

The Weakness of Self-Complementation

Orna Kupferman^{1*} and Moshe Y. Vardi^{2**}

¹ Hebrew University, The institute of Computer Science, Jerusalem 91904, Israel

Email: orna@cs.huji.ac.il, URL: <http://www.cs.huji.ac.il/~orna>

² Rice University, Department of Computer Science, Houston, TX 77251-1892, U.S.A.

Email: vardi@cs.rice.edu, URL: <http://www.cs.rice.edu/~vardi>

Abstract. Model checking is a method for the verification of systems with respect to their specifications. Symbolic model-checking, which enables the verification of large systems, proceeds by evaluating fixed-point expressions over the system's set of states. Such evaluation is particularly simple and efficient when the expressions do not contain alternation between least and greatest fixed-point operators; namely, when they belong to the *alternation-free μ -calculus* (AFMC). Not all specifications, however, can be translated to AFMC, which is exactly as expressive as *weak monadic second-order logic* (WS2S). Rabin showed that a set \mathcal{T} of trees can be expressed in WS2S if and only if both \mathcal{T} and its complement can be recognized by nondeterministic Büchi tree automata. For the “only if” direction, Rabin constructed, given two nondeterministic Büchi tree automata \mathcal{U} and \mathcal{U}' that recognize \mathcal{T} and its complement, a WS2S formula that is satisfied by exactly all trees in \mathcal{T} . Since the translation of WS2S to AFMC is nonelementary, this construction is not practical. Arnold and Niwiński improved Rabin's construction by a direct translation of \mathcal{U} and \mathcal{U}' to AFMC, which involves a doubly-exponential blow-up and is therefore still impractical. In this paper we describe an alternative and quadratic translation of \mathcal{U} and \mathcal{U}' to AFMC. Our translation goes through *weak alternating tree automata*, and constitutes a step towards efficient symbolic model checking of highly expressive specification formalisms.

1 Introduction

In *model checking*, we verify the correctness of a system with respect to a desired behavior by checking whether a structure that models the system satisfies a formula that specifies this behavior. Commercial model-checking tools need to cope with the exceedingly large state-spaces that are present in real-life designs, making the so-called *state-explosion problem* one of the most challenging areas in computer-aided verification. One of the most important developments in this area is the discovery of *symbolic* model-checking methods [BCM⁺92, McM93]. In particular, the use of BDDs [Bry86] for model representation has yielded model-checking tools that can handle systems with 10^{120} states and beyond [CGL93].

* Part of this work was done when this author was visiting Cadence Berkeley Laboratories.

** Supported in part by the NSF grants CCR-9628400 and CCR-9700061, and by a grant from the Intel Corporation. Part of this work was done when this author was a Varon Visiting Professor at the Weizmann Institute of Science.

Typically, symbolic model-checking tools proceed by computing fixed-point expressions over the model's set of states. For example, to find the set of states from which a state satisfying some predicate p is reachable, the model checker starts with the set S of states in which p holds, and repeatedly add to S the set $\exists \circ S$ of states that have a successor in S . Formally, the model checker calculates the least fixed-point of the expression $S = p \vee \exists \circ S$. The evaluation of such expressions is particularly simple when they contain no alternation between least and greatest fixed-point operators. Formally, the evaluation of expressions in *alternation-free μ -calculus* (AFMC) [Koz83, EL86] can be solved in time that is linear in both the size of the model and the length of the formula [AC88, CS91]. In contrast, the evaluation of expressions in which there is a single alternation takes time that is quadratic in the size of the model. Since the models are very large, the difference with the linear complexity of AFMC is very significant [HKS97]. Hence, it is desired to translate specifications to AFMC. Not all specifications, however, can be translated to AFMC [KV98a], and known translations to AFMC involve a blow-up that makes them impractical. In this paper we describe an alternative translation of specifications to AFMC.

Second-order logic is a powerful formalism for expressing properties of sequences and trees. We can view all common program logics as fragments of second-order logic. Second-order logic also serves as the specification language in the model-checking tool MONA [EKM98, Kla98]. While in first-order logic one can only quantify individual variables, second-order logic enables also the quantification of sets³. For example, the formula

$$\exists X. \epsilon \in X \wedge \forall z(z \in X \leftrightarrow \neg(\text{succ}(z) \in X)) \wedge \forall z(z \in X \rightarrow P(z))$$

specifies sequences in which P holds at all even positions. We distinguish between two types of logic, linear and branching. In second-order logic with one successor (S1S), the formulas describe sequences and contain, as the example above, the successor operator succ . In second-order logic with two successors (S2S), formulas describe trees and contain both left-successor and right-successor operators. For example, the S2S formula

$$\exists X. \epsilon \in X \wedge \forall z(z \in X \rightarrow (P(z) \wedge (\text{succ}_l(z) \in X \vee \text{succ}_r(z) \in X)))$$

specifies trees in which P holds along at least one path.

