Weak Alternating Automata Are Not That Weak*

Orna Kupferman[†]
EECS Department
UC Berkeley
Berkeley CA 94720-1770, U.S.A.
Email: orna@eecs.berkeley.edu

Moshe Y. Vardi[‡]
Department of Computer Science
Rice University
Houston TX 77005-1892, U.S.A.
Email: vardi@cs.rice.edu
URL: http://www.cs.rice.edu/~vardi

Abstract *Automata on infinite words are used for specification and verification of nonterminating programs. Different types of automata induce different levels of expressive power, of succinctness, and of complexity. Alternating automata have both existential and universal branching modes and are particularly suitable for specification of programs. In a weak alternating automaton, the state space is partitioned into partially ordered sets, and the automaton can proceed from a certain set only to smaller sets. Reasoning about weak alternating automata is easier than reasoning about alternating automata with no restricted structure. Known translations of alternating automata to weak alternating automata involve determinization, and therefore involve a double-exponential blow-up. In this paper we describe a quadratic translation, which circumvents the need for determinization, of Büchi and co-Büchi alternating automata to weak alternating automata. Beyond the independent interest of such a translation, it gives rise to a simple complementation algorithm for nondeterministic Büchi automata.*

1 Introduction

Finite automata on infinite objects were first introduced in the 60's. Motivated by decision problems in mathematics and logic, Büchi, McNaughton, and Rabin developed a framework for reasoning about infinite words and infinite trees [Büc62, McN66, Rab69]. 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 logics [Tho90]. Today, automata on infinite objects are used for specification and verification of nonterminating programs. The idea is simple: when a program is defined with respect to a finite set P of propositions, each of the program's states can be associated with a set of propositions that hold in the state. Then, each of the program's computations induces an infinite word over the alphabet 2^P, and the program itself induces a language of infinite words over this alphabet. This language can be defined by an automaton. Similarly, a specification for a program, 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 programs and their specifications to questions about automata. More specifically, questions such as satisfiability of specifications and correctness of programs with respect to their specifications are

*Part of this work was done in Bell Laboratories during the DIMACS Special Year on Logic and Algorithms.

†Supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

‡Supported in part by the NSF grant CCR-9628400.

reduced to questions such as nonemptiness and language containment [VW86, Kur94, VW94]. The automata-theoretic approach separates the logical and the combinatorial aspects of reasoning about programs. The translation of specifications to automata handles the logic and shifts all the combinatorial difficulties to automata-theoretic problems.

As automata on finite words, *automata on infinite words* either accept or reject an input word. Since a run on an infinite word does not have a final state, acceptance is determined with respect to the set of states visited infinitely often during the run. There are many ways to classify an automaton on infinite words. One is the type of its acceptance condition. For example, in *Büchi automata*, some of the states are designated as accepting states, and a run is accepting iff it visits states from the accepting set infinitely often [Büc62]. Dually, in *co-Büchi automata*, a run is accepting iff it visits states from the accepting set only finitely often. More general are *Muller automata*. Here, the acceptance condition is a set α of sets of states, and a run is accepting iff the set of states visited infinitely often is a member of α [Mul63].

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* or *universal*, 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, and in the universal mode, it should accept the suffix from all the states in the set. In an *alternating* automaton [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.

It turns out that different types of automata have different *expressive power*. For example, unlike automata on finite words, where deterministic and nondeterministic (existential) automata have the same expressive power, deterministic Büchi automata are strictly less expressive than nondeterministic Büchi automata [Lan69]. That is, there exists a language L over infinite words such that L can be recognized by a nondeterministic Büchi automaton but cannot be recognized by a deterministic Büchi automaton. It also turned out that some types of automata may be more *succinct* than other types. For example, though alternating Büchi automata are as expressive as nondeterministic Büchi automata (both recognize exactly all ω-regular languages), alternation makes Büchi automata exponentially more succinct. That is, translating an alternating Büchi automaton to a nondeterministic one might involve an exponential blow-up [DH94].

Since the combinatorial structure of alternating automata is rich, translating specifications to alternating automata is much simpler than translating them to nondeterministic automata. Alternating automata enable a complete partition between the logical and the combinatorial aspects of reasoning about programs, and they give rise to cleaner and simpler verification algorithms [Var96]. The ability of alternating automata to switch between existential and universal branching modes also makes their *complementation* very easy. For example, in order to complement an alternating Muller automaton on infinite words, one only has to dualize its transition function and acceptance condition [MH84, Lin88]. In contrast, complementation is a very challenging problem for nondeterministic automata on infinite words. In particular, complementing a nondeterministic Büchi automaton involves an exponential blow-up [Saf88, Mic88].

In [MSS86], Muller et al. introduced *weak alternating automata*. In a weak alternating automaton, the automaton's set of states is partitioned into partially ordered sets. Each set is classified as accepting or rejecting. The transition function is restricted so that in each transition, the automaton either stays at the same set or moves to a set smaller in the partial order. Thus, each run of a weak alternating automaton eventually gets trapped in some set in the partition. Acceptance is then determined according to the classification of this set. The special structure of weak alternating automata is reflected in their attractive computational properties and makes them very appealing. For example, while the best known complexity for solving the membership problem for Büchi alternating automata is quadratic time, we know how to solve the membership problem for weak alternating automata in linear time [BVW94].

