

Alternation Elimination for Automata over Nested Words^{*}

Christian Dax and Felix Klaedtke

Computer Science Department, ETH Zurich, Switzerland

Abstract. This paper presents constructions for translating alternating automata into nondeterministic nested-word automata (NWAs). With these alternation-elimination constructions at hand, we straightforwardly obtain translations from various temporal logics over nested words from the literature like CaRet and μ NWTL, and extensions thereof to NWAs, which correct, simplify, improve, and generalize the previously given translations. Our alternation-elimination constructions are instances of an alternation-elimination scheme for automata that operate over the tree unfolding of graphs. We obtain these instances by providing constructions for complementing restricted classes of automata with respect to the graphs given by nested words. The scheme generalizes our alternation-elimination scheme for word automata and the presented complementation constructions generalize existing complementation constructions for word automata.

1 Introduction

The regular nested-word languages [6] (a.k.a. visibly pushdown languages [5]) extend the classical regular languages by adding a hierarchical structure to words. Such hierarchical structures in linear sequences occur often and naturally. For instance, an XML document is a linear sequence of characters, where the opening and closing tags structure the document hierarchically. Another example from system verification are the traces of imperative programs, where the hierarchical structure is given by the calls and returns of subprograms. Many automata-theoretic methods for reasoning about regular languages carry over to regular nested-word languages. Instead of word automata one uses nested-word automata (NWAs) [6] or equivalently visibly pushdown automata [5], a restricted class of pushdown automata, where the input symbols determine when the pushdown automaton can push or pop symbols from its stack. For instance, model checking regular nested-word properties of recursive state machines, which can model control flows of imperative programs [3, 4], and of Boolean programs [7], which are widely used as abstractions in software model checking, can be carried out in an automata-theoretic setting, similar to finite-state model checking [23]. That is, the traces of a recursive state machine or a Boolean program are described by an NWA and the negation of the specification, which is given as a

^{*} This work was partially supported by the Swiss National Science Foundation (SNF).

formula in a temporal logic over nested words like CaRet [4], NWTL [2], and μ NWTL [10], is translated into a language-equivalent NWA. It is then checked whether the intersection of the automata’s languages is empty.

In this paper, we view a nested word as a graph with *linear* and *hierarchical* edges. The nodes of the graph are the positions of the nested word. A linear edge connects two neighboring positions and a hierarchical edge connects every call with its matching return position. We present constructions for translating alternating automata that take as input the graphs of nested words into NWAs. These constructions are of immediate relevance for translating temporal logics over nested words like CaRet, NWTL, and μ NWTL and extensions thereof to language-equivalent NWAs. A temporal-logic formula is first translated into such an alternating automaton and from this alternating automaton one obtains an NWA by applying such an alternation-elimination construction. 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 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.

We obtain the alternation-elimination constructions for automata that describe nested-word languages from a construction scheme, which we previously presented for word automata [11] and which we generalize in this paper to automata that operate over the tree unfolding of graphs. In a nutshell, the construction scheme shows that the problem of translating an alternating automaton into a nondeterministic automaton reduces to the problem of complementing an existential automaton, i.e., an automaton that nondeterministically inspects only a single branch in the tree unfolding of the given input graph. To obtain the instances of the construction scheme for nested words, we also provide complementation constructions for restricted classes of existential automata, namely, automata that operate over graphs that represent nested words.

The main benefit of our approach for translating temporal logics over nested words to NWAs is its simplicity and modularity compared to state-of-the-art approaches. By our scheme, complicated translations are divided into smaller independent parts. Moreover, ingredients of the presented constructions are based on existing well established and thoroughly optimized constructions and techniques for nondeterministic word automata, which we generalize to automata that operate over the tree unfolding of the graphs given by nested words. First, we extend our complementation constructions for classes of nondeterministic two-way co-Büchi word automata [11] to classes of existential co-Büchi automata, where the inputs are the graphs of nested words. Our new constructions take the non-local transitions, which stem from the hierarchical structure of nested words, of an existential automaton into account. Intuitively, in such transitions, the read-only head of the automaton jumps from a call directly to the corresponding return or vice versa. Second, in the presented alternation-elimination constructions for alternating parity automata, where the inputs are the graphs of nested words,

we also use and generalize techniques and constructions from [13, 14, 19, 22] for word automata. Finally, as a by product, we obtain a complementation construction for NWAs along the lines of the construction in [13] for complementing nondeterministic Büchi word automata.

We see our contributions as follows. First, based on a general alternation-elimination scheme for automata that operate over the tree unfolding of graphs and several complementation constructions, we provide alternation-elimination constructions for the class of automata that take the graphs of nested words as input with the Büchi and the parity acceptance conditions. Second, we modularize, simplify, and correct existing translations from temporal logics over nested words to NWAs. Third, with the presented complementation constructions we illustrate that various constructions for automata over words generalize with some modifications to constructions for automata that describe nested-word languages.

We proceed as follows. In Section 2, we recapitulate basic definitions and define alternating automata. In Section 3, we present our general alternation-elimination scheme. In Section 4, we present complementation constructions for restricted classes of existential automata with respect to nested-word languages. Furthermore, we instantiate our scheme with these constructions. Finally, in Section 5, we sketch applications of these instances. In particular, we present our translations of various temporal logics over nested words into language-equivalent NWAs. Additional proof details are given in the appendix.

2 Preliminaries

In this section, we fix the notation and terminology that we use in the remainder of the text.

Propositional Logic We denote the set of *positive Boolean formulas* over the set P of propositions by $\text{Bool}^+(P)$, i.e., $\text{Bool}^+(P)$ consists of the formulas that are inductively built from the Boolean constants tt and ff , the propositions in P , and the connectives \vee and \wedge . For $M \subseteq P$ and $b \in \text{Bool}^+(P)$, we write $M \models b$ iff b evaluates to true when assigning true to the propositions in M and false to the propositions in $P \setminus M$. Moreover, we write $M \models\!\!\equiv b$ if M is a *minimal model* of b , i.e., $M \models b$ and there is no $p \in M$ such that $M \setminus \{p\} \models b$.

Words and Trees We denote the set of finite words over the alphabet Σ by Σ^* , the set of infinite words over Σ by Σ^ω , and the empty word by ε . The length of a word w is written as $|w|$, where $|w| = \omega$ when w is an infinite word. For a word w , w_i denotes the symbol of w at position $i < |w|$. We write $v \preceq w$ if v is a prefix of the word w .

A (Σ -labeled) *tree* is a function $t : T \rightarrow \Sigma$, where $T \subseteq \mathbb{N}^*$ satisfies the conditions: (i) T is prefix-closed (i.e., $v \in T$ and $u \preceq v$ imply $u \in T$) and (ii) if $vi \in T$ and $i > 0$ then $v(i-1) \in T$. The elements in T are called the *nodes* of t and the empty word ε is called the *root* of t . A node $vi \in T$ with $i \in \mathbb{N}$ is called

a *child* of the node $v \in T$. A *branch* in t is a word $\pi \in \mathbb{N}^* \cup \mathbb{N}^\omega$ such that either $\pi \in T$ and π does not have any children, or π is infinite and every finite prefix of π is in T . We write $t(\pi)$ for the word $t(\varepsilon)t(\pi_0)t(\pi_0\pi_1)\dots t(\pi_0\pi_1\dots\pi_{n-1}) \in \Sigma^*$ if π is a finite word of length n and $t(\varepsilon)t(\pi_0)t(\pi_0\pi_1)\dots \in \Sigma^\omega$ if π is infinite.

Alternating Automata In the following, we define alternating automata, where the inputs are graphs. Such an automaton is essentially an alternating tree automaton that operates over the tree unfolding of the given input.¹ We obtain the classical automata models for words and trees when viewing words and trees in a rather straightforward way as graphs of the following form and restricting the inputs to the respective class of graphs.

Let D be a nonempty finite set. We call the elements in D *directions*. A D -*skeleton* is a directed, edge-labeled, and pointed graph $(V, (E_d)_{d \in D}, v_I)$, where V is a set of vertices, the relation $E_d \subseteq V \times V$ describes the edges with label $d \in D$, and $v_I \in V$ is the source. We denote the set of labels of the outgoing edges of the vertex $v \in V$ by $\ell(v)$. For an alphabet Σ and a set \mathcal{S} of D -skeletons, the set of *input graphs* $\Sigma^{\mathcal{S}}$ is the set of pairs (S, λ) with $S \in \mathcal{S}$ and $\lambda : V \rightarrow \Sigma$, where V is the set of vertices of S .

Let \mathcal{S} be a nonempty set of D -skeletons. An *alternating \mathcal{S} -automaton* is a tuple $\mathcal{A} = (Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, A)$, where Q is a finite set of states, Σ is a nonempty finite alphabet, $\delta_{D'} : Q \times \Sigma \rightarrow \text{Bool}^+(Q \times D')$ is the transition function for the directions $D' \subseteq D$, $q_I \in Q$ is the initial state, and $A \subseteq Q^\omega$ is the acceptance condition. The acceptance condition A is usually specified in a certain finite way—the *type* of an acceptance condition. Commonly used types of acceptance conditions are listed in Table 1, where $\text{inf}(\pi)$ denotes the set of states that occur infinitely often in $\pi \in Q^\omega$ and the integer k is the *index* of the automaton. If A is specified by the type τ , we say that \mathcal{A} is an alternating τ \mathcal{S} -automaton. Moreover, if the type of the acceptance condition is clear from the context, we just give the finite description α instead of A . For instance, an alternating Büchi \mathcal{S} -automaton is given as a tuple $(Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, \alpha)$ with $\alpha \subseteq Q$.

Let $\mathcal{A} = (Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, A)$ be an alternating \mathcal{S} -automaton and $G \in \Sigma^{\mathcal{S}}$ with $G = (S, \lambda)$ and $S = (V, (E_d)_{d \in D}, v_I)$. A *run* of \mathcal{A} on G is a tree $r : R \rightarrow V \times Q$ with some $R \subseteq \mathbb{N}^*$ such that $r(\varepsilon) = (v_I, q_I)$ and for each node $x \in R$ with $r(x) = (v, p)$, we have $M \models \delta_{\ell(v)}(p, \lambda(v))$, where

$$M := \{(q, d) \in Q \times D' \mid x \text{ has a child } y \text{ with } r(y) = (v', q) \text{ and } (v, v') \in E_d\}.$$

Roughly speaking, \mathcal{A} starts scanning an input graph from the skeleton's initial vertex, where \mathcal{A} is in its initial state. The label (v, p) of the node x in the run is the current configuration of \mathcal{A} . That is, \mathcal{A} is currently in the state p and the read-only head is at the position v in the input graph. The transition $\delta_{D'}(p, \lambda(v))$

¹ The reasons for having graphs as inputs is that it allows us to establish a broadly applicable alternation-elimination scheme (Section 3). In particular, we can use this automata model with the alternation-elimination scheme for translating temporal logics over nested words into NWA's (Section 5) by viewing nested words as graphs, where we restrict the inputs to that class of graphs (Section 4).

Table 1. Types of acceptance conditions

type	finite description α , acceptance condition A
Büchi	$\alpha = F \subseteq Q$ $A := \{\pi \in Q^\omega \mid \inf(\pi) \cap F \neq \emptyset\}$
co-Büchi	$A := \{\pi \in Q^\omega \mid \inf(\pi) \cap F = \emptyset\}$
parity	$\alpha = \{F_0, \dots, F_{2k-1}\} \subseteq 2^Q$, where $F_0 \subseteq F_1 \subseteq \dots \subseteq F_{2k-1}$ $A := \{\pi \in Q^\omega \mid \min\{i \mid F_i \cap \inf(\pi) \neq \emptyset\} \text{ is even}\}$
co-parity	$A := \{\pi \in Q^\omega \mid \min\{i \mid F_i \cap \inf(\pi) \neq \emptyset\} \text{ is odd}\}$
Rabin	$\alpha = \{(B_1, C_1), \dots, (B_k, C_k)\} \subseteq 2^Q \times 2^Q$ $A := \bigcup_i \{\pi \in Q^\omega \mid \inf(\pi) \cap B_i \neq \emptyset \text{ and } \inf(\pi) \cap C_i = \emptyset\}$
Streett	$A := \bigcap_i \{\pi \in Q^\omega \mid \inf(\pi) \cap B_i = \emptyset \text{ or } \inf(\pi) \cap C_i \neq \emptyset\}$

specifies a constraint that has to be fulfilled by the automaton's successor states, where D' is the set of labels $\ell(v)$ in which the read-only head can move at the current position. An infinite branch π in a run r with $r(\pi) = (v_0, q_0)(v_1, q_1) \dots$ is *accepting* if $q_0 q_1 \dots \in A$. The run r is *accepting* if every infinite branch in r is accepting. The *language* of \mathcal{A} is the set

$$L(\mathcal{A}) := \{G \in \Sigma^{\mathbb{S}} \mid \text{there is an accepting run of } \mathcal{A} \text{ on } G\}.$$

We call an alternating \mathcal{S} -automaton $\mathcal{A} = (Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, A)$ *existential* if $\delta_{D'}$ returns a disjunction for all inputs, for all $D' \subseteq D$. Note that a run r of an existential automaton consists of a single branch π . To increase readability, we call $r(\pi)$ also a run. Existential automata are closely related to nondeterministic automata in the sense that an existential automaton also nondeterministically chooses its successor state in a run with respect to the current configuration and its transition function. However, an existential automaton only inspects a single path of the input graph, since together with the chosen successor state it picks a single direction in which it moves its read-only head.

3 Alternation-Elimination Scheme

In this section, we generalize our alternation-elimination scheme for word automata, which we presented in [11], to automata that operate over graphs.

3.1 Reduction to Complementation

The scheme only applies to automata with an acceptance condition for which so-called memoryless runs are sufficient. Formally, for an alternating \mathcal{S} -automaton \mathcal{A} , we require that $L(\mathcal{A}) = M(\mathcal{A})$, where the set $M(\mathcal{A})$ is defined as follows. A run $r : R \rightarrow V \times Q$ of the alternating \mathcal{S} -automaton $\mathcal{A} = (Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, A)$ on $(S, \lambda) \in \Sigma^{\mathbb{S}}$ with $S = (V, (E_d)_{d \in D}, v_I)$ is *memoryless* if equally labeled nodes

have isomorphic subtrees, i.e., for all $x, y \in R$ and $z \in \mathbb{N}^*$, if $r(x) = r(y)$ then $xz \in R$ iff $yz \in R$ and whenever $xz \in R$ then $r(xz) = r(yz)$. We define

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

Obviously, we have $L(\mathcal{A}) \supseteq M(\mathcal{A})$. For an alternating \mathcal{S} -automata \mathcal{A} with the Büchi, co-Büchi, parity, or Rabin acceptance condition, it is well known that the converse $L(\mathcal{A}) \subseteq M(\mathcal{A})$ also holds. However, if \mathcal{A} is, e.g., an alternating \mathcal{S} -automata with the Streett acceptance condition, then $L(\mathcal{A}) \subseteq M(\mathcal{A})$ does not hold in general.

