SIAM J. COMPUT.
Vol. 36, No. 3, pp. 803–814

EXPONENTIAL DETERMINIZATION FOR ω-AUTOMATA WITH A STRONG FAIRNESS ACCEPTANCE CONDITION*

SHMUEL SAFRA†

Abstract. In [S. Safra, *Proceedings of the* 29*th IEEE Symposium on Foundations of Computer Science*, 1988, pp. 319–327] an exponential determinization procedure for Büchi automata was shown, yielding tight bounds for decision procedures of some logics (see [A. E. Emerson and C. Jutla, *Proceedings of the* 29*th IEEE Symposium on Foundations of Computer Science*, 1988, pp. 328–337; Safra (1988); S. Safra and M. Y. Vardi, *Proceedings of the* 21*st ACM Symposium on Theory of Computing*, 1989, pp. 127–137; and D. Kozen and J. Tiuryn, *Logics of program*, in Handbook of Theoretical Computer Science, Elsevier, Amsterdam, 1990, pp. 789–840]). In Safra and Vardi (1989) the complexity of determinization and complementation of ω-automata was further investigated, leaving as an open question the complexity of the determinization of a single class of ω-automata. For this class of ω-automata with strong fairness as an acceptance condition (*Streett automata*), Safra and Vardi (1989) managed to show an exponential complementation procedure; however, the blow-up of translating these automata—to any of the classes known to admit exponential determinization—is inherently exponential. This might suggest that the blow-up of the determinization of Streett automata is inherently doubly exponential. This paper shows an exponential determinization construction for Streett automata. In fact, the complexity of our construction is roughly the same as the complexity achieved in Safra (1988) for Büchi automata. Moreover, a simple observation extends this upper bound to the complementation problem. Since any ω-automaton that admits exponential determinization can be easily converted into a Streett automaton, we have obtained a single procedure that can be used for all of these conversions. Furthermore, this construction is optimal (up to a constant factor in the exponent) for all of these conversions. Our results imply that Streett automata (with strong fairness as an acceptance condition) can be used instead of Büchi automata (with the weaker acceptance condition) without any loss of efficiency.

Key words. ω-automata, verification, reactive systems

AMS subject classification. 68Q45

DOI. 10.1137/S0097539798332518

1. Introduction.

Finite automata on infinite words (ω-automata), despite their seemingly fantastic definition, have quite an earthly role in the formal analysis of ongoing (reactive) systems. A *reactive system* is one whose goal is to continuously interact with its environment, as opposed to computing a function on an input and terminating. Take, for example, a file editor; it is not computing a function of a preset input, and its execution should not terminate, unless the environment so insists. (Other examples of such systems are control programs of a robot or an unmanned spacecraft.) Suppose one would like to make sure that a reactive system functions properly. For systems that compute functions, we need to verify that the system always terminates and computes the correct value of the function; what would be the reactive system's equivalent?

First, one needs to make sure that every reaction produced by the system is proper (safety), and second, that every anticipated reaction is eventually produced (liveness). In our file editor example, once an editor command is given, it must eventually be

*Received by the editors January 6, 1998; accepted for publication (in revised form) February 4, 2004; published electronically November 3, 2006. Part of this work was carried out while the author was at MIT and was supported in part by a Weizmann fellowship and NSF grant CCR-8912586. Part of this work was also carried out while the author was at IBM Almaden and Stanford University.
http://www.siam.org/journals/sicomp/36-3/33251.html

†Computer Science Department, Tel-Aviv University, 69978 Tel Aviv, Israel (safra@math.tau.ac.il).

carried out. This demonstrates the notion usually referred to as *weak fairness*:

- Weak fairness: Any continuously enabled action is eventually carried out.

Now suppose we are working on a paper[1] on an operating system that can run several file editors concurrently, but there is only one display on which we can see how the paper will look once printed. So, every now and then, while we continue to work on the paper, we try to display it, but each time there is someone else's paper already on display. The system might be "weakly fair" and yet not display the paper, since this action is not continuously enabled. Still, some might find it unfair if they try again and again to display their file but never get the chance. This demonstrates the stronger notion of fairness usually referred to as *strong fairness*:

- Strong fairness: Any action that is repeatedly enabled is eventually carried out.

Notice a problem here; suppose some action is enabled every now and then, and the computation ends without the action having been carried out. Just by observing a finite computation, how can one distinguish between the two cases (a) the system is not strongly fair, and (b) the system is slow and whoever wanted the action taken eventually gave up. (Decided to print the paper?)

Our solution is to interpret reactive systems over infinite computations; it does not mean we actually run infinite computations,[2] rather that we *analyze the fairness* of the system on infinite computations. On such computations we can distinguish between the case of a slow system and an unfair one, using algorithms that run in *finite* time. The two fairness conditions above look as follows:

- Weak fairness: A computation is unfair if there is an action that is enabled continuously from some point on but is carried out only finitely many times.
- Strong fairness: A computation is unfair if there is an action that is enabled infinitely often but is carried out only finitely many times.

Therefore, our computations are infinite objects (an infinite sequence for linear time, and an infinite tree for branching time), and the formal meaning (semantics) of a system is the set of computations it may produce. The specification of the system is given in some specification language (logic) over these infinite objects. In order to verify that the system functions properly, we check that the set of computations produced by the system is a subset of the computations that meet the specification.

