Automata Theory and Model Checking

Orna Kupferman

Abstract We study automata on infinite words and their applications in system specification and verification. We first introduce Büchi automata and survey their closure properties, expressive power, and determinization. We then introduce additional acceptance conditions and the model of alternating automata. We compare the different classes of automata in terms of expressive power and succinctness, and describe decision problems for them. Finally, we describe the automata-theoretic approach to system specification and verification.

1 Introduction

Finite automata on infinite objects were first introduced in the 1960s. Motivated by decision problems in mathematics and logic, Büchi, McNaughton, and Rabin developed a framework for reasoning about infinite words and infinite trees [6, 52, 61]. The framework has proved to be very powerful. Automata and their tight relation to second-order monadic logics were the key to the solution of several fundamental decision problems in mathematics and logic [62, 74]. Today, automata on infinite objects are used for specification and verification of nonterminating systems. The idea is simple: when a system is defined with respect to a finite set AP of propositions, each of the system's states can be associated with a set of propositions that hold in this state. Then, each of the system's computations induces an infinite word over the alphabet 2^{AP}, and the system itself induces a language of infinite words over this alphabet. This language can be defined by an automaton. Similarly, a system specification, which describes all the allowed computations, can be viewed as a language of infinite words over 2^{AP} , and can therefore be defined by an automaton. In the automata-theoretic approach to verification, we reduce questions about sys-

Orna Kupferman
School of Computer Science and Engineering, Hebrew University, Jerusalem 91904, Israel. e-mail: orna@cs.huji.ac.il

tems and their specifications to questions about automata. More specifically, questions such as satisfiability of specifications and correctness of systems with respect to their specifications are reduced to questions such as non-emptiness and language containment [79, 48, 77].

The automata-theoretic approach separates the logical and the combinatorial aspects of reasoning about systems. The translation of specifications to automata handles the logic and shifts all the combinatorial difficulties to automata-theoretic problems, yielding clean and asymptotically optimal algorithms, as well as better understanding of the complexity of the problems. Beyond leading to tight complexity bounds, automata have proven to be very helpful in practice. Automata are the key to techniques such as on-the-fly model checking [21, 11], and they are useful also for modular model checking [41], partial-order model checking [75, 31, 23, 78], model checking of real-time and hybrid systems [26], open systems [1], and infinite-state systems [43, 40]. Automata also serve as expressive specification formalisms [39, 2] and in algorithms for sanity checks [37]. Automata-based methods have been implemented in both academic and industrial automated-verification tools (e.g., COSPAN [24], SPIN [27], ForSpec [72], and NuSMV [9]).

This chapter studies automata on infinite words and their applications in system specification and verification. We first introduce Büchi automata, survey their closure properties, expressive power, and determinization. We then introduce additional acceptance conditions and the model of alternating automata. We compare the different classes of automata in terms of expressive power and succinctness, and describe decision problems for them. Finally, we describe the automata-theoretic approach to system specification and verification.

2 Nondeterministic Büchi Automata on Infinite Words

2.1 Definitions

For a finite alphabet Σ, an infinite word $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots$ is an infinite sequence of letters from Σ. We use Σ^ω to denote the set of all infinite words over the alphabet Σ. A language $L \subseteq \Sigma^\omega$ is a set of words. We sometimes refer also to finite words, and to languages $L \subseteq \Sigma^*$ of finite words over Σ. A *prefix* of $w = \sigma_1 \cdot \sigma_2 \cdots$ is a (possibly empty) finite word $\sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots \sigma_i$, for some $i \geq 0$. A *suffix* of w is an infinite word $\sigma_i \cdot \sigma_{i+1} \cdots$, for some $i \geq 1$. A property of a system with a set AP of atomic propositions can be viewed as a language over the alphabet 2^{AP}. We have seen in Chapter 2 that languages over this alphabet can be defined by linear temporal-logic (LTL, for short) formulas. Another way to define languages is by automata.

A *nondeterministic finite automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, where Σ is a finite non-empty alphabet, Q is a finite non-empty set of *states*, $Q_0 \subseteq Q$ is a non-empty set of *initial states*, $\delta : Q \times \Sigma \to 2^Q$ is a *transition function*, and α is an *acceptance condition*, to be defined below.

Intuitively, when the automaton $\mathscr{A}$ runs on an input word over Σ, it starts in one of the initial states, and it proceeds along the word according to the transition function. Thus, $\delta(q,\sigma)$ is the set of states that $\mathscr{A}$ can move into when it is in state q and it reads the letter σ. Note that the automaton may be *nondeterministic*, since it may have several initial states and the transition function may specify several possible transitions for each state and letter. The automaton $\mathscr{A}$ is *deterministic* if $|Q_0| = 1$ and $|\delta(q,\sigma)| = 1$ for all states $q \in Q$ and symbols $\sigma \in \Sigma$. Specifying deterministic automata, we sometimes describe the single initial state or destination state, rather than a singleton set.

Formally, a *run* r of $\mathscr{A}$ on a finite word $w = \sigma_1 \cdot \sigma_2 \cdots \sigma_n \in \Sigma^*$ is a sequence $r = q_0, q_1, \ldots, q_n$ of $n+1$ states in Q such that $q_0 \in Q_0$, and for all $0 \leq i < n$ we have $q_{i+1} \in \delta(q_i, \sigma_{i+1})$. Note that a nondeterministic automaton may have several runs on a given input word. In contrast, a deterministic automaton has exactly one run on a given input word. When the input word is infinite, thus $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots \in \Sigma^\omega$, then a run of $\mathscr{A}$ on it is an infinite sequence of states $r = q_0, q_1, q_2, \ldots$ such that $q_0 \in Q_0$, and for all $i \geq 0$, we have $q_{i+1} \in \delta(q_i, \sigma_{i+1})$. For an infinite run r, let $inf(r) = \{q : q_i = q \text{ for infinitely many } i\text{'s}\}$. Thus, $inf(r)$ is the set of states that r visits infinitely often.

The acceptance condition α determines which runs are "good". For automata on finite words, $\alpha \subseteq Q$ and a run r is *accepting* if $q_n \in \alpha$. For automata on infinite words, one can consider several acceptance conditions. Let us start with the Büchi acceptance condition [6]. There, $\alpha \subseteq Q$, and a run r is accepting if it visits some state in α infinitely often. Formally, r is accepting iff $inf(r) \cap \alpha \neq \emptyset$. A run that is not accepting is *rejecting*. A word w is accepted by an automaton $\mathscr{A}$ if there is an accepting run of $\mathscr{A}$ on w. The language recognized by $\mathscr{A}$, denoted $\mathscr{L}(\mathscr{A})$, is the set of words that $\mathscr{A}$ accepts. We sometimes refer to $\mathscr{L}(\mathscr{A})$ also as the language of $\mathscr{A}$.

We use NBW and DBW to abbreviate nondeterministic and deterministic Büchi automata, respectively.[1] For a class γ of automata (so far, we have introduced $\gamma \in \{\text{NBW}, \text{DBW}\}$), we say that a language $L \subseteq \Sigma^\omega$ is γ-recognizable iff there is an automaton in the class γ that recognizes L. A language is *ω-regular* iff it is NBW-recognizable.

Example 1. Consider the DBW $\mathscr{A}_1$ appearing in Figure 1. When we draw automata, states are denoted by circles. Directed edges between states are labeled with letters and describe the transitions. Initial states (q_0, in the figure) have an edge entering them with no source, and accepting states (q_1, in the figure) are identified by double circles. The DBW moves to the accepting state whenever it reads the letter a, and it moves to the non-accepting state whenever it reads the letter b. Accordingly, the single run r on a word w visits the accepting state infinitely often iff w has infinitely many a's. Hence, $\mathscr{L}(\mathscr{A}_1) = \{w : w \text{ has infinitely many } a\text{'s}\}$.

Example 2. Consider the NBW $\mathscr{A}_2$ appearing in Figure 2. The automaton is nondeterministic, and in order for a run to be accepting it has to eventually move to

[1] The letter W indicates that the automata run on words (rather than, say, trees).

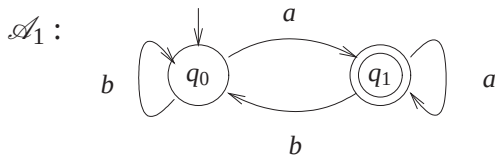

Fig. 1 A DBW for $\{w : w$ has infinitely many a's $\}$

the accepting state, where it has to stay forever while reading b. Note that if $\mathscr{A}_2$ reads a from the accepting state it gets stuck. Accordingly, $\mathscr{A}_2$ has an accepting run on a word w iff w has a position from which an infinite tail of b's starts. Hence, $\mathscr{L}(\mathscr{A}_2) = \{w : w$ has only finitely many a's$\}$.

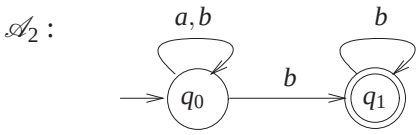

Fig. 2 An NBW for $\{w : w$ has only finitely many a's$\}$

Consider a directed graph $G = \langle V, E \rangle$. A *strongly connected set* of G (SCS) is a set $C \subseteq V$ of vertices such that for every two vertices $v, v' \in C$, there is a path from v to v'. An SCS C is *maximal* if it cannot be extended to a larger SCS. Formally, for every nonempty $C' \subseteq V \setminus C$, we have that $C \cup C'$ is not an SCS. The maximal strongly connected sets are also termed *strongly connected components* (SCC). An automaton $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ induces a directed graph $G_{\mathscr{A}} = \langle Q, E \rangle$ in which $\langle q, q' \rangle \in E$ iff there is a letter σ such that $q' \in \delta(q, \sigma)$. When we talk about the SCSs and SCCs of $\mathscr{A}$, we refer to those of $G_{\mathscr{A}}$. Consider a run r of an automaton $\mathscr{A}$. It is not hard to see that the set $inf(r)$ is an SCS. Indeed, since every two states q and q' in $inf(r)$ are visited infinitely often, the state q' must be reachable from q.

2.2 Closure Properties

Automata on finite words are closed under union, intersection, and complementation. In this section we study closure properties for nondeterministic Büchi automata.

2.2.1 Closure Under Union and Intersection

We start with closure under union, where the construction that works for nondeterministic automata on finite words, namely putting the two automata "one next to the other", works also for nondeterministic Büchi automata. Formally, we have the following.

Theorem 1. [8] *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be NBWs with n_1 and n_2 states, respectively. There is an NBW $\mathscr{A}$ such that $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}_1) \cup \mathscr{L}(\mathscr{A}_2)$ and $\mathscr{A}$ has $n_1 + n_2$ states.*

Proof: Let $\mathscr{A}_1 = \langle \Sigma, Q_1, Q_1^0, \delta_1, \alpha_1 \rangle$ and $\mathscr{A}_2 = \langle \Sigma, Q_2, Q_2^0, \delta_2, \alpha_2 \rangle$. We assume, without loss of generality, that Q_1 and Q_2 are disjoint. Since nondeterministic automata may have several initial states, we can define $\mathscr{A}$ as the NBW obtained by taking the union of $\mathscr{A}_1$ and $\mathscr{A}_2$. Thus, $\mathscr{A} = \langle \Sigma, Q_1 \cup Q_2, Q_1^0 \cup Q_2^0, \delta, \alpha_1 \cup \alpha_2 \rangle$, where for every state $q \in Q_1 \cup Q_2$, we have that $\delta(q,\sigma) = \delta_i(q,\sigma)$, for the index $i \in \{1,2\}$ such that $q \in Q_i$. It is easy to see that for every word $w \in \Sigma^\omega$, the NBW $\mathscr{A}$ has an accepting run on w iff at least one of the NBWs $\mathscr{A}_1$ and $\mathscr{A}_2$ has an accepting run on w. □

We proceed to closure under intersection. For the case of finite words, one proves closure under intersection by constructing, given $\mathscr{A}_1$ and $\mathscr{A}_2$, a "product automaton" that has $Q_1 \times Q_2$ as its state space and simulates the runs of both $\mathscr{A}_1$ and $\mathscr{A}_2$ on the input words. A word is then accepted by both $\mathscr{A}_1$ and $\mathscr{A}_2$ iff the product automaton has a run that leads to a state in $\alpha_1 \times \alpha_2$. As the example below demonstrates, this construction does not work for Büchi automata.

Example 3. Consider the two DBWs $\mathscr{A}_1$ and $\mathscr{A}_2$ on the left of Figure 3. The product automaton $\mathscr{A}_1 \times \mathscr{A}_2$ is shown on the right. Clearly, $\mathscr{L}(\mathscr{A}_1) = \mathscr{L}(\mathscr{A}_2) = \{a^\omega\}$, but $\mathscr{L}(\mathscr{A}_1 \times \mathscr{A}_2) = \emptyset$.

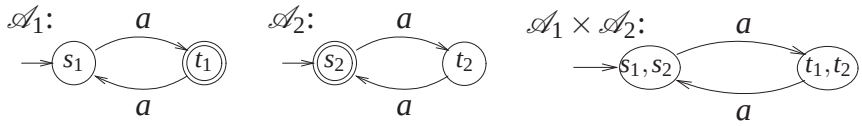

Fig. 3 Two Büchi automata accepting the language $\{a^\omega\}$, and their empty product

As demonstrated in Example 3, the problem with the product automaton is that the definition of the set of accepting states to be $\alpha_1 \times \alpha_2$ forces the accepting runs of $\mathscr{A}_1$ and $\mathscr{A}_2$ to visit α_1 and α_2 simultaneously. This requirement is too strong, as an input word may still be accepted by both $\mathscr{A}_1$ and $\mathscr{A}_2$, but the accepting runs on it visit α_1 and α_2 in different positions. As we show below, the product automaton is a good basis for proving closure under intersection, but one needs to take two copies of it: one that waits for visits of runs of $\mathscr{A}_1$ to α_1 (and moves to the second copy when such a visit is detected) and one that waits for visits of runs of $\mathscr{A}_2$ to α_2 (and

returns to the first copy when such a visit is detected). The acceptance condition then requires the run to alternate between the two copies infinitely often, which is possible exactly when both the run of $\mathscr{A}_1$ visits α_1 infinitely often, and the run of $\mathscr{A}_2$ visits α_2 infinitely often. Note that $\mathscr{A}_2$ may visit α_2 when the run is in the first copy, in which case the visit to α_2 is ignored, and in fact this may happen infinitely many times. Still, if there are infinitely many visits to α_1 and α_2, then eventually the run moves to the second copy, where it eventually comes across a visit to α_2 that is not ignored. Formally, we have the following.

Theorem 2. [8] *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be NBWs with n_1 and n_2 states, respectively. There is an NBW $\mathscr{A}$ such that $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}_1) \cap \mathscr{L}(\mathscr{A}_2)$ and $\mathscr{A}$ has $2n_1n_2$ states.*

Proof: Let $\mathscr{A}_1 = \langle \Sigma, Q_1, Q_1^0, \delta_1, \alpha_1 \rangle$ and $\mathscr{A}_2 = \langle \Sigma, Q_2, Q_2^0, \delta_2, \alpha_2 \rangle$. We define $\mathscr{A} = \langle \Sigma, Q, Q^0, \delta, \alpha \rangle$, where

- $Q = Q_1 \times Q_2 \times \{1,2\}$. That is, the state space consists of two copies of the product automaton.
- $Q^0 = Q_1^0 \times Q_2^0 \times \{1\}$. That is, the initial states are triples $\langle s_1, s_2, 1 \rangle$ such that s_1 and s_2 are initial in $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively. The run starts in the first copy.
- For all $q_1 \in Q_1$, $q_2 \in Q_2$, $c \in \{1,2\}$, and $\sigma \in \Sigma$, we define $\delta(\langle s_1, s_2, c \rangle, \sigma) = \delta_1(s_1, \sigma) \times \delta_2(s_2, \sigma) \times \{\mathit{next}(s_1, s_2, c)\}$, where

$$\mathit{next}(s_1, s_2, c) = \begin{cases} 1 & \text{if } (c = 1 \text{ and } s_1 \notin \alpha_1) \text{ or } (c = 2 \text{ and } s_2 \in \alpha_2), \\ 2 & \text{if } (c = 1 \text{ and } s_1 \in \alpha_1) \text{ or } (c = 2 \text{ and } s_2 \notin \alpha_2). \end{cases}$$

  That is, $\mathscr{A}$ proceeds according to the product automaton, and it moves from the first copy to the second copy when $s_1 \in \alpha_1$, and from the second copy to the first copy when $s_2 \in \alpha_2$. In all other cases it stays in the current copy.
- $\alpha = \alpha_1 \times Q_2 \times \{1\}$. That is, a run of $\mathscr{A}$ is accepting if it visits infinitely many states in the first copy in which the Q_1-component is in α_1. Note that after such a visit, $\mathscr{A}$ moves to the second copy, from which it returns to the first copy after visiting a state in which the Q_2-component is in α_2. Accordingly, there must be a visit to a state in which the Q_2-component is in α_2 between every two successive visits to states in α. This is why a run visits α infinitely often iff its Q_1-component visits α_1 infinitely often and its Q_2-component visits α_2 infinitely often.

□

Note that the product construction retains determinism; i.e., starting with deterministic $\mathscr{A}_1$ and $\mathscr{A}_2$, the product $\mathscr{A}$ is deterministic. Thus, DBWs are also closed under intersection. Also, while the union construction we have described does not retain determinism, DBWs are closed also under union. Indeed, if we take the product construction (one copy of it is sufficient), which retains determinism, and define the set of accepting states to be $(\alpha_1 \times Q_2) \cup (Q_1 \times \alpha_2)$, we get a DBW for the union. Note, however, that unlike the $n_1 + n_2$ blow-up in Theorem 1, the blow-up now is n_1n_2.

2.2.2 Closure Under Complementation

For deterministic automata on finite words, complementation is easy: the single run is rejecting iff its last state is not accepting, thus complementing a deterministic automaton can proceed by *dualizing* its acceptance condition: for an automaton with state space Q and set α of accepting states, the dual acceptance condition is $\tilde{\alpha} = Q \setminus \alpha$, and it is easy to see that dualizing the acceptance condition of a deterministic automaton on finite words results in a deterministic automaton for the complement language. It is also easy to see that such a simple dualization does not work for DBWs. Indeed, a run of a Büchi automaton is rejecting iff it visits α only finitely often, which is different from requiring it to visit $\tilde{\alpha}$ infinitely often. As a concrete example, consider the DBW $\mathscr{A}_1$ from Figure 1. Recall that $\mathscr{L}(\mathscr{A}_1) = \{w : w \text{ has infinitely many } a\text{'s }\}$. An attempt to complement it by defining the set of accepting states to be $\{q_0\}$ results in a DBW whose language is $\{w : w$ has infinitely many b's $\}$, which does not complement $\mathscr{L}(\mathscr{A}_1)$. For example, the word $(a \cdot b)^\omega$ belongs to both languages. In this section we study the complementation problem for Büchi automata. We start with deterministic automata and show that while dualization does not work, their complementation is quite simple, but results in a nondeterministic automaton. We then move on to nondeterministic automata, and describe a complementation procedure for them.

Theorem 3. [47] *Let $\mathscr{A}$ be a DBW with n states. There is an NBW $\mathscr{A}'$ such that $\mathscr{L}(\mathscr{A}') = \Sigma^\omega \setminus \mathscr{L}(\mathscr{A})$, and $\mathscr{A}'$ has at most $2n$ states.*

Proof: Let $\mathscr{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. The NBW $\mathscr{A}'$ should accept exactly all words w for which the single run of $\mathscr{A}$ on w visits α only finitely often. It does so by guessing a position from which no more visits of $\mathscr{A}$ to α take place. For that, $\mathscr{A}'$ consists of two copies of $\mathscr{A}$: one that includes all the states and transitions of $\mathscr{A}$, and one that excludes the accepting states of $\mathscr{A}$, and to which $\mathscr{A}'$ moves when it guesses that no more states in α are going to be visited. All the states in the second copy are accepting. Formally, $\mathscr{A}' = \langle \Sigma, Q', Q'_0, \delta', \alpha' \rangle$, where

- $Q' = (Q \times \{0\}) \cup ((Q \setminus \alpha) \times \{1\})$.
- $Q'_0 = \{\langle q_0, 0 \rangle\}$.
- For every $q \in Q$, $c \in \{0,1\}$, and $\sigma \in \Sigma$ with $\delta(q,\sigma) = q'$, we have

$$\delta'(\langle q,c \rangle, \sigma) = \begin{cases} \{\langle q',0 \rangle, \langle q',1 \rangle\} & \text{if } c = 0 \text{ and } q' \notin \alpha, \\ \{\langle q',0 \rangle\} & \text{if } c = 0 \text{ and } q' \in \alpha, \\ \{\langle q',1 \rangle\} & \text{if } c = 1 \text{ and } q' \notin \alpha, \\ \emptyset & \text{if } c = 1 \text{ and } q' \in \alpha. \end{cases}$$

- $\alpha' = (Q \setminus \alpha) \times \{1\}$.

