Automata on Infinite Words and Their Applications in Formal Verification

Orna Kupferman*

July 31, 2017

Abstract

In *formal verification*, we check the correctness of a system with respect to a desired property by checking whether a mathematical model of the system satisfies a specification that formally expresses the property. In the *automata-theoretic approach* to formal verification, we model both the system and the specification by automata. Questions about systems and their specifications are then reduced to questions about automata. The goal of this course is to teach the basics of automata on infinite words and their applications in formal verification.

Introduction

Finite automata on infinite objects were first introduced in the 60's, and were the key to the solution of several fundamental decision problems in mathematics and logic [Büc62, McN66, Rab69]. Today, automata on infinite objects are used for specification and verification of nonterminating programs [Kur94, VW94]. The idea is simple: when a system is defined with respect to a finite set P 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^P, and the system itself induces a language of infinite words over this alphabet. This language can be defined by an automaton. Similarly, a specification for a system, which describes all the allowed computations, can be viewed as a language of infinite words over 2^P, and can therefore be defined by an automaton. In the automata-theoretic approach to verification, we reduce questions about systems 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 nonemptiness and language containment. 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

*School of Computer Science and Engineering, Hebrew University,Jerusalem 91904, Israel

problems. We will define automata on infinite words, study some of their properties, and see how they are used in formal verification.

1 The temporal logic LTL

The logic *LTL* is a linear temporal logic [Pnu77]. 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.

We define the semantics of LTL with respect to an infinite *computation* $\pi = \sigma_0, \sigma_1, \sigma_2, \ldots$, where for every $j \geq 0$, the set $\sigma_j \subseteq AP$ is the set of atomic propositions that hold in the j-th position of π. 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 $\pi \models$ **true** and $\pi \not\models$ **false**.
- For an atomic proposition $p \in AP$, we have that $\pi \models p$ iff $p \in \sigma_0$.
- $\pi \models \neg\psi_1$ 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 $\pi^1 \models \psi_1$.
- $\pi \models \psi_1 U \psi_2$ iff there exists $k \geq 0$ such that $\pi^k \models \psi_2$ and $\pi^i \models \psi_1$ for all $0 \leq i < k$.

We denote the size of an LTL formula φ by $|\varphi|$ and we use the following abbreviations in writing formulas:

- $\vee, \rightarrow$, and $\leftrightarrow$, interpreted in the usual way.
- $F\psi =$ **true**$U\psi$ ("eventually", and the "F" comes from "Future").
- $G\psi = \neg F \neg \psi$ ("always", and the "G" comes from "Globally").

Example 1.1 We specify in LTL some properties that one may wish a mutual exclusion algorithm to satisfy.

- The *mutual exclusion* property states that two processes are never simultaneously in their critical sections. If c_i is an atomic proposition that hold when process i is in its critical section, then the LTL formula $\psi_{me}^{i,j} = G(\neg c_i \vee \neg c_j)$ expresses mutual exclusion between processes i and j. Note that the formula $\neg F(c_i \wedge c_j)$ is equivalent to $\psi_{me}^{i,j}$.

- The *finite waiting* property for process i states that if process i tries to access its critical section, it will eventually access it. If for each process i, the atomic proposition t_i holds when process i tries to enter the critical section, then the LTL formula $\psi^i_{fw} = G(t_i \rightarrow Fc_i)$ expresses finite waiting for process i. Note that the semantics of the operator U (and therefore also the one of F) includes the present in the future. Thus, the requirement to have c_i eventually is satisfied if the process i is already in the critical section when it tries. A different specification is $G(t_i \rightarrow XFc_i)$, in which the access of the critical section has to be in the strict future.

We interpret LTL formulas also with respect to *Kripke structures*, which may generate many computations. Formally, a Kripke structure is $K = \langle AP, W, R, W_0, L\rangle$, where W is a set of states, $R \subseteq W \times W$ is a total transition relation (that is, for every $w \in W$, there is at least one w' such that $R(w, w')$, the set $W_0 \subseteq W$ is a set of initial states, and $L : W \rightarrow 2^{AP}$ maps each state to the sets of atomic propositions that hold in it. A path of K is an infinite sequence $w_0, w_1, \ldots$ such that $w_0 \in W_0$ and for all $i \geq 0$ we have $R(w_i, w_{i+1})$. Every path $w_0, w_1, \ldots$ of K induces the computation $L(w_0), L(w_1), \ldots$ of K.

The *model-checking problem* for LTL is to determine, given an LTL formula ψ and a Kripke structure K, whether all the computations of K satisfy ψ [CE81, QS82, LP85, VW94].

2 Büchi word automata

We can view Kripke structures as generators of languages over the alphabet 2^{AP}. We can also view properties as descriptions of languages over this alphabet.

Example 2.1 The properties specified by LTL formulas in Example 1.1, corresponds to the following languages over the alphabet 2^{AP}.