Second-order logic motivated the introduction and study of *finite automata on infinite objects*. Like automata on finite objects, automata on infinite objects either accept or reject their input. Since a run on an infinite object does not have a final state, acceptance is determined with respect to the set of states visited infinitely often during the run. 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 [Bü62] (when the run is a tree, it is required to visit infinitely many accepting states along each path). More general are *Rabin* automata, whose acceptance conditions involve a set of pairs of sets of states. The tight relation between automata on infinite objects and second-order logic was first established for the linear paradigm. In [Bü62], Büchi translated S1S

³ Thus, we consider *monadic* second-order logic, where quantification is over unary relations

formulas to nondeterministic Büchi word automata. Then, in [Rab69], Rabin translated S2S formulas to nondeterministic Rabin tree automata. These fundamental works led to the solution of the decision problem for S1S and S2S, and were the key to the solution of many more problems in mathematical logic [Tho90].

Recall that we are looking for a fragment of S2S that can be translated to AFMC. Known results about the expressive power of different types of automata enabled the study of definability of properties within fragments of second-order logic. In [Rab70], Rabin showed that nondeterministic Büchi tree automata are strictly less expressive than nondeterministic Rabin tree automata, and that they are not closed under complementation. Rabin also showed that for every set \mathcal{T} of trees, both \mathcal{T} and its complement can be recognized by nondeterministic Büchi tree automata iff \mathcal{T} can be specified in a fragment of S2S, called *weak second-order logic* (WS2S), in which set quantification is restricted to finite sets. For the “only if” direction, Rabin constructed, given two nondeterministic Büchi tree automata \mathcal{U} and \mathcal{U}' that recognize \mathcal{T} and its complement, a WS2S formula that is satisfied by exactly all trees in \mathcal{T} .

It turned out that WS2S is exactly the fragment of S2S we are looking for, thus WS2S=AFMC. In [AN92], Arnold and Niwiński showed that every AFMC formula can be translated to an equivalent WS2S formula. For the other direction, they constructed, given \mathcal{U} and \mathcal{U}' as above, an AFMC formula that is satisfied by exactly all trees accepted by \mathcal{U} . The translation in [AN92] is doubly exponential. Thus, if \mathcal{U} and \mathcal{U}' has n and m states, respectively, the AFMC formula is of length $2^{2^{nm}}$. While this improves the nonelementary translation of Rabin’s WS2S formula to AFMC, it is still not useful in practice. In this paper we present a quadratic translation of \mathcal{U} and \mathcal{U}' to an AFMC formula that is satisfied by exactly all trees accepted by \mathcal{U} . Our translation goes through *weak alternating automata* [MSS86]. Thus, while the characterizations of WS2S in [Rab70, AN92] go from logic to automata and then back to logic, our construction provides a clean, purely automata-theoretic, characterization of WS2S.

In an *alternating* automaton [BL80, CKS81], the transition function can induce both existential and universal requirements on the automaton. For example, a transition $\delta(q, \sigma) = q_1 \vee (q_2 \wedge q_3)$ of an alternating word automaton means that the automaton in state q accepts a word $\sigma \cdot \tau$ iff it accepts the suffix τ either from state q_1 or from both states q_2 and q_3 . 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. It is shown in [MSS86] that formulas of WS2S can be translated to weak alternating tree automata. Moreover, it is shown in [KV98a] that weak alternating automata can be linearly translated to AFMC.

Given two nondeterministic Büchi tree automata \mathcal{U} and \mathcal{U}' that recognize a language and its complement, we construct a weak alternating tree automaton \mathcal{A} equivalent to \mathcal{U} . The number of states in \mathcal{A} is quadratic in the number of states of \mathcal{U} and \mathcal{U}' . Precisely, if \mathcal{U} and \mathcal{U}' has n and m states, respectively, the automaton \mathcal{A} has $(nm)^2$ states. The linear translation of weak alternating tree automata to AFMC then completes a translation to AFMC of the same complexity. Our translation can be viewed

as a step towards efficient symbolic model checking of highly expressive specification formalisms such as the fragment of the branching temporal logic CTL* that can be translated to WS2S. A step that is still missing in order to complete this goal is a translation of CTL* formulas to nondeterministic Büchi tree automata, when such a translation exists. From a theoretical point of view, our translation completes the picture of “quadratic weakening” in both the linear and the branching paradigm. The equivalence in expressive power of nondeterministic Büchi and Rabin word automata [McN66] implied that WS1S is as expressive as S1S [Tho90]. The latter equivalence is supported by an automaton construction: given a nondeterministic Büchi word automaton, one can construct an equivalent weak alternating word automaton of quadratic size [KV97]. In the branching paradigm, WS2S is strictly less expressive than S2S, and a nondeterministic Büchi tree automaton can be translated to a weak alternating tree automaton only if its complement can also be recognized by a nondeterministic Büchi tree automaton. It follows from our construction that the size of the equivalent weak alternating automaton is then quadratic in the sizes of the two automata.

