Complexity of Automata on Infinite Objects

Thesis for the Degree of
Doctor of Philosophy

by
Shmuel Safra

Submitted to the Scientific Council of
The Weizmann Institute of Science
Rehovot, Israel

March 1989

Abstract

We investigate in this thesis problems concerning the complexity of translation among, and decision procedure for, different types of finite automata on infinite words (ω-automata). An ω-automaton is the same as usual finite automata over finite strings but it accepts or rejects infinite strings. It may be either deterministic or nondeterministic, and may have different types of acceptance condition.

Our main result is a new, simpler, determinization construction that yields a single exponent upper bound for the translation of any Büchi nondeterministic ω-automaton into a deterministic ω-auomaton. This construction is optimal. We also look at the complexity of the complementation problem for different types of ω-automata, and, among other results, obtain an exponential complementation for Streett ω-automata.

These results can be used to improve the complexity of decision procedures for different logics that use automata-theoretic techniques.

Acknowledgement

First and foremost, I owe my deepest gratitudes to my advisor, Amir Pnueli, for all the work presented in this thesis, and for any scientific work I might do in the future. He gave me the oportunity to stop doing what I was doing and start doing better things. His deep intuition, sharp observations and methodological attitude made this work possible — I could not have expected a better teacher. On a personal note, one of the main disadvantages of finishing my Ph.D. is not being able to walk into Amir's office, have a laugh at ourselves in general and the world in particular, while insisting on finding the intuition behind some technical matter.

It would be hard to exaggerate the influence of Moshe Vardi on me and on this work. He first introduced me to the subject of ω-automata, in a most illuminating mini-course at the Weizmann institute. Later (serving as one half of my Ph.D. committee) he insisted that I read and understand a few major papers on the subject. I didn't read them all, and I didn't understand any of them, but I thought I understood one, and this misunderstanding led to the main result of this thesis. We also had a fruitful collaboration; indeed, many of the results that appear here are joint work with Moshe.

David Harel is not only the other half of my Ph.D. committee, but also his courses have been for me an introduction to computer *science*. Not to mention his jokes.

I'll mention only very few of the other people that had part in the strange course of events that led to me writing this thesis. Going back to undergraduate studies, the one to be blamed for me ending up in computer science is Noam Nisan; we started studying together, and, not being able to figure out the instruction for filling out the program-of-studies, I had to copy his. Lenore Zuck enlightened me with the possibility of changing my direction for my Ph.D., in what proved to be my best decision. Rafi Heiman have the rare combination of qualities; on one hand he would not agree on a proof unless it is *correct*, and on the other hand, he has the endless patience to listen to me while I mumble some "proof". I must thank some of the students at the Weizmann Institute, especially in the neighborhood of my office, for it was my main activity during work hours to disturb their work; among them are Avital, Avshalom, Irad, Jon, Moshe, Oded, Rafi, Roni and Uri. Occasionally we discussed work. Special thanks are due to my M.Sc. adviser for leading me to meet and work with many of the above mentioned people.

And to all my friends who are not mentioned here (and might prefer it that way), I owe many thanks for putting up with me through many years (and believe me it is not easy); Thanks, and sorry but you won't have me to put up with for a while.

"Herr Dr.! I don't know what to do, I keep talking to myself"
Freud: "How long has this been going on?"
".. 30 years"
Freud: "When does it usually occur?"
".. most of the time.."
Freud: ".. Well, it seems there is nothing that can be done.."
".. but Dr.! You don't know what a pain in the ass I am..."

Contents

Introduction

This thesis is organized as follows: in the rest of the Introduction we present a short survey on the theory of automata on infinite objects and their complexity. There are other aspects of automata on infinite objects (e.g., algebraic or set theoretic topology) which we do not discuss; the interested reader is referred to [Tho89]. In Chapter 1 we discuss the complexity of determinization of ω-automata. Chapter 2 is devoted to the complexity of complementation of various types of ω-automata. In Chapter 3 we show some applications of our results, mainly concerning decision procedures for logics of program. In Chapter 4 we summarize the known results and the open problems.

0.1 S1S and ω-Automata

Automata on infinite words were introduced by Büchi in [Büc62, Büc60]. He was interested in the decidability of S1S, the monadic second order theory of one successor. A decision procedure for a logic decides, given a formula in this logic, whether it is valid (satisfiable). A formula is valid if it is true in all the possible models, and it is satisfiable if there exists a model in which it is true.

The domain of S1S is $\mathbb{N}$, the set of non-negative integers. A variable can be one of x_1, x_2, ..., in which case it ranges over $\mathbb{N}$, or it can be one of X_1, X_2, ..., in which case it ranges over sets of elements from $\mathbb{N}$. We have a function symbol s (*successor*) that can be applied to domain variables, $s(n) = n + 1$. An atomic formula is either an equality of two terms or inclusion of a term in a set variable. General formulas are the closure of atomic formulas under the usual Boolean connectives and quantification over variables. The meaning of the Boolean connectives and quantification is defined in the usual way.

An interpretation of a tuple of set variables $X_1, ..., X_h$, is an assignment of a set $S_i \subseteq \mathbb{N}$ to each variable X_i. Such an interpretation can be represented as an infinite word:

$$\sigma = \sigma_0, \sigma_1, ...$$

over the alphabet $\Sigma_h = 2^{[1..h]}$ ($[1..h]$ is the set $\{1, 2, ..., h\}$). This representation identifies a non-negative integer $n \geq 0$ as a member of S_i, $i = 1, ..., h$, iff $i \in \sigma_n$. Consider an S1S formula $\varphi(X_1, ..., X_h)$ ($X_1, ..., X_h$ are the free variables of φ). Such a formula defines a language, L_φ, of infinite words over Σ_h, consisting of all the words that correspond to models (interpretations) satisfying φ.

Büchi showed that for any formula φ, L_φ can be accepted by a nondeterministic finite automaton appropriately extended to accept 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.

In order to establish that claim, Büchi proved that the atomic formulas of S1S are reducible to automata, and, inductively on the structure of the formula, that if φ_1 and φ_2 are reducible to automata, then $\varphi_1 \vee \varphi_2$, $\exists X{:}\, \varphi_1$ and $\neg\varphi_1$ are also reducible to automata. The case of disjunction is dealt with by an automaton whose first step is to choose nondeterministically between behaving like $\mathcal{A}_{\varphi_1}$ and behaving like $\mathcal{A}_{\varphi_2}$. The case of $\exists X{:}\, \varphi$ is dealt with by an automaton that, at each step, reading σ_n, guesses whether $n \in X$, and then lets $\mathcal{A}_\varphi$ perform a single step on the letter σ'_n which also reflects the effect of the decision. The hard part of the proof is the negation case, i.e., that if L_φ is accepted by an automaton then there is an automaton accepting $L_{\neg\varphi}$. In order to show that, Büchi proved, using the infinite Ramsey Theorem, that for any Büchi automaton there exists another automaton that accepts the complementary language.

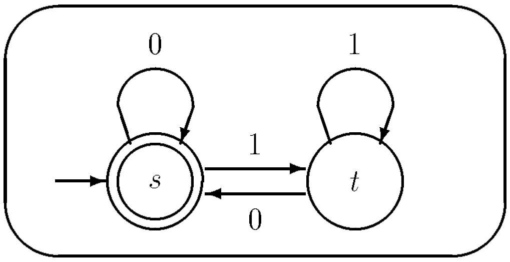
The size of the complementary automaton, according to Büchi's construction, is doubly exponential in the size of the original automaton. Sistla, Vardi and Wolper ([SVW87]) presented an exponential complementation procedure for Büchi automata. For any Büchi automaton of size n they construct a complementary Büchi automaton of size $\mathrm{O}(2^{4n^2})$. In Corollary 2.2 we slightly improve this upper bound to $2^{\mathrm{O}(n\log n)}$. It follows from a recent result of Michel ([Mic88]) that this construction is optimal up to a constant factor in the exponent.

In order to check if an S1S formula φ is satisfiable, we can now construct the automaton $\mathcal{A}_\varphi$ that accepts L_φ and check if it is not empty (i.e., if it accepts some word). It is not difficult to see that the emptiness problem for Büchi automata is decidable (actually it is in nondeterministic LOGSPACE).

Thus, Büchi showed that S1S is decidable, but his procedure is not elementary recursive. For a formula of size n the procedure might run for $\left.2^{2^{\cdot^{\cdot^{2^n}}}}\right\}\mathrm{O}(n)$ time. Meyer ([Mey75], see also [Sto74]) showed that this is essentially the best possible general procedure, that is, the decidability of S1S is not elementary recursive.

0.2 Deterministic ω-Automata

While Büchi was interested in the nondeterministic version of his automata, it is possible to study also the deterministic version. It turns out that deterministic Büchi automata are less expressive than nondeterministic Büchi automata. Moreover, deterministic Büchi automata are not closed under complementation. We demonstrate this by the following example. Consider the language $L = (01^*)^\omega$, i.e., the set of all the words in which 0 appears infinitely many times. This language is accepted by the following deterministic Büchi automaton:

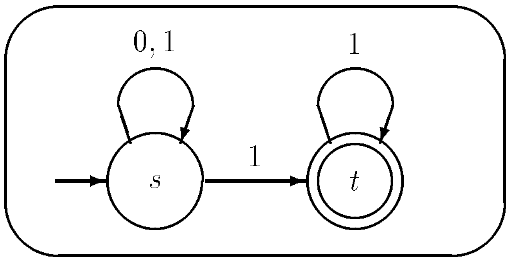


$\mathcal{D}$

Whenever $\mathcal{D}$ reads 0 it goes to s, and when it reads 1 it goes to t. $\mathcal{D}$ visits the accepting state s infinitely many times iff 0 appears infinitely many times in the input word.

The complementary language, $\overline{L} = (0+1)^*1^\omega$, i.e., the set of words with only finitely many 0's, is not acceptable by a deterministic Büchi automaton. To see that, assume by contradiction that there exists such an automaton $\overline{\mathcal{D}}$, and consider the word 1^ω. Since it is accepted by $\overline{\mathcal{D}}$, there is some i_1 such that the run of $\overline{\mathcal{D}}$ over 1^{i_1} leads to an accepting state $f_1 \in \overline{\mathcal{D}}$. Now, consider the word $1^{i_1}01^\omega$. Since it is accepted, there exists an i_2 such that the run of $\overline{\mathcal{D}}$ over $1^{i_1}01^{i_2}$ leads to an accepting state, $f_2 \in \overline{\mathcal{D}}$. But f_2 must be different from f_1, otherwise the word $1^{i_1}(01^{i_2})^\omega$ would have been accepted by $\overline{\mathcal{D}}$ (its run would visit infinitely many time the accepting state $f_2 = f_1$). We can continue this argument and construct an infinite sequence of states $f_1, f_2, ...$ that are all different from each other. Thus the number of states of $\overline{\mathcal{D}}$ cannot be finite, contradicting the assumption that $\overline{\mathcal{D}}$ is a finite-state automaton.

The language $\overline{L}$ can be accepted by a nondeterministic Büchi automaton (since they are closed under complementation). The following is an example for such an automaton:



$\mathcal{A}$

$\mathcal{A}$ stays in s and guesses nondeterministically when the last 0 appears, then it moves to t, and from then on only 1 can appear.

It follows that, unlike the finitary case, Büchi automata cannot be determinized, and there are simple ω-languages that cannot be expressed by deterministic 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 based on the infinity set, this is obviously the strongest possible condition.

This left open the question concerning the expressive power of deterministic Muller automata, which are equipped with a stronger acceptance condition than deterministic Büchi automata.