Since the children of equally labeled nodes in a memoryless run $r : R \rightarrow V \times Q$ are also equally labeled, we can represent a memoryless run by the function $\sigma^r : V \times Q \rightarrow 2^{Q \times D}$, where

$$\sigma^r(v, q) := \{(q', d) \in Q \times D \mid \text{there are nodes } x, y \in R \text{ such that } y \text{ is a child of } x, \\ r(x) = (v, q), r(y) = (v', q'), \text{ and } (v, v') \in E_d\}.$$

By “currying” the function σ^r , we obtain the function $\lambda^r : V \rightarrow \Gamma$, where Γ is the set of functions from Q to $2^{Q \times D}$. We represent the run r as the input graph $G^r := (S, \lambda^r) \in \Gamma^{\mathbb{S}}$. We point out that the graph representation of the run has the same skeleton S as the skeleton of the given input graph G .

We now define an existential \mathcal{S} -automaton \mathcal{R} that scans input graphs in $(\Sigma \times \Gamma)^{\mathbb{S}}$, i.e., input graphs of \mathcal{A} that are annotated with information about the configurations of the runs of \mathcal{A} . \mathcal{R} refutes whenever the annotations correspond to an accepting memoryless run of \mathcal{A} on \mathcal{A} 's input graph. Formally, \mathcal{R} is the existential \mathcal{S} -automaton $(Q, \Sigma \times \Gamma, (\eta_{D'})_{D' \subseteq D}, q_I, Q^\omega \setminus A)$, where its transition function $\eta_{D'} : Q \times (\Sigma \times \Gamma) \rightarrow \text{Bool}^+(Q \times D')$ for $D' \subseteq D$ is defined as

$$\eta_{D'}(q, (a, g)) := \begin{cases} \bigvee_{(p, d) \in g(q)} (p, d) & \text{if } g(q) \models \delta_{D'}(q, a), \\ \text{tt} & \text{otherwise.} \end{cases}$$

Intuitively, \mathcal{R} works as follows. It uses its nondeterminism to inspect a path in the skeleton of the input graph. There are two cases in which \mathcal{R} accepts the given input graph. (1) The annotations on the inspected path do not correspond to a branch in a memoryless run of \mathcal{A} . (2) The annotations yield an infinite sequence of states that is not accepting for \mathcal{A} , i.e., the sequence is not in \mathcal{A} 's acceptance condition A .

The formal statement about \mathcal{R} 's language is given in Lemma 1 below; its proof is given in Appendix A.1. The lemma uses the following notation. Let $G = (S, \lambda)$ be an input graph in $(\Sigma \times \Gamma)^{\mathbb{S}}$. $G_{\uparrow \Sigma}$ denotes the input graph in $\Sigma^{\mathbb{S}}$ by projecting G 's labeling to the first component, i.e., $G_{\uparrow \Sigma} := (S, \lambda_{\uparrow \Sigma})$ with $\lambda_{\uparrow \Sigma}(v) := a$ for $\lambda(v) = (a, g)$. Analogously, $G_{\uparrow \Gamma}$ denotes the input graph in $\Gamma^{\mathbb{S}}$ with the skeleton S and the labeling $\lambda_{\uparrow \Gamma}(v) := g$.

Lemma 1. *For any input graph $G \in (\Sigma \times \Gamma)^{\mathbb{S}}$, it holds*

$$G \notin L(\mathcal{R}) \quad \text{iff} \quad \text{there is an accepting memoryless run } r \text{ of } \mathcal{A} \text{ on } G_{\uparrow \Sigma} \\ \text{such that } G_{\uparrow \Gamma} \text{ and } G^r \text{ are isomorphic.}$$

The following theorem allows us to reduce the problem of constructing for \mathcal{A} a language-equivalent nondeterministic automaton to the problem of complementing \mathcal{R} . Note that from an existential automaton that accepts the complement of \mathcal{R} , we easily obtain a nondeterministic automaton that accepts $L(\mathcal{A})$ by projecting the alphabet $\Sigma \times \Gamma$ to Σ . The benefit of this reduction is that it only requires a complementation construction for existential automata. In Section 4, we give such complementation constructions for specific automata classes.

Theorem 2. *If $L(\mathcal{A}) = M(\mathcal{A})$ then $L(\mathcal{A}) = \{G_{\uparrow\Sigma} \mid G \notin L(\mathcal{R})\}$.*

3.2 Inherited Properties

In the following, we show that the existential \mathcal{S} -automaton \mathcal{R} from Section 3.1 inherits properties from the alternating \mathcal{S} -automaton \mathcal{A} . We exploit these properties in our complementation constructions in Section 4.

Let $\mathcal{A} = (Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, A)$ be an alternating \mathcal{S} -automaton and let $W \subseteq D$. The automaton \mathcal{A} is *W-way* if $\delta_{D'}(q, a) \in \text{Bool}^+(Q \times (D' \cap W))$, for all $D' \subseteq D$, $q \in Q$, and $a \in \Sigma$. Intuitively, \mathcal{A} moves its read-only head only along edges in the input graph that are labeled by directions in W . A weaker condition on the allowed movements of the automaton's read-only head is the following. Intuitively, the automaton \mathcal{A} is eventually *W-way* when it eventually moves its read-only head only along edges that are labeled by directions in W . Formally, this condition is defined as follows. Let $G \in \Sigma^{\mathcal{S}}$ be an input graph with $G = (S, \lambda)$ and $S = (V, (E_d)_{d \in D}, v_I)$. We define $\Pi_G(\mathcal{A})$ as the set of words $(q_0, v_0)(q_1, v_1) \dots \in (Q \times V)^\omega$ with $(q_0, v_0) = (q_I, v_I)$ and for all $i \in \mathbb{N}$, there is some $d \in \ell(v_i)$ and a minimal model M of $\delta_{\ell(v_i)}(q_i, \lambda(v_i))$ such that $(q_{i+1}, d) \in M$ and $(v_i, v_{i+1}) \in E_d$. The automaton \mathcal{A} is *eventually W-way* if for every input graph $G \in \Sigma^{\mathcal{S}}$ and every word $(q_0, v_0)(q_1, v_1) \dots \in \Pi_G(\mathcal{A})$, there is an index $n \in \mathbb{N}$ such that for all $i \geq n$, we have $(v_i, v_{i+1}) \in E_d$, for some $d \in W$.

The following definition of *weak* automata generalizes the standard definition [13, 17], where the automata's acceptance condition is a Büchi acceptance condition. Let \mathcal{A} be the alternating \mathcal{S} -automaton $(Q, \Sigma, (\delta_{D'})_{D' \subseteq D}, q_I, A)$. We call a set of states $S \subseteq Q$ *accepting* if $\inf(r(\pi)) \subseteq S$ implies $r(\pi) \in A$, for each run r and each infinite branch π in r . Analogously, we call S *rejecting* if $\inf(r(\pi)) \subseteq S$ implies $r(\pi) \notin A$, for each run r and each infinite branch π in r . The automaton \mathcal{A} is *weak* if there is a partition Q_1, \dots, Q_n of Q 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 $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D' \subseteq D$, and $d \in D'$, if (q, d) occurs in $\delta_{D'}(p, a)$ then $Q_j \preceq Q_i$. The automaton \mathcal{A} is *very weak* if each Q_i is a singleton. The intuition of weakness is that each infinite branch of a run of a weak automaton that gets trapped in one of the Q_i s is accepting iff Q_i is accepting.

Lemma 3. *Let \mathcal{R} be the existential \mathcal{S} -automaton as defined in Section 3.1 for the \mathcal{S} -automaton \mathcal{A} . Moreover, let $W \subseteq D$. The following properties hold.*

- (i) *If \mathcal{A} is (eventually) W-way then \mathcal{R} is (eventually) W-way.*
- (ii) *If \mathcal{A} is (very) weak then \mathcal{R} is (very) weak.*

4 Instances for Automata over Nested Words

In this section, we present alternation-elimination constructions for several classes of automata that take as input the graphs of nested words. We obtain these constructions from our alternation-elimination scheme by providing complementation constructions for existential automata.

4.1 Automata over Nested Words

Nested words [5, 6] are linear sequences equipped with a hierarchical structure. In this paper, we impose this structure by tagging letters with brackets.² More formally, a *nested word* over Σ is a word over the tagged alphabet $\hat{\Sigma} := \Sigma_{int} \cup \Sigma_{call} \cup \Sigma_{ret}$, where the sets $\Sigma_{int} := \Sigma$, $\Sigma_{call} := \{\langle a \mid a \in \Sigma \rangle\}$, and $\Sigma_{ret} := \{a \mid a \in \Sigma\}$ are pairwise disjoint. A position $i \in \mathbb{N}$ in a nested word $w \in \hat{\Sigma}^\omega$ with $w_i \in \Sigma_{int}$ is an *internal position*. Similarly, if $w_i \in \Sigma_{call}$ then i is a *call position* and if $w_i \in \Sigma_{ret}$ then i is a *return position*. Observe that with the attached brackets \langle and \rangle to the letters in Σ , we implicitly group words into subwords. This grouping can be nested. However, not every bracket at a position in a nested word needs to have a matching bracket. The call and return positions in a nested word without matching brackets are called *pending*.

Intuitively speaking, a *nested-word (Büchi) automaton* [5, 6], NWA for short, \mathcal{N} is a nondeterministic pushdown automaton³ that pushes a stack symbol when reading a letter in Σ_{call} , pops a stack symbol when reading a letter in Σ_{ret} (in case it is not the bottom stack symbol), and does not use its stack when reading a letter in Σ_{int} . The NWA \mathcal{N} accepts a word in $\hat{\Sigma}^\omega$ if there is run on that word that visits infinitely often an accepting state. We denote the set of nested words for which there is an accepting run of \mathcal{N} by $L(\mathcal{N})$.

In the following, we view nested words as input graphs, where the hierarchical structure is made explicit by adding to each position the edges that point to its successor and predecessor positions. Formally, these input graphs with their skeletons are defined as follows. Let D be the set $\{-2, -1, 0, 1, 2\}$ and let \mathfrak{S} be the set of D -skeletons $S = (V, (\curvearrowright_d)_{d \in D}, v_I)$, where $V = \mathbb{N}$, $v_I = 0$, and the edge relations are as follows: \curvearrowright_0 is the identity relation over \mathbb{N} , \curvearrowright_1 is the successor relation over \mathbb{N} , and \curvearrowright_2 is a *matching jump relation*. That is, for all $i, j \in \mathbb{N}$, the relation \curvearrowright_2 satisfies the conditions (1) if $i \curvearrowright_2 j$ then $i < j$, (2) $|\{k \mid i \curvearrowright_2 k\}| \leq 1$ and $|\{k \mid k \curvearrowright_2 j\}| \leq 1$, and (3) if $i \curvearrowright_2 j$ then there are no $i', j' \in \mathbb{N}$ with $i' \curvearrowright_2 j'$ and $i < i' \leq j < j'$. The relations \curvearrowright_{-1} and \curvearrowright_{-2} are the inverses of \curvearrowright_1 and \curvearrowright_2 , respectively. For a nested word $w \in \hat{\Sigma}^\omega$, the input graph G_w makes the matching jump relation, which is implicitly given by w , explicit. That is, the

² In [6], nested words are differently defined by not leaving the hierarchical structure implicit by tagging letters with brackets but by making it explicit with a so-called *matching relation* $\sim \subseteq (\{-\infty\} \cup \mathbb{N}) \times (\mathbb{N} \cup \{+\infty\})$. Both definitions are equivalent in the sense that there is a straightforward bijection between them [6].

³ We point out that the stack in the definition in [6] of nested-word automata is implicit. For a precise definition and more details, see Appendix B.1.

D -skeleton $S = (\mathbb{N}, (\curvearrowright_d)_{d \in D}, 0) \in \mathcal{S}$ and the labeling $\lambda : \mathbb{N} \rightarrow \hat{\Sigma}$ of the input graph G_w fulfill the following conditions: (a) For all $i \in \mathbb{N}$, it holds $\lambda(i) = w_i$. (b) For all $i, j \in \mathbb{N}$, if $i \curvearrowright_2 j$ then $\lambda(i) \in \Sigma_{call}$ and $\lambda(j) \in \Sigma_{ret}$. (c) Pending call and return positions do not cross, i.e., for all $k \in \mathbb{N}$ with $\lambda(k) \in \Sigma_{call}$, if there is no $k' \in \mathbb{N}$ with $k \curvearrowright_2 k'$ then for all $j > k$ with $\lambda(j) \in \Sigma_{ret}$, there is some $i \in \mathbb{N}$ with $i \curvearrowright_2 j$. (d) The pending positions do not cross with the matching jump relation \curvearrowright_2 , i.e., for all $k \in \mathbb{N}$ with $\lambda(k) \in \Sigma_{call} \cup \Sigma_{ret}$, if there is no $k' \in \mathbb{N}$ with $k \curvearrowright_2 k'$ or $k' \curvearrowright_2 k$ then there are no $i, j \in \mathbb{N}$ with $i \curvearrowright_2 j$ and $i < k < j$.

The following theorem shows that alternating automata are expressive enough to describe the class of nested-word languages recognizable by NWAs.

Theorem 4. *For every NWA \mathcal{N} , there is an alternating Büchi \mathcal{S} -automaton \mathcal{A} such that for every nested word $w \in \hat{\Sigma}^\omega$, we have $w \in L(\mathcal{N})$ iff $G_w \in L(\mathcal{A})$. Furthermore, \mathcal{A} is $\{1, 2\}$ -way and has $\mathcal{O}(n^2s)$ states, where n is the number of states of \mathcal{N} and s is the number of \mathcal{N} 's stack symbols.*

This result might be surprising since an NWA processes a nested word sequentially and has a stack to store additional information at call positions, which it can use later at the corresponding matching return positions. An alternating automaton does not have a stack. However, instead each node in the input graph of a nested-word explicitly carries the information whether it is a pending or non-pending position. Moreover, for a non-pending position, the matching return or call position, respectively, is also explicitly given to the alternating automaton.

The reason for the blowup in the alternating automaton's state space is that the alternating automaton splits the computation at each non-pending call position, which must synchronize at the corresponding return position. This synchronization is implemented by guessing and causes a blow-up of the factor $\mathcal{O}(ns)$ in the state space. We omit the details of this transformation construction since it is similar to a construction in [10] for so-called jumping automata, which are very similar to our alternating automata when restricting their inputs to the graph representation of nested words.

4.2 Complementing Existential co-Büchi Automata

In this subsection, we present a complementation construction that translates an eventually $\{1, 2\}$ -way existential co-Büchi \mathcal{S} -automaton \mathcal{A} into an NWA \mathcal{N} with $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \notin L(\mathcal{A})\}$. We also optimize this construction for more restricted automata classes. Recall that we immediately obtain translations of alternating Büchi automata over the graph representation of nested words to NWAs by instantiating our alternating-elimination scheme with these complementation constructions. The complementation constructions utilize the following lemma that characterizes the graph representations of nested words that are not accepted by the eventually $\{1, 2\}$ -way existential co-Büchi \mathcal{S} -automaton \mathcal{A} .

In the following, we abbreviate *existential co-Büchi \mathcal{S} -automaton* by the acronym ECA and assume that $\mathcal{A} = (Q, \hat{\Sigma}, (\delta_{D'})_{D' \subseteq D}, q_I, F)$. Furthermore, for $D' \subseteq D$, $\delta_{D'}^d(P, a)$ denotes the set of states that can be reached from a state

in $P \subseteq Q$ by reading the letter $a \in \hat{\Sigma}$ and following a d -labeled edge, i.e., $\delta_{D'}^d(P, a) := \bigcup_{p \in P} \{q \mid \text{the proposition } (q, d) \text{ occurs in } \delta_{D'}(p, a)\}$.