2 Tree Automata

A *full infinite binary tree* (tree) is the set $T = \{\mathbf{l}, \mathbf{r}\}^*$. The elements of T are called *nodes*, and the empty word ε is the *root* of T . For every $x \in T$, the nodes $x \cdot \mathbf{l}$ and $x \cdot \mathbf{r}$ are the *successors* of x . A *path* π of a tree T is a set $\pi \subseteq T$ such that $\varepsilon \in \pi$ and for every $x \in \pi$, exactly one successor of x is in π . For two nodes x_1 and x_2 of T , we say that $x_1 \leq x_2$ iff x_1 is a prefix of x_2 ; i.e., there exists $z \in \{\mathbf{l}, \mathbf{r}\}^*$ such that $x_2 = x_1 \cdot z$. We say that $x_1 < x_2$ iff $x_1 \leq x_2$ and $x_1 \neq x_2$. A *frontier* of an infinite tree is a set $E \subset T$ of nodes such that for every path $\pi \subseteq T$, we have $|\pi \cap E| = 1$. For example, the set $E = \{\mathbf{l}, \mathbf{rll}, \mathbf{rlr}, \mathbf{rr}\}$ is a frontier. For two frontiers E_1 and E_2 , we say that $E_1 \leq E_2$ iff for every node $x_2 \in E_2$, there exists a node $x_1 \in E_1$ such that $x_1 \leq x_2$. We say that $E_1 < E_2$ iff for every node $x_2 \in E_2$, there exists a node $x_1 \in E_1$ such that $x_1 < x_2$. Note that while $E_1 < E_2$ implies that $E_1 \leq E_2$ and $E_1 \neq E_2$, the other direction does not necessarily hold. Given an alphabet Σ , a Σ -*labeled tree* is a pair $\langle T, V \rangle$ where T is a tree and $V : T \rightarrow \Sigma$ maps each node of T to a letter in Σ . We denote by V_Σ the set of all Σ -labeled trees. For a Σ -labeled tree $\langle T, V \rangle$ and a set $A \subseteq \Sigma$, we say that E is an A -frontier iff E is a frontier and for every node $x \in E$, we have $V(x) \in A$.

Automata on infinite trees (tree automata) run on infinite Σ -labeled trees. We first define *nondeterministic Büchi tree automata* (NBT). An NBT is $\mathcal{U} = \langle \Sigma, Q, \delta, q_0, F \rangle$ where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \rightarrow \mathcal{P}^{2 \times Q}$ is a transition function, $q_0 \in Q$ is an initial state, and $F \subseteq Q$ is a Büchi acceptance condition. Intuitively, each pair in $\delta(q, \sigma)$ suggests a nondeterministic choice for the automaton’s next configuration. When the automaton is in a state q as it reads a node x labeled by a letter σ , it proceeds by first choosing a pair $\langle q_{\mathbf{l}}, q_{\mathbf{r}} \rangle \in \delta(q, \sigma)$, and then splitting into two copies. One copy enters the state $q_{\mathbf{l}}$ and proceeds to the node $x \cdot \mathbf{l}$ (the left successor of x), and the other copy enters the state $q_{\mathbf{r}}$ and proceeds to the node $x \cdot \mathbf{r}$ (the right successor of x). Formally, a *run* of \mathcal{U} on an input tree $\langle T, V \rangle$ is a Q -labeled tree $\langle T, r \rangle$, such that the following hold:

- $r(\varepsilon) = q_0$.
- Let $x \in T$ with $r(x) = q$. There exists $\langle q_1, q_r \rangle \in \delta(q, V(x))$ such that $r(x \cdot 1) = q_1$ and $r(x \cdot r) = q_r$.

Note that each node of the input tree corresponds to exactly one node in the run tree.

Given a run $\langle T, r \rangle$ and a path $\pi \subseteq T$, let $\text{inf}(\pi) \subseteq Q$ be such that $q \in \text{inf}(\pi)$ if and only if there are infinitely many $x \in \pi$ for which $r(x) = q$. That is, $\text{inf}(\pi)$ contains exactly all the states that are visited infinitely often in π . A path π satisfies a Büchi acceptance condition $F \subseteq Q$ if and only if $\text{inf}(\pi) \cap F \neq \emptyset$. A run $\langle T, r \rangle$ is *accepting* iff all its paths satisfy the acceptance condition. Equivalently, $\langle T, r \rangle$ is accepting iff $\langle T, r \rangle$ contains infinitely many F -frontiers $G_0 < G_1 < \dots$. A tree $\langle T, V \rangle$ is accepted by \mathcal{U} iff there exists an accepting run of \mathcal{U} on $\langle T, V \rangle$, in which case $\langle T, V \rangle$ belongs to the language, $\mathcal{L}(\mathcal{U})$, of \mathcal{U} . We say that a set \mathcal{T} of trees is in NBT iff there exists an NBT \mathcal{U} such that $\mathcal{L}(\mathcal{U}) = \mathcal{T}$. We say that \mathcal{T} is in co-NBT iff the complement of \mathcal{T} is in NBT; i.e., there exists an NBT \mathcal{U} such that $\mathcal{L}(\mathcal{U}) = V_\Sigma \setminus \mathcal{T}$.

Alternating tree automata generalize nondeterministic tree automata and were first introduced in [MS87]. In order to define alternating tree automata, we first need some notations. 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** and, as usual, \wedge has precedence over \vee . For a set $Y \subseteq X$ and a formula $\theta \in \mathcal{B}^+(X)$, we say that Y *satisfies* θ iff assigning **true** to elements in Y and assigning **false** to elements in $X \setminus Y$ satisfies θ .