Weak alternating automata are a special case of Büchi alternating automata. Indeed, the condition of getting trapped in an accepting set can be replaced by a condition of visiting states of accepting sets infinitely often. The other direction, as it is easy to see, is not true. In fact, it is proven in [MSS86], that, when defined on trees, a language L can be recognized by a weak alternating automaton iff L can be recognized by both Büchi and co-Büchi alternating automata. Nevertheless, when defined on words, weak alternating automata are not less expressive than Büchi alternating automata, and they can recognize all the ω-regular languages. To prove this, [MSS86, Lin88] suggest a linear translation of deterministic Muller automata to weak alternating automata. Using, however, the constructions in [MSS86, Lin88] in order translate a nondeterministic Büchi or co-Büchi automaton $\mathcal{A}$ into a weak alternating automaton, one has no choice but to first translate $\mathcal{A}$ into a deterministic Muller automaton. Such a determinization involves an exponential blow-up [Saf88]. Even worse, if $\mathcal{A}$ is an alternating automaton, then its determinization involves a doubly-exponential blow-up, and hence, so does the translation to weak alternating automata. Can these blow-ups be avoided?

In this paper we answer this question positively. We describe a simple quadratic translation of Büchi and co-Büchi alternating automata into weak alternating automata. Beyond the independent interest of such a translation, it gives rise to a simple complementation algorithm for nondeterministic Büchi automata. The closure of nondeterministic Büchi automata under complementation plays a crucial role in solving decision problems of second order logics. As a result, many efforts have been put in proving this closure and developing simple complementation algorithms. In [Büc62], Büchi suggested a complementation construction, which indeed solved the problem, yet involved a complicated combinatorial argument and a doubly-exponential blow-up in the state space. 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 an optimal determinization construction, which also enabled a $2^{O(n \log n)}$ complementation construction, matching the known lower bound [Mic88]. Another $2^{O(n \log n)}$ construction was suggested by Klarlund in [Kla91], which circumvented the need for determinization.

While being the heart of many complexity results in verification, the optimal constructions in [Saf88, Kla91] are complicated. In particular, the intricacy of the algorithms makes their implementation difficult. We know of no implementation of Klarlund's algorithm, and the implementation of Safra's algorithm [THB95] has to cope with the involved structure of the states in the complementary automaton. The lack of a simple implementation is not due to a lack of need. Recall that in the automata-theoretic approach to verification, we check correctness of a program with respect to a specification by checking containment of the language of the program in a language of an automaton that accepts exactly all computations that satisfy the specification. In order to check the latter, we check that the intersection of the program with an automaton that complements the specification automaton is empty. Due to the lack of a simple complementation construction, verification tools have to restrict the specification automaton or improvise other solutions. For example, in the verification tool COSPAN [Kur94], the specification automaton must be deterministic (it is easy to complement deterministic automata [CDK93]). In the verification tool SPIN [Hol91], the user has to complement the automaton by himself; thus, together with the program, SPIN gets as input a nondeterministic Büchi automaton, called the Never-Claim, which accepts exactly all computations that do not satisfy the specification.

The complementing automaton constructed in our procedure here is similar to the one constructed in [Kla91], but as our construction involves alternation, it is simpler and easily implementable. Consider a nondeterministic Büchi automaton $\mathcal{B}$. We can easily complement $\mathcal{B}$ by regarding it as a universal co-Büchi automaton. Now, using our construction, we translate this complementary automaton to a weak alternating automaton $\mathcal{W}$. By [MH84], weak alternating automata can be translated to nondeterministic Büchi automata. Applying their (exponential yet simple) translation to $\mathcal{W}$, we end up with a nondeterministic Büchi automaton $\mathcal{N}$ that complements $\mathcal{B}$. For $\mathcal{B}$ with n states, the size of $\mathcal{N}$ is $2^{O(n \log n)}$, meeting the known lower bound [Mic88] and the complicated constructions suggested in [Saf88, Kla91].

2 Alternating Automata

Given an alphabet Σ, an *infinite word over* Σ is an infinite sequence $w = \sigma_0 \cdot \sigma_1 \cdot \sigma_2 \cdots$ of letters in Σ. We denote by w^l the suffix $\sigma_l \cdot \sigma_{l+1} \cdot \sigma_{l+2} \cdots$ of w. An *automaton on infinite words* is $\mathcal{A} = \langle \Sigma, Q, q_{in}, \rho, \alpha \rangle$, where Σ is the input alphabet, Q is a finite set of states, $\rho : Q \times \Sigma \rightarrow 2^Q$ is a transition function, $q_{in} \in Q$ is an initial state, and α is an acceptance condition (a condition that defines a subset of Q^ω). Intuitively, $\rho(q, \sigma)$ is the set of states that $\mathcal{A}$ can move into when it is in state q and it reads the letter σ. Since the transition function of $\mathcal{A}$ may specify many possible transitions for each state and letter, $\mathcal{A}$ is not *deterministic*. If δ is such that for every $q \in Q$ and $\sigma \in \Sigma$, we have that $|\delta(q, \sigma)| = 1$, then $\mathcal{A}$ is a deterministic automaton.