Lemma 5. *For the eventually $\{1, 2\}$ -way ECA \mathcal{A} and a nested word $w \in \hat{\Sigma}^\omega$, we have $G_w \notin L(\mathcal{A})$ iff there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that fulfill the following conditions, where $(\curvearrowright_d)_{d \in D}$ is the family of edge relations of the D -skeleton of G_w :*

- (1) $q_I \in R_0$.
- (2) For all $i, j \in \mathbb{N}$ and $d \in D$ with $i \curvearrowright_d j$, we have $\delta_{\ell(i)}^d(R_i, w_i) \subseteq R_j$.
- (3) For all $i \in \mathbb{N}$ and $q \in R_i$, we have $\emptyset \not\equiv \delta_{\ell(i)}(q, w_i)$.
- (4) $S_0 = R_0 \setminus F$.
- (5) For all $i, j \in \mathbb{N}$ and $d \in D$ with $d > 0$ and $i \curvearrowright_d j$, we have $\delta_{\ell(i)}^d(S_i, w_i) \setminus F \subseteq S_j$.
- (6) There are infinitely many $n \in \mathbb{N}$ such that $S_n = \emptyset$, $S_{n+1} = R_{n+1} \setminus F$, and for all $i, j \in \mathbb{N}$ with $i \curvearrowright_2 j$ and $i \leq n$, we have $j \leq n$.

The conditions (1) and (2) ensure that the word R contains all the runs $(h_0, q_0)(h_1, q_1) \dots$ of the existential automaton \mathcal{A} on the given input graph, i.e., $q_i \in R_{h_i}$, for all $i \in \mathbb{N}$. The conditions (3) to (6) on the words R and S ensure that all the runs are rejecting. Recall that an input graph is rejected if it is not accepted by a finite run and every infinite run visits a state in F infinitely often. Condition (3) ensures that there is no finite accepting run. All the infinite runs are rejecting if the word R can be split into infinitely nonempty segments such that each run of the existential automaton that starts at the beginning of a segment will visit a state in F before reaching the end of the segment. The conditions (4) to (6) on the word S ensure the existence of such a splitting. In particular, the ns from condition (6) mark the end positions of the segments in the splitting.

We remark that we have given in [11] a similar characterization for word automata. The main differences to nested words are as follows. First, the conditions (2) and (5) additionally take the non-local moves of the existential automaton between matched call and return positions into account. Second, condition (6) additionally requires that a segment in the splitting of the word R must not end between a call and its matching return position. Without this additional requirement there might be runs that pass the end of a segment with a non-local move without visiting a state in F .

We now turn to the construction of the NWA, which generalizes our construction in [11] for complementing the word language of an eventually $\{1\}$ -way ECA, which in turn on is based on the breakpoint construction [16]. The additional constraints for the non-local moves in the conditions (2), (5), and (6) are handled by using the stack of the NWA. In particular, whenever the NWA is at a non-pending call position, it guesses its configuration at the matching return position and pushes it on the stack. When reaching the matching return position, it checks the correctness of the guess by popping an element from the stack. Furthermore, the NWA uses the stack to recognize whether it has processed a matched call while not having reached its matching return position yet

by using a bit that is pushed on the stack at non-pending calls and popped from the stack at their matching returns.

Theorem 6. *For an eventually $\{1, 2\}$ -way ECA \mathcal{A} with n states, there is an NWA \mathcal{N} with $\mathcal{O}(2^{4n})$ states, $\mathcal{O}(2^{4n})$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \notin L(\mathcal{A})\}$.*

In the following, we optimize our complementation construction for restricted classes of eventually $\{1, 2\}$ -way ECAs. When \mathcal{A} is also very weak, we can characterize the graph representations of nested words that are not accepted by \mathcal{A} by similar conditions as those given in Lemma 5. However, the existence of the word S together with the conditions (4) and (5) are not required anymore and condition (6) is replaced by the following condition:

$$\begin{aligned} &\text{There is no } q \in Q \setminus F \text{ and no } h \in \mathbb{N}^\omega \text{ such that } q \in R_{h_0} \text{ and for} \\ &\text{all } j \in \mathbb{N}, \text{ there is a direction } d \in \{1, 2\} \text{ such that } h_j \curvearrowright_d h_{j+1} \text{ and} \quad (6') \\ &q \in \delta_{\ell(h_j)}^d(q, w_{h_j}). \end{aligned}$$

Intuitively, condition (6') requires that no run of the existential automaton gets trapped in a state in $Q \setminus F$.

We exploit this specialized characterization to optimize our complementation construction from Theorem 6. Intuitively, the NWA checks that no run of the existential automaton \mathcal{A} gets trapped in a state in $Q \setminus F$. Again, the construction is similar to a construction in [11] for complementing the word language of very weak, eventually $\{1\}$ -way ECAs. However, a subtle difference is that if a run does not get trapped in a state in $Q \setminus F$ between a non-pending call position and the corresponding return position then we additionally must ensure that the run also does not get trapped along the hierarchical edges that directly connect call positions with their matching return positions.

Theorem 7. *For a very weak, eventually $\{1, 2\}$ -way ECA \mathcal{A} with n states, there is an NWA \mathcal{N} with $\mathcal{O}(2^{2n})$ states, $\mathcal{O}(2^{2n})$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \notin L(\mathcal{A})\}$.*

Finally, we consider the case where \mathcal{A} is $\{1, 2\}$ -way for which we can simplify condition (2), since the automaton moves its read-only head only forward:

$$\text{For all } i, j \in \mathbb{N} \text{ and } d \in D \text{ with } d > 0 \text{ and } i \curvearrowright_d j, \text{ we have} \quad (2') \\ \delta_{\ell(i)}^d(R_i, w_i) \subseteq R_j.$$

We directly obtain the following two theorems as special cases of the Theorems 6 and 7, respectively. In a nutshell, we reduce the state space of the NWA by removing the state components that are used to check the consistency of the transitions that move the read-only head along the backward edges $\{-1, -2\}$.

Theorem 8. *For a $\{1, 2\}$ -way ECA \mathcal{A} with n states, there is an NWA \mathcal{N} with $\mathcal{O}(2^{2n})$ states, $\mathcal{O}(2^{2n})$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \notin L(\mathcal{A})\}$.*

Theorem 9. *For a very weak, $\{1, 2\}$ -way ECA \mathcal{A} with n states, there is an NWA \mathcal{N} with $\mathcal{O}(2^n)$ states, $\mathcal{O}(2^n)$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \notin L(\mathcal{A})\}$.*

4.3 Alternation Elimination for Parity Automata

In this subsection, we present constructions that translate an alternating parity \mathcal{S} -automaton \mathcal{A} , APA for short from now on, into an NWA \mathcal{N} with $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \in L(\mathcal{A})\}$.

Our first alternating-elimination construction assumes that the given APA \mathcal{A} is eventually $\{1, 2\}$ -way. The construction comprises two steps: We first translate \mathcal{A} into an alternating Büchi automaton \mathcal{A}' from which we then obtain in a second construction step the NWA \mathcal{N} . In the second construction step, we use an optimized variant of the alternating-elimination construction based on the complementation construction from Theorem 6 that exploits the fact that the runs of \mathcal{A}' have some special form. We remark that both construction steps use and generalize techniques from [13, 14] for complementing nondeterministic automata over infinite words. Due to space limitations, construction details are given in Appendix B.5.

Theorem 10. *For an eventually $\{1, 2\}$ -way APA \mathcal{A} with index k and n states, there is an NWA \mathcal{N} with $2^{\mathcal{O}(nk \log n)}$ states, $2^{\mathcal{O}(nk \log n)}$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \in L(\mathcal{A})\}$.*

By some additional work, we obtain the more general alternation-elimination construction for APAs, where we do not require that the given APA is eventually $\{1, 2\}$ -way. Recall that by our alternation-elimination scheme, it suffices to give a construction for complementing existential parity automata over nested words. The first ingredient of that complementation construction is a generalization of Shepherdson's translation [19, 21] of 2-way nondeterministic finite word automata to deterministic ones that are 1-way. This generalization is obtained with only minor modifications and translates $\{-2, -1, 0, 1, 2\}$ -way existential automata to existential $\{1, 2\}$ -way automata. The second ingredient is a complementation construction for existential $\{1, 2\}$ -way automata, which we easily obtain from Theorem 10 by dualizing [18] the transition function of the given automaton and its acceptance condition, i.e., we swap the Boolean connectives (\wedge and \vee) and the Boolean constants (**tt** and **ff**) in the automaton's transitions, and we complement its acceptance condition, which can be easily done by incrementing the parities of the states by 1. By instantiating our alternation-elimination scheme with a combination—along the same lines as in [20, 22]—of these two ingredients we obtain the following result.

Corollary 11. *For an APA \mathcal{A} with index k and n states, there is an NWA \mathcal{N} with $2^{\mathcal{O}((nk)^2)}$ states, $2^{\mathcal{O}((nk)^2)}$ stack symbols, and $L(\mathcal{N}) = \{w \in \hat{\Sigma}^\omega \mid G_w \in L(\mathcal{A})\}$.*

5 Applications and Concluding Remarks

A first and immediate application of our alternation-elimination constructions is a construction for complementing NWAs: For a given NWA \mathcal{N} , we first construct by Theorem 4 a $\{1, 2\}$ -way alternating Büchi automaton \mathcal{A} . We complement \mathcal{A} 's language by dualizing \mathcal{A} [18]. Note that the Büchi acceptance condition of \mathcal{A}

becomes a co-Büchi acceptance condition in the dualized automaton. Since a co-Büchi acceptance condition can be written as a parity acceptance condition with index 2, we can apply Theorem 10 to the dualized automaton and obtain an NWA $\tilde{\mathcal{N}}$ with $L(\tilde{\mathcal{N}}) = \hat{\Sigma}^\omega \setminus L(\mathcal{N})$. The resulting NWA $\tilde{\mathcal{N}}$ has $2^{\mathcal{O}(n^2 s \log ns)}$ states and stack symbols, where n is the number of states of \mathcal{N} and s is the number of \mathcal{N} 's stack symbols.

This construction generalizes the complementation construction in [13] from nondeterministic Büchi word automata to NWAs. However, observe that we obtain a worse upper bound, namely, $2^{\mathcal{O}(n^2 s \log ns)}$ instead of $2^{\mathcal{O}(n \log n)}$. One reason is that the construction for NWAs has to take the stack into account. Another reason is that we first translate the NWA \mathcal{A} by Theorem 4 into an alternating automaton that does not have a stack but takes the graph representation of nested words as inputs. This translation causes a blowup of the factor $\mathcal{O}(ns)$ in the automaton's state space. It is also worth pointing out that our complementation construction based on alternating automata does not match the best known upper bound $2^{\mathcal{O}(n^2)}$ for complementing NWAs [6]. This better upper bound is achieved by splitting the complementation construction into two separate constructions, which are later combined by a simple product construction. Only one of these two constructions involves a complementation construction, where only nondeterministic Büchi word automata need to be complemented. It remains open whether our complementation construction based on Theorem 10 can be optimized so that it matches or improves the upper bound $2^{\mathcal{O}(n^2)}$.

Our second and main application area of the presented alternation-elimination constructions is the translation of temporal logics over nested words into NWAs for effectively solving the satisfiability problem and the model-checking problem for recursive state machines and Boolean programs. From the constructions in Section 4, we straightforwardly obtain such translations, which we sketch in the following and which improve, extend, and correct previously presented translations. Overall, our translations together with our results in [11] and [12] demonstrate that complementation constructions for restricted classes of nondeterministic automata are at the core in translating temporal logics into nondeterministic automata; thus they are also at the core in the automata-theoretic approach to model checking and satisfiability checking.

In [10], Bozzelli introduces the temporal logic μ NWTL, which extends the linear-time μ -calculus [8] by next modalities for the calls and returns in nested words. μ NWTL has the same expressive power as NWAs. Our alternation-elimination scheme allow us to modularize and optimize Bozzelli's monolithic translation to NWAs. Similar to Bozzelli, we first translate a μ NWTL formula into an alternating parity automaton (alternating jump automaton in Bozzelli's paper, respectively) with k parities, where k is the alternation depth of the given μ NWTL formula. The size of the automaton is linear in the formula length. We then apply Corollary 11 to obtain an NWA. The size of the resulting NWA is $2^{\mathcal{O}((nk)^2)}$, where n is the size of the alternating parity automaton. For formulas that do not refer to the past or only in a restricted way such that the alternating

parity automaton is eventually $\{1, 2\}$ -way, we can use Theorem 10 to reduce this upper bound to $2^{\mathcal{O}(nk \log n)}$.

In [2, 4], the respective authors introduce the temporal logics CaRet and NWTTL, which extend the classical linear-time temporal logic LTL. The extensions consist of new modalities that take the hierarchical structure of nested words into account. In other words, the new modalities allow one to express properties along the different paths in a nested word. NWTTL subsumes CaRet and is first-order complete. For both these logics, the authors of the respective papers also provide translations into NWAs. Their translations are direct, i.e., they do not use alternating automata as an intermediate step. Although the techniques used in such direct translations are rather standard, they are complex and their correctness proofs are cumbersome. As a matter of fact, the translation in [2] is flawed.⁴

Instead of directly constructing the NWA from a CaRet or an NWTTL formula, we utilize our alternation-elimination scheme. In more detail, we first translate the given formula into an alternating automaton with a Büchi acceptance condition. As for LTL, the translation for CaRet and NWTTL into alternating automata is straightforward and linear in the formula length, since each temporal operators in CaRet and also NWTTL only allows us to specify a property along a single path in the graph representation of nested words. Moreover, the obtained automaton is eventually $\{1, 2\}$ -way and very weak. Then, by instantiating the alternation-elimination scheme with the complementation constructions from Theorem 7, we obtain from such an alternating automaton an NWA.

The benefits of this translation is as follows. Its correctness is easier to establish. The difficult part is the alternation-elimination construction. However, by our scheme its correctness proof boils down of proving the correctness of a complementation construction for *existential* automata. Moreover, we can handle the future-only fragment of CaRet and NWTTL more efficiently by using the specialized instance of our alternation-elimination scheme that we obtain from Theorem 9. Finally, our translation can easily be adapted to extensions of NWTTL and other temporal logics. Similar to LTL and word automata [24], NWTTL and thus also CaRet are strictly less expressive than NWAs.⁵ For LTL, several extensions and variants have been proposed to overcome this limitation. Among them are Wolper’s ETL [24] and the industrial-strength logic PSL [1]. Similar extensions are possible for NWTTL to increase its expressiveness. For instance, we can extend NWTTL with the PSL-specific temporal operators that allow one to use

⁴ A counterexample is given by the NWTTL formula $\diamond^a \text{ff}$, where \diamond^a is the “abstract” version of classical eventually modality \diamond in LTL. The constructed NWA accepts the nested word $((\emptyset \ \emptyset \ \emptyset))^\omega$, which is not a model of the formula $\diamond^a \text{ff}$ since $\diamond^a \text{ff}$ is unsatisfiable. More generally speaking, the constructions in [2] disregards unfoldings of least fixpoint formulas along the jumps from call to return positions. It should be possible to correct their tableaux-based construction by using the technique for ensuring condition (6) of Lemma 5 in our automaton construction from Theorem 6.

⁵ The language of nested words in which every position is internal and the proposition p holds at every even position witnesses that NWTTL cannot express all nested-word regular languages.

