

Alternation Elimination by Complementation^{*}

(Extended Abstract)^{**}

Christian Dax and Felix Klaedtke

ETH Zurich, Computer Science Department, Switzerland

Abstract. In this paper, we revisit constructions from the literature that translate alternating automata into language-equivalent nondeterministic automata. Such constructions are of practical interest in finite-state model checking, since formulas of widely used linear-time temporal logics with future and past operators can directly be translated into alternating automata. We present a construction scheme that can be instantiated for different automata classes to translate alternating automata into language-equivalent nondeterministic automata. The scheme emphasizes the core ingredient of previously proposed alternation-elimination constructions, namely, a reduction to the problem of complementing nondeterministic automata. Furthermore, we clarify and improve previously proposed constructions for different classes of alternating automata by recasting them as instances of our construction scheme. Finally, we present new complementation constructions for 2-way nondeterministic automata from which we then obtain novel alternation-elimination constructions.

1 Introduction

Alternating automata are a powerful tool in finite-state model checking. Here, they serve as a glue between declarative specification languages like LTL [26] and PSL [1] and simple graph-like structures such as nondeterministic Büchi automata, which are well suited for algorithmic treatment, see e.g., [31]. By establishing translations from alternating automata to nondeterministic Büchi automata, one reduces the model checking problem for finite-state systems to a reachability problem on simple graph-like structures, see e.g., [13]. Similarly, such translations can be used to solve the satisfiability problem for declarative specification languages like LTL and PSL.

Translations of declarative specification languages into alternating automata are usually rather direct and easy to establish due to the rich combinatorial structure of alternating automata. Translating an alternating automaton into a nondeterministic Büchi automaton is a purely combinatorial problem. Hence, using alternating automata as an intermediate step is a mathematically elegant way to formalize such translations and to establish their correctness. Another more practical advantage of such translations is that several automata-based

^{*} Supported by the Swiss National Science Foundation (SNF).

^{**} Additional proof details are given in the appendix.

techniques are applicable to optimize the outcome of such translations, e.g., simulation-based reduction techniques [9, 10].

Different classes of alternating automata are used for these kinds of translations depending on the expressive power of the specification language. For instance, for LTL, a restricted class of alternating automata suffices, namely the so-called very-weak alternating Büchi automata [21, 27]. These restrictions have been exploited to obtain efficient translators from LTL to nondeterministic Büchi automata, see [11]. For more expressive languages like the linear-time μ -calculus μ LTL [2, 28], one uses alternating parity automata, and for fragments of the standardized property specification language PSL [1], one uses alternating Büchi automata [5]. If the temporal specification language has future and past operators, one uses 2-way alternating automata instead of 1-way alternating automata, see, e.g., [12, 15, 30]. Due to the immediate practical relevance in finite-state model checking, different constructions have been developed and implemented for translating a given alternating automaton into a language-equivalent nondeterministic automaton like the ones mentioned above.

In this paper, we present a general construction scheme for translating alternating automata into language-equivalent nondeterministic automata. In a nutshell, the general construction scheme shows that the problem of translating an alternating automaton into a language-equivalent nondeterministic automaton reduces to the problem of complementing a nondeterministic automaton. We also show that the nondeterministic automaton that needs to be complemented inherits structural and semantic properties of the given alternating automaton. We exploit these inherited properties to optimize the complementation constructions for special classes of alternating automata.

Furthermore, we instantiate the construction scheme to different classes of alternating automata. Some of the constructions that we obtain share similar technical details with previously proposed constructions as, e.g., the ones described in [11, 17, 22]. Some of them even produce the same nondeterministic Büchi automata modulo minor technical details. However, recasting these known constructions in such a way that they become instances of the construction scheme increases their accessibility. In particular, correctness proofs become modular and less involved. Another benefit of utilizing the construction scheme is that differences and similarities between the translations for the different classes of alternating automata become apparent.

We also present novel alternation-elimination constructions. These constructions are instances of our construction scheme and utilize a new complementation construction for so-called loop-free 2-way nondeterministic co-Büchi automata. In particular, we obtain an alternation-elimination construction that translates a loop-free 2-way alternating Büchi automaton with n states into a language-equivalent nondeterministic Büchi automaton with at most $O(2^{4n})$ states. This construction has potential applications for translating formulas from fragments of PSL extended with temporal past operators into nondeterministic Büchi automata. To the best of our knowledge, the best known construction for this class

of alternating automata results in nondeterministic Büchi automata of size at most $2^{O(n^2)}$ [15].

Overall, we see our contributions as twofold. On the one hand, the presented general construction scheme extracts and uniformly identifies essential ingredients for translating various classes of alternating automata into language-equivalent nondeterministic ones. Previously proposed alternation-elimination constructions for several classes of alternating automata, e.g. [11,12,15,25,28,30] are based on similar ingredients. On the other hand, we clarify and improve existing alternation-elimination constructions for different classes of alternating automata, and we provide novel ones.

We proceed as follows. In Section 2, we give background on alternating automata. In Section 3, we give the general construction scheme. In Section 4, we present instances of that construction scheme for different classes of alternating automata and revisit previously proposed alternation-elimination constructions. Finally, in Section 5, we draw conclusions.

2 Background

We assume that the reader is familiar with automata theory. In this section, we recall the relevant background in this area and fix the notation used throughout this paper.

Given an alphabet Σ , Σ^* is the set of finite words over Σ and Σ^ω is the set of infinite words over Σ . Let w be a word over Σ . We denote its length by $|w|$. Note that $|w| = \infty$ if $w \in \Sigma^\omega$. For $i < |w|$, w_i denotes the i th letter of w , and we write w^i for the word $w_0w_1 \dots w_{i-1}$, where $i \in \mathbb{N} \cup \{\infty\}$ with $i \leq |w|$. The word $u \in \Sigma^* \cup \Sigma^\omega$ is a *prefix* of w if $w^i = u$, for some $i \in \mathbb{N} \cup \{\infty\}$ with $i \leq |w|$.

A (Σ -labeled) *tree* is a function $t : T \rightarrow \Sigma$, where $T \subseteq \mathbb{N}^*$ satisfies the following conditions: (i) T is prefix-closed (i.e., if $w \in T$ and u is a prefix of w then $u \in T$) and (ii) if $xi \in T$ and $i > 0$ then $x(i-1) \in T$. The elements in T are called the *nodes* of t and the empty word ε is called the *root* of t . A node $xi \in T$ with $i \in \mathbb{N}$ is called a *child* of the node $x \in T$. An (infinite) *path* in t is a word $\pi \in \mathbb{N}^\omega$ such that $u \in T$, for every prefix u of π . We write $t(\pi)$ for the word $t(\pi^0)t(\pi^1) \dots \in \Sigma^\omega$.

For a set P of propositions, $\mathcal{B}^+(P)$ is the set of *positive Boolean formulas* over P , i.e., the formulas built from the propositions in P , and the connectives \wedge and \vee . Given $M \subseteq P$ and $\beta \in \mathcal{B}^+(P)$, we write $M \models \beta$ if the assignment that assigns true to the propositions in M and assigns false to the propositions in $P \setminus M$ satisfies β . Moreover, we write $M \models \beta$ if M is a *minimal model* of β , i.e., $M \models \beta$ and there is no $p \in M$ such that $M \setminus \{p\} \models \beta$.

In the following, we define 2-way alternating automata, which scan input words letter by letter with their read-only head. The meaning of “2-way” and “alternating” is best illustrated by the example transition $\delta(p, a) = (q, -1) \vee ((r, 0) \wedge (s, 1))$ of such an automaton, where p, q, r, s are states, a is a letter of the input alphabet, and δ is the transition function. The second coordinate of the tuples $(q, -1), (r, 0), (s, 1)$ specify in which direction the read-only head moves:

−1 for left, 0 for not moving, and 1 for right. The transition above can be read as follows. When reading the letter a in state p , the automaton has two choices: (i) It goes to state q and moves the read-only head to the left. In this case, the automaton proceeds scanning the input word from the updated state and position. (ii) Alternatively, it can go to state r and to state s , where the read-only head is duplicated: the first copy proceeds scanning the input word from the state r , where the position of the read-only head is not altered; the second copy proceeds scanning the input word from the state s , where the read-only head is moved to the right. Note that the choices (i) and (ii) are given by the minimal models of the example transition $\delta(p, a)$, which is a positive Boolean formula with propositions that are pairs of states and movements of the read-only head.