A finite alternating automaton over infinite binary trees is $\mathcal{A} = \langle \Sigma, Q, \delta, \emptyset, F \rangle$ where Σ, Q, q_0 , and F are as in NBT, and $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(\{1, r\} \times Q)$ is a transition function. A run of an alternating automaton \mathcal{A} over a tree $\langle T, V \rangle$ is a $(T \times Q)$ -labeled tree $\langle T_r, r \rangle$. The tree T_r is not necessarily binary and it may have states with no successors. Thus, $T_r \subseteq \mathbb{N}^*$ is such that if $x \cdot c \in T_r$ where $x \in \mathbb{N}^*$ and $c \in \mathbb{N}$, then also $x \in T_r$. For every $x \in T_r$, the nodes $x \cdot c$, with $c \in \mathbb{N}$, are the *successors* of x . Each node of T_r corresponds to a node of T . A node in T_r , labeled by (x, q) , describes a copy of the automaton that reads the node x of T and visits the state q . Note that many nodes of T_r can correspond to the same node of T ; in contrast, in a run of a nondeterministic automaton over $\langle T, V \rangle$ there is a one-to-one correspondence between the nodes of the run and the nodes of the tree. The labels of a node and its successors have to satisfy the transition function. Formally, $\langle T_r, r \rangle$ satisfies the following:

1. $\varepsilon \in T_r$ and $r(\varepsilon) = (\varepsilon, q_0)$.
2. Let $y \in T_r$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$. Then there is a possibly empty set $S = \{(c_0, q_0), (c_1, q_1), \dots, (c_n, q_n)\} \subseteq \{1, r\} \times Q$, such that the following hold:
 - S satisfies θ , and
 - for all $0 \leq i \leq n$, we have $y \cdot i \in T_r$ and $r(y \cdot i) = (x \cdot c_i, q_i)$.

For a run $\langle T_r, r \rangle$ and an infinite path $\pi \subseteq T_r$, we define $\text{inf}(\pi)$ to be the set of states that are visited infinitely often in π , thus $q \in \text{inf}(\pi)$ if and only if there are infinitely many $y \in \pi$ for which $r(y) \in T \times \{q\}$. A run $\langle T_r, r \rangle$ is accepting if all its infinite paths satisfy the Büchi acceptance condition. As with NBT, a tree $\langle T, V \rangle$ is accepted by \mathcal{A} iff there exists an accepting run of \mathcal{A} on $\langle T, V \rangle$, in which case $\langle T, V \rangle$ belongs to $\mathcal{L}(\mathcal{A})$.

Example 1. We define an alternating Büchi tree automaton \mathcal{A} that accepts exactly all $\{a, b, c\}$ -labeled binary trees in which all paths have a node labeled a and there exists a path with two successive b labels. Let $\mathcal{A} = \langle \{a, b, c\}, \{q_0, q_1, q_2, q_3\}, \delta, q_0, \emptyset \rangle$, where δ is defined in the table below.

q	$\delta(q, a)$	$\delta(q, b)$	$\delta(q, c)$
q_0	$(0, q_1) \vee (1, q_1)$	$(0, q_3) \wedge (1, q_3) \wedge ((0, q_2) \vee (1, q_2))$	$(0, q_3) \wedge (1, q_3) \wedge ((0, q_1) \vee (1, q_1))$
q_1	$(0, q_1) \vee (1, q_1)$	$(0, q_2) \vee (1, q_2)$	$(0, q_1) \vee (1, q_1)$
q_2	$(0, q_1) \vee (1, q_1)$	true	$(0, q_1) \vee (1, q_1)$
q_3	true	$(0, q_3) \wedge (1, q_3)$	$(0, q_3) \wedge (1, q_3)$

In the state q_0 , the automaton checks both requirements. If a is true, only the second requirement is left to be checked. This is done by sending a copy in state q_1 , which searches for two successive b 's in some branch, to either the left or the right child. If b is true, \mathcal{A} needs to send more copies. First, it needs to check that all paths in the left and right subtrees have a node labeled a . This is done by sending copies in state q_3 to both the left and the right children. Second, it needs to check that one of these subtrees contains two successive b 's. This is done (keeping in mind that the just read b may be the first b in a sequence of two b 's) by sending a copy in state q_2 to one of the children. Similarly, if c is true, \mathcal{A} sends copies that check both requirements. As before, a requirement about a is sent universally and a requirement about the b 's is sent existentially. \square

In [MSS86], Muller et al. introduce *weak alternating tree automata* (AWT). In an AWT, we have a Büchi acceptance condition $F \subseteq Q$ and there exists a partition of Q into disjoint sets, Q_i , such that for each set Q_i , either $Q_i \subseteq F$, in which case Q_i is an *accepting set*, or $Q_i \cap F = \emptyset$, 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 an AWT ultimately gets “trapped” within some Q_i . The path then satisfies the acceptance condition if and only if Q_i is an accepting set. Indeed, a run visits infinitely many states in F if and only if it gets trapped in an accepting set.