A *run* of $\mathcal{A}$ on w is a function $r : \mathbb{N} \rightarrow Q$ where $r(0) = q_{in}$ (i.e., the run starts in the initial state) and for every $l \geq 0$, we have $r(l+1) \in \delta(r(l), \sigma_l)$ (i.e., the run obeys the transition function). In automata over finite words, acceptance is defined according to the last state visited by the run. When the words are infinite, there is no such thing "last state", and acceptance is defined according to the set $Inf(r)$ of states that r visits *infinitely often*, i.e.,

$$inf(r) = \{q \in Q : \text{ for infinitely many } l \in \mathbb{N}, \text{ we have } r(l) = q\}.$$

As Q is finite, it is guaranteed that $Inf(r) \neq \emptyset$. The way we refer to $Inf(r)$ depends on the acceptance condition of $\mathcal{A}$. Several acceptance conditions are studied in the literature. We consider here two:

- *Büchi automata*, where $\alpha \subseteq Q$, and r accepts w iff $Inf(r) \cap \alpha \neq \emptyset$.
- *co-Büchi automata*, where $\alpha \subseteq Q$, and r accepts w iff $Inf(r) \cap \alpha = \emptyset$.

Since $\mathcal{A}$ is not deterministic, it may have many runs on w. In contrast, a deterministic automaton has a single run on w. There are two dual ways in which we can refer to the many runs. When $\mathcal{A}$ is an *existential* automaton (or simply a *nondeterministic* automaton, as we shall call it in the sequel), it accepts an input word w iff there exists an accepting run of $\mathcal{A}$ on w. When $\mathcal{A}$ is a *universal* automaton, it accepts an input word w iff all the runs of $\mathcal{A}$ on w are accepting. *Alternation* was studied in [CKS81] in the context of Turing machines and in [BL80, CKS81, MH84] for finite automata. In particular, [MH84] studied alternating automata on infinite words. Alternation enables us to have both existential and universal branching choices.

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 $\{q_1, q_3\}$ 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 $\mathcal{A}$ as above. We can represent ρ using $\mathcal{B}^+(Q)$. For example, a transition $\rho(q, \sigma) = \{q_1, q_2, q_3\}$ of a nondeterministic automaton $\mathcal{A}$ can be written as $\rho(q, \sigma) = q_1 \vee q_2 \vee q_3$. If $\mathcal{A}$ is universal, the transition can be written as $\rho(q, \sigma) = q_1 \wedge q_2 \wedge q_3$. 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 automata, and $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function. While a run of a nondeterministic automaton is a function $r : \mathbb{N} \to Q$, 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 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 $\theta = \textbf{true}$, 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 $\theta = \textbf{false}$, we cannot have a run that takes a transition with $\theta = \textbf{false}$.

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. Thus, each word automaton defines a subset of Σ^ω. We denote by $\overline{\mathcal{L}(\mathcal{A})}$ the complement language of $\mathcal{A}$, that is the set of all words in $\Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.

In [MSS86], Muller et al. introduce *weak alternating automata* (WAAs). In a WAA, 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 $\delta(q, \sigma)$, for some $\sigma \in \Sigma$, 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 WAA ultimately gets "trapped" within some Q_i. The path then satisfies the acceptance condition if and only if Q_i is an accepting set. Thus, we can view a WAA with an acceptance condition α as both a Büchi automaton with an acceptance condition α, and a co-Büchi automaton with an acceptance condition $Q \setminus \alpha$. Indeed, a run gets trapped in an accepting set iff it visits infinitely many states in

α, which is true iff it visits only finitely many states in $Q \setminus \alpha$.

3 Useful Observations on Runs of Alternating Co-Büchi Automata

Consider a co-Büchi alternating automaton $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$. Let $\langle T_r, r \rangle$ be an accepting run of $\mathcal{A}$ on a word w. For two nodes x_1 and x_2 in T_r, we say that x_1 and x_2 are *similar* iff $|x_1| = |x_2|$ and $r(x_1) = r(x_2)$. We say that the run $\langle T_r, r \rangle$ is *memoryless* iff for all similar nodes x_1 and x_2, and for all $y \in \mathbb{N}^*$, we have that $x_1 \cdot y \in T_r$ iff $x_2 \cdot y \in T_r$, and $r(x_1 \cdot y) = r(x_2 \cdot y)$. Intuitively, similar nodes correspond to two copies of $\mathcal{A}$ that have the same "mission": they should both accept the suffix $w^{|x_1|}$ from the state $r(x_1)$. In a memoryless run, subtrees of $\langle T_r, r \rangle$ with similar roots coincide. Thus, same missions are fulfilled in the same way. It turns out that when we consider runs of co-Büchi automata, we can restrict ourselves to memoryless runs. Formally, we have the following theorem.