Considering our example above, we can see that the language $\overline{L}$ has a deterministic Muller automaton, which is the automaton $\mathcal{D}$ above except that the only accepting infinity set is $\{t\}$. Thus, a word is accepted iff, from some point on, $\mathcal{D}$ stays forever in t, i.e., there are no 0's beyond that point.

This shows that deterministic Muller automata are more expressive than deterministic Büchi automata. But is it the case that deterministic Muller automata are as expressive as nondeterministic Büchi automata?

0.3 McNaughton's Determinization Theorem

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 that 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). We may denote this condition by $\bigvee_i(L_i \wedge \neg U_i)$. Streett ([Str82]) suggested the complementary condition to Rabin's condition, i.e., $\bigwedge_i(L_i \rightarrow U_i)$. Both these conditions provide important normal forms of ω-automata.

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 of correctness are quite complicated, and in view of the importance of this result, many authors have provided alternative constructions ([Büc73, Eil74, Cho74, Rab72, TB73, Sch73, Tho81]). However, none of these constructions yielded a better complexity and most of them are even triply exponential.

Thus, after settling the question of expressiveness of the two types of automata, we are left with the question of the complexity of the translations between them. The direction from Rabin automata to nondeterministic Büchi automata is simple and polynomial, but is there an exponential determinization, i.e., is there an exponential translation from nondeterministic Büchi automata to deterministic Rabin automata?

Emerson and Sistla ([ES84]) developed a special determinization process, which is only singly exponential, for the case of automata associated with linear temporal logic. The success of this construction is due to the special property (reverse deterministic) of such automata.

The main result of this thesis (Theorem 1.1) 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 any 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 factor in the exponent.

McNaughton's Theorem gives an alternative proof to the closure of Büchi automata under complementation (as noted in [McN66]). Given a Büchi automata, we can use McNaughton's Theorem to convert it into a deterministic Muller automaton. To complement a deterministic Muller automaton, we only have to switch the good infinity sets with the bad ones. Then we convert the resulting Muller automaton into a nondeterministic Büchi automaton.

0.4 Rabin Tree Automata and S2S

Rabin ([Rab69, Rab70]) introduced automata on infinite trees. The input of such an automaton is an infinite tree, whose nodes are labeled by letters from some finite alphabet Σ, $T: \{0,1\}^* \to \Sigma$. (We will discuss only binary trees whose nodes can be identified with the words of $\{0,1\}^*$. Any tree with higher outdegree, including ω-branching, is representable as a binary tree.) 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., $\delta: Q \times \Sigma \to 2^{Q \times Q}$). A $\mathcal{T}$-run is a Q-tree, $\Gamma: \{0,1\}^* \to Q$, 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 or Muller. A $\mathcal{T}$-run Γ is defined to be accepting if the infinity set of every infinite path in Γ satisfies C.

It turns out that there are differences in expressive power between different types of tree automata. Unlike the case of ω-automata, the difference between Büchi condition and Rabin or Muller conditions already exists in the nondeterministic case, and deterministic tree automata are less expressive than nondeterministic automata for all of these acceptance conditions.

Rabin proved the parallel to Büchi's theorem on tree automata; that Rabin (Muller) tree automata are closed under boolean operations and quantification, and that the emptiness problem for these automata is decidable. Thereby, he was able to show that a more general logic than S1S, the monadic second-order theory of *two* successors, S2S, is decidable. The hard part of the proof, similarly to Büchi's proof, is to show that Rabin tree automata are closed under complementation, though proving the decidability of the emptiness problem is not as simple as for ω-automata.

Hossley and Rackoff ([HR72]) gave a different procedure for checking the emptiness of Rabin tree automata. Gurevich and Harrington ([GH82]) used a game-theoretic approach to prove both the decidability of the emptiness problem, and the closure under complementation of Rabin tree automata.

Both Rabin's and Gurevich and Harrington's proofs that tree automata are closed under complementation use McNaughton's determinization theorem in a crucial way.

Rabin's procedure for checking the emptiness of tree automata in [Rab69] is non-elementary, Hossley and Rackoff's procedure runs in doubly exponential time. Rabin improved the complexity of the decision procedure to exponential time in [Rab72], and Vardi and Stockmeyer ([VS85]) gave an NP decision procedure. Recently, independently, Emerson and Jutla ([EJ88]) and Pnueli and Rosner ([PR89]) gave a decision procedure for Rabin tree automata, whose running time is exponential in the number of accepting pairs, but only polynomial in the number of states. Pnueli and Rosner's proof is a simplification of Hossley and Rackoff's proof. This result together with the exponential determinization, has some applications as shown in chapter 3.

Considering the complexity of the complementation of tree automata, since both Rabin's and Gurevich and Harrington's proofs use McNaughton's theorem, they depend on the complexity of the determinization construction for ω-automata. Using the exponential determinization procedure, the complexity of Gurevich and Harrington's proof is still at least doubly exponential, while the only known lower bound is exponential (due to the lower bound for ω-automata). Resolving this gap might have many applications.

0.5 Basic Definitions and Notations

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.

We can extend the definition of δ to sets of states and finite words: for $S \subseteq Q$, $\sigma \in \Sigma^*$ and $a \in \Sigma$, we define $\delta(S, \lambda) = S$ (λ denotes the empty word), and inductively $\delta(S, \sigma a) = \{q \mid \exists q' \in \delta(S, \sigma) \text{ such that } q \in \delta(q', a)\}$.

For an infinite word $\sigma \in \Sigma^\omega$ and two positions $0 \leq i < j$, we denote by $\sigma[i, j)$ the finite word $\sigma_i, ..., \sigma_{j-1}$.

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., $\xi_{i+1} \in \delta(\xi_i, \sigma_i)$.

An $\mathcal{A}$-run ξ *visits* a state $q \in Q$ if there exists an i such that $\xi_i = q$. The run ξ *visits q infinitely many times* if there exist infinitely many different i's such that $\xi_i = q$. The *infinity set* of a run ξ is the set of states that are visited infinitely many times in ξ, $\inf(\xi) = \{q \mid \text{ size}(\{i \mid \xi_i = q\}) = \infty\}$.

An $\mathcal{A}$-run ξ is *accepting* if the infinity set of ξ satisfies C. 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.

The acceptance conditions described above can be viewed as special forms of formulas over the states of the automaton. The states are considered variables, the infinity set — an assignment to these variables, and the acceptance condition is satisfied if the formula is true under the assignment. By a set $S \subseteq Q$, $S = \{q_1, ..., q_h\}$, we denote the disjunction of the elements of S, $q_1 \vee ... \vee q_h$. Thus, Büchi condition is $F \subseteq Q$, Muller condition allows only formulas that specify explicitly all the good input settings, Rabin condition has the form $\bigvee_i (L_i \wedge \neg U_i)$, and Streett condition has the form $\bigwedge_i (L_i \rightarrow U_i)$. We may also consider the general form, i.e., any Boolean formula f over the variables ([EL85]).

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, E for Büchi, Muller, Rabin, Streett and Emerson-Lei, 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	$\bigwedge_i (L_i \rightarrow U_i)$	$\forall i$: $\inf(\xi) \cap L_i = \phi \vee$ $\inf(\xi) \cap U_i \neq \phi$
E	a formula f	$\inf(\xi) \models f$

The *size* of an automaton, is measured by specifying the number of states and the size of the acceptance condition (except for Büchi automata were the acceptance condition may be neglected). For example, a Rabin automaton with n states and h accepting pairs $(L_i, U_i,\ i = 1, ..., h)$ is described as having the complexity (n, h). Using this notation, our main result may be written as $\mathrm{NB}(n) \rightarrow \mathrm{DR}(2^{\mathrm{O}(n \log n)}, n)$, implying that there is a construction that translates a nondeterministic Büchi automaton with n states into a deterministic Rabin automaton with $2^{\mathrm{O}(n \log n)}$ states and n accepting pairs.

0.6 Folk Lemmas

We quote the following Lemmas without proof. None of them is very difficult to prove.

Lemma 0.1 $\mathrm{NR}(n, h) \rightarrow \mathrm{NB}(n + hn)$. □

Lemma 0.2 $\mathrm{NM}(n,h) \to \mathrm{NB}(n+hn^2)$. $\square$

Lemma 0.3 $\mathrm{NE}(n,m) \to \mathrm{NM}(n,2^n)$. $\square$

Corollary 0.4 $\mathrm{NE}(n,m) \to \mathrm{NB}(n^2 2^n)$. $\square$

Lemma 0.5 $\mathrm{DE}(n,m) \to \mathrm{DM}(2^n)$. $\square$

Lemma 0.6 *The emptiness problem for* NB *automata is* NL*-complete.* $\square$

Lemma 0.7 (König [Kön27]) *Any tree in which every node has finitely many offsprings, containing infinitely many nodes, contains an infinite path.* $\square$

Theorem 0.8 (Ramsey) *(The infinite version) For every finite partition of the pairs of integers, $P_1 \cup ... \cup P_h = \mathbb{N}^2$, there exists an* infinite *set of integers $S \subseteq \mathbb{N}$, such that $S^2 \subseteq P_i$ for some $i = 1, ..., h$.* $\square$

Chapter 1

Determinization of ω-Automata

In this chapter we discuss the determinization problem for ω-automata. The main result is a new determinization construction, from NB to DR, which yields only an exponential blow-up. Using this construction and a simple transformation from DS to DR (and vice versa), which is exponential in the number of accepting pairs, but linear in the number of states, we obtain the same complexity for determinization into DS. For determinization of NB to DM automata, however, we show a doubly exponential lower bound. This leaves all of these problems with tight bounds up to a constant factor in the exponent.

The results reported in this chapter appear in [Saf88].

1.1 An Exponential Determinization Construction (NB→DR)

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 of some of the problems, inherent in such subset constructions, and their solutions and then the formal construction and its proof.

Overview:

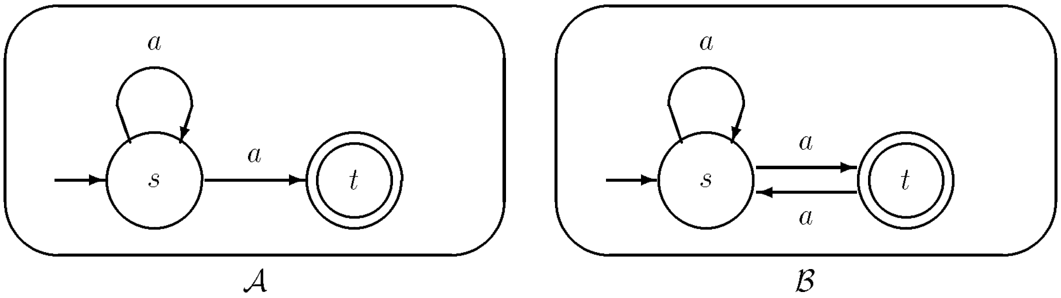
Assume a nondeterministic Büchi automaton $\mathcal{A}$. We describe a deterministic finite state automaton $\mathcal{D}$, that reads the (infinite) input sequentially, and simulates $\mathcal{A}$.

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 improved 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 improved subset construction to the states associated with the node. This construction is sound and complete.

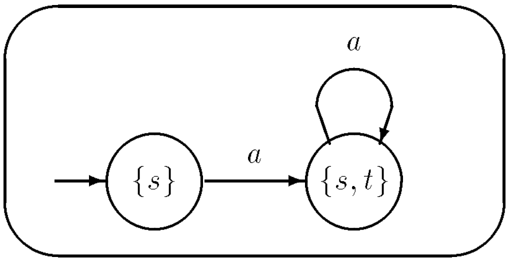
In the straightforward subset construction, the deterministic automaton, $\mathcal{D}$, maintains, at each step, the set of $\mathcal{A}$-states reachable by the prefix of the word read so far.