2.1 Traps

Let $\mathcal{U} = \langle \Sigma, S, s_0, M, F \rangle$ and $\mathcal{U}' = \langle \Sigma, S', s'_0, M', F' \rangle$ be two NBT, and let $|S| \cdot |S'| = m$. In [Rab70], Rabin studies the composition of a run of \mathcal{U} with a run of \mathcal{U}' . Recall that an accepting run of \mathcal{U} contains infinitely many F -frontiers $G_0 < G_1 < \dots$, and an accepting run of \mathcal{U}' contains infinitely many F' -frontiers $G'_0 < G'_1 < \dots$. It follows that for every tree $\langle T, V \rangle \in \mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}')$ and accepting runs $\langle T, r \rangle$ and $\langle T, r' \rangle$ of \mathcal{U} and \mathcal{U}' on $\langle T, V \rangle$, the composition of $\langle T, r \rangle$ and $\langle T, r' \rangle$ contains infinitely many frontiers $E_i \subset T$, with $E_i < E_{i+1}$, such that $\langle T, r \rangle$ reaches an F -frontier and $\langle T, r' \rangle$ reaches an F' -frontier between E_i and E_{i+1} . Rabin shows that the existence of m such

frontiers, in the composition of some runs of \mathcal{U} and \mathcal{U}' , is sufficient to imply that the intersection $\mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}')$ is not empty. Below we repeat Rabin's result, with some different notations.

Let \mathcal{U} and \mathcal{U}' be as above. We say that a sequence E_0, \dots, E_m of frontiers of T is a *trap for \mathcal{U} and \mathcal{U}'* iff $E_0 = \varepsilon$ and there exists a tree $\langle T, V \rangle$ and runs $\langle T, r \rangle$ and $\langle T, r' \rangle$ of \mathcal{U} and \mathcal{U}' on $\langle T, V \rangle$, such that for every $0 \leq i \leq m - 1$, the following hold.

- $\langle T, r \rangle$ contains an F -frontier G_i such that $E_i \leq G_i < E_{i+1}$, and
- $\langle T, r' \rangle$ contains an F' -frontier G'_i such that $E_i \leq G'_i < E_{i+1}$.

We say that $\langle T, r \rangle$ and $\langle T, r' \rangle$ *witness* the trap for \mathcal{U} and \mathcal{U}' .

Theorem 1. [Rab70] *Consider two nondeterministic Büchi tree automata \mathcal{U} and \mathcal{U}' . If there exists a trap for \mathcal{U} and \mathcal{U}' , then $\mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}')$ is not empty.*

3 From NBT and co-NBT to AWT

Theorem 2. *Let \mathcal{U} and \mathcal{U}' be two NBT with $\mathcal{L}(\mathcal{U}') = V_\Sigma \setminus \mathcal{L}(\mathcal{U})$. There exists an AWT \mathcal{A} such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{U})$ and the size of \mathcal{A} is quadratic in the sizes of \mathcal{U} and \mathcal{U}' .*

Proof. Let $\mathcal{U} = \langle \Sigma, S, s_0, M, F \rangle$ and $\mathcal{U}' = \langle \Sigma, S', s'_0, M', F' \rangle$, and let $|S| \cdot |S'| = m$. We define the AWT $\mathcal{A} = \langle \Sigma, Q, q_0, \delta, \alpha \rangle$ as follows.

- $Q = ((S \times \{\perp, \top\}) \times S' \times \{0, \dots, m\}) \setminus (S \times \{\top\}) \times S' \times \{m\}$. Intuitively, a copy of \mathcal{A} that visits the state $\langle (s, \gamma), s', i \rangle$ as it reads the node x of the input tree corresponds to runs r and r' of \mathcal{U} and \mathcal{U}' that visit the states s and s' , respectively, as they read the node x of the input tree. Let $\rho = y_0, y_1, \dots, y_{|x|}$ be the path from ε to x . Consider the joint behavior of r and r' on ρ . We can represent this behavior by a sequence $\tau_\rho = \langle t_0, t'_0 \rangle, \langle t_1, t'_1 \rangle, \dots, \langle t_{|x|}, t'_{|x|} \rangle$ of pairs in $S \times S'$ where $t_j = r(y_j)$ and $t'_j = r'(y_j)$. We say that a pair $\langle t, t' \rangle \in S \times S'$ is an F -pair iff $t \in F$ and is an F' -pair iff $t' \in F'$. We can partition the sequence τ_ρ to blocks $\beta_0, \beta_1, \dots, \beta_i$ such that we close block β_k and open block β_{k+1} whenever we reach the first F' -pair that is preceded by an F -pair in β_k . In other words, whenever we open a block, we first look for an F -pair, ignoring F' -pairness. Once an F -pair is detected, we look for an F' -pair, ignoring F -pairness. Once an F' -pair is detected, we close the current block and we open a new block. Note that a block may contain a single pair that is both an F -pair and an F' -pair. The number i in $\langle (s, \gamma), s', i \rangle$ is the *index* of the last block in τ_ρ . The *status* $\gamma \in \{\perp, \top\}$ indicates whether the block β_i already contains an F -pair, in which case $\gamma = \top$, or β_i does not contain an F -pair, in which case $\gamma = \perp$.

For a status $\gamma \in \{\perp, \top\}$ and an index $i \in \{0, \dots, m\}$, let $Q_{\gamma,i} = (S \times \{\gamma\}) \times S' \times \{i\}$.

