Weak Alternating Automata and Tree Automata Emptiness

Orna Kupferman*
UC Berkeley

Moshe Y. Vardi†
Rice University

Abstract *Automata on infinite words and trees are used for specification and verification of nonterminating programs. The verification and the satisfiability problems of specifications can be reduced to the nonemptiness problem of such automata. In a weak 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 automata is easier than reasoning about automata with no restricted structure. In particular, the nonemptiness problem for weak alternating automata over a singleton alphabet can be solved in linear time. Known translations of alternating automata to weak alternating automata involve determinization, and therefore involve a double exponential blow-up. In this paper we describe simple and efficient translations, which circumvent the need for determinization, of parity and Rabin alternating word automata to weak alternating word automata. Beyond the independent interest of such translations, they give rise to a simple algorithm for deciding the nonemptiness of nondeterministic parity and Rabin tree automata. In particular, our algorithm for Rabin automata runs in time $O(n^{2k+1} \cdot k!)$, where n is the number of states in the automaton and k is the number of pairs in the acceptance condition. This improves the known $O((nk)^{3k})$ bound for the problem.*

*Address: EECS Department, UC Berkeley, Berkeley CA 94720-1770, U.S.A. Email: orna@eecs.berkeley.edu, URL: http://www.eecs.berkeley.edu/~orna. Supported in part by ONR YIP award N00014-95-1-0520, by NSF CAREER award CCR-9501708, by NSF grant CCR-9504469, by AFOSR contract F49620-93-1-0056, by ARO MURI grant DAAH-04-96-1-0341, by ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

†Address: 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. Supported in part by NSF grants CCR-9628400 and CCR-9700061, and by a grant from the Intel Corporation.

1 Introduction

Finite automata on infinite objects were first introduced in the 1960's. Motivated by decision problems in mathematical logic, Büchi, McNaughton, and Rabin developed a framework for automata on infinite words and infinite trees [Büc62, McN66, Rab69]. The framework has proven 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 mathematical logic [Tho90]. Today, automata on infinite objects are used for specification and verification of nonterminating programs. By translating specifications to automata, 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 reduced to questions such as nonemptiness and language containment [VW86, Kur94, VW94]. The automata-theoretic approach separates the logical and the algorithmic aspects of reasoning about programs. The translation of specifications to automata handles the logic and shifts all the algorithmic difficulties to automata-theoretic problems.

Like 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 various 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*, *parity*, and *Rabin* automata, whose acceptance conditions involve several sets of states. For example, in parity automata [Mos84, EJ91], the acceptance condition is a sequence $\{F_1, F_2, \ldots, F_k\}$ of sets of states. A run is accepting iff the minimal index i for which the set F_i is visited infinitely often is even.

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 [BL80, 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.

Since the combinatorial structure of alternating automata is rich, translating specifications to alternating automata is much simpler than translating them to nondeterministic automata [Var94]. Alternating automata enable a complete partition between the logical and the algorithmic aspects of reasoning about programs, and they give rise to cleaner and simpler verification algorithms [Var96]. The rich structure of alternating automata also makes them more *succinct*. For example, translating an alternating Büchi automaton to a nondeterministic Büchi automaton might involve an exponential blow up [DH94]. The succinctness of alternating automata is crucial when we use automata for the verification of branching-time specifications. In this paradigm, each specification describes a set of allowed computation trees, which can be described by an *automaton over infinite trees*. By translating branching-time specifications to alternating tree automata, we can reduce satisfiability to the nonemptiness problem and reduce verification to the membership problem [BVW94, Var97]. Solving the nonemptiness problem for an alternating tree automaton is done by translating the automaton to a nondeterministic tree automaton. Deciding the membership of a program in a language of an alternating tree automaton is done by taking the product of the program and the automaton. This product can be defined as an alternating word automaton over a singleton alphabet, and the program is correct with respect to the specification iff this automaton is nonempty. Thus, reasoning about branching-time specifications concerns two problems: the nonemptiness problem for nondeterministic tree automata and the nonemptiness problem for alternating word automata over a singleton alphabet. It is shown in [BVW94] that these problems are equivalent and that their complexities coincide. We refer to both problems as the *nonemptiness problem*. The nonemptiness problem is important also for reasoning about linear-time specifications of open systems, where the interaction between a correct system and its environment can be formulated by a tree automaton [ALW89, PR89].

In [MSS86], Muller et al. introduced *weak automata*. In a weak 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 automaton eventually gets trapped in some set in the partition. Acceptance is then determined according to the classification of this set. Weak automata are a special case of Büchi 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 special structure of weak automata is reflected in their attractive computational properties. In particular, the nonemptiness problem for weak automata can be solved in linear time [BVW94]. As a comparison, the best known upper bound for the nonemptiness problem for Büchi automata is quadratic time.

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 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 [Mic88, Saf88, Saf92]. Even worse, if $\mathcal{A}$ is an alternating automaton, then its determinization involves a doubly-exponential blow-up [DH94], and hence, so does the translation to weak alternating automata. Can these blow-ups be avoided? In [KV97], we described a quadratic translation of Büchi and co-Büchi alternating word automata to weak alternating word automata, answering this question positively for the case $\mathcal{A}$ is either a Büchi or a co-Büchi automaton. In this paper we extend the ideas in [KV97] and describe an efficient translation of stronger types of alternating automata to weak alternating automata. Since the nonemptiness problem for weak automata can be solved in linear time, this enables us to improve the known upper bounds for the nonemptiness problem.

We start with *parity automata*. It is shown in [EJ91] that formulas of the μ-calculus [Koz83] can be linearly translated to alternating parity tree automata[1]. Since many properties of programs are naturally specified by means of fixed points, the μ-calculus is an expressive and important specification language [EL86]. Following [EJ91], the verification problem for μ-calculus can be linearly reduced to the nonemptiness problem for parity automata. This makes the nonemptiness problem for parity automata of particular interest; the verification problem for μ-calculus is known to be in NP $\cap$ co-NP [EJS93] and its precise complexity is an open problem. Given an alternating parity word automaton with n states and k sets, we construct an equivalent weak alternating word automaton with $O(n^k)$ states. The construction goes through a sequence of k intermediate automata. Each automaton in the sequence refines the state space of its predecessor and has one less set in its parity acceptance condition.