(semi-extended) regular expressions. With our alternation-elimination scheme, we obtain a translation to NWA's with only minor modifications. Namely, the translation into alternating automata is standard, see e.g., [9, 12]. Furthermore, since the alternating automata are not necessarily very weak any more, we use the complementation construction from Theorem 6 instead of the more specialized one from Theorem 7 to instantiate the alternation-elimination scheme. However, it is open whether and which PSL-like extensions [15] of NWTL are capable of expressing all NWA-recognizable languages.

References

1. IEEE standard for property specification language (PSL). IEEE Std 1850TM, Oct. 2005.
2. R. Alur, M. Arenas, P. Barceló, K. Etessami, N. Immerman, and L. Libkin. First-order and temporal logics for nested words. *Log. Methods Comput. Sci.*, 4(4), 2008.
3. R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. W. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Trans. Progr. Lang. Syst.*, 27(4):786–818, 2005.
4. R. Alur, K. Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lect. Notes Comput. Sci.*, pages 467–481. Springer, 2004.
5. R. Alur and P. Madhusudan. Visibly pushdown languages. In *ACM Symposium on Theory of Computing (STOC)*, pages 202–211. ACM Press, 2004.
6. R. Alur and P. Madhusudan. Adding nesting structure to words. *J. ACM*, 56(3):1–43, 2009.
7. T. Ball and S. K. Rajamani. Boolean programs: A model and process for software analysis. Technical Report MSR-TR-2000-14, Microsoft Research, 2000.
8. B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Temporal Logic in Specification*, volume 398 of *Lect. Notes Comput. Sci.*, pages 62–74. Springer, 1989.
9. S. Ben-David, R. Bloem, D. Fisman, A. Griesmayer, I. Pill, and S. Ruah. Automata construction algorithms optimized for PSL. Technical report, The Prosyd Project, <http://www.prosyd.org>, 2005.
10. L. Bozzelli. Alternating automata and a temporal fixpoint calculus for visibly pushdown languages. In *International Conference on Concurrency Theory (CONCUR)*, volume 4703 of *Lect. Notes Comput. Sci.*, pages 476–491. Springer, 2007.
11. C. Dax and F. Klaedtke. Alternation elimination by complementation. In *International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, volume 5530 of *Lect. Notes Comput. Sci.*, pages 214–229. Springer, 2008.
12. C. Dax, F. Klaedtke, and M. Lange. On regular temporal logics with past. *Acta Inform.*, 47(4):251–277, 2010.
13. O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Trans. Comput. Log.*, 2(3):408–429, 2001.
14. O. Kupferman and M. Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 3340 of *Lect. Notes Comput. Sci.*, pages 206–221. Springer, 2005.

15. M. Lange. Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In *International Conference on Concurrency Theory (CONCUR)*, volume 4703 of *Lect. Notes Comput. Sci.*, pages 90–104. Springer, 2007.
16. S. Miyano and T. Hayashi. Alternating finite automata on ω -words. *Theoret. Comput. Sci.*, 32(3):321–330, 1984.
17. D. Muller, A. Saoudi, and P. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *Theoret. Comput. Sci.*, 97(2):233–244, 1992.
18. D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoret. Comput. Sci.*, 54(2–3):267–276, 1987.
19. J. C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3(2):198–200, 1959.
20. M. Y. Vardi. A temporal fixpoint calculus. In *ACM Symposium on Principles of Programming Languages (POPL)*, pages 250–259. ACM Press, 1988.
21. M. Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Inform. Process. Lett.*, 30(5):261–264, 1989.
22. M. Y. Vardi. Reasoning about the past with two-way automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of *Lect. Notes Comput. Sci.*, pages 628–641. Springer, 1998.
23. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Symposium on Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society, 1986.
24. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.

A Additional Details for the Alternation-Elimination Scheme

A.1 Proof Details for Lemma 1

In the following, let $G \in (\Sigma \times \Gamma)^{\mathcal{S}}$ be an input graph with $G = (S, \lambda)$, where S is the skeleton $(V, (E_d)_{d \in D}, v_I)$ and the labeling $\lambda : V \rightarrow (\Sigma \times \Gamma)$. Recall that $G_{\uparrow \Sigma} = (S, \lambda_{\uparrow \Sigma})$ and $G_{\uparrow \Gamma} = (S, \lambda_{\uparrow \Gamma})$ denote the input graphs, where the labelings are projected to the alphabets Σ and Γ , respectively.

We first prove that if \mathcal{R} accepts G then \mathcal{A} does not accept $G_{\uparrow \Sigma}$ or $G_{\uparrow \Gamma}$ does not encode a run of \mathcal{A} on $G_{\uparrow \Sigma}$. Suppose r is an accepting run of \mathcal{R} on G . We consider the following two cases.

Case 1: r is infinite, i.e., $r = (v_0, q_0)(v_1, q_1) \dots \in (V \times Q)^\omega$ and $q_0 q_1 \dots \notin A$. Moreover, we have $\lambda_{\uparrow \Gamma}(v_i)(q_i) \equiv \delta_{\ell(v_i)}(q_i, \lambda_{\uparrow \Sigma}(v_i))$, for all $i \in \mathbb{N}$. We have to show that the input graph $G_{\uparrow \Gamma}$ does not encode an accepting run of \mathcal{A} on $G_{\uparrow \Sigma}$. It suffices to show that there is a branch $\pi \in T$ in the induced tree $t : T \rightarrow V \times Q$ of $G_{\uparrow \Gamma}$ such that $t(\pi) = r$. We construct the branch π recursively as follows.

- Define $\pi_0 := \varepsilon$. Then $t(\pi_0) = (v_0, q_0) = (v_I, q_I)$ by definition of t and r .
- For $i > 0$, define $\pi_i \in T$ such that π_i is some child of the node $\pi_{i-1} \in T$ that is labeled by (v_i, q_i) .

We show that the node $\pi_i \in T$ exists. By definition of the node π_{i-1} , we have $t(\pi_{i-1}) = (v_{i-1}, q_{i-1})$. Therefore, π_{i-1} has $|\lambda_{\uparrow \Gamma}(v_{i-1})(q_{i-1})|$ many children labeled by elements from the set

$$\{(v', q') \mid (q', d) \in \lambda_{\uparrow \Gamma}(v_{i-1})(q_{i-1}) \text{ and } (v_{i-1}, v') \in E_d\}.$$

That is, there exists a direction $d \in D$ such that some child is labeled by (v_i, q_i) , according to the definition of the induced tree.

Case 2: Suppose that r is finite, i.e., there is an integer $k \in \mathbb{N}$ such that $\lambda_{\uparrow \Gamma}(v_k)(q_k) \not\equiv \delta_{\ell(v_k)}(q_k, \lambda_{\uparrow \Sigma}(v_k))$. Without loss of generality, we assume that $k \in \mathbb{N}$ is minimal. It suffices to show that the induced tree $t : T \rightarrow V \times Q$ of $G_{\uparrow \Gamma}$ is not a run of \mathcal{A} on $G_{\uparrow \Sigma}$. Using the construction from Case 1, we construct a path $\pi = \pi_0 \dots \pi_k \in T$ such that $t(\pi_k) = (v_k, q_k)$. By definition, the children of π_k in T are labeled by the set

$$\{(v', q') \mid (q', d) \in \lambda_{\uparrow \Gamma}(v_k)(q_k) \text{ and } (v_k, v') \in E_d\}$$

and $\lambda_{\uparrow \Gamma}(v_k)(q_k) \not\equiv \delta_{\ell(v_k)}(q_k, \lambda_{\uparrow \Sigma}(v_k))$, by assumption. It follows that

$$\{(q', d) \mid t(y') = (v', q'), y' \text{ is a child of } v_k, \text{ and } (v_k, v') \in E_d\} \not\equiv \delta(q_k, \lambda_{\uparrow \Gamma}(v_k)).$$

We conclude that t is not a run of \mathcal{A} on $G_{\uparrow \Sigma}$.

Next, we prove by contraposition that if \mathcal{R} rejects G then $G_{\uparrow \Gamma}$ corresponds to an accepting run of \mathcal{A} on $G_{\uparrow \Sigma}$. There are two cases so that $G_{\uparrow \Gamma}$ does not correspond to an accepting run of \mathcal{A} on $G_{\uparrow \Sigma}$.

Case 1: Suppose $G_{\uparrow\Gamma}$ does not correspond to a run of \mathcal{A} on $G_{\uparrow\Sigma}$ at all. That is, the induced tree $t : T \rightarrow Q \times V$ is not a run of \mathcal{A} on $G_{\uparrow\Sigma}$. In particular, there is a node $x \in T$ with label $t(x) = (v, q)$ such that the set

$$\{(q', d) \mid t(y) = (v', q'), y \text{ is a child of } x, \text{ and } (v, v') \in E_d\} \not\equiv \delta_{\ell(v)}(q, \lambda_{\uparrow\Sigma}(v)).$$

Without loss of generality, we assume that x is chosen so that $k := |x|$ is minimal.

We construct an accepting run r of \mathcal{R} on G . Define $r_i := (v_i, q_i) = t(x_0 \dots x_i)$, for $i < k$. By the minimality of $|x|$, we have that r is a run of \mathcal{R} on G . Since r is finite, r is accepting.

Case 2: Suppose $G_{\uparrow\Gamma}$ encodes a run $t : T \rightarrow V \times Q$ of \mathcal{A} on $G_{\uparrow\Gamma}$, but the run is not accepting. We construct an accepting run for \mathcal{R} on G . Let π be an infinite branch in T with $t(\pi) = (v_0, q_0)(v_1, q_1) \dots \in (V \times Q)^\omega$ and $q_0 q_1 \dots \notin A$. Since $t : T \rightarrow V \times Q$ is a run of \mathcal{A} on $G_{\uparrow\Sigma}$, we have $\lambda_{\uparrow\Gamma}(v_i)(q_i) \equiv \delta_{\ell(v_i)}(q_i, \lambda_{\uparrow\Sigma}(v_i))$, for each $i \in \mathbb{N}$. Hence, $t(\pi)$ is an accepting run of \mathcal{R} on G .

A.2 Proof Details of Theorem 2

Let $G \in \Sigma^{\mathfrak{S}}$ be an input graph. By the assumption $L(\mathcal{A}) = M(\mathcal{A})$, we have $G \in L(\mathcal{A})$ iff there is an accepting memoryless run r of \mathcal{A} on G . Let $G^r \in \Gamma^{\mathfrak{S}}$ be the graph representation of the run r . By Lemma 1, the existence of the run r is equivalent to the fact that the combined input graph of G and G^r in $(\Sigma \times \Gamma)^{\mathfrak{S}}$ is not accepted by \mathcal{R} . We conclude $G \in L(\mathcal{A})$ iff \mathcal{R} rejects the combined graph of G and G^r .

A.3 Proof Details for Lemma 3

We first show that \mathcal{R} inherits weakness and very weakness from \mathcal{A} . Let Q_1, \dots, Q_n be a partition of \mathcal{A} 's state space such that (1) each Q_i is either accepting or rejecting, and (2) there is a partial order \preceq on Q_i s such that for every $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D' \subseteq D$, and $d \in D'$, if (q, d) occurs in $\delta_{D'}(p, a)$ then $Q_j \preceq Q_i$. Recall that \mathcal{R} has the same set of states Q as \mathcal{A} . We claim that Q_1, \dots, Q_n is a partition of \mathcal{R} 's state space with the desired property. That is, for every $p \in Q_i$, $q \in Q_j$, $(a, g) \in \Sigma \times \Gamma$, $D' \subseteq D$, and $d \in D'$, if $(q, d) \in \eta_{D'}(p, (a, g))$ then $Q_j \preceq Q_i$. We have two cases.

- If $g(p) \not\equiv \delta_{D'}(p, a)$ then $\eta_{D'}(p, (a, g)) = \text{tt}$ and there is nothing to prove.
- If $g(p) \equiv \delta_{D'}(p, a)$ then $(q, d) \in g(p)$. We have (q, d) occurs in $\delta_{D'}(p, a)$ because $g(p)$ is minimal. We conclude that $Q_j \preceq Q_i$.

Note that the arguments remain valid when the Q_i s are singletons.

Obviously, by definition, if \mathcal{A} is W -way then \mathcal{R} is W -way. We now prove the inheritance of eventually W -wayness. Assume \mathcal{A} is eventually W -way. Consider a word $(q_0, v_0)(q_1, v_1) \dots$ in the set $\Pi_G(\mathcal{R})$, for some input graph $G \in (\Sigma \times \Gamma)^{\mathfrak{S}}$. By the definition of the transition function η , we have for all $i \in \mathbb{N}$, $D' \subseteq D$, and $d \in D'$ with $(v_i, v_{i+1}) \in E_d$ that there is a set M such that $(q_{i+1}, d) \in M$ and $M \equiv \delta_{D'}(q_i, \lambda(v_i))$. It follows that the word $(q_0, v_0)(q_1, v_1) \dots$ is in the set $\Pi_{G_{\uparrow\Sigma}}(\mathcal{A})$. Since every word in $\Pi_{G_{\uparrow\Sigma}}(\mathcal{A})$ eventually changes its head position only along the directions in W , the automaton \mathcal{R} is also eventually W -way.

B Additional Details for the Alternation-Elimination Constructions

Throughout this section, we use the following notation and abbreviations. For an integer $n > 0$, let $[n] := \{0, 1, \dots, n-1\}$ and we write \mathbb{D} for the set $\{-2, -1, 0, 1, 2\}$ of directions in the graph representations of nested words. We denote a nested word in $\hat{\Sigma}^\omega$ by a pair (w, \curvearrowright) , where $w \in \hat{\Sigma}^\omega$ and \curvearrowright is the family of labeled edge relations of the underlying skeleton of the graph representation of the nested word w , that is, $\curvearrowright = (\curvearrowright_d)_{d \in \mathbb{D}}$ with $\curvearrowright_d \subseteq \mathbb{N} \times \mathbb{N}$, for $d \in \mathbb{D}$. Furthermore, we abuse notation and identify nested words (w, \curvearrowright) with their representation as input graphs G_w . Finally, if a proposition $p \in P$ occurs in a minimal model of a positive Boolean formula $b \in \text{Bool}^+(P)$, we write $p \in b$. Analogously, we write $\text{tt} \in b$ if $\emptyset \models b$.

B.1 Nested-Word Automata

We recall the definition of nested-word automata [5, 6]. A *nested-word (Büchi) automaton (NWA)* is a tuple $\mathcal{N} = (Q, \Gamma, \Sigma, \delta, q_I, F)$, where Q is a finite set of states, Γ is a finite set of stack symbols with $\perp \notin \Gamma$, Σ is the input alphabet, δ consists of three nondeterministic transition functions $\delta_{\text{int}} : Q \times \Sigma_{\text{int}} \rightarrow 2^Q$, $\delta_{\text{call}} : Q \times \Sigma_{\text{call}} \rightarrow 2^{Q \times \Gamma}$, and $\delta_{\text{ret}} : Q \times (\Gamma \cup \{\perp\}) \times \Sigma_{\text{ret}} \rightarrow 2^Q$, $q_I \in Q$ is the initial state, and $F \subseteq Q$ is the set of accepting states.

