On Regular Temporal Logics with Past*

Christian Dax[1], Felix Klaedtke[1], and Martin Lange[2]

[1] ETH Zurich, Switzerland
[2] Ludwig-Maximilians-University Munich, Germany

Abstract. The IEEE standardized *Property Specification Language*, PSL for short, extends the well-known linear-time temporal logic LTL with so-called semi-extended regular expressions. PSL and the closely related *SystemVerilog Assertions*, SVA for short, are increasingly used in many phases of the hardware design cycle, from specification to verification. In this paper, we extend the common core of these specification languages with past operators. We name this extension RTL. Although all ω-regular properties are expressible in PSL, SVA, and RTL, past operators often allow one to specify properties more naturally and concisely. In fact, we show that RTL is exponentially more succinct than the cores of PSL and SVA. Furthermore, we present a translation of RTL into language-equivalent nondeterministic Büchi automata, which is based on novel constructions for 2-way alternating automata. Our translation has almost the same worst-case complexity in terms of the size of the resulting nondeterministic Büchi automata as the existing translations for PSL and SVA. Consequently, the satisfiability and the model-checking problem for RTL fall into the same complexity classes as the corresponding problems for PSL and SVA. From the translation it also follows that the blowup of translating RTL formulas into initially equivalent PSL/SVA formulas is at most triply exponential.

1 Introduction

The industry standardized temporal logics PSL [1] and SVA (the assertion language of SystemVerilog [2]) are increasingly used in the hardware industry to formally express, validate, and verify the requirements of circuit designs. The linear-time core of PSL extends the well-known linear-time temporal logic LTL with semi-extended regular expressions (SEREs), which are essentially regular expressions with an additional operator for expressing the intersection of languages. The core of SVA can be seen as a subset of PSL.[3] The prominence of PSL and SVA in industry over other specification languages like LTL [25], μLTL [4], and ETL [31] is that PSL and SVA balance well the competing needs

* Partly supported by the Swiss National Science Foundation (SNF).

[3] For the ease of exposition, we identify, similar to [5, 7, 9, 26], PSL and SVA with their respective cores. In particular, the cores are "unclocked," they do not contain local variables (which are not part of the PSL standard), and their semantics is only defined over infinite words.

of a specification language like *expressiveness*, *usability*, and *implementability* [3]: all ω-regular languages are expressible in PSL/SVA, specifications in PSL/SVA are fairly easy to read and write, and relevant verification problems (e.g. model checking) for PSL/SVA are automatically solvable in practice.

Although temporal operators that refer to the past have been found natural and useful when expressing temporal properties [9, 10, 18, 22, 23], the PSL and SVA standards support temporal past operators only in a restrictive way. This design choice has already been made for the predecessor ForSpec [3] of PSL/SVA and has been justified by the argument that handling "arbitrary mixing of past and future operators results in nonnegligible implementation cost" [3]. One reason for this belief is that in the automata-theoretic approach to model checking [30], one uses 2-way automata to deal with past and future operators rather than 1-way automata when only future operators are present. The nowadays used automata constructions for 2-way automata are more involved than the corresponding ones for 1-way automata. For instance, with the state-of-the-art construction in [18], we can translate a 2-way alternating Büchi automaton with n states into a language-equivalent nondeterministic Büchi automaton (NBA) with $2^{\mathcal{O}(n^2)}$ states. For a given 1-way alternating Büchi automaton, we obtain with the Miyano-Hayashi construction [24] an NBA with only $2^{\mathcal{O}(n)}$ states. Nevertheless, in this paper, we give arguments in favor of extending PSL and SVA with past operators and we argue against this assumed additional implementation cost. In particular, one of our results shows that a restricted class of 2-way automata suffices and the additional cost for this class is small.

In more detail, the content of the paper is as follows. We first propose an extension of PSL with past operators, which we name *Regular Temporal Logic*, RTL for short. RTL extends PSL by the standard past operators from linear-time temporal logic and by the corresponding past operators of the PSL/SVA-specific operators for SEREs. For example, the PSL/SVA-specific operator $\alpha \diamond\!\!\rightarrow \varphi$ describes that a system trace fulfills from the current time point the pattern given by the SERE α and at the end the *post-condition* φ holds, where φ is a PSL/SVA formula. RTL additionally contains the corresponding counterpart $\alpha \mathbin{\diamond\!\!\!\!\leftarrow} \varphi$. This describes that the *pre-condition* φ holds at some time point in the past and at that time point the system trace fulfills up to the current time point the pattern α. Note that the temporal operator $\alpha \diamond\!\!\rightarrow \varphi$ is closely related to the modality $\langle \alpha \rangle \varphi$ in dynamic logic [16]. However, PSL/SVA uses SEREs over state predicates and in dynamic logic, the expressions are over program statements.

PSL, SVA, and RTL have the same expressive power: they all describe the class of ω-regular languages. However, RTL allows one to describe ω-regular languages more concisely than PSL and SVA. To show this, we establish a lower bound on the succinctness of RTL and SVA. We define a family of ω-regular languages and prove that these languages can be described in RTL exponentially more succinctly than in SVA. For the LTL-expressible properties, i.e, the ω-regular languages that are star-free (see, e.g., [14]), we obtain as a byproduct that RTL is double exponentially more succinct than LTL, even when extended with the classical temporal past operators Y (yesterday) and S (since).