Parity automata can be viewed as a special case of *Rabin automata*. In Rabin automata, the acceptance condition is a set $\alpha = \{\langle G_1, B_1\rangle, \langle G_2, B_2\rangle, \ldots, \langle G_k, B_k\rangle\}$ of pairs of sets of states. A run is accepting if there exists an index i for which the set G_i is visited infinitely often and the set B_i is visited only finitely often. In [Rab69], Rabin describes a translation of formulas of monadic second order logic to Rabin tree automata. Today, Rabin automata are used in order to reason about specifications of the full branching time logic CTL* [ES84, VS85], as well as to model programs with fairness conditions. The nonemptiness problem for Rabin automata plays a crucial role in solving various decision problems in logic. As a result, many efforts have been put in developing simple algorithms for nonemptiness checking. In [Rab69], Rabin described a non-elementary procedure for checking the nonemptiness of a given Rabin automaton and showed that the problem is decidable. In [HR72, Rab72], improved algorithms were described, of complexity exponential in both n and k. Only in [EJ88, PR89], algorithms that are exponential in k and only polynomial in n have been describes. Both works described algorithms that run in time $O((nk)^{3k})$. Given an alternating Rabin word automaton with n states and k pairs, we construct an equivalent weak alternating word automaton with $O(n^{2k+1} \cdot k!)$ states. Our constructions yield $O(n^k)$ and $O(n^{2k+1} \cdot k!)$ upper bounds for the nonemptiness problem for parity and Rabin automata, respectively, matching the known bound for parity automata [EJS93] and improving the known $O(nk)^{3k}$ bound for Rabin automata.

[1] In fact, alternating parity tree automata are exactly as expressive as the μ-calculus [Niw88, EJ91]. On the other hand, weak alternating tree automata are exactly as expressive as the alternation-free fragment of μ-calculus [KV98].

2 Alternating Automata

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.

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 *alternating automaton on infinite words* is $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$, where Σ is the input alphabet, Q is a finite set of states,

$\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is a transition function, $q_{in} \in Q$ is an initial state, and α is an acceptance condition. Intuitively, $\delta(q, \sigma)$ describes possible configurations that $\mathcal{A}$ can move into when it is in state q and it reads the letter σ. For example, a transition $\delta(q, \sigma) = (q_1 \wedge q_2) \vee (q_3 \wedge q_4)$ means that $\mathcal{A}$ accepts a suffix w^l of w from state q, if it accepts w^{l+1} from both q_1 and q_2 or from both q_3 and q_4. The acceptance condition α defines a subset of Q^ω. For a word $\eta = q_0 \cdot q_1 \cdots$ in Q^ω, we define the set $Inf(\eta)$ of states that η visits *infinitely often*, i.e.,

$$Inf(\eta) = \{q \in Q : \text{ for infinitely many } l \geq 0, \text{ we have } q_l = q\}.$$

As Q is finite, it is guaranteed that $Inf(\eta) \neq \emptyset$. The way α refers to $Inf(\eta)$ depends on its type. In *Büchi automata*, $\alpha \subseteq Q$, and η satisfies α iff $Inf(\eta) \cap \alpha \neq \emptyset$. Dually, in *co-Büchi automata*, η satisfies α iff $Inf(\eta) \cap \alpha = \emptyset$.

In order to define a run of an alternating automaton, we first define trees. A *tree* is a (finite or infinite) nonempty set $T \subseteq \mathbb{N}^*$ such that for all $x \cdot c \in T$, with $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, we have $x \in T$. The elements of T are called *nodes*, and the empty word ε is the *root* of T. For every $x \in T$, the nodes $x \cdot c \in T$ where $c \in \mathbb{N}$ are the *children* of x. A node with no children is a *leaf*. We refer to the length $|x|$ of x as its *level* in the tree. A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, either x is a leaf, or there exists a unique $c \in \mathbb{N}$ such that $x \cdot c \in \pi$. Given 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. 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 Observations on Runs of Alternating Co-Büchi Automata

Let $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha \rangle$ a co-Büchi alternating automaton. Consider an accepting run $\langle T_r, r \rangle$ 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, if a co-Büchi automaton $\mathcal{A}$ accepts a word w, then there exists a memoryless accepting run of $\mathcal{A}$ on w [EJ91].

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 $\langle q, l\rangle$ in G_r 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.

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

It is shown in [KV97] that every vertex $\langle q, l\rangle$ in G_r has a unique index $i \geq 0$ 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 $m \in \mathbb{N}$, let $[m]$ denote the set $\{0, 1, \ldots, m\}$, and let $[m]^{even}$ and $[m]^{odd}$ denote the sets of even and odd members of $[m]$, respectively. In [KV97], we proved that 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, proven in [KV97], we make this intuition formal.

Lemma 3.1 *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 $rank(q', l') \leq rank(q, l)$.*

Lemma 3.2 *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 $rank(q', l') = rank(q, l)$.*

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 [KV97] we employed this structured run in order to translate co-Büchi alternating automata to weak alternating automata:

Theorem 3.3 [KV97] *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 $\mathcal{A}$.*

We describe the automaton $\mathcal{A}'$. Let $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \alpha\rangle$, and let $n = |Q|$. The state space of the automaton $\mathcal{A}'$ is $Q \times [2n]$. Intuitively, when $\mathcal{A}'$ is in state $\langle q, j\rangle$ as it reads the letter σ_l (the l'th letter in the input), it guesses that in a memoryless accepting run of $\mathcal{A}$ on w, the rank of $\langle q, l\rangle$ is j. Accordingly, when $\mathcal{A}'$ is in state $\langle q, j\rangle$ and it

reads a letter σ, it follows the transition $\delta(q, \sigma)$ allowing the successors to move to states annotated by ranks in $[j]$. If, however, $q \in \alpha$ and j is odd, then, by the definition of ranks, the current guessed rank is wrong, and the run is rejecting. Each path in a run of $\mathcal{A}'$ eventually gets trapped in a set of the form $Q \times \{j\}$ for some $j \in [2n]$. The set is accepting iff j is odd.

In the next section we extend the ideas in [KV97] in order to translate parity and Rabin alternating automata to weak alternating automata.