- The language $L^{i,j}_{me}$ that corresponds to mutual exclusion contains all computations having no occurrences of letters containing both c_i and c_j. Formally,

$$L^{i,j}_{me} = \{\sigma_0 \cdot \sigma_1 \cdots : \text{ for all } l \geq 0, \text{ we have } c_i \notin \sigma_l \vee c_j \notin \sigma_l\}.$$

- The language L^i_{fw} that corresponds to finite waiting for process i contains all computations in which every occurrence of a letter containing t_i is followed by an occurrence of a letter containing c_i.

$$L^i_{fw} = \{\sigma_0 \cdot \sigma_1 \cdots : \text{ for all } l \geq 0, \text{ if } t_i \in \sigma_l, \text{ then there is } k \geq l \text{ with } c_i \in \sigma_k)\}.$$

We describe and reason about languages of infinite words using automata on infinite words. Let Σ be a finite alphabet. A *Büchi word automaton* is $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha\rangle$, where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $Q_0 \subseteq Q$ is a set of initial states, and

$\alpha \subseteq Q$ is a set of accepting states. Since $\mathcal{A}$ may have several initial states and since the transition function may specify many possible transitions for each state and letter, $\mathcal{A}$ may be *nondeterministic*. If $|Q_0| = 1$ and δ is such that for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| \leq 1$, then $\mathcal{A}$ is a *deterministic* automaton. We use NBW and DBW as abbreviations for nondeterministic Büchi word automata and deterministic Büchi word automata, respectively.

Given an input word $w = \sigma_0 \cdot \sigma_1 \cdots$ in Σ^ω, a *run* of $\mathcal{A}$ on w is a function $r : \mathbb{N} \to Q$ where $r(0) \in Q_0$ and for every $i \geq 0$, we have $r(i+1) \in \delta(r(i), \sigma_i)$; i.e., the run starts in one of the initial states and obeys the transition function. Note that a nondeterministic automaton can have many runs on w. In contrast, a deterministic automaton has a single run on w. For a run r, let $inf(r)$ denote the set of states that r visits infinitely often. That is,

$$inf(r) = \{q \in Q \ : r(i) = q \text{ for infinitely many } i \geq 0\}.$$

As Q is finite, it is guaranteed that $inf(r) \neq \emptyset$. The run r is *accepting* iff $inf(r) \cap \alpha \neq \emptyset$. That is, iff there exists a state in α that r visits infinitely often. A run that is not accepting is *rejecting*. An automaton $\mathcal{A}$ accepts an input word w iff there exists an accepting run of $\mathcal{A}$ on w. The *language* of $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of words that $\mathcal{A}$ accepts.

Example 2.2 In Figure 1 we describe two Büchi automata. The alphabet of both automata is $\{a, b\}$. The automaton $\mathcal{A}_1$, which is deterministic, accepts exactly all words with infinitely many a's. The automaton $\mathcal{A}_2$ complements it and accepts exactly all words with finitely many a's.

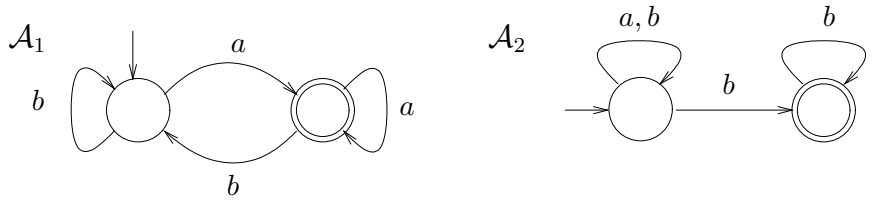

Figure 1: An example of nondeterministic Büchi automata.

The Büchi acceptance condition suggests one way to refer to $inf(r)$ in order to determine whether the run r is accepting. More acceptance conditions are defined in the literature. Below we define a generalization of the Büchi condition. A *generalized Büchi automaton* is $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, where Σ, Q, δ, and Q_0 are as in Büchi automata, and the acceptance condition $\alpha \subseteq 2^Q$ consists of sets $\alpha_i \subseteq Q$. A run r of $\mathcal{A}$ with $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ is accepting iff $inf(r) \cap \alpha_i \neq \emptyset$ for all $1 \leq i \leq k$. That is, r is accepting if every set in α is visited infinitely often. We refer to k as the *index* of the generalized Büchi automaton.

In Question 4, you will prove that generalized Büchi automata are not more expressive than Büchi automata: given a nondeterministic generalized Büchi

automaton with n states and index k, it is possible to construct an equivalent Büchi automaton with nk states.

3 Properties of Büchi Automata

It is easy to see that büchi automata are closed under union. Below we prove closure under intersection.

Theorem 3.1 [Cho74] *Given Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$, we can construct a Büchi automaton $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$.*

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

- $Q = Q_1 \times Q_2 \times \{1, 2\}$,
- $Q^0 = Q_1^0 \times Q_2^0 \times \{1\}$,
- $\delta(\langle s, t, i \rangle, a) = \delta_i(s, a) \times \delta_2(t, a) \times \{j\}$, where $i = j$ unless $i = 1$ and $s \in \alpha_1$, in which case $j = 2$, or $i = 2$ and $t \in \alpha_2$, in which case $j = 1$, and
- $\alpha = \alpha_1 \times Q_2 \times \{1\}$.