Furthermore, we investigate the additional computational cost for solving the satisfiability problem and the model-checking problem for RTL. As for PSL and SVA, these problems are EXPSPACE-complete for RTL. In practice, the satisfiability problem and the model-checking problem for PSL and SVA are solved by using an automata-theoretic approach [5, 7, 9], translating a given formula into an NBA. With the standard automata constructions for PSL and SVA, one obtains for a PSL/SVA formula of size n an NBA of size $\mathcal{O}(2^{2 \cdot 2^{2n}})$ [5,7]. We present a novel construction for RTL that translates an RTL formula of size n into an NBA of size $\mathcal{O}(2^{3 \cdot 2^{2n}})$. Note that the upper bounds of the sizes of the resulting automata for PSL/SVA and RTL only differ by a small constant in the exponent despite the richer structure of RTL. Our translation is based on alternation-elimination constructions for restricted classes of 2-way alternating automata that were recently presented in [12] and which we further improve in this paper for the alternating automata that we obtain from our translation of RTL formulas into alternating automata. We use this construction to translate a given RTL formula into an initially equivalent SVA formula. The size of the resulting formula is triple exponentially larger, not quite matching the lower bound mentioned above. One of these three exponentials is due to the fact that the resulting SVA formulas do not contain SEREs anymore, but only regular expressions.

We point out that our translation for RTL into NBAs significantly improves over translations that we obtain when utilizing automata constructions that do not take the given special class of alternating automata into account. For instance, when using the state-of-the-art construction [18] for translating 2-way alternating automata into NBAs, one obtains an NBA of size $\mathcal{O}(2^{4 \cdot 2^{4n} + 2^{2n}})$, where n is again the size of the given RTL formula. Overall, the presented translation indicates that extensions of temporal logics with past operators can be handled with only a minor overhead in the automata-theoretic model-checking approach when adequate constructions for 2-way alternating automata are used.

The remainder of the paper is organized as follows. In Section 2, we give preliminaries. In Section 3, we define RTL and its fragments PSL and SVA. In Section 4, we present the translation of RTL formulas into language-equivalent NBAs and draw some consequences from this translation. In Section 5, we show the succinctness gap between RTL and PSL/SVA. Finally, in Section 6, we draw conclusions. The appendix contains additional proof details.

2 Preliminaries

Words and Trees. We denote the set of finite words over the alphabet Σ by Σ^* and the set of infinite words over Σ by Σ^ω. The length of a word $w \in \Sigma^*$ is written as $|w|$ and ε denotes the empty word. For a finite or infinite word w, w_i denotes the symbol of w at position $i \in \mathbb{N}$, where we assume that $i < |w|$ if w is finite. We write $v \preceq w$ if v is a prefix of the word w. For $i, j < |w|$, we write $w_{i..}$ for the suffix $w_i w_{i+1} \dots$ and $w_{i..j}$ for the subword $w_i w_{i+1} \dots w_j$.

A (Σ-labeled) *tree* is a function $t : T \to \Sigma$, where $T \subseteq \mathbb{N}^*$ satisfies the conditions: (i) T is prefix-closed (i.e., if $v \in T$ and $u \preceq v$ then $u \in T$) 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 $v \in T$. An (infinite) *path* in t is a word $\pi \in \mathbb{N}^\omega$ such that $v \in T$, for every prefix v of π. We write $t(\pi)$ for the word $t(\pi_0)t(\pi_1)\ldots \in \Sigma^\omega$.

Propositional Logic. We denote the set of *Boolean formulas* over the set P of propositions by $\mathcal{B}(P)$, i.e., $\mathcal{B}(P)$ consists of the formulas that are inductively built from the propositions in P and the connectives $\vee$, $\wedge$, and $\neg$. For $M \subseteq P$ and $b \in \mathcal{B}(P)$, we write $M \models b$ iff b evaluates to true when assigning true to the propositions in M and false to the propositions in $P \setminus M$. We write $\mathcal{B}^+(P)$ for the set of *positive Boolean formulas* over P, i.e., the ser if Boolean formulas in which the connective $\neg$ does not occur.

Regular Expressions. The syntax of *semi-extended regular expressions* (SEREs) over the proposition set P is defined by the grammar $\alpha ::= \varepsilon \mid b \mid \alpha \star \alpha \mid \alpha^*$, where $b \in \mathcal{B}(P)$ and $\star \in \{\cup, \cap, ;, :\}$. The language of an SERE over the proposition set P is inductively defined: (i) $L(\varepsilon) := \{\varepsilon\}$, (ii) $L(b) := \{w \in (2^P)^* \mid |w| = 1 \text{ and } w \models \alpha\}$, for $b \in \mathcal{B}(P)$, (iii) $L(\beta \star \gamma) := L(\beta) \star L(\gamma)$, for $\star \in \{\cup, \cap, ;, :\}$, where $L\,;\,L' := \{uv \mid u \in L \text{ and } v \in L'\}$ is the concatenation of L and L', and $L : L' := \{ubv \mid ub \in L \text{ and } bv \in L' \text{ with } b \in 2^P\}$ the fusion, and (iv) $L(\beta^*) := \bigcup_{n\in\mathbb{N}} L^n(\beta)$, where $L^0 := \{\varepsilon\}$ and $L^{i+1} := L\,;\,L^i$, for all $i \in \mathbb{N}$. The *size* of an SERE is its syntactic length, i.e., $\|\varepsilon\| := 1$, $\|b\| := 1$, for $b \in \mathcal{B}(P)$, $\|\beta \star \gamma\| := 1 + \|\beta\| + \|\gamma\|$, for $\star \in \{\cup, \cap, ;, :\}$, and $\|\beta^*\| := 1 + \|\beta\|$.

Automata. In the following, we define 2-way alternating automata, which scan input words letter by letter with their read-only head. Let $\mathbb{D} := \{-1, 0, 1\}$ be the set of directions in which the read-only head can move. A *2-way alternating Büchi automaton* (2ABA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_I, F)$, where Q is a finite set of states, Σ is a finite nonempty alphabet, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q \times \mathbb{D})$ is the transition function, $q_I \in Q$ is the initial state, and $F \subseteq Q$ is the acceptance condition. The *size* $\|\mathcal{A}\|$ of the automaton $\mathcal{A}$ is $|Q|$.