Let $\mathbb{D} := \{-1, 0, 1\}$ be the set of directions in which the read-only head can move. Formally, a *2-way alternating automaton* \mathcal{A} is a tuple $(Q, \Sigma, \delta, q_{\mathbb{I}}, \mathcal{F})$, where Q is a finite set of states, Σ is a finite nonempty alphabet, $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(Q \times \mathbb{D})$ is the transition function, $q_{\mathbb{I}} \in Q$ is the initial state, and $\mathcal{F} \subseteq Q^\omega$ is the acceptance condition. The *size* $|\mathcal{A}|$ of the automaton \mathcal{A} is $|Q|$.

A *configuration* of \mathcal{A} is a pair $(q, i) \in Q \times \mathbb{N}$. Intuitively, q is the current state and i is the position of the read-only head in the input word. A *run* of \mathcal{A} on the word $w \in \Sigma^\omega$ is a tree $r : T \rightarrow Q \times \mathbb{N}$ such that $r(\varepsilon) = (q_{\mathbb{I}}, 0)$ and for each node $x \in T$ with $r(x) = (q, j)$, we have that

$$\{(q', j' - j) \in Q \times \mathbb{Z} \mid r(y) = (q', j'), \text{ where } y \text{ is a child of } x \text{ in } r\} \models \delta(q, w_j).$$

Observe that we require here that the set of labels of the children is a minimal model of the positive Boolean formula $\delta(q, w_j)$. Intuitively, the minimality requirement prevents the automaton from doing unnecessary work in an accepting run. We need this minimality requirement in Section 3.3. A path π in r is *accepting* if $q_0 q_1 \cdots \in \mathcal{F}$, where $r(\pi) = (q_0, i_0)(q_1, i_1) \cdots \in (Q \times \mathbb{N})^\omega$. The run r is *accepting* if every path in r is accepting. The *language* of \mathcal{A} is the set $L(\mathcal{A}) := \{w \in \Sigma^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$.

In the following, we introduce restricted classes of 2-way alternating automata. Let $\mathcal{A} = (Q, \Sigma, \delta, q_{\mathbb{I}}, \mathcal{F})$ be a 2-way alternating automaton.

Note that we do not have any restriction on the acceptance condition \mathcal{F} ; it can be any subset of Q^ω . However, since this is often too general, one usually considers automata where the acceptance conditions are specified in a certain finite way—the *type* of an acceptance condition. Commonly used types of acceptance conditions are listed in Table 1. Here, $\text{inf}(\pi)$ is the set of states that occur infinitely often in $\pi \in Q^\omega$ and the integer k is called the *index* of the automaton. If \mathcal{F} is specified by the type τ , we say that \mathcal{A} is a τ automaton. Moreover, if the type of the acceptance condition is clear from the context, we just give the finite description α instead of \mathcal{F} . For instance, a Büchi automaton is given as a tuple $(S, \Gamma, \eta, s_{\mathbb{I}}, \alpha)$ with $\alpha \subseteq S$.

The automaton \mathcal{A} is *1-way* if $\delta(q, a) \in \mathcal{B}^+(Q \times \{1\})$, for all $q \in Q$ and $a \in \Sigma$. That means, \mathcal{A} can only move the read-only head to the right. If \mathcal{A} is 1-way, we assume that δ is of the form $\delta : Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$.

The automaton \mathcal{A} is *nondeterministic* if δ returns a disjunction of propositions for all inputs; \mathcal{A} is *universal* if δ returns a conjunction of propositions for

type: τ	finite description, acceptance condition: α, \mathcal{F}
Büchi co-Büchi	$\alpha = F \subseteq Q$ $\mathcal{F} := \{\pi \in Q^\omega \mid \text{inf}(\pi) \cap F \neq \emptyset\}$ $\mathcal{F} := \{\pi \in Q^\omega \mid \text{inf}(\pi) \cap F = \emptyset\}$
parity co-parity	$\alpha = \{F_1, \dots, F_{2k}\} \subseteq 2^Q$, where $F_1 \subseteq F_2 \subseteq \dots \subseteq F_{2k}$ $\mathcal{F} := \{\pi \in Q^\omega \mid \min\{i \mid F_i \cap \text{inf}(\pi) \neq \emptyset\} \text{ is even}\}$ $\mathcal{F} := \{\pi \in Q^\omega \mid \min\{i \mid F_i \cap \text{inf}(\pi) \neq \emptyset\} \text{ is odd}\}$
Rabin Streett	$\alpha = \{(B_1, C_1), \dots, (B_k, C_k)\} \subseteq 2^Q \times 2^Q$ $\mathcal{F} := \bigcup_i \{\pi \in Q^\omega \mid \text{inf}(\pi) \cap B_i \neq \emptyset \text{ and } \text{inf}(\pi) \cap C_i = \emptyset\}$ $\mathcal{F} := \bigcap_i \{\pi \in Q^\omega \mid \text{inf}(\pi) \cap B_i = \emptyset \text{ or } \text{inf}(\pi) \cap C_i \neq \emptyset\}$
Muller	$\alpha = \{M_1, \dots, M_k\} \subseteq 2^Q$ $\mathcal{F} := \bigcup_i \{\pi \in Q^\omega \mid \text{inf}(\pi) = M_i\}$

Table 1. Types of acceptance conditions.

all inputs; \mathcal{A} is *deterministic* if it is nondeterministic and universal. For nondeterministic and deterministic automata, we use standard notation. For instance, if \mathcal{A} is nondeterministic, we view δ as a function of the form $\delta : Q \rightarrow 2^{Q \times \mathbb{D}}$. That means, a clause is written as a set. Note that a run $r : T \rightarrow Q \times \mathbb{N}$ of a nondeterministic automaton \mathcal{A} on $w \in \Sigma^\omega$ consists of the single path $\pi = 0^\omega$. To increase readability, we call $r(\pi) \in (Q \times \mathbb{N})^\omega$ also a run of \mathcal{A} on w . Moreover, for $R \subseteq Q$ and $a \in \Sigma$, we abbreviate $\bigcup_{q \in R} \delta(q, a)$ by $\delta(R, a)$.

3 Alternation-Elimination Scheme

In this section, we present a general construction scheme for translating alternating automata into language-equivalent nondeterministic automata. The construction scheme is general in the sense that it can be instantiated for different classes of alternating automata. We provide such instances in Section 4. Before presenting the construction scheme in Section 3.2, we need some preparatory work, which we present in Section 3.1.

3.1 Memoryless Runs as Words

Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$ be a 2-way alternating automaton and let $r : T \rightarrow Q \times \mathbb{N}$ be a run of \mathcal{A} on the word $w \in \Sigma^\omega$. The run r is *memoryless*¹ if all equally labeled nodes $x, y \in T$ have isomorphic subtrees, i.e., if $r(x) = r(y)$ then for all $z \in \mathbb{N}^*$, $xz \in T \Leftrightarrow yz \in T$ and whenever $xz \in T$ then $r(xz) = r(yz)$. We define

$$M(\mathcal{A}) := \{w \in \Sigma^\omega \mid \text{there is an accepting memoryless run on } w\}.$$

¹ The choice of the term “memoryless” becomes clear when viewing a run of an alternating automaton as a representation of a strategy of the first player in a two-person infinite game [24]. A memoryless run encodes a memoryless strategy (also known as a positional strategy) of the first player, i.e., a strategy that does not take the history of a play into account.

Note that $M(\mathcal{A}) \subseteq L(\mathcal{A})$; however, the converse does not hold in general. The languages are equal when \mathcal{A} is an alternating Büchi, co-Büchi, or parity automaton [8, 17], or an alternating Rabin automaton [14]. For alternating Streett and Muller automata, the languages can be different. However, such automata can be translated to language-equivalent alternating parity automata, see [20].

