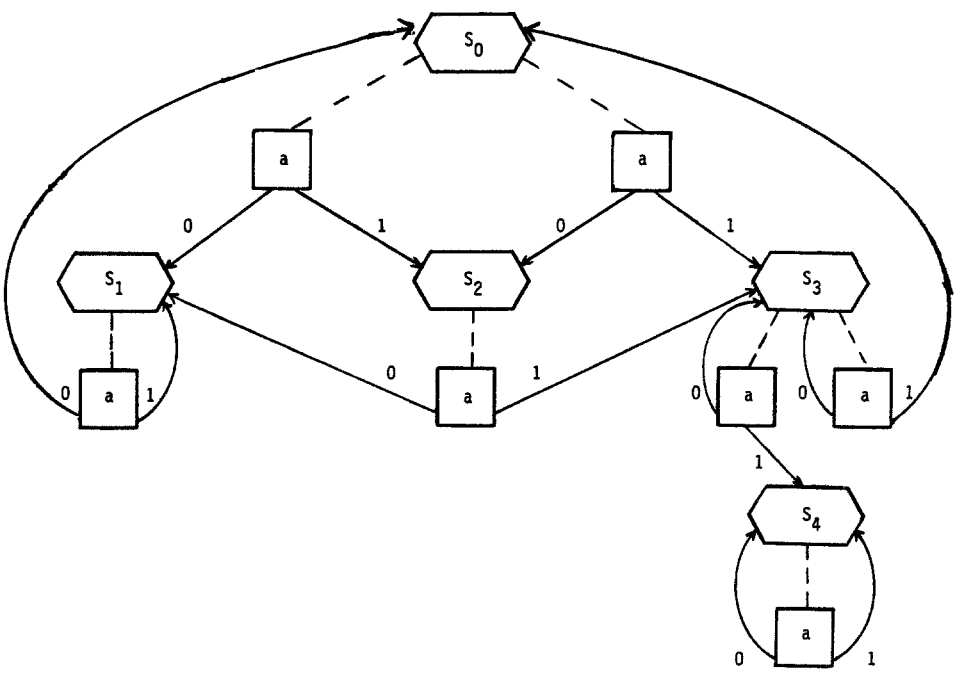


Figure 1.



$\delta(s_0,a) = \{(s_1,s_2), (s_2,s_3)\}$

$\delta(s_1,a) = \{(s_0,s_1)\}$

$\delta(s_2,a) = \{(s_1,s_3)\}$

$\delta(s_3,a) = \{(s_3,s_4), (s_3,s_0)\}$

$\delta(s_4,a) = \{(s_4,s_4)\}$

Transition Diagram for Tree Automaton with
Indicated Transition Function

Figure 2.