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

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

For an infinite sequence of configurations $\pi := (q_0, i_0)(q_1, i_1)\ldots \in (Q \times \mathbb{N})^\omega$, we define $\mathit{Inf}(\pi) := \{q \mid q \text{ occurs infinitely often in } q_0 q_1 \ldots \in Q^\omega\}$. A path $\pi \in T$ in a run r is *accepting* if $\mathit{Inf}(r(\pi)) \cap F \neq \emptyset$. The run r is *accepting* if every path in r is accepting. The *language* of $\mathcal{A}$ is the set $L(\mathcal{A}) := \{w \in \Sigma^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$.

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

is 1-way, we assume that δ is of the form $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$. We call a 1-way automaton a *nondeterministic Büchi automaton* (NBA) if its transition function returns a disjunction of states for all inputs. Similarly, we call a 1-way automaton a *universal Büchi automaton* (UBA) if its transition function returns a conjunction of states for all inputs. We view the transition function δ of an NBA or UBA as a function of the form $\delta : Q \times \Sigma \to 2^Q$. This means that clauses and monomials are written as sets. Note that a run $r : T \to Q \times \mathbb{N}$ of an NBA $\mathcal{A}$ on $w \in \Sigma^\omega$ can be reduced to a single path π in r that is consistent with the transition function. Using standard terminology, we also call $r(\pi) \in (Q \times \mathbb{N})^\omega$ a run of $\mathcal{A}$ on w.

A *nondeterministic finite automaton* (NFA) $\mathcal{B}$ is a quintuple that has the same components as an NBA. The *size* of an NFA is defined as for NBAs. For a finite word $w \in \Sigma^*$, a *run* of the NFA $\mathcal{B} = (Q, \Sigma, \delta, q_I, F)$ on w is a sequence of $|w| + 1$ states $q_0 q_1 \dots q_{|w|}$ such that $q_0 = q_I$ and $\delta(q_i, w_i) \ni q_{i+1}$, for all $i < |w|$. The run is *accepting* if $q_{|w|} \in F$. The *language* of $\mathcal{B}$ is the set $L(\mathcal{B}) := \{w \in \Sigma^* \mid \text{there is an accepting run of } \mathcal{B} \text{ on } w\}$.

3 Temporal Logics with Expressions and Past Operators

In this section, we extend LTL with SEREs and past operators. We call the extension *Regular Temporal Logic*, RTL for short. The cores of the two industrial-standard property-specification languages PSL [1] and SVA [2] are fragments of RTL. The syntax of RTL over the set P of propositions is given by the grammar

$$\varphi ::= p \mid \mathsf{cl}(\alpha) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \alpha \diamond\!\!\rightarrow \varphi \mid \mathsf{Y}\varphi \mid \varphi \,\mathsf{S}\, \varphi \mid \alpha \mathbin{\Diamond\!\!\!\!-\!\!\!\rightarrow} \varphi\,,$$

where $p \in P$ and α is an SERE over P. An RTL formula over P is interpreted at a position $i \in \mathbb{N}$ of an infinite word $w \in (2^P)^\omega$ as follows:

$w, i \models p$	iff	$p \in w_i$
$w, i \models \mathsf{cl}(\alpha)$	iff	$\exists k \geq i : w_{i..k} \in L(\alpha),$ or $\forall k \geq i : \exists v \in L(\alpha) : w_{i..k} \preceq v$
$w, i \models \varphi \wedge \psi$	iff	$w, i \models \varphi$ and $w, i \models \psi$
$w, i \models \neg\varphi$	iff	$w, i \not\models \varphi$
$w, i \models \mathsf{X}\varphi$	iff	$w, i+1 \models \varphi$
$w, i \models \varphi \,\mathsf{U}\, \psi$	iff	$\exists k \geq i : w, k \models \psi$ and $\forall j :$ if $i \leq j < k$ then $w, j \models \varphi$
$w, i \models \alpha \diamond\!\!\rightarrow \varphi$	iff	$\exists k \geq i : w_{i..k} \in L(\alpha)$ and $w, k \models \varphi$
$w, i \models \mathsf{Y}\varphi$	iff	$i > 0$ and $w, i-1 \models \varphi$
$w, i \models \varphi \,\mathsf{S}\, \psi$	iff	$\exists k \leq i : w, k \models \psi$ and $\forall j :$ if $k < j \leq i$ then $w, j \models \varphi$
$w, i \models \alpha \mathbin{\Diamond\!\!\!\!-\!\!\!\rightarrow} \varphi$	iff	$\exists k \leq i : w_{k..i} \in L(\alpha)$ and $w, k \models \varphi$

A word $w \in (2^P)^\omega$ is a *model* of an RTL formula φ if $w, 0 \models \varphi$. The *language* of an RTL formula φ is $L(\varphi) := \{w \in (2^P)^\omega \mid w, 0 \models \varphi\}$. The RTL formulas φ and ψ are *initially equivalent* if $L(\varphi) = L(\psi)$. They are *logically equivalent*, written as $\varphi \equiv \psi$, if $w, i \models \varphi \Leftrightarrow w, i \models \psi$, for all $i \in \mathbb{N}$ and $w \in (2^P)^\omega$. As for SEREs, we define the *size* $\|\varphi\|$ of an RTL formula φ as its syntactic length. That means, $\|p\| := 1$, $\|\mathsf{cl}(\alpha)\| := 1 + \|\alpha\|$, $\|\neg\varphi\| := \|\mathsf{X}\varphi\| := \|\mathsf{Y}\varphi\| := 1 + \|\varphi\|$, $\|\varphi \wedge \psi\| :=$

$\|\varphi \,\mathsf{U}\, \psi\| := \|\varphi \,\mathsf{S}\, \psi\| := 1 + \|\varphi\| + \|\psi\|$, and $\|\alpha \diamond\!\!\rightarrow \varphi\| := \|\alpha \mathbin{\diamond\!\!\!\!-} \varphi\| := 1 + \|\alpha\| + \|\varphi\|$. Moreover, $\mathit{Is}(\varphi)$ is the number of intersections that occur in the SEREs in the RTL formula φ and $\mathit{Sub}(\varphi)$ is the set of subformulas that occur in φ.

