On The Complexity of ω-Automata[*]

Shmuel Safra

Department of Applied Mathematics
Weizmann Institute of Science
Rehovot 76100, Israel

Abstract

Automata on infinite words were introduced by Büchi in order to give a decision procedure for S1S, the monadic second-order theory of one successor. Muller suggested deterministic ω-automata as a means of describing the behavior of non-stabilizing circuits. McNaughton proved that the classes of languages accepted by nondeterministic Büchi automata and by deterministic Muller automata are the same. His construction and its proof are quite complicated, and the blow-up of the construction is doubly exponential.

Our main result is a new determinization construction. The advantages of the construction are that it is simpler and yields a single exponent upper bound for the general case. This construction is optimal. Using the construction we can also obtain an improved complementation construction for Büchi automata, which is also optimal. Both constructions can be used to improve the complexity of decision procedures that use automata-theoretic techniques.

1 Introduction

Automata on infinite words were introduced by Büchi [Büc62, Büc60] in order to give a decision procedure for S1S, the monadic second-order theory of one successor. He showed that each well formed formula in this calculus is, in some sense, equivalent to a nondeterministic automaton on infinite words. Automata on infinite words 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. In a Büchi automaton, some of the states are designated as accepting, and a run is accepting if it visits infinitely many times the accepting set of states (i.e., if the intersection of the infinity set of the run with the set of accepting states is not

[*]A preliminary version of this paper appeared in [Saf88].

empty). While Büchi was interested in the nondeterministic version of these automata, it is possible to study also the deterministic version. It turns out that deterministic Büchi automata are less expressive than nondeterministic Büchi automata.

Muller [Mul63] suggested deterministic ω-automata, with a different acceptance condition, as a means of describing the behavior of non-stabilizing circuits that can be properly described only by their infinite behavior. The acceptance condition he suggested is to specify explicitly all the 'good' infinity sets. A run is accepting if its infinity set is one of the designated accepting sets. When we consider acceptance condition based on the infinity set, this is obviously the strongest possible condition.

The fundamental result in the theory of ω-automata, proved by McNaughton in [McN66], shows that the classes of languages accepted by nondeterministic Büchi automata and by deterministic Muller automata are the same. The hard part of the proof is to show that every nondeterministic automaton has an equivalent deterministic automaton. This fact was proven by a direct construction.

In fact, McNaughton's construction converts any Büchi automaton into a deterministic automaton with an acceptance condition which is a special case of Muller's condition. The condition used by McNaughton was later formalized by Rabin [Rab69]. 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). The blow-up of McNaughton's construction, however, is quite large; the size of the resulting Rabin automaton is doubly exponential in the size of the original Büchi automaton.

The construction and its proof are quite complicated, and in view of the importance of this result, many authors have proved alternative constructions [Büc73, Eil74, Cho74, Rab72, TB73, Sch73, Tho81]. However, the complexity of some of these constructions is even triply exponential.

The construction of a deterministic ω-automaton equivalent to a nondeterministic ω-automaton is an important and recurrent element in decision procedures for various logics. For example, it serves as a natural basic step in the decision procedures of S2S ([Rab69, GH82]), infinite games ([BL69]), CTL* ([ES84]), Δ-PDL, μ-calculus etc. ([VS85]), and probabilistic verification [Var85]. However, when concerned with complexity, due to the prohibitive cost associated with the determinization process, none of the decision procedures used the determinization process in its entirety. Emerson and Sistla [ES84] developed a special determinization process which is only singly exponential, based on the special properties (reverse deterministic) of the automata associated with linear temporal logic. Vardi and Stockmeyer [VS85] reduced the satisfiability problem for various modal logics to the emptiness problem for hybrid tree automata, in order to avoid the natural reduction to emptiness of tree automata ([Str82]), which involves determinization. Vardi ([Var85]) developed a polynomial time verification procedure for probabilistic programs, given a specification by a deterministic ω-automaton. The natural procedure, given a specification by a nondeterministic automaton, is first to determinize the automaton, and then apply the verification procedure. He showed that there is no need for complete determinization in order to apply the procedure, the automaton need only be deterministic in the limit. The partial determinization construction described in [Var85] is exponential, but the author has subsequently found that it contains an error, which is corrected by our results.

Another important problem that arises when using ω-automata for specification and decision procedures, is the complementation problem; given an automaton, to construct another automaton that accepts the complementary language. Büchi proved that his automata are closed under complementation [Büc62], using the infinite version of Ramsey's Theorem. This fact follows immediately from McNaughton's theorem, as noted in [McN66]. These involve at least a doubly exponential blow-up. An exponential construction was later found [SVW87] (see also [Pec86]), again using Ramsey's Theorem. For any Büchi automaton of size n they give a complementary one of size $O(2^{4n^2})$.

Our main result (Theorem 1) is a new determinization construction. The advantages of the construction are that it is simpler to understand and yields a single exponent upper bound for the general case. More precisely, given a Büchi automaton of size n, we construct a deterministic Rabin automaton with $2^{O(n \log n)}$ states and n accepting pairs. This construction can be shown to be optimal up to a constant in the exponent.

Using the small number of accepting pairs in the determinized 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 Büchi automata can be simultaneously complemented and determinized with the same complexity. In addition, we give an alternative simple complementation construction for nondeterministic Büchi automata (Corollary 6), which improves the known upper bound of [SVW87]. For a Büchi automaton of size n, we construct a complementary Büchi automaton of size $2^{O(n \log n)}$. It follows from a recent result of Michel ([Mic88]) that these constructions are essentially optimal.