- $q_0 = \langle (s_0, \perp), s'_0, 0 \rangle$.
- In order to define the transition function δ , we first define two functions, *new* $\gamma : Q \setminus Q_{\perp,m} \rightarrow \{\perp, \top\}$ and *new* $i : Q \setminus Q_{\perp,m} \rightarrow \{0, \dots, m\}$, as follows.

$$\text{new}\gamma(\langle(s, \gamma), s', i\rangle) = \begin{cases} \top & \text{If } s' \notin F' \text{ and } (\gamma = \top \text{ or } s \in F). \\ \perp & \text{Otherwise.} \end{cases}$$

$$\text{new}i(\langle(s, \gamma), s', i\rangle) = \begin{cases} i + 1 & \text{If } s' \in F' \text{ and } (\gamma = \top \text{ or } s \in F). \\ i & \text{Otherwise.} \end{cases}$$

Intuitively, $\text{new}\gamma$ and $\text{new}i$ are responsible for the recording and tracking of blocks. Recall that the status γ indicates whether an F -pair in the current block has already been detected. As such, the new status is \top whenever s is in F or γ is \top , unless s' is in F' , in which case $\langle s, s' \rangle$ is the last pair in the current block, and the new status is \perp . Similarly, the index i is increased to $i + 1$ whenever we detect an F' -pair that is either also an F -pair or, as indicated by γ , preceded by an F -pair in the same block.

The automaton \mathcal{A} proceeds as follows. Essentially, for every run $\langle T, r' \rangle$ of \mathcal{U}' , the automaton \mathcal{A} guesses a run $\langle T, r \rangle$ of \mathcal{U} such that for every path ρ of T , the run $\langle T, r \rangle$ visits F along ρ at least as many times as $\langle T, r' \rangle$ visits F' along ρ . Since $\mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}') = \emptyset$, no run $\langle T, r \rangle$ can witness with $\langle T, r' \rangle$ a trap for \mathcal{U} and \mathcal{U}' . Consequently, recording of visits to F and F' along ρ can be completed once \mathcal{A} detects that τ_ρ contains m blocks as above.

Formally, let $q = \langle(s, \gamma), s', i\rangle$ be such that $M(s, \sigma) = \{\langle u_1, v_1 \rangle, \dots, \langle u_n, v_n \rangle\}$ and $M'(s', \sigma) = \{\langle u'_1, v'_1 \rangle, \dots, \langle u'_{n'}, v'_{n'} \rangle\}$. We distinguish between two cases.

- If $q \notin Q_{\perp, m}$, then $\delta(q, \sigma)$ is

$$\bigwedge_{1 \leq k \leq n'} \bigvee_{1 \leq p \leq n} (\mathbf{1}, \langle(u_p, \text{new}\gamma(q)), u'_k, \text{new}i(q)\rangle) \wedge (\mathbf{r}, \langle(v_p, \text{new}\gamma(q)), v'_k, \text{new}i(q)\rangle).$$

- If $q \in Q_{\perp, m}$, then $\delta(q, \sigma)$ is

- * $\bigwedge_{1 \leq k \leq n'} \bigvee_{1 \leq p \leq n} (\mathbf{1}, \langle(u_p, \perp), u'_k, m\rangle) \wedge (\mathbf{r}, \langle(v_p, \perp), v'_k, m\rangle)$, if $s \notin F$.
- * **true**, otherwise.

- $\alpha = (S \times \{\top\}) \times S' \times \{0, \dots, m-1\}$. Thus, α makes sure that infinite paths of the run visits infinitely many states in which the status is \top .

The automaton \mathcal{A} is indeed an AWT. Clearly, each set $Q_{\gamma, i}$ is either contained in α or is disjoint from α . The partial order on the sets is defined by $Q_{\gamma', i'} \leq Q_{\gamma, i}$ iff either $i < i'$, or $i = i'$ and $\gamma' = \top$. Note that, by the definition of α , a run is accepting iff no path of it gets trapped in a set of the form $Q_{\perp, i}$, namely a set in which \mathcal{A} is waiting for a visit of \mathcal{U} in a state in F . The size of \mathcal{A} is $O(m^2)$.

We prove that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{U})$. We first prove that $\mathcal{L}(\mathcal{U}) \subseteq \mathcal{L}(\mathcal{A})$. Consider a tree $\langle T, V \rangle$. With every run $\langle T, r \rangle$ of \mathcal{U} on $\langle T, V \rangle$ we can associate a single run $\langle T_R, R \rangle$ of \mathcal{A} on $\langle T, V \rangle$. Intuitively, the run $\langle T, r \rangle$ directs $\langle T_R, R \rangle$ in the only nondeterminism in δ . Formally, recall that a run of \mathcal{A} on a tree $\langle T, V \rangle$ is a $(T \times Q)$ -labeled tree $\langle T_R, R \rangle$, where a node $y \in T_R$ with $R(y) = \langle x, q \rangle$ corresponds to a copy of \mathcal{A} that reads the node $x \in T$ and visits the state q . We define $\langle T_R, R \rangle$ as follows.

- $\varepsilon \in T_R$ and $R(\varepsilon) = (\varepsilon, \langle(s_0, \perp), s'_0, 0\rangle)$.