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 $\Sigma = \{a\}$):



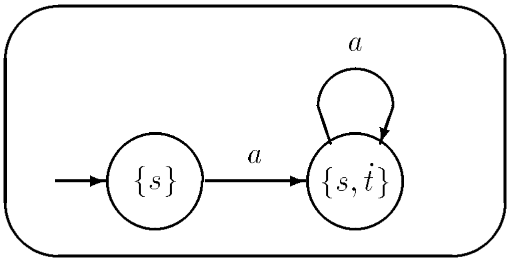
The automaton $\mathcal{A}$ does not accept any word (because, once the automaton enters t, it cannot continue), while $\mathcal{B}$ accepts a^ω (the run $(st)^\omega$ is accepting). Applying the subset construction to these two automata yields, in both cases, the automaton:



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.

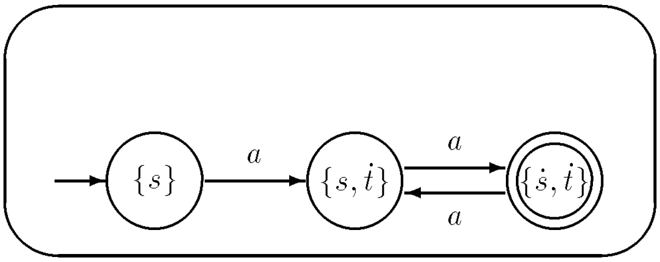
An improved construction tries to identify breakpoints in the course of reading a word σ and simulating the runs of $\mathcal{A}$ on σ. The initial stage is identified as a breakpoint. The next breakpoint occurs as soon as each reachable $\mathcal{A}$-state has a run that visited an accepting $\mathcal{A}$-state 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 the improved 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:

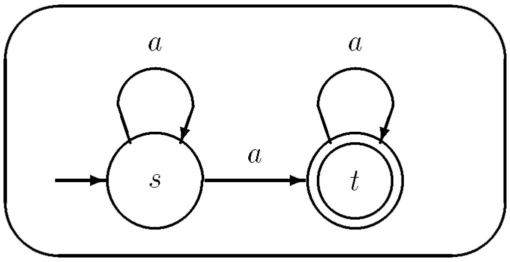


$\mathcal{B}'$

Note that here s is marked infinitely many times by the run which proceeds from t to s.

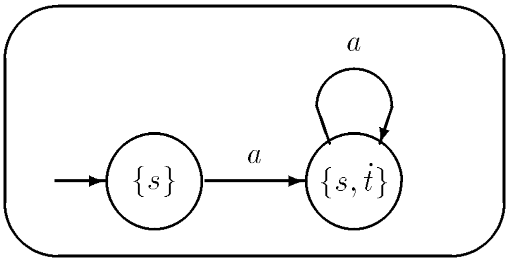
However, it is possible that, although there is an accepting $\mathcal{A}$-run, there exists, at each step, a reachable state q such that no run reaching q visited any accepting state. This causes each of the constructed subset states to contain an unmarked state, and hence the green light will never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



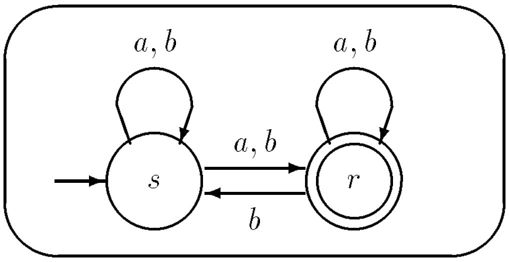
$\mathcal{A}_1$

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



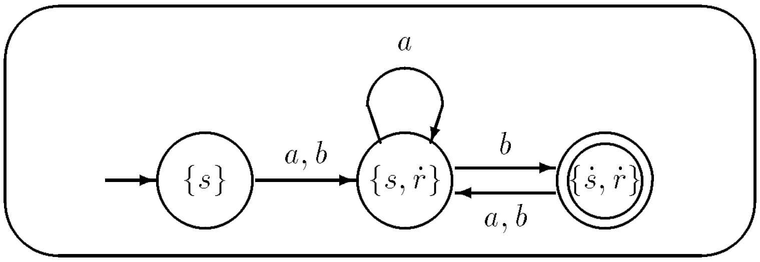
$\mathcal{A}'_1$

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:



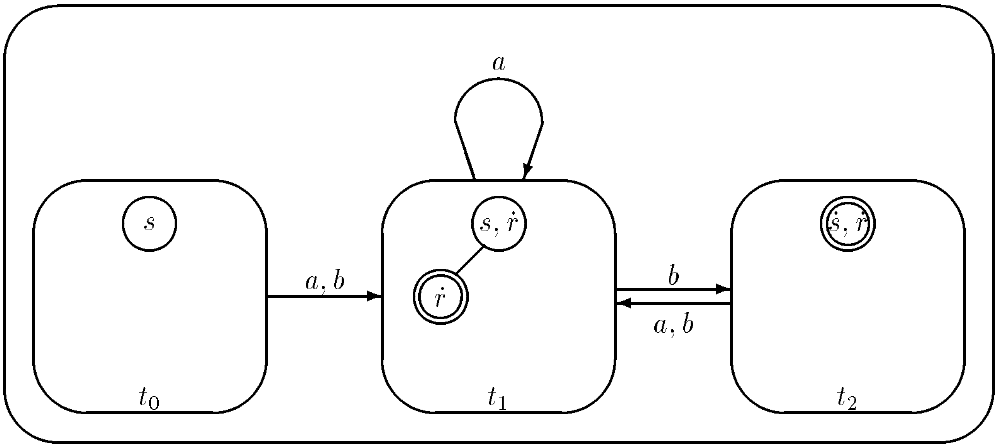
$\mathcal{A}'_2$

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 improved subset construction. If the green light does not flash at the root, $\mathcal{D}$ creates a son to the root node and copies the set of marked states into this new node. If, again, at the next step, there are some states which are marked in the root, but do not appear in the son, $\mathcal{D}$ creates a new son to the root and copies these states into the new 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 is contained in all the states of the run from a certain point on, and becomes infinitely often green.

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 $\sigma \in \Sigma^\omega$: if σ contains infinitely many b's, then the run visits t_2 infinitely many times, and σ is accepted because the root node turns green infinitely often. If, on the other hand, σ contains only finitely many b's, then beyond some point, the run is contained in t_1, and σ 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 descendants). 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.

Having gone through an informal description, we can continue with a more formal presentation of the construction.

Theorem 1.1 NB(n) $\rightarrow$ 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 node $d \in N - \{r\}$ a unique 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 $l: N \to L$ 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 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.

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

$c: V \to \{\text{white}, \text{green}, \text{red}\}$ – 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 each 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 to (T, l, c). In carrying out these actions, 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 trees.

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 $N \cap V$ 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 after step 5 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 steps 4 and 5 we remove nodes.

The acceptance condition requires that for some $v \in V$,
v is red ($c(v) = \text{red}$) 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 $0 \leq i < j$, 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 after the last time v is red. By the construction, it follows that for all positions $i \geq i_1$, 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 conditions 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 (Lemma 0.7), 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, $(q_0, 0), (q_1, 1), ...$, 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. A more careful analysis leads to the bound of $2^{O(n \log n)}$, as follows:

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 1.1. $\square$

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.

1.2 NB → DS

Theorem 1.1 shows the existence of an exponential determinization into DR. This raises the question whether this is true for determinization of NB to other deterministic automata. In this section, using Theorem 1.1 (and especially the small number of accepting pairs in the determinized DR), and a simple conversion from DR to DS, which is exponential only in the number of pairs, we obtain a determinization into DS, with roughly the same complexity as for DR. This has some applications as shown below (Corollaries 2.1 and 2.2).

Lemma 1.2 DS$(n,h) \rightarrow$ 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.*

Note that, since DS $= \overline{\text{DR}}$, this implies the same complexity for the translation of DR to DS.

Proof:

The construction: Given a DS, $\mathcal{D} = \left\langle \Sigma, Q, q_0, \delta, \bigwedge_{1 \le i \le h}(L_i \rightarrow U_i) \right\rangle$, we construct a DR, $\hat{\mathcal{D}} = \left\langle \Sigma, \hat{Q}, \hat{q}_0, \hat{\delta}, \bigvee_{1 \le i \le h+1}(\hat{L}_i \wedge \neg \hat{U}_i) \right\rangle$, 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 $g' \le h$; otherwise, $\pi' = \pi$.

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

Note that after $\hat{\mathcal{D}}$ reads 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 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, $r \ge j$ from some point on, and $g = j$ infinitely many times. Since $g = j$ infinitely many times, all the i's for which U_i is visited only finitely many times are eventually placed in π at a position earlier than j (i.e., $i = i_k$ for $k < j$), and by the construction, from then on, L_i is never visited. On the other hand, all the i's whose position is j or later are guaranteed to have infinitely many visits in U_i.

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, $r \ge j$ beyond that 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 of $\hat{\mathcal{D}}$ is $n \cdot h! \cdot (h+1)^2 < n \cdot 2^{h \log h}$ (for $h > 5$). $\square$

Corollary 1.3 NB(n) $\rightarrow$ DS($2^{O(n \log n)}, n+1$); *for any nondeterministic Büchi automaton with n states, there exists an equivalent deterministic Streett automaton, with* $2^{O(n \log n)}$ *states and* $n+1$ *accepting pairs.*

Proof: Use the determinization construction of Theorem 1.1 to obtain a DR of size $(2^{O(n \log n)}, n)$. Then use Lemma 1.2 to get a DS whose number of states is $2^{O(n \log n)} \cdot 2^{n \log n} = 2^{O(n \log n)}$, and that has $n+1$ accepting pairs. □

It is worthwhile mentioning that there is a strict hierarchy of ω-regular languages that corresponds to the DR automaton with the least number of accepting pairs that accepts the language ([Wag79, Kam85]). For every $i \geq 0$ there exists a language L_i that is accepted by a DR automaton with i accepting pairs, but there is no DR automaton with less than i accepting pairs that accepts L_i. There is also the complementary hierarchy corresponding to DS automata ([MP87]). Lemma 1.2 shows that the ith level of one hierarchy is contained in the $i+1$st level of the complementary hierarchy.

Lemma 1.2 also entails the same complexity for translation of (nondeterministic) tree automata:

Corollary 1.4 *For any nondeterministic Streett (Rabin) tree automaton with n states and h accepting pairs, there exists a Rabin (Streett) tree automaton with* $n \cdot 2^{h \log h}$ *states and* $h+1$ *accepting pairs.*

Proof: It is true, in general, that any transformation on deterministic ω-automata can be carried out on (nondeterministic) tree automata. □

For nondeterministic ω-automata there are better upper bounds.

1.3 NB → DM

Note that the above determinization constructions both yield a deterministic automaton with 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 1.5 *The determinization of* NB *into* DM *is at least doubly exponential.*

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

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

Every $\mathcal{L}_n$ can be accepted by an NB automaton of linear size:

$$\mathcal{A}_n = \langle \Sigma_n, [1..n] \cup \{q_0\}, q_0, \delta, \{q_0\} \rangle$$

where $\delta(q_0, Y) = \{y \in [1..n] \mid y \in Y\}$ for $Y \in \mathcal{S}_n$, and $\delta(y, y) = \{q_0\}$ for $y \in [1..n]$. For all other $q \in Q$ and $a \in \Sigma_n$, $\delta(q, a) = \phi$.