Using these results, the natural reduction of the satisfiability problem for Δ-PDL, μ-calculus, etc., to the emptiness problem for tree automata, results in a Rabin tree automaton whose size is exponential. Furthermore, since the number of pairs in the resulting Rabin ω-automaton is linear, the number of pairs in the resulting Rabin tree automaton is also linear. Using a decision procedure for emptiness of tree automata, whose running time is polynomial in the number of states and exponential in the number of accepting pairs ([EJ88], see also [PR89]), Emerson and Jutla were able to give a deterministic exponential time decision procedure for the satisfiability of formulas in these logics. These are essentially the lower bounds for these problems.

Our determinization construction obviously validates the upper bound (for partial determinization) claimed in [Var85]. Furthermore, a partial determinization (Corollary 7) with a slightly better complexity, namely $O(3^n)$, can be extracted from our determinization construction ([Var]). A similar construction for partial determinization was discovered independently by Courcoubetis and Yannakakis [CY88].

In the concluding remarks we discuss the complexity of translations between different classes of automata.

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* $\delta: Q \times \Sigma \to 2^Q$, and an *acceptance condition* C.

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, is defined as $\inf(\sigma) = \{a: |\{i: \sigma_i = a\}| = \infty\}$.

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.

Streett [Str82] suggested the complementary condition to Rabin's condition mentioned above, which is syntactically the same, a set of pairs of subsets of the states, and a run ξ is accepting 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 $\bigwedge_i L_i \to U_i$.

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	$\bigvee_i L_i \wedge \neg U_i$	$\exists i$: $\inf(\xi) \cap L_i \neq \phi \wedge$ $\inf(\xi) \cap U_i = \phi$
S	$\bigwedge_i L_i \to U_i$	$\forall i$: $\inf(\xi) \cap L_i = \phi \vee$ $\inf(\xi) \cap U_i \neq \phi$

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

3 Determinization

The determinization construction we show generalizes the subset construction [RS59]. States in the constructed deterministic automaton are ordered trees of subsets of states. We first give an overview and then the formal construction and its proof.

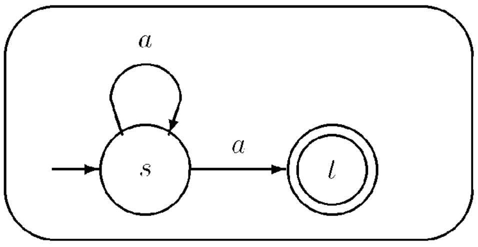
Overview:

Assume a nondeterministic Büchi automaton $\mathcal{A}$. We describe a deterministic finite state automaton $\mathcal{D}$, that reads the (infinite) input sequentially, and simulates $\mathcal{A}$.

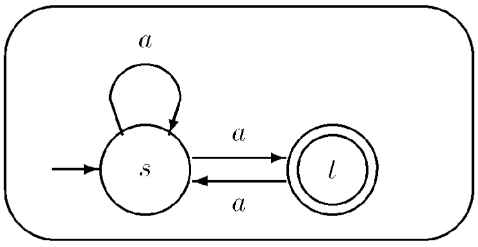
We first show why the usual subset construction is not sound, i.e., the constructed deterministic automaton may accept words not accepted by $\mathcal{A}$. We then consider an alternative version, leading to a subset construction which is sound but incomplete, i.e., it may reject words accepted by $\mathcal{A}$. Finally, we present the *subset tree construction* which consists of a tree, where each node maintains the result of applying the alternative subset construction to the states associated with the node. This construction is sound and complete.

In the straightforward subset construction, the deterministic automaton, $\mathcal{D}$, maintains, at each step, the set of $\mathcal{A}$-states reachable by the prefix of the word read so far. The accepting $\mathcal{D}$-states are those that contain an accepting $\mathcal{A}$-state. In the finitary case, since the acceptance condition is concerned only with the final state of the run, this convention is sufficient. In the infinitary case, for some word σ, there might be infinitely many *different* $\mathcal{A}$-runs, each visiting the accepting set only finitely many times, but no *single* $\mathcal{A}$-run that visits the accepting set infinitely many times. Such a word is accepted according to the suggested acceptance convention, although it is not accepted by $\mathcal{A}$.

To demonstrate the problem, consider the following two automata (over the singleton alphabet $\Sigma = \{a\}$):

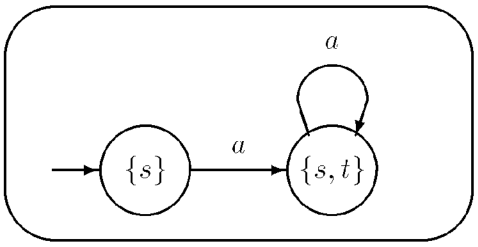


$\mathcal{A}$



$\mathcal{B}$

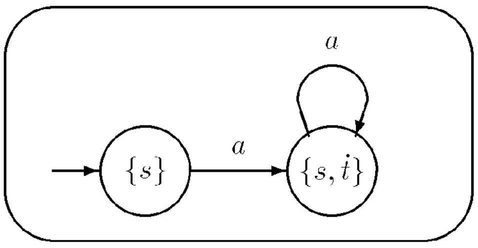
The automaton $\mathcal{A}$ does not accept any word (because, once the automaton enters t, it cannot continue), while $\mathcal{B}$ accepts a^ω (the run $(st)^\omega$ is accepting). Applying the subset construction to these two automata yields, in both cases, the automaton:



The initial state is $\{s\}$, and from then on, both s and t are reachable, which causes the subset automaton to remain in $\{s, t\}$. Obviously, this automaton cannot represent the deterministic version of both $\mathcal{A}$ and $\mathcal{B}$ which are inequivalent.

