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 essentially 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 [Buc62] 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 empty).

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.

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 [Buc73,Eil74,Cho74,Rab72,TB73]. 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]), infinitary 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 Streett 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, construct another automaton that accepts the complementary language. Büchi proved that his automata are closed under complementation [Buc62], but his proof was not completely constructive. Several explicit constructions were given [Buc73,Cho74,McN66,Sie70]. All of these involve a doubly exponential blow-up. An exponential construction was later found [SVW87] (see also [Pec86]). 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, the resulting Rabin automaton has $2^{O(n \log n)}$ states and n pairs. This construction can be shown to be essentially optimal.

Using this determinization construction, the natural reduction of the satisfiability problem for Δ-PDL, μ-calculus, etc., to the emptiness problem for Streett tree automata, results in a tree automaton whose size is exponential. Furthermore, since the number of pairs in the determinized Rabin automaton is linear, the number of pairs in the resulting Streett tree automaton is also linear. Using a decision procedure for emptiness of Streett tree automata, whose running time is polynomial in the number of states and exponential in the number of accepting pairs ([EJ88], see also [PR88]), 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 claimed in [Var85]. Furthermore, a partial determinization (Corollary 3) 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].

Similarly, using the small number of accepting pairs in the determinized automaton, and a simple conversion from the complement of a deterministic Rabin automaton to a Büchi automaton which is exponential only in the number of accepting pairs (Lemma 4), we give an alternative simple complementation construction (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 this construction is essentially optimal.

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* $\delta: Q \times \Sigma \to \mathbf{P}(Q)$, and an *acceptance condition* C.

A sequence of states, $\xi \in Q^\omega$, is an *$\mathcal{A}$-run* over a word $\sigma \in \Sigma^\omega$, iff $\xi_0 = q_0$ 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 inifinitary 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.

The *size* of an automaton, denoted $|\mathcal{A}|$, is the sum of the number of its states and the size of its acceptance condition (as a set). For Büchi automata we may neglect the acceptance condition.

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 $\bigwedge_i L_i \rightarrow U_i$.

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.

We summarize the acceptance conditions in the following table:

	Syntax	Semantics
B	$F \subseteq Q$	$\inf(\xi) \cap F \neq \phi$
M	$\mathbf{F} \subseteq \mathbf{P}(Q)$	$\inf(\xi) \in \mathbf{F}$
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	$\bigwedge_i L_i \rightarrow U_i$	$\forall i$: $\inf(\xi) \cap L_i = \phi \vee$ $\inf(\xi) \cap U_i \neq \phi$

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 (infinitary) 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 incomplete variant of the subset construction. 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 $\mathcal{A}$.

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. However, it is possible that, although there is an accepting $\mathcal{A}$-run, there is some integer $k \geq 0$, such that at each step, there exists a state q such that all runs reaching q at this step, visited at most k times an accepting state. This causes the green light to flash at most k times. Thus, the suggested acceptance convention may reject some words accepted by $\mathcal{A}$.

Note that if such a scenario occurs, then following the first k green flashes, there is always an unmarked state. Consequently, 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. After some finite time that 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, after some finite time, 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.

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

$\mathcal{D}$ accepts a word σ, if, during the run over σ, one of the nodes becomes infinitely often green.

In the actual construction we present, every accepting state is immediately added to one of the sons, thus there is no need for marking. Every state that appears in any of the sons is considered marked.

Theorem 1 *The determinization of* NB *into* DR *is exponential. More precisely, for any nondeterministic Büchi automaton of size n, there exists an equivalent deterministic Rabin automaton with* $2^{O(n \log n)}$ *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 labelled ordered trees. We introduce some notation concerning such trees.

An *ordered tree* is a structure $T = \langle N, r, p, S \rangle$, consisting of the following elements:

N – A set of nodes. The names of the nodes are taken from some global 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 $v \in N - \{r\}$, its parent $p(v) \in N$.

S – The *sibling ordering* relation, a partial ordering relation. We require that if $p(v) \neq p(v')$ then $(v, v') \notin S$, while if $p(v) = p(v')$ then either $(v, v') \in S$ or $(v', v) \in S$.