The automaton $\mathcal{A}$ has two copies of the product of $\mathcal{A}_1$ and $\mathcal{A}_2$. When executing a run, if we are in the first copy (the second copy), and we visit an accepting state of $\mathcal{A}_1$ ($\mathcal{A}_2$), we move to the other copy. The accepting condition requires to go infinitely often through $\alpha_1 \times Q_2 \times \{1\}$, i.e., to visit infinitely often accepting states of $\mathcal{A}_1$ in the first copy. Since we can return to the first copy only after having visited an accepting state of $\mathcal{A}_2$ on the second copy, both copies visit accepting states infinitely often. □

We now turn to study the expressive power of nondeterministic vs. deterministic Büchi automata. Recall that in the case of finite words, automata can be determinized, thus nondeterministic automata on finite words are not more expressive than deterministic automata on finite words. We now show that this is not the case for the infinite-word setting.

Theorem 3.2 [Lan69] *Deterministic Büchi automata are strictly less expressive than nondeterministic Büchi automata.*

Proof: Consider the language $L = (a + b)^* b^\omega$, (i.e., L consists of all infinite words in which a occurs only finitely many times). As shown in Example 2.2, the language L is recognizable by a nondeterministic Büchi automata. We now show that L is not recognizable by a deterministic Büchi automaton. Assume by way of contradiction that $L = \mathcal{L}(A)$, for a deterministic $\mathcal{A} = (\{a, b\}, Q, \{q_0\}, \delta, \alpha)$. 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 accepted by $\mathcal{A}$, so $\mathcal{A}$ has an accepting run on w_0. Thus, w_0 has a finite prefix u_0 such that $\delta(q_0, u_0) \in \alpha$.

Consider now the infinite word $w_1 = u_0 a b^\omega$. Clearly, w_1 is also accepted by $\mathcal{A}$, so $\mathcal{A}$ has an accepting run on w_1. Thus, w_1 has a finite prefix $u_0 b u_1$ such that $\delta(q_0, u_0 a u_1) \in \alpha$. In a similar way we can continue to find finite words u_i such that $\delta(q_0, u_0 a u_1 a \ldots a u_i) \in \alpha$. Since Q is finite, there are i, j, where $0 \leq i < j$, such that $\delta(q_0, u_0 a u_1 a \ldots a u_i) = \delta(q_0, u_0 a u_1 a \ldots a u_i a \ldots a u_j)$. It follows that $\mathcal{A}$ has an accepting run on

$$u_0 a u_1 a \ldots a u_i (a u_{i+1} \ldots u_{j-1} a u_j)^\omega.$$

But the latter word has infinitely many occurrences of a, so it is not in L. □

The *complement* of an NBW $\mathcal{A}$ is an NBW $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$. In the case of finite words, we complement automata by determinizing them and then dualizing the acceptance condition (that is, defining the set of accepting states to be $Q \setminus \alpha$). While it is possible to complement a DBW (but not by dualization, see Question 5), Theorem 3.2 implies we cannot use such a complementation as an intermediate step in general NBW complementation. We will get back to the complementation problem for NBW in Section 7.

4 From LTL to NBW

Given an LTL formula ψ, we construct a generalized Büchi word automaton $\mathcal{A}_\psi$ such that $\mathcal{A}_\psi$ accepts exactly all the computations that satisfy ψ. The construction was first suggested by Vardi and Wolper in 1986.

For an LTL formula ψ, the *closure* of ψ, denoted $cl(\psi)$, is the set of ψ's subformulas and their negation ($\neg\neg\psi$ is identified with ψ). Formally, $cl(\psi)$ is the smallest set of formulas that satisfy the following.

- $\psi \in cl(\psi)$.
- If $\psi_1 \in cl(\psi)$ then $\neg\psi_1 \in cl(\psi)$.
- If $\neg\psi_1 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$.
- If $\psi_1 \wedge \psi_2 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$ and $\psi_2 \in cl(\psi)$.
- If $X\psi_1 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$.
- If $\psi_1 U \psi_2 \in cl(\psi)$ then $\psi_1 \in cl(\psi)$ and $\psi_2 \in cl(\psi)$.

For example, $cl(p \wedge ((Xp)Uq))$ is

$$\{p \wedge ((Xp)Uq), \neg(p \wedge ((Xp)Uq)), p, \neg p, (Xp)Uq, \neg((Xp)Uq), Xp, \neg Xp, q, \neg q\}.$$

Theorem 4.1 [VW94] *Given an LTL formula ψ, we can construct an NBW $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi)$ is exactly the set of words satisfying ψ and the size of $\mathcal{A}_\psi$ is exponential in the length of ψ.*

Proof: We define $\mathcal{A}_\psi = \langle 2^{AP}, Q, \delta, Q_0, \alpha \rangle$, where

- We say that a set $S \subseteq cl(\psi)$ is *good in* $cl(\psi)$ if S is a maximal set of formulas in $cl(\psi)$ that does not have propositional inconsistency. Thus, S satisfies the following conditions.

  1. For all $\psi_1 \in cl(\psi)$, we have $\psi_1 \in S$ iff $\neg\psi_1 \notin S$, and
  2. For all $\psi_1 \wedge \psi_2 \in cl(\psi)$, we have $\psi_1 \wedge \psi_2 \in S$ iff $\psi_1 \in S$ and $\psi_2 \in S$.

  The state space $Q \subseteq 2^{cl(\psi)}$ is the set of all the good sets in $cl(\psi)$.