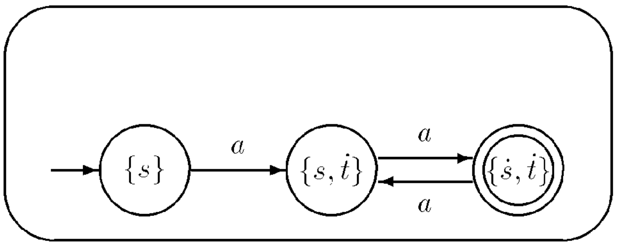
An alternative construction tries to identify breakpoints in the course of reading a word σ and simulating the runs of $\mathcal{A}$ on σ. The initial stage is identified as a breakpoint. The next breakpoint occurs as soon as each reachable $\mathcal{A}$-state has a run that visited an accepting $\mathcal{A}$-state since the last breakpoint. The word σ is accepted if the automaton $\mathcal{D}$ identifies infinitely many breakpoints while reading σ. Technically, we do that by marking every accepting state, and every state that has a marked predecessor; if all the states are marked, we identify a breakpoint (say, have a green light flash for one step), and delete the markings from all the $\mathcal{A}$-states. If the green light flashes infinitely many times, there must be a run that visited an accepting state between any two consecutive flashes.

For example, consider the two automata, $\mathcal{A}$ and $\mathcal{B}$ described above. The result of applying this construction to $\mathcal{A}$ yields the following deterministic automaton:



$\mathcal{A}'$

Note that s is never marked. Applying the same construction to $\mathcal{B}$ yields the following automaton:

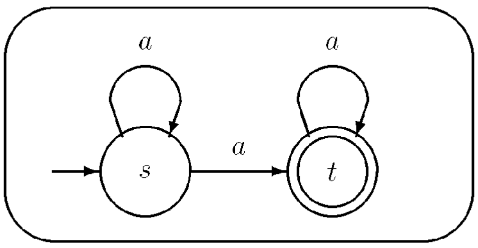


$\mathcal{B}'$

Note that here s is marked infinitely many times by the run coming from t.

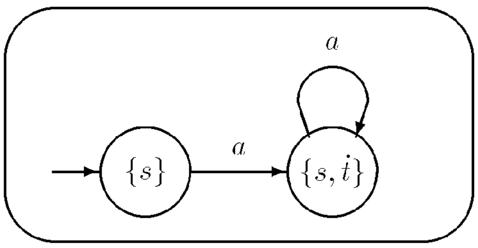
However, it is possible that, although there is an accepting $\mathcal{A}$-run, there exists, at each step, a reachable state q such that no run reaching q visited any accepting state. This causes each of the constructed subset states to contain an unmarked state, and hence the green light will never flash. Thus, the suggested acceptance convention may reject some words accepted by $\mathcal{A}$.

A simple example to that effect is the following automaton, which obviously accepts a^ω:



$\mathcal{A}_1$

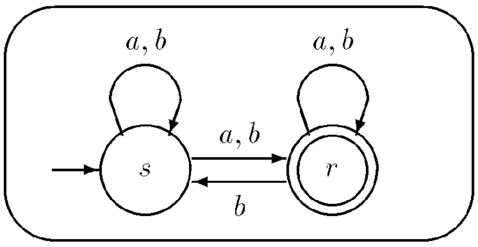
The construction suggested above yields the following automaton, which does not accept any word:



$\mathcal{A}'_1$

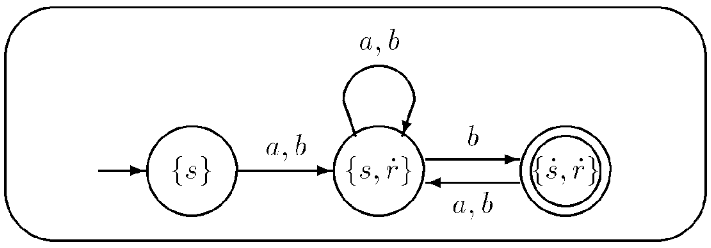
This phenomena may appear in a more complex form, in which the green light flashes k times for some $k > 0$, but beyond that last flash there is always a reachable unmarked state. This corresponds to the situation, in the original automaton, in which there is at each step a reachable state q, such that all runs reaching q at this step visit accepting states at most k times.

Consider, for example, the following nondeterministic automaton over $\Sigma = \{a, b\}$:



$\mathcal{A}_2$

Obviously, this automaton accepts all words in Σ^ω. Applying the construction described above, we obtain the following deterministic automaton:



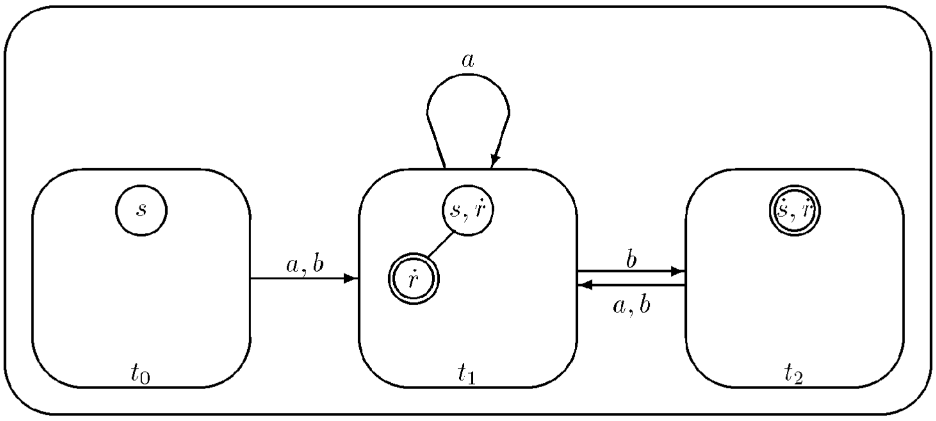
$\mathcal{A}'_2$

The automaton accepts only words which have infinitely many b's. For every word that contain k b's, the automaton can visit the accepting state $\{\dot{s}, \dot{r}\}$ at most k times, and beyond that step remain in $\{s, \dot{r}\}$ forever.