4 From Parity and Rabin to Weak Alternating Automata

A *parity alternating automaton* is $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \beta \rangle$, where $\beta = \langle F_1, F_2, \ldots, F_k \rangle$ with $F_1 \subset F_2 \subset \cdots \subset F_k = Q$. We refer to k (the number of sets in β) as the *degree* of β (or $\mathcal{A}$). A word $\eta \in Q^\omega$ satisfies a parity condition β iff the minimal index i for which $Inf(\eta) \cap F_i \neq \emptyset$ is even. We also refer to *co-parity* acceptance conditions. A word $\eta \in Q^\omega$ satisfies a co-parity condition β iff the minimal index i for which $Inf(\eta) \cap F_i \neq \emptyset$ is odd; that is, iff η does not satisfy β when referred to as a parity condition.

Consider a parity automaton $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \beta \rangle$ with $\beta = \{F_1, F_2, \ldots, F_k\}$, and the co-Büchi automaton $\mathcal{A}_c = \langle \Sigma, Q, q_{in}, \delta, F_1 \rangle$. That is, $\mathcal{A}_c$ differs from $\mathcal{A}$ only in the acceptance condition, which consists of the minimally indexed set in β. Clearly, every accepting run $\langle T_r, r \rangle$ of $\mathcal{A}$ is also an accepting run of $\mathcal{A}_c$. Indeed, all the paths of $\langle T_r, r \rangle$ must visit F_1 only finitely often (otherwise, the minimal index i for which F_i is visited infinitely often is 1, which is odd). It follows that when we consider accepting runs of $\mathcal{A}$, we can restrict ourselves to candidates from the set of accepting runs of $\mathcal{A}_c$.

In Theorem 3.3, we translated the co-Büchi alternating automaton $\mathcal{A}_c$ to a weak alternating automaton $\mathcal{A}'_c$. In the automaton $\mathcal{A}'_c$, each path of a run gets trapped in some set $Q \times \{j\}$. When j is odd, no visits to F_1 are possible. Therefore, a path η that gets trapped in a set $Q \times \{j\}$, for an odd j, satisfies the parity condition β iff it satisfies the co-parity condition $\beta' = \{F_2, F_3, \ldots, F_k\}$. Indeed, since sets with an even index in β have an odd index in β', the path η satisfies β' iff the minimal i in $\{2, \ldots, k\}$ for which $Inf(\eta) \cap F_i \neq \emptyset$ is even. In addition, as $Inf(\eta) \cap F_1 = \emptyset$, it is guaranteed that i is actually minimal in $\{1, \ldots, k\}$.

The above observation suggests an inductive scheme for translating a parity or a co-parity alternating automaton to a weak alternating automaton. Intuitively, the translation proceeds as follows. Let $\mathcal{A}$ be a parity automaton with $\beta = \{F_1, \ldots, F_k\}$. Translating the co-Büchi automaton $\mathcal{A}_c$ to a weak alternating automaton $\mathcal{A}'_c$, the copies of $\mathcal{A}'_c$ that get trapped in sets that enable infinitely many visits to F_1 (that is, sets $Q \times \{j\}$ for an even j) are rejecting. On the other hand, copies that get trapped in sets that disable visits to F_1 may satisfy β, and we check them further, with respect to the co-parity condition β' obtained from β by taking out the set F_1. Checking these copies is done inductively, by referring to each set $Q \times \{j\}$, for an odd j, as a co-parity automaton. Formally, the induction proceeds by refining the state space of the parity automaton by means of *weak-parity alternating automata*, defined below.

A *weak-parity alternating automaton* (*WPAA*, for short) is $\mathcal{A} = \langle \Sigma, S, P, q_{in}, \delta, \alpha, \beta \rangle$, where S and P are disjoint sets of states (called *simple* and *parity* states, respectively), $q_{in} \in S \cup P$ is an initial state, $\delta : (S \cup P) \times \Sigma \to \mathcal{B}^+(S \cup P)$ is a transition function, $\alpha \subseteq S$ is a Büchi acceptance condition, and β is a parity acceptance condition over P. We refer to the number of sets in β as the degree of β (or $\mathcal{A}$). There exists a partition of $S \cup P$ into disjoint sets, such that the following hold.

- For each set Q in the partition, one of the following holds.
  1. $Q \subseteq \alpha$, in which case Q is an *accepting set*,
  2. $Q \subseteq S$ and $Q \cap \alpha = \emptyset$, in which case Q is a *rejecting set*, or
  3. $Q \subseteq P$, in which case Q is a *parity set*.

  For a state $q \in S \cup P$, let $[q]$ denote the set of states in q's set in the partition.
- There exists a partial order $\leq$ on the collection of the sets such that for every two states q and q' for which q' occurs in $\delta(q, \sigma)$, for some $\sigma \in \Sigma$, we have $[q'] \leq [q]$. Thus, transitions from a state in a set Q lead to states in either the same set Q or a lower set.

It follows that every infinite path η of a run of a WPAA ultimately gets trapped within some set Q in the partition. The path η then satisfies the acceptance condition iff either Q is an accepting set, or Q is a parity set and η satisfies β. Thus, a WPAA $\mathcal{A}$ is very similar to a WAA,

only that in some of the sets in the partition, acceptance is determined according to a parity acceptance condition. In particular, if $P = \emptyset$, then $\mathcal{A}$ is a WAA. On the other hand, every parity automaton can be viewed as a WPAA with $S = \emptyset$ (and a trivial partition with a single set). If there exists $\rho \in \mathbb{N}$ such that the size of each set in the partition is exactly ρ, we say that $\mathcal{A}$ is a WPAA of *width* ρ.

Consider a WPAA $\mathcal{A}$ of width ρ with $\beta = \{F_1, F_2, \ldots, F_k\}$. With each parity set Q of $\mathcal{A}$ and state $q \in Q$, we can associate a co-Büchi automaton $\mathcal{A}_q = \langle \Sigma, Q, q, \delta_Q, F_1 \rangle$, where δ_Q is obtained from δ by replacing states not in Q by **true**. Each accepting run of $\mathcal{A}$ (in which q is participating) induces an accepting run $\mathcal{A}_q$. Formally, if $\langle T_r, r \rangle$ is an accepting run of $\mathcal{A}$ on some word w, and $x \in T_r$ is such that $r(x) = q$, then the subtree of $\langle T_r, r \rangle$ with root x and nodes $x \cdot y$ for which $r(x \cdot y) \in Q$, is an accepting run, embedded in $\langle T_r, r \rangle$, of $\mathcal{A}_q$ on $w^{|x|}$. For every node $z \in T_r$ such that $r(z)$ is a parity state, there exists a unique prefix x such that $r(x) \in [r(z)]$ and either $x = \varepsilon$ or $x = y \cdot c$ with $r(y) \notin [r(z)]$; that is, x is the first node labeled with a state in $[r(z)]$ that is visited along the path from the root to z. We say that x is a *seed*. If we consider, among the many runs of co-Büchi automata embedded in $\langle T_r, r \rangle$, only runs that start in seeds, then each node z for which $r(z)$ is a parity state belongs to exactly one run. Since the size of all parity sets (and thus also the size of the state space of the co-Büchi automata) is ρ, we can associate with each such node z with seed x a rank in $[2\rho]$, corresponding to the rank of the vertex $\langle r(z), |z| - |x| \rangle$ in the induced accepting run of $\mathcal{A}_{r(x)}$ on $w^{|x|}$.