A *run* of \mathcal{N} on a nested word $w_0 w_1 \dots \in \hat{\Sigma}^\omega$ is a word $(q_0, \gamma_0)(q_1, \gamma_1) \dots \in (Q \times \Gamma)^\omega$ with $q_0 = q_I$ and for each $i \in \mathbb{N}$, the following conditions hold:

1. If $w_i \in \Sigma_{\text{int}}$ then $q_{i+1} \in \delta_{\text{int}}(q_i, w_i)$.
2. If $w_i \in \Sigma_{\text{call}}$ then $(q_{i+1}, \gamma_{i+1}) \in \delta_{\text{call}}(q_i, w_i)$.
3. If $w_i \in \Sigma_{\text{ret}}$ and i is pending then $q_{i+1} \in \delta_{\text{ret}}(q_i, \perp, w_i)$.
4. If $w_i \in \Sigma_{\text{ret}}$ and i is non-pending then $q_{i+1} \in \delta_{\text{ret}}(q_i, \gamma_{j+1}, w_i)$, where j is i 's matching position.

The run is *accepting* if $\text{inf}(q_0 q_1 \dots) \cap F \neq \emptyset$. The *nested-word language* of \mathcal{N} is $L(\mathcal{N}) := \{w \in \hat{\Sigma}^\omega \mid \text{there is an accepting run of } \mathcal{N} \text{ on } w\}$.

Note that the NWA \mathcal{N} uses a stack implicitly here, where the stack operations are determined by the input letters. In every run the stack is initialized with the bottom stack symbol \perp . When the i th letter is internal, i.e., $w_i \in \Sigma_{\text{int}}$, \mathcal{N} pushes and immediately pops the symbol $\gamma_i \in \Gamma$. The transition is independent from γ_i and the top stack symbol. When the i th letter is a call, i.e., $w_i \in \Sigma_{\text{call}}$, \mathcal{N} pushes the symbol $\gamma_i \in \Gamma$ on the stack. The transition is independent from the top stack symbol. When the i th letter is a return, i.e., $w_i \in \Sigma_{\text{ret}}$, \mathcal{N} pushes and immediately pops the symbol $\gamma_i \in \Gamma$. Furthermore, it pops a symbol from the stack. If that symbol is \perp then i is a pending return position and \mathcal{N} pushes that symbol again on the stack. If the symbol is in Γ then that symbol was previously pushed on the stack at the matching call position. Finally, we remark that the $\gamma_i \in \Gamma$ in a run, where i is an internal position or a return position are irrelevant.

B.2 Proof Details for Lemma 5

We prove the *only if* direction. Assume $(w, \curvearrowright) \in \hat{\Sigma}^\omega \setminus L(\mathcal{A})$, i.e., every infinite run of \mathcal{A} on (w, \curvearrowright) visits a state in F infinitely often.

We construct a word $R \in (2^Q)^\omega$ that satisfies conditions (1) and (2). We need the following definition. A word $(h_0, q_0) \dots (h_n, q_n) \in (\mathbb{N} \times Q)^*$ is a *run segment* if for all $i < n$, we have $(q_{i+1}, d) \in \delta_{\ell(i)}(q_i, w_{h_i})$, for all $d \in \mathbb{D}$ and $h_i \curvearrowright_d h_{i+1}$. The run segment is *initial* if $(h_0, q_0) = (0, q_I)$. For $i \in \mathbb{N}$, we define

$$R_i := \{q_n \in Q \mid \text{there is an initial run segment } (h_0, q_0) \dots (h_n, q_n) \text{ with } h_n = i\}.$$

That is, R_i contains all states that can be reached by an initial run segment of \mathcal{A} on w that ends with its read-only head at position i . By definition, R satisfies conditions (1) and (2).

Condition (3) is also fulfilled, since otherwise there is an initial run segment of the form $(h_0, q_0) \dots (h_n, q_n)$ with $(h_n, q_n) = (h, q)$, for some $q \in Q$ and $h \in \mathbb{N}$ with $\emptyset \not\equiv \delta(q, w_h)$. But then \mathcal{A} accepts (w, \curvearrowright) , which contradicts the assumption $(w, \curvearrowright) \notin L(\mathcal{A})$.

Now, we define a word $S \in (2^{Q \setminus F})^\omega$ that satisfies the conditions (4)–(6) of the lemma. For a nested word, we call a position $k \in \mathbb{N}$ *sync position* if there is no edge $i \curvearrowright_2 j$ such that $i \leq k$ and $j > k$. Let $b \in \{0, 1\}^\omega$ be an infinite word such the bit $b_k = 0$ iff $k \in \mathbb{N}$ is a sync position. In the following, we define S inductively. For convenience, let $S_{-1} := \emptyset$ and $b_{-1} = 0$. Let $m \in \mathbb{N} \cup \{-1\}$ such that $S_m = \emptyset$ and $b_m = 0$. For every m , we define the word $T^m \in (Q \times \mathbb{N})^\omega$ as the set of F -avoiding run segments that start in $R_{m+1} \setminus F$. For brevity, we just write T instead of T^m . Formally, for $i \leq m$, we define $T_i := \emptyset$ and for $i > m$, we define

$$T_i := \{q_k \in Q \mid \text{there is an } F\text{-avoiding run segment } (h_0, q_0) \dots (h_k, q_k) \\ \text{with } q_0 \in R_{m+1}, h_0 = m + 1, \text{ and } h_k = i\}.$$

Next, we show that there is a position $n > m$ such that $T_n = \emptyset$ and $b_n = 0$. Intuitively, every run segment that starts in $R_{m+1} \setminus F$ and visits positions after n , (a) visits position n because of condition $b_n = 0$, and (b) visits an F -state before it reaches position n because of condition $T_n = \emptyset$. Let n be the smallest position after m such that $T_n = \emptyset$ and $b_n = 0$. We argue that the position n exists. For the sake of contradiction, assume that this n does not exist.

We first show that the b sequence has value 0 at infinitely many positions. Formally, for every position $i > m$, there is a position $j \geq i$ such that $b_j = 0$. Consider a position $i > m$. If $b_i = 0$, we are done. Otherwise, let h be the least matched call position such that its matching return position k is greater than i . Since edges in \curvearrowright_2 do not cross, we have $b_k = 0$.

Now, we consider the following graph. The vertices are elements of $\{(i, q) \in \mathbb{N} \times Q \mid i \in \mathbb{N}, q \in T_i, \text{ and } b_i = 0\}$. This set is infinite since by assumption, there are infinitely many positions $i \in \mathbb{N}$, where $b_i = 0$ and $T_i \neq \emptyset$. There is an edge from (h, q) to (h', q') if the automaton \mathcal{A} can move from configuration (h, q) to (h', q') , i.e., there is a direction $d \in \mathbb{D}$ such that $h \curvearrowright_d h'$ and $(q', d) \in$

$\delta_{\ell(h)}(q, w_h)$. Note that each node has only finitely many successors. Furthermore, every node is reachable by some node in the finite set $\{(m+1, q) \mid q \in T_{m+1}\}$. By König's Lemma, the graph contains an infinite path. Note that for each tuple (h, q) in that path, we have $q \notin F$. Thus, there is an accepting infinite run of \mathcal{A} on (w, \curvearrowright) . This contradicts the assumption $(w, \curvearrowright) \notin L(\mathcal{A})$.

For the positions $i \in \mathbb{N}$ with $m < i \leq n$, we define $S_i := T_i$.

By the definition of S , conditions (4) and (6) are fulfilled. The sequence S also fulfills condition (5). This proof is similar to the proof from above that shows that R fulfills condition (2).

Now, we prove the *if* direction. Assume there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that satisfy conditions (1)–(6) of the lemma.

For the sake of contradiction, assume there is an accepting run r of \mathcal{A} on (w, \curvearrowright) . We make a case distinction. Suppose that r is finite and has the form $(h_0, q_0)(h_1, q_1) \dots (h_n, q_n) \in (\mathbb{N} \times Q)^*$, for some $n \in \mathbb{N}$. Note that the conditions (1) and (2) ensure that $q_i \in R_{h_i}$, for all $i \leq n$. Since r ends in the configuration (h_n, q_n) , we have $\emptyset \models \delta_{\ell(h_n)}(q_n, w_{h_n})$. We obtain a contradiction to condition (3).

Suppose that $r := (h_0, q_0)(h_1, q_1) \dots \in (\mathbb{N} \times Q)^\omega$ is infinite. Note that the conditions (1) and (2) ensure that $q_i \in R_{h_i}$, for all $i \in \mathbb{N}$. Since \mathcal{A} is eventually $\{1, 2\}$ -way and r is accepting, there is an index $k \in \mathbb{N}$ such that for all $i \geq k$, we have $q_i \notin F$ and $h_i \curvearrowright_d h_{i+1}$ with $d \in \{1, 2\}$. By condition (6), there is a breakpoint at position $m > h_k$ with $S_m = \emptyset$ and $S_{m+1} = R_{m+1} \setminus F$. Moreover, there is no $i \curvearrowright_2 j$ such that $i \leq m$ and $j > m$. It follows that r must visit this breakpoint, i.e., there is an index l such that $h_l = m$ and thus, $q_l \in R_m$. By condition (6), the position m is not a matched call. Hence, $h_{l+1} = h_l + 1$. It follows that $q_{l+1} \in S_{m+1}$.

Now, we show that there is no further breakpoint after position m , which contradicts condition (6). Assume there is a breakpoint at position $n > m$ with $S_n = \emptyset$ and $S_{n+1} = R_{n+1} \setminus F$. By condition (5), the fact that $q_{l+1} \in S_{m+1}$, and the fact that there is no $i \curvearrowright_2 j$ such that $i \leq n$ and $j > n$, the run r must visit S_n , i.e., there is an index l' such that $h_{l'} = n$ and $q_{l'} \in S_n$. However, this contradicts the fact that S_n is empty. \square

B.3 Proof Details for Theorem 6

Let $\mathcal{A} = (Q, \Sigma, (\delta_D)_{D \subseteq \mathbb{D}}, q_I, F)$ be an eventually $\{1, 2\}$ -way ECA. We define an NWA \mathcal{B} that stepwise guesses the words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ from Lemma 5 and locally checks that the conditions of Lemma 5 hold. A state of \mathcal{B} is a window frame consisting of two successive letters (R_i, R_{i+1}) from R and two successive letters (S_i, S_{i+1}) from S , for $i \in \mathbb{N}$. A stack symbol of \mathcal{B} is a window frame consisting of two letters (R_i, R_j) from R and (S_i, S_j) from S , where $i, j \in \mathbb{N}$ are matching positions. Using these two frames, the automaton locally checks the first five conditions of Lemma 5. Furthermore, \mathcal{B} guesses whether a call position is pending or not. Whenever a pending call occurs, \mathcal{B} pushes the special symbol **pending** on the stack. The automaton is not allowed to pop the symbol **pending** later on. In an extra bit in its states and stack symbols, \mathcal{B} stores the information

whether a matched call has already been answered by its matching return. The bit of the following state is set, if the current bit is already set or a matched call is pushed on the stack. In the second case, the current bit is stored in the stack to restore the bit of the state when the matching return position is reached. Finally, the automaton ensures that the last condition of Lemma 5 holds by its Büchi acceptance condition.

Formally, we define the NWA $\mathcal{B} := (P, \Gamma, \Sigma, \eta, p_I, G)$ as follows.

- $P := \Gamma := (2^Q \times 2^{Q \setminus F} \times 2^Q \times 2^{Q \setminus F} \times \{0, 1\}) \cup \{p_I, \text{pending}\}$. A state can be the initial state p_I , the state **pending** or a quintuple. Consider the case where the automaton processes the i th input letter of (w, \curvearrowright) . Then, the first two components of the quintuple correspond to the actual guessed sets R_i and S_i of the words R and S , respectively. The next two components of the quintuple correspond to the following guessed sets R_{i+1} and S_{i+1} of the words R and S , respectively. The last component signals whether there is still a symbol on the stack that originates from a matched call and hence, has to be pushed later on.
- $G := 2^Q \times \{\emptyset\} \times 2^Q \times 2^{Q \setminus F} \times \{0\}$. That is, the last component must be 0 infinitely often, i.e., all symbols that have been pushed on the stack at matched calls are processed infinitely often. Furthermore, the second component gets empty infinitely often, i.e., every time the second component is filled by states that owe a to visit to F , there will be a time, where each state in that component has visited F .

For brevity, we use pattern matching in the definition of the transition function. For instance, we write $\eta((R_{-1}, S_{-1}, R_0, S_0, b), a) \ni (R_0, S_0, R_1, S_1, 1)$ meaning that the transition $\eta((R_{-1}, S_{-1}, R_0, S_0, b), a)$ contains $(R'_0, S'_0, R_1, S_1, b')$, where the following three constraints (i) $R'_0 = R_0$, (ii) $S'_0 = S_0$, and (iii) $b' = 1$ hold. Next, we define the transition function in two steps.

Step 1. First, we define the transitions from the initial state p_I . For an internal letter $a \in \Sigma_{int}$, we have

$$\eta_{int}(p_I, a) \ni (R_0, S_0, R_1, S_1, 0)$$

iff for $D := \{0, 1\}$, the following conditions hold:

1. $q_I \in R_0$.
2. For all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
3. For all $p \in R_0$, we have $\emptyset \not\equiv \delta_D(p, a)$.
4. $S_0 = R_0 \setminus F$.
5. For all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in S_d$.

For a call letter $a \in \Sigma_{call}$, we have

$$\eta_{call}(p_I, a) \ni ((R_0, S_0, R_1, S_1, 1), (R_0, S_0, R_2, S_2, 0))$$

iff for $D := \{0, 1, 2\}$, the following conditions hold:

1. $q_I \in R_0$.

2. For all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
3. For all $p \in R_0$, $q \in Q$, if $(q, 2) \in \delta_D(p, a)$ then $q \in R_2$.
4. For all $p \in R_0$, we have $\emptyset \not\equiv \delta_D(p, a)$.
5. $S_0 = R_0 \setminus F$.
6. For all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in S_d$.
7. For all $p \in S_0$, $q \notin F$, if $(q, 2) \in \delta_D(p, a)$ then $q \in S_2$.

Furthermore, $\eta_{call}(p_I, a) \ni ((R_0, S_0, R_1, S_1, 0), \text{pending})$ iff for $D := \{1\}$, the conditions 1., 2., 4., 5., and 6. from above hold.

For a return letter $a \in \Sigma_{ret}$, the transitions are the same as for an internal position, i.e., for a stack symbol $s \in S$, we have

$$\eta_{ret}(p_I, s, a) \ni (R_0, S_0, R_1, S_1, 0)$$

iff $\eta_{int}(p_I, a) \ni (R_0, S_0, R_1, S_1, 0)$.

Step 2. Now, we define the transitions from states in $P \setminus \{p_I\}$. For an internal letter $a \in \Sigma_{int}$, we have

$$\eta_{int}((R_{-1}, S_{-1}, R_0, S_0, b), a) \ni (R_0, S_0, R_1, S_1, b)$$

iff for $D := \{-1, 0, 1\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, we have $\emptyset \not\equiv \delta_D(p, a)$.
3. For all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in S_d$.
4. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

For a call letter $a \in \Sigma_{call}$, we have

$$\eta_{call}((R_{-1}, S_{-1}, R_0, S_0, b), a) \ni ((R_0, S_0, R_1, S_1, 1), (R_0, S_0, R_2, S_2, b))$$

iff for $D := \{-1, 0, 1, 2\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, $q \in Q$, if $(q, 2) \in \delta_D(p, a)$ then $q \in R_2$.
3. For all $p \in R_0$, we have $\emptyset \not\equiv \delta_D(p, a)$.
4. For all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in S_d$.
5. For all $p \in S_0$, $q \notin F$, if $(q, 2) \in \delta_D(p, a)$ then $q \in S_2$.
6. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

Furthermore, $\eta_{call}((R_{-1}, S_{-1}, R_0, S_0, b), a) \ni ((R_0, S_0, R_1, S_1, 0), \text{pending})$ iff for $D := \{-1, 0, 1\}$, the conditions 1., 3., 4., 5., and 7. from above hold.