Observe that in a memoryless run $r : T \rightarrow Q \times \mathbb{N}$ of \mathcal{A} on a word $w \in \Sigma^\omega$, we can merge nodes with isomorphic subtrees without losing any information. We obtain an infinite directed graph, which can be represented as an infinite word of functions $f \in (Q \rightarrow 2^{Q \times \mathbb{D}})^\omega$, where $f_j(q)$ returns the labels of the children of a node $x \in T$ with label (q, j) . Note that $f_j(q)$ is well-defined, since $x \in T$ and $y \in T$ have isomorphic subtrees whenever $r(x) = r(y)$.

Definition 1. *The induced tree² $t : T \rightarrow Q \times \mathbb{N}$ of the word $f \in (Q \rightarrow 2^{Q \times \mathbb{D}})^\omega$ is inductively defined:*

- (i) *we have that $\varepsilon \in T$ and $t(\varepsilon) := (q_1, 0)$, and*
- (ii) *for each $x \in T$ with $t(x) = (p, j)$ and $f_j(p) = \{(q_0, d_0), \dots, (q_k, d_k)\}$, we have that $x_0, \dots, x_k \in T$ and $t(x_i) := (q_i, j + d_i)$, for each $i \in \mathbb{N}$ with $i \leq k$.*

The word $f \in (Q \rightarrow 2^{Q \times \mathbb{D}})^\omega$ is a run-word of \mathcal{A} on $w \in \Sigma^\omega$ if the induced tree t is a run of \mathcal{A} on w . Moreover, f is accepting if t is accepting. Finally, we define $L'(\mathcal{A}) := \{w \in \Sigma^\omega \mid \text{there is an accepting run-word of } \mathcal{A} \text{ on } w\}$.

The following lemma states that automata that accept by run-words are as expressive as automata that accept by memoryless runs.

Lemma 2. *For every 2-way alternating automaton \mathcal{A} , $M(\mathcal{A}) = L'(\mathcal{A})$.*

3.2 Reduction to Complementation

For the following, fix a 2-way alternating automaton $\mathcal{A} = (Q, \Sigma, \delta, q_1, \mathcal{F})$. Moreover, we abbreviate the function space $Q \rightarrow 2^{Q \times \mathbb{D}}$ by Γ . Without loss of generality, we assume that \mathcal{A} has a rejecting sink state $s \in Q$, i.e., for every $a \in \Sigma$, $\delta(s, a) = (s, 1)$ and for every $u \in Q^*$, $us^\omega \notin \mathcal{F}$.

From \mathcal{A} we construct the automaton $\mathcal{B} := (Q, \Sigma \times \Gamma, \eta, q_1, Q^\omega \setminus \mathcal{F})$, which is 2-way and nondeterministic. For $q \in Q$ and $(a, g) \in \Sigma \times \Gamma$, we define the transition function as

$$\eta(q, (a, g)) := \begin{cases} g(q) & \text{if } g(q) \equiv \delta(q, a), \\ \{(s, 1)\} & \text{otherwise.} \end{cases}$$

The next lemma is at the core of the results of this paper. It states that the automaton \mathcal{B} rejects exactly those words $(w_0, f_0)(w_1, f_1) \cdots \in (\Sigma \times \Gamma)^\omega$, where f is an accepting run-word of \mathcal{A} on w .

² The tree t is actually not uniquely determined, since we do not uniquely order the children of a node in t . However, the order of the children is irrelevant for the results in this paper, and to simplify matters, we consider two trees as isomorphic if they only differ in the order of their subtrees.

Lemma 3. For all words $w \in \Sigma^\omega$ and $f \in \Gamma^\omega$, it holds that

$$(w_0, f_0)(w_1, f_1) \cdots \in L(\mathcal{B}) \quad \text{iff} \quad \begin{array}{l} \text{(i) } f \text{ is not a run-word of } \mathcal{A} \text{ on } w, \text{ or} \\ \text{(ii) } f \text{ is a rejecting run-word of } \mathcal{A} \text{ on } w. \end{array}$$

The next theorem shows that when $L(\mathcal{A}) = M(\mathcal{A})$, the problem of eliminating the alternation of \mathcal{A} (i.e., to construct a language-equivalent nondeterministic automaton) reduces to the problem of complementing the nondeterministic automaton \mathcal{B} .

Theorem 4. Let \mathcal{C} be a nondeterministic automaton that accepts the complement of $L(\mathcal{B})$ and let \mathcal{D} be the projection of \mathcal{C} on Σ . If $L(\mathcal{A}) = M(\mathcal{A})$ then $L(\mathcal{A}) = L(\mathcal{D})$.

Proof. For a word $w \in \Sigma^\omega$, the following equivalences hold:

$$\begin{array}{l} w \in L(\mathcal{A}) \\ \stackrel{\text{Lemma 2}}{\Leftrightarrow} w \in L'(\mathcal{A}) \\ \stackrel{\text{Lemma 3}}{\Leftrightarrow} (w_0, f_0)(w_1, f_1) \cdots \notin L(\mathcal{B}), \text{ for some } f \in \Gamma^\omega \\ \Leftrightarrow (w_0, f_0)(w_1, f_1) \cdots \in L(\mathcal{C}), \text{ for some } f \in \Gamma^\omega \\ \Leftrightarrow w \in L(\mathcal{D}). \end{array} \quad \square$$

3.3 On Weak and Loop-Free Automata

In the following, we show that the nondeterministic automaton \mathcal{B} inherits properties from the alternating automaton \mathcal{A} . We exploit these properties in Section 4.

Weak Automata. The notion of weakness for automata was introduced in [23]. It led to new insights (e.g., [6, 16, 17, 23]). Moreover, many operations on weak automata are often simpler and more efficient to implement than their counterparts for non-weak automata, see e.g., [11, 17].

The following definition of weakness for an arbitrary acceptance condition \mathcal{F} generalizes the standard definition of weakness for the Büchi acceptance condition. Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$ be a 2-way alternating automaton. A set of states $S \subseteq Q$ is *accepting* if $\text{inf}(r(\pi)) \subseteq S$ implies $r(\pi) \in \mathcal{F}$, for each run r and each path π in r . S is *rejecting* if $\text{inf}(r(\pi)) \subseteq S$ implies $r(\pi) \notin \mathcal{F}$, for each run r and each path π in r . The automaton \mathcal{A} is (inherently) *weak*, if there is a partition on Q into the sets Q_1, \dots, Q_n such that (i) each Q_i is either accepting or rejecting, and (ii) there is a partial order \preceq on the Q_i s such that for every $p \in Q_i, q \in Q_j, a \in \Sigma$, and $d \in \mathbb{D}$: if (q, d) occurs in $\delta(p, a)$ then $Q_j \preceq Q_i$. The automaton \mathcal{A} is *very-weak* (also known as 1-weak or linear), if each Q_i is a singleton. The intuition of weakness is that each path of any run of a weak automaton that gets trapped in one of the Q_i s is accepting iff Q_i is accepting.

The following lemma shows that in our alternation-elimination scheme, the weakness of an alternating automaton \mathcal{A} transfers to the nondeterministic automaton \mathcal{B} , which needs to be complemented (see Theorem 4).

Lemma 5. Let \mathcal{A} be a 2-way alternating automaton and let \mathcal{B} be the 2-way nondeterministic automaton as defined in Section 3.2. If \mathcal{A} is weak then \mathcal{B} is weak, and if \mathcal{A} is very-weak then \mathcal{B} is very-weak.

		VABA	ABA	APA	ARA
1-way	size	$O(n2^n)$	$O(2^{2n})$	$2^{O(nk \log n)}$	$2^{O(nk \log nk)}$
	compl.	by Corollary 11	by [4]	by [18]	by [18]
2-way	size	$O(n2^{3n})$	$2^{O(n^2)}$	$2^{O((nk)^2)}$	
	compl.	by Theorem 10	by [28]	by [28]	
2-way + loop-free	size	$O(n2^{2n})$	$O(2^{4n})$		
	compl.	by Corollary 9	by Theorem 8		

Table 2. Sizes of 1-way NBAs obtained by instances of the construction scheme.