We define the following fragments of RTL. We call an RTL formula a *PSL formula* if it does not contain the operators Y, S, and $\mathbin{\diamond\!\!\!\!-}$. An *LTL formula* is a PSL formula that does not contain the operators cl and $\diamond\!\!\rightarrow$. An *SVA formula* is a PSL formula that does not contain the operators cl, X, and U. The fragments PLTL and PSVA, which extend LTL and SVA, respectively, with past operators, are defined as expected. Note that RTL and PSL extended with the past operators Y, S, and $\mathbin{\diamond\!\!\!\!-}$ coincide.

We use standard syntactic sugar, like the Boolean constants and connectives ff, tt, $\vee$, $\rightarrow$, and we define $\varphi \,\mathsf{R}\, \psi := \neg(\neg\varphi \,\mathsf{U}\, \neg\psi)$, $\varphi \,\mathsf{T}\, \psi := \neg(\neg\varphi \,\mathsf{S}\, \neg\psi)$, $\mathsf{Z}\varphi := \mathsf{Y}\mathsf{tt} \rightarrow \mathsf{Y}\varphi$. Moreover, for an RTL formula φ and an SERE α, we write $\alpha \,\Box\!\!\rightarrow \varphi$ for $\neg(\alpha \diamond\!\!\rightarrow \neg\varphi)$ and $\alpha \boxminus \varphi$ for $\neg(\alpha \mathbin{\diamond\!\!\!\!-} \neg\varphi)$. Note that the standard unary temporal operators can easily be defined in the respective fragment. For instance, for PSVA we define $\mathsf{G}\varphi := \mathsf{tt}^* \,\Box\!\!\rightarrow \varphi$, $\mathsf{F}\varphi := \mathsf{tt}^* \diamond\!\!\rightarrow \varphi$, $\mathsf{H}\varphi := \mathsf{tt}^* \boxminus \varphi$, and $\mathsf{O}\varphi := \mathsf{tt}^* \mathbin{\diamond\!\!\!\!-} \varphi$.

Remark 1. In the PSL standard [1], we also have atomic formulas of the form $\mathsf{ended}(\alpha)$ and $\mathsf{prev}(\alpha)$, where α is an SERE. For instance, the word w satisfies $\mathsf{ended}(\alpha)$ at position i iff there is a subword u of w that ends at i and $u \in L(\alpha)$. The operators ended and prev can be seen as restricted variants of the past operator $\mathbin{\diamond\!\!\!\!-}$. For instance, in RTL, if $\varepsilon \notin L(\alpha)$, $\mathsf{ended}(\alpha)$ is syntactic sugar for $\alpha \mathbin{\diamond\!\!\!\!-} \mathsf{tt}$, and tt otherwise. Observe that ended and prev can only be applied to SEREs, and in contrast to $\mathbin{\diamond\!\!\!\!-}$, it is not possible to define the classical past operators Y, H, and O with them. We also remark that the literature, e.g. [5, 7, 9, 19, 26] usually considers the essential core of the PSL standard to which the operators ended and prev do not belong. We follow this convention, i.e., the formulas in our fragment PSL of RTL do not contain $\mathsf{ended}(\alpha)$ and $\mathsf{prev}(\alpha)$. Finally, we remark that the automata constructions [5, 7] for PSL and SVA cannot cope with the operators ended and prev, which are handled by our construction in Section 4 for RTL.

Example 2. A standard example for showing that the past operators of PLTL can lead to more intuitive specifications is $\mathsf{G}(\mathit{grant} \rightarrow \mathsf{O}\mathit{request})$, i.e., every grant is preceded by a request [22]. An initially equivalent LTL formula is $\mathit{request} \,\mathsf{R}\, (\neg\mathit{grant} \vee \mathit{request})$. Let us now illustrate the beneficial use of SEREs and past operators. Suppose that a request is not a single event but a sequence of events, e.g., a request consists of a *start* event followed eventually by an *end* event and no *cancel* event happens between the *start* and the *end* event. Such sequences are naturally described by the SERE $(\mathit{start} \,;\, \mathsf{tt}^* \,;\, \mathit{end}) \cap (\neg\mathit{cancel})^*$. Using this SERE and the new past operator $\mathbin{\diamond\!\!\!\!-}$, we can easily express in RTL the property that every grant is preceded by a request:

$$\mathsf{G}\big(\mathit{grant} \rightarrow \big(((\mathit{start} \,;\, \mathsf{tt}^* \,;\, \mathit{end}) \cap (\neg\mathit{cancel})^*) \,;\, \mathsf{tt}^* \mathbin{\diamond\!\!\!\!-} \mathsf{tt}\big)\big)\,. \tag{1}$$

Note that according to the semantics of the operator $\mathbin{\diamond\!\!\!\!-}$, the *end* event has to happen before or at the same time as the *grant* event. Alternatively, we can

express the property in PLTL as

$$\mathsf{G}\big(\mathit{grant} \rightarrow \mathsf{O}\big(\mathit{end} \wedge \neg \mathit{cancel} \wedge \mathsf{Y}(\neg \mathit{cancel}\, \mathsf{S}\, (\mathit{start} \wedge \neg \mathit{cancel}))\big)\big)\,. \tag{2}$$

Although debatable, we consider that the RTL formula (1) is easier to understand than the PLTL formula (2). In SVA, we can express the property as $\mathit{norequest} \,\square\!\!\rightarrow\, \neg \mathit{grant}$, where the SERE *norequest* describes the complement of the language $L\big(\mathsf{tt}^*\,;\,((\mathit{start}\,;\,\mathsf{tt}^*\,;\,\mathit{end}) \cap (\neg \mathit{cancel})^*)\,;\,\mathsf{tt}^*\big)$, that is, $\mathit{norequest} := (a \cup b\,;\,d^*\,;\,c)^*\,;\,(c^* \cup b\,;\,d)$, where a, b, c, and d are the Boolean formulas $\neg \mathit{start} \vee \mathit{cancel}$, $\mathit{start} \wedge \neg \mathit{cancel}$, cancel, and $\neg \mathit{cancel} \wedge \neg \mathit{end}$, respectively. Note that in general, complementation of SEREs is difficult and can result in an exponential blowup with respect to the size of the given SERE.