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

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

The sibling ordering can be extended to a larger ordering holding between any two nodes which are not ancestral, as follows: we say that v_1 is to the *left* of v_2 (or v_2 is to the right of v_1), if some ancestor of v_1 is an older brother of some ancestor of v_2.

A *labelled ordered tree* consists of a tree T and a mapping $l: N \to L$ 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 NB 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, $\mathcal{D} = \langle \Sigma, Q', q_0', \delta', C \rangle$. The states of the DR, Q', are ordered trees whose nodes are labelled with subsets of Q, and such that the following conditions are satisfied:

1. The union of the labels of the sons of a node v is a proper subset of the label of v, and

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

In addition, we associate with each node a color that can be either white or green.

We call a state $q \in Q$ *specific* to a node $v \in N$, if q belongs to v, and does not belong to any node which is not an ancestor of v. The above conditions, 1 and 2, imply that any state that appears in the tree is specific to a single node v (and appears exactly along the path from the root to v). In particular, if v is an older brother of v' then their labels are disjoint.

The initial state, q_0', is the tree of the single node whose value is the set of initial states and is colored white.

The deterministic transition function δ' transforms a labelled tree of the above form, given an input $a \in \Sigma$, by performing the following sequence of actions:

1. Set the color of all nodes to white.

2. For every node v with label $\hat{Q} \subseteq Q$, such that $\hat{Q} \cap F \neq \phi$, create a new node v', such that v' becomes the youngest son of v, and label it with $\hat{Q} \cap F$.

3. For every node v with label $\hat{Q}$, replace $\hat{Q}$ by the label $\delta(\hat{Q}, a)$ $(= \bigcup_{q \in \hat{Q}} \delta(q, a))$.

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

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

   Thus, a state which, as a result of steps 1–3, appears on more than one path, is removed from all but the leftmost path on which it appears.

5. Remove all nodes with empty labeles.

   Note that since the label of each node is contained in the label of its parent, if v is retained after step 5, and $v \neq r$, then also $p(v)$ is retained.

6. For every node v whose label is equal to the union of the labels of its sons, remove all the descendants of v and color v green.

We prove later that during the run over a word σ, some node $v \in \mathcal{V}$ turns green infinitely many times iff there is an accepting $\mathcal{A}$-run over σ.

We observe that the size of N at each stage is bounded by $n = |Q|$. Because of the requirements 1 and 2 holding over our trees, any node $v \in N$ must have at least one state which is specific to v (and to no other node in N). Consequently, N can contain at most n nodes.

The process of applying δ' to a tree T to obtain a successor $T' = \delta'(T, a)$, 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 5 and 6 we remove nodes. There are two possible policies for managing the names of nodes.

One policy assumes an infinite vocabulary $\mathcal{V}$, (say, all the natural numbers), and on creation of a new node takes a name which has never been used before. The acceptance condition we previously mentioned, i.e., that some node $v \in \mathcal{V}$ turns green infinitely often, corresponds to this policy. Unfortunately, this policy cannot guarantee a bounded representation for the set of all the trees generated in a run, even though each of them has at most n nodes.

A second policy assumes a finite vocabulary $\mathcal{V}_f$, such that $|\mathcal{V}_f| = 2n$ (e.g., the integers $1, ..., 2n$). In the expansion step, (step no. 2) we use any name in $\mathcal{V}_f - N$ to create the new node. Obviously, this can cause N to expand to at most $2n$. Then, in the construction steps, no. 5 and 6, some nodes are removed and leave a set of nodes N of size at most n. Now we can claim a bound on the representation since the vocabulary is finite. However, the acceptance condition now becomes the full Rabin condition. It requires that for some $v \in \mathcal{V}_f$,

$v \notin N$ only finitely many times
and
v is green infinitely many times.

Note that this leads to a Rabin condition with $O(n)$ pairs, namely all the pairs (L_v, U_v) for $v \in \mathcal{V}_f$. The set L_v are all the labelled trees in which v is green. The set U_v are all the labelled trees in which $v \notin N$.