On the other hand, let $\mathcal{D} = \langle \Sigma_n, Q', q_0', \delta', \mathbf{F} \rangle$ be a DM that accepts $\mathcal{L}_n$, we prove that $|\mathbf{F}| \geq 2^{2^n}$. The states of $\mathcal{D}$ can be partitioned into 2^n disjoint sets, according the the last letter in the prefixes leading to the state. A state that has a finite word leading to it that ends with a letter in $[1..n]$, belongs to the equivalence class denoted by $[\circ]$. A state that has a prefix leading to it that ends with a letter $Y \in \mathcal{S}_n$ belongs to $[Y]$. It is not difficult to see that no state belongs to more than one equivalence class (there is a suffix separating any two prefixes that lead to different equivalence sets).

Now, for every $R \in 2^{\mathcal{S}_n}$, $R = \{Y_1, ..., Y_h\}$, let $\sigma_R = (Y_0 y_0 ... Y_h y_h)^\omega$ be such that $y_i \in Y_i$ for $i = 1, .., h$. Let F_R denote the (accepting) infinity set of $\mathcal{D}$ over σ_R. We can show that if $R \neq R'$ then $F_R \neq F_{R'}$. Indeed, w.l.o.g. there exists $Y \in R$, $Y \notin R'$, and F_R has a nonempty intersection with $[Y]$ while $F_{R'}$ does not. $\square$

Chapter 2

Complementation of ω-Automata

As noted in the Introduction, 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 an exponential complementation construction. For any NB automaton of size n they construct a complementary NB automaton of size $O(2^{4n^2})$. A more general 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, since it complements and determinize in one construction.

In section 2.1 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 $\mathcal{A}$. Using this result, and a simple translation of Rabin automata to NB, we can improve the complexity of complementation of NB; for an NB of size n, we obtain a complementary NB automaton of size $2^{O(n \log n)}$. We also discuss some lower bounds for these problems.

We then proceed and consider complementation of the other types of automata. In section 2.2, we show that for NE automata the blow-up of complementation is inherently doubly exponential. The more surprising result, which is given in section 2.3, is that for NS automata there exists an exponential complementation. For an NS automaton with n states and h accepting pairs, we construct a complementary NB automaton of size $2^{n^2 h^3}$.

The complexity of the complementation construction has a close connection with the complexity of the universality problems for the different types of automata.

The results reported in this chapter appear in [SV89, Saf88].

2.1 Complementation and Co-determinization (NB, DR)

Both these results are simple corollaries of the determinization construction NB $\rightarrow$ DS (Corollary 1.3) shown above. Note that the conversion of DS to DR of Lemma 1.2 is actually a complementation of DR (or DS) automata.

Corollary 2.1 $\overline{\mathrm{NB}}(n) \to \mathrm{DR}(2^{\mathrm{O}(n\log n)}, n+1)$; *for any nondeterministic Büchi automaton with n states, there exists a complementary deterministic Rabin automaton, with $2^{\mathrm{O}(n\log n)}$ states and $n+1$ accepting pairs.*

Proof: Use the conversion of Corollary 1.3 to get a DS whose number of states is $2^{\mathrm{O}(n\log n)}$ and $n+1$ accepting pairs. Interpret it as DR to get the complement. □

A similar result, based on our determinization construction (Theorem 1.1), and leading to a DR of size $(2^{\mathrm{O}(n^2)}, \mathrm{O}(n^2))$, has been independently derived and is reported in [EJ89].

We use this Corollary, and a simple conversion from Rabin automata to NB automata, to obtain an efficient complementation construction for NB.

Corollary 2.2 $\overline{\mathrm{NB}}(n) \to \mathrm{NB}(2^{\mathrm{O}(n\log n)})$; *for any* NB *of size n, there exists a complementary* NB *of size $2^{\mathrm{O}(n\log n)}$.*

Proof: Use the co-determinization construction of Corollary 2.1 to get a DR of size $(2^{\mathrm{O}(n\log n)}, n+1)$, and then use Lemma 0.1 to get a complementary NB of size $2^{\mathrm{O}(n\log n)}$. □

Turning to match this upper bound with a lower bound, it is clear that complementation of NB is at least exponential. It is possible to extend the language, which is used to establish 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., $2^{\mathrm{O}(n\log n)}$. This establishes the optimality of our construction.

One might think that this exponential lower bound is due to the fact that the original automaton is nondeterministic. We now show that even if the automaton we want to complement is a DR automaton, the complementary NB automaton is, in general, of exponential size. This is the same as showing this exponential lower bound for the translation DS→NB. Note that this entails an exponential lower bound for the translations DE→NB, DS→NM, and NS→NB.

Lemma 2.3 *For every $n > 0$ there exists a* DS *automaton with $3n$ states and $2n$ accepting pairs, such that any equivalent* NB *automaton has at least 2^n states.*

Proof: Let $\Sigma = \left\{0, \hat{1}, \check{1}\right\}$, we look at a word $\sigma \in \Sigma^\omega$ as an infinite sequence of vectors each of size n. Our language X_n is the set of all words in which $\hat{1}$ appears infinitely many times in the ith place in a vector iff $\check{1}$ does so too; denote by

$$\hat{\inf}(\sigma) = \left\{i \in [1..n] \colon \left|\left\{j \colon j \bmod n = i, \sigma_j = \hat{1}\right\}\right| = \infty\right\}$$

and by

$$\check{\inf}(\sigma) = \left\{i \in [1..n] \colon \left|\left\{j \colon j \bmod n = i, \sigma_j = \check{1}\right\}\right| = \infty\right\}$$

we define

$$X_n = \left\{ \sigma \in \Sigma^\omega \mid \hat{\inf}(\sigma) = \check{\inf}(\sigma) \right\}.$$

X_n can be accepted by a DS automaton, $\mathcal{D}_n$, that consists of n triples of states, $Q = [1..n] \times \left\{0, \hat{1}, \check{1}\right\}$. When $\mathcal{D}_n$ reads a letter, it goes to the corresponding state in the first triple. Reading the next letter it goes to the corresponding state in the next triple, and so on. When it reaches the last triple it goes back to the first one. The acceptance condition requires that for each of the triples of states either the states that correspond to $\hat{1}$ and $\check{1}$ are both included in the infinity set or none of them does.

Assume that there is an NB automaton $\mathcal{A}$, with less than 2^n states, that accepts X_n. For each $R \subseteq [1..n]$, let $\hat{v}_R \in (0 + \hat{1})^n$ be the vector that is $\hat{1}$ exactly in all the indexes contained in R, and 0 elsewhere. Let $\check{v}_R$ be the corresponding $\check{1}$ vector. The word $\sigma_R = ((\hat{v}_R)^{2^n} \check{v}_R)^\omega$ clearly belongs to X_n. Consider some accepting $\mathcal{A}$-run over σ_R, ξ_R, and look at a sequence of states that corresponds to one cycle of σ_R at the periodic suffix of ξ_R. Consider the states that occur in ξ_R just before the $\hat{v}_R$ vectors in this sequence. Since $\mathcal{A}$ has less than 2^n states, at least one of the states occur more than once, denote it by q_R. We claim that for $R \neq R'$, $q_R \neq q_{R'}$. Indeed, assume that $q_R = q_{R'}$, we could add to σ_R, whenever ξ_R reaches $q_R = q_{R'}$, a sequence of $\hat{v}_{R'}$ vectors that leads from $q_{R'} = q_R$ to itself, thus obtaining an accepting run. But clearly the resulting word is not in X_n, since there are $\hat{1}$s infinitely many times for indexes both in R and in R', while there are $\check{1}$s only at the indexes of R. □

2.2 NE Automata

Lemma 2.4 *Complementation of* NE *automata is at least doubly exponential; for every $n > 0$ there exists a language L_n that can be accepted by an* NE *of size* $O(n)$, *but its complement $\overline{L_n}$ cannot be accepted by any ω-automaton with less than 2^{2^n} states.*

Proof:

Let $\Sigma = \left\{0, 1, \sharp, \mathbf{0}, \mathbf{1}, \boldsymbol{\sharp}\right\}$. For a word $\sigma \in \Sigma^\omega$, the *projection of σ on Σ'*, for $\Sigma' \subseteq \Sigma$, denoted $\sigma|_{\Sigma'}$, is the sequence obtained from σ by ignoring the letters not in Σ'.

An *n-counter* is a string of the form $\sharp(0+1)^n$, which can be interpreted as the binary representation of an integer in the range $0..2^n - 1$. For a counter c, we denote by $int(c)$ the integer it represents. An *n-counter word* is a finite or infinite concatenation of n-counters of the form $c_0 c_1 ...$, where $int(c_0) = 0$, and for every $i \geq 0$, $int(c_{i+1}) = int(c_i) + 1$ (mod 2^n).

Let $m = 2^n$. An *interleaved m-counter* is a string of the form $\boldsymbol{\sharp} c_0 i_0 ... c_{m-1} i_{m-1}$, where each $i_j \in \left\{\mathbf{0}, \mathbf{1}\right\}$ and the sequence $c_0 ... c_{m-1}$ is an n-counter word. We can interpret an interleaved m-counter C as the binary representation of an integer $i_0 ... i_{m-1}$, in the range $0..2^m - 1$, such that each bit i_j is preceded by an n-counter identifying the index j of the bit. We denote by $int(C)$ the integer represented by the m-counter C.

An *interleaved m-counter* word is a finite or infinite concatenation of interleaved m-counters of the form $C_0C_1...$, where $int(C_0) = 0$, and for every $i \geq 0$, $int(C_{i+1}) = int(C_i) + 1 \pmod{2^m}$.

The language X_n $(= \overline{L_n})$ consists of all words in $\Sigma^*\sigma$, where σ is the infinite interleaved word.

First, we show that any automaton that accepts X_n has at least 2^m states. Consider the special X_n word K^ω where $K = C_0C_1...C_{2^m-1}$ is an m-counter word consisting of 2^m consecutive m-counters. Consider any automaton $\mathcal{A}$ that accepts X_n, and an accepting $\mathcal{A}$-run ξ over K^ω. Let us consider the different sequences of states $q_1, ..., q_{|K|}$ which appear successively in ξ, starting at positions which are multiples of $|K|$. Each one of these sequences can be viewed as a finite $\mathcal{A}$-run over the finite word K. Obviously, there must be some particular sequence, $\overline{q} = q_1, ..., q_{|K|}$, and infinitely many positions $i_1 < i_2...$ all of which are multiples of $|K|$ at which $\overline{q}$ appears.

We claim that all of the states appearing in $\overline{q}$ must be distinct. Otherwise, $\overline{q}$ can be expressed as $q_1, ..., q_i, ..., q_j, ..., q_{|K|}$ with $q_i = q_j$. In that case we could perform surgery on $K = a_1, ..., a_i, ..., a_j, ..., a_{|K|}$, and replace each occurrence of K at i_{2k}, $k = 1, 2, ...$, by the shorter word $a_1, ..., a_i, a_{j+1}, ..., a_{|K|}$, maintaining the integrity of the run segment $q_1, ..., q_i, q_{j+1}, ..., q_{|K|}$ over this altered segment of the input. Let us denote by $\hat{w}$ and $\hat{\xi}$ the altered input and run, respectively. Since we have retained K and $\overline{q}$ at all the positions i_{2k+1}, $k = 1, 2, ...$, it is obvious that the infinity set of ξ and $\hat{\xi}$ are identical, and therefore $\hat{w}$ is accepted. However, it is also clear that $\hat{w} \notin X_n$. Thus, any automaton accepting X_n must have at least 2^m distinct states.