Loop-Free Automata. For a 2-way alternating automaton $\mathcal{A} = (Q, \Sigma, \delta, q_1, \mathcal{F})$, we define the set $\Pi(\mathcal{A})$ as the set of words $(q_0, j_0)(q_1, j_1) \cdots \in (Q \times \mathbb{N})^\omega$ such that $(q_0, j_0) = (q_1, 0)$ and for all $i \in \mathbb{N}$, there is some $a \in \Sigma$ and a minimal model $M \subseteq Q$ of $\delta(q_i, a)$ with $(q_{i+1}, j_{i+1} - j_i) \in M$. The automaton \mathcal{A} is *loop-free* if for all words $\pi \in \Pi(\mathcal{A})$, there are no integers $i, j \in \mathbb{N}$ with $i \neq j$ such that $\pi_i = \pi_j$. Recall that π_i and π_j are configurations, which consist of the current state and the position of the read-only head. So, \mathcal{A} does not loop on a branch in a partial run when scanning an input word.

As in the case of weak automata, the nondeterministic automaton \mathcal{B} inherits the loop freeness of the alternating automaton \mathcal{A} in the construction scheme.

Lemma 6. *Let \mathcal{A} be a 2-way alternating automaton and let \mathcal{B} be the 2-way nondeterministic automaton as defined in Section 3.2. If \mathcal{A} is loop-free then \mathcal{B} is loop-free.*

4 Instances of the Alternation-Elimination Scheme

In this section, we give instances of the construction scheme presented in Section 3. For brevity, we use the following acronyms: ABA for *1-way alternating Büchi automaton*, NBA for *1-way nondeterministic Büchi automaton*, and DBA for *1-way deterministic Büchi automaton*. We prepend the symbols 2, W, and V to the acronyms to denote 2-way, weak, and very-weak automata, respectively. Analogously, we use acronyms for co-Büchi, parity, Rabin, and Streett automata. For instance, co-2WNBA abbreviates *2-way weak nondeterministic co-Büchi automaton*.

Table 2 summarizes some instances of our construction scheme for obtaining language-equivalent NBAs from alternating automata. The table states the sizes of the resulting NBAs, where n is the size and k is the index of the given alternating automaton. Moreover, for each instance in Table 2, we reference the used complementation construction. We remark that the classes of alternating automata in the columns VABA, ABA, and APA in Table 2 are relevant in finite-state model checking, since system properties that are given as formulas of the widely used temporal logics like LTL, PSL, and μ LTL or fragments thereof can directly be translated into alternating automata that belong to one of these classes of automata.

All the instances in Table 2 follow the same pattern, which is as follows. Let us use the notation from Section 3.2. In particular, \mathcal{A} is the given alternating automaton over the alphabet Σ for which we want to construct a language-equivalent NBA \mathcal{D} .

1. From \mathcal{A} we construct the nondeterministic automaton \mathcal{B} over the extended alphabet $\Sigma \times \Gamma$ with the co-acceptance condition of \mathcal{A} .
2. We complement the nondeterministic automaton \mathcal{B} with the complementation construction that is referenced in Table 2. We obtain an NBA \mathcal{C} over the alphabet $\Sigma \times \Gamma$. Note that in some instances it is necessary to switch to another acceptance condition in order to apply the referenced complementation construction. In these cases, we first transform \mathcal{B} accordingly. Such transformations are given in [20].
3. Finally, we project the extended alphabet $\Sigma \times \Gamma$ of the NBA \mathcal{C} to Σ . This gives us the NBA \mathcal{D} .

For instance, if \mathcal{A} is an ARA, we construct an NSA \mathcal{B} . With the construction from [18], we complement the NSA \mathcal{B} and obtain an NBA \mathcal{C} .

Note that with the construction scheme at hand, the only remaining difficult part is the complementation construction in the second step. In the following Section 4.1, we present novel complementation constructions that are used by some of the instances of the construction scheme from Table 2.

4.1 Novel Complementation Constructions

Complementing Loop-Free co-2NBAs. The following construction can be seen as a combination of Vardi's complementation construction [29] for 2-way nondeterministic automata over finite words and the Miyano-Hayashi construction [17,22] for 1-way alternating Büchi automata. The construction is based on the following characterization of the words that are rejected by a loop-free co-2NBA.

Lemma 7. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_{\mathbb{I}}, F)$ be a loop-free co-2NBA and $w \in \Sigma^\omega$. It holds that $w \notin L(\mathcal{A})$ iff there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ such that*

- (i) $q_{\mathbb{I}} \in R_0$,
- (ii) for all $i \in \mathbb{N}$, $p, q \in Q$, and $d \in \mathbb{D}$, if $p \in R_i$, $(q, d) \in \delta(p, w_i)$, and $i + d \geq 0$ then $q \in R_{i+d}$,
- (iii) $S_0 = R_0 \setminus F$,
- (iv) for all $i \in \mathbb{N}$, $p, q \in Q \setminus F$, and $d \in \{0, 1\}$, if $p \in S_i$ and $(q, d) \in \delta(p, w_i)$ then $q \in S_{i+d}$,
- (v) for all $i \in \mathbb{N}$ and $p, q \in Q \setminus F$, if $p \in S_i$, $(q, -1) \in \delta(p, w_i)$, and $i - 1 \geq 0$ then $q \in S_{i-1}$ or $S_{i-1} = \emptyset$, and
- (vi) there are infinitely many $i \in \mathbb{N}$ such that $S_i = \emptyset$ and $S_{i+1} = R_{i+1} \setminus F$.

Proof (sketch). (\Rightarrow) Assume $w \notin L(\mathcal{A})$, i.e., all runs of \mathcal{A} on w visit a state in F infinitely often. We need the following definitions. A word $(q_0, j_0) \dots (q_n, j_n) \in (Q \times \mathbb{N})^*$ is a *run segment* if $(q_{i+1}, j_{i+1} - j_i) \in \delta(q_i, w_i)$, for all $i < n$. The

run segment is *initial* if $(q_0, j_0) = (q_I, 0)$. For $i \in \mathbb{N}$, we define $R_i := \{q_n \in Q \mid \text{there is an initial run segment } (q_0, j_0) \dots (q_n, j_n) \text{ with } j_n = i\}$. Since $(q_I, 0)$ is an initial run segment, R satisfies (i). To show that (ii) holds, assume $i \in \mathbb{N}$, $p, q \in Q$, and $d \in \mathbb{D}$. If $p \in R_i$, $(q, d) \in \delta(p, w_i)$, and $i + d \geq 0$ then there is an initial run segment $r_0 \dots r_n \in (Q \times \mathbb{N})^*$ such that $r_n = (p, i)$. Hence, $r_0 \dots r_n(q, i+d) \in (Q \times \mathbb{N})^*$ is also an initial run segment and therefore, $q \in R_{i+d}$.

It remains to define $S \in (2^{Q \setminus F})^\omega$ that satisfies (iii)–(vi). In the following, we call a run segment $(q_0, j_0) \dots (q_n, j_n) \in (Q \times \mathbb{N})^*$ *F-avoiding* if $q_i \notin F$, for all $i \leq n$. For defining S inductively, it is convenient to use the auxiliary set $S_{-1} := \emptyset$.

Let $m \in \mathbb{N} \cup \{-1\}$ such that $S_m = \emptyset$. We define $T \in (Q \times \mathbb{N})^\omega$ as the set of *F-avoiding* run segments that start in $R_{m+1} \setminus F$, i.e., $T_i := \{q_k \in Q \mid \text{there is an } F\text{-avoiding run segment } (q_0, j_0) \dots (q_k, j_k) \text{ with } q_0 \in R_{m+1}, j_0 = m+1, \text{ and } j_k = i\}$, for $i \in \mathbb{N}$. We show that there is an integer $n \in \mathbb{N}$ such that $T_n = \emptyset$. Assume that such an integer n does not exist. With König's Lemma it is easy to see that T contains an infinite *F-avoiding* run segment. Thus, there is an accepting infinite run of \mathcal{A} on w . This contradicts the assumption $w \notin L(\mathcal{A})$. We choose $n \in \mathbb{N}$ to be minimal and define $S_{m+1+i} := T_i$, for $i \leq n$.