Example 3. Let us give another example to illustrate the usefulness of past operators, in particular, the operator $\leftrightarrow\!\!\!\!\diamond$. For $N \geq 1$ and $i \in \{0, \ldots, N-1\}$, consider the RTL formula $\Phi_{N,i} := \mathsf{G}\big(\mathit{send}_i \rightarrow \big(\mathit{switch}_i \cap (\mathit{init}\,;\,(\neg \mathit{init})^*) \leftrightarrow\!\!\!\!\diamond\, \mathsf{tt}\big)\big)$, where switch_i counts the number of *switch* events modulo N, i.e.,

$$\begin{aligned} \mathit{switch}_i := \Big(\underbrace{(\neg \mathit{switch})^*\,;\,\mathit{switch}\,;\ldots;\,(\neg \mathit{switch})^*\,;\,\mathit{switch}}_{N \text{ times}} \Big)^*\,; \\ \underbrace{(\neg \mathit{switch})^*\,;\,\mathit{switch}\,;\ldots;\,(\neg \mathit{switch})^*\,;\,\mathit{switch}}_{i \text{ times}}\,;(\neg \mathit{switch})^*\,. \end{aligned} \tag{3}$$

Intuitively, $\Phi_{N,i}$ expresses the property that the process i is only allowed to send a data item if it possesses the token. The process i possesses the token iff $i \equiv 0 \mod N$ *switch* events occurred previously since the last *init* event. Note that this property is not expressible in LTL since it is not star-free (see, e.g., [14]).

The negation of the PSL formula

$$\big((\neg \mathit{init})^* \diamond\!\!\rightarrow \mathit{send}_i\big) \vee \mathsf{F}\big(\mathit{init} \wedge \big((\mathsf{tt}\,;\,(\neg \mathit{init})^*) \cap (\textstyle\bigcup_{j \neq i} \mathit{switch}_j) \diamond\!\!\rightarrow \mathit{send}_i\big)\big) \tag{4}$$

is initially equivalent to $\Phi_{N,i}$. Note that the size of the formula (4) is quadratic in N, whereas the size of the formula (3) is only linear in N. In Section 5, we prove that PSVA is exponentially more succinct than PSL.

In general, for writing specifications, RTL possesses the advantage of PLTL over LTL and the advantage of PSL/SVA over LTL, namely, additional operators for referring to the past and SEREs for describing sequences of events.

4 From RTL to Nondeterministic Automata

In this section, we present a translation from RTL formulas into language-equivalent NBAs. Similar to the well-known translation for LTL formulas into NBAs, our translation comprises two steps: for a given RTL formula, we first construct an alternating automaton, which we then translate into an NBA. Throughout this section, we fix a finite set P of propositions.

4.1 From RTL to Loop-free and Locally 1-Way 2ABAs

In this subsection, we assume that φ is an RTL formula over P and φ is in *negation normal form*, i.e., the negation symbol $\neg$ only occurs directly in front of the atomic subformulas of φ. Note that every RTL formula ψ can be rewritten into a logically equivalent RTL formula in negation normal form over an extended language, where we use the additional Boolean connective $\vee$ and the additional operators R, T, Z, $\Box\!\!\rightarrow$, and $\boxminus\!\!\rightarrow$ as primitives. The size of the resulting formula is at most $2\|\psi\|$. For rewriting a formula into negation normal form, we use the logical equivalences $\neg\neg\gamma \equiv \gamma$, $\neg\mathsf{X}\gamma \equiv \mathsf{X}\neg\gamma$, $\neg\mathsf{Y}\gamma \equiv \mathsf{Z}\neg\gamma$, and $\neg\mathsf{Z}\gamma \equiv \mathsf{Y}\neg\gamma$.

Before we present the construction of the 2ABA $\mathcal{A}_\varphi$ for the RTL formula φ, we briefly highlight the similarities and the differences to the standard constructions for LTL, PLTL, SVA, and PSL [5, 7, 15, 29]. The construction in [7] additionally handles SEREs with local variables. Our construction can easily be extended by this feature. However, for the ease of exposition, we focus here on how to handle the temporal past and future operators of RTL efficiently. As the standard construction for PSL [5], the state space of the 2ABA $\mathcal{A}_\varphi$ consists of the subformulas of the given RTL formula and the states of the automata for the SEREs. We introduce a special symbol # to mark the beginning of the input word. With this symbol, $\mathcal{A}_\varphi$ checks in a run whether the read-only head is at the first position of the input word. We need some auxiliary states for such a check. The new operators $\boxminus\!\!\rightarrow$ and $\diamondminus\!\!\rightarrow$ are then easily handled since $\mathcal{A}_\varphi$ is alternating and 2-way.

Construction Details For the construction, we need the following lemma about translating SEREs into automata. For proof details , see [5] and standard textbooks on automata theory like [17].

Lemma 4. *Let α be a SERE over the set P of propositions.*

1. *There is an NFA $\mathcal{A}_\alpha$ with $L(\mathcal{A}_\alpha) = L(\alpha)$ and $\|\mathcal{A}\| \leq 2^{\|\alpha\|}$.*
2. *There is an NFA $\mathcal{A}'_\alpha$ with $L(\mathcal{A}'_\alpha) = \{w_{n-1} \dots w_0 \mid w_0 \dots w_{n-1} \in L(\alpha)\}$ and $\|\mathcal{A}\| \leq 2^{\|\alpha\|}$.*
3. *There is an NBA $\mathcal{B}_\alpha$ with $L(\mathcal{B}_\alpha) = L(\mathsf{cl}(\alpha))$ and $\|\mathcal{B}_\alpha\| \leq 2^{\|\alpha\|}$.*
4. *There is a UBA $\mathcal{B}'_\alpha$ with $L(\mathcal{B}'_\alpha) = L(\neg\mathsf{cl}(\alpha))$ and $\|\mathcal{B}'_\alpha\| \leq 2^{\|\alpha\|}$.*

For the construction of the 2ABA $\mathcal{A}_\varphi$, let $\mathcal{A}_\alpha$, $\mathcal{A}'_\alpha$, $\mathcal{B}_\alpha$, and $\mathcal{B}'_\alpha$ be the corresponding automata according to Lemma 4, where α is an SERE that occurs in φ. We assume that the state sets of these automata are pairwise disjoint.