For a complete exposition of the above subjects and related ones, the reader is referred to [HP85, Fra86, MP91].

Finite memory systems. We now restrict our attention to systems that can be described as finite-state machines (at least for the purpose of the formal analysis). It turns out that any reasonable logic for specification in the finite-state case describes a set of computations acceptable by a finite automaton over infinite objects (described below). Moreover, the most efficient decision procedures for these logics are usually obtained by translating a formula in the logic to an automaton and checking emptiness of the language this automaton accepts [VW86]. The most efficient procedures for the problem of model checking (checking that a program meets some specification) are also usually obtained using automata. A finite-state program can be viewed as a finite-state machine, and in order to check that it meets some specification, it is enough to check the *containment* of the language accepted by this finite-state machine in the language accepted by the specification automaton [VW86a].

[1]at the latest possible time to meet the deadline, quite naturally.

[2]if anyone has doubts.

There are two basic automata conversions that come up in these procedures: *complementation* and *determinization.*

This type of procedure was first suggested by Büchi [Büc62] in his original paper introducing ω-automata, in order to show that the validity of S1S (the monadic second order theory of one successor) is decidable. Büchi showed that ω-automata are closed under complementation; however, the blow-up of the complementation procedure he suggested is doubly exponential. McNaughton [McN66] showed that ω-automata can be determinized (into a deterministic automaton with a stronger acceptance condition than the one Büchi suggested). The blow-up of his determinization construction, however, is also doubly exponential. Rabin introduced tree automata and used McNaughton's result to show that these automata are closed under complementation. He could then give a decision procedure for a stronger logic—S2S (the monadic second order theory of many successors).

The decision of these logics is known to be nonelementary [Mey75], and thus there is no hope of achieving a reasonable complexity. However, when considering simpler logics and attempting to obtain more efficient procedures, the blow-up of the above constructions was prohibitive.

Sistla, Vardi, and Wolper [SVW87] showed an exponential complementation procedure for Büchi automata and utilized this result to obtain tight bounds for various logics. The exponential determinization of Büchi automata [Saf88], which also improves on [SVW87], was used [EJ88] to show a tight bound for the complexity of the decision procedure of various logics, which allow quantification over time-paths and thus require translation to tree automata (e.g., CTL*, Δ-PDL, μ-calculus, etc.). An exponential complementation for Streett automata was shown [SV89] and was utilized so as to improve the upper bound for the decision of linear-time logics that are translatable more efficiently to automata with strong fairness as an acceptance condition.

Finite automata over infinite objects. Automata on infinite words (ω-automata) are the same as automata on finite words except that, since a run over a word does not have a final state, the acceptance condition is on the set of states visited infinitely often in the run. The simplest acceptance condition was suggested by Büchi [Büc62], in which some of the states are designated as accepting, and a run is accepting if it visits infinitely many times the accepting set of states.

Muller [Mul63] suggested deterministic ω-automata, with a different acceptance condition, as a means of describing the behavior of nonstabilizing circuits. The acceptance condition he suggested is to specify explicitly all the "good" infinity sets (the *infinity set* of a run ξ is the set of states that occur infinitely many times in ξ). A run is accepting if its infinity set is one of the designated accepting sets. When we consider acceptance conditions based on the infinity set, this is obviously the most expressive condition.

A Rabin acceptance condition is, syntactically, a set of pairs of subsets of the states, $\{(L_i, U_i)\}_i$. A run ξ is accepting if, for one of the pairs i, ξ visits infinitely many times some states in L_i (the "good" states), and only finitely often the states in U_i (the "bad" states).

Streett [Str82] suggested the complementary condition to Rabin's condition, which is syntactically the same: a set of pairs of subsets of the states. A run ξ is accepting according to Streett's condition if, for all pairs i, if the run visits infinitely many times L_i it also visits infinitely many times U_i.

We may write Rabin's condition as $\bigvee_i L_i \wedge \neg U_i$ and Streett's condition as $\bigwedge_i L_i \to U_i$. Streett's condition corresponds to strong fairness (as defined above) since for each event e the acceptance condition could contain a pair $\langle L_i, U_i \rangle$, in which L_i is the set of states in which e is enabled and U_i is the set of states in which e is taken (weak fairness can be expressed by Büchi automata).

Previous best results on determinization and complementation. In [Saf88, SV89, Kla91] the complexities of determinization and complementation of different classes of ω-automata were studied, and they were solved in full except for the complexity of determinization of Streett automata. An exponential complementation procedure was shown for Streett automata in [SV89] and with a better exponent in [Kla91]. It was shown [SV89] that the blow-up of the translation of Streett automata to any of the classes of ω-automata that were known to admit exponential determinization is inherently exponential.