For a return letter $a \in \Sigma_{ret}$, we have

$$\eta_{ret}((R_{-1}, S_{-1}, R_0, S_0, b), (R_{-2}, S_{-2}, R_0, S_0, c), a) \ni (R_0, S_0, R_1, S_1, c)$$

iff for $D := \{-2, -1, 0, 1\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, $q \in Q$, if $(q, -2) \in \delta_D(p, a)$ then $q \in R_{-2}$.

3. For all $p \in R_0$, we have $\emptyset \not\equiv \delta_D(p, a)$.
4. For all $p \in S_0$, $q \notin F$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in S_d$.
5. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

Note that there is no transition if the symbol pending is on the stack.

It remains to show that $L(\mathcal{B}) = \hat{\Sigma}^\omega \setminus L(\mathcal{A})$.

First, we prove $L(\mathcal{B}) \subseteq \hat{\Sigma}^\omega \setminus L(\mathcal{A})$. Assume that \mathcal{B} accepts w by the run r that has the form $(P_0, O_0)(P_1, O_1) \dots \in (P \times \Gamma)^\omega$. We show that there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that fulfill the conditions of Lemma 5.

We construct the words R and S . By definition of the transition function, $P_0 = p_I$ and for all $i > 0$, each P_i is a tuple of the form $(A_i, B_i, C_i, D_i, b_i) \in (2^Q \times 2^{Q \setminus F} \times 2^Q \times 2^{Q \setminus F} \times \{0, 1\})$. Define $R_i := A_{i+1}$ and $S_i := B_{i+1}$, for all $i \in \mathbb{N}$.

We show that the words R and S satisfy the conditions of Lemma 5. Obviously, conditions (1), (3) and (4) are fulfilled. We show that condition (2) is fulfilled.

Let $p \in R_0$ and $q \in \delta_{\ell(0)}^d(p, w_0)$, for $p, q \in Q$ and $d \in \{0, 1\}$. Then we have $p \in A_1$ and by definition of the transition function from the initial state p_I , we obtain $q \in A_{1+d}$. Hence, $q \in R_{0+d}$. Now, let $p \in R_i$, $q \in \delta_{\ell(i)}^d(p, w_i)$, for $i > 0$, $p, q \in Q$ and $d \in \{-1, 0, 1\}$. Then we have $p \in A_i$ and by definition of the transition function from the states in $P \setminus \{p_I\}$, we obtain $q \in A_{i+d}$. Hence, $q \in R_{i+d}$.

Let i be a call position in w with matching return position j . Furthermore, let (A, B, C, D, b) be the symbol that is pushed on the stack by \mathcal{B} when reading letter w_i (and popped when reading letter w_j). Finally, let $p \in R_i$ and $q \in \delta_{\ell(i)}^2(p, w_i)$, for $p, q \in Q$. By definition of the transition function when reading the call w_i , we have $q \in C$. By definition of the transition function when reading the return w_j , we have $C = R_j$ and hence, $q \in R_j$.

Let i be a call position in w with matching return position j , and (A, B, C, D, b) be the symbol that is pushed on the stack by \mathcal{B} when reading letter w_i (and popped when reading letter w_j). Furthermore, let $p \in R_j$ and $q \in \delta_{\ell(i)}^{-2}(p, w_j)$, for $p, q \in Q$. By definition of the transition function when reading the return w_j , we have $p \in A$. By definition of the transition function when reading the call w_i , we have $A = R_i$ and hence, $p \in R_i$. The proof for condition (5) is similar to the proof above for condition (2).

Condition (6) directly follows from the acceptance condition. In particular, note that for all $i \in \mathbb{N}$, if $b_i = 0$ then all non-pending calls at position $j \leq i$ in w are matched by a return at position $k \leq i$ in the run r . Furthermore, the transition function ensures that an empty Γ -component is filled up as soon as the bit component is 0. Therefore, condition (6) is fulfilled because a state from G occurs infinitely often.

Now, we prove the other direction, namely, $L(\mathcal{B}) \supseteq \hat{\Sigma}^\omega \setminus L(\mathcal{A})$. Assume $(w, \rightsquigarrow) \notin L(\mathcal{A})$. Then, there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that fulfill the conditions (1)–(6) of Lemma 5. Without loss of generality, we may assume

that there is no position $i \in \mathbb{N}$ such that 1. $S_i = S_{i+1} = \emptyset$ and 2. every non-pending call with position $j \leq i$ is matched by a return $k \leq i$. In other words, every S_i is filled up as soon as every non-pending call with position $j \leq i$ is matched before or at position i . For more details, see the construction of S in the proof of Lemma 5.

We define an accepting run $r := (P_0, O_0)(P_1, O_1) \dots \in (P \times \Gamma)^\omega$ of \mathcal{B} on (w, \curvearrowright) as follows: $P_0 := p_I$ and for $i > 0$, we define $P_i := (R_{i-1}, S_{i-1}, R_i, S_i, 0)$ if each matched call with position $j \leq i$ has its matching return with position $k \leq i$ and otherwise, $P_i := (R_{i-1}, S_{i-1}, R_i, S_i, 1)$. Next, we define the symbols that are pushed on the stack at call positions. For every pending call with position i , we define $O_i := \text{pending}$. For every non-pending call with position i and matching return position j , we define $O_i := (R_i, S_i, R_j, S_j, b)$, where b has the value of the bit in P_i .

In the following, we show that r is an accepting run. We show that for all $i \in \mathbb{N}$, there is a transition in \mathcal{B} from r_i to r_{i+1} when reading the letter w_i . First, there is a transition from r_0 to r_1 when reading the letter w_0 . Consider the case, where 0 is an internal or return position. By definition of r_0 , the automaton is in state p_I , reads w_0 and goes to state $(R_0, S_0, R_1, S_1, 0)$. Obviously, all five conditions for this transition are fulfilled. Consider the case, where 0 is a call position with matching return position $j \in \mathbb{N}$. By definition of r_0 , the automaton is in state p_I , reads w_0 and goes to state $(R_0, S_0, R_1, S_1, 1)$ and pushes $(R_0, S_0, R_j, S_j, 0)$ on the stack. Obviously, all seven conditions for this transition are fulfilled. Finally, consider the case, where 0 is a pending call position. By definition of r_0 , the automaton is in state p_I , reads w_0 and goes to state $(R_0, S_0, R_1, S_1, 1)$ and pushes pending on the stack. Obviously, conditions (1), (2), (4), (5), and (6) for this transition are fulfilled. In a similar way, we can show that there is a transition from r_i to r_{i+1} when reading the letter w_i , for any $i > 0$.

By condition (6) of Lemma 5 and the remark at the beginning of the proof, states in G occur infinitely often. So, \mathcal{B} accepts (w, \curvearrowright) .

B.4 Proof Details for Theorem 7

Let $\mathcal{A} = (Q, \Sigma, (\delta_D)_{D \in \mathbb{D}}, q_I, F)$ be a very weak, eventually $\{1, 2\}$ -way ECA. We define $\bar{F} := Q \setminus F$. For the ease of exposition, we assume, without loss of generality, that \bar{F} contains the unreachable state $\hat{q} \in Q$, i.e., \hat{q} does not occur in any $\delta_D(q, a)$, for all $D \subseteq \mathbb{D}$, $q \in Q$, and $a \in \hat{\Sigma}$.

The proof is similar to the proof of Theorem 6 and we will reuse several proof steps. We construct an NWA \mathcal{B} and then show that \mathcal{B} accepts the language $\hat{\Sigma}^\omega \setminus L(\mathcal{A})$. Intuitively, \mathcal{B} guesses the word $R \in (2^Q)^\omega$ from Lemma 5 and locally checks the modified conditions of the lemma. For the non-local conditions of the lemma, it uses the stack that links calls with its matching returns and the Büchi acceptance condition.

We define the NWA $\mathcal{B} := (P, \Gamma, \Sigma, \eta, p_I, G)$ as follows.

- $P := \Gamma := (2^Q \times 2^Q \times \bar{F} \times \{0, 1\}) \cup \{p_I\}$. A state can be the initial state p_I or a quadruple. Consider the case where the automaton processes the i th input

letter of (w, \curvearrowright) . Then, the first two components of the triple correspond to the actual guessed sets R_i and R_{i+1} of the word R . The third component is called *focus*. It is used to find unfolding of self-loops of runs of \mathcal{A} . The last component signals whether there is still a symbol on the stack that originates from a matched call and hence, has to be popped from the stack later on.

- $G := 2^Q \times 2^Q \times \{\hat{q}\} \times \{0\}$. That is, infinitely often the automaton \mathcal{B} visits a state, where \hat{q} is focused and each symbol that is pushed on the stack at a matched call is processed.

As in the proof of Theorem 6, we use pattern matching in the definition of the transition function. Consider a total ordering on the states in \overline{F} and a function $\text{next} : \overline{F} \rightarrow \overline{F}$ that maps the greatest state \hat{q} to the smallest one and each of the other states to the next greater state. We also use the synonym $\text{pending} := p_I$ as syntactic abbreviation.

Step 1. First, we define the transitions from the initial state p_I . For an internal letter $a \in \Sigma_{\text{int}}$, we have

$$\eta_{\text{int}}(p_I, a) \ni (R_0, R_1, \hat{q}, 0)$$

iff for $D := \{0, 1\}$ the following conditions hold:

1. $q_I \in R_0$.
2. For all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
3. For all $p \in R_0$, we have $\text{tt} \notin \delta_D(p, a)$.

For a call letter $a \in \Sigma_{\text{call}}$, we have

$$\eta_{\text{call}}(p_I, a) \ni ((R_0, R_1, \hat{q}, 1), (R_0, R_2, \hat{q}, 0))$$

iff for $D := \{0, 1, 2\}$ the following conditions hold:

1. $q_I \in R_0$.
2. For all $p \in R_0$, $q \in Q$, and $d \in \{0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
3. For all $p \in R_0$ and $q \in Q$, if $(q, 2) \in \delta_D(p, a)$ then $q \in R_2$.
4. For all $p \in R_0$, we have $\text{tt} \notin \delta_D(p, a)$.

Furthermore, $\eta_{\text{call}}(p_I, a) \ni ((R_0, R_1, \hat{q}, 0), \text{pending})$ iff for $D := \{0, 1\}$, the conditions 1., 2., and 4. from above hold.

For a return letter $a \in \Sigma_{\text{ret}}$, the transitions are the same as for an internal position, i.e., for a stack symbol $s \in \Gamma$, we have

$$\eta_{\text{ret}}(p_I, s, a) \ni (R_0, R_1, \hat{q}, 0)$$

iff $\eta_{\text{int}}(p_I, a) \ni (R_0, R_1, \hat{q}, 0)$.

Step 2. Now, we define the transitions from states in $P \setminus \{p_I\}$. For an internal letter $a \in \Sigma_{\text{int}}$, we have

$$\eta_{\text{int}}((R_{-1}, R_0, s_0, b), a) \ni (R_0, R_1, s_1, b)$$

iff for $D := \{-1, 0, 1\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, we have $\text{tt} \notin \delta_D(p, a)$.
3. $s_1 = \begin{cases} s_0 & \text{if either } s_0 \in R_0 \text{ and } (s_0, 1) \in \delta_D(s_0, a) \text{ or} \\ & s_0 = \hat{q} \text{ and } b = 1, \\ \text{next}(s_0) & \text{otherwise.} \end{cases}$

For a call letter $a \in \Sigma_{\text{call}}$, we have

$$\eta_{\text{call}}((R_{-1}, R_0, s_0, b), a) \ni ((R_0, R_1, s_1, 1), (R_0, R_2, s_2, b))$$

iff for $D := \{-1, 0, 1, 2\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, $q \in Q$, if $(q, 2) \in \delta_D(p, a)$ then $q \in R_2$.
3. For all $p \in R_0$, we have $\text{tt} \notin \delta_D(p, a)$.
4. For $d \in \{1, 2\}$, we have

$$s_d = \begin{cases} s_0 & \text{if either } s_0 \in R_0 \text{ and } (s_0, d) \in \delta(s_0, a) \text{ or } s_0 = \hat{q} \text{ and } b = 1, \\ \text{next}(s_0) & \text{otherwise.} \end{cases}$$

Furthermore, $\eta_{\text{call}}((R_{-1}, R_0, s_0, b), a) \ni ((R_0, R_1, s_1, 0), \text{pending})$ iff for $D := \{-1, 0, 1\}$, the conditions 1., 3., and 4. from above hold.

For a return letter $a \in \Sigma_{\text{ret}}$, we have

$$\eta_{\text{ret}}((R_{-1}, R_0, s_0, 1), (R_{-2}, R_0, s'_0, b), a) \ni (R_0, R_1, s_1, b)$$

iff for $D := \{-2, -1, 0, 1\}$, the following conditions hold:

1. For all $p \in R_0$, $q \in Q$, and $d \in \{-1, 0, 1\}$, if $(q, d) \in \delta_D(p, a)$ then $q \in R_d$.
2. For all $p \in R_0$, $q \in Q$, if $(q, -2) \in \delta_D(p, a)$ then $q \in R_{-2}$.
3. For all $p \in R_0$, we have $\text{tt} \notin \delta_D(p, a)$.
4. For s_{\min} being the minimum of s_0 and s'_0 , we have

$$s_1 = \begin{cases} s_{\min} & \text{if either } s_{\min} \in R_0 \text{ and } (s_{\min}, 1) \in \delta_D(s_{\min}, a) \text{ or} \\ & s_{\min} = \hat{q} \text{ and } b = 1, \\ \text{next}(s_{\min}) & \text{otherwise.} \end{cases}$$

Additionally, for a pending return position, we have $\eta_{\text{ret}}((R_{-1}, R_0, s_0, 0), \perp, a) \ni (R_0, R_1, s_1, 0)$ iff the transition $\eta_{\text{int}}((R_{-1}, R_0, s_0, 0), a)$ contains $(R_0, R_1, s_1, 0)$ as defined above. Note that there is no transition if the symbol **pending** is on the stack.

First, we prove $L(\mathcal{B}) \subseteq \hat{\Sigma}^\omega \setminus L(\mathcal{A})$. Assume that \mathcal{B} accepts w by the run r that has the form $(P_0, O_0)(P_1, O_1) \dots \in (P \times \Gamma)^\omega$. We further assume that the automaton \mathcal{B} pushed **pending** on the stack whenever it process a pending call position. Since this additional assumption just changes the bit component of the states in the run r to 0, the run remains accepting. We construct a word $R \in (2^Q)^\omega$ that fulfills the modified conditions of Lemma 5.

We construct the word R . By definition of the transition function, $P_0 = p_I$ and for all $i > 0$, each P_i is a tuple of the form $(A_i, B_i, s_i, b_i) \in (2^Q \times 2^{\overline{F}} \times \overline{F} \times \{0, 1\})$. Define $R_i := A_{i+1}$, for all $i \in \mathbb{N}$. We show that the word R satisfies the modified conditions of Lemma 5. Obviously, conditions (1) and (3) are fulfilled. Reusing the proof of Theorem 6, we obtain the fact that condition (2) is also fulfilled.