Now, the 2ABA $\mathcal{A}_\varphi := (Q, \Gamma, \delta, q_I, F)$ for the RTL formula φ is defined as follows, where Γ is the alphabet $\{\#\} \cup 2^P$. We use # as an auxiliary symbol to mark the beginning of the input word. So, the automaton $\mathcal{A}_\varphi$ is able to change its state depending on whether its read-only head is at the first position of the input word. As Lemma 5 below shows, $\mathcal{A}_\varphi$ accepts the language $\{\#w \mid w \in L(\varphi)\}$.

The state set Q is the disjoint union of the sets Q_1, Q_2, and Q_3. The states in $Q_1 := \{q_I, q_{acc}, q_{rej}, q_\#\}$ are the initial state q_I, the accepting and rejecting sink states q_{acc} and q_{rej}, and the state $q_\#$ for handling the auxiliary letter # at

the first position of an input word. The purpose of the states in $Q_2 := Sub(\varphi)$ is similar as in the standard constructions that translate LTL formulas into alternating automata. Roughly speaking, they take care of the models of the subformulas of φ. The remaining state set Q_3 is used to include the automata for the SEREs that occur in φ. It is defined as

$$\begin{aligned} Q_3 := \{&\mathsf{cl}(s) \mid \mathsf{cl}(\alpha) \in Sub(\varphi) \text{ and } s \text{ is a state of } \mathcal{B}_\alpha\} \cup \\ &\{\neg\mathsf{cl}(s) \mid \neg\mathsf{cl}(\alpha) \in Sub(\varphi) \text{ and } s \text{ is a state of } \mathcal{B}'_\alpha\} \cup \\ &\{s \star\!\!\rightarrow \psi \mid \star\!\!\rightarrow \in \{\Diamond\!\!\rightarrow, \Box\!\!\rightarrow\},\ \alpha \star\!\!\rightarrow \psi \in Sub(\varphi),\ \text{and } s \text{ is a state of } \mathcal{A}_\alpha\} \cup \\ &\{s \star\!\!\rightarrow \psi \mid \star\!\!\rightarrow \in \{\diamondsuit\!\!\rightarrow, \boxminus\!\!\rightarrow\},\ \alpha \star\!\!\rightarrow \psi \in Sub(\varphi),\ \text{and } s \text{ is a state of } \mathcal{A}'_\alpha\}\,. \end{aligned}$$

The set of accepting states F is the union of the sets F_1, F_2, and F_3. F_1 is the singleton $\{q_{acc}\}$. Similar to the standard construction from LTL to alternating automata, the set of states $F_2 := \{\gamma\,\mathsf{R}\,\psi \mid \gamma\,\mathsf{R}\,\psi \in Sub(\varphi)\}$ contains the greatest-fixpoint formulas that occur in φ. The states in

$$\begin{aligned} F_3 := \{&\mathsf{cl}(s) \mid \mathsf{cl}(\alpha) \in Sub(\varphi) \text{ and } s \text{ is an accepting state of } \mathcal{B}_\alpha\} \cup \\ &\{\neg\mathsf{cl}(s) \mid \neg\mathsf{cl}(\alpha) \in Sub(\varphi) \text{ and } s \text{ is an accepting state of } \mathcal{B}'_\alpha\} \cup \\ &\{s\,\Box\!\!\rightarrow \psi \mid \alpha\,\Box\!\!\rightarrow \psi \in Sub(\varphi) \text{ and } s \text{ is a state of } \mathcal{A}_\alpha\} \cup \\ &\{s\,\boxminus\!\!\rightarrow \psi \mid \alpha\,\boxminus\!\!\rightarrow \psi \in Sub(\varphi) \text{ and } s \text{ is a state of } \mathcal{A}'_\alpha\} \end{aligned}$$

are the accepting states for the subformulas with an SERE.

It remains to define the transition function δ. We start with the transitions of the states in Q_1. Let $b \in \Gamma$. For the states q_{rej} and q_{acc}, we define

$$\delta(q_{rej}, b) := (q_{rej}, 1) \qquad \text{and} \qquad \delta(q_{acc}, b) := \begin{cases} (q_{rej}, 1) & \text{if } b = \#, \\ (q_{acc}, 1) & \text{otherwise.} \end{cases}$$

For the state $q_\#$, we define

$$\delta(q_\#, b) := \begin{cases} (q_{acc}, 1) & \text{if } b = \#, \\ (q_{rej}, 1) & \text{otherwise.} \end{cases}$$

The transitions of the initial state q_I are $\delta(q_I, b) := (q_\#, 0) \wedge (\varphi, 1)$.

For a state $q \in Q_2 \cup Q_3$, $\mathcal{A}_\varphi$ rejects when reading the letter $\#$, i.e., we define $\delta(q, \#) := (q_{rej}, 1)$. For the remainder of the construction, let $a \in 2^P$.

The following definitions are similar as in the standard construction for translating LTL into alternating automata.