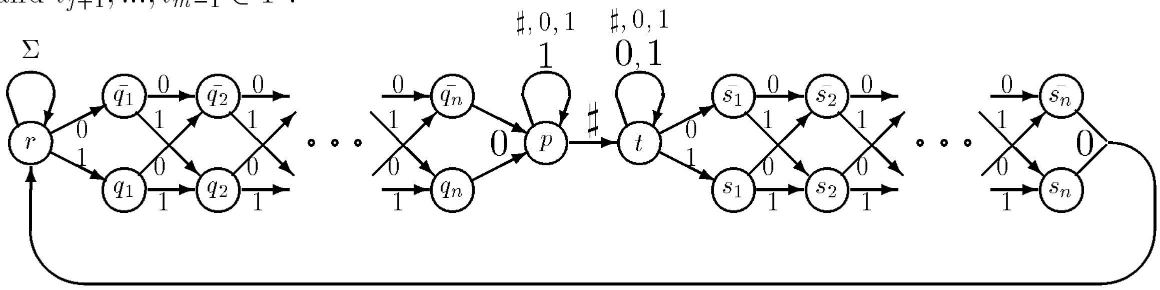
It only remains to show that $\overline{X_n}$ can be accepted by an NE automaton of size $O(n)$. A necessary and sufficient condition for a word to belong to X_n is that it has the form xy where $x \in \Sigma^*$ and y satisfies the following four requirements:

1. $y \in ((\sharp + \lambda)\sharp(0 + 1)^n(0 + 1))^\omega$.
2. Let $y|_{\{0,1,\sharp\}} = c_0c_1...$, then it is required that for every $j \geq 0$, $int(c_{j+1}) = int(c_j) + 1 \pmod{2^n}$.
3. $y \in (\sharp\sharp 0^n(\Sigma - \sharp)^*)^\omega$, $y \notin \Sigma^*(0 + 1)\sharp 0^n\Sigma^\omega$.
4. Let $y = C_0C_1...$, then it is required that for every $i \geq 0$, $int(C_{j+1}) = int(C_j) + 1 \pmod{2^m}$.

Consequently, an automaton recognizing $\overline{X_n}$ can start by nondeterministically branching into one of four sub-automata, each of which checks the violation of one of the four conditions. It is not difficult to see that the violation of conditions 1–3 can be checked by automata of size linear in n, since they only require the ability to count up to n.

Condition 4 requires that, from a certain point on, each m-counter is greater by 1 from its predecessor. The automaton checking for its violation has to detect infinitely many cases that an m-counter is different from its predecessor plus 1. To detect such a case, it is sufficient to compare two bits i_j, i'_j in corresponding positions in infinitely many contiguous m-counters, C, C', and verify that they are inconsistent with the requirement that $int(C') = int(C)+1 \pmod{2^m}$. Such inconsistency occurs in two cases, depending on the string of bits, $i_{j+1}, ..., i_{m-1}$, appearing to the right of i_j in C. If all of them are 1, then i_j and i'_j are inconsistent with incrementation if $i_j = i'_j$. If at least one of the bits, $i_{j+1}, ..., i_{m-1}$, is 0, then inconsistency is expressed by $i_j \neq i'_j$.

It is not difficult to see how to efficiently construct parts of an automaton that will check the inconsistency of the bits and whether the string $i_{j+1}, ..., i_{m-1}$ contains a 0. The part that requires more attention is how to confirm that i_j and i'_j are in *corresponding* positions. We illustrate how this is done, strongly utilizing the general acceptance condition we are allowed in NE automata, for the special case that $i_j = i'_j = 0$ and $i_{j+1}, ..., i_{m-1} \in 1^*$.



The acceptance condition associated with this sub-automaton is presented by the following formula, where we use the name of a state as a predicate claiming that this state belongs to the infinity set:

$$p \wedge \bigwedge_i (q_i \leftrightarrow \neg \bar{q}_i) \wedge \bigwedge_i (s_i \leftrightarrow \neg \bar{s}_i) \wedge \bigwedge_i (q_i \leftrightarrow s_i).$$

This formula consists of four clauses which can be interpreted as follows. The first clause p claims that we discovered infinitely many discrepancies. The second (third) clause claims, for every i, that we have visited infinitely many times at most one of q_i and $\bar{q}_i$ (s_i and $\bar{s}_i$). The fourth clause claims that q_i is visited infinitely many times iff s_i is visited infinitely many times.

It is obvious that an infinity set I that satisfies these conditions uniquely identifies an n-counter c whose ith bit in its binary representation is 1 iff q_i and s_i belong to I (0 iff $\bar{q}_i, \bar{s}_i \in I$). It is clear that a run producing such an infinity set must have infinitely many discrepancies in the c'th bit of its m-counter. □

Corollary 2.5 *Determinization of* NE *automata (to any class of deterministic automata) is doubly exponential.*

Lemma 2.6 *The Universality problem for* NE *automata is* EXPSPACE*-complete.*

Proof: We use the idea of the Lemma 2.4, combined with the general technique to prove such lower bounds of [MS72, Sto74]. □

2.3 NS Automata

Throughout this section, we are given an NS, $\mathcal{A} = \left\langle \Sigma, Q, q_0, \delta, \bigwedge_{1 \leq i \leq h}(L_i \to U_i) \right\rangle$, for which, in Theorem 2.10, we give a complementary NB, $\mathcal{B}$, of exponential size.

A *history graph*, $G = \langle V, E \rangle$, for $\mathcal{A}$, is a labeled directed finite graph over a subset of the states, $V \subseteq Q$, where each edge is labeled by a collection of sets of states, $E\colon V \times V \to 2^{2^Q}$. A history graph is intended to describe all the possible $\mathcal{A}$-runs over a word. We say that G *corresponds* to a finite word $\sigma \in \Sigma^*$ if

1. $\delta(V, \sigma) = V$, and
2. A history set $H \subseteq Q$ belongs to $E(q_1, q_2)$ iff there exists an $\mathcal{A}$-run from q_1 to q_2 over σ that visits exactly the set H.

We say that G is *idempotent* if for every (not necessarily distinct) states $q_1, q_2, q_3 \in V$, if $H_1 \in E(q_1, q_2)$ and $H_2 \in E(q_2, q_3)$ then $H_1 \cup H_2 \in E(q_1, q_3)$.

We now extend the definition of correspondence to infinite words: a history graph, $G = \langle V, E \rangle$, *corresponds* to an infinite word $\sigma \in \Sigma^\omega$ if there is an increasing sequence $i_0 < i_1 < \ldots$ such that $\delta(q_0, \sigma[0, i_0)) = V$, and for every $j \geq 0$, G corresponds to $\sigma[i_j, i_{j+1})$.

Lemma 2.7 *For any infinite word $\sigma \in \Sigma^\omega$ there exists an idempotent history graph G corresponding to σ.*

Proof: Given a word $\sigma \in \Sigma^\omega$, we assign to each pair of integers (i, j), $i < j$, a triple $\langle V_i, E_{ij}, V_j \rangle$ such that $V_i = \delta(q_0, \sigma[0, i))$, $V_j = \delta(q_0, \sigma[0, j))$ and $E_{ij}\colon V_i \times V_j \to 2^{2^Q}$ represents all the histories over $\sigma[i, j)$ starting at V_i. By the infinite Ramsey Theorem (Lemma 0.8), there exists a history graph, $G = \langle V, E \rangle$, and an *infinite* set $S \subseteq \mathbb{N}$ such that for every $i \in S$, $\delta(q_0, \sigma[0, i)) = V$, and for every $i, j \in S$, $i < j$, G corresponds to $\sigma[i, j)$. Thus, G corresponds to σ, and is idempotent. □

A history graph, $G = \langle V, E \rangle$, is *non-accepting* if for each state $q \in V$ and a history set $H \in E(q, q)$, H is non-accepting, i.e., there exists an $i = 1, \ldots, h$, such that $H \cap L_i \neq \phi$ but $H \cap U_i = \phi$.

Lemma 2.8 *A word $\sigma \in \Sigma^\omega$ is not accepted by $\mathcal{A}$ iff there exists a non-accepting idempotent history graph G corresponding to σ.*

Note that there may be different history graphs that correspond to a word σ. Nevertheless, we prove that they are all either accepting or non-accepting.

Proof: Assume there exists a non-accepting idempotent history graph $G = \langle V, E \rangle$ that corresponds to σ via the sequence of breakpoints $i_0, i_1, \ldots$. Consider any $\mathcal{A}$-run ξ over σ, and let q be a state which is visited infinitely often by ξ at the breakpoints. Let H

be the infinity set of ξ. By the idempotence condition, $H \in E(q, q)$, and because G is non-accepting, H is non-accepting.

In the other direction, assume that there is no non-accepting idempotent history graph. Then any idempotent history graph $G = \langle V, E \rangle$, whose existence is ensured by Lemma 2.7, must be accepting. That is, there is some $q \in V$ such that $q \in \delta(q_0, \sigma[0, i_0))$, and for every $j \geq 0$, $q \in \delta(q, \sigma[i_j, i_{j+1}))$, while visiting exactly some set $H \in E(q, q)$ which is accepting. It follows that the run proceeding from q_0 to q at i_0 and then repeatedly following the path from q to q while visiting H, is accepting. $\square$

What we would like to do, is to use history graphs for complementation. The complementary automaton can guess a non-accepting idempotent history graph and check that it corresponds to the input word. Unfortunately, the number of history graphs is doubly exponential, thus this complementation construction would yield a doubly exponential blow-up. The key point here, is that for Streett automata we can represent the non-accepting history graphs in a more concise way, such that the number of such graphs is only exponential, and we can still check that they correspond to the input word.

Let us consider the structure of the set $F_0 = E(q, q)$, for some $q \in V$, i.e., a state that is visited infinitely often by some runs over σ. From the requirements of idempotence, it is clear that F_0 is closed under union. In particular, the union of all the sets in F_0 is a member of F_0, let us call it H_1. Since H_1 is non-accepting, there must exist some index i_1 such that $L_{i_1} \cap H_1 \neq \phi$, while $U_{i_1} \cap H_1 = \phi$. It follows that $H \cap U_{i_1} = \phi$ for every $H \in F_0$. Let us remove from F_0 all the sets that have nonempty intersection with L_{i_1}, e.g. H_1, and call the remaining family F_1. It is not difficult to see that F_1 is also closed under union, and H_2, the union of all members of F_1, also belongs to F_1. Consequently, we can find an index $i_2 \neq i_1$ such that $H_2 \cap L_{i_2} \neq \phi$, while $H_2 \cap U_{i_2} = \phi$.

Continuing in this manner, we obtain a sequence of distinct indexes, $\Delta = i_1, ..., i_k$, $k \leq h$, and a decreasing sequence of families,

$$E(q, q) = F_0 \supset F_1 \supset ... \supset F_k = \phi$$

such that $H \in F_{j-1} - F_j$ (i.e., H is removed in the jth step) iff

$H \cap L_{i_j} \neq \phi$, and

$H \cap U_{i_{j'}} = \phi$ for every $j' \leq j$.

We refer to the sequence Δ as the *removal order* corresponding to $E(q, q)$. For a given set H let us denote by

$l_\Delta(H)$ the minimal j such that $H \cap L_{i_j} \neq \phi$ (and $h + 1$ if there is no such j), and by

$u_\Delta(H)$ the minimal j such that $H \cap U_{i_j} \neq \phi$ (and $h + 1$ if there is no such j).