Recall that we want to translate parity alternating automata to weak alternating automata. Thus, we want to start with a WPAA with $S = \emptyset$, go through a chain of WPAA of decreasing degrees, and end-up with a WPAA with $P = \emptyset$. Defining the intermediate automata, it is convenient to alternate between parity and co-parity acceptance conditions. A *co-weak-parity alternating automaton* (*co-WPAA*, for short) is $\mathcal{A} = \langle \Sigma, S, P, q_{in}, \delta, \alpha, \beta \rangle$, with the same structure as a WPAA, only that an infinite path η that gets trapped within a set Q satisfies the acceptance condition iff either Q is an accepting set, or Q is a parity set and η satisfies the co-parity condition β. Proceeding from a WPAA with degree $k > 1$ to its successor in the chain is described in the theorem below.

Theorem 4.1 *Let $\mathcal{A}$ be a weak-parity alternating automaton of degree $k > 1$ and width ρ, with m simple states and n parity states. There is a co-weak-parity alternating automaton $\mathcal{A}'$ of degree $k - 1$ and width ρ, such that $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})$, the number of simple states in $\mathcal{A}'$ is $m + n(\rho + 1)$, and the number of parity states in $\mathcal{A}'$ is $n\rho$.*

Proof: Let $\mathcal{A} = \langle \Sigma, S, P, q_{in}, \delta, \alpha, \beta \rangle$, with $\beta = \{F_1, \ldots, F_k\}$. We define $\mathcal{A}' = \langle \Sigma, S', P', q'_{in}, \delta', \alpha', \beta' \rangle$, where

- $S' = S \cup (P \times [2\rho]^{even})$. That is, the simple states of $\mathcal{A}'$ are the simple states of $\mathcal{A}$, with no annotation, and the parity states of $\mathcal{A}$ annotated with even ranks in $[2\rho]$.
- $P' = P \times [2\rho]^{odd}$. That is, the parity states of $\mathcal{A}'$ are parity states of $\mathcal{A}$ annotated with odd ranks in $[2\rho]$.
- If $q_{in} \in S$, then $q'_{in} = q_{in}$. Otherwise, $q'_{in} = \langle q_{in}, 2\rho \rangle$.
- We define δ' by means of two functions (both parameterized with ρ).

  $$annotate_\rho : \mathcal{B}^+(S \cup P) \to \mathcal{B}^+(S' \cup P')$$

  and

  $$release_\rho : \mathcal{B}^+(S \cup P) \times [2\rho] \times 2^P \to \mathcal{B}^+(S' \cup P').$$

  For a formula $\theta \in \mathcal{B}^+(S \cup P)$, the formula $annotate_\rho(\theta)$ is obtained from θ by replacing an atom $q \in P$ by the disjunction $\bigvee_{j \in [2\rho]} \langle q, j \rangle$. For example, if $\{p, p'\} \subseteq P$ and $s \in S$, then

  $$annotate_4((s \vee p) \wedge p') = (s \vee \bigvee_{j \in [8]} \langle p, j \rangle) \wedge \bigvee_{j \in [8]} \langle p', j \rangle.$$

  For a formula $\theta \in \mathcal{B}^+(S \cup P)$, a rank $i \in [2\rho]$, and a set $Q \subseteq P$, the formula $release_\rho(\theta, i, Q)$ is obtained from θ by replacing an atom $q \in Q$ by the disjunction $\bigvee_{j \in [i]} \langle q, j \rangle$, and replacing an atom $q \in P \setminus Q$ by the disjunction $\bigvee_{j \in [2\rho]} \langle q, j \rangle$. For example,

  $$release_4((s \vee p) \wedge p', 2, \{p\}) = (s \vee \bigvee_{j \in [2]} \langle p, j \rangle) \wedge \bigvee_{j \in [8]} \langle p', j \rangle.$$

  Note that in $annotate_\rho(\theta)$ atoms in P can be annotated by any rank in $[2\rho]$. On the other hand, in $release_\rho(\theta, i, Q)$ some atoms in P (these in Q) can be annotated only by ranks in $[i]$. Now, $\delta' : (S' \cup P') \times \Sigma \to \mathcal{B}^+(S' \cup P')$ is defined as follows.

- For a state $q \in S$ and $\sigma \in \Sigma$, we have $\delta'(q,\sigma) = \mathit{annotate}_\rho(\delta(q,\sigma))$.
- For a state $\langle q, i\rangle \in P \times [2\rho]$ and $\sigma \in \Sigma$, the transition $\delta'(\langle q, i\rangle, \sigma)$ is:
  - $\mathit{release}_\rho(\delta(q,\sigma), i, [q])$, if $q \notin F_1$ or i is even.
  - **false**, if $q \in F_1$ and i is odd.

That is, states that originate from $\mathcal{A}$'s simple states follow their transitions in $\mathcal{A}$, allowing the run to move in its successors that belong to parity sets of $\mathcal{A}$ to any rank in $[2\rho]$. On the other hand, states that originate from $\mathcal{A}$'s parity states follow the transitions of the weak alternating automata that correspond to the co-Büchi automata induced by their parity sets. Intuitively, annotating a state $q \in P$ by a rank in $[2\rho]$ corresponds to guessing its rank in an accepting run of the co-Büchi alternating automaton with state space $[q]$. The initial state and states that are reachable by a transition from states that originate from $\mathcal{A}$'s simple states label nodes that are seeds. Therefore, we can annotate them with any rank in $[2\rho]$ (the rank of a seed is independent of the rank of its predecessor in the run). On the other hand, states that are reachable by a transition from states that originate from $\mathcal{A}$'s parity states label nodes that are not seeds. Therefore, the guessed rank of such a state is bounded by the rank of the state labeling its predecessor.