- For a proposition $p \in P$, we define

$$\delta(p, a) := \begin{cases} (q_{acc}, 1) & \text{if } p \in a, \\ (q_{rej}, 1) & \text{otherwise} \end{cases} \quad \text{and} \quad \delta(\neg p, a) := \begin{cases} (q_{acc}, 1) & \text{if } p \notin a, \\ (q_{rej}, 1) & \text{otherwise.} \end{cases}$$

- For the Boolean connectives $\wedge$ and $\vee$, we define

$$\delta(\gamma \wedge \psi, a) := (\gamma, 0) \wedge (\psi, 0) \qquad \text{and} \qquad \delta(\gamma \vee \psi, a) := (\gamma, 0) \vee (\psi, 0)\,.$$

– For the unary temporal operators X, Y, and Z, we define

$$\delta(\mathsf{X}\psi, a) := (\psi, 1)\,,\ \delta(\mathsf{Y}\psi, a) := (\psi, -1)\,,\ \text{and}\ \delta(\mathsf{Z}\psi, a) := (\psi, -1) \vee (q_{\#}, -1)\,.$$

Note that for the state $\mathsf{Z}\psi$, the automaton $\mathcal{A}_\varphi$ guesses whether its read-only head is at the first position by moving to state $q_{\#}$. In that case, it does not need to go to the state ψ but it has to accept the word from $q_{\#}$ and hence, the position of its read-only head must be at the beginning of the word.

– For the binary temporal operators U, R, S, and T, we define

$$\begin{aligned}
&\delta(\gamma\ \mathsf{U}\ \psi, a) := (\psi, 0) \vee \big((\gamma, 0) \wedge (\gamma\ \mathsf{U}\ \psi, 1)\big)\,,\\
&\delta(\gamma\ \mathsf{R}\ \psi, a) := (\psi, 0) \wedge \big((\gamma, 0) \vee (\gamma\ \mathsf{R}\ \psi, 1)\big)\,,\\
&\delta(\gamma\ \mathsf{S}\ \psi, a) := (\psi, 0) \vee \big((\gamma, 0) \wedge (\gamma\ \mathsf{S}\ \psi, -1)\big)\,, \qquad \text{and}\\
&\delta(\gamma\ \mathsf{T}\ \psi, a) := (\psi, 0) \wedge \big((\gamma, 0) \vee (\gamma\ \mathsf{T}\ \psi, -1) \vee (q_{\#}, -1)\big)\,.
\end{aligned}$$

Let us now turn to the transitions for the subformulas with an SERE. We follow the construction given in [5] for PSL.

– For a state $\mathsf{cl}(\alpha) \in \mathit{Sub}(\varphi)$, the automaton $\mathcal{A}_\varphi$ moves to the initial state of the NBA $\mathcal{B}_\alpha = (S, 2^P, \eta, s_I, E)$ without moving its read-only head. Then, it simulates a run of $\mathcal{B}_\alpha$ on the input word. Formally, for $s \in S$, we define

$$\delta(\mathsf{cl}(\alpha), a) := (\mathsf{cl}(s_I), 0) \quad \text{and} \quad \delta(\mathsf{cl}(s), a) := \bigvee_{t \in \eta(s,a)} (\mathsf{cl}(t), 1)\,.$$

Similarly, for a state $\neg\mathsf{cl}(\alpha) \in \mathit{Sub}(\varphi)$, $\mathcal{A}_\varphi$ simulates the UBA $\mathcal{B}'_\alpha$:

$$\delta(\neg\mathsf{cl}(\alpha), a) := (\neg\mathsf{cl}(s_I), 0) \quad \text{and} \quad \delta(\neg\mathsf{cl}(s), a) := \bigwedge_{t \in \eta(q,a)} (\neg\mathsf{cl}(t), 1)\,,$$

where $\mathcal{B}'_\alpha = (S, 2^P, \eta, s_I, E)$ and $s \in S$.

– The state $\alpha \diamond\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$ is used to start a simulation of the NFA $\mathcal{A}_\alpha = (S, 2^P, \eta, s_I, E)$ on the input word. If the simulation reaches a final state of the NFA, $\mathcal{A}_\varphi$ may terminate the simulation and proceed with the state ψ. Formally, we define $\delta(\alpha \diamond\!\!\rightarrow \psi, a) := (s_I \diamond\!\!\rightarrow \psi, 0)$ and for $s \in S$,

$$\delta(s \diamond\!\!\rightarrow \psi, a) := \begin{cases} \bigvee_{t \in \eta(s,a)} (t \diamond\!\!\rightarrow \psi, 1) \vee (\psi, 0) & \text{if } \eta(s,a) \cap E \neq \emptyset,\\ \bigvee_{t \in \eta(s,a)} (t \diamond\!\!\rightarrow \psi, 1) & \text{otherwise.} \end{cases}$$

The transitions for a subformula $\alpha \mathrel{\diamond\!\!\!\!\!-\!\!\!\rightarrow} \psi \in \mathit{Sub}(\varphi)$ are defined similarly. Instead of simulating the NFA $\mathcal{A}_\alpha$, $\mathcal{A}_\varphi$ simulates the NFA $\mathcal{A}'_\alpha$, where it moves the read-only head to the left instead of to the right.

– If the state is $\alpha \mathrel{\Box\!\!\rightarrow} \psi \in \mathit{Sub}(\varphi)$, the automaton $\mathcal{A}_\varphi$ simulates a run of the NFA $\mathcal{A}_\alpha = (S, 2^P, \eta, s_I, E)$ seen as a universal automaton. If the simulation reaches a final state, $\mathcal{A}_\varphi$ has to proceed with the state ψ. Formally, we define $\delta(\alpha \mathrel{\Box\!\!\rightarrow} \psi, a) := (s_I \mathrel{\Box\!\!\rightarrow} \psi, 0)$ and for $s \in S$,

$$\delta(s \mathrel{\Box\!\!\rightarrow} \psi, a) := \begin{cases} \bigwedge_{t \in \eta(s,a)} (t \mathrel{\Box\!\!\rightarrow} \psi, 1) \wedge (\psi, 0) & \text{if } \eta(s,a) \cap E \neq \emptyset,\\ \bigwedge_{t \in \eta(s,a)} (t \mathrel{\Box\!\!\rightarrow} \psi, 1) & \text{otherwise.} \end{cases}$$