Our results. Our main result (Theorem 1) is a new determinization construction for Streett automata. Given a Streett automaton with n state and h accepting pairs, we construct a deterministic Rabin automaton with $2^{O(nh \log nh)}$ states and nh accepting pairs. Using the small number of accepting pairs in the determinized Rabin automaton, and a simple complementation construction for deterministic Rabin automata, which is exponential only in the number of accepting pairs (Lemma 3), we show that nondeterministic Streett automata can be converted into deterministic Streett automata with the same (exponential) blow-up (Corollary 4). Since the same deterministic automaton interpreted as a Streett or Rabin automaton accepts two complementary languages, this implies that Streett automata can be simultaneously complemented and determinized (codeterminized) into both Streett or Rabin automata with only an exponential blow-up.

The exact complexity of the complementation procedures obtained in this way matches the complexity of the complementation procedure of [Kla91].

The results reported herein were first published as an extended abstract [Saf92]. It is worthwhile noting that an independent work reported in [Wa93] may share some of these methods in a somewhat different setting.

2. Basic definitions.

An *ω-automaton* over an *alphabet* Σ, $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, C \rangle$, consists of a finite set of *states* Q, an *initial state* $q_0 \in Q$, a *transition relation* $\delta \colon Q \times \Sigma \to 2^Q$, and an *acceptance condition* C. We extend δ to sets of states and sequences of letters in the usual way.

A sequence of states, $\xi \in Q^\omega$, is an *$\mathcal{A}$-run* over a word $\sigma \in \Sigma^\omega$ if $\xi_0 = q_0$ and, for every i, ξ_{i+1} is a σ_i successor of ξ_i, i.e., $\xi_{i+1} \in \delta(\xi_i, \sigma_i)$.

The *infinity set* of a sequence of letters (or states) σ, $\inf(\sigma)$, is the set of letters that appear infinitely many times in σ (i.e., $\inf(\sigma) = \{a \text{ s.t. } |\{i \text{ s.t. } \sigma_i = a\}| = \infty\}$).

An infinite word $\sigma \in \Sigma^\omega$ is *accepted* by an automaton $\mathcal{A}$ if there exists an accepting $\mathcal{A}$-run over σ. The *language* accepted by an automaton is the set of all words accepted by it.

An automaton is *deterministic* if for all $a \in \Sigma$, $q \in Q$, $|\delta(q, a)| = 1$, i.e., δ is a function into Q. Obviously, any word has exactly one run in a deterministic automaton.

We define classes of automata corresponding to the different acceptance conditions. We write N for nondeterministic and D for deterministic, and B, M, R, S for Büchi, Muller, Rabin, and Streett, respectively.

The acceptance conditions are summarized in the following table:

	Syntax	Semantics
B	$F \subseteq Q$	$\inf(\xi) \cap F \neq \phi$
M	$\mathbf{F} \subseteq 2^Q$	$\inf(\xi) \in \mathbf{F}$
R	$\bigvee_i L_i \wedge \neg U_i$	$\exists i\colon\ \inf(\xi) \cap L_i \neq \phi \ \wedge$ $\inf(\xi) \cap U_i = \phi$
S	$\bigwedge_i L_i \rightarrow U_i$	$\forall i\colon\ \inf(\xi) \cap L_i \neq \phi \ \rightarrow$ $\inf(\xi) \cap U_i \neq \phi$

Concerning the *size* of an automaton, we denote both the number of states and the size of the acceptance condition (except for Büchi automata, where the acceptance condition may be neglected). For example, our main result can be written as $\mathrm{NS}(n,h) \rightarrow \mathrm{DR}(2^{\mathrm{O}(nh\log(nh))}, nh)$.

3. Determinization of NS.

THEOREM 1. $\mathrm{NS}(n,h) \rightarrow \mathrm{DR}(2^{O(nh\log nh)}, nh)$*; i.e., for any* NS *automaton* $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \bigwedge_{0<i\leq h} L_i \rightarrow U_i \rangle$ *with* n *states and* h *acceptance pairs, there exists an equivalent* DR *automaton* $\mathcal{D} = \langle \Sigma, \widetilde{Q}, \widetilde{q}_0, \widetilde{\delta}, \bigvee_{0<i\leq nh} G_i \wedge \neg B_i \rangle$*, with* $2^{\mathrm{O}(nh\log(nh))}$ *states and* nh *acceptance pairs.*

Proof of Theorem 1. Throughout this proof we denote by H the set of indexes $[1..h]$.

Intuition. It is easier to look at the deterministic Rabin automaton $\mathcal{D}$ as a program with bounded memory and some infinitary acceptance condition. This program reads the input one letter at a time and changes its memory accordingly. The corresponding finite ω-automaton has a different state for each of the possible states of the program's memory. The infinite string is accepted if the set of memory states visited infinitely often satisfies the acceptance condition. We now describe $\mathcal{D}$ informally.

An accepting $\mathcal{A}$-run ξ has a *witness set* $J \subseteq H$ for which ξ visits infinitely many times each U_j for $j \in J$ and only finitely many time any L_j for $j \notin J$.

Given a witness set J, one can construct a small nondeterministic Büchi automaton that accepts all strings for which there is an accepting run ξ with witness set J. This automaton consists of two parts; the first one is a copy of $\mathcal{A}$ (without the acceptance condition). Each run at each point can nondeterministically guess that no state in any of the sets L_j, for $j \notin J$, will be visited from now on, and choose to move to the second part. The second part consists of $|J|+1$ copies of $\mathcal{A}$, in which the run can move to the next copy only after visiting the set U_j corresponding to the current copy. A run is accepting if it cycles infinitely through all the copies. All states $q \in L_j$ for $j \notin J$ are removed from all the copies of $\mathcal{A}$ in the second part. Hence an accepting run visits only finitely many times copies of $q \in L_j$ for $j \notin J$, and infinitely many times copies of $q \in U_j$ for each $j \in J$.