Thus, $\mathscr{A}'$ can stay in the first copy forever, but in order for a run of $\mathscr{A}'$ to be accepting, it must eventually move to the second copy, from where it cannot go back to the first copy and must avoid states in α. □

The construction described in the proof of Theorem 3 can be applied also to nondeterministic automata. Since, however, $\mathscr{A}'$ accepts a word w iff there exists a run of $\mathscr{A}$ on w that visits α only finitely often, whereas a complementing automaton should accept a word w iff all the runs of $\mathscr{A}$ on w visit α only finitely often, the construction has a one-sided error when applied to nondeterministic automata. This is not surprising, as the same difficulty exists when we complement nondeterministic automata on finite words. By restricting attention to deterministic automata, we guarantee that the existential and universal quantification on the runs of $\mathscr{A}$ coincide.

We now turn to consider complementation for nondeterministic Büchi automata. In the case of finite words, one first determinizes the automaton and then complements the result. An attempt to follow a similar plan for NBWs, namely a translation to a DBW and then an application of Theorem 3, does not work: as we shall see in Section 2.3, DBWs are strictly less expressive than NBWs, thus not all NBWs can be determinized. Nevertheless, NBWs are closed under complementation.

Efforts to develop a complementation construction for NBWs started in the early 1960s, motivated by decision problems for second-order logics. Büchi introduced a complementation construction that involved a complicated Ramsey-based combinatorial argument and a doubly-exponential blow-up in the state space [6]. Thus, complementing an NBW with n states resulted in an NBW with $2^{2^{O(n)}}$ states. In [70], Sistla et al. suggested an improved implementation of Büchi's construction, with only $2^{O(n^2)}$ states, which is still not optimal. [2] Only in [64], Safra introduced a determinization construction that involves an acceptance condition that is stronger than Büchi, and used it in order to present a $2^{O(n \log n)}$ complementation construction, matching the known lower bound [54]. The use of complementation in practice has led to a resurgent interest in the exact blow-up that complementation involves and the feasibility of the complementation construction (e.g., issues like whether the construction can be implemented symbolically, whether it is amenable to optimizations or heuristics – these are all important criteria that complementation constructions that involve determinization do not satisfy). In [33], Klarlund introduced an optimal complementation construction that avoids determinization. Rather, the states of the complementing automaton utilize *progress measures* – a generic concept for quantifying how each step of a system contributes to bringing a computation closer to its specification. In [44], Kupferman and Vardi used ranks, which are similar to progress measures, in a complementation construction that goes via intermediate alternating co-Büchi automata. Below we describe the construction of [44] circumventing the intermediate alternating automata.

Let $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ be an NBW with n states. Let $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots$ be a word in Σ^ω. We define an infinite DAG G that embodies all the possible runs of $\mathscr{A}$ on w. Formally, $G = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is the union $\bigcup_{l \geq 0}(Q_l \times \{l\})$, where for all $l \geq 0$, we have $Q_{l+1} = \bigcup_{q \in Q_l} \delta(q, \sigma_{l+1})$.

[2] Interestingly, by carrying out some simple optimizations, the Ramsey-based approach in the constructions in [6] and [70] can be improved to produce complementing NBWs with the optimal $2^{O(n \log n)}$ blow-up [5].

- $E \subseteq \bigcup_{l \geq 0}(Q_l \times \{l\}) \times (Q_{l+1} \times \{l+1\})$ is such that for all $l \geq 0$, we have $E(\langle q,l\rangle, \langle q',l+1\rangle)$ iff $q' \in \delta(q, \sigma_{l+1})$.

We refer to G as the *run* DAG of $\mathscr{A}$ on w. We say that a vertex $\langle q',l'\rangle$ is a *successor* of a vertex $\langle q,l\rangle$ iff $E(\langle q,l\rangle, \langle q',l'\rangle)$. We say that $\langle q',l'\rangle$ is *reachable* from $\langle q,l\rangle$ iff there exists a sequence $\langle q_0,l_0\rangle, \langle q_1,l_1\rangle, \langle q_2,l_2\rangle, \ldots$ of successive vertices such that $\langle q,l\rangle = \langle q_0,l_0\rangle$, and there exists $i \geq 0$ such that $\langle q',l'\rangle = \langle q_i,l_i\rangle$. We say that a vertex $\langle q,l\rangle$ is an α*-vertex* iff $q \in \alpha$. Finally, we say that G is an *accepting* run DAG if G has a path with infinitely many α-vertices. Otherwise, we say that G is rejecting. It is easy to see that $\mathscr{A}$ accepts w iff G is accepting.

For $k \in \mathbb{N}$, let $[k]$ denote the set $\{0,1,\ldots,k\}$. A *ranking* for G is a function $f : V \to [2n]$ that satisfies the following two conditions:

1. For all vertices $\langle q,l\rangle \in V$, if $f(\langle q,l\rangle)$ is odd, then $q \notin \alpha$.
2. For all edges $\langle\langle q,l\rangle, \langle q',l'\rangle\rangle \in E$, we have $f(\langle q',l'\rangle) \leq f(\langle q,l\rangle)$.

Thus, a ranking associates with each vertex in G a rank in $[2n]$ so that the ranks along paths decrease monotonically, and α-vertices get only even ranks. Note that each path in G eventually gets trapped in some rank. We say that the ranking f is an *odd ranking* if all the paths of G eventually get trapped in an odd rank. Formally, f is odd iff for all paths $\langle q_0,0\rangle, \langle q_1,1\rangle, \langle q_2,2\rangle, \ldots$ in G, there is $j \geq 0$ such that $f(\langle q_j,j\rangle)$ is odd, and for all $i \geq 1$, we have $f(\langle q_{j+i}, j+i\rangle) = f(\langle q_j,j\rangle)$.

We are going to prove that G is rejecting iff it has an odd ranking. The difficult direction is to show that if G is rejecting, then it has an odd ranking. Below we make some observations on rejecting run DAGs that help us with this direction. We say that a vertex $\langle q,l\rangle$ is *finite* in a DAG $G' \subseteq G$ iff only finitely many vertices in G' are reachable from $\langle q,l\rangle$. The vertex $\langle q,l\rangle$ is α*-free* in G' iff all the vertices in G' that are reachable from $\langle q,l\rangle$ are not α-vertices. Note that, in particular, an α-free vertex is not an α-vertex. We define an infinite sequence $G_0 \supseteq G_1 \supseteq G_2 \supseteq \ldots$ of DAGs inductively as follows.

- $G_0 = G$.
- For $i \geq 0$, we have $G_{2i+1} = G_{2i} \setminus \{\langle q,l\rangle \ : \ \langle q,l\rangle \text{ is finite in } G_{2i}\}$.
- For $i \geq 0$, we have $G_{2i+2} = G_{2i+1} \setminus \{\langle q,l\rangle \ : \ \langle q,l\rangle \text{ is } \alpha\text{-free in } G_{2i+1}\}$.

Lemma 1. *If G is rejecting, then for every $i \geq 0$, there exists l_i such that for all $l \geq l_i$, there are at most $n-i$ vertices of the form $\langle q,l\rangle$ in G_{2i}.*

Proof: We prove the lemma by an induction on i. The case where $i = 0$ follows from the definition of $G_0 = G$. Indeed, in G all levels $l \geq 0$ have at most n vertices of the form $\langle q,l\rangle$. Assume that the lemma's requirement holds for i; we prove it for $i+1$. Consider the DAG G_{2i}. We distinguish between two cases. First, if G_{2i} is finite, then G_{2i+1} is empty, G_{2i+2} is empty as well, and we are done. Otherwise, we claim that there must be some α-free vertex in G_{2i+1}. To see this, assume, by way of contradiction, that G_{2i} is infinite and no vertex in G_{2i+1} is α-free. Since G_{2i} is infinite, G_{2i+1} is also infinite. Also, each vertex in G_{2i+1} has at least one successor. Consider some vertex $\langle q_0,l_0\rangle$ in G_{2i+1}. Since, by the assumption, it is not α-free, there exists an α-vertex $\langle q'_0,l'_0\rangle$ reachable from $\langle q_0,l_0\rangle$. Let $\langle q_1,l_1\rangle$ be a successor

of $\langle q'_0, l'_0\rangle$. By the assumption, $\langle q_1, l_1\rangle$ is also not α-free. Hence, there exists an α-vertex $\langle q'_1, l'_1\rangle$ reachable from $\langle q_1, l_1\rangle$. Let $\langle q_2, l_2\rangle$ be a successor of $\langle q'_1, j'_1\rangle$. By the assumption, $\langle q_2, l_2\rangle$ is also not α-free. Thus, we can continue similarly and construct an infinite sequence of vertices $\langle q_j, l_j\rangle$, $\langle q'_j, l'_j\rangle$ such that for all j, the vertex $\langle q'_j, l'_j\rangle$ is an α-vertex reachable from $\langle q_j, l_j\rangle$, and $\langle q_{j+1}, l_{j+1}\rangle$ is a successor of $\langle q'_j, l'_j\rangle$. Such a sequence, however, corresponds to a path in G with infinitely many α-vertices, contradicting the assumption that G is rejecting.

So, let $\langle q, l\rangle$ be an α-free vertex in G_{2i+1}. We claim that taking $l_{i+1} = \max\{l, l_i\}$ satisfies the requirement of the lemma. That is, we claim that for all $j \geq \max\{l, l_i\}$, there are at most $n-(i+1)$ vertices of the form $\langle q, j\rangle$ in G_{2i+2}. Since $\langle q, l\rangle$ is in G_{2i+1}, it is not finite in G_{2i}. Thus, there are infinitely many vertices in G_{2i} that are reachable from $\langle q, l\rangle$. Hence, by König's Lemma, G_{2i} contains an infinite path $\langle q, l\rangle, \langle q_1, l+1\rangle, \langle q_2, l+2\rangle, \ldots$. For all $k \geq 1$, the vertex $\langle q_k, l+k\rangle$ has infinitely many vertices reachable from it in G_{2i} and thus, it is not finite in G_{2i}. Therefore, the path $\langle q, l\rangle, \langle q_1, l+1\rangle, \langle q_2, l+2\rangle, \ldots$ exists also in G_{2i+1}. Recall that $\langle q, l\rangle$ is α-free. Hence, being reachable from $\langle q, l\rangle$, all the vertices $\langle q_k, l+k\rangle$ in the path are α-free as well. Therefore, they are not in G_{2i+2}. It follows that for all $j \geq l$, the number of vertices of the form $\langle q, j\rangle$ in G_{2i+2} is strictly smaller than their number in G_{2i}. Hence, by the induction hypothesis, we are done. □

Note that, in particular, by Lemma 1, if G is rejecting then G_{2n} is finite. Hence the following corollary.

Corollary 1. *If G is rejecting then G_{2n+1} is empty.*

We can now prove the main lemma required for complementation, which reduces the fact that all the runs of $\mathscr{A}$ on w are rejecting to the existence of an odd ranking for the run DAG of $\mathscr{A}$ on w.

Lemma 2. *An NBW $\mathscr{A}$ rejects a word w iff there is an odd ranking for the run* DAG *of $\mathscr{A}$ on w.*

Proof: Let G be the run DAG of $\mathscr{A}$ on w. We first claim that if there is an odd ranking for G, then $\mathscr{A}$ rejects w. To see this, recall that in an odd ranking, every path in G eventually gets trapped in an odd rank. Hence, as α-vertices get only even ranks, it follows that all the paths of G, and thus all the possible runs of $\mathscr{A}$ on w, visit α only finitely often.

Assume now that $\mathscr{A}$ rejects w. We describe an odd ranking for G. Recall that if $\mathscr{A}$ rejects w, then G is rejecting and thus, by Corollary 1, each vertex $\langle q, l\rangle$ in G is removed from G_j, for some $0 \leq j \leq 2n$. Thus, there is $0 \leq i \leq n$ such that $\langle q, l\rangle$ is finite in G_{2i} or α-free in G_{2i+1}. Given a vertex $\langle q, l\rangle$, we define the *rank* of $\langle q, l\rangle$, denoted $f(q, l)$, as follows.

$$f(q,l) = \begin{cases} 2i & \text{if } \langle q,l\rangle \text{ is finite in } G_{2i}. \\ 2i+1 & \text{if } \langle q,l\rangle \text{ is } \alpha\text{-free in } G_{2i+1}. \end{cases}$$

We claim that f is an odd ranking for G. First, by Lemma 1, the subgraph G_{2n} is finite. Hence, the maximal rank that a vertex can get is $2n$. Also, since an α-free

vertex cannot be an α-vertex and $f(\langle q,l\rangle)$ is odd only for α-free $\langle q,l\rangle$, the first condition for f being a ranking holds. We proceed to the second condition. We first argue (and a proof proceeds easily by an induction on i) that for every vertex $\langle q,l\rangle$ in G and rank $i \in [2n]$, if $\langle q,l\rangle \notin G_i$, then $f(q,l) < i$. Now, we prove that for every two vertices $\langle q,l\rangle$ and $\langle q',l'\rangle$ in G, if $\langle q',l'\rangle$ is reachable from $\langle q,l\rangle$ (in particular, if $\langle\langle q,l\rangle,\langle q',l'\rangle\rangle \in E$), then $f(q',l') \leq f(q,l)$. Assume that $f(q,l) = i$. We distinguish between two cases. If i is even, in which case $\langle q,l\rangle$ is finite in G_i, then either $\langle q',l'\rangle$ is not in G_i, in which case, by the above claim, its rank is at most $i-1$, or $\langle q',l'\rangle$ is in G_i, in which case, being reachable from $\langle q,l\rangle$, it must be finite in G_i and have rank i. If i is odd, in which case $\langle q,l\rangle$ is α-free in G_i, then either $\langle q',l'\rangle$ is not in G_i, in which case, by the above claim, its rank is at most $i-1$, or $\langle q',l'\rangle$ is in G_i, in which case, being reachable from $\langle q,l\rangle$, it must by α-free in G_i and have rank i.

It remains to be proved that f is an odd ranking. By the above, in every infinite path in G, there exists a vertex $\langle q,l\rangle$ such that all the vertices $\langle q',l'\rangle$ in the path that are reachable from $\langle q,l\rangle$ have $f(q',l') = f(q,l)$. We need to prove that the rank of $\langle q,l\rangle$ is odd. Assume, by way of contradiction, that the rank of $\langle q,l\rangle$ is some even i. Thus, $\langle q,l\rangle$ is finite in G_i. Then, the rank of all the vertices in the path that are reachable from $\langle q,l\rangle$ is also i, so they all belong to G_i. Since the path is infinite, there are infinitely many such vertices, contradicting the fact that $\langle q,l\rangle$ is finite in G_i. □

By Lemma 2, an NBW $\mathscr{A}'$ that complements $\mathscr{A}$ can proceed on an input word w by guessing an odd ranking for the run DAG of $\mathscr{A}$ on w. We now define such an NBW $\mathscr{A}'$ formally. We first need some definitions and notations.

A *level ranking* for $\mathscr{A}$ is a function $g : Q \to [2n] \cup \{\bot\}$, such that if $g(q)$ is odd, then $q \notin \alpha$. For two level rankings g and g', we say that g' *covers* g if for all q and q' in Q, if $g(q) \geq 0$ and $q' \in \delta(q,\sigma)$, then $0 \leq g'(q') \leq g(q)$. For a level ranking g, let $even(g)$ be the set of states that g maps to an even rank. Formally, $even(g) = \{q : g(q) \text{ is even}\}$.

Theorem 4. *Let $\mathscr{A}$ be an NBW with n states. There is an NBW $\mathscr{A}'$ such that $\mathscr{L}(\mathscr{A}') = \Sigma^\omega \setminus \mathscr{L}(\mathscr{A})$, and $\mathscr{A}'$ has at most $2^{O(n\log n)}$ states.*

Proof: Let $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha\rangle$. Let $\mathscr{R}$ be the set of all level rankings for $\mathscr{A}$. When $\mathscr{A}'$ runs on a word w, it guesses an odd ranking for the run DAG of $\mathscr{A}$ on w. Each state of $\mathscr{A}'$ is a pair $\langle g,P\rangle \in \mathscr{R} \times 2^Q$. The level ranking g maintains the states in the current level of the DAG (those that are not mapped to $\bot$) and the guessed rank for them. The set P is a subset of these states, used for ensuring that all paths visit odd ranks infinitely often, which, by the definition of odd rankings, implies that all paths get stuck in some odd rank.

Formally, $\mathscr{A}' = \langle \Sigma, \mathscr{R} \times 2^Q, Q'_0, \delta', \mathscr{R} \times \{\emptyset\}\rangle$, where

- $Q'_0 = \{\langle g_0, \emptyset\rangle\}$, where $g_0(q) = 2n$ for $q \in Q_0$, and $g_0(q) = \bot$ for $q \notin Q_0$. Thus, the odd ranking that $\mathscr{A}'$ guesses maps the vertices $\langle q,0\rangle$ of the run DAG to $2n$.
- For a state $\langle g,P\rangle \in \mathscr{R} \times 2^Q$ and a letter $\sigma \in \Sigma$, we define $\delta'(\langle g,P\rangle,\sigma)$ as follows.
  - If $P \neq \emptyset$, then $\delta'(\langle g,P\rangle,\sigma) = \{\langle g', \delta(P,\sigma) \cap even(g')\rangle : g' \text{ covers } g\}$.

– If $P = \emptyset$, then $\delta'(\langle g, P\rangle, \sigma) = \{\langle g', even(g')\rangle : g' \text{ covers } g\}$.

Thus, when $\mathscr{A}'$ reads the l-th letter in the input, for $l \geq 1$, it guesses the level ranking for level l in the run DAG. This level ranking should cover the level ranking of level $l-1$. In addition, in the P component, $\mathscr{A}'$ keeps track of states whose corresponding vertices in the DAG have even ranks. Paths that traverse such vertices should eventually reach a vertex with an odd rank. When all the paths of the DAG have visited a vertex with an odd rank, the set P becomes empty (a formal proof of the latter requires the use of König's Lemma, showing that if P does not become empty we can point to an infinite path that visits only even ranks). The set P is then initiated by new obligations for visits to vertices with odd ranks according to the current level ranking. The acceptance condition $\mathscr{R} \times \{\emptyset\}$ then checks that there are infinitely many levels in which all the obligations have been fulfilled.

Since there are $(2n+1)^n$ level rankings and 2^n subsets of Q, the automaton $\mathscr{A}'$ indeed has $2^{O(n \log n)}$ states. □

The blow-up of NBW complementation is thus $2^{O(n \log n)}$ and goes beyond the 2^n blow-up of the subset construction used in determinization and complementation of nondeterministic automata on finite words. As we see below, this blow-up cannot be avoided.