- $\alpha' = \alpha$. That is, getting trapped in a set Q of simple states, a path is accepting if Q is an accepting set of $\mathcal{A}$, and is rejecting if Q is either a rejecting set of $\mathcal{A}$ or it corresponds to a copy of a parity set of $\mathcal{A}$ annotated with an even rank. Indeed, such sets enable infinitely many visits to F_1.
- $\beta' = \{F_2 \times [2\rho]^{odd}, F_3 \times [2\rho]^{odd}, \ldots, F_k \times [2\rho]^{odd}\}$. That is, getting trapped in a parity set of $\mathcal{A}'$, a path should satisfy the co-parity condition obtained from β by taking out F_1 and annotating the other sets by odd ranks in $[2\rho]$.

We first prove that $\mathcal{A}'$ is a co-WPAA of width ρ. The partition of $S' \cup P'$ into sets is as follows. First, each accepting of rejecting set $Q \subseteq S$ in $\mathcal{A}$ yields the set Q in $\mathcal{A}'$. In addition, each parity set $Q \subseteq P$ in $\mathcal{A}$ yields $2\rho + 1$ sets, $Q \times \{j\}$ for $j \in [2\rho]$, in $\mathcal{A}'$. Clearly, in both cases, the size of the sets in $\mathcal{A}'$ is the same as their size in $\mathcal{A}$, thus $\mathcal{A}'$ is of width ρ. It is easy to see that each of the sets of $\mathcal{A}'$ is either accepting, rejecting, or parity. The partial order on the collection of sets in $\mathcal{A}'$ is induced by the partial order in $\mathcal{A}$. For two sets O and O' in $\mathcal{A}'$, we have $O < O'$ iff there exist sets Q and Q' in $\mathcal{A}$ such that one of the following hold:

- $Q < Q'$ and the following both hold:
  - $O = Q$ or $O = Q \times \{j\}$ for some $j \in [2\rho]$, and
  - $O' = Q'$ or $O' = Q' \times \{j'\}$ for some $j' \in [2\rho]$.
- $Q = Q'$ and the following both hold:
  - $O = Q \times \{j\}$ for some $j \in [2\rho]$,
  - $O' = Q' \times \{j'\}$ for some $j' \in [2\rho]$, and
  - $j < j'$.

It is easy to see that transitions from a state q in $\mathcal{A}'$ leads to states q' for which $[q'] \leq [q]$, thus the structural conditions for a co-WPAA hold.

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 $(S \cup P)$-labeled tree $\langle T_r, r\rangle$ where for all $x \in T_r$ with $r'(x) = q$ or $r'(x) = \langle q, i\rangle$, we have $r(x) = q$. Thus, $\langle T_r, r\rangle$ projects the labels of $\langle T_r, r'\rangle$ on their $S \cup P$ 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 (or replace them by **false**, which cannot be taken in an accepting run of $\mathcal{A}'$). We show that $\langle T_r, r\rangle$ is an accepting run. Consider an infinite path $\pi \subseteq T_r$. Since $\mathcal{A}'$ is a co-WPAA and $\langle T_r, r'\rangle$ is accepting, there are two possible fates for π in $\langle T_r, r'\rangle$:

1. It gets trapped in an accepting set. Then, as $\alpha' = \alpha$, it must be that in $\langle T_r, r\rangle$, the path π gets trapped in an accepting set as well.
2. It gets trapped in a parity set and satisfies the co-parity condition β'. Then, as the parity sets are $P \times \{i\}$ for some odd i, it is guaranteed, by the definition of δ' (where no run can visit a state $\langle q, i\rangle$ with an odd i and $q \in F_1$), that π actually gets trapped in the subset $(P \setminus F_1) \times \{i\}$ of $P \times \{i\}$. Hence, it must be that in $\langle T_r, r\rangle$, the path π gets trapped in a parity set and satisfies β.

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 a memoryless accepting run of $\mathcal{A}$ on w. Consider the $(S' \cup P')$-labeled tree $\langle T_r, r'\rangle$ where where for every node $x \in T_r$, we have

$$r'(x) = \begin{cases} r(x) & \text{If } r(x) \in S. \\ \langle r(x), 2\rho\rangle & \text{If } r(x) \in P \text{ and } x = \varepsilon. \\ \langle r(x), \mathit{rank}(x)\rangle & \text{If } r(x) \in P \text{ and } x \neq \varepsilon. \end{cases}$$