- Let S and S' be two good sets in $cl(\psi)$, 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 $X\psi_1 \in cl(\psi)$, we have $X\psi_1 \in S$ iff $\psi_1 \in S'$, and
  3. For all $\psi_1 U \psi_2 \in cl(\psi)$, 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 $\neg(\psi_1 U \psi_2) \in cl(\psi)$, 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 α the set

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

□

5 Alternating Automata on Infinite Words

In Section 2 we defined Büchi automata and mentioned that automata on infinite words can be classified according to the type of acceptance condition. Another way to classify an automaton on infinite words is by the type of its branching mode. In a *deterministic* automaton, the transition function δ maps a pair of a state and a letter into a single state. The intuition is that when the automaton is in state q and it reads a letter σ, then the automaton moves to state $\delta(q, \sigma)$, from which it should accept the suffix of the word. When the branching mode is existential, the automaton is nondeterministic and δ maps q and σ into a set of states. In the existential mode, the automaton should accept the suffix of the word from one of the states in the set. Accordingly, a word is accepted by a nondeterministic automaton if the automaton has some accepting run on it. We have met deterministic and nondeterministic automata in Section 2. In this section we meet more sophisticated branching modes. The *universal* branching mode is dual to the existential one. Thus, as there, δ maps q and σ into a set of states, yet the automaton should accept the suffix of the word from all of

the states in the set. Accordingly, a word is accepted by a universal automaton if all the runs of the automaton on the word are accepting. In *alternating* automata [CKS81], both existential and universal modes are allowed, and the transitions are given as Boolean formulas over the set of states. For example, $\delta(q, \sigma) = q_1 \vee (q_2 \wedge q_3)$ means that the automaton should accept the suffix of the word either from state q_1 or from both states q_2 and q_3. We now define alternating automata formally and see how they can serve as an intermediate step in the translation of LTL to nondeterministic Büchi automata.

For a given set X, let $\mathcal{B}^+(X)$ 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 $\theta \in \mathcal{B}^+(X)$ iff the truth assignment that assigns *true* to the members of Y and assigns *false* to the members of $X \setminus Y$ satisfies θ. For example, the sets $\{q_1, q_3\}$ and $\{q_2, q_3\}$ both satisfy the formula $(q_1 \vee q_2) \wedge q_3$, while the set $\{q_1, q_2\}$ does not satisfy this formula.

Consider an automaton $\mathcal{A} = \langle \Sigma, Q, Q_0, \delta, \alpha \rangle$. For a word $w = \sigma_0 \cdot \sigma_1 \cdots$ and an index $i \geq 0$, let $w^i = \sigma_i \cdot \sigma_{i+1} \cdots$ be the suffix of w that starts in position i. We can represent δ using $\mathcal{B}^+(Q)$. For example, a transition $\delta(q, \sigma) = \{q_1, q_2, q_3\}$ of a nondeterministic automaton $\mathcal{A}$ can be written as $\delta(q, \sigma) = q_1 \vee q_2 \vee q_3$. If $\mathcal{A}$ is universal, 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 $\mathcal{B}^+(Q)$. 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 suffix w^i of w from state q, if it accepts w^{i+1} 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 $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$, where Σ, Q, q_{in}, and α are as in nondeterministic automata (for technical simplicity we assume that the set of initial states is a singleton), and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function. While a run of a nondeterministic automaton is an infinite sequence of states, a run of an alternating automaton is a tree $r : T_r \to Q$ for some $T_r \subseteq \mathbb{N}^*$. Formally, a tree is a (finite or infinite) nonempty prefix-closed set $T \subseteq \mathbb{N}^*$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c \in T$ where $c \in \mathbb{N}$ are the *children* of x. A node with no children is a *leaf*. 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 $\epsilon \in \pi$ 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 a finite set 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. A run of $\mathcal{A}$ on an infinite word $w = \sigma_0 \cdot \sigma_1 \cdots$ is a Q-labeled tree $\langle T_r, r \rangle$ such that the following hold:

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

For example, if $\delta(q_{in}, \sigma_0) = (q_1 \vee q_2) \wedge (q_3 \vee q_4)$, then possible runs of $\mathcal{A}$ on w have a root labeled q_{in}, 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 need not have children. This is the reason why T_r may have leaves. Also, since there exists no set S as required 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 $\mathcal{A}$ 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 $\mathcal{A}$, denoted $\mathcal{L}(\mathcal{A})$, is the set of infinite words that $\mathcal{A}$ accepts. We use ABW to abbreviate alternating Büchi automata on infinite words.

6 From LTL to NBW via alternating Büchi automata

In this section we show an alternative translation of LTL to NBW. The translation goes via alternating automata: we first translate the LTL formula to an ABW, and then translate the ABW to an an NBW. Using alternating automata as an intermediate step was first suggested in the branching framework, for the translation of branching temporal logics to tree automata [KVW00]. There, the use of alternating automata enables an efficient automata-based solution to the model checking problem. In the linear framework, an advantage of the intermediate alternating automaton is the ability to apply optimization algorithms on both the intermediate ABW and the final NBW [Fri03, FW02, GKSV03].

For simplicity, we assume that LTL formulas are in positive normal form, where negation is applied only to atomic propositions. Having no negation, we should have both $\wedge$ and $\vee$, and have the temporal operator G in addition to X and U.