Theorem 5. [54] *There is a family of languages $L_1, L_2, \ldots$ such that $L_n \subseteq \Sigma_n^\omega$ can be recognized by an NBW with $n+1$ states but an NBW for $\Sigma_n^\omega \setminus L_n$ has at least $n!$ states.*

Proof: For $n \geq 1$, we define L_n as the language of the NBW $\mathscr{A}_n = \langle \Sigma_n, Q_n, Q_n^0, \delta_n, \alpha\rangle$, where (see Figure 4)

- $\Sigma_n = \{1, \ldots, n, \#\}$,
- $Q_n = \{q_0, q_1, \ldots, q_n\}$,
- $Q_n^0 = \{q_1, \ldots, q_n\}$,
- δ_n is defined as follows:

$$\delta_n(q_i, \sigma) = \begin{cases} \emptyset & \text{if } i = 0 \text{ and } \sigma = \#, \\ \{q_\sigma\} & \text{if } i = 0 \text{ and } \sigma \in \{1, \ldots, n\}, \\ \{q_i\} & \text{if } i \notin \{0, \sigma\}, \\ \{q_0, q_i\} & \text{if } \sigma = i. \end{cases}$$

- $\alpha = \{q_0\}$.

Note that a run of $\mathscr{A}_n$ is accepting if it contains infinitely many segments of the form $q_{i_1}^+ q_0 q_{i_2}^+ q_0 \cdots q_0 q_{i_k}^+ q_0 q_{i_1}^+$ for some distinct $i_1, \ldots, i_k \geq 1$. Accordingly, a word w is accepted by $\mathscr{A}_n$ iff there are k letters $\sigma_1, \sigma_2, \ldots, \sigma_k \in \{1, \ldots, n\}$, such that all the pairs $\sigma_1\sigma_2$, $\sigma_2\sigma_3$, …, $\sigma_k\sigma_1$ appear in w infinitely many times. A good intuition to keep in mind is that a word $w \in \Sigma_n^\omega$ induces a directed graph $G_w = \langle \{1, \ldots, n\}, E_w\rangle$ such that $E_w(i, j)$ iff the subword $i \cdot j$ appears in w infinitely often. Then, L_n accepts exactly all words w such that G_w contains a cycle.

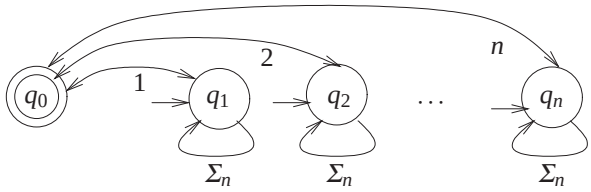

Fig. 4 The NBW $\mathscr{A}_n$

Consider an NBW $\mathscr{A}'_n = \langle \Sigma_n, Q'_n, Q'^0_n, \delta'_n, \alpha'_n \rangle$ that complements $\mathscr{A}_n$, and consider a permutation $\pi = \langle \sigma_1, \ldots, \sigma_n \rangle$ of $\{1, \ldots, n\}$. Note that the word $w_\pi = (\sigma_1 \cdots \sigma_n \cdot \#)^\omega$ is not in L_n. Thus, w_π is accepted by $\mathscr{A}'_n$. Let r_π be an accepting run of $\mathscr{A}'_n$ on w_π, and let $S_\pi \subseteq Q'_n$ be the set of states that are visited infinitely often in r_π. We prove that for every two different permutations π_1 and π_2 of $\{1, \ldots, n\}$, it must be that $S_{\pi_1} \cap S_{\pi_2} = \emptyset$. Since there are $n!$ different permutations, this implies that $\mathscr{A}'_n$ must have at least $n!$ states.

Assume by way of contradiction that π_1 and π_2 are such that $S_{\pi_1} \cap S_{\pi_2} \neq \emptyset$. Let $q \in Q'_n$ be a state in $S_{\pi_1} \cap S_{\pi_2}$. We define three finite words in Σ_n^*:

- a prefix h of w_{π_1} with which r_{π_1} moves from an initial state of $\mathscr{A}'_n$ to q,
- an infix u_1 of w_{π_1} that includes the permutation π_1 and with which r_{π_1} moves from q back to q and visits α'_n at least once when it does so, and
- an infix u_2 of w_{π_2} that includes the permutation π_2 and with which r_{π_2} moves from q back to q.

Note that since $\mathscr{A}'_n$ accepts w_{π_1} and w_{π_2}, the words h, u_1, and u_2 exist. In particular, since r_{π_1} is accepting and q is visited infinitely often in r_{π_1}, there is at least one (in fact, there are infinitely many) infix in r_{π_1} that leads from q to itself and visits α'_n.

Consider the word $w = h \cdot (u_1 \cdot u_2)^\omega$. We claim that $w \in L_n$ and $w \in \mathscr{L}(\mathscr{A}'_n)$, contradicting the fact that $\mathscr{A}'_n$ complements $\mathscr{A}_n$. We first point to an accepting run r of $\mathscr{A}'_n$ on w. The run r first follows r_{π_1} and gets to q while reading h. Then, the run r repeatedly follows the run r_{π_1} when it moves from q via α'_n back to q while reading u_1, and the run r_{π_2} when it moves from q back to q while reading u_2. It is easy to see that r is a run on w that visits α'_n infinitely often, thus $w \in \mathscr{L}(\mathscr{A}'_n)$.

Now, let $\pi_1 = \langle \sigma^1_1, \ldots, \sigma^1_n \rangle$ and $\pi_2 = \langle \sigma^2_1, \ldots, \sigma^2_n \rangle$, and let j be the minimal index for which $\sigma^1_j \neq \sigma^2_j$. There must exist $j < k, l \leq n$ such that $\sigma^1_j = \sigma^2_k$, and $\sigma^2_j = \sigma^1_l$. Since u_1 includes the permutation π_1 and u_2 includes the permutation π_2, the pairs $\sigma^1_j\sigma^1_{j+1}$, $\sigma^1_{j+1}\sigma^1_{j+2}, \ldots, \sigma^1_{l-1}\sigma^1_l$, $\sigma^1_l\sigma^2_{j+1} (= \sigma^2_j\sigma^2_{j+1})$, $\sigma^2_{j+1}\sigma^2_{j+2}, \ldots, \sigma^2_{k-1}\sigma^2_k$, $\sigma^2_k\sigma^1_{j+1} (= \sigma^1_j\sigma^1_{j+1})$ repeat infinitely often. Hence, $w \in L_n$ and we are done. □

Remark 1. Note that the alphabets of the languages L_n used in the proof of Theorem 5 depend on n. As shown in [50], it is possible to encode the languages and prove a $2^{\Omega(n \log n)}$ lower bound with a fixed alphabet.

We note that the upper and lower bounds here are based on classical and relatively simple constructions and proofs, but are still not tight. A tighter upper bound, based

on a restriction and a more precise counting of the required level rankings has been suggested in [18], and tightened further in [66]. An alternative approach, yielding a similar bound, is based on tracking levels of "split trees" – run trees in which only essential information about the history of each run is maintained [17, 30]. A tighter lower bound, based on the notion of full automata, is described in [80].

2.3 Determinization

Nondeterministic automata on finite words can be determinized by applying the subset construction [63]. Starting with a nondeterministic automaton $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$, the subset construction generates a deterministic automaton $\mathscr{A}'$ with state space 2^Q. The intuition is that the single run of $\mathscr{A}'$ is in state $S \in 2^Q$ after reading a word $w \in \Sigma^*$ iff S is the set of states that $\mathscr{A}$ could have been at, in one of its runs, after reading w. Accordingly, the single initial state of $\mathscr{A}'$ is the set Q_0, and the transition of $\mathscr{A}'$ from a state $S \in 2^Q$ and a letter $\sigma \in \Sigma$ is the set $\bigcup_{s \in S} \delta(s, \sigma)$. Since $\mathscr{A}'$ accepts exactly all words on which there is a run of $\mathscr{A}$ that ends in α, the set of accepting states of $\mathscr{A}'$ consists of these sets S such that $S \cap \alpha \neq \emptyset$. The exponential blow-up that the subset construction involves is justified by a matching lower bound.

It is not hard to see that the subset construction does not result in an equivalent automaton when applied to an NBW. For example, applying the subset construction to the NBW $\mathscr{A}_2$ from Example 2 results in the DBW $\mathscr{A}_2'$ in Figure 5. Recall that $\mathscr{A}_2$ recognizes the language of all words with finitely many a's. On the other hand, $\mathscr{A}_2'$ recognizes the language of all words with infinitely many b's. Thus, $\mathscr{L}(\mathscr{A}_2') \neq \mathscr{L}(\mathscr{A}_2)$. For example, the word $(a \cdot b)^\omega$ is in $\mathscr{L}(\mathscr{A}_2') \setminus \mathscr{L}(\mathscr{A}_2)$.

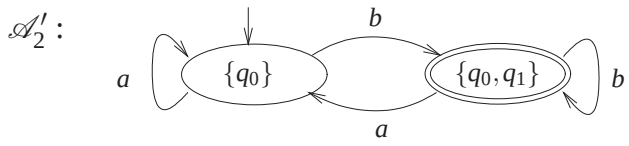

Fig. 5 The DBW obtained by applying the subset construction to $\mathscr{A}_2$

Note that not only $\mathscr{L}(\mathscr{A}_2') \neq \mathscr{L}(\mathscr{A}_2)$, there is no way to define a Büchi acceptance condition on top of the structure of $\mathscr{A}_2'$ and obtain a DBW that would be equivalent to $\mathscr{A}_2$. In fact, as we shall see now, there is no DBW that is equivalent to $\mathscr{A}_2$.

Theorem 6. [49] *There is a language L that is NBW-recognizable but not DBW-recognizable.*

Proof: Consider the language L described in Example 2. I.e., L is over the alphabet $\{a, b\}$ and it consists of all infinite words in which a occurs only finitely many times.

The language L is recognized by the NBW $\mathscr{A}_2$ appearing in Figure 2. We prove that L is not DBW-recognizable. Assume by way of contradiction that $\mathscr{A}$ is a DBW such that $\mathscr{L}(\mathscr{A}) = L$. Let $\mathscr{A} = \langle \{a,b\}, Q, q_0, \delta, \alpha \rangle$. Recall that δ can be viewed as a partial mapping from $Q \times \{a,b\}^*$ to Q.

Consider the infinite word $w_0 = b^\omega$. Clearly, w_0 is in L, so the run of $\mathscr{A}$ on w_0 is accepting. Thus, there is $i_1 \geq 0$ such that the prefix b^{i_1} of w_0 is such that $\delta(q_0, b^{i_1}) \in \alpha$. Consider now the infinite word $w_1 = b^{i_1} \cdot a \cdot b^\omega$. Clearly, w_1 is also in L, so the run of $\mathscr{A}$ on w_1 is accepting. Thus, there is $i_2 \geq 0$ such that the prefix $b^{i_1} \cdot a \cdot b^{i_2}$ of w_1 is such that $\delta(q_0, b^{i_1} \cdot a \cdot b^{i_2}) \in \alpha$. In a similar fashion we can continue to find indices $i_1, i_2, \ldots$ such that $\delta(q_0, b^{i_1} \cdot a \cdot b^{i_2} \cdot a \cdots a b^{i_j}) \in \alpha$ for all $j \geq 1$. Since Q is finite, there are iterations j and k, such that $1 \leq j < k \leq |\alpha| + 1$ and there is a state q such that $q = \delta(q_0, b^{i_1} \cdot a \cdot b^{i_2} \cdot a \cdots a \cdot b^{i_j}) = \delta(q_0, b^{i_1} \cdot a \cdot b^{i_2} \cdot a \cdots a \cdot b^{i_k})$. Since $j < k$, the extension $a b^{i_{j+1}} \cdots b^{i_{k-1}} \cdot a \cdot b^{i_k}$ is not empty and at least one state in α is visited when $\mathscr{A}$ loops in q while running through it. It follows that the run of $\mathscr{A}$ on the word

$$w = b^{i_1} \cdot a \cdot b^{i_2} \cdot a \cdots a b^{i_j} \cdot (a b^{i_{j+1}} \cdots b^{i_{k-1}} \cdot a \cdot b^{i_k})^\omega$$

is accepting. But w has infinitely many occurrences of a, so it is not in L, and we have reached a contradiction. □

Note that the complementary language $(a+b)^\omega \setminus L$, which is the language of infinite words in which a occurs infinitely often, is recognized by the DBW described in Example 1. It follows that DBWs are not closed under complementation.

A good way to understand why the subset construction does not work for determinization on NBWs is to note that the DBW $\mathscr{A}_2'$ discussed above accepts exactly all words that have infinitely many prefixes on which there is a run of $\mathscr{A}_2$ that reaches an accepting state. Since $\mathscr{A}_2$ is nondeterministic, the different runs need not extend each other, and thus they need not induce a single run of $\mathscr{A}_2$ that visits the accepting state infinitely often. In Section 3.2, we are going to return to this example and study NBW determinization in general. Here, we use the "extend each other" intuition for the following characterization of languages that are DBW-recognizable.

For a language $R \subseteq \Sigma^*$, let $lim(R) \subseteq \Sigma^\omega$ be the set of infinite words that have infinitely many prefixes in R. Formally, $lim(R) = \{w = \sigma_1 \cdot \sigma_2 \cdots : \sigma_1 \cdots \sigma_i \in R \text{ for infinitely many } i \geq 0\}$. Thus, lim is an operator that takes a language of finite words and turns it into a language of infinite words. For example, if R is the language of words ending with a, then $lim(R)$ is the language of words with infinitely many a's.

Theorem 7. [49] *A language $L \subseteq \Sigma^\omega$ is DBW-recognizable iff there is a regular language $R \subseteq \Sigma^*$ such that $L = lim(R)$.*

Proof: Assume first that L is DBW-recognizable. Let $\mathscr{A}$ be a DBW that recognizes L, let $\mathscr{A}_F$ be $\mathscr{A}$ when viewed as an automaton on finite words, and let $R = \mathscr{L}(\mathscr{A}_F)$. It is easy to see that since $\mathscr{A}$, and therefore also $\mathscr{A}_F$, are deterministic, we have that $\mathscr{L}(\mathscr{A}) = lim(R)$. Assume now that there is a regular language $R \subseteq \Sigma^*$ such

that $L = lim(R)$. Let $\mathscr{A}$ be a deterministic automaton on finite words that recognizes R, and let $\mathscr{A}_B$ be $\mathscr{A}$ when viewed as a DBW. Again, since $\mathscr{A}$ is deterministic, and thus runs on different prefixes of a word extend each other, it is easy to see that $\mathscr{L}(\mathscr{A}_B) = lim(\mathscr{L}(\mathscr{A}))$. Hence, L is DBW-recognizable. □

Note that a DBW-recognizable language may be the limit of several different regular languages. As we demonstrate in Theorem 8 below, this explains why, unlike the case of automata on finite words, a language may have different minimal DBWs. In fact, while minimization of automata on finite words can be done in polynomial time, the problem of DBW minimization in NP-complete [68].

Theorem 8. *A DBW-recognizable language L may not have a unique minimal DBW.*

Proof: Let $\Sigma = \{a,b\}$. Consider the language L of all words that contain infinitely many a's and infinitely many b's. It is not hard to prove that L cannot be recognized by a DBW with two states. Figure 6 describes two three-state, and thus minimal, DBWs for the language. In fact, each of the states in the automata may be the initial state, so the figure describes six such (non-isomorphic) automata.

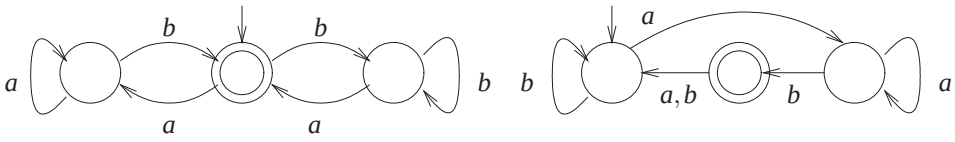

Fig. 6 Two minimal DBWs for L

□

Theorem 6 implies that we cannot hope to develop a determinization construction for NBWs. Suppose, however, that we have changed the definition of acceptance, and work with a definition in which a run is accepting iff it visits the set of accepting states only finitely often; i.e., $inf(r) \cap \alpha = \emptyset$. It is not hard to see that using such a definition, termed *co-Büchi*, we could have a deterministic automaton that recognizes the language L used in the proof of Theorem 6. In particular, the language is recognized by the deterministic automaton $\mathscr{A}_1$ from Figure 1 when we view it as a co-Büchi automaton. While the co-Büchi condition enables us to recognize the language L with a deterministic automaton, it is not expressive enough to recognize all languages that are recognizable by NBWs. In Section 3, we are going to introduce and study several acceptance conditions, and see how NBWs can be determinized using acceptance conditions that are stronger than the Büchi and co-Büchi conditions.

3 Additional Acceptance Conditions

The Büchi acceptance condition suggests one possible way to refer to $inf(r)$ for defining when a run r is accepting. The fact that DBWs are strictly less expressive than NBWs motivates the introduction of other acceptance conditions. In this section we review some acceptance conditions and discuss the expressive power and succinctness of the corresponding automata.

Consider an automaton with state space Q. We define the following acceptance conditions.

- *Co-Büchi*, where $\alpha \subseteq Q$, and a run r is accepting iff $inf(r) \cap \alpha = \emptyset$.
- *Generalized Büchi*, where $\alpha = \{\alpha_1, \ldots, \alpha_k\}$, with $\alpha_i \subseteq Q$, and a run r is accepting if $inf(r) \cap \alpha_i \neq \emptyset$ for all $1 \leq i \leq k$.
- *Rabin*, where $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$, with $\alpha_i, \beta_i \subseteq Q$, and a run r is accepting if for some $1 \leq i \leq k$, we have that $inf(r) \cap \alpha_i \neq \emptyset$ and $inf(r) \cap \beta_i = \emptyset$.
- *Streett*, where $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$, with $\alpha_i, \beta_i \subseteq Q$ and a run r is accepting if for all $1 \leq i \leq k$, we have that $inf(r) \cap \alpha_i = \emptyset$ or $inf(r) \cap \beta_i \neq \emptyset$.
- *Parity*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ with $\alpha_1 \subseteq \alpha_2 \subseteq \cdots \subseteq \alpha_k = Q$, and a run r is accepting if the minimal index i for which $inf(r) \cap \alpha_i \neq \emptyset$ is even.
- *Muller*, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$, with $\alpha_i \subseteq Q$ and a run r is accepting if for some $1 \leq i \leq k$, we have that $inf(r) = \alpha_i$.

The number of sets in the generalized Büchi, parity, and Muller acceptance conditions or pairs in the Rabin and Streett acceptance conditions is called the *index* of the automaton. We extend our NBW and DBW notations to the above classes of automata, and we use the letters C, R, S, P, and M to denote co-Büchi, Rabin, Streett, parity, and Muller automata, respectively. Thus, for example, DPW stands for deterministic parity automata. We sometimes talk about satisfaction of an acceptance condition α by a set S of states. As expected, S satisfies α iff a run r with $inf(r) = S$ is accepting. For example, a set S satisfies a Büchi condition α iff $S \cap \alpha \neq \emptyset$.

It is easy to see that the co-Büchi acceptance condition is dual to the Büchi acceptance condition in the sense that a run r is accepting with a Büchi condition α iff r is not accepting when α is viewed as a co-Büchi condition, and vice versa. This implies, for example, that for a deterministic automaton $\mathscr{A}$, we have that $\mathscr{L}(\mathscr{A}_B) = \Sigma^\omega \setminus \mathscr{L}(\mathscr{A}_C)$, where $\mathscr{A}_B$ and $\mathscr{A}_C$ are the automata obtained by viewing $\mathscr{A}$ as a Büchi and co-Büchi automaton, respectively. Similarly, the Rabin acceptance condition is dual to the Streett acceptance condition. Indeed, if $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$, then for every run r, there is no $1 \leq i \leq k$ such that $inf(r) \cap \alpha_i \neq \emptyset$ and $inf(r) \cap \beta_i = \emptyset$ iff for all $1 \leq i \leq k$, we have that $inf(r) \cap \alpha_i = \emptyset$ or $inf(r) \cap \beta_i \neq \emptyset$.

