INFORMATION AND COMPUTATION **133**, 117–134 (1997)
ARTICLE NO. IC972624

Determinization and Memoryless Winning Strategies

Charanjit S. Jutla

T.J. Watson Research Center, IBM,
P.O. Box 704, Yorktown Heights, New York 10598

We show that Safra's determinization of ω-automata with Streett (strong fairness) acceptance condition also gives memoryless winning strategies in infinite games, for the player whose acceptance condition is the complement of the Streett condition. Both determinization and memorylessness are essential parts of known proofs of Rabin's tree automata complementation lemma. Also, from Safra's determinization construction, along with its memoryless winning strategy extension, a single exponential complementation of Streett tree automata follows. A different single exponential construction and proof first appeared in [N. Klarlund (1992), Progress measures, immediate determinacy, and a subset construction for tree automata, *in* "Proceedings, 7th IEEE Symposium on Logics in Computer Science"]. © 1997 Academic Press

1. INTRODUCTION

In 1969, Rabin [Ra69] proved that automata on finite trees are closed under complementation. This result, which is also known as the complementation lemma, was used by Rabin to show that the monadic second order theory of n successors (SnS) is decidable. Many other important decidability results were shown to be reducible to decidability of SnS. Since the models of branching time [CE81, EH82] logics are infinite trees, the theory of automata on infinite trees was also found applicable to branching time logics [ESi84, EJ88, PR89].

In this paper we show that two important lemmas which are used in the proof of the complementation lemma, and in other theorems, can be proved using almost the same argument, thus simplifying the overall theory of infinite trees (and strings).

As noted in [EJ91], almost all proofs of the lemma can be seen upon reflection to have taken the following approach, best brought out by [MS87]:

Non-deterministic tree automata are trivially alternating tree automata. Alternating tree automata are easy to complement, given that (1) *certain infinite games are determined*, i.e., at least one of the players has a winning strategy. Alternating tree automata can be translated to non-deterministic tree automata using (2) *determinization* of automata on infinite strings and (3) a *forgetful strategy* theorem.

All three of the above are fundamental theorems. Determinization of automata on infinite strings was first shown by McNaughton in [Mc66], and that result was used in [Ra69]. In 1988, Safra obtained a relatively simple single exponential determinization. Determinacy of infinite games has been studied both in the context of automata theory [BL69, Ra69, GH82, EJ91] and descriptive set theory [Ma75, Bu83]. Forgetful strategy theorems show that the memory needed for representing winning strategies is bounded (see the following paragraphs for more details). Such forgetful strategy theorems have been proved in [GH82, EJ91].

We show the surprising result that forgetful strategy theorems arise from analyzing Safra's determinization construction. In fact, we show that Safra's construction not only is a determinization construction, but simultaneously allows one to prove the forgetful strategy theorems (thus proving two of the above three key theorems).

Forgetful and Memoryless Winning Strategies

We give an informal definition of games and memoryless winning strategies. For a formal treatment, see Section 2.2. Consider a two player infinite game. A strategy of a player in a game is a function which, given the current history of the play, gives the player's next move. A strategy is a winning strategy iff regardless of how the other player plays, this strategy will enable the player to win each play in the game. A strategy is *forgetful* if the map is oblivious of the history of the play, except for a finite history of the play or a finite state condition obtained from the play. Even nicer is a strategy which is *memoryless*, i.e., the map is oblivious of the entire history of the play except the current state and the length of the play.[1] . We would like to prove that if a player has a winning strategy in a game then the player has a memoryless winning strategy.

In [EJ91], memoryless winning strategies were shown for *parity* acceptance conditions [Mo84, EJ91] which are simpler (though less succinct) equivalents of Rabin's pairs condition [Ra69]. Straightforward ranks were defined recursively for the different states in the game, and a memoryless strategy was obtained by following strategies which picked the least ranked states, in case of contention. Memoryless winning strategies for these games were also shown in [Mos91]. In [Kl92] memoryless strategies were obtained by Klarlund for Rabin's pairs condition by defining another kind of rank (called *progress measures*). Forgetful strategies obtained in [GH82] also employed a rank, defined recursively. Informally, all of the above techniques were different in spirit (from the one in this paper) because they built the ranks in a bottom-up manner.[2]

We now show that ranks on these games can be computed in a top-down (on-line) fashion, using the same construction which allows us to determinize ω-automata [Sa88, Sa92]. Essentially, one starts with a rank for the starting node of the run of the tree automaton on the input tree, and assigns ranks to the following

[1] Most earlier work (exceptions include [EJ91, Kl92]) has considered memoryless strategy to mean a map which is oblivious of the history as well as the length of the play. It turns out that our weaker definition suffices to prove the tree complementation lemma (see [J90, EJ91, Kl92], or the last section).

[2] However, see the paragraph following the next.

nodes of the run, using the rank of the current node of the run and the automaton's transition diagram. Since, determinization of ω-automata is used in any case in the complementation lemma, this result makes the lemma even more transparent.

In reality, our technique for showing memoryless winning strategies is a two stage technique (as is Klarlund's technique): a first stage in which one either assumes that a winning strategy exists, and a top-down second stage. Klarlund's technique is also a two stage technique: a bottom-up stage in which a global structure (with lot of information) is built, and a simple top-down second stage. Our technique seems more advantageous for proving the complementation lemma, as this top-down second stage is the same as the determinization construction. Moreover, (the hypothesis about) the existence of the winning strategy is proved in step 1 in any case (as described previously in the scheme for proving the complementation lemma).

In [Sa92], it is shown that ω-automata with Streett (strong fairness) acceptance condition can be determinized with a single exponential blow up. From the memoryless winning strategies we show using this determinization, along with the determinization itself, a single exponential blowup Streett tree automata complementation follows (see, e.g., [EJ91]). A different proof (and construction of a single exponential complementation) using progress measures first appeared in [Kl92].

The rest of the paper is organized as follows. We begin by defining automata on infinite strings in Section 2. Since Safra's determinization construction of Streett automata is a generalization of his construction for Buchi automata, we begin by a description of the latter construction in Section 2.1. Just as Safra's construction for Streett automata allows one to prove memoryless winning strategies for games with the complement of Streett condition as winning condition, Safra's construction for Buchi automata allows one to prove memoryless winning strategy for games with the complement of Buchi condition as the winning condition. Since the overall structure of these two proofs (i.e., for Buchi and Streett) is almost the same, we first prove memoryless winning strategies for the simpler game (i.e., with winning condition the complement of the Buchi condition) in Section 2.2. This should help the reader who wants to proceed one step at a time, or just wants to get the main idea. The more advanced reader can skip directly to Section 2.3 where Safra's generalized determinization construction is described in detail. In Section 2.4 memoryless winning strategies for the more complicated game are shown. In Section 3 we briefly describe the application of these results to tree automata complementation.