Theorem 3.1 [EJ91] *If a co-Büchi automaton $\mathcal{A}$ accepts a word w, then there exists a memoryless accepting run of $\mathcal{A}$ on w.*

Let $|Q| = n$. It is easy to see that for every run $\langle T_r, r \rangle$, every set of more than n nodes of the same level contains at least two similar nodes. Therefore, in a memoryless run of $\mathcal{A}$, every level contains at most n nodes that are roots of different subtrees. Accordingly, we represent a memoryless run $\langle T_r, r \rangle$ by an infinite DAG (directed acyclic graph) $G_r = \langle V, E \rangle$, where

- $V \subseteq Q \times \mathbb{N}$ is such that $\langle q, l \rangle \in V$ iff there exists $x \in T_r$ with $|x| = l$ and $r(x) = q$. For example, $\langle q_{in}, 0 \rangle$ is the only vertex of G_r in $Q \times \{0\}$.
- $E \subseteq \bigcup_{l \geq 0} (Q \times \{l\}) \times (Q \times \{l+1\})$ is such that $E(\langle q, l \rangle, \langle q', l+1 \rangle)$ iff there exists $x \in T_r$ with $|x| = l$, $r(x) = q$, and $r(x \cdot c) = q'$ for some $c \in \mathbb{N}$.

Thus, G_r is obtained from $\langle T_r, r \rangle$ by merging similar nodes into a single vertex. 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 $\langle T_r, r \rangle$ is accepting iff all paths in G_r have only finitely many α-vertices.

Consider a (possibly finite) DAG $G \subseteq G_r$. We say that a vertex $\langle q, l \rangle$ is *eventually safe* in G iff only finitely many vertices in G are reachable from $\langle q, l \rangle$. We say that a vertex $\langle q, i \rangle$ is *currently safe* in G iff all the vertices in G that are reachable from $\langle q, l \rangle$ are not α-vertices. Note that, in particular, $\langle q, i \rangle$ is not an α-vertex.

Given a memoryless accepting run $\langle T_r, r \rangle$, we define an infinite sequence $G_0 \supseteq G_1 \supseteq G_2 \supseteq \ldots$ of DAGs inductively as follows.

- $G_0 = G_r$.
- $G_{2i+1} = G_{2i} \setminus \{\langle q, l \rangle \mid \langle q, l \rangle \text{ is eventually safe in } G_{2i}\}$.
- $G_{2i+2} = G_{2i+1} \setminus \{\langle q, l \rangle \mid \langle q, l \rangle \text{ is currently safe in } G_{2i+1}\}$.

Lemma 3.2 *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. Indeed, in G_r 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 currently safe 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 currently safe. 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 currently safe, 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 currently safe. 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 currently safe. 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 i, 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 $\langle T_r, r \rangle$ that visits α infinitely often, contradicting the assumption that $\langle T_r, r \rangle$ is an accepting run.

So, let $\langle q, l \rangle$ be a currently safe vertex in G_{2i+1}. We claim that taking $l_{i+1} = l$ satisfies the lemma's requirement. That is, we claim that for all $j \geq l$, 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 eventually safe 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 eventually safe 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 currently safe. Hence, being reachable from $\langle q, l \rangle$, all the vertices $\langle q_k, l+k \rangle$ in the path are currently safe 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. □

By Lemma 3.2, G_{2n} is finite. Hence the following corollary.

Corollary 3.3 *G_{2n+1} is empty.*

Each vertex $\langle q, l \rangle$ in G_r has a unique index $i \geq 1$ such that $\langle q, l \rangle$ is either eventually safe in G_{2i} or currently safe in G_{2i+1}. Given a vertex $\langle q, l \rangle$, we define the *rank* of $\langle q, l \rangle$, denoted $rank(q, l)$, to be:

- $2i$, if $\langle q, l \rangle$ is eventually safe in G_{2i}.
- $2i+1$, if $\langle q, l \rangle$ is currently safe in G_{2i+1}.

For $k \in \mathbb{N}$, let $[k]$ denote the set $\{0, 1, \ldots, k\}$, and let $[k]^{odd}$ denote the set of odd members of $[k]$. By Corollary 3.3, the rank of every vertex in G_r is in $[2n]$. Recall that when $\langle T_r, r \rangle$ is accepting, all the paths in G_r visit only finitely many α-vertices. Intuitively, $rank(q, l)$ hints how difficult it is to get convinced that all the paths of G_r that visit the vertex $\langle q, l \rangle$ visit only finitely many α-vertices. Easiest to get convinced about are vertices that are eventually safe in G_0. Accordingly, they get the minimal rank 0. Then come vertices that are currently safe in the graph G_1, which is obtained from G_0 by throwing vertices with rank 0. These vertices get the rank 1. The process repeats with respect to the graph G_2, which is obtained from G_1 by throwing vertices with rank 1. As before, we start with the eventually safe vertices in G_2, which get the rank 2. We continue with the currently safe vertices in G_3, which get the rank 3. The process repeats until all vertices get some rank. Note that no α-vertex gets an odd rank.

In the lemmas below we make this intuition formal.