For two classes γ and κ of automata, we say that γ is *at least as expressive as* κ if for every κ-automaton $\mathscr{A}$, there is a γ-automaton $\mathscr{A}'$ such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$. If both γ is at least as expressive as κ and κ is at least as expressive as γ, then γ is *as expressive as* κ. One way to prove that γ is at least as expressive as κ is to show a translation of κ-automata to γ-automata. In the next section we are going to see such translations. As we shall see there, NBWs are as expressive

as NRWs, NSWs, NPWs, and NMWs. On the other hand, NCWs are strictly less expressive than NBWs. Also, as we shall see in Section 3.2, nondeterminism does not add expressive power in automata with the richer acceptance conditions. Thus, DRWs, DSWs, DPWs, and DMWs recognize all ω-regular languages, and are as expressive as NBWs. This is in contrast with the Büchi condition, where, as we have seen in Theorem 6, NBWs are strictly more expressive than DBWs. Finally, nondeterminism does not add expressive power also in co-Büchi automata, thus NCWs are as expressive as DCW, where both are weaker than NBW and coincide with the set of languages whose complement languages are NBW-recognizable (see Remark 3).

3.1 Translations Among the Different Classes

We distinguish between three types of translations among automata of the different classes: (1) Translations among the different conditions. This is the simplest case, where it is possible to translate the acceptance condition itself, regardless of the automaton on top of which the condition is defined. For example, a Büchi acceptance condition α is equivalent to the Rabin condition $\{\langle \alpha, \emptyset \rangle\}$. (2) Translations in which we still do not change the structure of the automaton, yet the definition of the acceptance condition may depend on its structure. Following the terminology of [35], we refer to such translations as *typed*. (3) Translations that manipulate the state space. This is the most general case, where the translation may involve a blow-up in the state space of the automaton. Accordingly, here we are interested also in the *succinctness* of the different classes, namely the worst-case bound on the blow-up when we translate. In this section we survey the three types.

3.1.1 Translations Among the Different Conditions

Some conditions are special cases of other conditions, making the translation among the corresponding automata straightforward. We list these cases below. Consider an automaton with state space Q.

- A Büchi condition α is equivalent to the Rabin condition $\{\langle \alpha, \emptyset \rangle\}$, the Streett condition $\{\langle Q, \alpha \rangle\}$, and the parity condition $\{\emptyset, \alpha, Q\}$.
- A co-Büchi condition α is equivalent to the Rabin condition $\{\langle Q, \alpha \rangle\}$, the Streett condition $\{\langle \alpha, \emptyset \rangle\}$, and the parity condition $\{\alpha, Q\}$.
- A generalized Büchi condition $\{\alpha_1, \ldots, \alpha_k\}$ is equivalent to the Streett condition $\{\{\langle Q, \alpha_1 \rangle, \langle Q, \alpha_2 \rangle, \ldots, \langle Q, \alpha_k \rangle\}\}$.
- A parity condition $\{\alpha_1, \ldots, \alpha_k\}$ (for simplicity, assume that k is even; otherwise, we can duplicate α_k) is equivalent to the Rabin condition $\{\langle \alpha_2, \alpha_1 \rangle, \langle \alpha_4, \alpha_3 \rangle, \ldots, \langle \alpha_k, \alpha_{k-1} \rangle\}$, and to the Streett condition $\{\langle \alpha_1, \emptyset \rangle, \langle \alpha_3, \alpha_2 \rangle, \ldots, \langle \alpha_{k-1}, \alpha_{k-2} \rangle\}$. (Recall that $\alpha_k = Q$, so there is no need to include the pair $\langle Q, \alpha_k \rangle$ in the Streett condition.)

- A Büchi, co-Büchi, Rabin, Streett, or parity acceptance condition α is equivalent to the Muller condition $\{F : F \text{ satisfies } \alpha\}$.

3.1.2 Typeness

In [35], the authors studied the expressive power of DBWs and introduced the notion of typeness for automata. For two classes γ and κ of automata, we say that γ is κ-type if for every γ-automaton $\mathscr{A}$, if $\mathscr{L}(\mathscr{A})$ is κ-recognizable, then it is possible to define a κ-automaton $\mathscr{A}'$ such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$ and $\mathscr{A}'$ differs from $\mathscr{A}$ only in the definition of the acceptance condition. Clearly, if an acceptance condition can be translated to another acceptance condition, as discussed in Section 3.1.1, then typeness for the corresponding classes follows. Interestingly, typeness may be valid also when γ is more expressive than κ. We demonstrate this below.

Theorem 9. [35] *DRWs are DBW-type.*

Proof: Consider a DRW $\mathscr{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. Let $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\}$. We say that a state $q \in Q$ is good in $\mathscr{A}$ if all the cycles $C \subseteq Q$ that contain q satisfy the acceptance condition α. Consider the DBW $\mathscr{A}' = \langle \Sigma, Q, q_0, \delta, \alpha' \rangle$, where $\alpha' = \{q : q \text{ is good in } \mathscr{A}\}$. We prove that if $\mathscr{A}$ is DBW-recognizable, then $\mathscr{L}(\mathscr{A}) = \mathscr{L}(\mathscr{A}')$. Hence, if $\mathscr{A}$ is DBW-recognizable, then there is a DBW equivalent to $\mathscr{A}$ that can be obtained by only changing the acceptance condition of $\mathscr{A}$.

We first prove that $\mathscr{L}(\mathscr{A}') \subseteq \mathscr{L}(\mathscr{A})$. In fact, this direction is independent of $\mathscr{A}$ being DBW-recognizable. Consider a word $w \in \mathscr{L}(\mathscr{A}')$. Let r be the accepting run of $\mathscr{A}'$ on w. Since r is accepting, there is a state $q \in \mathit{inf}(r) \cap \alpha'$. Recall that the states in $\mathit{inf}(r)$ constitute an SCS and thus also constitute a cycle that contains q. Therefore, as q is good, $\mathit{inf}(r)$ satisfies α, and r is also an accepting run of $\mathscr{A}$ on w. Hence, $w \in \mathscr{L}(\mathscr{A})$ and we are done.

We now prove that $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{A}')$. Consider a word $w \in \mathscr{L}(\mathscr{A})$. Let r be the accepting run of $\mathscr{A}$ on w. We prove that $\mathit{inf}(r) \cap \alpha' \neq \emptyset$. Assume by way of contradiction that $\mathit{inf}(r) \cap \alpha' = \emptyset$. Thus, no state in $\mathit{inf}(r)$ is good, so for each state $q \in \mathit{inf}(r)$, there is a cycle C_q that contains q and does not satisfy α. By [49], a deterministic automaton $\mathscr{A}$ recognizes a language that is in DBW iff for every strongly connected component C of $\mathscr{A}$, if C satisfies α, then all the strongly connected components C' with $C' \supseteq C$ satisfy α too. Consider the strongly connected component $C' = \bigcup_{q \in \mathit{inf}(r)} C_q$. Since C' contains $\mathit{inf}(r)$, and $\mathit{inf}(r)$ satisfies α, then, by the above, C' satisfies α too. Therefore, there is $1 \leq i \leq k$ such that $C' \cap \alpha_i \neq \emptyset$ and $C' \cap \beta_i = \emptyset$. Consider a state $s \in C' \cap \alpha_i$. Let q be such that $s \in C_q$. Observe that $C_q \cap \alpha_i \neq \emptyset$ and $C_q \cap \beta_i = \emptyset$, contradicting the fact that C_q does not satisfy α. □

Theorem 10. [35] *DSWs are not DBW-type.*

Proof: Consider the automaton $\mathscr{A}_1$ appearing in Figure 1, now with the Streett condition $\{\langle \{q_0, q_1\}, \{q_0\} \rangle, \langle \{q_0, q_1\}, \{q_1\} \rangle\}$. The language L of $\mathscr{A}_1$ then consists of exactly all words with infinitely many a's and infinitely many b's. As we have seen

in the proof of Theorem 8, L is DBW-recognizable. Yet, none of the four possibilities to define a DBW on top of the structure of $\mathscr{A}_1$ result in a DBW that recognizes L. □

Note that, by dualization, we get from Theorems 9 and 10 that DSWs are DCW-type and DRWs are not DCW-type.

The definition of typeness may be applied to nondeterministic automata too. As we show below, typeness need not coincide for nondeterministic and deterministic automata.

Theorem 11. [38] *DBWs are DCW-type, but NBWs are not NCW-type.*

Proof: The first claim follows from the fact that DBWs are a special case of DSWs, and the latter are DCW-type. For the second claim, consider the NBW $\mathscr{A}$ appearing in Figure 7. The NBW $\mathscr{A}$ has two initial states, in two disjoint components. Thus, the language of $\mathscr{A}$ is the union of the languages of the two NBWs associated with its two components. The NBW on the left accepts all words over the alphabet $\{a,b\}$ that satisfy "eventually a and infinitely many b's". The NBW on the right accepts all words that satisfy "eventually b and infinitely many a's". While each of these languages is not NCW-recognizable, their union recognizes the language L of all words satisfying "eventually a and eventually b", which is NCW-recognizable. It is not hard to see that none of the four possibilities to define a co-Büchi acceptance condition on top of $\mathscr{A}$ result in an NCW that recognizes L.

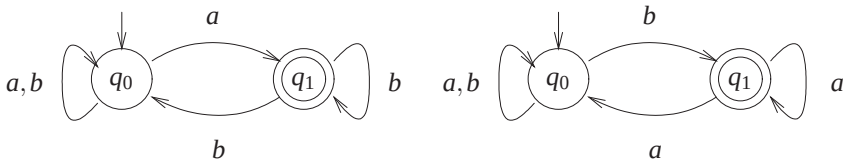

Fig. 7 An NBW that recognizes an NCW-recognizable language but has no equivalent NCW on the same structure

□

Researchers have considered additional variants of typeness. We mention two here. Let γ and κ be two acceptance conditions. In *powerset typeness*, we ask whether a deterministic κ-automaton can be defined on top of the subset construction of a nondeterministic γ-automaton. For example, NCWs are DBW-powerset-type: if the language of an NCW $\mathscr{A}$ is DBW-recognizable, then a DBW for $\mathscr{L}(\mathscr{A})$ can be defined on top of the subset construction of $\mathscr{A}$ [51]. In *combined typeness*, we ask whether the ability to define a certain language on top of the same automaton using two different acceptance conditions implies we can define it using a third, weaker, condition. For example, DRWs+DSWs are DPW-type: if a language L can be defined on top of a deterministic automaton $\mathscr{A}$ using both a Streett and a Rabin acceptance condition, then L can be defined on top of $\mathscr{A}$ also using a parity acceptance condition [4, 81]. For more results on typeness, see [38].

3.1.3 Translations That Require a New State Space

We now turn to the most general type of translations – those that may involve a blow-up in the state space. We do not specify all the translations, and rather describe the translation of nondeterministic generalized Büchi, Rabin, and Streett automata into NBWs. For the case of NSW, where the translation involves a blow-up that is exponential in the index, we also describe a lower bound.

Theorem 12. *Let $\mathscr{A}$ be a nondeterministic generalized Büchi automaton with n states and index k. There is an NBW $\mathscr{A}'$ with $n \cdot k$ states such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.*

Proof: Let $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \{\alpha_1, \ldots, \alpha_k\} \rangle$. The idea of the construction of $\mathscr{A}'$ is similar to the one used for defining the intersection of NBWs. Informally, $\mathscr{A}'$ consists of k copies of $\mathscr{A}$, and it stays in the i-th copy until it visits a state in α_i, in which case it moves to the next copy (modulo k). The acceptance condition of $\mathscr{A}'$ then makes sure that all copies are visited infinitely often.

Formally, $\mathscr{A}' = \langle \Sigma, Q', Q'_0, \delta', \alpha \rangle$, where

- $Q' = Q \times \{1, \ldots, k\}$.
- $Q'_0 = Q \times \{1\}$.
- For every $q \in Q$, $i \in \{1, \ldots, k\}$, and $\sigma \in \Sigma$, we have $\delta(\langle q, i \rangle, \sigma) = \delta(q, \sigma) \times \{j\}$, where $j = i$ if $q \notin \alpha_i$ and $j = (i \bmod k) + 1$ if $q \in \alpha_i$.
- $\alpha = \alpha_1 \times \{1\}$. Note that after a visit to α_1 in the first copy, the run moves to the second copy, where it waits for visits to α_2, and so on until it visits α_k in the k-th copy, and moves back to the first copy. Therefore, infinitely many visits in α_1 in the first copy indeed ensure that all copies, and thus also all α_i's are visited infinitely often.

□

Theorem 13. *Let $\mathscr{A}$ be an NRW with n states and index k. There is an NBW $\mathscr{A}'$ with at most $n(k+1)$ states such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.*

Proof: Let $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \{\langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\} \rangle$. It is easy to see that $\mathscr{L}(\mathscr{A}) = \bigcup_{i=1}^{k} \mathscr{L}(\mathscr{A}_i)$, where $\mathscr{A}_i = (\Sigma, Q, Q_0, \delta, \{\langle \alpha_i, \beta_i \rangle\})$. By Theorem 1, NBWs are closed under union. It therefore suffices to show a translation to NBWs of NRWs with index 1.

Consider an NRW $\mathscr{U} = \langle \Sigma, Q, Q_0, \delta, \{\langle \alpha, \beta \rangle\} \rangle$ with index 1. We translate $\mathscr{U}$ to an NBW $\mathscr{U}'$. The idea of the construction is similar to the one used for complementing DBWs: the NBW $\mathscr{U}'$ consists of two copies of $\mathscr{U}$, and it nondeterministically moves to the second copy, which contains only states that are not in β, and in which it has to visit infinitely many states in α. Formally, $\mathscr{U}' = \langle \Sigma, Q', Q'_0, \delta', \alpha' \rangle$, where

- $Q' = (Q \times \{0\}) \cup ((Q \setminus \beta) \times \{1\})$.
- $Q'_0 = Q_0 \times \{0\}$.
- For all $q \in Q$ and $\sigma \in \Sigma$, we have $\delta'(\langle q, 0 \rangle, \sigma) = (\delta(q, \sigma) \times \{0\}) \cup ((\delta(q, \sigma) \setminus \beta) \times \{1\})$, and $\delta'(\langle q, 1 \rangle, \sigma) = (\delta(q, \sigma) \setminus \beta) \times \{1\}$ for $q \in Q \setminus \beta$.
- $\alpha' = \alpha \times \{1\}$.

Since for an NRW $\mathscr{U}$ with n states, the NBW $\mathscr{U}'$ has at most $2n$ states, the union NBW has at most $2nk$ states. Now, in order to reduce the state space to $n(k+1)$, we observe that the first copy of $\mathscr{U}'$ can be shared by all $\mathscr{A}_i$'s. Thus, $\mathscr{A}'$ guesses both the pair α_i, β_i with which the acceptance condition is satisfied and the point from which states from β_i are no longer visited. Formally, we define $\mathscr{A}' = \langle \Sigma, Q', Q_0 \times \{0\}, \delta', \alpha' \rangle$, where

- $Q' = (Q \times \{0\}) \cup \bigcup_{1 \leq i \leq k} ((Q \setminus \beta_i) \times \{i\})$.
- For all $q \in Q$ and $\sigma \in \Sigma$, we have $\delta'(\langle q, 0 \rangle, \sigma) = (\delta(q, \sigma) \times \{0\}) \cup \bigcup_{1 \leq i \leq k} ((\delta(q, \sigma) \setminus \beta_i) \times \{i\})$, and $\delta'(\langle q, i \rangle, \sigma) = (\delta(q, \sigma) \setminus \beta_i) \times \{i\}$ for $1 \leq i \leq k$ and $q \in Q \setminus \beta_i$.
- $\alpha' = \bigcup_{1 \leq i \leq k} \alpha_i \times \{i\}$.

□

Translating NRWs to NBWs, we took the union of the NRWs of index 1 that are obtained by decomposing the acceptance condition. For NSW, it is tempting to proceed dually, and define the NBW as the intersection of the NSWs of index 1 that are obtained by decomposing the acceptance condition. Such an intersection, however, may accept words that are not in the language of the NSW. To see this, consider an automaton $\mathscr{A}$, a Streett acceptance condition $\alpha = \{\langle \alpha_1, \beta_1 \rangle, \langle \alpha_2, \beta_2 \rangle\}$, and a word w. It may be that there is a run r_1 of $\mathscr{A}$ on w that satisfies the Streett acceptance condition $\{\langle \alpha_1, \beta_1 \rangle\}$ and also a run r_2 of $\mathscr{A}$ on w that satisfies the Streett acceptance condition $\{\langle \alpha_2, \beta_2 \rangle\}$. Yet, the runs r_1 and r_2 may be different, and there need not be a run of $\mathscr{A}$ on w that satisfies both $\{\langle \alpha_1, \beta_1 \rangle\}$ and $\{\langle \alpha_2, \beta_2 \rangle\}$. Consequently, the translation of NSWs to NBWs has to consider the relation among the different pairs in α, giving rise to a blow-up that is exponential in k. Formally, we have the following.

Theorem 14. *Let $\mathscr{A}$ be an NSW with n states and index k. There is an NBW $\mathscr{A}'$ with at most $n(1+k2^k)$ states such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.*

Proof: Let $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \{\langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_k, \beta_k \rangle\} \rangle$. Recall that in an accepting run r of $\mathscr{A}$, we have that $inf(r) \cap \alpha_i = \emptyset$ or $inf(r) \cap \beta_i \neq \emptyset$ for all $1 \leq i \leq k$. For $I \subseteq \{1, \ldots, k\}$, we define an NBW $\mathscr{A}_I$ that accepts exactly all words w such that there is a run r of $\mathscr{A}$ on w for which $inf(r) \cap \alpha_i = \emptyset$ for all $i \in I$ and $inf(r) \cap \beta_i \neq \emptyset$ for all $i \notin I$. Thus, I indicates how the acceptance condition α is satisfied. It is easy to see that $\mathscr{L}(\mathscr{A}) = \bigcup_{I \subseteq \{1,\ldots,k\}} \mathscr{L}(\mathscr{A}_I)$.

The idea behind the construction of $\mathscr{A}_I$ is similar to the "two copies" idea we have seen above, except that now, in the copy in which $\mathscr{A}_I$ avoids the states in α_i, for all $i \in I$, it also has to visit all the states in β_i, for $i \notin I$. This can be easily achieved by first defining $\mathscr{A}_I$ as a nondeterministic generalized Büchi automaton. Formally, we define $\mathscr{A}_I = \langle \Sigma, Q_I, Q_0', \delta_I, \beta_I \rangle$ as follows. Let $\alpha_I = \bigcup_{i \in I} \alpha_i$. Then,

- $Q_I = (Q \times \{0\}) \cup ((Q \setminus \alpha_I) \times \{1\})$.
- $Q_0' = Q_0 \times \{0\}$.
- For every $q \in Q$ and $\sigma \in \Sigma$, we have $\delta_I(\langle q, 0 \rangle, \sigma) = (\delta(q, \sigma) \times \{0\}) \cup ((\delta(q, \sigma) \setminus \alpha_I) \times \{1\})$. For $q \in Q \setminus \alpha_I$, we also have $\delta_I(\langle q, 1 \rangle, \sigma) = (\delta(q, \sigma) \setminus \alpha_I) \times \{1\}$.

- $\beta_I = \{\beta_i \times \{1\} : i \notin I\}$.

Since $\mathscr{A}_I$ has at most $2n$ states and index k, an equivalent Büchi automaton has at most $2nk$ states. A slightly more careful analysis observes that the generalized Büchi condition applies only to the second copy of $\mathscr{A}_I$, thus a translation to NBW results in an automaton with at most $n+nk$ states. The automaton $\mathscr{A}'$ is then the union of all the 2^k NBWs obtained from the different $\mathscr{A}_I$ and thus has at most $(n+nk)2^k$ states. Moreover, as in the proof of Theorem 13, the first copy of all the NBWs in the union can be shared, tightening the bound further to $n+nk2^k$. □

In Theorem 15 below we show that the exponential blow-up in the translation of NSWs to NBWs cannot be avoided. In fact, as the theorem shows, the blow-up may occur even when one starts with a DSW.