To resolve this problem, we construct each state of $\mathcal{D}$ as a tree of subsets. We start $\mathcal{D}$ as a simple singleton tree that has only a root, and apply the suggested subset construction. If the green light does not flash at the root, $\mathcal{D}$ copies the set of marked states, and makes it a son of the root node. If, again, at the next step, there are some states which are marked in the root, but do not appear in the son, $\mathcal{D}$ adds them as an additional son, and so on. This procedure is applied recursively to all nodes of the tree, using different marks and green flashes for each node. In the good case, where eventually the root becomes green, all the descendants of the root can be eliminated and we start afresh. In the problematic case, where, beyond some point, there is always an unmarked state in the root, this state does not propagate to any of the sons. In a similar way, each node of the tree must contain at least one unmarked state which is not contained in any of its descendants. Since the number of states of $\mathcal{A}$ is finite, this bounds the depth of the tree.

$\mathcal{D}$ accepts a word σ, if, during the run over σ, one of the nodes becomes infinitely often green, and is contained in all the states of the run, from a certain point on.

The subset tree construction for the automaton $\mathcal{A}_2$ is presented below:



Note that there are two ways in which this automaton accepts a word from Σ^ω: if the word contain infinitely many b's, then the run visits t_2 infinitely many times, and the word is accepted because the root node turns green infinitely often. If, on the other hand, the word contain only finitely many b's, then beyond some point, the run is contained in t_1, and the word is accepted because the leaf node containing r alone turns green infinitely often.

In order to bound the width of the tree, whenever two sons contain the same state, (different runs may converge to the same state) we leave the responsibility of that state to the senior son, thus removing it from the younger one (and from all its decendants). Since seniority is a well founded relation, every run eventually finds itself in one of the sons forever.

Note that, in the actual construction, there is no need for marking states; every state that appears in any of the sons is considered marked.

Theorem 1 $\mathrm{NB}(n) \rightarrow \mathrm{DR}(2^{\mathrm{O}(n\log n)}, n)$; *for any nondeterministic Büchi automaton of size n, there exists an equivalent deterministic Rabin automaton with $2^{\mathrm{O}(n\log n)}$ states and n accepting pairs.*

Proof: We describe the construction in more details, give a proof of correctness and evaluate the complexity.

In the sequel we will use labeled ordered trees. We introduce some notation concerning such trees.

Let $\mathcal{V}$ be a vocabulary, i.e., a set of symbols. An *ordered $\mathcal{V}$-tree* is a structure $T = \langle N, r, p, S\rangle$, consisting of the following elements:

$N \subseteq \mathcal{V}$ – A set of nodes, taken from the vocabulary $\mathcal{V}$.

$r \in N$ – A distinguished node called the *root* of T.

p – A *parenthood function* defined over $N - \{r\}$, and defining for each $d \in N - \{r\}$, its parent $p(d) \in N$.

S – The *sibling ordering* relation, a partial ordering relation. We require that S is total on any two nodes d_1, d_2, which share the same parent, i.e., either $(d_1, d_2) \in S$ or $(d_2, d_1) \in S$.

If $(d_1, d_2) \in S$ we say that d_1 is an older brother of d_2 (d_2 is a younger brother of d_1).

For nodes d_1 and d_2, we define d_1 to be an *ancestor* of d_2, if for some $k > 0$, $p^k(d_2) = d_1$. The nodes d_1 and d_2 are said to be *ancestral* if either d_1 is an ancestor of d_2, or d_2 is an ancestor of d_1.

The sibling ordering can be extended to a larger ordering holding between any two nodes which are not ancestral, as follows: we say that d_1 is to the *left* of d_2 (or d_2 is to the right of d_1), if some ancestor of d_1 is an older brother of some ancestor of d_2.

A *labeled ordered $\mathcal{V}$-tree* consists of a $\mathcal{V}$-tree T and a mapping $l: N \rightarrow L$ where L is some set of elements, called the *label set*. In our case, the label of each node is a set of states of the automaton to be determinized, and we will refer to them as the states that *belong* to the node.

The Construction:

Given an NB, $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, F \rangle$, we construct a DR, $\mathcal{D} = \langle \Sigma, Q', q'_0, \delta', C \rangle$. Let $|Q| = n$. Fix a vocabulary V such that $|V| = n$. The states of the DR, Q', are labeled ordered V-trees whose nodes are labeled with subsets of Q, and such that the following conditions are satisfied:

1. The union of the labels of the sons of a node $d \in N$ is a proper subset of the label of d, and
2. The labels of two nodes which are not ancestral are disjoint.

In addition, we associate with each element $v \in V$ a color that can be white, green or red.

To summarize, each state $q' \in Q'$ consists of the following elements:

T – a V-tree as defined above.

$l: N \rightarrow 2^Q$ – the labeling function.

$c: V \rightarrow \{\text{white}, \text{green}, \text{red}\}$ – the coloring function.

We call a state $q \in Q$ *specific* to a node $d \in N$, if q belongs to d, and does not belong to any node which is not an ancestor of d. The above conditions, 1 and 2, imply that any state that appears in the tree is specific to a single node d (and appears exactly in all the nodes along the path from the root to d).

The initial state of $\mathcal{D}$, q'_0, consists of the singleton tree whose only node is labeled by $\{q_0\}$, and the coloring function which colors all elements of V white.

For any $\mathcal{D}$-state of the form $q' = (T, l, c)$, and a letter $a \in \Sigma$, $\delta'(q', a)$ is the result of applying the following sequence of action on (T, l, c). In carrying out this translation, we work with an extended vocabulary $V \cup V'$, where V' is a set of elements of size n which is disjoint from V. Consequently, we will be considering intermediate trees with $N \subseteq V \cup V'$.