This automaton can be determinized with only an exponential blow-up [Saf88]. However, since the number of possible witness sets is exponential, a construction of an automaton that deterministically considers all the witness sets results in a doubly exponential blow-up.

The determinization construction suggested here may be viewed as a deterministic dynamic process that at each point in time considers only a polynomial number of witness sets.

The deterministic automaton $\mathcal{D}$, while maintaining the subset of $\mathcal{A}$-states reached by reading the prefix of the input, starts by assuming that the witness set of the accepting run (if one exists) is H, i.e., $\mathcal{D}$ tries, during each run, to cycle through all the U_j's. Whenever a run is waiting to visit some U_{j_1}, $\mathcal{D}$, assuming (the worst) that the run will never again visit U_{j_1}, spawns off a parallel construction, with possibly a smaller subset of the $\mathcal{A}$-states, and with the witness set $J' = J \setminus \{j_1\}$ (disallowing any state $q \in L_{j_1}$ in the subprocess). Any run that eventually visits U_{j_1} is advanced to the next index. In the subprocess, if again a run is waiting for U_{j_2}, $\mathcal{D}$ branches off recursively with a smaller witness set. An important observation is that, for each such parallel process and for each state that appears in the subset maintained by the process, one needs to consider only one index—the most advanced one; hence the subset of the $\mathcal{A}$-states maintained by the process is partitioned among the different indexes. In the good case, in which eventually all $\mathcal{A}$-states have runs that completed a cycle, all the subprocesses are killed and the process is restarted. If any of these processes is restarted infinitely many times, $\mathcal{D}$ accepts.

However, suppose that some runs completed a cycle, but some other runs are stuck waiting for some U_{j_1}. The latter runs prevent the former runs from restarting the process. Therefore, for all runs that completed a cycle through U_j for every $j \in J$, $\mathcal{D}$ spawns off a parallel process with a smaller set of $\mathcal{A}$-states (this is similar to the determinization construction of [Saf88]). In addition (again following [Saf88]) $\mathcal{D}$ maintains an order among the subprocesses according to which subprocess was spawned first. Whenever a state appears in more than one subprocess (with the same index; otherwise, as mentioned above, the more advanced index takes priority) it is removed from all but the one spawned first.

Since for each state in each process one needs to consider only one U_j it is waiting for, the number of witness sets we need to try in parallel at any given time is polynomial.

We now return to the formal proof of Theorem 1. We start with some definitions we need for the construction of the set $\widetilde{Q}$ of states of $\mathcal{D}$.

Let $V = [1..2nh]$ be the set of *names* (these are used by $\mathcal{D}$ to preserve the identity of different parallel applications of some basic construction).

For $S \subseteq Q$, let an *S-atom* be $\langle v, S\rangle$, where $v \in V$.

For $S \subseteq Q$ and $J \subseteq H$, we give a recursive definition of an *(S, J)-decomposition*:

1. An S-atom is an (S, J)-decomposition.
2. Let $v \in V$.
   Let $S_1, \dots, S_l$ be a partition of S (i.e., $\bigcup_i S_i = S$ and for any $i \neq j \in [1..l]$, $S_i \cap S_j = \phi$).
   Let $j_1, \dots, j_l \in J \cup \{0\}$, where at least one of the j's is nonzero. Denote by J_i, for each $i \in [1..l]$, the set $J \setminus \{j_i\}$.
   Let $\Pi_1, \dots, \Pi_l$ be such that for each $i \in [1..l]$, Π_i is an (S_i, J_i)-decomposition.
   Then $\langle v, (\Pi_1, j_1), \dots, (\Pi_l, j_l)\rangle$ is an (S, J)-decomposition.

For an (S, J)-decomposition we refer to the decompositions used in the recursion of this definition as *subdecompositions*.

An (S, J)-decomposition has a *good name* if the names assigned to each of the subdecompositions are all different. We consider, from now on, only decompositions with a good name.

Note that the recursion in the definition of an (S, J)-decomposition is finite, since not all the indexes j can be 0, and thus either the set of states S or the set of indexes J decreases each level down the recursion.

We can even give a more specific bound on the size and number of decompositions, as follows.

LEMMA 2. *The number of subdecompositions in an (S, H)-decomposition ($S \subseteq Q$) is at most nh, and the total number of (S, H)-decompositions ($S \subseteq Q$) is at most $2^{O(nh \log(nh))}$.*

Proof. For an (S, J)-decomposition Π we say that a pair (q, j), for $q \in Q$, $j \in H$, is *special* for a (S', J')-subdecomposition Π' of Π if the following three conditions hold:

- $q \in S'$;
- the index set of the immediate subdecomposition of Π' that contains q is a strict subset of J' (i.e., this subdecomposition is not indexed by 0);
- if a subdecomposition has index set $J'' \neq J'$ such that $J' \subseteq J''$, then $j \in J''$.