Theorem 15. [65] *There is a family of languages $L_1, L_2, \ldots$ such that L_n can be recognized by a DSW with $3n$ states and index $2n$, but an NBW for L_n has at least 2^n states.*

Proof: Let $\Sigma = \{0,1,2\}$. We can view an infinite word over Σ as a word $w \in (\Sigma^n)^\omega$, thus $w = u_1 \cdot u_2 \cdot u_3 \cdots$, where each u_j is a word in Σ^n. We refer to such words as *blocks*. We say that index $i \in \{0,\ldots,n-1\}$ is 0-active in w iff there are infinitely many j's such that the i-th letter in u_j is 0. Similarly, i is 1-active in w iff there are infinitely many j's such that the i-th letter in u_j is 1. For $n \geq 1$, let

$$L_n = \{w : \text{ for all } 0 \leq i \leq n-1, \text{ the index } i \text{ is 0-active in } w \text{ iff } i \text{ is 1-active in } w\}.$$

We first describe a DSW $\mathscr{A}_n$ with $3n$ states such that $\mathscr{L}(\mathscr{A}_n) = L_n$. We define $\mathscr{A}_n = \langle \{0,1,2\}, Q_n, \{\langle 0,0\rangle\}, \delta_n, \alpha_n\rangle$, where

- $Q_n = \{1,\ldots,n\} \times \{0,1,2\}$. Intuitively, $\mathscr{A}_n$ moves to the state $\langle i,\sigma\rangle$ after it reads the $(i-1)$-th letter in the current block, and this letter is σ. Accordingly, an index $0 \leq i \leq n-1$ is σ-active in w iff the run of $\mathscr{A}_n$ on w visits states in $\{i+1\} \times \{\sigma\}$ infinitely often.
- For all $0 \leq i \leq n-1$ and $\sigma, \sigma' \in \{0,1,2\}$, we have $\delta_n(\langle i,\sigma\rangle, \sigma') = \langle (i+1) \bmod n, \sigma'\rangle$.
- $\alpha_n = \bigcup_{1 \leq i \leq n} \{\langle \{\langle i,0\rangle\}, \{\langle i,1\rangle\}\rangle, \langle \{\langle i,1\rangle\}, \{\langle i,0\rangle\}\rangle\}$.

It is easy to see that $\mathscr{A}_n$ has $3n$ states and that $\mathscr{L}(\mathscr{A}_n) = L_n$. Now, assume by way of contradiction that there is an NBW $\mathscr{A}'_n$ that recognizes L_n and has fewer than 2^n states. We say that a position in a word or in a run of $\mathscr{A}'_n$ is *relevant* if it is 0 mod n. That is, $\mathscr{A}'_n$ starts to read each block in a relevant position. For a set $S \subseteq \{0,\ldots,n-1\}$, let $w^0_S \in \{0,2\}^n$ be the word of length n in which for all $0 \leq i \leq n-1$, the i-th letter is 0 iff $i \in S$, and is 2 otherwise. Similarly, let $w^1_S \in \{1,2\}^n$ be the word in which the i-th letter is 1 iff $i \in S$, and is 2 otherwise. Note that if w^0_S appears in a word w in infinitely many relevant positions, then all the indices in S are 0-active, and similarly for w^1_S and 1-active. Consider the infinite word $w_S = ((w^0_S)^{2^n} \cdot w^1_S)^\omega$. Clearly, for index $i \in \{0,\ldots,n-1\}$, we have that i is 0-active in w_S iff i is 1-active in w_S iff $i \in S$. Hence, $w_S \in L_n$. Let r_S be an accepting run of $\mathscr{A}'_n$ on w_S. We say that a position $p \geq 0$ in r_S is *important* if it is relevant and there is a state q and a

position $p' > p$ such that q is visited in both positions p and p' and the subword read between them is in $(w_S^0)^*$. We then say that q *supports* p. Let Q_S be the set of states that support infinitely many important positions. Since Q is finite and there are infinitely many relevant positions, the set Q_S is not empty. Since $\mathscr{A}'_n$ has fewer than 2^n states, there must be two subsets S and T, such that $T \neq S$ and $Q_S \cap Q_T \neq \emptyset$. Let S and T be two such subsets. Assume without loss of generality that $T \setminus S \neq \emptyset$, and let q be a state in $Q_S \cap Q_T$. By the definition of Q_T, there is $i \geq 1$ such that $\mathscr{A}'_n$ can move from q back to itself when it reads $(w_T^0)^i$. We claim that we can then obtain from w_S a word w'_S that is not in L_n and is accepted by $\mathscr{A}'_n$. We obtain w'_S by inserting the word $(w_T^0)^i$ inside the $(w_S^0)^{2^n}$ subwords whenever the run of $\mathscr{A}'_n$ reaches the state q in important positions. The accepting run of $\mathscr{A}'_n$ is then similar to r_S, except that we pump visits to q in important positions to traverse the cycle along which $(w_T^0)^i$ is read. Since $\mathscr{A}'_n$ is a Büchi automaton, the run stays accepting, whereas the word it reads has indices (those in $T \setminus S$) that are 0-active but not 1-active, and is therefore not in L'_n. □

3.2 Determinization of NBWs

Recall that NBWs are strictly more expressive than DBWs. In this section we describe the intuition behind a determinization construction that translates a given NBW to an equivalent DPW. Detailed description of the construction can be found in [64, 59, 67]. As in the case of NBW complementation, efforts to determinize NBWs started in the 1960s, and involve several landmarks. In [52], McNaughton proved that NBWs can be determinized and described a doubly-exponential translation of NBWs to DMWs. Only in 1988, Safra improved the bound and described an optimal translation of NBWs to DRWs: given an NBW with n states, the equivalent DRW has $2^{O(n \log n)}$ states and index n. A different construction, with similar bounds, was given in [58]. The same considerations that hold for NBW complementation can be used in order to show a matching $2^{\Omega(n \log n)}$ lower bound [54, 50]. While Safra's determinization construction is asymptotically optimal, efforts to improve it have continued, aiming at reducing the state space and generating automata with the parity acceptance condition. Indeed, the parity acceptance condition has important advantages: it is easy to complement, and when used as a winning condition in a two-player game, both players can proceed according to memoryless strategies. Also, solving parity games is easier than solving Rabin games [12, 29] (see Chapter 25). In [59], Piterman described a direct translation of NBW to DPW, which also reduces the state blow-up in Safra's determinization. Piterman's construction has been further tightened in [67]. The translation is a variant of Safra's determinization construction, and we present the intuition behind it here. It is important to note that in addition to efforts to improve Safra's determinization construction, there have been efforts to develop algorithms that avoid determinization in constructions and methodologies that traditionally involve determinization; e.g., complementation of NBW [44], LTL synthesis [45], and more [36].

Before we describe the intuition behind the determinization construction, let us understand why NBW determinization is a difficult problem. Consider the NBW $\mathscr{A}_2$ from Example 2. In Figure 5 we described the DBW $\mathscr{A}_2'$ obtained by applying the subset construction to $\mathscr{A}_2$. While $\mathscr{A}_2$ recognizes the language of all words with finitely many a's, the DBW $\mathscr{A}_2'$ recognizes the language of all words with infinitely many b's. Why does the subset construction work for finite words and fail here? Consider the word $w = (b \cdot a)^\omega$. The fact the run of $\mathscr{A}_2'$ on w visits the state $\{q_0, q_1\}$ infinitely often implies that there are infinitely many prefixes of w such that $\mathscr{A}_2$ has a run on the prefix that ends in q_1. Nothing, however, is guaranteed about our ability to compose the runs on these prefixes into a single run. In particular, in the case of w, the run on each of the prefixes visits q_1 only once, as the destination of its last transition, and there is no way to continue and read the suffix of w from q_1.

Consider an NBW $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$ and an input word $w = \sigma_1 \cdot \sigma_2 \cdot \sigma_3 \cdots$. As the example above demonstrates, an equivalent deterministic automaton should not only make sure that w has infinitely many prefixes on which $\mathscr{A}$ can reach α, but also that $\mathscr{A}$ does so with runs that can be composed into a single run. Let $S_0, S_1, S_2, \ldots \in (2^Q)^\omega$ be the result of applying the subset construction of $\mathscr{A}$ on w. That is, S_i is the set of states that $\mathscr{A}$ can be at after reading $\sigma_1 \cdot \sigma_2 \cdots \sigma_i$. The deterministic automaton that is equivalent to $\mathscr{A}$ tries to find a sequence $\tau = T_0, T_1, T_2, \ldots \in (2^Q)^\omega$ such that $T_0 \subseteq S_0$ and for all $i \geq 0$, we have that $T_{i+1} \subseteq \delta(T_i, \sigma_{i+1})$. In addition, there are infinitely many positions $j_1, j_2, j_3, \ldots$ such that for all $k \geq 1$, each of the states in $T_{j_{k+1}}$ is reachable from some state in T_{j_k} via a run that visits α. We refer to τ as a *witness sequence* and refer to the positions $j_1, j_2, j_3, \ldots$ as *break-points*. Note that for all $i \geq 0$ we have $T_i \subseteq S_i$, and that indeed $\mathscr{A}$ accepts w iff such a witness sequence exists. First, if $\mathscr{A}$ accepts w with a run $q_0, q_1, \ldots$, then we can take $T_i = \{q_i\}$. Also, if τ exists, then we can generate an accepting run of $\mathscr{A}$ on w by reaching some state in T_{j_1}, then reaching, via α, some state in T_{j_2}, then reaching, via α, some state in T_{j_3}, and so on. The big challenge in the determinization construction is to detect a witness sequence without guessing.

One naive way to detect a witness sequence is to maintain full information about the runs of $\mathscr{A}$ on w. In Section 2.2.2, we defined the run DAG G that embodies all the possible runs of $\mathscr{A}$ on w. The prefix of G up to level i clearly contains all the information one needs about S_i and the history of all the states in it. The prefixes of G, however, are of increasing and unbounded sizes. A key point in the determinization construction is to extract from each prefix of G a finite presentation that is sufficiently informative. For the case of finite words, this is easy – the set of states in the last level of the prefix (that is, S_i) is sufficient. For the case of infinite words, the presentation is much more complicated, and is based on the data structure of *history trees*.

Essentially, the history tree that is reached after reading a prefix of length i of w maintains subsets of S_i that may serve as T_i. One challenge is to maintain these subsets in a compact way. A second challenge is to use the parity acceptance condition in order to guarantee that one of the maintained subsets can indeed serve as T_i in a witness sequence. The first challenge is addressed by arranging all candidate subsets in a tree in which each state in S_i is associated with at most one node of the tree.

This bounds the number of history trees by $n^{O(n)}$. The second challenge is addressed by updating the history trees in each transition in a way that relates the choice of the subset that would serve as T_i with the choice of the even index that witnesses the satisfaction of the parity condition: the subsets are ordered, essentially, according to their seniority – the point at which the deterministic automaton started to take them into account as a possible T_i. In each update, each subset may be declared as "stable", meaning that it continues to serve as a possible T_i, and may also be declared as "accepting", meaning that the position i is a break-point in the witness sequence in which T_i is a member. The parity acceptance condition then uses labels of seniority in order to look for a subset that is eventually always stable and infinitely often accepting. The above is only a high-level intuition, and in particular it misses the way in which the subsets are ordered and how the updates interfere with this order. As pointed out above, details can be found in the original papers [64, 59, 67].

4 Decision Procedures

Automata define languages, which are sets of words. Natural questions to ask about sets are whether they are trivial (that is, empty or universal), and whether two sets contain each other. Note that equivalence between two sets amounts to containment in both directions. In this section we study the following three problems, which address the above questions for languages defined by automata.

- The *non-emptiness* problem is to decide, given an automaton $\mathscr{A}$, whether $\mathscr{L}(\mathscr{A}) \neq \emptyset$.
- The *non-universality* problem is to decide, given an automaton $\mathscr{A}$, whether $\mathscr{L}(\mathscr{A}) \neq \Sigma^\omega$.
- The *language-containment* problem is to decide, given automata $\mathscr{A}_1$ and $\mathscr{A}_2$, whether $\mathscr{L}(\mathscr{A}_1) \subseteq \mathscr{L}(\mathscr{A}_2)$.

It is not hard to see that the non-emptiness and non-universality problems are dual, in the sense that an automaton is non-empty iff its complement is non-universal, and that both can be viewed as a special case of the language-containment problem. Indeed, if $\mathscr{A}_\bot$ and $\mathscr{A}_\top$ are such that $\mathscr{L}(\mathscr{A}_\bot) = \emptyset$ and $\mathscr{L}(\mathscr{A}_\top) = \Sigma^\omega$, then an automaton $\mathscr{A}$ is empty if $\mathscr{L}(\mathscr{A}) \subseteq \mathscr{L}(\mathscr{A}_\bot)$ and is universal if $\mathscr{L}(\mathscr{A}_\top) \subseteq \mathscr{L}(\mathscr{A})$. As we shall see below, however, the non-emptiness problem is easier than the other two. We note that the hardness results and proofs described in this section hold already for automata on finite words. We still present direct proofs for NBWs. An alternative would be to carry out a reduction from the setting of finite words.

Theorem 16. [14, 15, 77] *The non-emptiness problem for NBWs is decidable in linear time and is NLOGSPACE-complete.*

Proof: Consider an NBW $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. Recall that $\mathscr{A}$ induces a directed graph $G_\mathscr{A} = \langle Q, E \rangle$ where $\langle q, q' \rangle \in E$ iff there is a letter σ such that $q' \in \delta(q, \sigma)$. We claim that $\mathscr{L}(\mathscr{A})$ is non-empty iff there are states $q_0 \in Q_0$ and $q_{acc} \in \alpha$ such that

$G_{\mathscr{A}}$ contains a path leading from q_0 to q_{acc} and a cycle going though q_{acc}. Assume first that $\mathscr{L}(\mathscr{A})$ is non-empty. Then, there is an accepting run $r = q_0, q_1, \ldots$ of $\mathscr{A}$ on some input word, which corresponds to an infinite path of $G_{\mathscr{A}}$. Since r is accepting, some state $q_{acc} \in \alpha$ occurs in r infinitely often; in particular, there are i, j, where $0 \leq i < j$, such that $q_{acc} = q_i = q_j$. Thus, $q_0, \ldots, q_i$ corresponds to a (possibly empty) path from q_0 to q_{acc}, and $q_i, \ldots, q_j$ to a cycle going through q_{acc}.

Conversely, assume that $G_{\mathscr{A}}$ contains a path leading from q_0 to a state $q_{acc} \in \alpha$ and a cycle going though q_{acc}. We can then construct an infinite path of $G_{\mathscr{A}}$ starting at q_0 and visiting q_{acc} infinitely often. This path induces a run on a word accepted by $\mathscr{A}$.

Thus, NBW non-emptiness is reducible to graph reachability. The algorithm that proves membership in NLOGSPACE first guesses states $q_0 \in Q_0$ and $q_{acc} \in \alpha$, and then checks the reachability requirements by guessing a path from q_0 to q_{acc} and a path from q_{acc} to itself. Guessing these paths is done by remembering the current state on the path and the value of a counter for the length of the path traversed so far, and proceeding to a successor state while increasing the counter. When the counter value exceeds $|Q|$, the algorithm returns "no" (that is, the guess is not good). Note that the algorithm has to remember q_0, q_{acc}, the current state and counter value, each requiring logarithmic space.

NLOGSPACE-hardness can be proved by an easy reduction from the reachability problem in directed graphs [28]. There, one is given a directed graph $G = \langle V, E \rangle$ along with two vertices s and t, and the goal is to decide whether there is a path from s to t. It is easy to see that such a path exists iff the NBW $\mathscr{A}_G = \langle \{a\}, V, \{s\}, \delta, \{t\} \rangle$ with $v' \in \delta(v,a)$ iff $E(v,v')$ or $v' = v = t$ is not empty.

To check non-emptiness in linear time, we first find the decomposition of $G_{\mathscr{A}}$ into SCCs [10, 73]. An SCC is nontrivial if it contains an edge, which means, since it is strongly connected, that it contains a cycle. It is not hard to see that $\mathscr{A}$ is non-empty iff from an SCC whose intersection with Q_0 is not empty it is possible to reach a nontrivial SCC whose intersection with α is not empty. □

Theorem 17. [70] *The non-universality problem for NBWs is decidable in exponential time and is PSPACE-complete.*

Proof: Consider an NBW $\mathscr{A}$. Clearly, $\mathscr{L}(\mathscr{A}) \neq \Sigma^\omega$ iff $\Sigma^\omega \setminus \mathscr{L}(\mathscr{A}) \neq \emptyset$, which holds iff $\mathscr{L}(\mathscr{A}') \neq \emptyset$, where $\mathscr{A}'$ is an NBW that complements $\mathscr{A}$. Thus, to test $\mathscr{A}$ for non-universality, it suffices to test $\mathscr{A}'$ for non-emptiness. The construction of $\mathscr{A}'$ can proceed "on-the-fly" (that is, there is no need to construct and store $\mathscr{A}'$ and then perform the non-emptiness test, but rather it is possible to construct only the components required for the non-emptiness test on demand; such a construction requires only polynomial space). Hence, as $\mathscr{A}'$ is exponentially bigger than $\mathscr{A}$, the time and space bounds from Theorem 16 imply the two upper bounds.

For the lower bound, we do a reduction from polynomial-space Turing machines. The reduction does not use the fact that Büchi automata run on infinite words and follows the same considerations as the reduction showing that the non-universality problem is PSPACE-hard for nondeterministic automata on finite words [53]. Note

that we could also have reduced from this latter problem, but preferred to give the details of the generic reduction.

Given a Turing machine T of space complexity $s(n)$, we construct an NBW $\mathscr{A}_T$ of size linear in T and $s(n)$ such that $\mathscr{A}_T$ is universal iff T does not accept the empty tape. We assume, without loss of generality, that all the computations of T eventually reach a final state. Also, once T reaches a final state it loops there forever. The NBW $\mathscr{A}_T$ accepts a word w iff w is not an encoding of a legal computation of T over the empty tape or if w is an encoding of a legal yet rejecting computation of T over the empty tape. Thus, $\mathscr{A}_T$ rejects a word w iff w encodes a legal and accepting computation of T over the empty tape. Hence, $\mathscr{A}_T$ is universal iff T does not accept the empty tape.

We now give the details of the construction of $\mathscr{A}_T$. Let $T = \langle \Gamma, Q, \rightarrow, q_0, q_{acc}, q_{req} \rangle$, where Γ is the alphabet, Q is the set of states, $\rightarrow \subseteq Q \times \Gamma \times Q \times \Gamma \times \{L,R\}$ is the transition relation (we use $(q,a) \rightarrow (q',b,\Delta)$ to indicate that when T is in state q and it reads the input a in the current tape cell, it moves to state q', writes b in the current tape cell, and its reading head moves one cell to the left/right, according to Δ), and q_0, q_{acc}, and q_{rej} are the initial, accepting, and rejecting states.

We encode a configuration of T by a word $\#\gamma_1\gamma_2 \ldots (q,\gamma_i) \ldots \gamma_{s(n)}$. That is, a configuration starts with #, and all its other letters are in Γ, except for one letter in $Q \times \Gamma$. The meaning of such a configuration is that the j-th cell in T, for $1 \leq j \leq s(n)$, is labeled γ_j, the reading head points at cell i, and T is in state q. For example, the initial configuration of T is $\#(q_0,b)b \ldots b$ (with $s(n)-1$ occurrences of b) where b stands for an empty cell. We can now encode a computation of T by a sequence of configurations.