It is not difficult to see that the two name-management policies lead to the acceptance of the same infinitary languages.

Correctness:

We prove the correctness of the construction for the first policy (which uses an infinitary vocabulary).

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

For two positions $0 \leq i < j$, 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. Define also $i_0 = 0$. Let $S_0 = \{q_0\}$, and $S_j \subseteq Q$, for $j > 0$, be the label of v at position i_j. The condition to make a node green (step 6), and the condition to create and maintain nodes (steps 2 and 3) ensure that for each $q \in S_{j+1}$, $j \geq 0$, there exists some $q' \in S_j$, and a run-segment 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-segment 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, $(q_0, 0), (q_1, 1), ...$, in the tree. By the construction of this tree, for each $j \geq 0$ there is a run-segment, 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.

The state of ξ, at every stage, is included in the root, thus the root is never removed (step 5). 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 4), 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 5). 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 $\mathcal{V}_f$ is $2^{O(n \log n)}$. We can represent the labels (the subsets) by a function from Q to $\mathcal{V}_f$, which for each state q gives the node v such that q is specific to v. The label of a node v is the set of states which are specific to v or to any descendant of v. The number of such functions is also $2^{O(n \log n)}$. Thus, the total number of ordered trees is $2^{O(n \log n)}$.

We have already observed that the number of accepting pairs is of order $O(n)$ (a slightly more careful construction results in the number of pairs being exactly n).

This completes the proof of Theorem 1. □

We note that a modification of an argument in [Mic88] can be used to show a $2^{O(n \log n)}$ lower bound for determinization (into DR), thus the above construction is optimal.

Note also that on our construction the size of the acceptance condition is linear; 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.* □

4 Applications

The following section describes some applications of Theorem 1.

Partial Determinization

Vardi ([Var85]) investigated automatic verification of probabilistic concurrent finite state programs; given such a program and a specification, 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 3 *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. □

Complementation

One of the important properties of ω-automata, established in [Buc62] 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 $2^{O(n \log n)}$.

First, we mention the following result:

Lemma 4 *([Var]) For any Streett automaton with n states and h accepting pairs, there exists an equivalent nondeterministic Büchi automaton of size $n \cdot 2^{O(h)}$.*

Proof: Given an NS, $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \bigwedge_i L_i \rightarrow U_i \rangle$, we construct an equivalent NB, whose states have the form of a triple $\langle q, S_1, S_2 \rangle$. The first element is a state of $\mathcal{A}$, which is manipulated by the transition relation orthogonally to the other two elements. The second and third elements contain either a special value, representing an initial stage, or contain two sets $S_1, S_2 \subseteq [1..h]$, where h is the number of accepting pairs of $\mathcal{A}$. The automaton guesses when the finite prefix of a run is over by choosing nondeterministically, at each point, either to stay in the initial stage, or to set $S_1 = S_2 = \phi$. After initialization, the subsets are used to remember which of the L_is and U_is were visited, by placing i into S_1 on each visit to some $q \in L_i$, and placing i into S_2 on each visit to some state $q \in U_i$. Whenever $S_1 \subseteq S_2$, we reset S_2 to ϕ. A run is accepting if S_2 is infinitely often reset to ϕ. □

Corollary 5 *It follows that an* NS *with logarithmic number of accepting pairs can be converted into a polynomial size* NB. □

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

Proof: Use the determinization construction to get a DR of size $(2^{O(n \log n)}, 2n)$. Interpret it as DS to get the complement, and then use the last conversion to get an NB of size $2^{O(n \log n)} \cdot 2^{O(n)} = 2^{O(n \log n)}$. □

The following lemma shows that an exponential blow-up for complemetation is unavoidable.

Lemma 7 *Complementation of* NB *is at least exponential.* □

This leaves a gap, of a $\log n$ factor in the exponent, between the lower and upper bounds for complementation of NB. A recent result by Michel [Mic88] gives a $2^{O(n \log n)}$ lower bound, thus our complementation construction is essentially optimal.