Now, each pair (q, j) is special for a subdecomposition, and each subdecomposition Π' has a pair (q, j) which is special for it. Therefore, Π can be represented as a partial function from the set of pairs (q, j) to the set of names V. $\square$

The construction of $\mathcal{D}$.

The set of states. $\widetilde{Q}$ is the set of all (S, H)-decompositions for a subset of the states $S \subseteq Q$.

The initial state. $\widetilde{q}_0 = \langle 1, \{q_0\} \rangle$, i.e., the $\{q_0\}$-atom with 1 (arbitrarily) as its name.

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

1. Replace each atom $\langle v, S \rangle$ in $\widetilde{q}$ by $\langle v, \delta(S, a) \rangle$.
   This results in some structure that may violate the requirement, in the definition of an (S, J)-decomposition above, that the sets $S_1, \ldots, S_l$ be disjoint. At the end of the following several steps, this requirement is restored.
2. For any nonatomic (S, J)-subdecomposition in $\widetilde{q}$, let $j_1, \ldots, j_l$ be the indexes as in the definition of a decomposition above, and let $S'_1, \ldots, S'_l$ be its sets of $\mathcal{A}$-states after step 1. For each $\mathcal{A}$-state q and i such that $q \in S'_i$,
   - if $q \in L_{j_i}$, then remove q from S'_i and append to the list of subdecompositions an atom $\langle v, \{q\} \rangle$ with index $j = \max\{J\}$;
   - otherwise, if $q \in U_{j_i}$, then append an atom $\langle v, \{q\} \rangle$ with index $j = \max\{(J \cup \{0\}) \cap \{0, \ldots, j_i - 1\}\}$.

   In both cases $v \in V$ is an unused name in $\widetilde{q}$, and each new atom is assigned a different name. This is possible since, by Lemma 2, only nh out of $2nh$ names in V are used in $\widetilde{q}$.
   For atomic (S, J)-subdecompositions, we follow the same procedure, assuming S is a one-item list with the maximal index in J as its index.
3. For a nonatomic subdecomposition in $\widetilde{q}$ for which, after the previous steps, an $\mathcal{A}$-state q appears in a set S'_i with index j and a set $S'_{i'}$ with index $j' > j$, remove q from $S'_{i'}$.
4. For a nonatomic subdecomposition in $\widetilde{q}$ for which, after the previous steps, an $\mathcal{A}$-state q appears in a set S'_i and a set $S'_{i'}$, where $i' > i$, remove q from $S'_{i'}$.
5. Remove any empty set from any list.
6. Replace any nonatomic (S, J)-subdecomposition whose name is v, in which after the previous steps all indexes are 0, by an atom $\langle v, S \rangle$.

The acceptance condition. For each name $v \in V$, let G_v be the set of states $\widetilde{q}$ in which v is the name of an atom, and let B_v be the set of states $\widetilde{q}$ in which v is not used in the decomposition.

Correctness. $L(\mathcal{D}) \subseteq L(\mathcal{A})$: Given that there is a name v, which in the $\mathcal{D}$-run ξ over a word σ is used (in the decompositions ξ_i) continuously from some point on, and is the name of an atom infinitely many times, we prove that there is an accepting $\mathcal{A}$-run over σ.

For two positions $0 \leq l < k$, we denote by $\sigma[l, k)$ the finite word $\sigma_l, \ldots, \sigma_{k-1}$.

Let l be the largest such that v is not used in the decomposition ξ_l, and let S_i, for $i > l$, be the set such that v is the name of an (S_i, J)-subdecomposition of ξ_i. Let

$l_1 < l_2 < \cdots$ (where $l_1 > l$) be the positions at which v is the name of an atom in ξ_{l_k}. By the construction, $S_{l_1} \subseteq \delta(q_0, \sigma[1, l_1))$. The condition to make a subdecomposition become an atom (step 6) and the conditions to create new subdecompositions and maintain them (steps 2 and 1) ensure that for each $q \in S_{l_{k+1}}$, for $k > 0$, there exists some $q' \in S_{l_k}$, as well as an $\mathcal{A}$-run over $\sigma[l_k, l_{k+1})$ which leads from q' to q while visiting all U_j for $j \in J$ and no L_j for $j \notin J$.

Intending to use König's lemma, we construct a tree whose nodes are all the pairs of the form (q, k) for $q \in S_{l_k}$. As the parent of a node $(q, k+1)$ we pick one of the pairs (q', k) such that $q' \in S_{l_k}$ and there exists an $\mathcal{A}$-run from q' to q, as described above. The root of the tree is $(q_0, 0)$.

By König's lemma, since there are infinitely many pairs, and the number of pairs at each level of the tree is bounded, there is an infinite path, $(q_0, 0), (q_1, 1), \ldots$, in the tree. By the construction of this tree, for each $k > 0$ there is an $\mathcal{A}$-run, as described above, from q_k to q_{k+1}, over $\sigma[l_k, l_{k+1})$. The infinite concatenation of these segments gives an $\mathcal{A}$-run over σ which visits each $\mathcal{A}$-state in U_j for $j \in J$ infinitely many times, and visits any state in L_j for $j \notin J$ only at the first segment. This $\mathcal{A}$-run is accepting.