By construction of S , conditions (iii) and (vi) are satisfied. With a similar argumentation that we used to show (ii), we see that (iv)–(v) hold.

(\Leftarrow) Assume there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ with the conditions (i)–(vi). Let $r := (q_0, j_0)(q_1, j_1) \dots \in (Q \times \mathbb{N})^\omega$ be a run of \mathcal{A} on w . Due to conditions (i) and (ii), we have $q_i \in R_{j_i}$, for each $i \in \mathbb{N}$. We show that r is rejecting.

Suppose that r is accepting. There is a $k \in \mathbb{N}$ such that $q_i \notin F$, for all $i > k$. Due to condition (vi), there is a breakpoint $S_m = \emptyset$ with $m > j_k$ and $S_{m+1} = R_{m+1} \setminus F$. Since r is loop-free, there is an $h > k$ such that $j_h = m+1$. Without loss of generality, we assume that h is maximal. Since r is loop-free and the set Q is finite, such an h exists. We have $j_i > m+1$, for all $i > h$.

Since $q_h \in R_{j_h}$ and $q_h \notin F$, we have $q_h \in S_{j_h}$. Using the conditions (iv) and (v), we obtain by induction that $q_i \in S_{j_i}$, for all $i > h$. Since r is loop-free, there is no $n > m$ such that $S_n = \emptyset$. We obtain a contradiction to condition (vi). \square

The following theorem extends the Miyano-Hayashi construction to 2-way automata. Roughly speaking, the constructed NBA \mathcal{C} guesses a run that satisfies the conditions of Lemma 7, for the given co-2NBA and an input word.

Theorem 8. *For a loop-free co-2NBA \mathcal{B} , there is an NBA \mathcal{C} that accepts the complement of $L(\mathcal{B})$ and has $1 + 2^{4|\mathcal{B}|}$ states.*

Proof (sketch). Let $\mathcal{B} = (Q, \Sigma, \delta, q_I, F)$ be a loop-free co-2NBA. We define the NBA $\mathcal{C} := (P, \Sigma, \eta, p_I, G)$, where $P := (2^Q \times 2^{Q \setminus F} \times 2^Q \times 2^{Q \setminus F}) \cup \{p_I\}$, and $G := 2^Q \times \{\emptyset\} \times 2^Q \times 2^{Q \setminus F}$. The transition function η is defined as follows. For the initial state p_I and $a \in \Sigma$, we have that $\eta(p_I, a) \ni (R_0, S_0, R_1, S_1)$ iff the following conditions hold:

- $q_I \in R_0$,

- for all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$: if $(q, d) \in \delta(p, a)$ then $q \in R_d$,
- $S_0 = R_0 \setminus F$,
- for all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta(p, a)$ then $q \in S_d$.

For the other states in P and $a \in \Sigma$, we have that $\eta((R_{-1}, S_{-1}, R_0, S_0), a) \ni (R_0, S_0, R_1, S_1)$ iff the following conditions hold:

- for all $p \in R_0$, $q \in Q$, and $d \in \mathbb{D}$, if $(q, d) \in \delta(p, a)$ then $q \in R_d$,
- for all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta(p, a)$ then $q \in S_d$,
- for all $p \in S_0$ and $q \notin F$, if $(q, -1) \in \delta(p, a)$ then $q \in S_{-1}$ or $S_{-1} = \emptyset$, and
- if $S_0 = \emptyset$ then $S_1 = R_1 \setminus F$.

Intuitively, for an input word w , the automaton guesses the words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ from Lemma 7. With the first and third component of P , it checks the conditions (i) and (ii). With the second and last component, it checks that (iii)–(v) holds. Finally, the acceptance condition ensures that (vi) is satisfied. It easy to check that \mathcal{C} accepts the complement of $L(\mathcal{B})$. \square

Complementing (Loop-Free) co-2VNBA. If the given automaton is a loop-free co-2VNBA, we can simplify the 2-way breakpoint construction presented in Theorem 8. The simplification is based on the following observation: each run of a very-weak automaton will eventually get trapped in a state with a self-loop. Thus, the conditions (iii)–(vi) from Lemma 7 can be simplified accordingly. The simpler conditions allow us to optimize the complementation construction for loop-free co-2VNBA. Roughly speaking, instead of guessing the word $S \in (2^{Q \setminus F})^\omega$ from Lemma 7 and checking that S fulfills the conditions (iii)–(vi), the constructed automaton only has to check that no run of the loop-free co-2VNBA gets trapped in a state $q \notin F$.

Additionally, for very-weak automata, we can extend the above 2-way breakpoint construction so that it can deal with non-loop-free co-2VNBA. This extension is based on the observation that there are only two types of loops: a very-weak automaton loops if (1) it gets trapped in a state without moving the read-only head or (2) it gets trapped in a state in which it first moves the read-only head to the right and then to the left. Such loops can be locally detected.

Based on these two observations, we obtain from Lemma 7 the following corollary that characterizes the words that are rejected by a given (loop-free) co-2VNBA. We exploit this new characterization in the Theorem 10 below for complementing (loop-free) co-2VNBA.

Corollary 9. *Let $\mathcal{A} = (Q, \Sigma, \delta, q_1, F)$ be a co-2VNBA and $w \in \Sigma^\omega$. It holds that $w \notin L(\mathcal{A})$ iff there is a word $R \in (2^Q)^\omega$ such that*

- (i) $q_1 \in R_0$,
- (ii) for all $i \in \mathbb{N}$, $p, q \in Q$, and $d \in \mathbb{D}$, if $p \in R_i$, $(q, d) \in \delta(p, w_i)$, and $i + d \geq 0$ then $q \in R_{i+d}$,
- (iii) there is no $n \in \mathbb{N}$ such that $q \in R_i \setminus F$ and $(q, 1) \in \delta(q, w_i)$, for all $i \geq n$.
- (iv) there is no $i \in \mathbb{N}$ and $q \in R_i \setminus F$ such that $(q, 0) \in \delta(q, w_i)$, and

- (v) there is no $i \in \mathbb{N}$ and $q \in R_i \setminus F$ such that $(q, 1) \in \delta(q, w_i)$ and $(q, -1) \in \delta(q, w_{i+1})$.

Furthermore, when \mathcal{A} is loop-free only (i)–(iii) must hold.

Theorem 10. *For a co-2VNBA \mathcal{B} , there is an NBA \mathcal{C} that accepts the complement of $L(\mathcal{B})$ and has $O(|\mathcal{B}| \cdot 2^{3|\mathcal{B}|})$ states. If \mathcal{B} is loop-free then we can construct \mathcal{C} with $O(|\mathcal{B}| \cdot 2^{2|\mathcal{B}|})$ states.*

Proof (sketch). Let $\mathcal{B} = (Q, \Sigma, \delta, q_{\mathbb{I}}, F)$ be a co-2VNBA, where we assume that $Q = \{1, \dots, n\}$ and $Q \setminus F = \{1, \dots, m\}$, for $m, n \in \mathbb{N}$. If $m = 0$ then $F = Q$ and hence, $L(\mathcal{B}) = \emptyset$. So, assume $m > 0$. Furthermore, assume $m < n$. Otherwise, we add an additional accepting state to Q . Let $k := m + 1$.

We define the NBA $\mathcal{C} := (P, \Sigma, \eta, p_{\mathbb{I}}, G)$, where $P := (2^Q \times 2^Q \times 2^{Q \setminus F} \times \{1, \dots, k\}) \cup \{p_{\mathbb{I}}\}$ and $G := 2^Q \times 2^Q \times 2^{Q \setminus F} \times \{1\}$. The transition function η is defined as follows. For the initial state $p_{\mathbb{I}}$ and $a \in \Sigma$, we have that $\eta(p_{\mathbb{I}}, a) \ni (R_0, R_1, R'_0, 1)$ iff the following conditions hold:

- $q_{\mathbb{I}} \in R_0$,
- for all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$, if $(q, d) \in \delta(p, a)$ then $q \in R_d$,
- there is no $q \in R_0 \setminus F$ such that $(q, 0) \in \delta(q, a)$, and
- $R'_0 = \{q \in R_0 \mid (q, 1) \in \delta(q, a)\}$.