Let $\Sigma = \{\#\} \cup \Gamma \cup (Q \times \Gamma)$ and let $\#\sigma_1 \ldots \sigma_{s(n)} \# \sigma'_1 \ldots \sigma'_{s(n)}$ be two successive configurations of T. We also set σ_0, σ'_0, and $\sigma_{s(n)+1}$ to #. For each triple $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle$ with $1 \leq i \leq s(n)$, we know, by the transition relation of T, what σ'_i should be. In addition, the letter # should repeat exactly every $s(n)+1$ letters. Let $next(\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle)$ denote our expectation for σ'_i. That is,

- $next(\langle \gamma_{i-1}, \gamma_i, \gamma_{i+1} \rangle) = next(\langle \#, \gamma_i, \gamma_{i+1} \rangle) = next(\langle \gamma_{i-1}, \gamma_i, \# \rangle) = \gamma_i$.
- $next(\langle (q,\gamma_{i-1}), \gamma_i, \gamma_{i+1} \rangle) = next(\langle (q,\gamma_{i-1}), \gamma_i, \# \rangle) =$
$$\begin{cases} \gamma_i & \text{if } (q,\gamma_{i-1}) \rightarrow (q',\gamma'_{i-1},L) \\ (q',\gamma_i) & \text{if } (q,\gamma_{i-1}) \rightarrow (q',\gamma'_{i-1},R) \end{cases}$$
- $next(\langle \gamma_{i-1}, (q,\gamma_i), \gamma_{i+1} \rangle) = next(\langle \#, (q,\gamma_i), \gamma_{i+1} \rangle) =$
$next(\langle \gamma_{i-1}, (q,\gamma_i), \# \rangle) = \gamma'_i$ where $(q,\gamma_i) \rightarrow (q',\gamma'_i,\Delta)$. [3]
- $next(\langle \gamma_{i-1}, \gamma_i, (q,\gamma_{i+1}) \rangle) = next(\langle \#, \gamma_i, (q,\gamma_{i+1}) \rangle) =$
$$\begin{cases} \gamma_i & \text{if } (q,\gamma_{i+1}) \rightarrow (q',\gamma'_{i+1},R) \\ (q',\gamma_i) & \text{if } (q,\gamma_{i+1}) \rightarrow (q',\gamma'_i,L) \end{cases}$$
- $next(\langle \sigma_{s(n)}, \#, \sigma'_1 \rangle) = \#$.

[3] We assume that the reading head of T does not "fall" from the right or the left boundaries of the tape. Thus, the case where $(i=1)$ and $(q,\gamma_i) \rightarrow (q',\gamma'_i,L)$ and the dual case where $(i=s(n))$ and $(q,\gamma_i) \rightarrow (q',\gamma'_i,R)$ are not possible.

Consistency with *next* now gives us a necessary condition for a trace to encode a legal computation. In addition, the computation should start with the initial configuration.

In order to check consistency with *next*, the NBW $\mathscr{A}_T$ can use its nondeterminism and guess when there is a violation of *next*. Thus, $\mathscr{A}_T$ guesses $\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle \in \Sigma^3$, guesses a position in the trace, checks whether the three letters to be read starting in this position are σ_{i-1}, σ_i, and σ_{i+1}, and checks whether $next(\langle \sigma_{i-1}, \sigma_i, \sigma_{i+1} \rangle)$ is not the letter to come $s(n)+1$ letters later. Once $\mathscr{A}_T$ sees such a violation, it goes to an accepting sink. In order to check that the first configuration is not the initial configuration, $\mathscr{A}_T$ simply compares the first $s(n)+1$ letters with $\#(q_0, b)b \ldots b$. Finally, checking whether a legal computation is rejecting is also easy: the computation should reach a configuration in which T visits q_{rej}. □

Theorem 18. [70] *The containment problem for NBWs is decidable in exponential time and is PSPACE-complete.*

Proof: Consider NBWs $\mathscr{A}_1$ and $\mathscr{A}_2$. Note that $\mathscr{L}(\mathscr{A}_1) \subseteq \mathscr{L}(\mathscr{A}_2)$ iff $\mathscr{L}(\mathscr{A}_1) \cap (\Sigma^\omega \setminus \mathscr{L}(\mathscr{A}_2)) = \emptyset$, which holds iff $\mathscr{L}(\mathscr{A}') = \emptyset$, where $\mathscr{A}'$ is an NBW for the intersection of $\mathscr{A}_1$ with an NBW that complements $\mathscr{A}_2$. Thus, to check the containment of $\mathscr{A}_1$ in $\mathscr{A}_2$ we can test $\mathscr{A}'$ for emptiness. Since the construction of $\mathscr{A}'$ can proceed on-the-fly and its size is linear in the size of $\mathscr{A}_1$ and exponential in the size of $\mathscr{A}_2$, the required complexity follows, as in the proof of Theorem 17. Since $\mathscr{A}_2$ is universal iff $\Sigma^\omega \subseteq \mathscr{L}(\mathscr{A}_2)$ and Σ^ω can be recognized by an NBW with one state, hardness in PSPACE follows from hardness of the universality problem. □

Recall that the algorithm for deciding non-emptiness of an NBW $\mathscr{A}$ operates on the graph $G_\mathscr{A}$ induced by $\mathscr{A}$ and thus ignores the alphabet of $\mathscr{A}$. In particular, the algorithm does not distinguish between deterministic and nondeterministic automata. In contrast, the algorithms for deciding universality and containment complement the NBW and thus, can benefit from determinization.

Theorem 19. *The non-emptiness, non-universality, and containment problems for DBWs are NLOGSPACE-complete.*

Proof: When applied to DBWs, the intermediate automaton $\mathscr{A}'$ used in the proofs of Theorems 17 and 18 is polynomial in the size of the input, thus its non-emptiness can be tested in NLOGSPACE. Hardness in NLOGSPACE follows from the fact that reachability in directed graphs can be reduced to the three problems. □

Theorems 16, 17, and 18 refer to Büchi automata. For the other types of automata, one can translate to NBWs and apply the algorithm for them. While in many cases this results in an optimal algorithm, sometimes it is more efficient to work directly on the input automaton. In particular, for NSW, the translation to NBW results in an NBW with $O(n^2 2^k)$ states, whereas non-emptiness can be checked in sub-quadratic time [16, 25]. We note, however, that unlike NBWs and NRWs, for which the non-emptiness problem is NLOGSPACE-complete, it is PTIME-complete for NSWs [42]. For NPWs, the translation to NBWs results in an NBW with $O(nk)$ states, whereas non-emptiness can be checked in time $O(n \log k)$ [32].

5 Alternating Automata on Infinite Words

In [7], Chandra et al. introduced alternating Turing machines. In the alternating model, the states of the machine, and accordingly also its configurations, are partitioned into existential and universal ones. When the machine is in an existential configuration, one of its successors should lead to acceptance. When the machine is in a universal configuration, all its successors should lead to acceptance. In this section we define alternating Büchi automata [55] and study their properties.

5.1 Definition

For a given set X, let $\mathscr{B}^+(X)$ be the set of positive Boolean formulas over X (i.e., Boolean formulas built from elements in X using $\wedge$ and $\vee$), where we also allow the formulas **true** and **false**. For $Y \subseteq X$, we say that Y *satisfies* a formula $\theta \in \mathscr{B}^+(X)$ iff the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ. We say that Y satisfies θ *in a minimal manner* if no strict subset of Y satisfies θ. For example, the sets $\{q_1,q_3\}$, $\{q_2,q_3\}$, and $\{q_1,q_2,q_3\}$ all satisfy the formula $(q_1 \vee q_2) \wedge q_3$, yet only the first two sets satisfy it in a minimal manner. Also, the set $\{q_1,q_2\}$ does not satisfy this formula.

Consider an automaton $\mathscr{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. We can represent δ using $\mathscr{B}^+(Q)$. For example, a transition $\delta(q,\sigma) = \{q_1,q_2,q_3\}$ of a nondeterministic automaton $\mathscr{A}$ can be written as $\delta(q,\sigma) = q_1 \vee q_2 \vee q_3$. The dual of nondeterminism is universality. A word w is accepted by a universal automaton $\mathscr{A}$ if all the runs of $\mathscr{A}$ on w are accepting. Accordingly, if $\mathscr{A}$ is universal, then the transition can be written as $\delta(q,\sigma) = q_1 \wedge q_2 \wedge q_3$. While transitions of nondeterministic and universal automata correspond to disjunctions and conjunctions, respectively, transitions of alternating automata can be arbitrary formulas in $\mathscr{B}^+(Q)$. We can have, for instance, a transition $\delta(q,\sigma) = (q_1 \wedge q_2) \vee (q_3 \wedge q_4)$, meaning that the automaton accepts a word of the form $\sigma \cdot w$ from state q, if it accepts w from both q_1 and q_2 or from both q_3 and q_4. Such a transition combines existential and universal choices.

Formally, an *alternating automaton on infinite words* is a tuple $\mathscr{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, where Σ, Q, and α are as in nondeterministic automata, $q_0 \in Q$ is an initial state (we will later explain why it is technically easier to assume a single initial state), and $\delta : Q \times \Sigma \to \mathscr{B}^+(Q)$ is a transition function. In order to define runs of alternating automata, we first have to define trees and labeled trees. A *tree* is a prefix-closed set $T \subseteq \mathbb{N}^*$ (i.e., if $x \cdot c \in T$, where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T$). The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c$, for $c \in \mathbb{N}$, are the *successors* of x. A node is a *leaf* if it has no successors. We sometimes refer to the length $|x|$ of x as its *level* in the tree. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. Given an alphabet Σ, a *Σ-labeled tree* is a pair $\langle T, V \rangle$ where T is a tree and $V : T \to \Sigma$ maps each node of T to a letter in Σ.

While a run of a nondeterministic automaton on an infinite word is an infinite sequence of states, a run of an alternating automaton is a Q-labeled tree. Formally, given an infinite word $w = \sigma_1 \cdot \sigma_2 \cdots$, a run of $\mathscr{A}$ on w is a Q-labeled tree $\langle T_r, r \rangle$ such that the following hold:

- $\varepsilon \in T_r$ and $r(\varepsilon) = q_0$.
- Let $x \in T_r$ with $r(x) = q$ and $\delta(q, \sigma_{|x|+1}) = \theta$. There is a (possibly empty) set $S = \{q_1, \ldots, q_k\}$ such that S satisfies θ in a minimal manner and for all $1 \leq c \leq k$, we have that $x \cdot c \in T_r$ and $r(x \cdot c) = q_c$.

For example, if $\delta(q_0, \sigma_1) = (q_1 \vee q_2) \wedge (q_3 \vee q_4)$, then possible runs of $\mathscr{A}$ on w have a root labeled q_0, have one node in level 1 labeled q_1 or q_2, and have another node in level 1 labeled q_3 or q_4. Note that if $\theta = \mathbf{true}$, then x does not have children. This is the reason why T_r may have leaves. Also, since there exists no set S satisfying θ for $\theta = \mathbf{false}$, we cannot have a run that takes a transition with $\theta = \mathbf{false}$.

A run $\langle T_r, r \rangle$ is *accepting* iff all its infinite paths, which are labeled by words in Q^ω, satisfy the acceptance condition. A word w is accepted iff there exists an accepting run on it. Note that while conjunctions in the transition function of $\mathscr{A}$ are reflected in branches of $\langle T_r, r \rangle$, disjunctions are reflected in the fact we can have many runs on the same word. The language of $\mathscr{A}$, denoted $\mathscr{L}(\mathscr{A})$, is the set of infinite words that $\mathscr{A}$ accepts. We use ABW to abbreviate alternating Büchi word automata.

Example 4. For $n \geq 1$, let $\Sigma_n = \{1, 2, \ldots, n\}$. We describe an ABW $\mathscr{A}_n$ such that $\mathscr{A}_n$ accepts exactly all words $w \in \Sigma_n^\omega$ such that w contains the subword $i \cdot i \cdot i$ for all letters $i \in \Sigma_n$.

We define $\mathscr{A}_n = \langle \Sigma_n, Q_n, q_0, \delta, \emptyset \rangle$, where

- $Q_n = \{q_0\} \cup (\Sigma \times \{3, 2, 1\})$. Thus, in addition to an initial state, $\mathscr{A}_n$ contains three states for each letter $i \in \Sigma_n$, where state $\langle i, c \rangle$, for $c \in \{1, 2, 3\}$, waits for a subword i^c.
- In its first transition, $\mathscr{A}_n$ spawns into n copies, with copy i waiting for the subword i^3 (or i^2, in case the first letter read is i). Thus, for all $i \in \Sigma_n$, we have $\delta_n(q_0, i) = \langle i, 2 \rangle \wedge \bigwedge_{j \neq i} \langle j, 3 \rangle$. In addition, for all $i \in \Sigma_n$ and $c \in \{3, 2, 1\}$, we have

$$\delta_n(\langle i, c \rangle, j) = \begin{cases} \langle i, c-1 \rangle & \text{if } j = i \text{ and } c \in \{3, 2\}, \\ \mathbf{true} & \text{if } j = i \text{ and } c = 1, \\ \langle i, 3 \rangle & \text{if } j \neq i. \end{cases}$$

Note that no state in Q_n is accepting. Thus, all copies have to eventually take the transition to **true**, guaranteeing that $i \cdot i \cdot i$ is indeed read, for all $i \in \Sigma_n$.

Note also that while $\mathscr{A}_n$ has $3n + 1$ states, it is not hard to prove that an NBW for the language is exponential in n, as it has to remember the subsets of letters for which the subword $i \cdot i \cdot i$ has already appeared.

A slightly more general definition of alternating automata could replace the single initial state by an *initial transition* in $\mathscr{B}^+(Q)$, describing possible subsets of states from which the word should be accepted. Staying with the definition of a set

of initial states used in nondeterministic automata would have broken the symmetry between the existential and universal components of alternation.

5.2 Closure Properties

The rich structure of alternating automata makes it easy to define the union and intersection of ABWs. Indeed, the same way union is easy for automata with nondeterminism, intersection is easy for automata with universal branches.

Theorem 20. *Let $\mathscr{A}_1$ and $\mathscr{A}_2$ be ABWs with n_1 and n_2 states, respectively. There are ABWs $\mathscr{A}_\cup$ and $\mathscr{A}_\cap$ such that $\mathscr{L}(\mathscr{A}_\cup) = \mathscr{L}(\mathscr{A}_1) \cup \mathscr{L}(\mathscr{A}_2)$, $\mathscr{L}(\mathscr{A}_\cap) = \mathscr{L}(\mathscr{A}_1) \cap \mathscr{L}(\mathscr{A}_2)$, and $\mathscr{A}_\cup$ and $\mathscr{A}_\cap$ have $n_1 + n_2 + 1$ states.*

Proof: Let $\mathscr{A}_1 = \langle \Sigma, Q_1, q_1^0, \delta_1, \alpha_1 \rangle$ and $\mathscr{A}_2 = \langle \Sigma, Q_2, q_2^0, \delta_2, \alpha_2 \rangle$. We assume, without loss of generality, that Q_1 and Q_2 are disjoint. We define $\mathscr{A}_\cup$ as the union of $\mathscr{A}_1$ and $\mathscr{A}_2$, with an additional initial state that proceeds like the union of the initial states of $\mathscr{A}_1$ and $\mathscr{A}_2$. Thus, $\mathscr{A}_\cup = \langle \Sigma, Q_1 \cup Q_2 \cup \{q_0\}, q_0, \delta, \alpha_1 \cup \alpha_2 \rangle$, where $\delta(q_0, \sigma) = \delta_1(q_1^0, \sigma) \vee \delta_2(q_2^0, \sigma)$, and for every state $q \in Q_1 \cup Q_2$, we have that $\delta(q, \sigma) = \delta_i(q, \sigma)$, for the index $i \in \{1, 2\}$ such that $q \in Q_i$. It is easy to see that for every word $w \in \Sigma^\omega$, the ABW $\mathscr{A}$ has an accepting run on w iff at least one of the ABWs $\mathscr{A}_1$ and $\mathscr{A}_2$ has an accepting run on w. The definition of $\mathscr{A}_\cap$ is similar, except that from q_0 we proceed with the conjunction of the transitions from q_1^0 and q_2^0. □

We note that with a definition of ABWs in which an initial transition in $\mathscr{B}^+(Q)$ is allowed, closing ABWs under union and intersection can be done by applying the corresponding operation on the initial transitions.

We proceed to closure of ABWs under complementation. Given a transition function δ, let $\tilde{\delta}$ denote the function dual to δ. That is, for every q and σ with $\delta(q, \sigma) = \theta$, we have that $\tilde{\delta}(q, \sigma) = \tilde{\theta}$, where $\tilde{\theta}$ is obtained from θ by switching $\vee$ and $\wedge$ and switching **true** and **false**. If, for example, $\theta = p \vee (\mathbf{true} \wedge q)$, then $\tilde{\theta} = p \wedge (\mathbf{false} \vee q)$. Given an acceptance condition α, let $\tilde{\alpha}$ be an acceptance condition that dualizes α. Thus, a set of states S satisfies α iff S does not satisfy $\tilde{\alpha}$. In particular, if α is a Büchi condition, then $\tilde{\alpha}$ is a co-Büchi condition.

For deterministic automata, it is easy to complement an automaton $\mathscr{A}$ by dualizing the acceptance condition. In particular, given a DBW $\mathscr{A}$, viewing $\mathscr{A}$ as a DCW complements its language. For an NBW $\mathscr{A}$, the situation is more involved as we have to make sure that all runs satisfy the dual condition. This can be done by viewing $\mathscr{A}$ as a universal co-Büchi automaton. As Lemma 3 below argues, this approach can be generalized to all alternating automata and acceptance conditions.

Lemma 3. [58] *Given an alternating automaton $\mathscr{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$, the alternating automaton $\tilde{\mathscr{A}} = \langle \Sigma, Q, q_0, \tilde{\delta}, \tilde{\alpha} \rangle$ is such that $\mathscr{L}(\tilde{\mathscr{A}}) = \Sigma^\omega \setminus \mathscr{L}(\mathscr{A})$.*

Lemma 3 suggests a straightforward translation of an ABW $\mathscr{A}$ to a complementing ACW $\tilde{\mathscr{A}}$, and vice versa. In order to end up with an ABW, one has to translate $\tilde{\mathscr{A}}$ to an ABW [44], which uses the ranking method described in the context of NBW complementation and involves a quadratic blow-up:

Theorem 21. [44] *Given an ABW $\mathscr{A}$ with n states, there is an ABW $\mathscr{A}'$ with $O(n^2)$ states such that $\mathscr{L}(\mathscr{A}') = \Sigma^\omega \setminus \mathscr{L}(\mathscr{A})$.*

We note that the ABW constructed in the proof of Theorem 21 is a *weak alternating automaton* [56]. In a weak automaton, each SCC of the automaton is either contained in α or is disjoint from α. Every infinite path of a run ultimately gets "trapped" within some SCC. The path then satisfies the acceptance condition iff this component is contained in α.

It is easy to see that weak automata are a special case of both Büchi and co-Büchi alternating automata. A run gets trapped in a component contained in α iff it visits α infinitely often iff it visits $Q \setminus \alpha$ only finitely often. The study of weak alternating automata is motivated by the fact that the translation of formulas in several temporal logics to alternating automata results in weak automata [56, 46]. Another motivation is the fact that dualizing a weak automaton is straightforward: taking $\tilde{\alpha} = Q \setminus \alpha$ amounts to switching the classification of accepting and rejecting sets, and thus dualizes the acceptance condition.

Remark 2. In the non-elementary translation of monadic second-order logic formulas to NBWs [6], an exponential blow-up occurs with each negation. While a blow-up that is non-elementary in the quantifier alternation depth is unavoidable, the fact that complementation is easy for alternating automata raises the question whether ABWs may be used in a simpler decision procedure. The negative answer follows from the fact that the *existential projection* operator, which is easy for nondeterministic automata, involves an exponential blow-up when applied to alternating automata. For a language $L \subseteq (\Sigma_1 \times \Sigma_2)^\omega$, we define the existential projection of L on Σ_1 as the language L_1 of all words $w_1 \in \Sigma_1^\omega$ such that there is a word $w_2 \in \Sigma_2^\omega$ for which $w_1 \otimes w_2 \in L$, where $w_1 \otimes w_2$ is the word over $\Sigma_1 \times \Sigma_2$ obtained by "merging" the letters of w_1 and w_2 in the expected way. For example, $abba \otimes 0010 = \langle a0\rangle\langle b0\rangle\langle b1\rangle\langle a0\rangle$. Given an NBW for L, it is easy to see that an NBW for L_1 can be obtained by replacing a letter $\langle \sigma_1, \sigma_2\rangle$ by the letter σ_1. Such a simple replacement, however, would not work for alternating automata. Indeed, there, one has to ensure that different copies of the automaton proceed according to the same word over Σ_2. Consequently, existential projection requires alternation removal. In the context of translations of formulas to automata, the exponential blow-up with each negation when working with NBWs is traded for an exponential blow-up with each existential quantifier when working with ABWs. It is easy to see, say by pushing negations inside, that negations and existential quantifiers can be traded also at the syntactic level of the formula.