1. For every node d with label $\hat{Q}$, replace $\hat{Q}$ by the label $\delta(\hat{Q}, a)$ $(= \bigcup_{q \in \hat{Q}} \delta(q, a))$.
2. For every node d with label $\hat{Q} \subseteq Q$, such that $\hat{Q} \cap F \neq \phi$, add a new node $d' \in V'$ to the tree (increasing N), such that d' becomes the youngest son of d, and label it with $\hat{Q} \cap F$.

   The following several steps reestablish the special structure required from our tree.

3. For every node d with label $\hat{Q}$ and state $q \in \hat{Q}$, such that q also belongs to a node $\hat{d}$ to the left of d, remove q from $\hat{Q}$.

   Thus, a state which, as a result of steps 1–2, appears on more than one path, is removed from all but the leftmost path on which it appears.

4. Remove all nodes with empty labels.

   Note that since the label of each node is contained in the label of its parent, if d is retained after step 4, then all of its ancestors are also retained.

5. For every node $d \in N \cap V$ whose label is equal to the union of the labels of its sons, remove all the descendants of d and color d green; color all other elements of $V \cap N$ white, and all elements of $V - N$ red.

   Note that nodes in $N \cap V'$ have no descendants.

6. Rename each node $d' \in N \cap V'$ to some $d \in V - N$, i.e., an unused element of V.

   We observe that the size of N at each stage is bounded by n. Because of the requirements 1 and 2 holding over our trees, any node $d \in N$ must have at least one state which is specific to d (and to no other node in N). Consequently, N can contain at most n nodes. Therefore, there are always sufficiently many elements in $V - N$ at that point.

The process of applying δ' involves in general both removal of nodes and creation of new nodes. In step 2 of the process we create new nodes, while in step 4 and 5 we remove nodes.

The acceptance condition requires that for some $v \in V$,
v is red ($c(v) =$ red) only finitely many times
and
v is green infinitely many times.

Note that this leads to a Rabin condition with n pairs, namely all the pairs (L_v, U_v) for $v \in V$. The set L_v are all the $\mathcal{D}$-states in which v is green. The set U_v are all the $\mathcal{D}$-states in which v is red.

Correctness:

Soundness: Given that there is a element $v \in V$, which in a run over a word σ, turns red only finitely many times and turns green infinitely many times, we prove that there is an accepting $\mathcal{A}$-run over σ.

For two positions $0 \le i < j$, we denote by $\sigma[i, j)$ the finite word $\sigma_i, ..., \sigma_{j-1}$.

Let $0 < i_1 < i_2 < ...$ be the positions at which v turns green after the last time v is red. By the construction, it follows that for all positions $j \ge i_1$, v is in the tree, and therefore has a defined label. Let $S_j \subseteq Q$, for $j > 0$, be the label of v at position i_j, and also define $i_0 = 0$ and $S_0 = \{q_0\}$. The condition to make a node green (step 5), and the condition to create and maintain nodes (steps 2 and 1) ensure that for each $q \in S_{j+1}$, $j \geq 0$, there exists some $q' \in S_j$, and a run of $\mathcal{A}$ over $\sigma[i_j, i_{j+1})$ which leads from q' to q while visiting an accepting state.

Intending to use König's Lemma, we construct a tree whose nodes are all the pairs of the form (q, j) for $q \in S_j$. As the parent of a node $(q, j+1)$ we pick one of the pairs (q', j) such that $q' \in S_j$ and there exists a run from q' to q as described above.

Obviously, this is a well formed tree, in which every node (q, i) has a unique path leading from the root $(q_0, 0)$ to (q, i).

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 $j \geq 0$ there is a run, as described above, from q_j to q_{j+1}, over $\sigma[i_j, i_{j+1})$. The infinite concatenation of these segments gives a run over σ which visits an accepting state between each two consecutive levels. This run is accepting.

Completeness: Given that there is an accepting $\mathcal{A}$-run, ξ, we prove that there is a node which is infinitely often green, and only finitely many times red.

The state of ξ, at every stage, is included in the root, thus the root is never removed (step 4), and hence it is never red. If the root is green infinitely many times, we are done. Otherwise, consider the first visit of ξ to an accepting state after the last time the root is green. At that point, the state of ξ is placed in one of the sons of the root (step 2). The state of ξ can be moved to an older son (step 3), but since there are only finitely many older sons (condition 2), it eventually remains in the same son forever. This son is never removed (step 4), thus it is never red. Either it becomes green infinitely many times, or by repeating the argument, the state of ξ is eventually placed in the next level. Since the depth of the tree is finite (condition 1), there must be a node which turns green infinitely often.

Complexity:

It is simple to see that the number of different trees of the above form is at most singly exponential in n. However, more careful analysis leads to the bound of $2^{O(n \log n)}$.

The number of ordered trees (without labels) over V is at most $2^{n \log n + O(n)}$. We can represent the labels (the subsets) by a function from Q to V, which for each state q gives the node d such that q is specific to d. The label of a node d is the set of states which are specific to d or to any descendant of d. The number of such functions is also $2^{n \log n}$, and each element has a color. Thus, the total number of ordered trees is $2^{2n \log n + O(n)} \leq 2^{O(n \log n)}$.

We have already observed that the number of accepting pairs is n.

This completes the proof of Theorem 1. □

We note that a modification of an argument in [Mic88] can be used to show an $n!$ lower bound for determinization (into DR), thus the above construction is optimal up to a constant in the exponent.

Note also that our construction leads to an acceptance condition of linear size; if we want a Muller acceptance condition, the size of the acceptance condition becomes doubly exponential. The following lemma shows that this blow-up is unavoidable.

Lemma 2 *The determinization of* NB *into* DM *is at least doubly exponential.*

The proof is given in appendix A.

4 Applications

The following section describes some applications of Theorem 1.

Complementation

One of the important properties of ω-automata, established in [Büc62] is their closure under complementation. The work in [SVW87] presented for the first time a complementation construction that yields an automaton of exponential size. The precise size obtained there was $O(2^{4n^2})$. Using the determinization procedure described above, we can obtain a complement of size $2^{O(n \log n)}$.