Lemma 3.4 *For every vertex $\langle q, l \rangle$ in G_r and rank $i \in [2n]$, if $\langle q, l \rangle \notin G_i$, then $rank(q, l) < i$.*

Proof: We prove the lemma by an induction on i. Since $G_0 = G_r$, the case where $i = 0$ is immediate. For the induction step, we distinguish between two cases. For the case $i+1$ is even, consider a vertex $\langle q, l \rangle \notin G_{i+1}$. If $\langle q, l \rangle \notin G_i$, the lemma's requirement follows from the induction hypothesis. If $\langle q, l \rangle \in G_i$, then $\langle q, l \rangle$ is currently safe in G_i. Accordingly, $rank(q, l) = i$, meeting the lemma's requirement. For the case $i+1$ is odd, consider a vertex $\langle q, l \rangle \notin G_{i+1}$. If $\langle q, l \rangle \notin G_i$, the lemma's requirement follows from the induction hypothesis. If $\langle q, l \rangle \in G_i$, then $\langle q, l \rangle$ is eventually safe in G_i. Accordingly, $rank(q, l) = i$, meeting the lemma's requirement. □

Lemma 3.5 *For every two vertices $\langle q, l\rangle$ and $\langle q', l'\rangle$ in G_r, if $\langle q', l'\rangle$ is reachable from $\langle q, l\rangle$, then $rank(q', l') \leq rank(q, l)$.*

Proof: Assume that $rank(q, l) = i$. We distinguish between two cases. If i is even, in which case $\langle q, l\rangle$ is eventually safe in G_i, then either $\langle q', l'\rangle$ is not in G_i, in which case, by Lemma 3.4, 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 eventually safe in G_i and have rank i. If i is odd, in which case $\langle q, l\rangle$ is currently safe in G_i, then either $\langle q', l'\rangle$ is not in G_i, in which case, by Lemma 3.4, 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 currently safe in G_i and have rank i. $\square$

Lemma 3.6 *In every infinite path in G_r, 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 $rank(q', l') = rank(q, l)$.*

Proof: By Lemma 3.5, in every infinite path in G_r, 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 $rank(q', l') = rank(q, l)$. 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 eventually safe in G_i. Then, the rank of all the vertices in the path that are reachable from $\langle q, l\rangle$ is also i. By Lemma 3.4, 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 eventually safe in G_i. $\square$

We have seen that if a co-Buchi alternating automaton has an accepting run on w, then it also has a very structured accepting run on w. In the next section we employ this structured run in order to translate Büchi and co-Büchi alternating automata to weak alternating automata.

Progress measures were introduced in [Kla90] as a generic concept for quantifying how each step of a program contributes to bringing a computation closer to its specification. Progress measures are used in [Kla91] for reasoning about automata on infinite words. The ranks defined above also measure progress: they indicate how far the automaton is from satisfying its co-Büchi acceptance condition. When we use these ranks, we consider, unlike [Kla91], alternating automata. Consequently, much of the complication in [Kla91] is handled by the rich structure of the automata.

4 From Büchi and Co-Büchi to Weak Alternating Automata

Theorem 4.1 *Let $\mathcal{A}$ be an alternating co-Büchi automaton. There is a weak alternating automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ and the number of states in $\mathcal{A}'$ is quadratic in that of $\mathcal{A}$.*

Proof: Let $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$, and let $n = |Q|$. We define $\mathcal{A}' = \langle \Sigma, Q', q'_{in}, \delta', \alpha'\rangle$, where

- $Q' = Q \times [2n]$. Intuitively, when the automaton is in state $\langle q, i\rangle$ as it reads the letter σ_l (the l'th letter in the input), then it guesses that in a memoryless accepting run of $\mathcal{A}$ on w, the rank of $\langle q, l\rangle$ is i. An exception is the initial state q'_{in} explained below.

- $q'_{in} = \langle q_{in}, 2n\rangle$. That is, q_{in} is paired with $2n$, which is an upper bound on the rank of $\langle q_{in}, 0\rangle$.

- We define δ' by means of a function

  $$release : \mathcal{B}^+(Q) \times [2n] \to \mathcal{B}^+(Q').$$

  Given a formula $\theta \in \mathcal{B}^+(Q)$, and a rank $i \in [2n]$, the formula $release(\theta, i)$ is obtained from θ by replacing an atom q by the disjunction $\bigvee_{i' \leq i}\langle q, i'\rangle$. For example,

  $$\begin{aligned} release(q \wedge p, 2) = & \quad (\langle q, 2\rangle \vee \langle q, 1\rangle \vee \langle q, 0\rangle) \\ \wedge & \quad (\langle p, 2\rangle \vee \langle p, 1\rangle \vee \langle p, 0\rangle). \end{aligned}$$

  Now, $\delta' : Q' \times \Sigma \to \mathcal{B}^+(Q')$ is defined as follows. For a state $\langle q, i\rangle \in Q'$ and $\sigma \in \Sigma$, the transition $\delta'(\langle q, i\rangle, \sigma)$ is:

  - $release(\delta(q, \sigma), i)$, if $q \notin \alpha$ or i is even.
  - **false**, if $q \in \alpha$ and i is odd.

That is, if the current guessed rank is i then, by employing *release*, the run can move in its successors to any rank that is smaller than i. If, however, $q \in \alpha$ and the current guessed rank is odd, then, by the definition of ranks, the current guessed rank is wrong, and the run is rejecting.

- $\alpha' = Q \times [2n]^{odd}$. That is, infinitely many guessed ranks along each path should be odd.