Theorem 6.1 [KVW00] *Given an LTL formula ψ in positive normal form, we can construct an ABW $\mathcal{A}_\psi$ such that $\mathcal{L}(\mathcal{A}_\psi)$ is exactly the set of words satisfying ψ and the size of $\mathcal{A}_\psi$ is linear in the length of ψ.*

Proof: We define $\mathcal{A}_\psi = \langle 2^{AP}, cl(\psi), \delta, \psi, \alpha \rangle$, as follows. The set α of accepting states consists of all the G-formulas in $cl(\psi)$; that is, formulas of the form $G\varphi_2$. The transition function δ is defined, for all $\sigma \in 2^{AP}$, as follows.

- $\delta(p, \sigma) = \mathbf{true}$ if $p \in \sigma$.
- $\delta(p, \sigma) = \mathbf{false}$ if $p \notin \sigma$.
- $\delta(\neg p, \sigma) = \mathbf{true}$ if $p \notin \sigma$.
- $\delta(\neg p, \sigma) = \mathbf{false}$ if $p \in \sigma$.

- $\delta(\varphi_1 \wedge \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \wedge \delta(\varphi_2, \sigma)$.
- $\delta(\varphi_1 \vee \varphi_2, \sigma) = \delta(\varphi_1, \sigma) \vee \delta(\varphi_2, \sigma)$.
- $\delta(X\varphi_2, \sigma) = \varphi_2$.
- $\delta(\varphi_1 U \varphi_2, \sigma) = \delta(\varphi_2, \sigma) \vee (\delta(\varphi_1, \sigma) \wedge \varphi_1 U \varphi_2)$.
- $\delta(G\varphi_2, \sigma) = \delta(\varphi_2, \sigma) \wedge G\varphi_2$.

Intuitively, $\mathcal{A}_\psi$ follows the structure of the formula, and uses the acceptance condition α to guarantee that eventualities of U-formulas are eventually satisfied. □

In order to complete the translation of LTL to NBW, we need to remove alternation from $\mathcal{A}_\psi$:

Theorem 6.2 [MH84] *Let $\mathcal{A}$ be an alternating Büchi automaton. There is a nondeterministic Büchi automaton $\mathcal{A}'$, with exponentially many states, such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$.*

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

$$\{\langle S', O' \setminus \alpha \rangle \mid S' \text{ satisfies } \bigwedge_{q \in S} \delta(q, \sigma), O' \subseteq S', \text{ and } O' \text{ satisfies } \bigwedge_{q \in O} \delta(q, \sigma)\}.$$

- If $O = \emptyset$, then $\delta'(\langle S, O \rangle, \sigma) =$

$$\{\langle S', S' \setminus \alpha \rangle \mid S' \text{ satisfies } \bigwedge_{q \in S} \delta(q, \sigma)\}.$$

□

In [MSS86], Muller et al. introduce *alternating weak automata*. In a weak automaton, the acceptance condition is $\alpha \subseteq Q$ and there exists a partition of Q into disjoint sets, Q_i, such that for each set Q_i, either $Q_i \subseteq \alpha$, in which case Q_i is an *accepting set*, or $Q_i \cap \alpha = \emptyset$, in which case Q_i is a *rejecting set*. In addition, there exists a partial order $\leq$ on the collection of the Q_i's such that for every $q \in Q_i$ and $q' \in Q_j$ for which q' occurs in $\delta(q, \sigma, k)$, for some $\sigma \in \Sigma$ and $k \in \mathcal{D}$, we have $Q_j \leq Q_i$. Thus, transitions from a state in Q_i lead to states in either

the same Q_i or a lower one. It follows that every infinite path of a run of a weak automaton ultimately gets "trapped" within some Q_i. The path then satisfies the acceptance condition if and only if Q_i is an accepting set. Note that this corresponds to the Büchi acceptance condition. Indeed, a run visits infinitely many states in α iff it gets trapped in an accepting set.

The automaton A_ψ defined in the proof of Theorem 6.1 is weak. To see this, consider the partition of Q into disjoint sets in which each formula $\varphi \in cl(\psi)$ constitutes a (singleton) set $\{\varphi\}$ in the partition. The partial order between the sets is then defined by $\{\varphi_1\} \leq \{\varphi_2\}$ iff $\varphi_1 \in cl(\varphi_2)$. Since each transition of the automaton from a state φ leads to states associated with formulas in $cl(\varphi)$, the weakness conditions hold. In particular, each set is either contained in α or disjoint from α.

As pointed out in [GO01], the fact $\mathcal{A}_\psi$ is weak (in fact, it is *very weak* – the sets Q_i in the partition are singletons) enables a simpler removal of alternation than the one described in Theorem 6.2. In general, the construction we presented in Theorem 4.1 and the one that follows from the combination of Theorems 6.1 and 6.2 are very basic ones. Due to the heavy use of the construction in practice, numerous improvements have been suggested, cf. [GPVW95, SB00, GO01, Fri03].

7 Complementation of Büchi automata

In Section 3, we saw that NBW are closed under union and intersection. In this section we prove their closure under complementation.