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} is either q_{in}, in the case $q_{in} \in S$, or $\langle q_{in}, 2\rho\rangle$, in the case $q_{in} \in P$, we have that $r'(\varepsilon) = q'_{in}$, thus the root of the tree $\langle T_r, r'\rangle$ is labeled legally. We now consider the first level of $\langle T_r, r'\rangle$. Consider the state q'_{in}. Note that for every $\sigma \in \Sigma$ and $Q \subseteq P$, we have $\mathit{annotate}_\rho(\delta(q_{in}, \sigma)) = \mathit{release}_\rho(\delta(q_{in}, \sigma), 2\rho, Q)$. Hence, by the definition of δ', we have (independently of whether $q'_{in} = q_{in}$ or $q'_{in} = \langle q_{in}, 2\rho\rangle$) that $\delta'(q'_{in}, \sigma_0)$ is obtained from $\delta(q_{in}, \sigma_0)$ by replacing an atom $q \in P$ by the disjunction $\bigvee_{j\in[2\rho]}\langle q, j\rangle$. Let $[k]$ be the set of ε's successors in T_r and let $Q = \langle q_0, \ldots, q_k\rangle$ be the set of labels of ε's successors in $\langle T_r, r\rangle$. Thus, Q satisfies $\delta(q_{in}, \sigma_0)$. (We refer to Q as an ordered set, so it may contain repetitions.) Consider the set $Q' = \langle q'_0, \ldots, q'_k\rangle$, where for all $c \in [k]$, we have $q'_c = q_c$ in the case $q_c \in S$, and $q'_c = \langle q_c, \mathit{rank}(c)\rangle$ in the case $q_c \in P$. As 2ρ is the maximal rank that a node can get, each successor c of ε in T_r has $\mathit{rank}(c) \leq 2\rho$. Therefore, the set Q' satisfies $\delta'(q'_{in}, \sigma_0)$. 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 $x \neq \varepsilon$. Let $[x{\cdot}0, \ldots, x{\cdot}k]$ be the set of x's successors in T_r and let $Q = \langle q_0, \ldots, q_k\rangle$ be the set of labels of x's successors in $\langle T_r, r\rangle$. Consider the set $Q' = \langle q'_0, \ldots, q'_k\rangle$, where for all $c \in [k]$, we have $q'_c = q_c$ in the case $q_c \in S$, and $q'_c = \langle q_c, \mathit{rank}(x \cdot c)\rangle$ in the case $q_c \in P$. We claim that Q' satisfies $\delta'(r'(x), \sigma_{|x|})$. To prove this, we distinguish between two cases. Assume first that $r'(x) = q \in S$. Then, $\delta'(r'(x), \sigma_{|x|}) = \mathit{annotate}_\rho(\delta(q, \sigma_{|x|}))$. As 2ρ is the maximal rank that a node can get, each successor $x \cdot c$ of x in T_r has $\mathit{rank}(x \cdot c) \leq 2\rho$, and we are done. Assume now that $r'(x) = \langle q, i\rangle \in P \times [2\rho]$. Then, by the definition of r', we have that $i = \mathit{rank}(x)$. Since r is accepting, then, by the definition of ranks, it cannot be that $q \in F_1$ and i is odd. Hence, $\delta'(r'(x), \sigma_{|x|}) = \mathit{release}_\rho(\delta(q, \sigma_{|x|}), i, [q])$. Since for each atom $q_c \in [q]$, we have, by Lemma 3.1, that $\mathit{rank}(x \cdot c) \leq \mathit{rank}(x)$, and since for each atom $q_c \in P \setminus [q]$, we have $\mathit{rank}(x{\cdot}c) \leq 2\rho$, the set Q' satisfies $\mathit{release}_\rho(\delta(q, \sigma_{|x|}), i, [q])$, and we are done. Hence, the tree $\langle T_r, r'\rangle$ is a run of $\mathcal{A}'$ on w.

Finally, to see that $\langle T_r, r'\rangle$ is accepting, consider an infinite path $\pi \subseteq T_r$. Since $\mathcal{A}$ is a WPAA and $\langle T_r, r\rangle$ is accepting, there are two possible fates for π in $\langle T_r, r\rangle$:

1. It gets trapped in an accepting set. Then, for all $x \in \pi$ we have $r(x) = r'(x)$ and therefore, as $\alpha' = \alpha$, it must be that in $\langle T_r, r'\rangle$, the path π gets trapped in an accepting set as well.

2. It gets trapped in a parity set and satisfies β. Then, it visits F_1 only finitely often, which implies, according to Lemma 3.2, that in $\langle T_r, r'\rangle$, the path π gets trapped in a parity set $P \times \{i\}$ for some odd i. In addition, since π satisfies the parity condition β in $\langle T_r, r\rangle$, it satisfies the co-parity condition β' as well.

□

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 γ is a condition that accepts exactly all the words that are not accepted by γ. In particular, when we dualize a WPAA, we get a co-WPAA. Consider a co-WPAA $\mathcal{A}$. Let $\tilde{\mathcal{A}}$ be its dual WPAA, and let $\tilde{\mathcal{A}}'$ be the co-WPAA constructed from $\tilde{\mathcal{A}}$ in Theorem 4.1. By dualizing $\tilde{\mathcal{A}}'$, we obtain a WPAA that is equivalent to $\mathcal{A}$. Hence, the construction in Theorem 4.1 can be used also to go from a co-WPAA of degree $k > 1$ to a WPAA of degree $k - 1$.

Recall that a parity automaton $\mathcal{A}$ with n states can be viewed as a WPAA with $S = \emptyset$ and width n. By repeatedly employing the construction in Theorem 4.1 (and its dual construction), we can translate $\mathcal{A}$ to a WPAA or a co-WPAA $\mathcal{A}'$ of degree 1. Such an automaton, however, can be viewed as a WAA. Indeed, its parity sets are either rejecting, in the case $\mathcal{A}'$ is a WPAA, or accepting, in the case $\mathcal{A}'$ is a co-WPAA. We can therefore conclude with the following theorem.

Theorem 4.2 *Let $\mathcal{A}$ be a parity alternating automaton with n states and degree k. 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 $O(n^k)$.

Proof: Given $\mathcal{A}$, consider the chain $\mathcal{A}_0, \ldots, A_{k-1}$ of WPAA and co-WPAA, where $A_0 = \mathcal{A}$, and $\mathcal{A}_{i+1}$ is obtained from $\mathcal{A}_i$ following the construction in Theorem 4.1 (or its dual construction). For $i \in [k]$, let m_i and n_i denote the number of simple and parity states in $\mathcal{A}_i$, respectively. In particular, $m_0 = 0$ and $n_0 = n$. As the width ρ of all the automata in the chain is n, then for $i \in [k-2]$, we have $m_{i+1} = m_i + n_i \cdot (n+1)$, and $n_{i+1} = n_i \cdot n$. Hence, $n_i = n^{i+1}$. Accordingly, $m_i = \frac{n \cdot (n+1) \cdot (n^i - 1)}{n-1}$. In particular, both n_{k-1} and m_{k-1} are $O(n^k)$. The automaton $\mathcal{A}_{k-1}$ is of degree 1, and can therefore be viewed as a WAA with $n_{k-1} + m_{k-1}$ states. Hence, we are done. □

A *Rabin alternating automaton* is $\mathcal{A} = \langle \Sigma, Q, q_{in}, \delta, \beta \rangle$, where $\beta = \{\langle G_1, B_1 \rangle, \ldots, \langle G_k, B_k \rangle\} \subseteq 2^Q \times 2^Q$. We refer to k (the number of pairs in β) as the *degree* of β (or $\mathcal{A}$). A word $\eta \in Q^\omega$ satisfies a Rabin condition β iff there exists an index i for which $Inf(\eta) \cap G_i \neq \emptyset$ and $Inf(\eta) \cap B_i \neq \emptyset$. Note that a parity condition β with either an even degree k or an odd degree $k+1$ is equivalent to the Rabin condition $\{\langle F_2, F_1 \rangle, \langle F_4, F_3 \rangle, \ldots, \langle F_k, F_{k-1} \rangle\}$. Rabin automata can be translated to parity automata with an exponential blow up [Tho97].