It remains to show that R satisfies condition (6'). For the sake of contradiction, suppose that condition (6') does not hold. Let $q \in \overline{F}$ and $i \in \mathbb{N}^\omega$ be an infinite sequence of positions such that $q \in R_{i_0}$ and for all $j \in \mathbb{N}$, there is a direction $d \in \{1, 2\}$ such that $(i_j, i_{j+1}) \in E_d$ and $(q, d) \in \delta_{\ell(i_j)}(q, w_{i_j})$. We show that the automaton \mathcal{B} will eventually *catch* this sequence of self-repeating q 's with its third component of its states, which implies that the run r is rejecting.

Let $m \in \mathbb{N}$ be the first position after i_0 such that $s_m = \hat{q}$ and $b_m = 0$ and $n \in \mathbb{N}$ be the first position after m such that $s_n = \hat{q}$ and $b_n = 0$. Note that by definition of the transition function, for any position k between m and n with $s_k < q$, there is a position l after k and before m with $s_l = q$. Otherwise, s_n cannot be assigned the value \hat{q} .

By definition of the transition function, s_{m+1} focuses the minimal state. Hence, there is a k before m such that $s_k = q$. Pick k such that k is the largest position before m such that $s_k = q$. We conclude with the following contradiction. 1. If there is some $j \in \mathbb{N}$ such that $i_j = k$ then $q \in R_k$. If i_j is an internal or return position then $i_{j+1} = i_j + 1$ and $q \in R_{i_{j+1}}$. By definition of the transition function, we have $s_{k+1} = q$. That contradicts the choice of k . If i_j is a call position with matching return position i_{j+1} then $b_h = 1$, for all $i_j < h \leq i_{j+1}$, and $s_{i_{j+1}} = q$. Note that position m comes after position i_{j+1} since $b_m = 0$. The fact that $i_{j+1} > k$ contradicts the choice of k . 2. If there is no $j \in \mathbb{N}$ such that $i_j = k$ then let $j \in \mathbb{N}$ such that $i_j < k$ and $i_{j+1} > k$. Note that i_j is a call with matching return i_{j+1} and both positions are between m and n . If $s_{i_j} = q$ then $s_h \geq q$, for all $i_j \leq h < i_{j+1}$. Hence, by definition of the transition function, we have $s_{j+1} = q$. This contradicts the choice of k . If $s_{i_j} > q$ then s_k must also be greater than k since there are no crossings of 2-direction edges. If $s_{i_j} < q$ then $s_{i_{j+1}} \leq q$, by definition of the transition function. From the argumentation above, we infer that there is a position $l > k$ such that $s_l = q$. This contradicts the choice of k .

Now, we prove $L(\mathcal{B}) \supseteq \hat{\Sigma}^\omega \setminus L(A)$. Assume $(w, \curvearrowright) \notin L(A)$. Then, there is a word $R \in (2^Q)^\omega$ that fulfills the modified conditions of Lemma 5.

We construct an accepting run of \mathcal{B} on (w, \curvearrowright) . We define the run $r := (P_0, O_0)(P_1, O_1) \dots \in (P \times \Gamma)^\omega$ as follows. We first define the sequence of the bit component $b \in \{0, 1\}^\omega$. For $i \in \mathbb{N}$, b_i is 0 if each matched call position $j \leq i$ has its matching return position $k \leq i$. Otherwise, b_i is 1. We first define the sequences $s, s' \in \overline{F}^\omega$ recursively. We define $s'_0 = s_0 := \hat{q}$ and for $i \in \mathbb{N}$: 1. if i is

an internal, call, or pending-return position then for $D := \{-1, 0, 1\}$,

$$s_{i+1} := \begin{cases} s_i & \text{if } s_i \in R_i \text{ and } (s_i, 1) \in \delta_D(s_i, w_i) \\ & \text{or } s_i = \hat{q} \text{ and } b_i = 1 \\ \text{next}(s_i) & \text{otherwise.} \end{cases}$$

and

$$s'_{i+1} := \begin{cases} s_i & \text{if } s_i \in R_i \text{ and } (s_i, 2) \in \delta_D(s_i, w_i) \\ & \text{or } s_i = \hat{q} \text{ and } b_i = 1 \\ \text{next}(s_i) & \text{otherwise.} \end{cases}$$

2. if i is a return position with matched call j then for $D := \{-2, -1, 0, 1\}$,

$$s'_{i+1} = s_{i+1} := \begin{cases} s_{min} & \text{if } s_{min} \in R_0 \text{ and } (s_{min}, 1) \in \delta_D(s_{min}, w_i) \\ & \text{or } s_{min} = \hat{q} \text{ and } b_i = 1 \\ \text{next}(s_{min}) & \text{otherwise,} \end{cases}$$

where s_{min} is the minimum of s'_{j+1} and s_i . We define $P_0 := p_I$ and for all $i \in \mathbb{N}$, we define $P_{i+1} := (R_i, R_{i+1}, s_i, b_i)$. Next, we define the symbols that are pushed on the stack at call positions. For every pending call with position i , we define $O_{i+1} := \text{pending}$. For every non-pending call with position i and matching return position j , we define $O_{i+1} := (R_i, R_j, s'_j, b_i)$.

By construction, r is a run. In the following, we show that r is accepting. Assume the opposite. Then, there is a sequence of positions $i \in \mathbb{N}^\omega$ such that $s_{i_0} \in \overline{F}$ and for all $j \in \mathbb{N}$, we either have (a) $i_{j+1} = i_j + 1$ and $s_{i_{j+1}} = s_{i_j}$, or (b) i_j is a call with matching return i_{j+1} and $s_{i_{j+1}} = s_{i_j}$. Thus, condition (6') does not hold.

B.5 Proof Details for Theorem 10

For generality, we show how to translate an eventually $\{1, 2\}$ -way alternating Streett automaton (ASA) that accepts by memoryless runs into an NWA. Note that an APA is a restricted ASA, which accepts by memoryless runs. The outline of our translation is as follows.

- (a) We translate the eventually $\{1, 2\}$ -way ASA \mathcal{A} into a $\{1, 2\}$ -way alternating generalized Büchi automaton \mathcal{B} that accepts by memoryless and so-called *consistent* runs (see definitions below). For this translation, we first show that an eventually $\{1, 2\}$ -way ASA \mathcal{A} accepts a nested word iff it has a so-called *accepting Streett ranking* (see definition below). Then, we give the construction of the automaton \mathcal{B} and show that it accepts a nested word iff \mathcal{A} has an accepting Streett ranking.
- (b) Finally, we translate the automaton \mathcal{B} into an NWA. In the construction of the NWA, we exploit the fact that \mathcal{B} accepts by memoryless and consistent runs.

Streett Ranking In the following, let $(w, \curvearrowright) \in \hat{\Sigma}^\omega$ be a nested word and $\mathcal{A} = (Q, \Sigma, (\delta_D)_{D \subseteq \mathbb{D}}, q_I, F)$ an eventually $\{1, 2\}$ -way ASA, where its acceptance condition is $F = \{(C_1, B_1), \dots, (C_k, B_k)\}$. Furthermore, we assume that \mathcal{A} accepts by memoryless runs, i.e., we require $L(\mathcal{A}) = M(\mathcal{A})$.

For a memoryless run $r : R \rightarrow \mathbb{N} \times Q$ of \mathcal{A} on (w, \curvearrowright) , we define the directed graph $G := (V, E)$, which represents r , as follows:

$$V := \{v \in \mathbb{N} \times Q \mid v = r(x), \text{ for some } x \in R\}$$

and

$$E := \{e \in V \times V \mid e = (r(x), r(xi)), \text{ for some } x \in R \text{ and } i \in \mathbb{N}\}.$$

For $P \subseteq Q$, we call $(h, q) \in V$ a P -vertex if $q \in P$. The integer h is called the *head position* of v . Obviously, if r is accepting then every infinite path $(h_0, q_0)(h_1, q_1) \dots$ in G that starts with $(0, q_I)$ fulfills the Streett acceptance condition F , i.e., $\inf(q_0 q_1 \dots) \cap B_i \neq \emptyset$ or $\inf(q_0 q_1 \dots) \cap C_i = \emptyset$, for all $i \in \{1, \dots, k\}$.

Consider a function $f : V \rightarrow [2n]^k$. We denote the projection of f on the i th component by $f_i : V \rightarrow [2n]$ and call it i -rank, for $i \in \{1, \dots, k\}$. The function f is a *Streett ranking* for G if the following two conditions hold.

- (i) For all $v \in V$ and $i \in \{1, \dots, k\}$, if $f_i(v)$ is odd then v is not a C_i -vertex.
- (ii) For all $(v, v') \in E$ and $i \in \{1, \dots, k\}$, either $f_i(v) \geq f_i(v')$ or v is a B_i -vertex.

For $i \in \{1, \dots, k\}$, we say that the i -rank f_i is *accepting* if every infinite path in G either visits B_i -vertices infinitely often or gets trapped in an odd rank. Formally, f_i is accepting if for every infinite path $(h_0, q_0)(h_1, q_1) \dots$ in G , either $\inf(q_0 q_1 \dots) \cap B_i \neq \emptyset$ or there is an integer $n \in \mathbb{N}$ such that $f_i(h_n, q_n)$ is odd and $f_i(h_j, q_j) = f_i(h_n, q_n)$, for all $j \geq n$. The Streett ranking f is *accepting* if the i -rank f_i is accepting, for every $i \in \{1, \dots, k\}$.

Theorem 12. *Every infinite path in G fulfills the Streett acceptance condition F iff there is an accepting Streett ranking for G .*

In the following, we prove this theorem. It suffices to only consider the case, where $k = 1$. Obviously, by the definition of a Streett ranking, the lemma's statement generalizes then to any $k \geq 1$. To increase readability, we assume in the following that F is the singleton $\{(C, B)\}$.

The *if* direction is easy to see. Assume that there is an accepting Streett ranking f for G . That is, every infinite path in G visits B -vertices infinitely often or gets trapped in an odd 1-rank. By definition, this means, every infinite path visits B -vertices infinitely often or eventually avoids visiting C -vertices. Hence, every infinite path in G fulfills the Streett acceptance condition F .

For the *only if* direction, assume that every infinite path in G fulfills the Streett acceptance condition, i.e., every infinite path visits B -vertices infinitely often or eventually avoids visiting C -vertices. In the following, we construct a Streett ranking for G and show that it is accepting.

Consider a subgraph G' of G . We call a vertex $v \in V$ *finite* in G' if only finitely many vertices are reachable from v in G' . We call it *C-free* in G' if no C -vertex is reachable from v in G' . We define an infinite sequence of subgraphs G_0, G_1, G_2, \dots of G , where the vertices V_i and the edges E_i of a graph G_i are inductively defined as follows:

$$V_0 := V \quad \text{and} \quad E_0 := E \setminus \{(v, v') \in E \mid v \text{ is a } B\text{-vertex}\}$$

and for $i \in \mathbb{N}$, we define

$$V_{2i+1} := V_{2i} \setminus \{v \mid v \text{ is finite in } G_{2i}\} \quad \text{and} \quad E_{2i+1} := E_{2i} \cap V_{2i+1} \times V_{2i+1}$$

and

$$V_{2i+2} := V_{2i+1} \setminus \{v \mid v \text{ is } C\text{-free in } G_{2i}\} \quad \text{and} \quad E_{2i+2} := E_{2i+1} \cap V_{2i+2} \times V_{2i+2}.$$

Note that by removing edges from B -vertices in G , we obtain the graph G_0 , where every infinite path avoids B -vertices and eventually avoids visiting C -vertices.

Lemma 13. *For every $i \in \mathbb{N}$, the graph G_{2i+1} is empty or has a C -free vertex from which an infinite path of C -free vertices starts.*

Proof. Let $i \in \mathbb{N}$. Consider the graph G_{2i} . We make a case distinction. If G_{2i} is finite then G_{2i+1} is empty and we are done. Otherwise, if G_{2i} is infinite then G_{2i+1} is infinite. For the sake of contradiction, assume that there is no C -free vertex in G_{2i+1} . Note that every vertex in G_{2i+1} has at least one successor. Consider some vertex v_1 in G_{2i+1} . Let v'_1 be a successor of v_1 . Since v'_1 is not C -free, there is a C -vertex v_2 reachable from v'_1 . Let v'_2 be a successor of v_2 . Since v'_2 is not C -free, there is a C -vertex v_3 reachable from v'_2 . Let v'_3 be a successor of v_3 . If we continue this way, we can construct an infinite path that does not visit any B -vertex but visits C -vertices infinitely often. This path corresponds to a rejecting path in G , which contradicts the assumption that every infinite G fulfills the Streett acceptance condition. \square

In the next lemma, we show that we obtain an empty graph from G in $2n$ steps by alternately removing infinite paths according to Lemma 13 and finite vertices from G . Let $H \subseteq \mathbb{N}$ be the set of head position h such that h is not strictly between a call position and its matching return position, i.e.,

$$H := \{h \in \mathbb{N} \mid \text{there are no } i, j \in \mathbb{N} \text{ such that } i < h < j \text{ and } i \curvearrowright_2 j\}.$$

Lemma 14. *For every $i \in \mathbb{N}$, either G_{2i+1} is empty or there is a position $n \in \mathbb{N}$ such that for every $h \in H$ with $h \geq n$, we have $|\{q \mid (h, q) \in V_{2i+2}\}| < |\{q \mid (h, q) \in V_{2i+1}\}|$.*

Proof. If G_{2i+1} is empty, we are done. Otherwise, consider an infinite path π of the form $(h_0, q_0)(h_1, q_1) \dots \in (\mathbb{N} \times Q)^\omega$ in G_{2i+1} that consists of only C -free vertices. This path exists by Lemma 13. It suffices to show that for each $h \in H$ with $h \geq h_0$, there is a vertex (h, q) occurring in π . This is obvious since \mathcal{A} is eventually $\{1, 2\}$ -way and infinite paths cannot jump over the positions in H . \square

Corollary 15. G_{2n} is finite and G_{2n+1} is empty.

Proof. First, observe that H is infinite. Second, note that G_{2n} does not contain any vertex (h, q) with $h \in H$, since G contains at most n vertices with head position h and from Lemma 14, it follows that for each $i \in [n]$, at least one vertex in G_{2i+1} with head position h gets removed.

Assume (j, q) is a vertex of G_{2n} with $j \notin H$. Since H is infinite, there is a head position $h \in H$ with $h > j$. Since \mathcal{A} cannot jump over h , the vertex (j, q) can have only finitely many successors in G_{2n} . Therefore, it is not a vertex of G_{2n+1} . \square

We define the Streett ranking $f : V \rightarrow [2n]$ as follows and show that it is accepting.

$$f(v) := \begin{cases} 2i & \text{if } v \text{ is finite in } G_{2i}, \\ 2i + 1 & \text{if } v \text{ is } C\text{-free in } G_{2i+1}. \end{cases}$$

Obviously, f fulfills the first condition of the definition of a Streett ranking. The second condition follows from the next lemma.

Lemma 16. For all vertices $v, v' \in V$, we have $f(v') \leq f(v)$ if v' is reachable from v without visiting an B -vertex.

Proof. By the definition of f , we have the following fact. A vertex is not in G_i anymore if it has been removed from G_j before and hence, has rank j , for $i, j \in [2n]$ with $j < i$. Formally, for every $\hat{v} \in V$ and $i \in [2n]$, if $\hat{v} \notin G_i$ then $f(\hat{v}) < i$.