2. AUTOMATA ON INFINITE STRINGS

We begin by studying automata on infinite strings (also called ω-automata), because most constructions for automata on infinite trees use constructions for ω-automata as a tool.

Automata on infinite strings are similar to automata on finite strings, except for the fact that since there is no final state in the former automata, acceptance has to be defined in a different way.

Definition 2.0.1. A non-deterministic Buchi ω-automaton is a tuple $\mathcal{N} = (\Sigma, Q, \delta, q_0, F)$, where

Σ is the input alphabet,

Q is the set of finite states,

δ is the non-deterministic transition function, $\delta: \Sigma \times Q \to 2^Q$,

q_0 is the initial state, and

$F \subseteq Q$ is the set of green (privileged) states used in defining the Buchi acceptance condition.

The language accepted (or defined) by such an automaton is a set of infinite strings over Σ, i.e., it is a subset of Σ^ω. We will use the convention that an infinite string (or sequence) σ is $\sigma_0 \sigma_1 \sigma_2 \ldots$. A *run* ρ of $\mathcal{N}$ on a ω-string σ is an infinite sequence of states, such that

$$\rho_0 = q_0,$$
$$\rho_i \in \delta(\sigma_{i-1}, \rho_{i-1}).$$

If the range of the transition function δ is the set of singleton subsets of Q, then the automaton is called a deterministic automaton, and δ is instead defined as a function $\delta: \Sigma \times Q \to Q$.

As mentioned earlier, the main difference between automata on infinite objects and automata on finite objects is the way acceptance is defined. Under the *Buchi* acceptance condition, a run ρ is accepting if the set of states appearing infinitely often in the string ρ have a non-empty intersection with F. In other words, some state from F appears infinitely often in ρ. In the *co-Buchi* acceptance condition a run is accepting iff every state from F appears only finitely many times in ρ. Let $\mathrm{Inf}(\rho)$ be the set of states appearing infinitely often in ρ.

Rabin's pairs acceptance condition [Ra69] is defined by a set of pairs of subsets of Q, say $\{(G_1, R_1), \ldots, (G_k, R_k)\}$. A run is accepting if $\exists i$, $1 \leqslant i \leqslant k$, such that $\mathrm{Inf}(\rho) \cap G_i \neq \phi$, and $\mathrm{Inf}(\rho) \cap R_i = \phi$. *Streett's complemented-pairs* acceptance condition [St81] is the complement of the above. A run is accepting if $\forall i$, $1 \leqslant i \leqslant k$, $\mathrm{Inf}(\rho) \cap G_i \neq \phi \Rightarrow \mathrm{Inf}(\rho) \cap R_i \neq \phi$. Another acceptance condition is the *parity* acceptance condition [Mo84, EJ91], in which the ordering of the pairs is important. A run is accepting if $\exists i$, $1 \leqslant i \leqslant k$, such that $\mathrm{Inf}(\rho) \cap G_i \neq \phi$, and $\forall j$, $j \geqslant i$, $\mathrm{Inf}(\rho) \cap R_j = \phi$. Automata with the above acceptance conditions are respectively called pairs automata, complemented pairs automata, and parity automata.

The language accepted by an automaton is the set of infinite strings, for which there is an accepting run of the automaton.

Theorem 2.0.1. *Given a non-deterministic Buchi automaton $\mathcal{N}$, there exists a deterministic pairs automaton $\mathcal{D}$ which accepts the same language.*

The above determinization theorem was first proved by McNaughton, who gave a determinization construction [Mc66] which causes a double exponential blowup in the size of the automata. Recently, Safra [Sa88] has given a new construction with only a single exponential blowup in size. Later, Safra [Sa92] gave another (generalized) construction which determinized a complemented pairs non-deterministic automaton to a deterministic automaton with only a single exponential blowup.

2.1. Safra's Determinization Construction

To understand the connection between determinization and memoryless winning strategies, we should first grasp the intricacies involved in determinizing an ω-automaton.

A non-deterministic automaton has several (possibly infinite) runs on a single input string. The idea of determinization is to keep track of all these runs in a single run of the new (deterministic) automaton. Since there could be infinitely many runs, a state may have to keep arbitrarily large amount of information, resulting in infinitely many states. The key idea is to keep bounded amount of information, and still manage to decide whether there is an accepting run in the non-deterministic automaton or not.

We first describe a plausible idea, which unfortunately does not seem to work. To facilitate the presentation, we will say that a Buchi automaton flashes a green light whenever its run is in a state in F. Similarly, a pairs automaton flashes $green_i$ or red_i whenever it is in a state in the corresponding set. Let the non-deterministic automaton $\mathscr{N}$ have n states. We will try to build a deterministic pairs automaton $\mathscr{D}$. Since $\mathscr{N}$ has n states, it is possible that we need to keep track of only one of the many runs whose current state is the same. This way, at any moment we need to keep track of (in $\mathscr{D}$) at most n runs. We could associate a pair each in the accepting condition to these runs (thus requiring n pairs). Whenever a run in $\mathscr{N}$ flashes a green light, the automaton $\mathscr{D}$ flashes the green light of the pair associated with it. Whenever two runs have the same current state, one of the runs is discarded and the red light of the pair associated with it is flashed. The main problem is to decide which of the two runs should be discarded. Also, when a run splits into two runs, which run should retain the pair associated with the parent run?

Safra's construction has a slightly different solution. In this construction a single pair could be associated with a set of runs. When a run splits, the split runs are kept as a set, say Π, and all of them share the pair associated with the parent run. If all of them flash green in $\mathscr{N}$ since a previous checkpoint, then $\mathscr{D}$ flashes green corresponding to the pair associated with them, and a new checkpoint is declared. However, to take care of the possibility that only a subset of this set flashes green infinitely often, whenever a run ρ in this set Π flashes green, recursively it is assigned a new pair, and the run ρ is itself made a set $\Pi' = \{\rho\}$, and Π' is made a member of Π. The transitive closure of the member relation is called descendant. When all leaf descendants of a set have flashed green in $\mathscr{N}$ since the previous checkpoint, all descendant sets are deleted, their corresponding red lights flashed in $\mathscr{D}$ and all the leaf descendants are made members of the set Π.