For the other states in P and $a \in \Sigma$, we have that $\eta((R_{-1}, R_0, R'_{-1}, s), a) \ni (R_0, R_1, R'_0, s')$ iff the following conditions hold:

- for all $p \in R_0$, $q \in Q$, and $d \in \mathbb{D}$, if $(q, d) \in \delta(p, a)$ then $q \in R_d$,
- $s' = s$, if $s \leq m$, $s \in R_0$, and $(s, 1) \in \delta(s, a)$; otherwise, $s' = (s \bmod k) + 1$,
- there is no $q \in R_0 \setminus F$ such that $(q, 0) \in \delta(q, a)$,
- there is no $q \in R'_{-1}$ such that $(q, -1) \in \delta(q, a)$, and
- $R'_0 = \{q \in R_0 \mid (q, 1) \in \delta(q, a)\}$.

It remains to show that \mathcal{C} accepts the complement of $L(\mathcal{B})$. Note that \mathcal{C} locally checks all conditions of Corollary 9 except for (iii). Condition (iii) is satisfied if the run is accepting.

We remark that we need the third component in a state because \mathcal{C} forgets the previously read letter. There is an alternative construction, namely, we construct an automaton with the state space $(2^Q \times 2^Q \times \Sigma \times (Q \setminus F)) \cup \{p_{\mathbb{I}}\}$ that stores the letter in the third component of a state.

When \mathcal{B} is loop-free, the automaton \mathcal{C} does not have to check (iv) and (v). Hence we can drop the third component in \mathcal{C} 's state space. \square

If the given automaton \mathcal{A} is a co-VNBA, we can further simplify the construction. To ensure that a word w is rejected by the co-VNBA \mathcal{A} , one only has to check the first three conditions of Corollary 9, where we can restrict d to 1 instead of $d \in \mathbb{D}$ in condition (ii). We point out that the idea of this construction is implicitly used in the translation [3, 11] of VABAs to NBAs and in the “focus approach” of the satisfiability checking of LTL formulas in [19].

Corollary 11. *For a co-VNBA \mathcal{B} , there is a DBA \mathcal{C} that accepts the complement of $L(\mathcal{B})$ and has $O(|\mathcal{B}| \cdot 2^{|\mathcal{B}|})$ states.*

Proof (sketch). Let $\mathcal{B} = (Q, \Sigma, \delta, q_{\mathbb{I}}, F)$. Assume that $Q = \{1, \dots, n\}$ and $Q \setminus F = \{1, \dots, m\}$, for $m, n \in \mathbb{N}$. If $m = 0$ then $F = Q$ and hence, $L(\mathcal{B}) = \emptyset$. So, assume $m > 0$. Furthermore, assume that $m < n$. Otherwise, we add an additional accepting state to Q . Let $k := m + 1$. We define the DBA $\mathcal{C} := (2^Q \times \{1, \dots, k\}, \Sigma, \eta, (\{q_{\mathbb{I}}\}, 1), 2^Q \times \{1\})$, where

$$\eta((R, s), a) := \begin{cases} (\delta(R, a), s) & \text{if } s \leq m, s \in R, \text{ and } s \in \delta(q, a), \\ (\delta(R, a), (s \bmod k) + 1) & \text{otherwise.} \end{cases}$$

\mathcal{B} accepts a word w iff there is a run that gets trapped in a state $q \notin F$ iff \mathcal{C} detects the existence of such a run with its second component and rejects. \square

4.2 Revisiting Alternation-Elimination Constructions

Let us first review the construction of the nondeterministic automaton \mathcal{B} according to the construction scheme in Section 3.2. Observe that \mathcal{B} possesses the alphabet $\Sigma \times \Gamma$, which is exponential in the size of the given alternating automaton \mathcal{A} . In practice, it will not be feasible to explicitly construct \mathcal{B} . Fortunately, for the instances in Table 2, we can optimize the constructions by merging the steps of constructing \mathcal{B} and complementing \mathcal{B} : we build the transitions of \mathcal{B} only locally and we directly project the extended alphabet $\Sigma \times \Gamma$ to Σ when constructing the complement automaton of \mathcal{B} .

For the remainder of this section, let us revisit previously proposed alternation-elimination constructions. The alternation-elimination constructions in [7, 15, 28, 30] for specific classes of alternating automata have a similar flavor as the instances that we obtain from the construction scheme presented in Section 3. In fact, at the core of all these constructions is the complementation of a nondeterministic automaton \mathcal{B} that processes inputs of the given alternating automaton \mathcal{A} augmented with additional information about the runs of the automaton \mathcal{A} . However, the previously proposed constructions and the corresponding instances from our construction scheme differ in the following technical detail. The constructions in [7, 15, 28, 30] use an additional automaton \mathcal{B}' that checks whether such an augmented input is valid, i.e., in our terminology that the additional information is a run-word. In the worst-case, the size of \mathcal{B}' is exponential in the size of \mathcal{A} . We do not need this additional automaton \mathcal{B}' . Instead, the requirement in our construction scheme that the given alternating automaton \mathcal{A} has a rejecting sink state takes care of invalid inputs. This technical detail leads to slightly better upper bounds on the size of the constructed nondeterministic automata, since we do not need to apply the product construction with the automaton \mathcal{B}' to check whether an input is valid.

Finally, we remark that the alternation-elimination construction by Miyano and Hayashi [22] for ABAs, and the constructions by Gastin and Oddoux [11, 12]

for VABAs and loop-free 2VABAs coincide (modulo some minor technical details) with the corresponding instances that we obtain from the presented construction scheme. Moreover, these three alternation-elimination constructions can be seen as special cases of the alternation-elimination construction for loop-free 2ABAs that we obtain from the construction scheme by using the complementation construction in Theorem 8. We are not aware of any other alternation-elimination construction for this class of automata except the one that also handles non-loop-free ABAs. However, the upper bounds for the construction for 2ABAs is worse than the upper bound that we obtain by this new construction for loop-free 2ABAs (see Table 2).

5 Conclusion

We have presented a general construction scheme for translating alternating automata into language-equivalent nondeterministic automata. Furthermore, we have given instances of this construction scheme for various classes of alternating automata. Some of these instances clarify, simplify, or improve existing ones; some of these instances are novel. Since declarative specification languages for reactive systems like LTL or fragments of PSL can directly be translated into some of the considered classes of alternating automata, the presented constructions are of immediate practical interest in finite-state model checking and satisfiability checking.

We remark that the presented constructions depend on complementation constructions for nondeterministic automata. Improving the latter ones, will immediately result in better alternation-elimination constructions. A comparison of the upper bounds on the sizes of the produced nondeterministic automata suggests that alternation elimination for 2-way alternating automata causes a slightly larger blow-up than for 1-way alternating automata (see Table 2). It remains as future work to close this gap, e.g., by providing worst-case examples that match these upper bounds or by improving the constructions.

Acknowledgments. The authors thank Martin Lange, Nir Piterman, and Moshe Vardi for helpful discussions and comments.

References

1. *IEEE standard for property specification language (PSL)*. IEEE Std 1850TM, Oct. 2005.
2. B. BANIEQBAL AND H. BARRINGER, *Temporal logic with fixed points*, in Proceedings of Temporal Logic in Specification 1987, vol. 398 of Lect. Notes Comput. Sci., Springer, 1989, pp. 62–74.
3. R. BLOEM, A. CIMATTI, I. PILL, AND M. ROVERI, *Symbolic implementation of alternating automata*, Int. J. Found. Comput. Sci., 18 (2007), pp. 727–743.
4. B. BOIGELOT, S. JODOGNE, AND P. WOLPER, *An effective decision procedure for linear arithmetic over the integers and reals*, ACM Trans. Comput. Log., 6 (2005), pp. 614–633.