An even more important transformation is to find, for an NB automaton $\mathcal{A}$, a DR automaton $\mathcal{D}$, such that $\mathcal{D}$ accepts the ω-language which is complementary to the one that $\mathcal{A}$ accepts. We may refer to this transformation as co-determinization. In the following we show that for any NB automaton of size n, it is possible to construct a DR automaton $\mathcal{D}$, of size $(2^{O(n \log n)}, n+1)$, which is the deterministic complement of $\mathcal{A}$.

Both these results are based on the following Lemma:

Lemma 3 $\mathrm{DS}(n, h) \to \mathrm{DR}(n \cdot 2^{h \log h}, h+1)$*; 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.*

Proof:

The construction: Given a DS, $\mathcal{D} = \left\langle \Sigma, Q, q_0, \delta, \bigwedge_{1 \le i \le h} L_i \to U_i \right\rangle$, we construct a DR, $\hat{\mathcal{D}} = \left\langle \Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \bigvee_{1 \le i \le h+1} \neg \hat{L}_i \wedge \hat{U}_i \right\rangle$, whose states $\hat{Q}$ have the form of a tuple, (q, π, r, g), where $q \in Q$, π is a permutation over $[1..h]$ which is represented as a list of indexes, and $r, g \in [1..h+1]$. The initial state $\hat{q}_0 = (q_0, (1, ..., h), h+1, h+1)$.

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

$q' = \delta(q, a)$.

g' is the minimal index j such that $q' \in U_{i_j}$, if it exists, and otherwise $g' = h + 1$.

r' is the minimal index j such that $q' \in L_{i_j}$, if it exists, and otherwise $r' = h + 1$.

$\pi' = i_1, ..., i_{g'-1}, i_{g'+1}, ..., i_h, i_{g'}$ if $g' \leq h$; otherwise, $\pi' = \pi$.

For j, $1 \leq j \leq h+1$, $\hat{L}_j$ consists of all the states $\hat{q}$ in which $r < j$, and $\hat{U}_j$ consists of all the states $\hat{q}$ in which $g = j$.

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 contain all the indexes i such that U_i is visited only finitely many times. If that prefix is of length j, then from then on $g > j$.

Soundness: Assume that for some j, $r \geq j$ from some point on, and $g = j$ infinitely many times. Since $g = j$ infinitely many times, for every i, if U_i is visited only finitely many times, eventually i is placed in π with an index smaller than j ($i = i_k$ for $k < j$) and by the construction, from then on, L_i is never visited.

Completeness: Assume there is an accepting $\mathcal{D}$-run, then there exists a maximal set $G \subseteq [1..h]$ such that for $k \in G$, U_{i_k} is visited infinitely many times, and for $k \in \overline{G}$, neither L_{i_k} nor U_{i_k} are visited from some point on. There exists a further point, after which $\overline{G}$ occupies the leftmost positions in π, and none of its indexes changes its place. Let $j = \left|\overline{G}\right| + 1$. Obviously, $r \geq j$ beyond this point. We claim that $g = j$ infinitely many times. At any point, let $i_j = k$. Since the run visits U_k infinitely many times, on the next visit to U_k, g will be j.

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

Corollary 4 $\overline{\text{NB}}(n) \rightarrow \text{DR}(2^{\text{O}(n \log n)}, n+1)$*; for any nondeterministic Büchi automaton with* n *states, there exists a complementary deterministic Rabin automaton, with* $2^{\text{O}(n \log n)}$ *states and* $n+1$ *accepting pairs.*

Proof: Use the determinization construction of Theorem 1 to obtain a DR of size $(2^{\text{O}(n \log n)}, n)$. Interpret it as a DS to get the complement, and then use the above conversion to get a DR whose number of states is $2^{\text{O}(n \log n)} \cdot 2^{n \log n} = 2^{\text{O}(n \log n)}$ and $n+1$ accepting pairs. □

A similar result, , based on our determinization construction, and leading to a DR of size $(2^{\text{O}(n^2)}, \text{O}(n^2))$ has been independently derived and is reported in [EJ89].

Lemma 5 *(Folk)* $\text{NR}(n, h) \rightarrow \text{NB}(2nh)$. □

Corollary 6 $\overline{\text{NB}}(n) \rightarrow \text{NB}(2^{\text{O}(n \log n)})$*; for any* NB *of size* n*, there exists a complementary* NB *of size* $2^{\text{O}(n \log n)}$.

Proof: Use the co-determinization construction of Corollary 4 to get a DR of size $(2^{O(n\log n)}, n+1)$, and then use Lemma 5 to get a complementary NB of size $2 \cdot 2^{O(n\log n)} \cdot n+1 = 2^{O(n\log n)}$. □

Complementation of NB is clearly at least exponential. It is possible to extend the language, establishing this lower bound for the finitary case (automata on finite words), into an infinitary language which establishes the same lower bound for ω-automata. A recent result by Michel [Mic88] raises the lower bound to $n!$, i.e. $2^{O(n\log n)}$. This establishes the optimality of our construction.

Partial Determinization

Vardi ([Var85]) investigated automatic verification of probabilistic concurrent finite state programs. Given such a program and a specification, the problem is to verify that the program meets the specification. He developed a polynomial time verification procedure, given a specification by a deterministic ω-automaton. A temporal logic specification can be translated exponentially into an NB ([SVW87]). The natural procedure, given a specification by a nondeterministic ω-automaton, is first to determinize the automaton, and then apply the verification procedure for deterministic automata ([Var85]). Lacking an exponential determinization construction, this process takes doubly exponential time (thus triply exponential for temporal logic). Using our exponential determinization construction, this process takes exponential time for NB, thus doubly exponential time for temporal logic, which are, essentially, the upper bounds claimed in [Var85].