We first show that $\mathcal{A}'$ is weak. For that, we define a partition of the states of $\mathcal{A}'$ and an order on this partition so that the weakness conditions hold. Each rank $i \in [2n]$ induces the set $Q_i = Q \times \{i\}$ in the partition. Thus, two states $\langle q, i\rangle$ and $\langle q', i'\rangle$ are in the same set iff $i = i'$. We define the order $\leq$ by $Q_i \leq Q_{i'}$ iff $i \leq i'$. It is easy to see that the weakness conditions hold: for every state $\langle q, i\rangle \in Q'$ and $\sigma \in \Sigma$, the states appearing in $\delta'(\langle q, i\rangle, \sigma)$ belongs to sets $Q_{i'} \leq Q_i$, and every set Q_i is either contained in α or disjoint from α. By the definition of α', it follows that the copies of $\mathcal{A}'$ are allowed to get trapped in sets with odd ranks and are not allowed to get trapped in sets with even ranks.

We now prove the correctness of the construction. We first prove that $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{L}(\mathcal{A})$. Consider a word w accepted by $\mathcal{A}'$. Let $\langle T_r, r'\rangle$ be the accepting run of $\mathcal{A}'$ on w. Consider the Q-labeled tree $\langle T_r, r\rangle$ where for all $x \in T_r$ with $r'(x) = \langle q, i\rangle$, we have $r(x) = q$. Thus, $\langle T_r, r\rangle$ projects the labels of $\langle T_r, r'\rangle$ on their Q element. It is easy to see that $\langle T_r, r\rangle$ is a run of $\mathcal{A}$ on w. Indeed, the transitions of $\mathcal{A}'$ only annotate transitions of $\mathcal{A}$ by ranks. We show that $\langle T_r, r\rangle$ is an accepting run. Since $\langle T_r, r'\rangle$ is accepting, then, by the definition of α', each infinite path of $\langle T_r, r'\rangle$ gets trapped in a set $Q \times \{i\}$ for some odd i. By the definition of δ', no accepting run can visit a state $\langle q, i\rangle$ with an odd i and $q \in \alpha$. Hence, the infinite path actually gets trapped in the subset $(Q \setminus \alpha) \times \{i\}$ of $Q \times \{i\}$. Consequently, in $\langle T_r, r\rangle$, all the paths visits states in α only finitely often, and we are done.

It is left to prove that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$. Consider a word w accepted by $\mathcal{A}$. Let $\langle T_r, r\rangle$ be the accepting run of $\mathcal{A}$ on w. Consider the Q'-labeled tree $\langle T_r, r'\rangle$ where $r'(\varepsilon) = \langle r(\varepsilon), 2n\rangle$, and for all other $x \in T_r$, we have $r'(x) = \langle r(x), i\rangle$, where i is the rank of $\langle r(x), |x|\rangle$ in G_r. We claim that $\langle T_r, r\rangle$ is an accepting run of $\mathcal{A}'$. We first prove that it is a run. Since $r(\varepsilon) = q_{in}$ and $q'_{in} = \langle q_{in}, 2n\rangle$, the root of the tree $\langle T_r, r'\rangle$ is labeled legally. We now consider the other nodes of $\langle T_r, r'\rangle$. Let $S = \{q_1, \ldots, q_k\}$ be the set of labels of ε's successors in $\langle T_r, r\rangle$. As $2n$ is the maximal rank that a vertex can get, each successor c of ε in T_r has $rank(r(c), 1) \leq 2n$. Therefore, the set $S' = \{\langle q_1, rank(q_1, 1)\rangle, \ldots, \langle q_k, rank(q_k, 1)\rangle\}$ satisfies $\delta'(\langle q_{in}, 2n\rangle, \sigma_0)$. Hence, the first level of $\langle T_r, r'\rangle$ is also labeled legally. For the other levels, consider a node $x \in T_r$ such that $x \neq \varepsilon$ and $rank(r(x), |x|) = i$. Let $S = \{q_1, \ldots, q_k\}$ be the set of labels of x's successors in $\langle T_r, r\rangle$. By Lemma 3.5, each successor $x \cdot c$ of x in T_r has $rank(r(x \cdot c), |x \cdot c|) \leq i$. Also, by the definition of ranks, it cannot be that $r(x) \in \alpha$ and i is odd. Therefore, the set $S' = \{\langle q_1, rank(q_1, |x|+1)\rangle, \ldots, \langle q_k, rank(q_k, |x|+1)\rangle\}$ satisfies $\delta'(\langle r(x), i\rangle, \sigma_{|x|})$. Hence, the tree $\langle T_r, r'\rangle$ is a run of $\mathcal{A}'$ on w. Finally, by Lemma 3.6, each infinite path of $\langle T_r, r'\rangle$ gets trapped in a set with an even index, thus $\langle T_r, r'\rangle$ is accepting. $\square$

As discussed in [MS87], one can complement an alternating automaton by dualizing its transition function and acceptance condition. Formally, given a transition function δ, let $\tilde{\delta}$ denote the dual function of δ. That is, for every q and σ with $\delta(q, \sigma) = \theta$, we have $\tilde{\delta}(q, \sigma) = \tilde{\theta}$, where $\tilde{\theta}$ is obtained from θ by switching $\vee$ and $\wedge$ and by switching **true** and **false**. If, for example, $\theta = q_1 \vee (\mathbf{true} \wedge q_2)$ then $\tilde{\theta} = q_1 \wedge (\mathbf{false} \vee q_2)$. The dual of an acceptance condition α is a condition that accepts exactly all the words in Q^ω that are not accepted by α. In particular, we have the following.