These sets are also timestamped by their time of creation. Whenever there are two runs with the same current state, the run in the set with a later timestamp is discarded, unless one set is a descendant of other, in which case the run is retained in the descendant set, and discarded from the ancestor.

We formalize this construction below. Safra proved that the run in $\mathscr{D}$ is accepting iff there is a run in $\mathscr{N}$ which is accepting. We strengthen this result by showing that if the run in $\mathscr{D}$ is accepting then there is an accepting run in $\mathscr{N}$ which is never discarded in the above construction. These undiscarded runs are important as they give us history free winning strategies, as we shall see. Readers who are not interested in the formal proof of Safra's construction may skip the rest of this section.

Let $\mathscr{N}$ be a non-deterministic Buchi automaton as above with $|Q| = n$. We construct a deterministic pairs automaton $\mathscr{D} = (\Sigma, \tilde{Q}, \tilde{\delta}, \tilde{q}_0, P)$, where P is a set of pairs, which accepts the same language.

Let $V = [1, ..., 2n]$ be a set of *names* which will be used to identity the pairs in $\mathscr{D}$.

A labeled tree T is a structure $T = (N, r, p: N \to N, \psi, S: N \to 2^Q, \text{name}: N \to V, G \subseteq V, R \subseteq V)$, where

N is a set of nodes,

r being the root node,

$p: N \to N$ is the parenthood function defined over $N - \{r\}$, and defining for each $v \in N - \{r\}$, its parent $p(v) \in N$,

ψ is a partial order defining "older than" on siblings (i.e., children of the same node),

$S: N \to 2^Q$ is a labelling of the nodes with subsets of Q. $S(v)$ will be called the label of $v \in N$,

name: $N \to V$ is a one to one map from the nodes to the set of names.

The states of $\mathscr{D}$ will be *labeled trees* in which the labels enjoy the following properties:

1. The union of the labels of the children of a node v is a proper subset of the label of v.

2. The labels of two nodes which are not ancestral are disjoint.

The following lemma follows easily (see [Sa91]).

LEMMA 2.1.1. *The number of nodes in a state of $\mathscr{D}$ is at most n.*

Safra's Construction. The initial $\tilde{q}_0$ is the tree with a single node labeled with the initial states of $\mathscr{N}$.

The Transition Function. For each $\mathscr{D}$-state $\tilde{q} \in \tilde{Q}$ and a letter $a \in \Sigma$, $\tilde{\delta}(\tilde{q}, a)$ is the result of applying the following sequence of operations to $\tilde{q}$:

1. For every node v with label Q', replace Q' by $\delta(Q', a)$, and set G and R to empty sets.

2. For every node with label Q', such that $Q' \cap F \neq \phi$, create a new node v' which becomes the youngest child of v. Set its label to be $Q' \cap F$. Assign this node an unused name. That such an unused name is available follows by Lemma 2.1.1.

3. For every node v with label Q' and state $q \in Q'$, such that q also belongs to the label of an older sibling v' of v, remove q from Q' and all its descendants.

4. Remove all nodes with empty labels. In particular, the names associated with such nodes are now unused. Add the names of these nodes to R.

5. As a result of the previous steps, for every node v whose label is equal to the union of the labels of its children, remove all descendants of v. Add the names of the removed nodes to R. Add the name of v, name(v) to G.

A state $q \in Q$ is *specific* to a node v if q is in the label of v, and not in the label of any descendant of v (recall that the labels of nodes which are not ancestral are disjoint). It is easy to see that a state can be specific to at most one node in a $\mathscr{D}$ state.

Acceptance Condition. A state is in G_i, if $i \in G$, and a state is in R_i if i is in R.

We skip the proof of Theorem 2.0.1. It can be found in [Sa88]. The argument is also contained in the proof of Theorem 2.2.2 (see next section).

2.2. *Memoryless Winning Strategies*

We start this section by defining two person infinite games, strategies and winning strategies.

DEFINITION 2.2.1. An infinite binary tree T over a countable alphabet X, is a function T: $\{0, 1\}^* \rightarrow X$. The tree is *total* iff T is defined for each element in $\{0, 1\}^*$.

Although we are considering binary trees here, this section can be done similarly for finitely branching trees. Since X can in general be countable it still allows for each node of the tree to be labeled with a unique element of X.

DEFINITION 2.2.2. A two player (players I and II), perfect information, no draw *infinite game* $G(T, W_{\mathrm{I}})$ over a total infinite binary tree T, and alphabet X is defined by the winning set $W_{\mathrm{I}} \subseteq X^\omega$ of player I. A play of the game s is an infinite sequence over X, i.e., $s \in X^\omega$. A play s is valid iff it is an infinite path in the tree T. From now on we will only consider valid plays. A play s is winning for player I iff $s \in W_{\mathrm{I}}$, otherwise it is winning for player II.

A *strategy* ψ for player I is a function ψ: $(\{0, 1\} \cdot \{0, 1\})^* \rightarrow \{0, 1\}$. Essentially, the strategy picks a successor for each even depth node in the tree T. Similarly a strategy for player II can be defined which picks a successor from each odd depth node. A strategy ψ defines an infinite tree T_ψ in the game $G(T, W_{\mathrm{I}})$ as follows:

$T_\psi(\varepsilon) = T(\varepsilon)$, where ε is the empty string. If for $a \in (\{0, 1\} \cdot \{0, 1\})^*$, $T_\psi(a)$ is defined then,

$T_\psi(a \cdot \psi(a)) = T(a \cdot \psi(a))$, and

$T_\psi(a \cdot \psi(a) \cdot b) = T(a \cdot \psi(a) \cdot b)$, for $b \in \{0, 1\}$.

Essentially, T_ψ is the subtree of T obtained by employing the strategy ψ on T. The strategy ψ (for player I) picks a successor from each even depth node (also called OR nodes), and picks all successors from each odd depth node (also called AND nodes).

A strategy ψ for player I is a *winning strategy* for player I iff all infinite paths in T_ψ are in W_{I}.

DEFINITION 2.2.3. A *co-Buchi game* is an infinite game $G(T, W_{\mathrm{I}})$, such that some subset of alphabet X is colored green, and W_{I} is the set of all infinite strings over X which satisfy the co-Buchi acceptance condition (see Definition 2.0.1—i.e., all infinite strings such that green colored elements of X appear only finitely many times).

Now we introduce a notion of state in the trees defining the above games.