The complementation problem for nondeterministic word automata has numerous applications in formal verification. In particular, the language-containment problem, to which many verification problems is reduced, involves complementation. For automata on finite words, which correspond to safety properties, complementation involves determinization. The 2^n blow-up that is caused by the subset construction is justified by a tight lower bound. For Büchi automata on infinite words, which are required for the modeling of liveness properties, optimal complementation constructions are quite complicated, as the subset construction is not sufficient. Efforts to develop simple complementation constructions for nondeterministic automata started early in the 60s, motivated by decision problems of second-order logics. Büchi suggested a complementation construction for nondeterministic Büchi automata that involved a complicated combinatorial argument and a doubly-exponential blow-up in the state space [Büc62]. Thus, complementing an automaton with n states resulted in an automaton with $2^{2^{O(n)}}$ states. In [SVW87], Sistla et al. suggested an improved construction, with only $2^{O(n^2)}$ states, which is still, however, not optimal. Only in [Saf88], Safra introduced a determinization construction, which also enabled a $2^{O(n \log n)}$ complementation construction, matching a lower bound described by Michel [Mic88].

In this section we describe a complementation construction that avoids Safra's determinization. The construction, described in [KV01], uses instead in-

termediate universal co-Büchi automata and alternating weak automata. Here, we describe the construction without the intermediate automata, and go directly to a complementary NBW. The idea behind the construction is to assign ranks to nodes in a directed acyclic graph that embodies all the runs of the NBW. The idea can be applied also to richer types of acceptance conditions [KV05].

Let $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$ be a nondeterministic Büchi automaton with $|Q| = n$, and let $w = \sigma_0 \cdot \sigma_1 \cdot$ be a word in Σ^ω. We define an infinite DAG G that embodies all the possible runs of $\mathcal{A}$ 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 $Q_0 = \{q_{in}\}$ and $Q_{l+1} = \bigcup_{q \in Q_l} \delta(q, \sigma_l)$.
- $E \subseteq \bigcup_{l \geq 0}(Q_l \times \{l\}) \times (Q_{l+1} \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff $q' \in \delta(q, \sigma_l)$.

We refer to G as the *run* DAG of $\mathcal{A}$ 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 $\langle q', l' \rangle = \langle q_i, l_i \rangle$. Finally, we say that a vertex $\langle q, l \rangle$ is an *α-vertex* iff $q \in \alpha$. It is easy to see that $\mathcal{A}$ accepts w iff G has a path with infinitely many α-vertices. Indeed, such a path corresponds to an accepting run of $\mathcal{A}$ on w.

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 decreased 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 $f(\langle q_{j+i}, j+i \rangle) = f(\langle q_j, j \rangle)$.

Lemma 7.1 *$\mathcal{A}$ rejects w iff there is an odd ranking for G.*

Proof: We first claim that if there is an odd ranking for G, then $\mathcal{A}$ 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 $\mathcal{A}$ on w, visit α only finitely often.

Assume now that $\mathcal{A}$ rejects w. We describe an odd ranking for G. We say that a vertex $\langle q, l \rangle$ is *finite* in a (possibly finite) 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.

- $G_0 = G$.
- $G_{2i+1} = G_{2i} \setminus \{\langle q, l\rangle \mid \langle q, l\rangle \text{ is finite in } G_{2i}\}$.
- $G_{2i+2} = G_{2i+1} \setminus \{\langle q, l\rangle \mid \langle q, l\rangle \text{ is } \alpha\text{-free in } G_{2i+1}\}$.

Consider the function $f : V \to \mathbb{N}$ where

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

Recall that $\mathcal{A}$ rejects w. Thus, each path in G has only finitely many α-vertices. It is shown in [KV01] that for every $i \geq 0$, the transition from G_{2i+1} to G_{2i+2} involves the removal of an infinite path from G_{2i+1}. Intuitively, it follows from the fact that as long as G_{2i+1} is not empty, it contains a at least one α-free vertex, from which an infinite path of α-free vertices start. Since the width of G_0 is bounded by n, it follows that the width of G_{2i} is at most $n - i$. Hence, G_{2n} is finite, and G_{2n+1} is empty. Thus, f above maps the vertices in V to $[2n]$. We claim further that f is an odd ranking. First, since an α-free vertex cannot be an α-vertex and $f(\langle q, l\rangle)$ is odd only for α-free vertices $\langle q, l\rangle$, the first condition for f being a ranking holds. Second, as argued in [KV01], for every two vertices $\langle q, l\rangle$ and $\langle q', l'\rangle$ in G, if $\langle q', l'\rangle$ is reachable from $\langle q, l\rangle$, then $f(\langle q', l'\rangle) \leq f(\langle q, l\rangle)$. In particular, this holds for $\langle q', l'\rangle$ that is a successor of $\langle q, l\rangle$. Hence, the second condition for ranking holds too. Finally, as argued in [KV01] for every infinite path in G, there exists a vertex $\langle q, l\rangle$ with an odd rank such that all the vertices $\langle q', l'\rangle$ in the path that are reachable from $\langle q, l\rangle$ have $f(\langle q', l'\rangle) = f(\langle q, l\rangle)$. Hence, f is an odd ranking. □

By Lemma 7.1, an automaton $\mathcal{A}'$ that complements $\mathcal{A}$ can proceed on an input word w by guessing an odd ranking for the run DAG of $\mathcal{A}$ on w. We now define such an automaton $\mathcal{A}'$ formally. We first need some definitions and notations.

A *level ranking* for $\mathcal{A}$ and w is a function $g : Q \to [2n] \cup \{\bot\}$, such that if $g(q)$ is odd, then $q \notin \alpha$. Let $\mathcal{R}$ be the set of all level rankings. 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 $0 \leq g'(q') \leq g(q)$.