$L(\mathcal{A}) \subseteq L(\mathcal{D})$: Given that, for a string σ, there is an accepting $\mathcal{A}$-run, ξ, we prove that there is a name v which in the $\mathcal{D}$-run over σ is used continuously from some point on and is the name of an atom infinitely many times.

Let l be the length of the finitary prefix of ξ; i.e., every state ξ_k for $k > l$ appears infinitely many times in ξ. Let the ith state $\widetilde{q}_i$ of the $\widetilde{\mathcal{D}}$-run over σ be an (S_i, H)-decomposition; then it must be that $\xi_i \in S_i$. Therefore, S_i is never empty and its name v_1 remains fixed in all $\widetilde{q}_i$. If $\widetilde{q}_i$ becomes an atom infinitely many times, we are done. Otherwise, let i_1 be the largest such that $\widetilde{q}_i$ is an atom. For $i > i_1$, as i increases, the $\mathcal{A}$-state ξ_i appears in subdecompositions with monotonically nonincreasing index, and thus its index is eventually fixed. Thereafter, ξ_i can move only closer to the beginning of the sequence of immediate subdecompositions. Hence, there is an i'_1 such that for all $i > i'_1$, ξ_i appears in a subdecomposition with a fixed name v_2. We can now repeat the argument and show that if v_2 is not a name of an atom infinitely many times, then eventually the state of ξ appears one level down in the decomposition. Since the depth of the decomposition is finite, there must be a subdecomposition which becomes an atom infinitely many times.

This completes the proof of our main theorem. ∎

4. Complementation of DR.

In order to see how to codeterminize (construct a deterministic automaton that accepts the complement) Streett automata, we need the following lemma.[3]

LEMMA 3. $\mathrm{DS}(n, h) \rightarrow \mathrm{DR}(n \cdot 2^{h \log h}, h+1)$*; i.e., for any deterministic Streett automaton with* n *states and* h *accepting pairs, there exists an equivalent deterministic Rabin automaton with* $n \cdot 2^{h \log h}$ *states and* $h+1$ *accepting pairs.*

One should comment here that one can quite easily complement such automata while leaving the set of states fixed, translating the acceptance condition from Rabin's to Streett's. Such a translation, however, would incur an exponential blow-up in the size of the acceptance condition. The above lemma is useful when applying it to the main theorem, as the size of the acceptance condition remains polynomial in the size of the original automaton, while the number of states becomes exponential.

Proof. We show an explicit construction, given a DS automaton, $\mathcal{D} = \langle \Sigma, Q, q_0, \delta,$

[3]This lemma appears in [Saf88] but only in the journal version and is repeated here for the 21st century reader.

$\bigwedge_{1\le i\le h} L_i \to U_i\rangle$, of a DR automaton, $\widetilde{\mathcal{D}} = \langle \Sigma, \widetilde{Q}, \widetilde{q}_0, \widetilde{\delta}, \bigvee_{1\le i\le h+1} \neg\widetilde{L}_i \wedge \widetilde{U}_i\rangle$.

Intuition. Again let us think of the automaton $\widetilde{\mathcal{D}}$ as a program with bounded memory; the states of $\widetilde{\mathcal{D}}$ will be all possible data states that this program's memory can be in.

$\widetilde{\mathcal{D}}$ maintains, aside from the state $\mathcal{D}$ reaches after reading the prefix of the input, a permutation of the set of indexes $[1..h]$. Whenever a state in some U_j is visited, the index j is moved to the end of the permutation (if several are visited, all of their indexes are moved to the end with no particular order). Hence, for every accepting run, let J be its witness set and let $i = h - |J|$; then eventually the first i elements of the permutation are fixed, and for each index j in the suffix of the permutation, U_j is visited infinitely many times. The Rabin acceptance condition contains, for each $i \in [0..h]$, a pair in which the "bad" set contains all states in which some L_j for j in the first i elements in the permutation is visited, and the "good" set contains all states in which U_j is visited for j being the $(i+1)$th element in the permutation.

We now give the formal proof of the lemma.

The construction. The states of $\widetilde{\mathcal{D}}, \widetilde{Q}$ have the form of a tuple, (q, π, r, g), where $q \in Q$, π is a permutation of $[1..h]$, and $r, g \in [1..h+1]$. The initial state $\widetilde{q}_0 = (q_0, \langle 1, \dots, h\rangle, h+1, h+1)$.

Consider a state $\widetilde{q} = (q, \pi, r, g)$, where $\pi = \langle i_1, \dots, i_h\rangle$. For a letter $a \in \Sigma$ define $\widetilde{\delta}(\widetilde{q}, a)$ to be the state $\widetilde{q}' = (q', \pi', r', g')$ as follows:

- $q' = \delta(q, a)$.
- g' is the minimal index i such that $q' \in U_{j_i}$, if it exists, and otherwise $g' = h+1$.
- r' is the minimal index i such that $q' \in L_{j_i}$, if it exists, and otherwise $r' = h+1$.
- $\pi' = \langle j_1, \dots, j_{g'-1}, j_{g'+1}, \dots, j_h, j_{g'}\rangle$ if $g' \le h$; otherwise, $\pi' = \pi$.