DEFINITION 2.2.4. Two nodes of even depth in a tree are in the same *pseudo-state* iff the subtrees starting at the nodes are identical (w.r.t. the coloring of the nodes). A strategy ψ for player I is memoryless iff

$\forall i \geq 0$, $\forall a, b \in (\{0, 1\} \cdot \{0, 1\})^i$, if a and b have the same pseudo-state then $\psi(a) = \psi(b)$.

Note that we require that a and b be at the same depth (moreover, the depth is even). Most earlier work (exceptions include [EJ91, Kl92]) has considered memorylessness (or forgetfulness [GH82]) to mean that the strategy is independent of the depth as well. But as we show in the last section our "weaker" definition suffices for proving the tree complementation lemma.

The theorem we are going to prove in this section states that for the game described above, if player I has a winning strategy then player I has a memoryless winning strategy. Let ψ be a winning strategy, and let x and y be two nodes with the same pseudo-state. If we can show that we can pick one of the two nodes (say x), such that if in a new strategy we emulate at y the strategy ψ at x, we still get a winning strategy, we are closer to proving a memoryless winning strategy.

This may sound familiar, given that in Safra's determinization construction, whenever there were two runs in the same state, we managed to discard the continuation of one of the runs, and continued it instead along the other run. The nondeterministic runs of an automaton form a tree, and Safra's construction managed to assign a rank to nodes in this tree, so that we could use this rank to discard one of the two runs in the same state. We will now show that, if we employ Safra's construction on the winning strategy ψ (which is a tree), then we can use the same rank to decide which of the two nodes x and y to pick, so that the resulting strategy is still winning for player I. Recall that this rank is a combination of the sibling and parenthood relationship.

What is remarkable about Safra's construction is that it assigns ranks to nodes based solely on their past history. In this respect it may be termed an *on-line* method of assigning ranks, as opposed to methods which look at the whole tree (i.e., also at the future), and then assign ranks to the nodes (however, see the third from last paragraph of the Introduction).

We begin by strengthening Safra's Theorem 2.0.1. Also, as opposed to Safra's construction, where we assumed that the number of states is finite, we will now work under the assumption that the number of states is countable. Under this assumption, Safra's construction does not help in proving Theorem 2.0.1, since the depth of the labeled trees used as states in the previous section is no longer bounded (see Lemma 2.1.1 of previous section). However, for the purpose of showing memoryless winning strategies, Safra's construction still works.

DEFINITION 2.2.5. Let the states of $\mathcal{N}$ be numbered, and hence well-ordered by $<$. Let $\mathcal{D}$ be the deterministic automaton obtained by Safra's construction from $\mathcal{N}$. Let π be the run of $\mathcal{D}$ on an input string σ.

An $\mathcal{N}$-run ρ' is said to have *merged at level k* with another $\mathcal{N}$-run ρ iff:

1. $\rho'_k = \rho_k$ and ρ'_k is removed from the $\mathcal{D}$-state π_k in deference to ρ_k in step 3 of Safra's construction. Or,

2. If (1) does not hold, then if $\rho'_k = \rho_k$, $\rho'_{k-1} \neq \rho_{k-1}$, and the deepest node in which ρ'_{k-1} occurs is an ancestor of the deepest node in which ρ_{k-1} occurs. Or,

3. If (1) and (2) do not hold, then in the first step of the construction, replacing Q' by $\delta(Q', \sigma_{k-1})$, the two runs ρ and ρ' are in Q' at level $k-1$, i.e., ρ_{k-1}, $\rho'_{k-1} \in Q'$, $\rho'_{k-1} < \rho_{k-1}$, and $\rho_k = \rho'_k = \delta(\rho_{k-1}, \sigma_{k-1}) = \delta(\rho'_{k-1}, \sigma_{k-1})$.

Since $<$ is well founded, and the depth of the tree π_k is bounded by k, "merges at level k" is a well founded relation.

An $\mathcal{N}$-run ρ is called *original* in the $\mathcal{D}$-run π iff for all $k > 0$, ρ does not merge-at-level-k with any other run.

LEMMA 2.2.1. For all $k > 0$, if an $\mathcal{N}$-run ρ merges-at-level-k with another run, then ρ merges-at-level-k with a run ρ', such that for $j \leqslant k$, ρ' has not merged-at-level-j with any run.

Proof. For $k = 1$, we let ρ' be a minimal run w.r.t. merges-at-level-1, among the runs with which ρ merges-at-level-1.

Suppose ρ merges-at-level-k with a minimal (w.r.t. merges-at-level-k) run ρ''. If ρ'' has merged-at-level-j, for $j < k$, let *jmax* be the largest such j. Then, by induction, ρ'' has merged-at-level-*jmax* with a run ρ', such that ρ' has never merged-at-level-j for all $j \leqslant jmax$. Clearly, ρ' has never merged-at-level-j, for all $j \leqslant k$ as well. Since $\rho''_k = \rho'_k$, ρ merges-at-level-k with ρ'. ∎

THEOREM 2.2.2. *On an input string σ, all original $\mathcal{N}$-runs satisfy co-Buchi condition iff all $\mathcal{N}$-runs satisfy the co-Buchi condition.*

Proof. The if part is trivial. For the only if part, suppose some $\mathcal{N}$-run ρ satisfies Buchi condition instead. Let π be the run in the deterministic automaton. Recall that the states of π are labelled trees. If a node is present in the state π_i of π, and this node is not deleted in any π_j, $j \geqslant i$, then we say that the node is eventually forever in π. Clearly, in such a case a name is permanently assigned to this node.

We will first show:

(1) There is an infinite sequence of names $v = v_0 v_1 \cdots v_i \cdots$, an infinite sequence of times (or levels) $t_0, ..., t_i, ...,$ and an infinite sequence of nodes

$n = n_0 n_1 \cdots n_i \cdots$ such that for all i, eventually n_i is assigned name v_i forever, and n_i is either a child of n_{i-1} or same as n_{i-1} (for $i > 0$). Moreover,

(2) let P be the set of $\mathcal{N}$-runs which for all i are eventually (from time $\leqslant t_i$) forever in n_i. Then ρ is in P. Also,

(3) for all $i \geqslant 0$ $\exists j > i$, between t_i and t_j, every run in P either encounters a green state, or merges with another run which encounters a green state.

By definition, ρ is in the root node of the states in π, and the root node n_0 is assigned a fixed name, say v_0 forever. Let $t_0 = 0$. Assume we have demonstrated subsequences $v_0 \cdots v_i$, $t_0 \cdots t_i$, and $n_0 \cdots n_i$ with the above properties. Thus, at time t_i, ρ is in n_i, and ρ is in n_i for all times $\geqslant t_i$.