In the full paper we show how the same ideas used for parity automata can be used in order to translate alternating Rabin automata to weak alternating automata. As in the parity case, the construction goes through a sequence of intermediate automata. Each automaton in the sequence refines the state space of its predecessor and has one less pair in its Rabin acceptance condition. Unlike the parity case, where the sets in the acceptance condition are ordered, here there is no order between the pairs the acceptance condition. Therefore, while in the parity case refinement essentially requires n copies of each state, resulting in an $O(n^k)$ overall blow-up, here refinement also requires a guess of the pair to be removed, resulting in an additional $k!$ blow up.

Theorem 4.3 *Let $\mathcal{A}$ be a Rabin alternating automaton with n states and degree k. 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 $O(n^{2k+1} \cdot k!)$.*

Note that while the degree of a parity condition β denotes the number of sets in β, the degree of a Rabin condition β denotes the number of pairs, which is half the number of sets, in β. Thus, the blow ups in Theorems 4.2 and 4.3 indeed differ only in $k!$ (the explanation to the additional $+1$ factor in the blow up in Theorem 4.3 is the fact that a parity condition of an odd degree $2k+1$ is equivalent to a Rabin condition with k pairs).

5 The Nonemptiness Problem

In this section we show how the translations described in Section 4 can be used in order to solve the nonemptiness problem for nondeterministic tree automata. A *nondeterministic tree automaton* is $\mathcal{A} = \langle \Sigma, d, Q, q_{in}, \delta, \alpha \rangle$, where Σ, Q, q_{in}, and α are as in alternating word automata, $d \in \mathbb{N}$ is a branching degree, and $\delta : Q \times \Sigma \to 2^{Q^d}$ is a transition function that maps a state and a letter to a set of d-tuples over Q. The automaton $\mathcal{A}$ runs on infinite Σ-labeled trees $\langle T, V \rangle$ of branching degree d, thus $T = \{1, \ldots, d\}^*$. As with alternating word automata, $\delta(q, \sigma)$ describes a possible configuration that $\mathcal{A}$ can move into when it is in state q and reads the letter σ, which labels the root of the input tree. For example, a transition $\delta(q, \sigma) = \{\langle q_1, q_2 \rangle, \langle q_3, q_4 \rangle\}$ means that $\mathcal{A}$ accepts a binary tree with root labeled σ from state q if it accepts the left subtree from state q_1 and the right subtree from state q_2, or it accepts the left subtree from state q_3 and the right subtree from state q_4. A *run* of $\mathcal{A}$ on $\langle T, V \rangle$ is a Q-labeled tree $\langle T, r \rangle$, such that the following hold:

- $r(\varepsilon) = q_{in}$.
- Let $x \in T$ with $r(x) = q$. There exists $\langle q_1, \ldots, q_d \rangle \in \delta(q, V(x))$ such that for all $1 \leq c \leq d$, we have $r(x \cdot c) = q_c$.

Note that each node of the input tree corresponds to exactly one node in the run tree. A run $\langle T, r \rangle$ is *accepting* iff all its paths satisfy the acceptance condition.

It is shown in [BVW94] that the nonemptiness problem for nondeterministic tree automata and the nonemptiness problem for alternating word automata over a singleton alphabet are equivalent and that their complexities coincide. We refer to both problems as the *nonemptiness problem*. Since the nonemptiness problem for weak alternating automata can be solved in linear time [BVW94], Theorems 4.2 and 4.3 imply the following.

Theorem 5.1

(1) *The nonemptiness problem for parity automata with n states and degree k can be solved in time $O(n^k)$.*

(2) *The nonemptiness problem for Rabin automata with n states and degree k can be solved in time $O(n^{2k+1}\cdot k!)$.*

The $O(n^{2k+1}\cdot k!)$ bound in Theorem 5.1 improves the known $O((nk)^{3k})$ upper-bound for the nonemptiness problem [EJ88, PR89] for Rabin automata. A similar bound for Streett automata follows. Solving, however, the nonemptiness problem by translating a given automaton to an equivalent weak automaton is not very appealing in practice, as such a solution never performs better than its worst-case complexity. Indeed, the blow-up is introduced already in the translation of $\mathcal{A}$ to $\mathcal{A}'$. We now describe an algorithm that uses the special structure of $\mathcal{A}'$ without constructing it first. The worst-case complexity of this algorithm is as above, yet in practice it may perform better. We consider here the case where $\mathcal{A}$ is a parity automaton. The algorithm of Rabin automata follows the same ideas and is described in the full paper.

An *extended parity* automaton is $\mathcal{A}=\langle\Sigma,Q,q_{in},\delta,\alpha\rangle$, where $\alpha=\langle\beta,V,A\rangle$ is an acceptance condition consisting of a parity condition β over Q, a *visiting* set $V\subseteq Q$, and an *avoiding* set $A\subseteq Q$. It is required that $A\cap V=\emptyset$. The extended parity automaton $\mathcal{A}$ is equivalent to the parity automaton $\mathcal{A}'=\langle\Sigma,Q,q_{in},\delta',\beta\rangle$, where δ' is defined, for all $\sigma\in\Sigma$, as follows:

- For all $q\in V$, we have $\delta'(q,\sigma)=\mathbf{true}$.
- For all $q\in A$, we have $\delta'(q,\sigma)=\mathbf{false}$.
- For all $q\in Q\setminus(V\cup A)$, we have $\delta'(q,\sigma)=\delta(q,\sigma)$.

Thus, in a run $\langle T_r,r\rangle$ of $\mathcal{A}$, no node $x\in T_r$ has $r(x)\in A$, a node $x\in T_r$ with $r(x)\in V$ need not have children, and $\langle T_r,r\rangle$ is accepting iff all its infinite paths satisfy the parity condition β. An *extended co-parity* alternating automaton is defined similarly with $\alpha=\langle\beta,V,A\rangle$ for a co-parity condition β. For a parity condition β, let $\sim\beta$ be β when referred to as a co-parity condition.

Let $\mathcal{A}$ be an alternating word automaton with a singleton alphabet Σ, state space Q, and transition function δ. For a generalized parity or co-parity condition α, let $acc_{\mathcal{A}}(\alpha)$ be the set of states in Q for which the automaton $\mathcal{A}^q=\langle\Sigma,Q,q,\delta,\alpha\rangle$ is not empty. Similarly, let $\widetilde{acc}_{\mathcal{A}}(\alpha)$ be the set of states in Q for which the automaton $\mathcal{A}^q=\langle\Sigma,Q,q,\tilde{\delta},\alpha\rangle$ is not empty. Clearly, a parity alternating automaton $\mathcal{A}=\langle\Sigma,Q,q_{in},\delta,\beta\rangle$ is not empty iff $q_{in}\in acc_{\mathcal{A}}(\langle\beta,\emptyset,\emptyset\rangle)$.