5. D. BUSTAN, D. FISMAN, AND J. HAVLICEK, *Automata construction for PSL*, tech. rep., Computer Science and Applied Mathematics, The Weizmann Institute of Science, Israel, 2005.
6. E. CHANG, Z. MANNA, AND A. PNUELI, *The safety-progress classification*, in Logic and Algebra of Specifications, vol. 79 of NATO Advanced Science Institutes Series, Springer, 1993, pp. 143–202.
7. C. DAX, M. HOFMANN, AND M. LANGE, *A proof system for the linear time μ -calculus*, in Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), vol. 4437 of Lect. Notes Comput. Sci., Springer, 2006, pp. 273–284.
8. E. EMERSON AND C. JUTLA, *Tree automata, mu-calculus and determinacy (extended abstract)*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 1991, pp. 368–377.
9. K. ETESSAMI, T. WILKE, AND R. SCHULLER, *Fair simulation relations, parity games, and state space reduction for Büchi automata*, SIAM J. Comput., 34 (2005), pp. 1159–1175.
10. C. FRITZ AND T. WILKE, *Simulation relations for alternating Büchi automata*, Theoret. Comput. Sci., 338 (2005), pp. 275–314.
11. P. GASTIN AND D. ODDOUX, *Fast LTL to Büchi automata translation*, in Proceedings of the 13th International Conference on Computer Aided Verification (CAV), vol. 2102 of Lect. Notes Comput. Sci., Springer, 2001, pp. 53–65.
12. ———, *LTL with past and two-way very-weak alternating automata*, in Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS), vol. 2747 of Lect. Notes Comput. Sci., Springer, 2003, pp. 439–448.
13. R. GERTH, D. PELED, M. VARDI, AND P. WOLPER, *Simple on-the-fly automatic verification of linear temporal logic*, in Proceedings of the 15th IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification (PSTV), vol. 38 of IFIP Conference Proceedings, Chapman & Hall, 1996, pp. 3–18.
14. C. JUTLA, *Determinization and memoryless winning strategies*, Inf. Comput., 133 (1997), pp. 117–134.
15. O. KUPFERMAN, N. PITERMAN, AND M. VARDI, *Extended temporal logic revisited*, in Proceedings of the 12th International Conference on Concurrency Theory (CONCUR), vol. 2154 of Lect. Notes Comput. Sci., Springer, 2001, pp. 519–535.
16. O. KUPFERMAN AND M. VARDI, *Weak alternating automata and tree automata emptiness*, in Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC), ACM Press, 1998, pp. 224–233.
17. ———, *Weak alternating automata are not that weak*, ACM Trans. Comput. Log., 2 (2001), pp. 408–429.
18. ———, *Complementation constructions for nondeterministic automata on infinite words*, in Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS), vol. 3440 of Lect. Notes Comput. Sci., Springer, 2005, pp. 206–221.
19. M. LANGE AND C. STIRLING, *Focus games for satisfiability and completeness of temporal logic*, in Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science (LICS), IEEE Computer Society, 2001, pp. 357–365.
20. C. LÖDING, *Methods for the transformation of omega-automata: Complexity and connection to second order logic*, Master’s thesis, University of Kiel, Germany, 1998.
21. C. LÖDING AND W. THOMAS, *Alternating automata and logics over infinite words*, in Proceedings of the 1st IFIP International Conference on Theoretical Computer Science (TCS), vol. 1872 of Lect. Notes Comput. Sci., Springer, 2000, pp. 521–535.

22. S. MIYANO AND T. HAYASHI, *Alternating finite automata on ω -words*, Theoret. Comput. Sci., 32 (1984), pp. 321–330.
23. D. MULLER, A. SAOUDI, AND P. SCHUPP, *Alternating automata, the weak monadic theory of trees and its complexity*, Theoret. Comput. Sci., 97 (1992), pp. 233–244.
24. D. MULLER AND P. SCHUPP, *Alternating automata on infinite trees*, Theoret. Comput. Sci., 54 (1987), pp. 267–276.
25. ———, *Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra*, Theoret. Comput. Sci., 141 (1995), pp. 69–107.
26. A. PNUELI, *The temporal logic of programs*, in Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS), IEEE Computer Society, 1977, pp. 46–57.
27. G. ROHDE, *Alternating automata and the temporal logic of ordinals*, PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1997.
28. M. VARDI, *A temporal fixpoint calculus*, in Proceedings of the 15th Annual ACM Symposium on Principles of Programming Languages (POPL), ACM Press, 1988, pp. 250–259.
29. ———, *A note on the reduction of two-way automata to one-way automata*, Inform. Process. Lett., 30 (1989), pp. 261–264.
30. ———, *Reasoning about the past with two-way automata*, in Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP), vol. 1443 of Lect. Notes Comput. Sci., Springer, 1998, pp. 628–641.
31. ———, *Automata-theoretic model checking revisited*, in Proceedings of the 8th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI), vol. 4349 of Lect. Notes Comput. Sci., Springer, 2007, pp. 137–150.

A Additional Proof Details

A.1 Proof Details of Lemma 2

Proof. (\supseteq) Obvious, since the induced tree of a run-word is a memoryless run.

(\subseteq) Suppose that $\mathcal{A} = (Q, \Sigma, \delta, q_{\mathbf{1}}, \mathcal{F})$ and that $r : T \rightarrow Q \times \mathbb{N}$ is an accepting memoryless run on $w \in \Sigma^\omega$. We must show that there is an accepting run-word $f \in (Q \rightarrow 2^{Q \times \mathbb{D}})^\omega$ on w . For $j \in \mathbb{N}$, we define

$$f_j(p) := \left\{ (q, k - j) \mid \begin{array}{l} \text{there are nodes } x, y \in T \text{ with } r(x) = (p, j), \\ r(y) = (q, k), \text{ and } y \text{ is a child of } x \end{array} \right\}.$$

Let $r' : T' \rightarrow Q \times \mathbb{N}$ be the induced tree of f . We first show that for all $x \in T$ and $x' \in T'$ with $r(x) = r'(x')$, we have that

$$\{r'(y) \mid y \text{ is a child of } x' \text{ in } r'\} = \{r(y) \mid y \text{ is a child of } x \text{ in } r\}. \quad (1)$$

Assume that $r(x) = r'(x') = (p, j)$. The following equivalences hold:

$$\begin{aligned} & \text{there is a child } y \text{ of } x' \text{ in } r' \text{ with } r'(y) = (q, k) \\ \Leftrightarrow & (q, k - j) \in f_j(p) \\ \Leftrightarrow & \text{there is a child } y \text{ of } x \text{ in } r \text{ with } r(y) = (q, k). \end{aligned}$$

Note that latter equivalence holds because r is memoryless.

Next, we show that for each node $x' \in T'$, there is a node $x \in T$ such that $r'(x') = r(x)$. We prove this by induction on the length of the nodes in T' . For $|x'| = 0$, choose x as ε . It holds that $r'(x') = (q_{\mathbf{1}}, 0) = r(\varepsilon)$. For $|x'| > 0$, by induction hypothesis, there is a node $z \in T$ such that $r'(x'_0 \dots x'_{|x'|-2}) = r(z)$. With (1) we conclude that there is a child x of z in r with $r(x) = r'(x')$.

Now, we prove that the induced tree r' of f is a run of \mathcal{A} on w . Note that $r'(\varepsilon) = (q_{\mathbf{1}}, 0)$ by definition. Let $x' \in T'$ be any node in r' with $r'(x') = (p, j)$. We have that $r'(x') = r(x)$, for some node $x \in T$. By (1), we have that $\{r'(y) \mid y \text{ is a child of } x'\} = \{r(y) \mid y \text{ is a child of } x\}$. Since r is a run, we conclude that $\{r'(y) \mid y \text{ is a child of } x'\} \equiv \delta(p, w_j)$.