Since, ρ encounters green states infinitely often, n_i has children infinitely often. Now, either all children of n_i are deleted, or else some child m of n_i is eternal (i.e., is assigned a fixed name and is never deleted). In the former case, let $v_{i+1} = v_i$, $n_{i+1} = n_i$, and let t_{i+1} be the first instance since t_i when step (5) of Safra's construction is employed on n_i (which is the only way all children of a node are deleted).

In the latter case, we claim that ρ is in one of the eternal children of n_i forever. After the addition of m, whenever ρ is in a green state, it is always in some child of n_i. The run ρ may keep moving to older siblings, but since the older-sibling relationship is well founded (the countable children of a node are numbered by the order of their creation), ρ is eventually always in the same child of n_i. Let this child be n_{i+1}, and v_{i+1} be the name assigned to n_{i+1}. Let t_{i+1} be the first time greater than t_i, when ρ is in n_{i+1}.

This proves the first two claims.

To prove claim (3), after t_i let t' be the first time instance when some n_k $(k > i)$ in the sequence n is added to the state π, if such a t' exists. Otherwise, let t' be the second time after t_i when all children of some n_i in n are deleted. Then there is a $j > i$ such that $t_j \geqslant t'$. Since the runs in P are a subset of the runs in n_i for all i, all the runs in P either encounter a green state, or merge with a run which encounters a green state between time instances t_i and t_j.

Next, we show that there is an original $\mathcal{N}$-run χ which is in P. Finally we show that χ satisfies the Buchi condition, thus leading to a contradiction.

Now, either (a) there is an original run in P, which never merges, or (b) all runs in P merge by level k, or (c) for all k, there is a run in P which has never merged till level k. In case (a) we are done. Case (b) is not possible, for if it is, let $j \leqslant k$ be the largest such that some $\chi' \in P$ merges with another run, say χ (which by Lemma 2.2.1 has never merged till level j). Since χ' is in P forever, χ is also in P, and moreover, χ has never merged till level k.

For case (c), we construct a prefix forest (with finite number of roots) of all the runs in P. Nodes at level k are prefixes of runs which have never merged by level k. The parent of a node is its immediate prefix. There are infinitely many nodes, as for all k we have runs which have never merged by level k. The finite number of roots are the runs which existed in v_0 at level 0. Also the tree is finitely branching, as we are only considering finitely branching transition functions. Thus, by Konig's lemma, there is an infinite path in this tree which never merges, giving us an original $\mathcal{N}$-run.

Finally, the original $\mathcal{N}$-run obtained above satisfies Buchi condition because, an original run never merges, and hence it encounters green states infinitely often (see (3) above). ▮

THEOREM 2.2.3. *If player* I *has a winning strategy in a co-Buchi game* $G(T, W_{\mathrm{I}})$, *then player* I *has a memoryless winning strategy in* $G(T, W_{\mathrm{I}})$.

Proof. Consider the winning strategy ψ of player I. By Definition 2.2.2, T_ψ is a tree over X such that all paths satisfy co-Buchi condition. The nodes of T_ψ can be viewed as states of an ω-automaton, each node being given a distinct state, unless two nodes have the same pseudo-state (as in Definition 2.2.4), in which case they are given the same state. A state is in F iff the corresponding node is colored green. The states having been defined, T_ψ can then be viewed as the transition diagram of a single alphabet (say $\{a\}$) non-deterministic automaton; i.e., each transition is on the same element a. Each odd-depth state in T_ψ has two successors, and each even-depth state has one successor, unless a different successor is picked by ψ at two different nodes in T_ψ with the same pseudo-state. In this case, the even-depth state has two successors as well.

If we employ Safra's construction on this automaton, then the original runs (Definition 2.2.5) in this construction are the paths in the tree T_ψ. We obtain a memoryless strategy ψ' as follows. For every even-depth node u of depth k, such that a run (having the path upto this node as a prefix) merges at level k with another run whose k depth prefix ends in v (which w.l.o.g. by Lemma 2.2.1 has not merged with any other run), define in ψ' the successor of the node u to be the successor picked by ψ at v. By definition ψ' is memoryless.

The paths of $T_{\psi'}$ are also runs of the automaton defined above. Thus, all paths of $T_{\psi'}$ satisfy co-Buchi condition by Theorem 2.2.2, since the paths of T_ψ contain all the original runs of the automaton. ▮

2.3. Safra's Generalized Determinization Construction

Safra [Sa91] generalized his construction of determinizing a Buchi automaton to that of determinizing a non-deterministic Streett automaton (with only a single exponential blowup in size).

We will say that a run of the non-deterministic automaton $\mathcal{N}$ flashes $Green_i$ (resp. Red_i) whenever the run is in a state in G_i (resp. R_i). Recall that in Safra's construction for Buchi automaton, we maintained a pair in the deterministic automaton for a set of runs in the non-deterministic automaton. When a run split, we kept the split runs as a set. However, when a subset of this set flashed green we spawned a new subset of runs (essentially the ones which saw the green state) as a child of the original set. The main idea there was that only this subset maybe the one which sees green infinitely often.

Now, we have the additional problem of deciding when to spawn a new child, as there is not just one green light in the non-deterministic automaton. However, note that a run in the Streett automaton is accepting iff for a subset of indices J, Red_j flashes infinitely often for all j in J, and for no i in $[1, ..., k] - J$, $Green_i$ flashes infinitely often.

Thus, with each subset of states we maintain, we attach a potential index set J. Whenever, a state in this set flashes a *Green* of index i not in J, we delete that node from this set and return it to its parent with index set $J \cup \{i\}$. Also, we maintain with the set a current index from J (and we keep decreasing the index as the current Red_j is seen). This may require spawning child subsets as only a subset of states may encounter the current Red_j (just as in the previous construction). When all the Red_j from J are seen, and this node has no children, we flash the corresponding green in the deterministic automaton. However, if a subset of states encounter a Red_j such that j is not the current index i, we spawn a child with index set $J - \{i\}$.

The merging of runs is same as before, except that we now give precedence to a run which has a lower current index. We formalize the construction below.

THEOREM 2.3.1. *Given a non-deterministic Streett automaton* $\mathscr{N}$ *as in Definition* 2.0.1 *with* k *pairs, and* $|Q| = n$, *there exists a deterministic pairs automaton* $\mathscr{D} = (\Sigma, \tilde{Q}, \tilde{\delta}, \tilde{q}_0, P)$, *where* P *is a set of pairs, which accepts the same language.*

Of course, given Theorem 2.0.1 the above theorem has a simple proof because non-deterministic Streett automata can easily be converted to equivalent non-deterministic Buchi automata. However, Safra's generalized construction gives a single exponential blowup conversion, and memoryless winning strategies as we show in the next section.