It was shown in [Var85] that there is no need for complete determinization, the automaton need only be deterministic in the limit. An automaton is deterministic in the limit if any accepting run makes only finitely many nondeterministic choices. A partial determinization with a slightly better complexity than the complete determinization, namely $O(3^n)$, can be extracted from our determinization construction ([Var]).

Corollary 7 *For every* NB *of size n there is an equivalent, deterministic in the limit, Büchi automaton, of size* $O(3^n)$.

Proof: Given an NB, $\mathcal{A}$, we construct an equivalent, deterministic in the limit NB, $\mathcal{A}'$, consisting of two disjoint parts. The first is a copy of $\mathcal{A}$, in which no state is accepting; the second maintains the sound but incomplete version of the subset construction described in the overview of the determinization construction. The automaton, $\mathcal{A}'$, starts at the first part and chooses nondeterministically, at each point, either to stay in the first part, or to initialize the second part. Being in a state q in the first part, it initializes the second part with the set $\{q\}$. The second part maintains the usual subset construction; in addition, each state contained in the subset may be marked. A state is marked if either it is an accepting state, or it has a marked predecessor; if all the $\mathcal{A}$-states contained in the subset are marked, a green light flashes, and we delete all the marks. A run is accepting if the green light flashes infinitely often. □

Decision Procedures for Modal Logics

As mentioned in the Introduction the satisfiability problem for various modal logics can be reduced to the emptiness problem for tree automata ([Str82, ES84, VS85]). Using the co-determinization construction described above, and, in particular, the small number of accepting pairs in the resulting DR, satisfiability of a Δ-PDL formula can be reduced to the emptiness of a Rabin tree automaton with exponential number of states, and linear number of accepting pairs. In the case of CTL*, the resulting tree automaton has doubly exponential number of states and exponential number of accepting pairs.

Using a decision procedure for emptiness of tree automata, whose running time is polynomial in the number of states and exponential in the number of accepting pairs ([EJ88, PR89]), satisfiability of formulas in these logics can be decided in deterministic exponential time, in the case of Δ-PDL, μ-calculus, etc., and in deterministic doubly exponential time, in the case of CTL*.

5 Concluding Remarks

All the classes of automata we have discussed have the same expressive power. They define the class of ω-regular languages. Nevertheless, translating between the different classes can be quite expensive. Our results bring up the question of the complexity of the translations between the different classes. For example, we have shown that the translation from NB to DR is inherently exponential. In Figure 1 we represent a map of the translations between the different classes. We label an edge, representing a translation, by $\lfloor f \rfloor$, for f some complexity function, to denote a *lower bound* of f on the translation. Similarly, a label of $\lceil f \rceil$ denotes an *upper bound* on the translation. A label of $[f]$ denotes that f is a tight bound (both upper and lower) on the translation. Some of the results represented in the diagram are established in [SV89].

We hope that our results contribute to a better understanding of the intricacy of ω-automata, and prove useful for decision procedures for various logics.

Acknowledgments

The author thanks his advisor, Amir Pnueli, for his encouragement and many fruitful discussions on this research.

Moshe Vardi initiated this research by a most illuminating mini-course on ω-automata he presented at the Weizmann Institute. He suggested the problems and helped in clarifying the solutions. Without him this work would not have started, progressed, or ended.

Indispensable was the help of Rafi Heiman, whose signature at the bottom of a proof is more valuable than a Q.E.D..

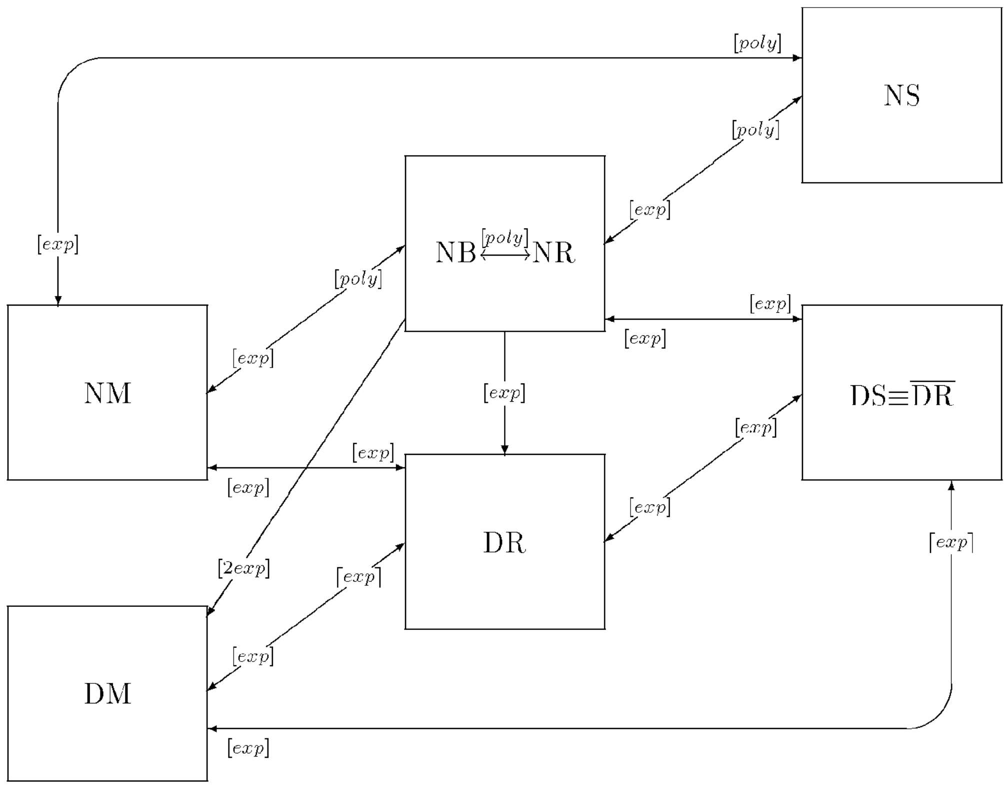


Figure 1: A map

Noam Nisan helped in the complexity evaluation of the determinization construction.