For i, $1 \le i \le h+1$, $\widetilde{L}_i$ consists of all the states $\widetilde{q}$ in which $r < i$, and $\widetilde{U}_i$ consists of all the states $\widetilde{q}$ in which $g = i$.

Note that after reading a finite prefix of the input, there is a part to the left of π which is fixed from then on, and that contains all the indexes j such that U_j is visited only finitely many times. If that prefix is of length i, then from then on $g > i$.

$L(\widetilde{\mathcal{D}}) \subseteq L(\mathcal{D})$: Assume that for some i, $r \ge i$ from some point on, and $g = i$ infinitely many times. Since $g = i$ infinitely many times, for every j, if U_j is visited only finitely many times, eventually j is placed in π with an index smaller than i ($j = j_k$ for $k < i$), and by the construction, from then on, L_j is never visited.

$L(\mathcal{D}) \subseteq L(\widetilde{\mathcal{D}})$: Assume there is an accepting $\mathcal{D}$-run; then there exists a maximal set $J \subseteq [1..h]$ such that for $k \in J$, U_{j_k} is visited infinitely many times, and for $k \notin J$, neither L_{j_k} nor U_{j_k} is visited from some point on. There exists a further point, after which $[1..h] \setminus J$ occupies the leftmost positions in π, and none of its indexes changes its place. Let $i = h - |J|$. Obviously, $r \ge j$ beyond this point. We claim that $g = i$ infinitely many times. At any point, let $j_i = k$. Since the run visits U_k infinitely many times, on the next visit to U_k, g will be i.

Complexity. The number of states is $n \cdot h! \cdot (h+1)^2 < n \cdot 2^{h \log h}$ (for $h > 5$). ☐

Since in their deterministic versions the same automaton interpreted as a Rabin automaton and as a Streett automaton accept two complementary languages, we can conclude the following.

COROLLARY 4. $\overline{\mathrm{NS}(n,h)} \to \mathrm{DR}(2^{\mathrm{O}(nh\log(nh))}, nh+1)$. ☐

Hence, NS can be translated to DS with this complexity.

5. Complementation of Streett tree automata. Rabin [Rab69, Rab70] introduced automata on infinite trees. The input of such an automaton is an infinite binary tree, whose nodes are labeled by letters from some finite alphabet Σ, $T\colon \{0,1\}^* \to \Sigma$. A Σ-tree automaton, $\mathcal{T} = \langle \Sigma, Q, q_0, \delta, C \rangle$, is similar to an ω-automaton except that the transition function specifies, for a state $q \in Q$ and a letter $a \in \Sigma$, a set of pairs of states containing both a left successor state q_l and a right successor state q_r (i.e., $\delta\colon Q \times \Sigma \to 2^{Q \times Q}$). A $\mathcal{T}$-run is a Q-tree, $\Gamma\colon \{0,1\}^* \to Q$, in which the root node is q_0, and all the nodes satisfy δ, i.e., for each node $p \in \{0,1\}^*$, $(\Gamma(p0), \Gamma(p1)) \in \delta(\Gamma(p), T(p))$. The acceptance condition C is any ω-condition, such as those of Büchi, Rabin, Muller, or Streett. A $\mathcal{T}$-run Γ is defined to be accepting if the infinity set of *every* infinite path in Γ satisfies C.

Given a tree automaton $\mathcal{A}$, the complementary tree automaton $\bar{\mathcal{A}}$ may be viewed as supplying a proof, given some input tree, that every nondeterministic $\mathcal{A}$-run has a path that does not satisfy $\mathcal{A}$'s acceptance condition. Following the procedure of Gurevich and Harrington [GH82] the proof is supplied by a finite memory strategy. A finite memory strategy is one that can be represented by a set of finite tables, for each node in the tree, giving for each state and some finite information about the history of the run so far either a left direction or a right direction. Such a strategy serves as a proof that the input tree is not accepted by $\mathcal{A}$ if, for any choice of nondeterministic moves, and following the direction the strategy supplies for each move, the resulting infinite sequence of states is not accepted according to $\mathcal{A}$'s acceptance condition.

Using the techniques of [EJ91] it can be shown [Jutla] that the complement of a Streett tree automaton has a memoryless strategy; i.e., $\bar{\mathcal{A}}$, for each node in the input tree, needs to guess a table showing for each *state* (i.e., no information at all on the history of the run) the direction to go for a nonaccepting path. Using the exponential determinization for Streett automata shown here this implies an exponential complementation procedure for Streett tree automata. A different proof for this theorem appeared first in [Kla92].

Discussion. This paper showed that ω-automata with a strong fairness acceptance condition can replace Büchi automata in all applications without loss of efficiency. This should have further applications than those shown here, in the formal analysis of finite-state systems, and, in general, for a better understanding of the notion of fairness.

The main result of this paper has applications with regards to the complementation of tree automata: Streett tree automata were shown to have exponential complementation construction [Kla92]; the results reported herein imply a simple such exponential complementation procedure for these automata. The main open problem left in this area is the complexity of complementation of Rabin tree automata; is it doubly exponential, or can one show an upper bound similar to the one for Streett tree automata?

Acknowledgments. I would like to thank Moshe Vardi and Amir Pnueli for many most insightful discussions on this problem, and the unnamed referees for most constructive comments.