Let $V = [1, ..., 2n * (k+1)]$ be a set of *names* which will be used to identify the pairs in $\mathscr{D}$.

A labeled tree T is a structure $T = (N, r, p: N \to N, \chi, S: N \to 2^Q, \text{index}: N \to [0, ..., k], \text{name}: N \to V, G \subseteq V, R \subseteq V)$, where

N is a set of nodes,

r is the root node,

$p: N \to N$ is the parenthood function defined over $N - \{r\}$, and defining for each $v \in N - \{r\}$ its parent $p(v) \in N$,

χ is a partial order defining "older than" on siblings (i.e., children of the same node),

$S: N \to 2^Q$ is a labelling of the nodes with subsets of Q. $S(v)$ will be called the label of $v \in N$,

index: $N \to [0, ..., k]$ is the current index,

name: $N \to V$ is a one to one map from the nodes to the set of names.

The states of $\mathscr{D}$ will be *labelled trees* in which the labels and indices enjoy the following properties:

1. The union of the labels of the children of a node v is a subset of the label of v. If all the children have index 0, then the union of the labels of the children is a proper subset of the label of v.

2. The labels of two nodes which are not ancestral are disjoint.

3. The index of a node is different from the index of all its ancestors, unless the index is 0.

LEMMA 2.3.2. *The number of nodes in a state of $\mathscr{D}$ is at most $n * (k+1)$.*

For a proof of this lemma, see [Sa92].

Safra's Generalized Construction. The initial state $\tilde{q}_0$ is the tree with just the root node, indexed zero, and labeled with the initial states of $\mathscr{N}$. The root is given a name from V.

The Transition Function. For each $\mathscr{D}$-state $\tilde{q} \in \tilde{Q}$ and a letter $a \in \Sigma$, $\tilde{\delta}(\tilde{q}, a)$ is the result of applying the following sequence of operations to $\tilde{q}$:

1. For every node v with label Q', replace Q' by $\delta(Q', a)$, and set G and R to empty sets.

2. For every $j > 0$, for every state $s \in Q \cap G_j$, if s is in a node with index j, remove s from all descendants of this node.

3. For every node v with label Q', and index $j > 0$, such that $Q'' = Q' \cap R_j$ is non-empty, remove states in Q'' from v and all its descendants. Make a new node with label Q'', and index the largest natural number (i.e., $\geqslant 0$) less than j, which is not an index of an ancestor of v. If no such number exists then make the index zero. Also, give this node an unused name from V. Make this new node the youngest child of the parent of v.

4. For every node v with label Q' and state $q \in Q'$, such that q also belongs to the label of a sibling v' of v with lower index, remove q from Q' and labels of all descendants of v.

5. For every node v with label Q' and state $q \in Q'$, such that q also belongs to the label of an older sibling v' of v, remove q from Q' and labels of all descendants of v.

6. Remove all nodes with empty labels. In particular, the names associated with such nodes are now unused. Add the names of these nodes to R.

7. As a result of the previous steps, for every node v whose label is equal to the union of the labels of its children, and all of whose children are indexed zero, remove all descendants of v. Add the names of the removed nodes to R. Add the name of v to G.

8. For each leaf node v, such that for all i, $0 < i \leqslant k$, i is the index of v or one of its ancestors, add the name of v, name(v) to G.

9. For every leaf node v, such that there is a number i, $0 < i \leqslant k$, such that i is neither the index of v, nor the index of any ancestor of v, make a new node with label same as that of v, a child of v. Index the new node with the largest such i. Repeat this step till there are no such leaf nodes. (*Remark.* This step could alternatively be done in a lazy manner, but we prefer not to do so.)

Acceptance Condition. A state is in G_i if $i \in G$, and a state is in R_i if i is in R.

We skip the proof of Theorem 2.3.1 as the argument is contained in the proof of Theorem 2.4.2. Also, see [Sa92], where it was originally proved.

2.4. Memoryless Winning Strategies for Pairs Games

In Section 2.2 we proved memoryless winning strategy for the co-Buchi game, using Safra's construction for determinizing Buchi Automata. In this case we will use Safra's generalized construction for determinizing complemented-pairs (Streett) automata. Since the generalized construction has the same overall structure as the original construction, the proof of memoryless winning strategy can be expected to be almost the same as in Section 2.2. In this section we prove the counterpart of Theorem 2.2.2.

As before, we begin by strengthening Safra's Theorem 2.3.1. Also, as opposed to Safra's generalized construction, where we assumed that the number of states is finite, we will now work under the assumption that the number of states are countable. Under this assumption, Safra's construction does not help in proving Theorem 2.3.1, since the depth of the labeled trees used as states in the previous section is no longer bounded (see Lemma 2.3.2 of previous section). However, for the purpose of showing memoryless winning strategies, Safra's generalized construction still works.

DEFINITION 2.4.1. Let the states of $\mathscr{N}$ be numbered, and hence well-ordered by $<$. Let $\mathscr{D}$ be the deterministic automaton obtained by Safra's generalized construction from $\mathscr{N}$. Let π be the run of $\mathscr{D}$ on an input string σ.

An $\mathscr{N}$-run ρ' is said to have *merged at level k* with another $\mathscr{N}$-run ρ iff:

1. $\rho'_k = \rho_k$, and ρ'_k is removed from the $\mathscr{D}$-state π_k in deference to ρ_k in step 4 of Safra's generalized construction. Or,

2. If (1) does not hold, then $\rho'_k = \rho_k$, and ρ'_k is removed from the $\mathscr{D}$-state π_k in deference to ρ_k in step 5 of Safra's generalized construction. Or,

3. If (1) and (2) do not hold, then if $\rho'_k = \rho_k$, and $\rho'_{k-1} \neq \rho_{k-1}$, and the deepest node in which ρ'_{k-1} occurs is an ancestor of the deepest node in which ρ_{k-1} occurs. Or,

4. If (1), (2), and (3) do not hold, then in the first step of the construction, replacing Q' by $\delta(Q', \sigma_{k-1})$, the two runs ρ and ρ' are in Q' at level $k-1$; i.e., $\rho_{k-1}, \rho'_{k-1} \in Q'$, and $\rho'_{k-1} < \rho_{k-1}$, and $\rho_k = \rho'_k = \delta(\rho_{k-1}, \sigma_{k-1}) = \delta(\rho'_{k-1}, \sigma_{k-1})$.