Which leaves open the question of what is the author's contribution to the paper.

A

Proof of Lemma 2 (Sketch):

For every $n > 0$ let $\mathcal{E}_n = [1..n]$, $\mathcal{S}_n = 2^{\mathcal{E}_n}$ and $\Sigma_n = \mathcal{S}_n \cup \mathcal{E}_n$. Consider the language:

$$X_n = \{\sigma = Y_0, y_0, Y_1, y_1, \ldots \mid \text{ for all } i, Y_i \in \mathcal{S}_n, y_i \in \mathcal{E}_n \text{ and } y_i \in Y_i\}\,.$$

Every X_n can be accepted by an NB of linear size: $\mathcal{A}_n = \langle \Sigma_n, \mathcal{E}_n \cup \{q_0\}, q_0, \delta, \{q_0\}\rangle$, where $\delta(q_0, Y) = Y$ for $Y \in \mathcal{S}_n$, and $\delta(y, y) = \{q_0\}$ for $y \in \mathcal{E}_n$. For all other values, δ gives the empty set.

On the other hand, let $\mathcal{D} = \langle \Sigma, Q', q_0', \delta', \mathbf{F}\rangle$ be a DM that accepts X_n. We prove that $|\mathbf{F}| \geq 2^{2^n}$. For every $R \in 2^{\mathcal{S}}$, $R = \{Y_1, \ldots, Y_h\}$, consider some word $\sigma_R = (Y_0, y_0, \ldots, Y_h, y_h)^\omega$ such that for each i, $1 \leq i \leq h$, $y_i \in Y_i$. This word is accepted by $\mathcal{D}$. We claim that for different such R's the infinty sets of the $\mathcal{D}$-run over σ_R are different. Indeed, the states of $\mathcal{D}$ can be partitioned according to the last letter read. For $R, R' \in 2^S$ such that there exists $Y \in R$, $Y \notin R'$, the infinity set of the accepting run for σ_R has a nonempty intersection with the set of $\mathcal{D}$-states that corresponds to Y, while the infinity set of the $\mathcal{D}$-run of $\sigma_{R'}$ does not.

References

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

[Büc60] J. R. Büchi. Weak second order arithmetic and finite automata. *Zeitschrift für Math. Logik und Grundlagen Math.*, 6:66–92, 1960.

[Büc62] J. R. Büchi. On a decision method in resricted second order arithmetics. In E. Nagel et al., editor, *Proc. International Congr. on Logic, Method. and Phil. of Sci., 1960*, pages 1–12. Stanford Uni. Press, 1962.

[Büc73] J. R. Büchi. The monadic theory of ω_1. In *Decidable Theories II, Lecture Notes in Mathematics 328*, pages 1–128. Springer, 1973.

[Cho74] Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and System Science*, 8:117–141, 1974.

[CY88] C. Courcoubetis and M. Yannakakis. Verifying temporal properties of finite-state probabilistic programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 338–345, 1988.

[Eil74] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[EJ88] A. E. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 328–337, 1988.

[EJ89] A. E. Emerson and C. Jutla. On simultaneously determinizing and complementing ω-automata. In *Proc. 4th IEEE Symp. on Logic in Computer Science*, 1989. To appear.

[ES84] A. E. Emerson and P. A. Sistla. Deciding full branching time logic. *Information and Control*, 61:175–201, 1984.

[GH82] Y. Gurevich and L. Harrington. Trees, automata, and games. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 60–65, 1982.

[McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.

[Mic88] M. Michel. Complementation is more difficult with automata on infinite words. Manuscript, CNET, Paris, 1988.

[Mul63] D. E. Muller. Infinite sequences and finite machines. In *Proc. 4th IEEE Symp. on Switching Circuit Theory and Logical Design*, pages 3–16, 1963.

[Pec86] J. P. Pecuchet. On the complementation of büchi autamata. *Theoretical Computer Science*, 47:95–98, 1986.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symp. on Principles of Programming Languages*, pages 179–190, 1989.

[Rab69] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Trans. AMS*, 141:1–35, 1969.

[Rab72] M. O. Rabin. Automata on infinite objects and church's problem. Regional Conf. Ser. Math., 13, AMS, Providence, Rhode Island, 1972.

[RS59] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research*, 3(2):115–125, 1959.

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988. An extended version to appear in Journal of Computer and System Science. This version

[Sch73] M. P. Schützenberger. Sur les relations rationelles fonctionelles. In M. Nivat, editor, *Automata, Languages and Programming*, pages 103–114. North-Holland, 1973.

[Str82] R. S. Streett. Propositional dynamic logic of looping and converse is elementary decidable. *Information and Control*, 54:121 141, 1982.

[SV89] S. Safra and M. Y. Vardi. On ω-automata and temporal logic. In *Proc. 21th ACM Symp. on Theory of Computing*, 1989. To appear.

[SVW87] A. P. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for büchi autamata with application to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.

[TB73] B. A. Trachtenbrot and Y. M. Barzdin. *Finite Automata Behavior and Synthesis.* North-Holland, 1973.

[Tho81] W. Thomas. A combinatorial approach to the theory of ω-automata. *Information and Control*, 48:261–283, 1981.

[Var] M. Y. Vardi. Personal communication.

[Var85] M. Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *Proc. 26th IEEE Symp. on Foundations of Computer Science*, pages 327–338, 1985.

[VS85] M. Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of program. In *Proc. 17th ACM Symp. on Theory of Computing*, pages 240–251, 1985.