Assume that $f(v) = i$. We distinguish between three cases. (a) If $v' \notin G_i$ then $f(v') < f(v)$ by the fact from above. (b) If $v' \in G_i$ and i is even then v and v' are finite in G_i since v' is reachable from v without visiting an B -vertex in G_i , and hence, $f(v') \leq f(v)$. (c) If $v' \in G_i$ and i is odd then v and v' are C -free in G_i , and hence, $f(v') \leq f(v)$. \square

Finally, we show that f is accepting.

Lemma 17. Every infinite path in G that does not visit B -vertices infinitely often gets trapped in an odd rank.

Proof. Let $\pi = v_0 v_1 \dots$ be an infinite path in G that avoids visiting B -vertices infinitely often. According to Lemma 16, there must be a position $k \in \mathbb{N}$ in π such that $f(v_m) = f(v_k)$, for all $m \geq k$. We show that $f(v_k)$ is odd. By the sake of contradiction, assume that $f(v_k)$ is even. Then, every vertex v in the path that is reachable from v_k is finite in $G_{f(v_k)}$. The path is infinite, and therefore, v_k cannot be finite in $G_{f(v_k)}$. \square

Alternation Elimination We first construct an eventually $\{1, 2\}$ -way generalized Büchi automaton \mathcal{B} that accepts the same nested words as the given ASA \mathcal{A} . Additionally, we show that \mathcal{B} accepts by runs that have a special structure.

Second, we exploit this structure in the alternation-elimination construction to avoid an overall double-exponential blow-up when translating \mathcal{A} into an NWA.

Before we present these two constructions, we need the following definitions. A *generalized Büchi automaton* (AGA) is an automaton, where its acceptance condition A is given as a finite set of Büchi acceptance conditions $\{F_0, \dots, F_{k-1}\}$. This set is a finite description of the acceptance condition $A := \bigcap_{0 \leq i < k} \{\pi \in Q^\omega \mid \inf(\pi) \cap F_i \neq \emptyset\}$. Analogously, the acceptance condition A of a *generalized coBüchi automaton* is finitely represented as a set $\{F_0, \dots, F_{k-1}\}$, where $A := \bigcup_{0 \leq i < k} \{\pi \in Q^\omega \mid \inf(\pi) \cap F_i = \emptyset\}$.

Let Q be a set of states and $k \in \mathbb{N}$ an index. We call a set $P \subseteq Q \times [2n]^k$ *consistent* if for all $(q, r), (q', r') \in P$, it holds $r = r'$ whenever $q = q'$. We call a tree $t : T \rightarrow \mathbb{N} \times (Q \times [2n]^k)$ *consistent* if for every head position $h \in \mathbb{N}$, the set

$$\{(q, r) \mid x \in T \text{ and } t(x) = (h, (q, r))\}$$

is consistent. We call the first component of $t(x)$ the *head position* of the node $x \in T$.

Theorem 18. *For an eventually $\{1, 2\}$ -way ASA \mathcal{A} with n states and index k that accepts by memoryless runs, there is an eventually $\{1, 2\}$ -way AGA \mathcal{B} with $n^{\mathcal{O}(k)}$ states and $L(\mathcal{B}) \cap \hat{\Sigma}^\omega = L(\mathcal{A}) \cap \hat{\Sigma}^\omega$. Furthermore, \mathcal{B} accepts by memoryless and consistent runs.*

Proof. Assume that $\mathcal{A} = (Q, \hat{\Sigma}, (\delta_D)_{D \subseteq \mathbb{D}}, q_I, \{(C_0, B_0), \dots, (C_{k-1}, B_{k-1})\})$. For a state $q \in Q$ and two ranks $r, r' \in \mathbb{N}^k$, we write $r' \leq_q r$ when for every $i \in [k]$, we either have $r'_i \leq r_i$ or $q \in B_i$. Intuitively, the automaton may move to all possible successor states whose rank is lower. For a given formula $\varphi \in \text{Bool}^+(Q \times \mathbb{D})$, a state $q \in Q$ and a rank $r \in [2n]^k$, we define $\text{release}_q(\varphi, r)$ as the positive Boolean formula that we obtain by replacing each proposition (p, d) in φ by the disjunction $\bigvee_{r' \leq_q r} ((p, r'), d)$.

We define the AGA \mathcal{B} as $(Q \times [2n]^k, \hat{\Sigma}, (\eta_D)_{D \subseteq \mathbb{D}}, p_I, \{F_0, \dots, F_{k-1}\})$, where $p_I, (\eta_D)_{D \subseteq \mathbb{D}}$, and the F_i s are as follows:

- $p_I := (q_I, 2n, \dots, 2n)$.
- For $D \subseteq \mathbb{D}$, $q \in Q$, $r := (r_0, \dots, r_{k-1}) \in [2n]^k$, and $a \in \hat{\Sigma}$, we define

$$\eta_D((q, r), a) := \begin{cases} \text{release}_q(\delta_D(q, a), r) & \text{if } q \notin C_i \text{ or } r_i \text{ is even, for all } i \in [k], \\ \text{ff} & \text{otherwise.} \end{cases}$$

- For $j \in [k]$, the acceptance set F_j contains the state (q, i_0, \dots, i_{k-1}) iff $q \in B_j$ or i_j is odd. That is, every path in a run either visits B_j infinitely often or its j th's component gets trapped in an odd rank, for every $j \in [k]$.

It remains to prove that $L(\mathcal{B}) \cap \hat{\Sigma}^\omega = L(\mathcal{A}) \cap \hat{\Sigma}^\omega$. We first show that $L(\mathcal{B}) \cap \hat{\Sigma}^\omega \subseteq L(\mathcal{A}) \cap \hat{\Sigma}^\omega$. Let $(w, \curvearrowright) \in L(\mathcal{B})$ and let $r' : T \rightarrow \mathbb{N} \times (Q \times [2n]^k)$ be an accepting run of \mathcal{B} on (w, \curvearrowright) . Consider the tree $r : T \rightarrow \mathbb{N} \times Q$ with $r(x) := (h, q)$, for every $x \in T$ with $r'(x) = (h, (q, i))$. That is, r is the projection of r' on Q and

\mathbb{N} . The tree r is a run of \mathcal{A} on (w, \curvearrowright) since the transitions of \mathcal{B} just annotate the transitions of \mathcal{A} by ranks. We show that r is accepting. Since r' is accepting, every path in r' visits some (q, r) , where for every component r_i of r , we either have $q \in B_i$ or r_i is odd. Let $i \in [k]$. Consider a path in r' that does not visit any state from B_i . By definition of the acceptance condition, the path gets trapped in the set $\{(q, j) \in Q \times [2n]^k \mid j_i \text{ is odd}\}$. Thus, by definition of $(\eta_D)_{D \subseteq \mathbb{D}}$, the path eventually avoids visiting states from C_i . Consequently, the projection of the path on Q is an accepting path in r . Hence, r is accepting.

Now, we prove that $L(\mathcal{B}) \cap \hat{\Sigma}^\omega \subseteq L(\mathcal{A}) \cap \hat{\Sigma}^\omega$. Let $(w, \curvearrowright) \in L(\mathcal{A})$ and $r : T \rightarrow \mathbb{N} \times Q$ be an accepting run of \mathcal{A} on (w, \curvearrowright) . We define a tree $r' : T \rightarrow \mathbb{N} \times (Q \times [2n]^k)$ and show that it is an accepting run of \mathcal{B} on (w, \curvearrowright) . For $\varepsilon \in T$ with $r(\varepsilon) = (h, q)$, we define $r'(\varepsilon) := (h, (q, i_0, \dots, i_{k-1}))$, where $i_j = 2n$, for all $j \in [k]$. For $x \in T \setminus \{\varepsilon\}$ with $r(x) = (h, q)$, we define $r'(x) := (h, (q, i_0, \dots, i_{k-1}))$, where i_j is the value of the j -rank of the node (h, q) in G . We show that r is a run. By definition of r' , we have $r'(\varepsilon) = (0, (q_I, i_0, \dots, i_{k-1}))$ with $i_j = 2n$, for all $j \in [k]$. So, the root is well-labeled. Since every rank is smaller or equal $2n$, the first level of r' is also well-labeled. Now, consider a node $x \in T \setminus \{\varepsilon\}$ with $r(x) = (h, q)$. Let $(i_0, \dots, i_{k-1}) \in [2n]^k$ such that i_j is the value of the j -rank of (q, h) in G . Let $S := \{(h_1, q_1), \dots, (h_m, q_m)\}$ be the set of labels of the successors of node x . Consider some j -rank $f_j : \mathbb{N} \times Q \rightarrow [2n]$, for $j \in [k]$. By Lemma 16, the value of the j -rank of each node $f_j(h_l, q_l) \leq i_j$, for each $1 \leq l \leq m$. Furthermore, $q \in C_j$ and i_j is odd cannot hold at the same time, by definition of j -ranks. Thus, $S' := \{(h_1, (q_1, i'_1)), \dots, (h_m, (q_m, i'_m))\}$, where $i'_l = (i'_{l_0}, \dots, i'_{l_{k-1}})$ with $i'_{l_j} := f_j(h_l, q_l)$, for all $1 \leq l \leq m$ and $j \in [k]$, satisfies the transition $\eta_D((q, i_0, \dots, i_{k-1}), w_h)$. We have shown that r' is a run. Finally, by Lemma 17, every infinite path in G that does not visit B_j -vertices infinitely often, gets trapped in an odd j -rank, for every $j \in [k]$. By definition of the Büchi acceptance condition F , every infinite path in r' is accepting. Furthermore, note that r is a memoryless run and for every two configurations (h, q) and (h', q') in r with $(h, q) = (h', q')$, we have $f(h, q) = f(h', q')$. By definition of r' , it follows that r' is also memoryless. Hence, \mathcal{B} accepts by memoryless runs.

It remains to show that the run r' is consistent. By definition of r' , for every $x \in T$ with $r'(x) = (h, (q, i_0, \dots, i_{k-1}))$, (i_0, \dots, i_{k-1}) is the rank of vertex (h, q) in G . Since every vertex in G has a unique rank, r' is consistent. \square

Theorem 19. *For an eventually $\{1, 2\}$ -way ASA \mathcal{A} with n states and index k that accepts by memoryless runs, there is an NWA \mathcal{N} with $2^{\mathcal{O}(nk \log n)}$ states and stack symbols, and $L(\mathcal{N}) = L(\mathcal{A}) \cap \hat{\Sigma}^\omega$.*

Proof. Using Theorem 18, we translate the ASA \mathcal{A} into an eventually $\{1, 2\}$ -way AGA $\mathcal{B} = (Q, \hat{\Sigma}, (\delta_D)_{D \subseteq \mathbb{D}}, q_I, F)$ of size $n^{\mathcal{O}(k)}$ that accepts by memoryless and consistent runs. Let \mathcal{R} be the refuter automaton for \mathcal{B} according to the alternation-elimination scheme. Recall that \mathcal{R} is an existential generalized coBüchi automaton.

Consider a nested word $(w, \curvearrowright) \in \hat{\Sigma}^\omega$ and a memoryless consistent run $r : T \rightarrow \mathbb{N} \times Q$ of \mathcal{B} on (w, \curvearrowright) . We can represent this run by an infinite nested word

$(s, \curvearrowright) \in \hat{\Gamma}^\omega$, where $\Gamma := Q \rightarrow 2^{Q \times \mathbb{D}}$. Consider now the nested word $(w \times s, \curvearrowright)$, where $w \times s \in (\hat{\Sigma} \times \hat{\Gamma})^\omega$ with $(w \times s)(i) := (w(i), s(i))$, for all $i \in \mathbb{N}$, and a tree $r' : T' \rightarrow \mathbb{N} \times Q$ that represents all runs of the refuter automaton \mathcal{R} on $(w \times s, \curvearrowright)$, i.e., each path of the tree encodes a run of the automaton. We show that this tree is consistent. Intuitively, the refuter automaton just follows a path in the run given by (s, \curvearrowright) . So, the tree that represents all runs of the refuter automaton can only be a part of the run tree r . Pick an arbitrary head position $h \in \mathbb{N}$. We have to show that the set

$$S' := \{r'(x) \mid x \in T' \text{ and } r'(x) \text{ has head position } h\}$$

is consistent. We show that S' is a subset of the set

$$S := \{r(x) \mid x \in T \text{ and } r(x) \text{ has head position } h\},$$

which is consistent by assumption. Let $(h, q) \in S'$. Then, there is a prefix of a run of \mathcal{R} on the nested word $(w \times s, \curvearrowright)$ that has the form $(h_0, q_0)(h_1, q_1) \dots (h_n, q_n) \in (\mathbb{N} \times Q)^*$ with $(h_n, q_n) = (h, q)$. Since, $q_{i+1} \in s_i(q_i)$, for all $i \in [n]$, there is a node in r that is labeled by (h, q) . Hence, $(h, q) \in S$.

Since r' is consistent, we can optimize an extension of the construction of an NWA \mathcal{N} in Theorem 6 for complementing the language of the automaton \mathcal{R} . Let $\{F_0, \dots, F_{k-1}\}$ be the acceptance condition of \mathcal{R} . We add a counter $i \in [k]$ to each state of the resulting NWA \mathcal{N} . The NWA \mathcal{N} sequentially checks that there is no run of \mathcal{R} that gets trapped in some F_i . Along the lines of the construction of Theorem 6, the NWA \mathcal{N} guesses infinitely many sync positions $i_0, i_1, \dots \in \mathbb{N}$ of the given input word $w \in \hat{\Sigma}^\omega$ such that for every such sync position i_j , for $j \in \mathbb{N}$, and every run of \mathcal{R} that starts in a state that can be reached by reading the prefix $w_0 \dots w_{i_j+1}$, visits the set $Q \setminus F_{j \bmod k}$ before reaching position i_{j+1} in the input word. This is accomplished by changing the transition function from non-initial states in the construction of Theorem 6. Namely, condition “if $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$ ” is changed to: whenever $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F_i$ and $i' = (i + 1) \bmod k$, where i is the current counter value and i' is the next counter value.

Note that the state and stack space of \mathcal{N} is $P := (2^Q \times 2^{Q \setminus F} \times 2^Q \times 2^{Q \setminus F} \times \{0, 1\} \times [k]) \cup \{p_I\}$. Consider a state $(R, S, R', S', b, i) \in P$. The R and R' component correspond to labels in the tree having the same head position h and $h + 1$, for some $h \in \mathbb{N}$, respectively. Hence, both sets are consistent. Since $S \subseteq R$ and $S' \subseteq R'$, we can restrict tuples in P to tuples whose components are consistent.

Finally, we show that the restricted state space of \mathcal{N} is in $2^{\mathcal{O}(nk \log n)}$, where $n := |Q|$. Note that a state of the eventually $\{1, 2\}$ -way AGA \mathcal{B} has the form $(q, i_0, \dots, i_{k-1}) \in Q \times [2n]^k$. That is, it consists of a state of the eventually $\{1, 2\}$ -way ASA \mathcal{A} and a rank. So, we can represent a tuple (R, S, R', S') of a state $(R, S, R', S', b, i) \in P$ of the NWA \mathcal{N} by four functions from Q to $[2n]^k$. Therefore, we have only $\mathcal{O}(k((2n)^k)^n) = 2^{\mathcal{O}(nk \log n)}$ different states in P . \square