We define $\mathcal{A}' = \langle \Sigma, \mathcal{R} \times 2^Q, q'_{in}, \delta', \mathcal{R} \times \{\emptyset\}\rangle$, where

- $q'_{in} = \langle g_{in}, \emptyset\rangle$, where $g_{in}(q_{in}) = 2n$ and $g_{in}(q) = \bot$ for all $q \neq q_{in}$. Thus, the odd ranking that $\mathcal{A}'$ guesses maps the root $\langle q_{in}, 0\rangle$ of the run DAG to $2n$.
- For a state $\langle g, P\rangle \in \mathcal{R} \times 2^Q$ and a letter $\sigma \in \Sigma$, we define $\delta'(\langle g, P\rangle, \sigma)$ as follows.
  - If $P \neq \emptyset$, then

$$\delta'(\langle g, P\rangle, \sigma) = \{ \langle g', P'\rangle : \ g' \text{ covers } g, \text{ and } P' = \{q' : \text{ there is } q \in P \text{ such that } q' \in \delta(q, \sigma) \text{ and } g'(q') \text{ is even}\}\}.$$

- If $P = \emptyset$, then

$$\delta'(\langle g, P\rangle, \sigma) = \{\langle g', P'\rangle : g' \text{ covers } g, \text{ and } P' = \{q' : g'(q') \text{ is even}\}\}.$$

Thus, when $\mathcal{A}'$ 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, $\mathcal{A}'$ 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, and is initiated by new obligations for visits in odd ranks according to the current level ranking. The acceptance condition $\mathcal{R} \times \{\emptyset\}$ then checks that there are infinitely many levels in which all the obligations have been fulfilled.

8 Exercises

Question 1

For each pair $\varphi_1; \varphi_2$ of formulas below, decide which of the following hold (note that possibly both a and b hold, or none of them):

a. $\varphi_1 \rightarrow \varphi_2$.

b. $\varphi_1 \leftarrow \varphi_2$.

When the an implication does not hold, describe a counter example (when a does not hold, describe a model for φ_1 that does not satisfy φ_2, and when b does not hold, describe a model for φ_2 that does not satisfy φ_1).

1.	Gp	;	$\neg FX\neg p$
2.	$G(p \vee q)$	;	$Gp \vee Gq$
3.	$G(p \wedge q)$	;	$Gp \wedge Gq$
4.	qUp	;	$q \wedge XqUp$
5.	$pU(qUr)$	;	$(pUq)Ur$
6.	$p \wedge Xq$	;	$pUq \wedge qUp$

Question 2

Describe nondeterministic Büchi automata for the following properties.

1. $F(p \wedge Xp)$.
2. $FG(p \vee Xp)$.
3. $Gp \rightarrow Gq$.
4. $GFp \rightarrow GFq$.

Question 3

Prove or give a counter example:

1. Every nondeterministic Büchi automaton $\mathcal{A}$ has an equivalent nondeterministic Büchi automaton $\mathcal{A}'$ with a single initial state.
2. Every nondeterministic Büchi automaton $\mathcal{A}$ has an equivalent nondeterministic Büchi automaton $\mathcal{A}'$ with a single accepting state.

Question 4

Given a generalized Büchi automaton with n states and index k, construct an equivalent Büchi automaton with nk states.

Question 5

Given a deterministic Büchi automaton $\mathcal{A} = \langle \Sigma, Q, \delta, q_0, \alpha \rangle$ with n states, describe a nondeterministic Büchi automaton $\mathcal{A}'$ with $O(n)$ states such that $\mathcal{L}(\mathcal{A}') = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.

Hint: the NBW $\mathcal{A}'$ uses its nondeterminism in order to guess when the run of $\mathcal{A}$ stops visiting α.

Question 6

Consider a nondeterministic word automaton $\mathcal{A}$. Let $\mathcal{L}_*(\mathcal{A})$ be the language of $\mathcal{A}$ when regarded as an automaton on finite words, and let $\mathcal{L}_\omega(\mathcal{A})$ be the language of $\mathcal{A}$ when regarded as a Büchi automaton. Prove or give a counter example:

1. $\mathcal{L}_\omega(\mathcal{A}) = lim(\mathcal{L}_*(\mathcal{A}))$.
2. $\mathcal{L}_\omega(\mathcal{A}) = lim(\mathcal{L}_*(\mathcal{A}))$ iff there is some deterministic Büchi automaton that recognizes $\mathcal{L}_\omega(\mathcal{A})$.

Question 7

In a *co-Büchi* word automaton, the acceptance condition is a set $\alpha \subseteq Q$ and a run r is accepting iff it visits α only finitely often; that is $inf(r) \cap \alpha = \emptyset$.

1. Prove that a language L is recognizable by a deterministic Büchi automaton iff $\Sigma^\omega \setminus L$ is recognizable by a deterministic co-Büchi automaton.
2. Given a nondeterministic co-Büchi automaton $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, define a deterministic co-Büchi automaton $\mathcal{A}' = \langle \Sigma, Q', \delta', q_0', \alpha' \rangle$ equivalent to $\mathcal{A}$.