An equivalent formulation of the conditions is that there exists an ordering Δ such that for every $H \in E(q, q)$, $l_\Delta(H) < u_\Delta(H)$.

This observation suggests a more concise encoding for the set of histories labeling the history graph. We start by identifying, for each of the states $q_m \in V$, a removal ordering, Δ_m, corresponding to $E(q_m, q_m)$. For each edge and an ordering Δ_m, we can replace each history set, H, currently labeling this edge, by the pair $(l_{\Delta_m}(H), u_{\Delta_m}(H))$. Since every edge connecting q_i to q_j can, in principle, participate in several cycles from some state q to itself, the full label of this edge has the form $\Lambda(q_i, q_j) = \Lambda_1, ..., \Lambda_n$, where Λ_m contains all the pairs, $(l_{\Delta_m}(H), u_{\Delta_m}(H))$, for every $H \in E(q_i, q_j)$. The obvious gain in complexity is from 2^{2^n} different E-labels to 2^{nh^2} different Λ-labels.

An *encoded history graph*, $G = \langle V, \Lambda_1, ..., \Lambda_n, \Delta_1, ..., \Delta_n \rangle$, consists of a finite graph where each label $\Lambda_m(q_i, q_j)$ is a set of pairs of indexes, and each node $q_m \in V$ is associated with a removal order Δ_m. We can translate the notions of idempotence and non-acceptance, previously defined for the full history graphs, into the corresponding notions for the encoded graphs.

We call an encoded history graph G *idempotent* if, for every m and every (not necessarily distinct) states $q_1, q_2, q_3 \in V$, if $(l_1, u_1) \in \Lambda_m(q_1, q_2)$ and $(l_2, u_2) \in \Lambda_m(q_2, q_3)$, then $(min(l_1, l_2), min(u_1, u_2)) \in \Lambda_m(q_1, q_3)$. G is *non-accepting* if for every state $q_m \in V$ and every $(l, u) \in \Lambda_m(q_m, q_m)$, $l < u$. G *corresponds* to a finite word $\sigma \in \Sigma^*$ if $\delta(V, \sigma) = V$ and a pair (l, u) belongs to $\Lambda_m(q_i, q_j)$ iff there exists an $\mathcal{A}$-run over σ from q_i to q_j with a history set H such that $(l_{\Delta_m}(H), u_{\Delta_m}(H)) = (l, u)$. G *corresponds* to an infinite word $\sigma \in \Sigma^\omega$ if there exists an increasing sequence $i_0 < i_1 < ...$, such that $\delta(q_0, \sigma[0, i_0)) = V$, and for every j, G corresponds to $\sigma[i_j, i_{j+1})$.

Given a non-accepting idempotent history graph $G = \langle V, E \rangle$, we can construct a non-accepting encoded history graph corresponding to G, denoted by $G' = en(G)$, by the procedure outlined above. That is, identifying the removal orders Δ_m for each of the states $q_m \in V$, and encoding the labels of the edges by putting $(l_{\Delta_m}(H), u_{\Delta_m}(H))$ into $\Lambda_m(q_i, q_j)$ for each $m = 1, ..., n$, $H \in E(q_i, q_j)$. It is not difficult to see that if G corresponds to an infinite word σ via the breakpoints sequence $i_0, i_1, ...$, then $en(G)$ corresponds to σ with the same sequence. The converse is not necessarily true, i.e., we may have a non-accepting idempotent encoded history graph G' corresponding to the infinite word σ via the sequence $i_0, i_1, ...$, while there is no history graph G that corresponds to σ with the same sequence.

We will now prove the analog to Lemma 2.8 stating that the existence of a non-accepting idempotent encoded history graph is a necessary and sufficient condition for the rejection of a word σ by $\mathcal{A}$.

Lemma 2.9 *A word $\sigma \in \Sigma^\omega$ is not accepted by $\mathcal{A}$ iff there exists a non-accepting idempotent encoded history graph G corresponding to σ.*

Proof: Assume that the word σ is rejected by the automaton $\mathcal{A}$, by Lemma 2.8 there exists a non-accepting idempotent history graph G corresponding to σ. By the observation above, $en(G)$ is a non-accepting idempotent encoded history graph corresponding to σ.

In the other direction, assume the existence of a non-accepting idempotent encoded history graph G, which corresponds to σ via the sequence $i_0, i_1, \ldots$. Consider any run ξ over σ, and let q_m be a state which is visited infinitely often at the breakpoints $i_0, i_1, \ldots$. Let H be the infinity set of ξ. By the idempotence condition $(l_{\Delta_m}(H), u_{\Delta_m}(H)) \in \Lambda_m(q_m, q_m)$, and from the non-acceptance condition we have that $l_{\Delta_m}(H) < u_{\Delta_m}(H)$, implying $H \cap L_{l_{\Delta_m}} \neq \phi$, while $H \cap U_{u_{\Delta_m}} = \phi$. □

Theorem 2.10 $\overline{\mathrm{NS}}(n,h) \rightarrow \mathrm{NB}(2^{n^3 h^2})$*; for any* NS *automaton,* $\mathcal{A}$*, with* n *states and* h *accepting pairs, there exists a complementary* NB *automaton,* $\mathcal{B}$*, of size* $2^{n^3 h^2}$*.*

Proof: The automaton $\mathcal{B}$ first guesses nondeterministically a non-accepting idempotent encoded history graph $G = \langle V, \Lambda_1, \ldots, \Lambda_n, \Delta_1, \ldots, \Delta_n \rangle$, then, while reading σ, $\mathcal{B}$ guesses nondeterministically the breakpoints and checks that:

1. V is the set of reachable states for all the breakpoints,
2. for every $j \geq 0$, $\sigma[i_j, i_{j+1})$ corresponds to G.

A straightforward subset construction is sufficient to check the first condition. To check the second condition, $\mathcal{B}$ maintains, for each $q_i \in V$ and a removal order Δ_j, a set $S(q_i, \Delta_j)$ consisting of triples (q, l, u) such that there is an $\mathcal{A}$-run starting at q_i at the most recent breakpoint, and reaching q with history H such that $(l_{\Delta_j}(H), u_{\Delta_j}(H)) = (l, u)$. When we reach the next breakpoint, we verify that for every $q_i, q_j \in V$ and Δ_m

$$(q_j, l, u) \in S(q_i, \Delta_m) \text{ iff } (l, u) \in \Lambda_m(q_i, q_j).$$

□

Corollary 2.11 *The universality problem for* NS *automata is* PSPACE*-complete.*

Proof: We only have to observe that the above complementation construction can be carried out *on the fly* (i.e., without actually constructing the complementary automaton), and by Lemma 0.6 we know that emptiness of NB automata is in nondeterministic LOGSPACE. The result of PSPACE hardness follows from the same lower bound for automata on finite words ([MS72]). □

Note that an exponential determinization of NS automata into either DR or DS automata would entail an exponential complementation.

2.4 DM Automata

Lemma 2.12 *Complementation of* DM *automata is at least exponential; for every* n *there exists a language* L_n *that can be accepted by a* DM *automaton of linear size, and the complementary language* $\overline{L_n}$ *cannot be accepted by a* DM *automaton with less than* $2^n - 1$ *accepting sets.*

Proof: Let $\Sigma_n = \left\{1, ..., n, \dot{1}, ..., \dot{n}\right\}$, and

$$X_n = \left\{\sigma \in \Sigma^{\omega} \mid \{1, ..., n\} \subseteq \inf(\sigma)\right\}$$

be the languages of all words in which all the letters $1, ..., n$ appear infinitely many times. Let $Y_n = \bigcup_{0<m\leq n} \Sigma^* m\dot{m}\Sigma^{\omega}$ be the language that permits any word in which an $\dot{m}$ appears immediately after m. Let $L_n = X_n \cup Y_n$, be like X_n except that a $\dot{m}$ after m permits any suffix. In a similar way to the proof of Lemma 1.5 we can prove that the states of any DM automaton that accept $\overline{L_n}$ can be partitioned into equivalence classes according to the last letter of the prefix leading to them, and that the number of different accepting sets is exponential. □

An interesting question is concerned with the complexity of the *equivalence* (or containment) problem for DM automata. The usual procedure to check whether the language accepted by an automaton, $\mathcal{A}_1$, is contained in the language accepted by another automaton, $\mathcal{A}_2$, is first to complement $\mathcal{A}_2$, and take the union $\mathcal{A}_1 \cap \overline{\mathcal{A}_2}$ and check if it is empty. Since (by Lemma 2.12) complementation of DM automata is exponential, a naive application of this procedure would yield a PSPACE complexity. However, it is shown in [CGK89, CDK89] that the complementation of DM automata to NB automata can be carried out with only a polynomial blow-up, and thus the equivalence of DM automata is in nondeterministic LOGSPACE.

Using techniques similar to those of Lemma 2.12 we can prove the following lemma:

Lemma 2.13 *The translation of* DR *automata (even* DB*) into* NM *automata is at least exponential.* □

Chapter 3

Applications

In this chapter we present some applications of our results. We give a short survey on logics of programs and their decision problems.

3.1 Logics of Programs

The field of logics of programs was motivated mainly by the need to provide precise mathematical tools for the specification and verification of programs. Starting with Floyd ([Flo67]), Engeler ([Eng67]), and Hoare ([Hoa67]), many formalisms were suggested for this purpose. Some of these formalisms addressed programs that are terminating, and studied their input–output relations. Other approaches considered *reactive systems* ([HP85]) whose role is to maintain an ongoing interaction with their environment. The verification problem is the following: given a program P and a specification φ, to prove that all computations of P satisfy φ. An even more ambitious problem is that of synthesis, where, given a specification φ, we are asked to construct a program P, all of whose computations satisfy φ. These problems are in general unsolvable. Therefore, the general approach to verification is based on a proof system that may be used to establish the correctness of a program. Such proof systems are shown to be complete according to a very liberal notion of completeness, but strongly rely on the ingenuity of the prover. However, for the restricted case of finite-state programs, these questions may be solved algorithmically. Therefore, for such simplified programs we can consider the problem of *automatic verification* which is to check that a propositional specification is satisfied by all computations of a finite state program.

In this section we give a short survey on some of the logics suggested for these purposes, and the relation of their decision procedures to automata on infinite objects. For more thorough surveys of these topics and related ones, the reader is referred to [Har84, KT89, Eme89, MP01].

Dynamic Logic (DL) was introduced by Pratt in [Pra76] (see also [Har79]), and may be viewed as a combination of Hoare logic ([Hoa67]) and modal logic. DL is a two sorted modal logic where one sort denotes programs, which are viewed as state transformers, and the other sort represents predicates, which describe the states before and after the execution of a program. Programs are represented as regular expressions over some atomic actions, standing for basic statements, such as assignments or tests. The syntax of predicates includes a first order language and also the special modal formula $\langle\alpha\rangle p$, which is true, in a state s, if it is possible to execute the program α from s and reach a state s' in which p holds.

Fischer and Ladner ([FL79]) considered the restricted case of *Propositional Dynamic Logic* (PDL), in which the basic predicates are propositions, and the atomic programs are left uninterpreted. This version of the logic provides a more abstract description of programs and their properties.

A structure over which PDL formulas can be interpreted, consists of a set of states, S, each state giving an interpretation to the basic propositions, and an interpretation assigning to each atomic action a a relation $\rho_a \subseteq S \times S$. The intended meaning is that $\rho_a(s, s')$ holds if the execution of a from s can result in s'. It is convenient to view such a structure as a directed graph whose nodes are labeled by the states, and which has an edge, connecting s to s' and labeled by a, iff $(s, s') \in \rho_a$. Any such graph can be unwound into a (possibly) infinite tree.