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 Streett tree automata ([Str82,ES84,VS85]). Using the 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 Streett tree automaton with exponential number of states, and polynomial number of accepting pairs. In the case of CTL*, the resulting Streett tree automaton has doubly exponential number of states and exponential number of pairs.

Using a decision procedure for emptiness of Streett tree automata, whose running time is polynomial in the number of states and exponential in the number of accepting pairs ([EJ88,PR88]), 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 function. Results concerning the diagram will be described in a future paper.

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.

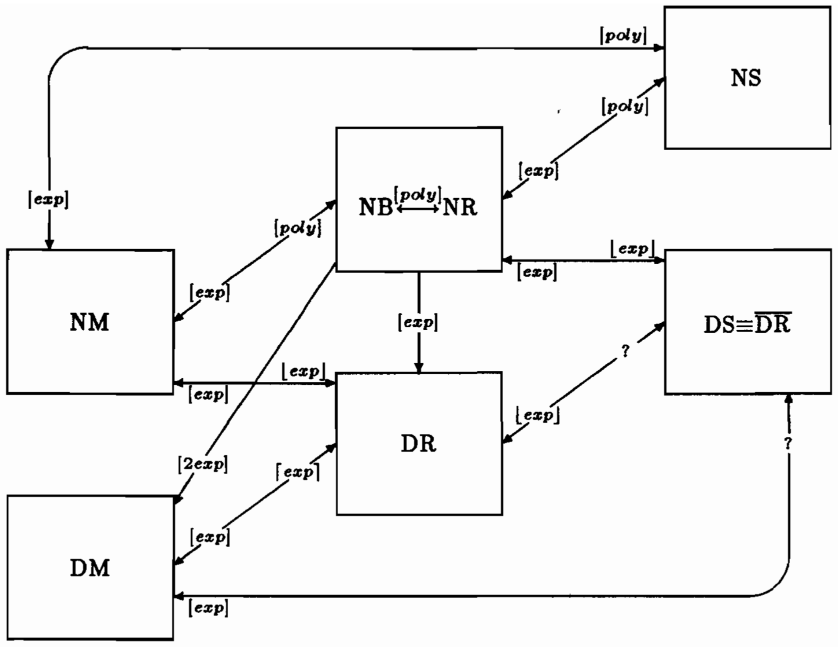

Figure 1: A map of the dungeon

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

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.

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.

[Buc62] J. R. Büchi, On a decision method in resricted second-order arithmetics, *Proc. Int'r Congr. on Logic, Method and Phil. of Sci., 1960*, Stanford Uni. Press, 1962, pp. 1–12.

[Buc73] J. R. Büchi, *Decidable Theories II — The Monadic second-order theory of* ω_1, pp. 1–128, Volume 328 of *Lecture Notes in Mathematics*, Springer-Verlag, 1973.

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

[CY88] C. Courcoubetis and M. Yannakakis, Verifying temporal properties of finite-state probabilistic programs, *Proc. 29th IEEE Symp. on Foundations of Computer Science*, October 1988. These Proceedings.

[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, *Proc. 29th IEEE Symp. on Foundations of Computer Science*, October 1988. These Proceedings.

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

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

[McN66] R. McNaughton, Testing and generating infinite sequences by a finite automaton, *In-*

formation and Control **9**, 1966, pp. 521–530.

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

[Mul63] D. E. Muller, Infinite sequences and finite machines, *Switching Circuit Theory and Logical Design: Proc. Fourth Ann. Symp.*, Inst. of Electrical and Electronic Engineers, New York, 1963, pp. 3–16.

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

[PR88] A. Pnueli and R. Rosner, On the synthesis of a reactive module, 1988. Submitted to the 16th POPL.

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

[Rab72] M. O. Rabin, *Automata on Infinite Objects and Church's Problem*, Regional Conf. Ser. Math., 13, Amer. Math. Soc., Providence, Rhode Island, 1972.

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

[Sie70] D. Siefkes, *Decidable Theories I — Büchi's Monadic second-order successor arithmetics*, Volume 120 of *Lecture Notes in Mathematics*, Springer-Verlag, 1970.

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

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

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

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

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

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