Recall that our translation of parity and co-parity alternating automata to WAA proceeds inductively. In each iteration, we remove from the parity or the co-parity condition the minimally indexed set and continue with a refined state space and a dual acceptance condition. The algorithm that follows works similarly. In each iteration, we calculate the set of accepting states in an automaton with an acceptance condition $\{F_1,F_2,\ldots,F_k\}$ by calculating, recursively, the accepting states in a dual automaton with an acceptance condition $\{F_2,\ldots,F_k\}$. Formally, the algorithm employs the following two equivalences.

Reduce:

$$acc_{\mathcal{A}}(\{F_1,F_2,\ldots,F_k\},V,A)=\\ \mu Y.acc_{\mathcal{A}}(\sim\{F_2,\ldots,F_k\},V\cup Y,A\cup F_1\setminus Y),$$

where μ is the least fixed-point operator (see the explanation below).

Dual:

$$acc_{\mathcal{A}}(\sim\beta,V,A)=Q\setminus\widetilde{acc}_{\mathcal{A}}(\beta,A,V).$$

Using **Reduce**, we calculate $acc_{\mathcal{A}}(\{F_1,F_2,\ldots,F_k\},V,A)$ as the least fixed-point of the equation

$$Y=acc_{\mathcal{A}}(\sim\{F_2,\ldots,F_k\},V\cup Y,A\cup F_1\setminus Y).$$

Let $Y_0=\emptyset$ and $Y_{j+1}=acc_{\mathcal{A}}(\sim\{F_2,\ldots,F_k\},V\cup Y_j,A\cup F_1\setminus Y_j)$. Intuitively, the set Y_i, for $i\geq 1$, contains all states q for which there exists an accepting run of $\mathcal{A}^q$ in which all paths either satisfy the co-parity condition $\{F_2,\ldots,F_k\}$, or visit F_1 at most $j-1$ times.

Using **Dual**, we can calculate the set of accepting states in an extended co-parity automaton by complementing the set of accepting states in an extended parity automaton with a dual transition function and acceptance condition.

References

[ALW89] M. Abadi, L. Lamport, and P. Wolper. Realizable and unrealizable concurrent program specifications. In *Proc. 16th Int. Colloquium on Automata, Languages and Programming*, volume 372, pages 1–17. Lecture Notes in Computer Science, Springer-Verlag, July 1989.

[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. Internat. Congr. Logic, Method and Philos. Sci. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.

[BVW94] O. Bernholtz, M.Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In D. L. Dill, editor, *Computer Aided Verification, Proc. 6th Int. Conference*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford, June 1994. Springer-Verlag, Berlin.

[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.

[EJ88] E.A. Emerson and C. Jutla. The complexity of tree automata and logics of programs. In *Proc. 29th IEEE Symposium on Foundations of Computer Science*, pages 368–377, White Plains, October 1988.

[EJ91] E.A. Emerson and C. Jutla. Tree automata, Mu-calculus and determinacy. In *Proc. 32nd IEEE Symposium on Foundations of Computer Science*, pages 368–377, San Juan, October 1991.

[EJS93] E.A. Emerson, C. Jutla, and A.P. Sistla. On model-checking for fragments of μ-calculus. In *Computer Aided Verification, Proc. 5th Int. Conference*, volume 697, pages 385–396, Elounda, Crete, June 1993. Lecture Notes in Computer Science, Springer-Verlag.

[EL86] E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the proposoitional Mu-calculus. In *Proc. 1st Symposium on Logic in Computer Science*, pages 267–278, Cambridge, June 1986.

[ES84] A.E. Emerson and A.P. Sistla. Deciding full branching time logics. *Information and Control*, 61(3):175–201, 1984.

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

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

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

[KV97] O. Kupferman and M.Y. Vardi. Weak alternating automata are not that weak. In *Proc. 5th Israeli Symposium on Theory of Computing and Systems*, pages 147–158. IEEE Computer Society Press, 1997.

[KV98] O. Kupferman and M.Y. Vardi. Freedom, weakness, and determinism: from linear-time to branching-time. In *Proc. 13th IEEE Symposium on Logic in Computer Science*, Indiana, June 1998.

[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.

[Mos84] A.W. Mostowski. Regular expressions for infinite trees and a standard form of automata. In *Computation Theory*, volume 208 of *Lecture Notes in Computer Science*, pages 157–168. Springer-Verlag, 1984.

[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 Int. Colloquium on Automata, Languages and Programming*. Springer-Verlag, 1986.

[Niw88] D. Niwinski. Fixed-points vs. infinite generation. In *Proc. 3rd Symposium on Logic in Computer Science*, 1988.

[PR89] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proc. 16th ACM Symposium on Principles of Programming Languages*, Austin, January 1989.

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

[Rab72] M.O. Rabin. Automata on infinite objects and Church's problem. *Amer. Mathematical Society*, 1972.

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

[Saf92] S. Safra. Exponential determinization for ω-automata with strong-fairness acceptance condition. In *Proc. 24th ACM Symposium on Theory of Computing*, Victoria, May 1992.

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

[Tho97] W. Thomas. Languages, automata, and logic. *Handbook of Formal Language Theory*, III:389–455, 1997.

[Var94] M.Y. Vardi. Nontraditional applications of automata theory. In *Proc. Int'l Symp. on Theoretical Aspects of Computer Software*, volume 789, pages 575–597. Lecture Notes in Computer Science, Springer-Verlag, 1994.

[Var96] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, Berlin, 1996.

[Var97] M.Y. Vardi. Alternating automata – unifying truth and validity checking for temporal logics. In W. McCune, editor, *Proc. 14th International Conference on Automated Deduction*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 191–206. Springer-Verlag, Berlin, july 1997.

[VS85] M.Y. Vardi and L. Stockmeyer. Improved upper and lower bounds for modal logics of programs. In *Proc*

17th ACM Symp. on Theory of Computing, pages 240–251, 1985.

[VW86] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. First Symposium on Logic in Computer Science*, pages 322–331, Cambridge, June 1986.

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