5.3 Decision Procedures

The rich structure of alternating automata makes them exponentially more succinct than nondeterministic automata. On the other hand, reasoning about alternating automata is complicated. For example, while the algorithm for testing the non-emptiness of a nondeterministic automaton can ignore the alphabet and be reduced to reachability questions in the underlying graph of the automaton, ignoring the alphabet in an alternating automaton leads to an algorithm with a one-sided error. Indeed, as noted in the context of existential projection in Remark 2, the algorithm should make sure that the different copies it spawns into follow the same word. Consequently, many algorithms for alternating automata involve alternation removal – a translation to an equivalent nondeterministic automaton. Below we describe such a translation for the case of Büchi automata.

Theorem 22. [55] *Consider an ABW $\mathscr{A}$ with n states. There is an NBW $\mathscr{A}'$ with 3^n states such that $\mathscr{L}(\mathscr{A}') = \mathscr{L}(\mathscr{A})$.*

Proof: The automaton $\mathscr{A}'$ guesses a run of $\mathscr{A}$. At a given point of a run of $\mathscr{A}'$, it keeps in its memory the states in a whole level of the run tree of $\mathscr{A}$. As it reads the next input letter, it guesses the states in the next level of the run tree of $\mathscr{A}$. In order to make sure that every infinite path visits states in α infinitely often, $\mathscr{A}'$ keeps track of states that "owe" a visit to α. Let $\mathscr{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$. Then $\mathscr{A}' = \langle \Sigma, 2^Q \times 2^Q, \langle \{q_0\}, \emptyset \rangle, \delta', 2^Q \times \{\emptyset\} \rangle$, where δ' is defined, for all $\langle S, O \rangle \in 2^Q \times 2^Q$ and $\sigma \in \Sigma$, as follows.

- If $O \neq \emptyset$, then $\delta'(\langle S, O \rangle, \sigma) = \{\langle S', O' \setminus \alpha \rangle \ : \ S' \text{ satisfies } \bigwedge_{q \in S} \delta(q, \sigma), O' \subseteq S', \text{ and } O' \text{ satisfies } \bigwedge_{q \in O} \delta(q, \sigma)\}$.
- If $O = \emptyset$, then $\delta'(\langle S, O \rangle, \sigma) = \{\langle S', S' \setminus \alpha \rangle \ : \ S' \text{ satisfies } \wedge_{q \in S} \delta(q, \sigma)\}$.

Note that all the reachable states $\langle S, O \rangle$ in $\mathscr{A}'$ satisfy $O \subseteq S$. Accordingly, if the number of states in $\mathscr{A}$ is n, then the number of states in $\mathscr{A}'$ is at most 3^n. □

Note that the construction has the flavor of the subset construction [63], but in a dual interpretation: a set of states is interpreted conjunctively: the suffix of the word has to be accepted from all the states in S. While such a dual subset construction is sufficient for automata on finite words, the case of Büchi requires also the maintenance of a subset O of S, leading to a $3^{O(n)}$, rather than a $2^{O(n)}$, blow-up. As shown in [3], this additional blow-up cannot be avoided.

Remark 3. It is not hard to see that if $\mathscr{A}$ is a universal automaton (that is, the transition function δ only has conjunctions), then the automaton $\mathscr{A}'$ constructed in the proof of Theorem 22 is deterministic. Indeed, in the definition of $\delta'(\langle S, O \rangle, \sigma)$, there is a single set S' that satisfies $\wedge_{q \in S} \delta(q, \sigma)$ in a minimal manner. It follows that universal Büchi automata are not more expressive than DBWs. Dually, NCWs are not more expressive than DCW: Given an NCW $\mathscr{A}$, we can apply the construction above on the dual universal Büchi automaton $\tilde{\mathscr{A}}$ (see Lemma 3), and then dualize the obtained DBW. We end up with a DCW equivalent to $\mathscr{A}$.

We can now use alternation removal in order to solve decision problems for alternating automata.

Theorem 23. *The non-emptiness, non-universality, and containment problems for ABW are PSPACE-complete.*

Proof: We describe the proof for the non-emptiness problem. Since ABWs are easily closed for negation and intersection, the proof for non-universality and containment is similar. Consider an ABW $\mathscr{A}$. In order to check $\mathscr{A}$ for non-emptiness, we translate it into an NBW $\mathscr{A}'$ and check the non-emptiness of $\mathscr{A}'$. By Theorem 22, the size of $\mathscr{A}'$ is exponential in the size of $\mathscr{A}$. Since the construction of $\mathscr{A}'$ can proceed on-the-fly, and, by Theorem 16, its non-emptiness can be checked in NLOGSPACE, membership in PSPACE follows.

In order to prove hardness in PSPACE, we do a reduction from NBW non-universality. Given an NBW $\mathscr{A}$, we have that $\mathscr{L}(\mathscr{A}) \neq \Sigma^\omega$ iff $\Sigma^\omega \setminus \mathscr{L}(\mathscr{A}) \neq \emptyset$. Thus, non-universality of $\mathscr{A}$ can be reduced to non-emptiness of an automaton $\mathscr{A}'$ that complements $\mathscr{A}$. Since we can define $\mathscr{A}'$ as an ABW with quadratically many states, hardness in PSPACE follows. □

6 Automata-Based Algorithms

In this section we describe the application of automata theory in formal verification. Recall that the logic LTL is used for specifying properties of reactive systems. The syntax and semantics of LTL are described in Chapter 2. For completeness, we describe them here briefly. Formulas of LTL are constructed from a set *AP* of atomic propositions using the usual Boolean operators and the temporal operators X ("next time") and U ("until"). Formally, an LTL formula over *AP* is defined as follows:

- **true**, **false**, or p, for $p \in AP$.
- $\neg\psi_1$, $\psi_1 \wedge \psi_2$, $X\psi_1$, or $\psi_1 U \psi_2$, where ψ_1 and ψ_2 are LTL formulas.

The semantics of LTL is defined with respect to infinite computations $\pi = \sigma_1, \sigma_2, \sigma_3, \ldots$, where for every $j \geq 1$, the set $\sigma_j \subseteq AP$ is the set of atomic propositions that hold in the j-th position of π. Systems that generate computations are modeled by *Kripke structures*. A (finite) Kripke structure is a tuple $K = \langle AP, W, W_0, R, \ell \rangle$, where *AP* is a finite set of atomic propositions, W is a finite set of states, $W_0 \subseteq W$ is a set of initial states, $R \subseteq W \times W$ is a transition relation, and $\ell : W \to 2^{AP}$ maps each state w to the set of atomic propositions that hold in w. We require that each state has at least one successor. That is, for each state $w \in W$ there is at least one state w' such that $R(w, w')$. A path in K is an infinite sequence $\rho = w_0, w_1, w_2, \ldots$ of states such that $w_0 \in W_0$ and for all $i \geq 0$, we have $R(w_i, w_{i+1})$. The path ρ induces the computation $\ell(w_0), \ell(w_1), \ell(w_2), \ldots$.

Consider a computation $\pi = \sigma_1, \sigma_2, \sigma_3, \ldots$. We denote the suffix $\sigma_j, \sigma_{j+1}, \ldots$ of π by π^j. We use $\pi \models \psi$ to indicate that an LTL formula ψ holds in the computation π. The relation $\models$ is inductively defined as follows:

- For all π, we have that $\pi \models \mathbf{true}$ and $\pi \not\models \mathbf{false}$.
- For an atomic proposition $p \in AP$, we have that $\pi \models p$ iff $p \in \sigma_1$.
- $\pi \models \neg\psi_1$ iff $\pi \not\models \psi_1$.
- $\pi \models \psi_1 \wedge \psi_2$ iff $\pi \models \psi_1$ and $\pi \models \psi_2$.
- $\pi \models X\psi_1$ iff $\pi^2 \models \psi_1$.
- $\pi \models \psi_1 U \psi_2$ iff there exists $k \geq 1$ such that $\pi^k \models \psi_2$ and $\pi^i \models \psi_1$ for all $1 \leq i < k$.

Writing LTL formulas, it is convenient to use the abbreviations G ("always"), F ("eventually"), and R ("release"). Formally, the abbreviations follow the following semantics.

- $F\psi_1 = \mathbf{true} U \psi_1$. That is, $\pi \models F\psi_1$ iff there exists $k \geq 1$ such that $\pi^k \models \psi_1$.
- $G\psi_1 = \neg F \neg \psi_1$. That is, $\pi \models G\psi_1$ iff for all $k \geq 1$ we have that $\pi^k \models \psi_1$.
- $\psi_1 R \psi_2 = \neg((\neg\psi_1) U (\neg\psi_2))$. That is, $\pi \models \psi_1 R \psi_2$ iff for all $k \geq 1$, if $\pi^k \not\models \psi_2$, then there is $1 \leq i < k$ such that $\pi^i \models \psi_1$.

Each LTL formula ψ over AP defines a language $\mathcal{L}(\psi) \subseteq (2^{AP})^\omega$ of the computations that satisfy ψ, Formally,

$$\mathcal{L}(\psi) = \{\pi \in (2^{AP})^\omega \ : \ \pi \models \psi\}.$$

Two natural problems arise in the context of systems and their specifications:

- *Satisfiability:* given an LTL formula ψ, is there a computation π such that $\pi \models \psi$?
- *Model Checking:* given a Kripke structure K and an LTL formula ψ, do all the computations of K satisfy ψ?

We describe a translation of LTL formulas into Büchi automata and discuss how such a translation is used for solving the above two problems.

6.1 Translating LTL to Büchi Automata

In this section we describe a translation of LTL formulas to NBW. We start with a translation that goes via ABWs. For completeness, we also present the original translation of [77], which directly generates NBWs. The translation involves an exponential blow-up, which we show to be tight.

6.1.1 A Translation via ABWs

Consider an LTL formula ψ. For simplicity, we assume that ψ is given in *positive normal form*. Thus, negation is applied only to atomic propositions. Formally, given a set AP of atomic propositions, an LTL formula in positive normal form is defined as follows:

- **true**, **false**, p, or $\neg p$, for $p \in AP$.

- ψ_1, $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $X\psi_1$, $\psi_1 U \psi_2$, or $\psi_1 R \psi_2$, where ψ_1 and ψ_2 are LTL formulas in positive normal form.

Note that the fact negation is restricted to atomic propositions has forced us to add not only the Boolean operator $\vee$ but also the temporal operator R. Still, it is easy to see that transforming an LTL formula to a formula in positive normal form involves no blow-up. The *closure* of an LTL formula ψ, denoted $cl(\psi)$, is the set of all its subformulas. Formally, $cl(\psi)$ is the smallest set of formulas that satisfy the following.

- $\psi \in cl(\psi)$.
- If $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$, $\psi_1 U \psi_2$ or $\psi_1 R \psi_2$ is in $cl(\psi)$, then $\psi_1 \in cl(\psi)$ and $\psi_2 \in cl(\psi)$.
- If $X\psi_1$ is in $cl(\psi)$, then $\psi_1 \in cl(\psi)$.

For example, $cl(p \wedge ((Xp)Uq))$ is $\{p \wedge ((Xp)Uq), p, (Xp)Uq, Xp, q\}$. It is easy to see that the size of $cl(\psi)$ is linear in $|\psi|$.

Theorem 24. *For every LTL formula ψ, there is an ABW $\mathscr{A}_\psi$ with $O(|\psi|)$ states such that $\mathscr{L}(\mathscr{A}_\psi) = \mathscr{L}(\psi)$.*

Proof: We define $\mathscr{A}_\psi = \langle 2^{AP}, cl(\psi), \psi, \delta, \alpha \rangle$, where

- The transition $\delta(\varphi, \sigma)$ is defined according to the form of φ as follows.
  - $\delta(p, \sigma) = \begin{cases} \textbf{true} & \text{if } p \in \sigma, \\ \textbf{false} & \text{if } p \notin \sigma. \end{cases}$
  - $\delta(\neg p, \sigma) = \begin{cases} \textbf{true} & \text{if } p \notin \sigma, \\ \textbf{false} & \text{if } p \in \sigma. \end{cases}$
  - $\delta(\varphi_1 \wedge \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \wedge \delta(\varphi_2, \sigma)$.
  - $\delta(\varphi_1 \vee \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \vee \delta(\varphi_2, \sigma)$.
  - $\delta(X\varphi, \sigma) = \varphi$.
  - $\delta(\varphi_1 U \varphi_2, \sigma) = \delta(\varphi_2, \sigma) \vee (\delta(\varphi_1, \sigma) \wedge \varphi_1 U \varphi_2)$.
  - $\delta(\varphi_1 R \varphi_2, \sigma) = \delta(\varphi_2, \sigma) \wedge (\delta(\varphi_1, \sigma) \vee \varphi_1 R \varphi_2)$.
- The set α of accepting states consists of all the formulas in $cl(\psi)$ of the form $\varphi_1 R \varphi_2$.

The proof of the correctness of the construction proceeds by induction on the structure of ψ. For a formula $\varphi \in cl(\psi)$, we prove that when $\mathscr{A}_\psi$ is in state φ, it accepts exactly all words that satisfy φ. The base case, when φ is an atomic proposition or its negation, follows from the definition of the transition function. The other cases follow from the semantics of LTL and the induction hypothesis. In particular, the definition of α guarantees that in order for a word to satisfy $\varphi_1 U \varphi_2$, it must have a suffix that satisfies φ_2. Indeed, otherwise, the run of $\mathscr{A}_\psi$ has an infinite path that remains forever in the state $\varphi_1 U \varphi_2$, and thus does not satisfy α. □

Example 5. We describe an ABW $\mathscr{A}_\psi$ for the LTL formula $\psi = GFp$. Note that $\psi = \textbf{false} R (\textbf{true} U p)$. In the example, we use the F and G abbreviations. The alphabet of $\mathscr{A}_\psi$ consists of the two letters in $2^{\{p\}}$. The set of accepting states is $\{GFp\}$, and the states and transitions are described in the table in Fig. 8.

	$\{p\}$	$\emptyset$
GFp	GFp	$GFp \wedge Fp$
Fp	**true**	Fp

Fig. 8 The transition function of an ABW for GFp

Example 6. We describe an ABW $\mathcal{A}_\psi$ for the LTL formula $\psi = p \wedge ((Xp)Uq)$. The alphabet of $\mathcal{A}_\psi$ consists of the four letters in $2^{\{p,q\}}$. The states and transitions are described in the table in Fig. 9. No state is accepting. Note that only the initial state is reachable.

	$\{p,q\}$	$\{p\}$	$\{q\}$	$\emptyset$
$p \wedge ((Xp)Uq))$	**true**	$p \wedge ((Xp)Uq)$	**false**	**false**
p	**true**	**true**	**false**	**false**
$(Xp)Uq$	**true**	$p \wedge ((Xp)Uq))$	**true**	$p \wedge ((Xp)Uq))$

Fig. 9 The transition function of an ABW for $p \wedge ((Xp)Uq))$

Combining Theorems 24 and 22, we get the following.

Theorem 25. *For every LTL formula ψ, there is an NBW $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi) = \mathcal{L}(\psi)$ and the size of $\mathcal{A}_\psi$ is exponential in $|\psi|$.*

In Section 6.1.3 we show a matching exponential lower bound. Let us note here that while the 3^n blow-up in Theorem 22 refers to general ABWs, the ABWs obtained from LTL in the proof of Theorem 24 have a special structure: all the cycles in the automata are self-loops. For such automata (termed *very-weak* alternating automata, as they are weak alternating automata in which all SCCs are singletons), alternation can be removed with only an $n2^n$ blow-up [20, 3].

6.1.2 A Direct Translation to NBWs

The original translation of LTL to NBW [77] does not go via intermediate alternating automata. For completeness, we detail it here. The translation does not assume a positive normal form, and uses the *extended closure* of the given formula: For an LTL formula ψ, the extended closure of ψ, denoted $ecl(\psi)$, is the set of ψ's subformulas and their negations ($\neg\neg\psi$ is identified with ψ). Formally, $ecl(\psi)$ is the smallest set of formulas that satisfy the following.

- $\psi \in ecl(\psi)$.
- If $\psi_1 \in ecl(\psi)$ then $\neg\psi_1 \in ecl(\psi)$.
- If $\neg\psi_1 \in ecl(\psi)$ then $\psi_1 \in ecl(\psi)$.
- If $\psi_1 \wedge \psi_2 \in ecl(\psi)$ then $\psi_1 \in ecl(\psi)$ and $\psi_2 \in ecl(\psi)$.

- If $X\psi_1 \in ecl(\psi)$ then $\psi_1 \in ecl(\psi)$.
- If $\psi_1 U \psi_2 \in ecl(\psi)$ then $\psi_1 \in ecl(\psi)$ and $\psi_2 \in ecl(\psi)$.

For example, $ecl(p \wedge ((Xp)Uq))$ is $\{p \wedge ((Xp)Uq), \neg(p \wedge ((Xp)Uq)), p, \neg p, (Xp)Uq, \neg((Xp)Uq), Xp, \neg Xp, q, \neg q\}$.

The translation is based on the observation that the question of satisfaction of an LTL formula ψ in a computation π can be reduced to questions about the satisfaction of formulas in $ecl(\psi)$ in the suffixes of π. More formally, given a computation π, it is possible to (uniquely) label each suffix of π by the subset of formulas in $ecl(\psi)$ that are satisfied in this suffix. The correctness of this labeling can be verified by local consistency checks, which relate the labeling of successive suffixes, and by a global consistency check, which takes care of satisfaction of eventualities. Since it is easier to check the satisfaction of each eventuality in isolation, we describe a translation to nondeterministic automata with the generalized Büchi acceptance condition. One can then use Theorem 12 in order to obtain an NBW.

Formally, given an LTL formula ψ over AP, we define $\mathscr{A}_\psi = \langle 2^{AP}, Q, Q_0, \delta, \alpha \rangle$, as follows.

- We say that a set $S \subseteq ecl(\psi)$ is *good in* $ecl(\psi)$ if S is a maximal set of formulas in $ecl(\psi)$ that does not have propositional inconsistency. Thus, S satisfies the following conditions.
  1. For all $\psi_1 \in ecl(\psi)$, we have $\psi_1 \in S$ iff $\neg\psi_1 \notin S$, and
  2. For all $\psi_1 \wedge \psi_2 \in ecl(\psi)$, we have $\psi_1 \wedge \psi_2 \in S$ iff $\psi_1 \in S$ and $\psi_2 \in S$.

  The state space $Q \subseteq 2^{ecl(\psi)}$ is the set of all the good sets in $ecl(\psi)$.
- Let S and S' be two good sets in $ecl(\psi)$, and let $\sigma \subseteq AP$ be a letter. Then $S' \in \delta(S, \sigma)$ if the following hold.
  1. $\sigma = S \cap AP$,
  2. For all $X\psi_1 \in ecl(\psi)$, we have $X\psi_1 \in S$ iff $\psi_1 \in S'$, and
  3. For all $\psi_1 U \psi_2 \in ecl(\psi)$, we have $\psi_1 U \psi_2 \in S$ iff either $\psi_2 \in S$ or both $\psi_1 \in S$ and $\psi_1 U \psi_2 \in S'$.

  Note that the last condition also means that for all $\neg(\psi_1 U \psi_2) \in ecl(\psi)$, we have that $\neg(\psi_1 U \psi_2) \in S$ iff $\neg\psi_2 \in S$ and either $\neg\psi_1 \in S$ or $\neg(\psi_1 U \psi_2) \in S'$.
- $Q_0 \subseteq Q$ is the set of all states $S \in Q$ for which $\psi \in S$.
- Every formula $\psi_1 U \psi_2$ contributes to the generalized Büchi condition α the set

$$\alpha_{\psi_1 U \psi_2} = \{S \in Q : \psi_2 \in S \text{ or } \neg(\psi_1 U \psi_2) \in S\}.$$

We now turn to discuss the size of $\mathscr{A}_\psi$. It is easy to see that the size of $ecl(\psi)$ is $O(|\psi|)$, so $\mathscr{A}_\psi$ has $2^{O(|\psi|)}$ states. Note that since ψ has at most $|\psi|$ subformulas of the form $\psi_1 U \psi_2$, the index of α is at most $|\psi|$. It follows from Theorem 12 that ψ can also be translated into an NBW with $2^{O(|\psi|)}$ states.