- Consider a node $y \in T_R$ with $R(y) = (x, \langle (s, \gamma), s', i \rangle)$. By the definition of $\langle T_R, R \rangle$ so far, we have $r(x) = s$. Let $r(x \cdot \mathbf{1}) = u$ and $r(x \cdot \mathbf{r}) = v$. Also, let $M'(s', \sigma) = \{\langle u'_1, v'_1 \rangle, \dots, \langle u'_{n'}, v'_{n'} \rangle\}$, $\gamma' = \text{new}\gamma(\langle (s, \gamma), s', i \rangle)$, and $i' = \text{new}i(\langle (s, \gamma), s', i \rangle)$. We define $S = \{\langle \mathbf{1}, \langle (u, \gamma'), u'_1, i' \rangle \rangle, \langle \mathbf{r}, \langle (v, \gamma'), v'_1, i' \rangle \rangle, \dots, \langle \mathbf{1}, \langle (u, \gamma'), u'_{n'}, i' \rangle \rangle, \langle \mathbf{r}, \langle (v, \gamma'), v'_{n'}, i' \rangle \rangle\}$. By the definition of δ , the set S satisfies $\delta(\langle (s, \gamma), s', i \rangle, V(x))$. For all $0 \leq j \leq n' - 1$, we have $y \cdot 2j \in T_R$ with $R(y \cdot 2j) = (x \cdot \mathbf{1}, \langle (u, \gamma'), u'_j, i' \rangle)$, and $y \cdot 2j + 1 \in T_R$ with $R(y \cdot 2j + 1) = (x \cdot \mathbf{r}, \langle (v, \gamma'), v'_j, i' \rangle)$.

Consider a tree $\langle T, V \rangle \in \mathcal{L}(\mathcal{U})$. Let $\langle T, r \rangle$ be an accepting run of \mathcal{U} on $\langle T, V \rangle$, and let $\langle T_R, R \rangle$ be the run of \mathcal{A} on $\langle T, V \rangle$ induced by $\langle T, r \rangle$. It is easy to see that $\langle T_R, R \rangle$ is accepting. Indeed, as $\langle T, r \rangle$ contains infinitely many F -frontiers, no infinite paths of $\langle T_R, R \rangle$ can get trapped in a set $Q_{\perp, i}$.

It is left to prove that $\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{U})$. For that, we prove that $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{U}') = \emptyset$. Since $\mathcal{L}(\mathcal{U}) = V_{\Sigma} \setminus \mathcal{L}(\mathcal{U}')$, it follows that every tree that is accepted by \mathcal{A} is also accepted by \mathcal{U} . Consider a tree $\langle T, V \rangle$. With each run $\langle T_R, R \rangle$ of \mathcal{A} on $\langle T, V \rangle$ and run $\langle T, r' \rangle$ of \mathcal{U}' on $\langle T, V \rangle$, we can associate a run $\langle T, r \rangle$ of \mathcal{U} on $\langle T, V \rangle$. Intuitively, $\langle T, r \rangle$ makes the choices that $\langle T_R, R \rangle$ has made in its copies that correspond to the run $\langle T, r' \rangle$. Formally, we define $\langle T, r \rangle$ as follows.

- $r(\varepsilon) = s_0$.
- Consider a node $x \in T$ with $r(x) = s$. Let $r'(x) = s'$. The run $\langle T, r' \rangle$ fixes a pair $\langle u', v' \rangle \in M'(s', V(x))$ that \mathcal{U}' proceeds with when it reads the node x . Formally, let $\langle u', v' \rangle$ be such that $r'(x \cdot \mathbf{1}) = u'$ and $r'(x \cdot \mathbf{r}) = v'$. By the definition of $r(x)$ so far, the run $\langle T_R, R \rangle$ contains a node $y \in T_R$ with $R(y) = \langle x, \langle (s, \gamma), s', i \rangle \rangle$ for some γ and i . If $\delta(\langle (s, \gamma), s', i \rangle, V(x)) = \text{true}$, we define the remainder of $\langle T, r \rangle$ arbitrarily. Otherwise, by the definition of δ , the successors of y in T_R fix the pair in $M(s, V(x))$ that \mathcal{A} proceeds with per each pair in $M'(s', V(x))$. In particular, T_R contains at least two nodes $y \cdot c_1$ and $y \cdot c_2$ such that $R(y \cdot c_1) = \langle x \cdot \mathbf{1}, \langle (u, \gamma'), u', i' \rangle \rangle$ and $R(y \cdot c_2) = \langle x \cdot \mathbf{r}, \langle (v, \gamma'), v', i' \rangle \rangle$, for some γ' and i' . We then define $r(x \cdot \mathbf{1}) = u$ and $r(x \cdot \mathbf{r}) = v$.

We can now prove that $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{U}') = \emptyset$. Assume, by way of contradiction, that there exists a tree $\langle T, V \rangle$ such that $\langle T, V \rangle$ is accepted by both \mathcal{A} and \mathcal{U}' . Let $\langle T_R, R \rangle$ and $\langle T, r' \rangle$ be the accepting runs of \mathcal{A} and \mathcal{U}' on $\langle T, V \rangle$, respectively, and let $\langle T, r \rangle$ be the run of \mathcal{U} on $\langle T, V \rangle$ induced by $\langle T_R, R \rangle$ and $\langle T, r' \rangle$. We claim that then, $\langle T, r \rangle$ and $\langle T, r' \rangle$ witness a trap for \mathcal{U} and \mathcal{U}' . Since, however, $\mathcal{L}(\mathcal{U}) \cap \mathcal{L}(\mathcal{U}') = \emptyset$, it follows from Theorem 1, that no such trap exists, and we reach a contradiction. To see that $\langle T, r \rangle$ and $\langle T, r' \rangle$ indeed witness a trap, define $E_0 = \varepsilon$, and define, for $0 \leq i \leq m - 1$, the set E_{i+1} to contain exactly all nodes x for which there exists $y \in T_R$ with $R(y) = \langle x, \langle (r(x), \gamma), r'(x), i \rangle \rangle$ and $\text{new}i(\langle (r(x), \gamma), r'(x), i \rangle) = i + 1$. That is, for every path ρ of T , the set E_{i+1} consists of the nodes in which the i 'th block is closed in τ_ρ . By the definition of δ , for all $0 \leq i \leq m - 1$, the run $\langle T, r \rangle$ contains an F -frontier G_i such that $E_i \leq G_i < E_{i+1}$ and the run $\langle T, r' \rangle$ contains an F' -frontier G'_i such that $E_i \leq G'_i < E_{i+1}$. Hence, E_0, \dots, E_m is a trap for \mathcal{U} and \mathcal{U}' . \square