In contrast with DL, whose validity problem is highly undecidable ([HMP77]), the validity problem for PDL was shown in [FL79] to be decidable, actually with an upper bound of nondeterministic exponential time. A key concept introduced in their decision procedure is that of a closure of a formula, since then called the Fischer-Ladner closure. The closure of a formula φ is a set of formulas, which includes φ itself, all of its subformulas, and some expansions of these subformulas. The size of the closure of a formula φ is linear in $|\varphi|$. The decision procedure in [FL79] is based on the fact that if a formula φ is satisfiable, it must be satisfiable in a model such that any two distinct states are distinguishable by a formula in the closure of φ. This shows that there must be a satisfying model of at most exponential size. They also established a deterministic exponential time lower bound for this problem.

A better upper bound, matching the lower bound, was given by Pratt in [Pra79, Pra80]. Pratt's decision procedure first constructs a structure containing states with all possible truth assignments to formulas in the closure (the *tableau*), and then successively deletes states that are found to violate some of their commitments. The structure remaining after deletion, if not empty, is a model.

Streett considered in [Str80, Str82] some extensions of PDL first suggested (with different notations) in [HP78], that can express the existence of an infinite computation, i.e., particular infinite paths in the model. He showed that the extended logic is still elementary decidable, by reducing its satisfiability problem to the non-emptiness problem for tree automata. (The fact that these logics are decidable, without the elementary recursive upper bound, can be derived by a reduction into S2S ([Str80]).)

Pratt ([Pra82]) and Kozen ([Koz83]) have defined two restricted versions of the *propositional μ-calculus* (see [Par69, Par76, dBdR73, HP73] for the definition and applications of the full μ-calculus). Both versions are more expressive than PDL, and possess exponential time decision procedures. However, none of these versions covered the full propositional μ-calculus.

The propositional μ-calculus shares with PDL the basic modalities, $\langle a \rangle p$ and $[a]p$, for any atomic program a. However, instead of allowing more complex program expression, it introduces recursion on the predicates. Thus, the formula $\langle a^* \rangle p$ of PDL is expressed in the propositional μ-calculus as $\mu X.(p \vee \langle a \rangle X)$, standing for the *minimal* solution to the equation $X = p \vee \langle a \rangle X$. The semantic models for the propositional μ-calculus are identical to those of PDL, i.e., structures of states connected by atomic programs.

Independently of this line of research, the application of another modal logic, *temporal logic* (TL), to the specification and verification of concurrent programs, was suggested in [Pnu77]. TL can be viewed as a simplified version of DL, in which we give up the identification of the actions responsible for changes in states, and are interested only in the fact that one state is an immediate successor of another. Going back to our structures, which serve as the models for these logics, the simplification amounts to deleting the labels from the edges.

Two general versions of TL have been discussed. The linear time version, as defined in [GPSS80], requires that each state has precisely one successor. As a result, the natural models for this logic are infinite sequences of states. A second version of TL is the one obtained by allowing several successors to each state. The natural models for this version are, therefore, trees. This requires introducing two versions of each of the temporal operators, one holding for all paths originating at a state, and the other claiming the existence of a path on which an associated property holds. Branching time TL was discussed in [BMP81], which provided an exponential time decision procedure for the propositional version of the logic. Note that models for the branching time logic are again structures of states which can be unwound into a (possibly) infinite tree.

Wolper ([Wol83]) observed that the linear version of PTL (propositional temporal logic), when considered as a formalism for defining ω-languages, cannot express the full range of the ω-regular languages. In fact, PTL can express exactly the *non-counting* ω-regular languages ([Tho81], see [MP71] for a full exposition of non-counting languages). Consequently, Wolper defined an extended PTL (ETL) where ω-regular languages are considered as operators that can be represented by ω-automata ([WVS83]). He provided an exponential time decision procedure for ETL. This obviously entails the same upper bound for PTL. Sistla, Vardi and Wolper ([SVW87]) studied $\mathrm{ETL_B}$, where temporal connectives are represented by nondeterministic Büchi automata, and showed that the decision problem for this logic is PSPACE-complete, just like the decision problem for PTL ([HR83, SC85]). In [SV89] $\mathrm{ETL_S}$, i.e., the version of ETL in which operators are represented by Streett automata, is investigated, and it is shown, using the exponential complementation of Theorem 2.10, that the decision procedure for this logic is also PSPACE-complete.

[CE81] considered the synthesis problem based on a branching time temporal logic CTL (which is very similar to the logic suggested in [BMP81]). They showed that the exponential decision procedure for satisfiability can be used to synthesize a program that satisfies the specification. A similar synthesis approach was demonstrated in [MW84], in which they extract a program from the tableau constructed in the decision procedure for linear time TL.

It turns out that for expressing program properties, CTL and TL are incomparable ([Lam80]), i.e., each one can express properties that are not expressible in the other. It is quite natural that branching time logics can express properties which are not expressible in linear time, such as the *possibility* of an occurrence of an event. The more surprising fact is that TL can express properties not expressible in CTL, such as the property expressed by the linear temporal formula $\Box p \vee \Box q$. This deficiency was later corrected by defining a stronger version of branching time temporal logic called CTL* ([CES86, EH82, ES84]).

The problem of automatic verification of finite-state programs, relative to a CTL specification, has been considered in [CES86], where it is shown that this problem is solvable in time polynomial in the size of the formula and of the program. In [SC85] the same problem has been studied for linear time specifications, and it has been established that both the automatic verification problem and the satisfiability problem are PSPACE-complete for this version. This has been often quoted as an argument for the advantages of branching time logic. A more discriminating analysis of automatic verification for linear specifications, carried out in [LP84], gave a running time upper bound which is polynomial in the size of the program and exponential in the size of the formula to be verified.

We suggest the following automata theoretic perspective on the decision procedures for the logics discussed above.

Pratt's decision procedure for satisfiability of PDL ([Pra79, Pra80]) can be described as follows: Given a formula φ, we construct the tableau of φ whose states are all the possible (consistent) truth assignments to formulas in the Fischer-Ladner closure of φ. There is a edge from s to s', labeled by a, iff for every $[a]p$ true in s, p is true in s'. The tableau can be viewed as a finite state machine over trees, where the nodes of the input tree are labeled by truth assignments to the atomic propositions of φ, and the different successors of a node correspond to the application of different atomic programs. The tableau, viewed in this way, checks that a given tree satisfies the *local* requirements of φ. In order to check the *global* requirements, i.e., that for every $\langle\alpha\rangle p$ that appears in a node, there is an execution of α that leads to a state in which p is true, we add to the tableau some additional constructs and a Büchi acceptance condition, resulting in a tree automaton. A possible augmentation consists of a queue of diamond formulas to be satisfied. In each step, the automaton nondeterministically decides which formulas are going to be satisfied on each of the descendants of the node, and erases the formulas not to be satisfied on that node. New diamond formulas join the queue on the right, and every formula that is satisfied, is removed from the queue. Whenever the leftmost formula is erased (or is empty), the automaton enters an accepting state. The resulting tree automaton $\mathcal{T}_\varphi$ is non-empty iff φ is satisfiable. Since emptiness of Büchi tree automata is in deterministic polynomial time ([Rab70, VW86]), we obtain a deterministic exponential time decision procedure for satisfiability of PDL. A similar procedure can be applied to other simple branching time logics, such as CTL and weaker versions of CTL*.

In the corresponding linear time logics, such as PTL, a similar procedure results in a Büchi ω-automaton. Since emptiness of Büchi ω-automata is in nondeterministic LOGSPACE, and the whole process can be carried out on the fly (without actually constructing the tableau), satisfiability of PTL is in PSPACE ([HR83, SC85]).

However, for more general logics, such as Δ-PDL ([Str80, Str82]), propositional μ-calculus in the branching case, and μ-TL ([Var88]) in the linear time case, just adding an acceptance condition to the tableau is not sufficient. Streett ([Str82], see also [VS85, ES84, SE84, VW86]) suggested that it is possible to express the hard parts of these logics, such as the negation of the Δ operator in Δ-PDL, by a linear size nondeterministic Büchi ω-automaton that has to *reject* any path of the tree. By Corollary 2.1, it is possible to obtain a deterministic Rabin ω-automaton which is the complement of this nondeterministic Büchi ω-automaton, and that has an exponential number of states and a linear number of accepting pairs. It is then possible to combine that deterministic ω-automaton with the simple tree automaton resulting from the tableau, into a single Rabin tree automaton with an exponential number of states and a linear number of accepting pairs (this requires the ω-automaton to be deterministic). 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]), 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.

It is worthwhile mentioning that ω-automata were proposed not only as a tool for decision procedures but also for specification of program properties ([AS85, AS87]).

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. An automaton is deterministic in the limit if any accepting run makes only finitely many nondeterministic choices. The partial determinization construction described in [Var85] is exponential, but the author has subsequently found that it contains an error. Our determinization construction (Theorem 1.1) obviously validates the upper bound (for partial determinization) claimed in [Var85]. Furthermore, a partial determinization (Corollary 3.1 presented below) 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]).

Corollary 3.1 *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. $\square$

Chapter 4

Discussion

All the classes of ω-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 4.1 we represent a map

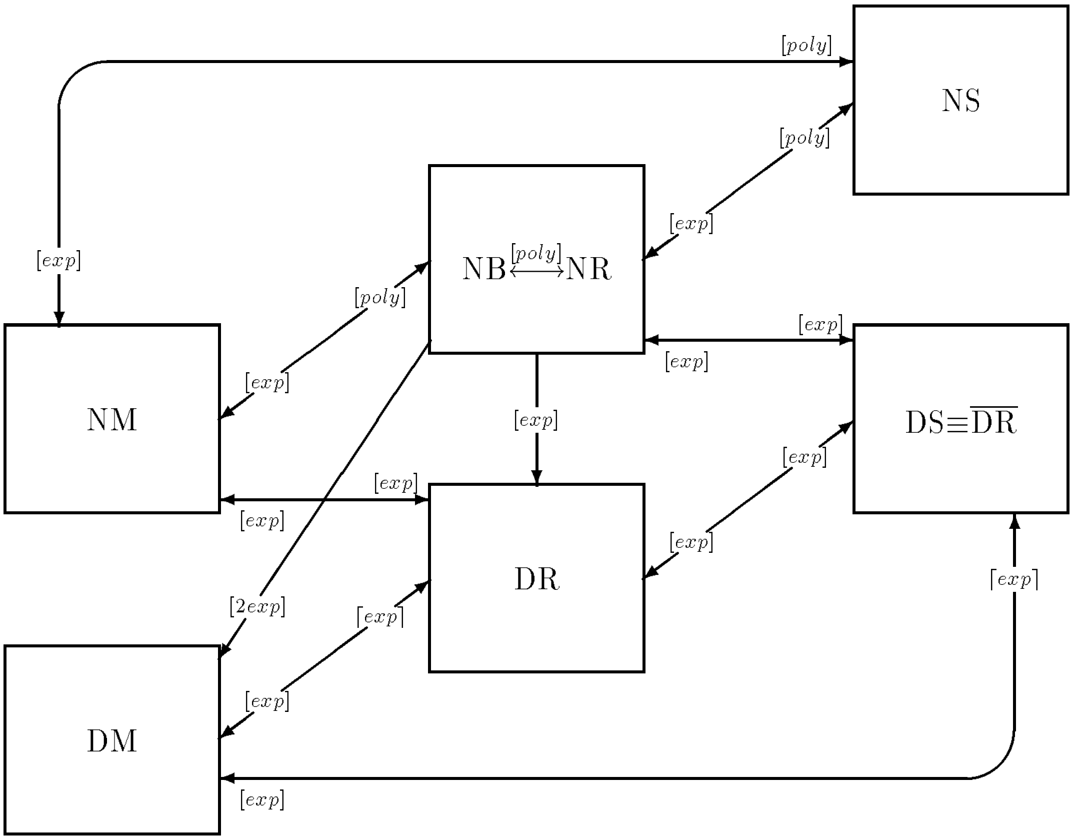


Figure 4.1: A map

of the translations between the different classes (The NE and DE classes are absent for lack of space, it is not difficult to complete the picture). 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.