REFERENCES

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

[Büc62] J. R. Büchi, *On a decision method in restricted second order arithmetics*, in Proceedings of the International Congr. on Logic, Method. and Phil. of Sci., 1960, E. Nagel et al., eds., Stanford University Press, Stanford, CA, 1962, pp. 1–12.

[EJ88] A. E. Emerson and C. Jutla, *The complexity of tree automata and logics of programs*, in Proceedings of the 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 328–337.

[EJ91] A. E. Emerson and C. Jutla, *Tree automata mu-calculus and determinacy*, in Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, 1991, pp. 368–377.

[ES84] A. E. Emerson and P. A. Sistla, *Deciding full branching time logic*, Inform. and Control, 61 (1984), pp. 175–201.

[Fra86] N. Francez, *Fairness*, Springer-Verlag, New York, 1986.

[GH82] Y. Gurevich and L. Harrington, *Trees, automata, and games*, in Proceedings of the 14th ACM Symposium on Theory of Computing, 1982, pp. 60–65.

[HP85] D. Harel and A. Pnueli, *On the development of reactive systems*, in Logics and Models of Concurrent Systems, K. R. Apt, ed., Springer, Berlin, 1985, pp. 477–498.

[Jutla] C. Jutla, *personal communication*.

[Kla91] N. Klarlund, *Progress measures for complementation of ω-automata with application to temporal logic*, in Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science, 1991, pp. 358–367.

[Kla92] N. Klarlund, *Progress measures, immediate determinacy, and a subset construction for tree automata*, in Proceedings of the 7th IEEE Symposium on Logic in Computer Science, 1992, pp. 382–393.

[KT90] D. Kozen and J. Tiuryn, *Logics of programs*, in Handbook of Theoretical Computer Science, Vol. B, J. Van Leeuwen, ed., Elsevier, Amsterdam, 1990, pp. 789–840.

[MP91] Z. Manna and A. Pnueli, *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer, New York, 1991.

[McN66] R. McNaughton, *Testing and generating infinite sequences by a finite automaton*, Inform. and Control, 9 (1966), pp. 521–530.

[Mey75] A. R. Meyer, *Weak monadic second order theory of successor is not elementary recursive*, in Proceedings of the Boston University Logic Colloquium, 1973, Lecture Notes in Math. 453, Springer, Berlin, 1975, pp. 132–154.

[Mul63] D. E. Muller, *Infinite sequences and finite machines*, in Proceedings of the 4th IEEE Symposium on Switching Circuit Theory and Logical Design, 1963, pp. 3–16.

[Pec86] J. P. Pecuchet, *On the complementation of Büchi automata*, Theoret. Comput. Sci., 47 (1986), pp. 95–98.

[Rab69] M. O. Rabin, *Decidability of second-order theories and automata on infinite trees*, Trans. Amer. Math. Soc., 141 (1969), pp. 1–35.

[Rab70] M. O. Rabin, *Weakly definable relations and special automata*, in Proceedings of the Symposium on Mathematical Logic and Foundation of Set Theory, Y. Bar-Hillel, ed., North-Holland, Amsterdam, 1970, pp. 1–23.

[Saf88] S. Safra, *On the complexity of ω-automata*, in Proceedings of the 29th IEEE Symposium on Foundations of Computer Science, 1988, pp. 319–327. An extended version to appear in J. Comput. System Sci.

[Saf92] S. Safra, *Exponential determinization for ω-automata with strong-fairness acceptance condition (extended abstract)*, in Proceedings of the 24th ACM Symposium on Theory of Computing, 1992, pp. 275–282.

[SV89] S. Safra and M. Y. Vardi, *On ω-automata and temporal logic*, in Proceedings of the 21st ACM Symposium on Theory of Computing, 1989, pp. 127–137.

[SVW87] A. P. Sistla, M. Y. Vardi, and P. Wolper, *The complementation problem for Büchi automata with application to temporal logic*, Theoret. Comput. Sci., 49 (1987), pp. 217–237.

[Str82] R. S. Streett, *Propositional dynamic logic of looping and converse is elementary decidable*, Inform. and Control, 54 (1982), pp. 121–141.

[Var85] M. Y. Vardi, *Automatic verification of probabilistic concurrent finite-state programs*, in Proceedings of the 26th IEEE Symposium on Foundations of Computer Science, 1985, pp. 327–338.

[VS85] M. Y. Vardi and L. Stockmeyer, *Improved upper and lower bounds for modal logics of program*, in Proceedings of the 17th ACM Symposium on Theory of Computing, 1985, pp. 240–251.

[VW86] M. Y. Vardi and P. Wolper, *Automata theoretic techniques for modal logics of programs*, J. Comput. System Sci., 32 (1986), pp. 183–221.

[VW86a] M. Y. Vardi and P. Wolper, *An automata-theoretic approach to automatic program verification*, in Proceedings of the 1st IEEE Symposium on Logic in Computer Science, 1986, pp. 332–344.

[Wa93] I. Walukiewicz, *A complete deductive system for the μ-calculus*, in Proceedings of the 8th IEEE Symposium on Logic in Computer Science, 1993, pp. 136–147.