4 Discussion

Today, automata on infinite objects are used for specification and verification of non-terminating 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, BVW94, Kur94, VW94]. The automata-theoretic approach separates the logical and the combinatorial aspects of verification. The translation of specifications to automata handles the logic and shifts all the combinatorial difficulties to automata-theoretic problems. There are many types of automata, and choosing the appropriate type for the application is important.

We believe that weak alternating automata are often a good choice. The special structure of weak alternating automata is reflected in their attractive computational properties. For example, while the best known complexity for solving the 1-letter emptiness problem for Büchi alternating automata is quadratic time, we know how to solve the problem for weak alternating automata in linear time [BVW94]. In addition, weak alternating automata can be very easily complemented. In the linear paradigm, where $WS1S=S1S$, weak alternating word automata (AWW) can recognize all the ω -regular languages. In particular, the translation of LTL formulas to AWW is linear, and follows the syntax of the formula [Var96]. Moreover, it is known how to translate other types of automata to AWW efficiently [KV97, KV98b]. In the branching paradigm, where $WS2S<S2S$, AWT can recognize exactly all specifications that can be efficiently checked symbolically. The translation of CTL and AFMC formulas to AWT is linear and simple [BVW94]. As we have seen in this paper, the translation of two NBT for a specification and its complementation to AWT involves only a quadratic blow up. In particular, we believe that model-checking tools like MONA [EKM98, Kla98], which have $WS1S$ and $WS2S$ as their specification languages, may benefit from employing weak alternating automata.

References

- [AC88] A. Arnold and P. Crubille. A linear algorithm to solve fixed-point equations. *Information Processing Letters*, 29(2):57–66, September 1988.
- [AN92] A. Arnold and D. Niwiński. Fixed point characterization of weak monadic logic definable sets of trees. In *Tree Automata and Languages*, pp. 159–188, 1992. Elsevier.
- [BCM⁺92] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, 1992.
- [BL80] J.A. Brzozowski and E. Leiss. Finite automata and sequential networks. *Theoretical Computer Science*, 10:19–35, 1980.
- [Bry86] R.E. Bryant. Graph-based algorithms for boolean-function manipulation. *IEEE Trans. on Computers*, C-35(8), 1986.
- [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, 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.
- [CGL93] E.M. Clarke, O. Grumberg, and D. Long. Verification tools for finite-state concurrent systems. In *Decade of Concurrency – Reflections and Perspectives (Proceedings of REX School)*, LNCS 803, pages 124–175, 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.
- [CS91] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal μ -calculus. In *Proc. 3rd CAV*, LNCS 575, pages 48–58, 1991.
- [EKM98] J. Elgaard, N. Klarlund, and A. Möller. Mona 1.x: new techniques for WS1S and WS2S. In *Proc 10th CAV*, LNCS 1427, pages 516–520, 1998.
- [EL86] E.A. Emerson and C.-L. Lei. Efficient model checking in fragments of the propositional μ -calculus. In *Proc. 1st LICS*, pages 267–278, 1986.
- [HKSV97] R.H. Hardin, R.P. Kurshan, S.K. Shukla, and M.Y. Vardi. A new heuristic for bad cycle detection using BDDs. In *Proc. 9th CAV*, LNCS 1254, pages 268–278, 1997.
- [Kla98] N. Klarlund. Mona & Fido: The logic-automaton connection in practice. In *Computer Science Logic, '97*, Lecture Notes in Computer Science, 1998.
- [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 ISTCS*, pages 147–158. IEEE Computer Society Press, 1997.
- [KV98a] O. Kupferman and M.Y. Vardi. Freedom, weakness, and determinism: from linear-time to branching-time. In *Proc. 13th LICS*, pages 81–92, 1998.
- [KV98b] O. Kupferman and M.Y. Vardi. Weak alternating automata and tree automata emptiness. In *Proc. 30th STOC*, pages 224–233, 1998.
- [McM93] K.L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
- [McN66] R. McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966.
- [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.
- [Rab69] M.O. Rabin. Decidability of second order theories and automata on infinite trees. *Transaction of the AMS*, 141:1–35, 1969.
- [Rab70] M.O. Rabin. Weakly definable relations and special automata. In *Proc. Symp. Math. Logic and Foundations of Set Theory*, pages 1–23. North Holland, 1970.
- [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 F. Moller and G. Birtwistle, editors, *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.