Since $<$ is well founded, and the depth of the tree π_k is bounded by k, "merges at level k" is a well-founded relation.

An $\mathscr{N}$-run ρ is called *original* in the $\mathscr{D}$-run π iff for all $k > 0$, ρ does not merge-at-level-k with any other run.

LEMMA 2.4.1. *For all $k > 0$, if an $\mathscr{N}$-run ρ merges-at-level-k with another run, then ρ merges-at-level-k with a run ρ', such that for $j \leqslant k$, ρ' has not merged-at-level-j with any run.*

Proof. Same as proof of Lemma 2.2.1. ∎

THEOREM 2.4.2. *On an input string σ, all original $\mathscr{N}$-runs satisfy pairs condition iff all $\mathscr{N}$-runs satisfy pairs condition.*

Proof. The if part is trivial. For the only if part, suppose some $\mathcal{N}$-run ρ satisfies complemented-pairs (Streett) condition instead. Let π be the run in the deterministic automaton. Recall that the states of π are labeled trees. If a node is present in the state π_i of π, and this node is not deleted in any π_j, $j \geqslant i$, then we say that the node is eventually forever in π. Clearly, in such a case a name is permanently assigned to this node.

Let M be the set of indices (possibly empty) such that for all m in M, ρ encounters states in R_m infinitely often, and for all l in $[1, ..., k] - M$, ρ encounters states from G_l only finitely often.

We will first show:

(1) There is an infinite sequence of names $v = v_0 v_1 \cdots v_i \cdots$, an infinite sequence of times (or levels) $t_0, ..., t_i, ...$, and an infinite sequence of nodes $n = n_0 n_1 \cdots n_i \cdots$ such that for all i, eventually n_i is assigned name v_i forever, and n_i is either a child of n_{i-1} or same as n_{i-1} (for $i > 0$). Moreover,

(2) Let P be the set of $\mathcal{N}$-runs which for all i are eventually (from time $\leqslant t_i$) forever in n_i. Then ρ is in P. Also,

(3) for all $i \geqslant 0$ $\exists j > i$, between t_i and t_j, every run in P either encounters a Red_m state for each $m \in M$ and no $Green_l$ for $l \notin M$, or it merges with such a run.

By definition, ρ is in the root node of the states in π, and the root node n_0 is assigned a fixed name, say v_0 forever. Let $t_0 = 0$. Assume we have demonstrated subsequences $v_0 \cdots v_i$, $t_0 \cdots t_i$, and $n_0 \cdots n_i$ with the above properties. Thus, at time t_i, ρ is in n_i, and ρ is in n_i for all times $\geqslant t_i$.

Suppose there is an l, $0 < l \leqslant k$, such that l is not the index of n_i or any of its ancestors. Thus, infinitely often, n_i has a child with ρ in it (by step 9). Also, no element from M can be the index of n_i or any of its ancestors. For otherwise, if $m \in M$ was the index of some ancestor of n_i (or of n_i), then by step 3, ρ would be eventually removed from n_i. Also, since for all $l \in [1, ..., k] - M$, G_l is encountered in ρ finitely often, eventually no such state from G_l would be encountered. Thus by the last two statements, eventually, step 2 cannot remove ρ from a child of n_i. After this, whenever ρ is in some child of n_i, it will always be in some child of n_i (unless deleted by step 7). The index of the child of n_i in which ρ is, may decrease by almost $k+1$ (either by step 3 or 4). After that, ρ may move to older siblings (which are only finitely many)—by step 5.

Thus, eventually ρ is permanently in some child n_{i+1} of n_i, unless this child is deleted by step 7 (which is the only other way of deleting ρ from some child of n_i). In the latter case, let $v_{i+1} = v_i$, $n_{i+1} = n_i$, and let t_{i+1} be the first instance since t_i, when step (7) of the construction is employed on n_i. In the former case let v_{i+1} be the name assigned to n_{i+1}. Let t_{i+1} be the first time greater than t_i, when ρ is in n_{i+1}.

If for all l, $0 < l \leqslant k$, l is the index of n_i or one of its ancestors, then since ρ is eventually forever in n_i, M must be empty. Let n_{i+1} be n_i, and v_{i+1} be its name, and t_{i+1} be a time instance after t_i.

To prove claim (3), after t_i let t' be the first time instance, when some n_k $(k > i)$ in the sequence n, is added to the state π, if such a t' exists. Else, let t' be the second time after t_i, when all children of some n_j in n are deleted, if such a t' exists. Otherwise, M must be empty, and there is a n_j in n which is forever a leaf node. In such a case, let t' be a time after t_i such that n_j exists in π.

Then, there is a $j > i$, such that $t_j \geqslant t'$. Since, the runs in P are a subset of the runs in $n_{i'}$ for all i', all the runs in P either encounter a Red_m state for each $m \in M$ and no $Green_l$ for $l \in [1, ..., k] - M$, or merge with a such a run, between time instances t_i and t_j.

The rest of the proof is the same as the proof of Theorem 2.2.2, with the Buchi condition replaced with the Streett condition. ▮

Using the construction in the previous section, we now prove that games in which player I's winning condition is a pairs condition have the property that if player I has a winning strategy then player I has a memoryless winning strategy.

DEFINITION 2.4.3. Let I be a finite set of indices, say $[1, ..., k]$. Suppose there is a coloring on the alphabet X, i.e., each element of X is colored with colors $Green_i$, Red_i, where $i \in I$. It is convenient to define each $Green_i$ (and Red_i) to be a subset of X. We say, $x \in X$ is colored $Green_i$ (resp. Red_i) if $x \in Green_i$ (resp. $x \in Red_i$). A *pairs game* is an infinite game $G(T, W_{\mathrm{I}})$, such that the alphabet X is colored as in this definition, and W_{I} is the set of all infinite strings over X which satisfy the pairs acceptance condition (see Definition 2.0.1).

The definition of memoryless winning strategies is given in Definition 2.2.4.

THEOREM 2.4.3. *If player* I *has a winning strategy in a pairs game* $G(T, W_{\mathrm{I}})$, *then player* I *has a memoryless winning strategy in* $G(T, W_{\mathrm{I}})$.

Proof. Consider the winning strategy ψ of player I. By Definition 2.2.2, T_ψ is a tree over X such that all paths satisfy pairs acceptance condition. The nodes of T_ψ can be viewed as states of an ω-automaton, each node being given a distinct state, unless two nodes have the same pseudo-state (as in Definition 2.2.4) in which case they are given the same state. Each state is in $Green_i$ (or Red_i) iff the corresponding node is in $Green_i$ (resp. Red_i).