   Hint: $Q' = 2^Q \times 2^Q$, and all the reachable states $\langle S, P \rangle$ in Q' are such that $P \subseteq S$. The acceptance condition $\alpha' = 2^Q \times \{\emptyset\}$.
3. Let $L = \{w : w \text{ has infinitely many } a\text{'s } \} \subseteq \{a, b\}^\omega$. Can L be recognized by a nondeterministic co-Büchi word automaton? Justify your answer.

Question 8

In this question we prove an exponential lower bound on the translation of LTL to nondeterministic Büchi automata. Let $AP = \{p, q\}$. For $n \geq 1$, we define the language $\mathcal{L}_n$ over the alphabet 2^{AP} as follows.

$$\mathcal{L}_n = \{\{\{p\},\{q\},\emptyset,\}^* \cdot \emptyset \cdot w \cdot \emptyset \cdot \{\{p\},\{q\},\emptyset,\}^* \cdot \{p,q\} \cdot w \cdot \{p,q\}^\omega : w \in \{\{p\},\{q\}\}^n\}.$$

Thus, a word is in $\mathcal{L}_n$ iff the word between the first and second $\{p, q\}$ is of length n, it is composed of letters in $\{\{p\},\{q\}\}$ only, and it has appeared between $\emptyset$'s somewhere before the first $\{p, q\}$. In addition, the second $\{p, q\}$ starts an infinite tail of $\{p, q\}$'s.

1. Describe two words in $\mathcal{L}_3$ and two words not in $\mathcal{L}_3$.
2. Prove that the smallest nondeterministic Büchi automaton that recognizes $\mathcal{L}_n$ has at least 2^n states.
3. Specify $\mathcal{L}_n$ with an LTL formula of length quadratic in n.

Question 9

Consider the translation of an LTL formula ψ to nondeterministic Büchi automata $\mathcal{A}_\psi$ we saw in class. Recall that $\mathcal{A}_\psi^S$ (that is, $\mathcal{A}_\psi$ with initial state $S \subseteq cl(\psi)$) accepts exactly these words in $(2^{AP})^\omega$ that satisfy exactly all the formulas in S.

1. Construct the automaton for $\psi = F(p \wedge Xp)$. Note you are asked to construct exactly the automaton we saw in class (which may not be the minimal automaton for ψ).
2. Describe a linear-time procedure for complementing the automaton $\mathcal{A}_\psi$ (for an arbitrary ψ).
3. Which of the following statements are correct? (prove or give a counter example)
   (a) For every LTL formula ψ, if there is a deterministic Büchi automaton for ψ, then $\mathcal{A}_\psi$ is deterministic.
   (b) For every LTL formula ψ and a word $w \in (2^{AP})^\omega$ such that $w \models \psi$, there is a single run of $\mathcal{A}_\psi$ that accepts w.

Question 10

Describe an ABW with $O(n)$ states for the language $\mathcal{L}_n = \{w \cdot w \cdot \#^\omega : w \in (0+1)^n\}$.

Question 11

For a word $w \in \Sigma^\omega$, let $\mathit{suff}(w) = \{y : x \cdot y = w, \text{ for some } x \in \Sigma^*\}$. Note that $\mathit{suff}(w)$ contains w and ϵ. For a language $L \subseteq \Sigma^\omega$, let $\mathit{suff\_limit}(L) = \{w : \mathit{suff}(w) \subseteq L\}$. Thus, $\mathit{suff\_limit}(L)$ contains exactly all words w such that all the suffixes of w are in L.

Given an NBW $\mathcal{A} = \langle \Sigma, Q, \delta, Q_0, \alpha \rangle$, describe an ABW $\mathcal{A}'$ with the same state space Q such that $\mathcal{L}(\mathcal{A}') = \mathit{suff\_limit}(\mathcal{L}(\mathcal{A}))$. Prove the correctness of the construction formally.

References

[Büc62] 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.

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

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

[CKS81] A.K. Chandra, D.C. Kozen, and L.J. Stockmeyer. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114–133, 1981.

[Fri03] 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.

[FW02] C. Fritz and T. Wilke. State space reductions for alternating Büchi automata: Quotienting by simulation equivalences. In *Proc. 22nd Conf. on Foundations of Software Technology and Theoretical Computer Science*, volume 2556 of *Lecture Notes in Computer Science*, pages 157–169, 2002.

[GKSV03] S. Gurumurthy, O. Kupferman, F. Somenzi, and M.Y. Vardi. On complementing nondeterministic Büchi automata. In *Proc. 12th Conf. on Correct Hardware Design and Verification Methods*, volume 2860 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2003.

[GO01] 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.

[GPVW95] 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.

[Kur94] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.

[KV01] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(2):408–429, 2001.

[KV05] O. Kupferman and M.Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 3440 of *Lecture Notes in Computer Science*, pages 206–221. Springer, 2005.

[KVW00] 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.

[Lan69] L.H. Landweber. Decision problems for ω–automata. *Mathematical Systems Theory*, 3:376–384, 1969.

[LP85] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 97–107, 1985.

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

[MH84] S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.

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

[MSS86] 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.

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

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 8th ACM Symp. on Principles of Programming Languages*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982.

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

[Saf88] S. Safra. On the complexity of ω-automata. In *Proc. 29th IEEE Symp. on Foundations of Computer Science*, pages 319–327, 1988.

[SB00] 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.

[SVW87] 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.

[VW94] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, 1994.