The following table describes the complexity of translations between different classes of automata, with a similar notation to Figure 4.1, and with a reference to the main Lemma or Theorem that entailes the upper and lower bounds. n represents the number of states in the original automaton and h represents the size of acceptance condition, we may write two functions as the resulting complexity, the first represents the number of states and the second the size of acceptance condition. We write p to denote some polynomial function, e for some exponential function, and e^e to denote a doubly exponential complexity. An entry is empty if a translation without change of size follows from definitions. Question mark represents an open question.

	DM	DR	DS	DE	NM	NB	NS	NE
DM	–	$e]^{0.5}_{1.5}$	$e]^{0.5}_{1.5}$	$e]^{0.5}_{1.5}$	$e^e]^{1.1}_{1.5}$	$e^e]^{1.1}_{1.5}$	$e^e\rfloor_{1.5}$	$e^e\rfloor_{1.5}$
DR	$e\rceil^{1.1}$	–	$n \cdot e(h), h+1]^{1.2}_{2.3}$	?	$e]^{1.1}_{1.5}$	$e(n), n]^{1.1}_{1.5}$	$e^e\rceil^{1.1}$	$e^e]^{1.1}_{2.5}$
DS	$e\rceil^{1.3}$	$n \cdot e(h), h+1]^{1.2}_{2.3}$	–	?	$e\rceil^{1.3}$	$e(n), n+1]^{1.3}_{1.5}$	$e^e\rceil^{NB-DS:u}$	$e^e\rceil^{2.5}$
DE				–	$e]^{1.1}_{1.5}$	$e]^{1.1}_{1.5}$	$e^e\rceil^{1.1}$	$e^e]^{1.1}_{2.5}$
NM	–	$e]^{0.3}_{2.13}$	$e]^{0.3}_{2.3}$	$e]^{0.3}_{2.13}$	–	$e]^{0.3}_{2.13}$	$e]^{0.3}_{2.3}$	$e]^{0.3}_{2.13}$
NB	$p]^{0.2}$	$p]^{0.1}$	$e]^{0.4}_{2.3}$	$e]^{0.4}_{2.3}$	$p]^{0.2}$	–	$e]^{0.4}_{2.3}$	$e]^{0.4}_{2.3}$
NS	$p]^{0.2}$	$p]^{0.1}$	–	?	$p]^{0.2}$		–	$e]^{0.4}_{2.10,2.4}$
NE				–				–

Our determinization construction (Theorem 1.1 and Lemma 1.3) entails an exponential determinization constructions for NR and NM automata too (By Lemmas 0.1 and 0.2). Corollary 2.5 shows that determinization of NE automata is doubly exponential. Thus, concerning determiniztion of these classes, we are left with the following open problem:

Open Problem 4.1 *Is there an exponential determinization for* NS *automata?*

We do not have an upper bound better than doubly exponential, and the best lower bound we know is exponential. Note that an exponential determinization of NS automata into DR or DS automata would entail an exponential complementation for NS automata, thus removing the need for the construction of Theorem 2.10.

Another problem which is still open concerns complementation of tree automata:

Open Problem 4.2 *Is there an exponential complementation procedure for (Rabin) tree automata?*

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.

Bibliography

[AS85] B. Alpern and F. B. Schneider. Verifying temporal properties without using temporal logic. Technical Report 85-723, Cornell University, 1985.

[AS87] B. Alpern and F. B. Schneider. Proving boolean combinations of deterministic properties. In *Proc. 2nd IEEE Symp. on Logic in Computer Science*, pages 131–137, 1987.

[BMP81] M. Ben-Ari, Z. Manna, and A. Pnueli. The temporal logic of branching time. In *Proc. 8th ACM Symp. on Principles of Programming Languages*, pages 164–176, 1981.

[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.

[CDK89] E. M. Clarke, I. A. Draghicescu, and R. P. Kurshan. A unified approach for showing language containment and equivalence between various types of ω-automata. To appear, 1989.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proc. IBM Workshop on Logics of Programs*, pages 52–71. Lecture Notes in Computer Science 131, Springer, 1981.

[CES86] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite state concurrent systems using temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 8:244–263, 1986.

[CGK89] E. M. Clarke, O. Grümberg, and R. P. Kurshan. A synthesis of two approaches for verifiying finite state concurrent systems. In *Proc. Logic at Botik*, 1989. To appear.

[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.

[dBdR73] J. de Bakker and W. P. de Roever. A calculus for recursive program schemes. In *Proc. 1st International Colloq. on Automata, Languages, and Programming*, pages 167–196, 1973.

[EH82] E. A. Emerson and J. Y. Halpern. Decision procedures and expressiveness in temporal logic of branching time. In *Proc. 14th ACM Symp. on Theory of Computing*, pages 169–180, 1982.

[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.

[EL85] E. A. Emerson and C. L. Lei. Modalities for model checking: Branching time strikes back. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 84–96, 1985.

[Eme89] E. A. Emerson. Temporal and modal logic. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, 1989. To appear.

[Eng67] E. Engeler. Algorithmic properties of structures. *Math. Sys. Theory*, 1:183–195, 1967.

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

[FL79] M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, 18(2):194–211, 1979.

[Flo67] R. W. Floyd. Assigning meanings to programs. In *19th AMS Symp. Applied Math.*, AMS, Providence, R.I., pages 19–31, 1967.

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

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pages 163–173, 1980.

[Har79] D. Harel. *First-Order Dynamic Logic*, volume 68 of *Lecture Notes in Computer Science*. Springer, 1979.

[Har84] D. Harel. Dynamic logic. In Gabbay and Guenthner, editors, *Handbook of Philosophical Logic. II: Extensions of Classical Logic*, pages 497–604. D. Reidel, Boston, 1984.

[HMP77] D. Harel, A. R. Meyer, and V. R. Pratt. Computability and completeness in logics of programs. In *Proc. 9th ACM Symp. on Theory of Computing*, 1977.

[Hoa67] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12:576–580, 1967.

[HP73] P. Hitchkock and D. Park. Induction rules and termination proofs. In *Proc. 1st International Colloq. on Automata, Languages, and Programming*, pages 225–252, 1973.

[HP78] D. Harel and V. R. Pratt. Nondeterminism in logics of programs. In *Proc. 5th ACM Symp. on Principles of Programming Languages*, pages 203–213, 1978.

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

[HR72] R. Hossley and C. Rackoff. The emptiness problem for automata on infinite trees. In *Proc. 13th IEEE Symp. Switching and Automata Theory*, pages 121–124, 1972.

[HR83] J. Y. Halpern and J. H. Reif. The propositional dynamic logic of deterministic well-structured programs. *Theoretical Computer Science*, 27:127–165, 1983.

[Kam85] M. Kaminski. A classification of ω-regular languages. *Theoretical Computer Science*, 36:217–229, 1985.

[Kön27] D. König. Über eine schlubweise aus dem endlichen ins unendliche. *Acta Litt. Sci. Szeged*, 3:121–130, 1927.

[Koz83] D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

[KT89] D. Kozen and J. Tiuryn. Logics of program. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, 1989. To appear (also TR 88-962, Cornall University).

[Lam80] L. Lamport. "Somptimes" is somtimes "Not Never". In *Proc. 7th ACM Symp. on Principles of Programming Languages*, pages 174–185, 1980.

[LP84] O. Lichtenstien and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 11th ACM Symp. on Principles of Programming Languages*, pages 97–107, 1984.

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

[Mey75] A. R. Meyer. Weak monadic second order theory of successor is not elementary recursive. In *Proc. Boston Univ. Logic Colloquium, 1973*, volume 453, pages 132–154. Lecture Notes in Mathematics 453, Springer, 1975.

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

[MP71] R. McNaughton and S. Papert. *Counter Free Automata.* MIT Press, 1971.

[MP87] Z. Manna and A. Pnueli. A hierarchy of temporal properties. Technical report, Dept. of Comp. Sci., Stanford University, 1987.

[MP01] Z. Manna and A. Pnueli. *The Temporal Logic of Reactive Systems.* 2001. To appear.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proc. 13th IEEE Symp. Switching and Automata Theory*, pages 125–129, 1972.

[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.

[MW84] Z. Manna and P. Wolper. Synthesis of communicating processes from temporal logic specifications. *ACM Trans. on Programming Languages and Systems*, 6:68–93, 1984.

[Par69] D. Park. Fixpoint induction and proofs of program properties. In *Machine Intelligence 5.* Edinburgh Univ. Press, Edinburgh, 1969.

[Par76] D. Park. Finiteness is μ-ineffable. *Theoretical Computer Science*, 3:176–181, 1976.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. on Foundations of Computer Science*, pages 46–57, 1977.

[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.

[Pra76] V. R. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proc. 17th IEEE Symp. on Foundations of Computer Science*, pages 109–121, 1976.

[Pra79] V. R. Pratt. Models of program logic. In *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pages 115–122, 1979.

[Pra80] V. R. Pratt. A near-optimal method for reasoning about action. *Journal of Computer and System Science*, 20:231–254, 1980.

[Pra82] V. R. Pratt. A decidable μ-calculus: Preliminary report. In *Proc. 23rd IEEE Symp. on Foundations of Computer Science*, pages 421–427, 1982.

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

[Rab70] M. O. Rabin. Weakly definable relations and special automata. In Y. Bar-hillel, editor, *Proc. Symp. Math. Logic and Foundation of Set Theory*, pages 1–23. North-Holland, 1970.

[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.

[SC85] A. P. Sistla and E. M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32(3):733–749, 1985.

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

[SE84] A. P. Streett and A. E. Emerson. The propositional μ-calculus is elementary. In *Proc. 11th International Colloq. on Automata, Languages, and Programming*, pages 465–472. Lecture Notes in Computer Science 172, 1984.

[She84] R. Sherman. *On Variants of Propositional Program Logics*. PhD thesis, Weizmann Institute, 1984.

[Sto74] L. J. Stockmeyer. *The Complexity of Decision Problems in Automata Theory and Logic*. PhD thesis, MIT, July 1974.

[Str80] R. S. Streett. A propositional dynamic logic for reasoning about program divergence. Master's thesis, MIT, 1980.

[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.

[Tho89] W. Thomas. Automata on infinite objects. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science*. North-Holland, 1989. To appear.

[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.

[Var88] M. Y. Vardi. A temporal fixpoint calculus. In *Proc. 15th ACM Symp. on Principles of Programming Languages*, pages 250–259, 1988.

[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.

[VW86] M.Y. Vardi and P. Wolper. Automata theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32:183–221, 1986.

[Wag79] K. Wagner. On ω-regular sets. *Information and Control*, 43:123–177, 1979.

[Wol83] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.

[WVS83] P. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about infinite computation paths. In *Proc. 24th IEEE Symp. on Foundations of Computer Science*, pages 185–194, 1983.