Finally, we show that r' is accepting. Since all paths in r are accepting, it suffices to show that for each path $\pi' \in \mathbb{N}^\omega$ in r' , there is a path $\pi \in \mathbb{N}^\omega$ in r such that $r'(\pi') = r(\pi)$. Let $\pi' \in \mathbb{N}^\omega$ be any path in r' . We define $\pi \in \mathbb{N}^\omega$ recursively:

- Let π_0 be a child of ε in r such that $r(\pi_0) = r'(\pi'_0)$. Note that the existence of such a node $\pi_0 \in T$ is guaranteed, since $r(\varepsilon) = r'(\varepsilon)$ and (1).
- For $i > 0$, let $\pi^i \pi_i$ be a child of the node $\pi^i \in T$ such that $r(\pi^i \pi_i) = r'(\pi^i \pi'_i)$. As above, the existence of such a node $\pi^i \pi_i$ in r is guaranteed, since $r(\pi^i) = r'(\pi^i)$ by construction and (1). \square

A.2 Proof Details of Lemma 3

Proof. (\Rightarrow) Suppose $r = (q_0, j_0)(q_1, j_1) \cdots \in (Q \times \mathbb{N})^\omega$ is an accepting run of the nondeterministic automaton \mathcal{B} on the word $(w_0, f_0)(w_1, f_1) \dots$. Note that j_i denotes the position of the letter (w_{j_i}, f_{j_i}) that is read by the automaton \mathcal{B} when it is in configuration (q_i, j_i) and goes to configuration (q_{i+1}, j_{i+1}) .

Case 1: Suppose that $f_{j_i}(q_i) \models \delta(q_i, w_{j_i})$, for all $i \in \mathbb{N}$. It suffices to show that there is a path $\pi \in \mathbb{N}^\omega$ in the induced tree $t : T \rightarrow Q \times \mathbb{N}$ of f such that $t(\pi) = r \notin \mathcal{F}$ and t cannot be an accepting run-word. We construct the path $\pi \in \mathbb{N}^\omega$ recursively as follows. To simplify notation, we write $\pi = \pi_1 \pi_2 \dots$.

Note that $t(\varepsilon) = (q_\perp, 0) = (q_0, j_0)$ by definition of t and r . For $i \in \mathbb{N}$, define $\pi_{i+1} \in \mathbb{N}$ such that $\pi^i \pi_{i+1} \in T$ is a child of the node $\pi^i \in T$ and $t(\pi^{i+1}) = (q_{i+1}, j_{i+1})$. We show that such a node π^{i+1} exists. By definition of the node π^i , we have that $t(\pi^i) = (q_i, j_i)$. Therefore, π^i has $|f_{j_i}(q_i)|$ children with the set of labels $\{(q', j_i + d) \mid (q', d) \in f_{j_i}(q_i)\}$. Since $(q_{i+1}, j_{i+1}) \in f_{j_i}(q_i)$, there is a child of π^i that is labeled by $(q_{i+1}, j_i + (j_{i+1} - j_i)) = (q_{i+1}, j_{i+1})$.

Case 2: Suppose that there is an integer $i \in \mathbb{N}$ such that $f_{j_i}(q_i) \not\models \delta(q_i, w_{j_i})$. Let $k \in \mathbb{N}$ be the least number such that $f_{j_k}(q_k) \not\models \delta(q_k, w_{j_k})$. Using the construction from Case 1, we can construct a node $\pi_1 \dots \pi_k \in T$ such that $t(\pi_1 \dots \pi_k) = (q_k, j_k)$. Since the children of $\pi_1 \dots \pi_k$ are labeled by the configurations in $C := \{(q', j_k + d) \mid (q', d) \in f_{j_k}(q_k)\}$, we have that $\{(q', (j_k + d) - j_k) \mid (q', j_k + d) \in C\} = f_{j_k}(q_k)$. Since $f_{j_k}(q_k) \not\models \delta(q_k, w_{j_k})$ by assumption, it follows that t is not a run of \mathcal{A} on w . Hence, f is not a run-word of \mathcal{A} on w .

(\Leftarrow) Case 1: Suppose f is not a run-word of \mathcal{A} on w , i.e., the induced tree $t : T \rightarrow Q \times \mathbb{N}$ is not a run of \mathcal{A} on w . There is a node $x \in T$ with label $t(x) = (q, j)$ such that the set $\{(q', j' - j) \mid t(y) = (q', j'), \text{ where } y \text{ is a child of } x\} \not\models \delta(q, w_j)$. Without loss of generality, we assume that x is chosen so that $k := |x|$ is minimal.

Define $r \in (Q \times \mathbb{N})^k$ by $r_i := t(x^i) = (q_i, j_i)$, for $i < k$. By the minimality of x and the definition of t , we have that $f_{j_i}(q_i) \models \delta(q_i, w_{j_i})$, for each $i < k$. Hence, $r_0 \dots r_{k-1}(s, j_{k-1} + 1)(s, j_{k-1} + 2) \cdots \in (Q \times \mathbb{N})^\omega$ is an accepting run of \mathcal{B} on w .

Case 2: Suppose f is a nonaccepting run-word of \mathcal{A} on w , i.e., the induced tree $t : T \rightarrow Q \times \mathbb{N}$ is a nonaccepting run of \mathcal{A} on w . Let $\pi \in \mathbb{N}^\omega$ be a path in t such that $t(\pi) =: (q_0, j_0)(q_1, j_1) \cdots \in (Q \times \mathbb{N})^\omega$ is nonaccepting, i.e., $t(\pi) \notin \mathcal{F}$. By definition of t we have that $f_{j_i}(q_i) \models \delta(q_i, w_{j_i})$, for each $i \in \mathbb{N}$. Hence, $t(\pi)$ is an accepting run of \mathcal{B} on w . \square

A.3 Proof Details of Lemma 5

Proof. Let Q_1, \dots, Q_n be a partition of \mathcal{A} 's state space such that (i) each Q_i is either accepting or rejecting, and (ii) there is a partial order \preceq on Q_i s such that for every $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, and $d \in \mathbb{D}$: if (q, d) occurs in $\delta(p, a)$ then $Q_j \preceq Q_i$. Without loss of generality, we assume that the rejecting sink state s of \mathcal{A} is in the singleton partition Q_n and $Q_n \preceq Q_i$, for all $i \leq n$. Recall that \mathcal{B} has the same set of states Q as \mathcal{A} . It suffices to show that Q_1, \dots, Q_n is a partition

of \mathcal{B} 's state space such that for every $p \in Q_i$, $q \in Q_j$, $(a, f) \in \Sigma \times (Q \rightarrow 2^{Q \times \mathbb{D}})$, and $d \in \mathbb{D}$, if $(q, d) \in \eta(p, (a, f))$ then $Q_j \preceq Q_i$. We have two cases.

- If $f(p) \not\equiv \delta(p, a)$ then $(q, d) \in \{(s, 1)\}$. Hence, $j = n$ and $Q_n \preceq Q_i$.
- If $f(p) \equiv \delta(p, a)$ then $(q, d) \in f(p)$. We have that (q, d) occurs in $\delta(p, a)$ because $f(p)$ is minimal. We conclude that $Q_j \preceq Q_i$.

Note that the arguments in this poof are also valid if the Q_i s are singletons. \square

A.4 Proof Details of Lemma 6

Proof. Assume that $\mathcal{A} = (Q, \Sigma, \delta, q_1, \mathcal{F})$ and that $s \in Q$ is the rejecting sink state of \mathcal{A} . We prove the lemma by contraposition. Let π be a word in $\Pi(\mathcal{B})$ such that there are integers $i, j \in \mathbb{N}$ with $i \neq j$ such that $\pi_i = \pi_j$. Without loss of generality, we assume that $i < j$.

Case 1: $\pi_i = (s, k)$, for some $k \in \mathbb{N}$. By the definition of \mathcal{B} 's transition function, we conclude that $\pi_j = (s, k + (j - i))$. This contradicts the assumption that $\pi_i = \pi_j$.

Case 2: $\pi_i \neq (s, k)$, for all $k \in \mathbb{N}$. From the definition of \mathcal{B} 's transition function, we conclude that $\pi \in \Pi(\mathcal{A})$. So, \mathcal{A} is not loop-free. \square