Theorem 4.2 [MS87] *For an alternating Büchi automaton* $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$*, the alternating* *co-Büchi automaton* $\tilde{\mathcal{A}} = \langle \Sigma, Q, q_{in}, \tilde{\delta}, \alpha \rangle$ *satisfies* $\mathcal{L}(\tilde{\mathcal{A}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$.

The complementation construction in Theorem 4.2 is not only conceptually simple, but it also involves no blow-up. In addition, complementing a WAA does not sacrifice its weakness. Hence, Theorems 4.1 and 4.2 imply the following theorem.

Theorem 4.3 *Let $\mathcal{A}$ be an alternating Büchi automaton. There is a weak alternating automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$ and the number of states in $\mathcal{A}'$ is quadratic in that of $\mathcal{A}$.*

In Section 5, we use the translation described in Theorem 4.1 in order to obtain a simple complementation construction for nondeterministic Büchi automata. As we shall note there, the known lower bound on the complexity of the latter then implies that the quadratic blow-up involved in moving from co-Büchi alternating automata to WAA cannot be reduced to a linear one.

5 Complementing Nondeterministic Büchi Automata

Unlike alternating automata, complementation of nondeterministic automata is a complicated problem. Following Theorem 4.2, all one needs in order to complement a nondeterministic Büchi automaton is some translation of universal co-Büchi automata to nondeterministic Büchi automata. In [MH84], Miyano and Hayashi suggest a translation of alternating Büchi automata to nondeterministic Büchi automata. We present (a simplified version of) their translation in Theorem 5.1 below.

Theorem 5.1 [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 $\delta'(\langle S, O \rangle, \sigma) =$

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

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

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

$\square$

The translation in Theorem 5.1, however, does not handle alternating (and in particular universal) co-Büchi automata, which is what one gets by dualizing a nondeterministic Büchi automaton. Here is where our construction in Theorem 4.1 becomes essential. Thus, given nondeterministic Büchi automaton $\mathcal{B}$, we suggest the following complementation construction for $\mathcal{B}$.

1. Following Theorem 4.2, construct from $\mathcal{B}$ its dual co-Büchi universal automaton $\tilde{\mathcal{B}}$. The automaton $\tilde{\mathcal{B}}$ satisfies $\mathcal{L}(\tilde{\mathcal{B}}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{B})$.

2. Following Theorem 4.1, construct from $\tilde{\mathcal{B}}$ its equivalent weak alternating automaton $\mathcal{W}$. The automaton $\mathcal{W}$ satisfies $\mathcal{L}(\mathcal{W}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{B})$.

3. Following Theorem 5.1, construct from $\mathcal{W}$ its equivalent nondeterministic Büchi automaton $\mathcal{N}$. The automaton $\mathcal{N}$ satisfies $\mathcal{L}(\mathcal{N}) = \Sigma^\omega \setminus \mathcal{L}(\mathcal{B})$.

If $\mathcal{B}$ has n states, then $\tilde{\mathcal{B}}$ has n states as well, $\mathcal{W}$ has $O(n^2)$ states, and $\mathcal{N}$ has $2^{O(n^2)}$ states. By [Mic88, Saf88], however, an optimal complementation construction for nondeterministic

Büchi automata results in an automaton $\mathcal{N}$ with $2^{O(n\log n)}$ states. Before we describe how we do get, using Theorem 4.1, such an optimal automaton $\mathcal{N}$, let us note that the above scheme implies that the translation described in Theorem 4.1 cannot be improved to a linear translation. Indeed, being able to construct from $\tilde{\mathcal{B}}$ to an equivalent WAA $\mathcal{W}$ with only $O(n)$ states, we are also able to construct $\mathcal{N}$ with $2^{O(n)}$ states, contradicting the $2^{O(n\log n)}$ lower bound.

In order to get $\mathcal{N}$ with $2^{O(n\log n)}$ states, we exploit the special structure of $\mathcal{W}$ as follows. Let $\mathcal{B} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$. Consider a state $\langle S, O\rangle$ of $\mathcal{N}$. Each of the sets S and O is a subset of $Q \times [2n]$. We say that $P \subseteq Q\times[2n]$ is *consistent* iff for every two states $\langle q, i\rangle$ and $\langle q', i'\rangle$ in P, if $q = q'$ then $i = i'$. We claim the following.

Claim 1. Restricting the states in $\mathcal{N}$ to pairs $\langle S, O\rangle$ for which S is a consistent subset of $Q\times[2n]$ is allowable; that is, the resulted $\mathcal{N}$ still complements $\mathcal{B}$.

Claim 2. There are $2^{O(n\log n)}$ consistent subsets of $Q \times [2n]$.