Remark 4. Note that the construction of $\mathscr{A}_\psi$ can proceed *on-the-fly*. Thus, given a state S of $\mathscr{A}_\psi$ and a letter $\sigma \in 2^{AP}$, it is possible to compute the set $\delta(S, \sigma)$ based on the formulas in S. As we shall see in Section 6.2, this fact is very helpful, as it

implies that reasoning about $\mathcal{A}_\psi$ need not construct the whole state space of $\mathcal{A}_\psi$ but can rather proceed in an on-demand fashion.

6.1.3 The blow-up in the LTL to NBW Translation

In this section we describe an exponential lower bound for the translation of LTL to NBW, implying that the blow-up that both translations above involve cannot in general be avoided. We do so by describing a doubly-exponential lower bound for the translation of LTL to DBW. Recall that NBWs are strictly more expressive than DBWs. The expressiveness gap carries over to languages that can be specified in LTL. For example, the formula FGb ("eventually always b", which is similar to the language used in the proof of Theorem 6), cannot be translated into a DBW. We now show that when a translation exists, it is doubly-exponential. Thus, the exponential blow-ups in Theorem 25 and determinization (when possible) are additive:

Theorem 26. *When possible, the translation of LTL formulas to deterministic Büchi automata is doubly-exponential.*

Proof: Let ψ be an LTL formula of length n and let $\mathcal{A}_\psi$ be an NBW that recognizes ψ. By Theorem 25, the automaton $\mathcal{A}_\psi$ has $2^{O(n)}$ states. By determinizing $\mathcal{A}_\psi$, we get a DPW $\mathcal{U}_\psi$ with $2^{2^{O(n)}}$ states [64, 59]. By Büchi typeness of DPWs [35] (see also Theorem 9), if $\mathcal{U}_\psi$ has an equivalent DBW, it can be translated into a DBW with the same state space. Hence the upper bound.

For the lower bound, consider the following ω-regular language L_n over the alphabet $\{0,1,\#,\$\}$:[4]

$$L_n = \{\{0,1,\#\}^* \cdot \# \cdot w \cdot \# \cdot \{0,1,\#\}^* \cdot \$ \cdot w \cdot \#^\omega \ : w \in \{0,1\}^n\}.$$

A word τ is in L_n iff the suffix of length n that comes after the single \$ in τ appears somewhere before the \$. By [7], the smallest deterministic automaton that accepts L_n has at least 2^{2^n} states. (The proof in [7] considers the language of the finite words obtained from L_n by omitting the $\#^\omega$ suffix. The proof, however, is independent of this technical detail: reaching the \$, the automaton should remember the possible set of words in $\{0,1\}^n$ that have appeared before.) We can specify L_n with an LTL formula of length quadratic in n. The formula is a conjunction of two formulas. The first conjunct, ψ_1, makes sure that there is only one \$ in the word, followed by a word in $\{0,1\}^n$, which is followed by an infinite tail of #'s. The second conjunct, ψ_2, states that eventually there exists a position with # and for all $1 \leq i \leq n$, the i-th letter from this position is 0 or 1 and it agrees with the i-th letter after the \$. Also, the $(n+1)$-th letter from this position is #. Formally,

- $\psi_1 = (\neg\$)U(\$ \wedge X((0 \vee 1) \wedge X(0 \vee 1) \wedge \overset{n}{\cdots} X((0 \vee 1) \wedge XG\#)))\cdots)$.
- $\psi_2 = F(\# \wedge X^{n+1}\# \wedge \bigwedge_{1 \leq i \leq n}((X^i 0 \wedge G(\$ \rightarrow X^i 0)) \vee (X^i 1 \wedge G(\$ \rightarrow X^i 1)))))$.

[4] Note that, for technical convenience, the alphabet of L_n is not of the form 2^{AP}. It is easy to adjust the proof to this setting, say by encoding $\{0,1,\#,\$\}$ by two atomic propositions.

Note that the argument about the size of the smallest deterministic automaton that recognizes L_n is independent of the automaton's acceptance condition. Thus, the theorem holds for deterministic Rabin, Streett, and Muller automata as well. □

6.2 Model Checking and Satisfiability

In this section we describe the automata-theoretic approach to LTL satisfiability and model checking. We show how, using the translation of LTL into NBW, these problems can be reduced to problems about automata and their languages.

Theorem 27. *The LTL satisfiability problem is PSPACE-complete.*

Proof: An LTL formula ψ is satisfiable iff the automaton $\mathscr{A}_\psi$ is not empty. Indeed, $\mathscr{A}_\psi$ accepts exactly all the computations that satisfy ψ. By Theorem 16, the non-emptiness problem for NBWs is in NLOGSPACE. Since the size of $\mathscr{A}_\psi$ is exponential in $|\psi|$, and its construction can be done on-the-fly, membership in PSPACE follows. Hardness in PSPACE is proved in [69], and the proof is similar to the hardness proof we detailed for NBW non-universality in Theorem 17. Indeed, as there, given a polynomial-space Turing machine T, we can construct an LTL formula ψ_T of polynomial size that describes exactly all words that either do not encode a legal computation of T on the empty tape, or encode a rejecting computation. The formula $\neg\psi$ is then satisfiable iff T accepts the empty tape. □

Theorem 28. *The LTL model-checking problem is PSPACE-complete.*

Proof: Consider a Kripke structure $K = \langle AP, W, W_0, R, \ell\rangle$. We construct an NBW $\mathscr{A}_K$ such that $\mathscr{A}_K$ accepts a computation $\pi \in (2^{AP})^\omega$ iff π is a computation of K. The construction of $\mathscr{A}_K$ essentially moves the labels of K from the states to the transitions. Thus, $\mathscr{A}_K = \langle 2^{AP}, W, W_0, \delta, W\rangle$, where for all $w \in W$ and $\sigma \in 2^{AP}$, we have

$$\delta(w,\sigma) = \begin{cases} \{w' : R(w,w')\} & \text{if } \sigma = \ell(w). \\ \emptyset & \text{if } \sigma \neq \ell(w). \end{cases}$$

Now, K satisfies ψ iff all the computations of K satisfy ψ, thus $\mathscr{L}(\mathscr{A}_K) \subseteq \mathscr{L}(\mathscr{A}_\psi)$. A naive check of the above would construct $\mathscr{A}_\psi$ and complement it. Complementation, however, involves an exponential blow-up, on top of the exponential blow-up in the translation of ψ to $\mathscr{A}_\psi$. Instead, we exploit the fact that LTL formulas are easy to complement and check that $\mathscr{L}(\mathscr{A}_K) \cap \mathscr{L}(\mathscr{A}_{\neg\psi}) = \emptyset$, where $\mathscr{A}_{\neg\psi}$ is the NBW for $\neg\psi$. Accordingly, the model-checking problem can be reduced to the non-emptiness problem of the intersection of A_K and $\mathscr{A}_{\neg\psi}$. Let $\mathscr{A}_{K,\neg\psi}$ be an NBW accepting the intersection of the languages of $\mathscr{A}_K$ and $\mathscr{A}_{\neg\psi}$. Since $\mathscr{A}_K$ has no acceptance condition, the construction of $\mathscr{A}_{K,\neg\psi}$ can proceed by simply taking the product of $\mathscr{A}_K$ with $\mathscr{A}_{\neg\psi}$. Then, K satisfies ψ iff $\mathscr{A}_{K,\neg\psi}$ is empty. By Theorem 25, the size of $\mathscr{A}_{\neg\psi}$ is exponential in $|\psi|$. Also, the size of $\mathscr{A}_K$ is linear in $|K|$. Thus, the size of $\mathscr{A}_{K,\neg\psi}$ is $|K| \cdot 2^{O(|\psi|)}$. Since the construction of $\mathscr{A}_{\neg\psi}$, and hence also $\mathscr{A}_{K,\neg\psi}$, can be done

on-the-fly, membership in PSPACE follows from the membership in NLOGSPACE of the non-emptiness problem for NBW. Hardness in PSPACE is proved in [69], and again, proceeds by a generic reduction from polynomial-space Turing machines. □

As described in the proof of Theorem 28, the PSPACE complexity of the LTL model-checking problem follows from the exponential size of the product automaton $\mathcal{A}_{K,\neg\psi}$. Note that $\mathcal{A}_{K,\neg\psi}$ is exponential only in $|\psi|$, and is linear in $|K|$. Nevertheless, as K is typically much bigger than ψ, and the exponential blow-up of the translation of ψ to $\mathcal{A}_{\neg\psi}$ rarely appears in practice, it is the linear dependency in $|K|$, rather than the exponential dependency in $|\psi|$, that makes LTL model checking so challenging in practice.

We note that the translations described in Section 6.1 are the classic ones. Since their introduction, researchers have suggested many heuristics and optimizations, with rapidly changing state of the art. Prominent ideas involve a reduction of the state space by associating states with smaller subsets of the closure [21], possibly as a result of starting with alternating automata [46, 20], reductions based on relations between the states, in either the alternating or nondeterministic automaton [71, 19], working with acceptance conditions that are defined with respect to edges rather than states [22], and a study of easy fragments [34]. In addition, variants of NBWs are used for particular applications, such as *testers* in the context of composition reasoning [60]. Finally, the automata-theoretic approach has been extended also to *branching temporal logics*. There, formulas are interpreted over branching structures, and the techniques are based on automata on infinite trees [76, 12, 13, 57, 46].

Acknowledgement I thank Javier Esparza and Moshe Y. Vardi for many helpful comments and discussions.

References

1. R. Alur, T.A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proc. 38th IEEE Symp. on Foundations of Computer Science*, pages 100–109, 1997.
2. R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M.Y. Vardi, and Y. Zbar. The ForSpec temporal logic: A new temporal property-specification logic. In *Proc. 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 2280 of *Lecture Notes in Computer Science*, pages 196–211. Springer, 2002.
3. U. Boker, O. Kupferman, and A. Rosenberg. Alternation removal in Büchi automata. In *Proc. 37th Int. Colloq. on Automata, Languages, and Programming*, volume 6199, pages 76–87, 2010.
4. U. Boker, O. Kupferman, and A. Steinitz. Parityzing Rabin and Streett. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, pages 412–423, 2010.
5. S. Breuers, C. Löding, and Jörg Olschewski. Improved Ramsey-based Büchi complementation. In *Proc. 15th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 7213 of *Lecture Notes in Computer Science*, pages 150–164. Springer, 2012.

6. J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Int. Congress on Logic, Method, and Philosophy of Science. 1960*, pages 1–12. Stanford University Press, 1962.
7. A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.
8. Y. Choueka. Theories of automata on ω-tapes: A simplified approach. *Journal of Computer and Systems Science*, 8:117–141, 1974.
9. A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model checker. *Software Tools for Technology Transfer*, 2(4):410–425, 2000.
10. T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 1990.
11. J.-M. Couvreur. On-the-fly verification of linear temporal logic. In *World Congress on Formal Methods*, volume 1708 of *Lecture Notes in Computer Science*, pages 253–271. Springer, 1999.
12. E.A. 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.
13. E.A. Emerson and C. Jutla. Tree automata, μ-calculus and determinacy. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 368–377, 1991.
14. E.A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 84–96, 1985.
15. E.A. Emerson and C.-L. Lei. Temporal model checking under generalized fairness constraints. In *Proc. 18th Hawaii Int. Conf. on System Sciences*. Western Periodicals Company, 1985.
16. E.A. Emerson and C.-L. Lei. Modalities for model checking: Branching time logic strikes back. *Science of Computer Programming*, 8:275–306, 1987.
17. S. Fogarty, O. Kupferman, M.Y. Vardi, and T. Wilke. Unifying Büchi complementation constructions. In *Proc. 20th Annual Conf. of the European Association for Computer Science Logic*, pages 248–263, 2011.
18. E. Friedgut, O. Kupferman, and M.Y. Vardi. Büchi complementation made tighter. *Int. J. Found. Comput. Sci.*, 17(4):851–868, 2006.
19. C. Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In *Proc. 8th Int. Conf. on Implementation and Application of Automata*, number 2759 in Lecture Notes in Computer Science, pages 35–48. Springer, 2003.
20. P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *Proc. 13th Int. Conf. on Computer Aided Verification*, volume 2102 of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.
21. R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In P. Dembiski and M. Sredniawa, editors, *Protocol Specification, Testing, and Verification*, pages 3–18. Chapman & Hall, 1995.
22. D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *Proc. 22nd International Conference on Formal Techniques for Networked and Distributed Systems*, volume 2529 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2002.
23. P. Godefroid and P. Wolper. A partial approach to model checking. *Information and Computation*, 110(2):305–326, 1994.
24. R.H. Hardin, Z. Har'el, and R.P. Kurshan. COSPAN. In *Proc. 8th Int. Conf. on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 423–427. Springer, 1996.
25. M. Henzinger and J.A. Telle. Faster algorithms for the nonemptiness of Streett automata and for communication protocol pruning. In *Proc. 5th Scandinavian Workshop on Algorithm Theory*, volume 1097 of *Lecture Notes in Computer Science*, pages 10–20. Springer, 1996.
26. T.A. Henzinger, O. Kupferman, and M.Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Proc. 7th Int. Conf. on Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 514–529. Springer, 1996.
27. G.J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.

28. N. Immerman. Nondeterministic space is closed under complement. *Information and Computation*, 17:935–938, 1988.
29. M. Jurdzinski. Small progress measures for solving parity games. In *Proc. 17th Symp. on Theoretical Aspects of Computer Science*, volume 1770 of *Lecture Notes in Computer Science*, pages 290–301. Springer, 2000.
30. D. Kähler and T. Wilke. Complementation, disambiguation, and determinization of Büchi automata unified. In *Proc. 35th Int. Colloq. on Automata, Languages, and Programming*, volume 5126 of *Lecture Notes in Computer Science*, pages 724–735. Springer, 2008.
31. S. Katz and D. Peled. Interleaving set temporal logic. *Theoretical Computer Science*, 75(3):263–287, 1990.
32. V. King, O. Kupferman, and M.Y. Vardi. On the complexity of parity word automata. In *Proc. 4th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 2030 of *Lecture Notes in Computer Science*, pages 276–286. Springer, 2001.
33. N. Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *Proc. 32nd IEEE Symp. on Foundations of Computer Science*, pages 358–367, 1991.
34. J. Kretínský and J. Esparza. Deterministic automata for the (f, g)-fragment of ltl. In *Proc. 24th Int. Conf. on Computer Aided Verification*, volume 7358 of *Lecture Notes in Computer Science*, pages 7–22. Springer, 2012.
35. S.C. Krishnan, A. Puri, and R.K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. In *Algorithms and Computations*, volume 834 of *Lecture Notes in Computer Science*, pages 378–386. Springer, 1994.
36. O. Kupferman. Avoiding determinization. In *Proc. 21st IEEE Symp. on Logic in Computer Science*, pages 243–254, 2006.
37. O. Kupferman. Sanity checks in formal verification. In *Proc. 17th Int. Conf. on Concurrency Theory*, volume 4137 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2006.
38. O. Kupferman, G. Morgenstern, and A. Murano. Typeness for ω-regular automata. *International Journal on the Foundations of Computer Science*, 17(4):869–884, 2006.
39. O. Kupferman, N. Piterman, and M.Y. Vardi. Extended temporal logic revisited. In *Proc. 12th Int. Conf. on Concurrency Theory*, volume 2154 of *Lecture Notes in Computer Science*, pages 519–535, 2001.
40. O. Kupferman, N. Piterman, and M.Y. Vardi. Model checking linear properties of prefix-recognizable systems. In *Proc. 14th Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2002.
41. O. Kupferman and M.Y. Vardi. On the complexity of branching modular model checking. In *Proc. 6th Int. Conf. on Concurrency Theory*, volume 962 of *Lecture Notes in Computer Science*, pages 408–422. Springer, 1995.
42. O. Kupferman and M.Y. Vardi. Verification of fair transition systems. In *Proc. 8th Int. Conf. on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 372–382. Springer, 1996.
43. O. Kupferman and M.Y. Vardi. An automata-theoretic approach to reasoning about infinite-state systems. In *Proc. 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 36–52. Springer, 2000.
44. O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429, 2001.
45. O. Kupferman and M.Y. Vardi. Safraless decision procedures. In *Proc. 46th IEEE Symp. on Foundations of Computer Science*, pages 531–540, 2005.
46. O. Kupferman, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. *Journal of the ACM*, 47(2):312–360, 2000.
47. R.P. Kurshan. Complementing deterministic Büchi automata in polynomial time. *Journal of Computer and Systems Science*, 35:59–71, 1987.
48. R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
49. L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

50. C. Löding. Optimal bounds for the transformation of omega-automata. In *Proc. 19th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 1738 of *Lecture Notes in Computer Science*, pages 97–109, 1999.
51. O. Maler and L. Staiger. On syntactic congruences for ω-languages. *Theoretical Computer Science*, 183(1):93–112, 1997.
52. R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
53. A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *Proc. 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972.
54. M. Michel. Complementation is more difficult with automata on infinite words. CNET, Paris, 1988.
55. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
56. D.E. Muller, A. Saoudi, and P.E. Schupp. Alternating automata, the weak monadic theory of the tree and its complexity. In *Proc. 13th Int. Colloq. on Automata, Languages, and Programming*, volume 226 of *Lecture Notes in Computer Science*, pages 275 – 283. Springer, 1986.
57. D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. In *Automata on Infinite Words*, volume 192 of *Lecture Notes in Computer Science*, pages 100–107. Springer, 1985.
58. D.E. Muller and P.E. Schupp. Simulating alternating tree automata by nondeterministic automata: New results and new proofs of theorems of Rabin, McNaughton and Safra. *Theoretical Computer Science*, 141:69–107, 1995.
59. N. Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):5, 2007.
60. A. Pnueli and A. Zaks. On the merits of temporal testers. In *25 Years of Model Checking*, volume 5000 of *Lecture Notes in Computer Science*, pages 172–195. Springer, 2008.
61. M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
62. M.O. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 595–629. North-Holland, Amsterdam, 1977.
63. M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:115–125, 1959.
64. S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.
65. S. Safra and M.Y. Vardi. On ω-automata and temporal logic. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 127–137, 1989.
66. S. Schewe. Büchi complementation made tight. In *Proc. 26th Symp. on Theoretical Aspects of Computer Science*, volume 3 of *LIPIcs*, pages 661–672. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany, 2009.
67. S. Schewe. Tighter bounds for the determinisation of büchi automata. In *Proc. 12th Int. Conf. on Foundations of Software Science and Computation Structures*, volume 5504 of *Lecture Notes in Computer Science*, pages 167–181. Springer, 2009.
68. S. Schewe. Beyond Hyper-Minimisation—Minimising DBAs and DPAs is NP-Complete. In *Proc. 30th Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 8 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 400–411, 2010.
69. A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logic. *Journal of the ACM*, 32:733–749, 1985.
70. A.P. Sistla, M.Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with applications to temporal logic. *Theoretical Computer Science*, 49:217–237, 1987.
71. F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *Proc. 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer, 2000.
72. Synopsys. Assertion-based verification. 2003.

73. R.E. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal of Computing*, 1(2):146–160, 1972.
74. W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 133–191, 1990.
75. A. Valmari. A stubborn attack on state explosion. *Formal Methods in System Design*, 1:297–322, 1992.
76. M.Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and Systems Science*, 32(2):182–221, 1986.
77. M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.
78. B. Willems and P. Wolper. Partial-order methods for model checking: From linear time to branching time. In *Proc. 11th IEEE Symp. on Logic in Computer Science*, pages 294–303, 1996.
79. 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.
80. Q. Yan. Lower bounds for complementation of ω-automata via the full automata technique. In *Proc. 33rd Int. Colloq. on Automata, Languages, and Programming*, volume 4052 of *Lecture Notes in Computer Science*, pages 589–600. Springer, 2006.
81. W. Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.