The rest of the proof is the same as the proof of Theorem 2.2.3 except that here we employ Safra's generalized construction and Theorem 2.4.2 instead of Safra's construction for Buchi automata and Theorem 2.2.2. ▮

3. TREE AUTOMATA COMPLEMENTATION

In this section we briefly describe how Safra's determinization of ω-automata and memoryless winning strategy theorems of the previous sections can be used to complement tree automata. A detailed treatment is beyond the scope of this paper (see [J90] for details).

A non-deterministic tree automaton is trivially an alternating tree automaton [MS87]. Moreover, Muller and Schupp have shown [MS87] that, using the determinacy of infinite games [Ma75, GH82, EJ91], it is easy to complement an alternating tree automaton (this holds for Buchi, Rabin, Streett, Muller, and other such acceptance conditions). They define the transition function of an alternating tree automaton to be a mapping from the state set (times the alphabet) into the free distributive lattice generated by all possible pairs of (direction, state). The complement automaton is obtained by just dualizing the transition function, and complementing the acceptance condition.

Converting an alternating automaton $\mathscr{A}$ to an equivalent non-deterministic tree automaton $\mathscr{N}$ requires a Safra-like determinization and a forgetful strategy theorem (in fact, we will be using the memoryless strategy theorems).

This conversion requires collection of all "forall" (universal) runs along a particular path of the input tree, and choice of a non-deterministic (existential) run, into a single non-deterministic run (guaranteeing soundness and completeness). This is just the dual of an online string automaton determinization, like that of Safra [Sa88, Sa92]. Furthermore, the resulting non-deterministic transition function (of $\mathscr{N}$) must provide for all the choices available to the individual for-all runs of $\mathscr{A}$. However, the number of such runs increases arbitrarily as the depth of the tree increases. The memoryless strategy theorems of the previous sections show that it is sufficient to provide the same choice to different for-all runs as long as their current state is the same (also note that the depth of the runs is the same, as it is equal to the current depth in the tree t).

Thus, using [MS87], a Streett non-deterministic tree automaton $\mathscr{D}$ can be converted to a complement Rabin alternating tree automaton. Then, using Safra's single exponential determinization of Streett ω-automata (note that we need the dual in collecting the for-all runs above) and the memoryless strategy theorem of Section 3, one obtains a Streett non-deterministic tree automaton equivalent to the complement of $\mathscr{D}$ (with only an exponential blowup in state size, and a linear blowup in number of pairs); This result first appeared (with a different proof) in [Kl92].

The problem of complementing Rabin's tree automata with only a single exponential blowup remains open.

ACKNOWLEDGMENTS

The author thanks the referees for making several useful suggestions, which have helped to give a better presentation.

Received April 19, 1993; final manuscript received November 18, 1996

REFERENCES

[Bu83] Buchi, J. R. (1983), State-strategies for games in $F_{\sigma\delta} \cap G_{\delta\sigma}$, *J. Symbolic Logic* **48**, 1171–1198.

[BL69] Buchi, J. R., and Landweber, L. H. (1969), Solving sequential conditions by finite-state strategies, *Trans. Am. Math. Soc.* **138**, 295–311.

[CE81] Clarke, E. M., and Emerson, E. A. (1981), Design and synthesis of synchronization skeletons using branching time temporal logic, *in* "Proceedings IBM Workshop on Logics of Programs," Lecture Notes in Computer Science, No. 131, Springer-Verlag, Berlin/New York.

[EH82] Emerson, E. A., and Halpern, J. Y. (1982), Decision procedures and expressiveness in the temporal logic of branching time, *in* "Proceedings, 14th ACM Symposium on Theory of Computation," pp. 169–180.

[EJ88] Emerson, E. A., and Jutla, C. S. (1988), Complexity of tree automata and modal logics of programs, *in* "Proceedings, 25th IEEE Symposium on Foundations of Computer Science."

[EJ91] Emerson, E. A., and Jutla, C. S. (1991), Mu-calculus, tree automata and determinacy, *in* "Proceedings, 28th IEEE Symposium on Foundations of Computer Science."

[ESi84] Emerson, E. A., and Sistla, A. P. (1984), Deciding branching time logic, *Inform. and Control.* **61**, 175–201.

[GH82] Gurevich, Y., and Harrington, L. (1982), Trees, automata, and games, *in* "Proceedings, 14th ACM Symposium on Theory of Computer Science."

[J90] Jutla, C. S. (1990), "Automata on Infinite Objects and Modal Logics of Programs," Ph.D. thesis, The University of Texas at Austin.

[Kl90] Klarlund, N. (1990), "Progress Measures and Finite Arguments for Infinite Computations," Ph.D. thesis, Cornell University.

[Kl92] Klarlund, N. (1992), Progress measures, immediate determinacy, and a subset construction for tree automata, *in* "Proceedings, 7th IEEE Symposium on Logics in Computer Science."

[Ma75] Martin, D. A. (1975), Borel determinacy, *Ann. of Math.* **102**, 363–371.

[Mc66] McNaughton, R. (1966), Testing and generating infinite sequences by a finite automaton, *Inform. and Control.* **9**, 521–530.

[Mo84] Mostowski, A. W. (1984), Regular expressions for infinite trees and a standard form of automata, *in* "Computation Theory" (A. Skowron, Ed.), Lecture Notes in Computer Science, Vol. 208, Springer-Verlag, Berlin/New York.

[MS87] Muller, D. E., and Schupp, P. E. (1987), Alternating automata on infinite trees, *Theoret. Comput. Sci.* **54**, 267–276.

[Ra69] Rabin, M. O. (1969), Decidability of second order theories and automata on infinite trees, *Trans. Amer. Math. Soc.* **141**, 1–35.

[Sa88] Safra, S. (1988), On complexity of ω-automata, *in* "Proceedings, 29th IEEE Symposium on Foundations of Computer Science."

[Sa89] Safra, S. (1989), "Complexity of Automata on Infinite Objects," Ph.D. thesis, Weizmann Institute of Science, Rehovot, Israel.

[Sa92] Safra, S. (1992), Exponential determinization for ω-automata with strong fairness acceptance condition, *in* "Proceedings, 24th ACM Symposium on Theory of Computer Science."

[St81] Street, R. S. (1981), "A Propositional Dynamic Logic of Looping and Converse," MIT LCS Technical Report TR-263.