By the two claims, as O is always a subset of S, it is easy to restrict the state space of $\mathcal{N}$ to $2^{O(n\log n)}$ states. In order to prove Claim 1, recall that the automaton $\mathcal{W}$ visiting a state $\langle q, i\rangle$ after reading l letters of an input word w corresponds to a guess that the rank of $\langle q, l\rangle$ in an accepting and memoryless run of $\tilde{\mathcal{B}}$ on w is i. We have seen that if there is an accepting and memoryless run $\langle T_r, r\rangle$ of $\tilde{\mathcal{B}}$ on w, then a run of $\mathcal{W}$ that follows the ranks in G_r is accepting. Since every vertex in G_r has a unique rank, the copies of $\mathcal{W}$ that are created in each level l in this accepting run are consistent, in the sense that the set of states visited by copies of $\mathcal{W}$ in level l in the run is consistent. In $\mathcal{N}$, all the states in S correspond to copies of $\mathcal{W}$ that read the same prefix of w. Hence, a state $\langle S, O\rangle$ for which S is inconsistent corresponds to a level l in a run of $\mathcal{W}$ whose copies are inconsistent. Hence, the automaton $\mathcal{N}$ can ignore states $\langle S, O\rangle$ with inconsistent S.

In order to prove Claim 2, observe that we can characterize a consistent set by the projection of its pairs on Q, augmented by an assignment $f : Q \to [2n]$. Since there are 2^n such projections and $n^{O(n)} = 2^{O(n\log n)}$ such assignments, we are done.

Composing the three constructions is straightforward. Below we define the automaton $\mathcal{N}$ directly, by means of $\mathcal{B}$'s components. Given a nondeterministic Büchi automaton $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$, we define a nondeterministic Büchi automaton $\mathcal{A}'$ such that $\mathcal{L}(\mathcal{A}') = \Sigma^\omega \setminus \mathcal{L}(\mathcal{A})$. Let $|Q| = n$. For a set $P \in 2^{Q\times[2n]}$, we say that P is *possible* iff there exists no pair $\langle q, i\rangle$ in P such that i is odd and $q \in \alpha$. For two sets P and P' in $2^{Q\times[2n]}$ and a letter $\sigma \in \Sigma$, we say that P' *covers* $\langle P, \sigma\rangle$ iff for every pair $\langle q, i\rangle \in P$ and state $q' \in \delta(q, \sigma)$, there exists $i' \leq i$ such that the pair $\langle q', i'\rangle$ is in P'.

The automaton $\mathcal{A}' = \langle \Sigma, Q', q'_{in}, \delta', \alpha'\rangle$, where

- $Q' = \{\langle S, O\rangle : S \in 2^{Q\times[2n]}, O \subseteq S,$ and S is possible and consistent $\}$.
- $q'_{in} = \langle \{\langle q_{in}, 2n\rangle\}, \emptyset\rangle$.
- For a state $\langle S, O\rangle \in Q'$ and a letter $\sigma \in \Sigma$, we define $\delta'(\langle S, O\rangle, \sigma)$ as follows.
  - ∘ If $O \neq \emptyset$, then $\delta'(\langle S, O\rangle, \sigma) =$

    $$\{\langle S', O' \setminus (Q \times [2n]^{odd})\rangle : S' \text{ covers } \langle S, \sigma\rangle, O' \subseteq S', O' \text{ covers } \langle O, \sigma\rangle, \text{ and } S' \text{ is possible and consistent } \}.$$

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

    $$\{\langle S', S' \setminus (Q \times [2n]^{odd})\rangle : S' \text{ covers } \langle S, \sigma\rangle \text{ and } S' \text{ is possible and consistent } \}.$$

- $\alpha' = \{\langle S, \emptyset\rangle : S \in 2^{Q\times[2n]}$ and S is possible and consistent$\}$.

References

[BL80] J.A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.

[Büc62] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. International Congress on Logic, Method and Philos. Sci. 1960*, pages 1–12, 1962.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Proc. 6th CAV*, LNCS 818, pages 142–155, 1994.

[CDK93] E. M. Clarke, I. A. Draghicescu, and R. P. Kurshan. A unified approach for showing language containment and equivalence between various types of ω-automata. *Information Processing Letters* **46**, pages 301–308, (1993).

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

[DH94] D. Drusinsky and D. Harel. On the power of bounded concurrency I: Finite automata. *Journal of the ACM*, 41(3):517–539, 1994.

[EJ91] E.A. Emerson and C. Jutla. Tree automata, mu-calculus and determinacy. In *Proc. 32nd FOCS*, pages 368–377, 1991.

[Hol91] G. Holzmann. *Design and Validation of Computer Protocols*. Prentice-Hall International Editions, 1991.

[Kla90] N. Klarlund. *Progress Measures and finite arguments for infinite computations*. PhD thesis, Cornell University, 1990.

[Kla91] N. Klarlund. Progress measures for complementation of ω-automata with applications to temporal logic. In *Proc. 32nd FOCS*, pages 358-367, 1991.

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

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

[Lin88] P. Lindsay. On alternating ω-automata. *Theoretical computer science*, 43:107–116, 1988.

[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.

[MS87] D.E. Muller and P.E. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54,:267–276, 1987.

[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 ICALP*, 1986.

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

[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 omega-automata. In *Proc. 29th FOCS*, pages 319-327, 1988.

[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.

[THB95] S. Tasiran, R. Hojati, and R.K. Brayton. Language containment using non-deterministic omega-automata. In *Proc. CHARME '95*, LNCS 987, pages 261–277, 1995.

[Tho90] W. Thomas. Automata on infinite objects. *Handbook of Theoretical Computer Science*, pages 165–191, 1990.

[Var96] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, LNCS 1043, pages 238–266, 1996.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st LICS*, pages 322–331, 1986.

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