The transitions for a subformula $\alpha \boxminus\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$ are similarly defined. However, if the read-only head is at the beginning of the input word, $\mathcal{A}_\varphi$ can stop the simulation. Formally, for the NFA $\mathcal{A}'_\alpha = (S, 2^P, \eta, s_I, E)$ and $s \in S$, we define $\delta(\alpha \boxminus\!\!\rightarrow \psi, a) := (s_I \boxminus\!\!\rightarrow \psi, 0)$ and

$$\delta(s \boxminus\!\!\rightarrow \psi, a) := \begin{cases} (q_\#, -1) \vee \bigwedge_{t \in \eta(s,a)} (t \boxminus\!\!\rightarrow \psi, -1) \wedge (\psi, 0) & \\ & \text{if } \eta(s,a) \cap E \neq \emptyset, \\ (q_\#, -1) \vee \bigwedge_{t \in \eta(s,a)} (t \boxminus\!\!\rightarrow \psi, -1) & \text{otherwise.} \end{cases}$$

We remark that the ε-transitions in our construction (i.e., the transitions of $\mathcal{A}_\varphi$ in which the read-only head does not move) can be easily eliminated by replacing a proposition $(s, 0)$ that occurs in $\delta(q, b)$ by $\delta(s, b)$, where $q, s \in Q$ and $b \in \Gamma$.

The following lemma about the accepted language of the constructed automaton $\mathcal{A}_\varphi$ is not difficult to prove. The proof details are given in Appendix A.

Lemma 5. *The 2ABA $\mathcal{A}_\varphi$ accepts the language $\{\#w \mid w \in L(\varphi)\}$.*

From the definition of the state set Q and Lemma 4, we directly obtain Lemma 6.

Lemma 6. *The 2ABA $\mathcal{A}_\varphi$ has size at most $4 + 2^{\|\varphi\|}$.*

Additional Properties of the Construction The 2ABA $\mathcal{A}_\varphi$ has some additional properties, which we exploit in Section 4.2 for constructing the NBA. Namely, $\mathcal{A}_\varphi$ is loop-free [12, 15] and locally 1-way.

Intuitively speaking, loop-freeness means that an automaton cannot visit a configuration twice on the same computation branch. Formally, it is defined as follows for a 2ABA $\mathcal{B} = (S, \Sigma, \eta, s_I, E)$. Let $\Pi(\mathcal{B})$ be the set of words of the form $(s_0, j_0)(s_1, j_1) \ldots \in (S \times \mathbb{N})^\omega$ such that $(s_0, j_0) = (s_I, 0)$ and for all $i \in \mathbb{N}$, there is some $a \in \Sigma$ and a set $M \subseteq S \times \mathbb{Z}$ with $(s_{i+1}, j_{i+1} - j_i) \in M$ and M is a minimal model of $\eta(s_i, a)$, i.e, $M \vDash \eta(s_i, a)$ and $M \setminus \{c\} \nvDash \eta(s_i, a)$, for all $c \in M$. The automaton $\mathcal{B}$ is *loop-free* if for all words $\pi \in \Pi(\mathcal{B})$, there are no integers $i, j \in \mathbb{N}$ with $i \neq j$ such that $\pi_i = \pi_j$. Recall that π_i and π_j are configurations, which consist of the current state and the current position of the read-only head.

Lemma 7. *The 2ABA $\mathcal{A}_\varphi$ is loop-free.*

Proof. We start by defining the following function that assigns weights to states:

$$\mathit{weight}(q) := \begin{cases} 2|\mathit{Sub}(\varphi)| + 1 & \text{if } q = q_I, \\ 2|\mathit{Sub}(q)| & \text{if } q \in Q_2, \\ 2|\mathit{Sub}(\psi)| + 1 & \text{if } q \in Q_3 \text{ and } q \text{ is of the form } s \star\!\!\rightarrow \psi \\ & \qquad \text{with } \star\!\!\rightarrow \in \{\Diamond\!\!\rightarrow, \diamondminus\!\!\rightarrow, \Box\!\!\rightarrow, \boxminus\!\!\rightarrow\}, \\ 0 & \text{otherwise.} \end{cases}$$

Note that $\mathit{weight}(\alpha \star\!\!\rightarrow \psi) > \mathit{weight}(s \star\!\!\rightarrow \psi) > \mathit{weight}(\psi)$. Furthermore, observe that by the definition of the transition function, we have that $\mathit{weight}(q') \leq$

weight(q), for all $q, q' \in Q$ whenever the proposition (q', d) occurs in $\delta(q, a)$, where $a \in \Gamma$ and $d \in \mathbb{D}$.

We need to show that for all paths $\pi \in \Pi(\mathcal{A}_\varphi)$, there are no positions $i, j \in \mathbb{N}$ with $i < j$ and $\pi_i = \pi_j$. Let $\pi = (q_0, h_0)(q_1, h_1)\dots$ be a path in $\Pi(\mathcal{A}_\varphi)$ and $i, j \in \mathbb{N}$ with $i < j$. Suppose that $q_i = q_j$. We consider the following cases.

Case $q_i \in Q_1$. Assume that $q_i = q_I$. Since there is no transition to q_I, the state q_j cannot be equal to q_i. Assume that $q_i \in \{q_\#, q_{acc}, q_{rej}\}$. By the definition of the transition function, $q_k \in \{q_{acc}, q_{rej}\}$, for all $i \leq k \leq j$, and the head position increases by 1 whenever $\mathcal{A}_\varphi$ moves from a configuration π_k to π_{k+1}, for $i \leq k < j$. Therefore, h_j is greater than h_i.

Case $q_i \in Q_2$. Let us first assume that q_i is a state of the form $\psi \,\mathsf{U}\, \psi'$, $\psi \,\mathsf{R}\, \psi'$, $\psi \,\mathsf{S}\, \psi'$, or $\psi \,\mathsf{T}\, \psi'$ with $\psi, \psi' \in \mathit{Sub}(\varphi)$. We distinguish two subcases. In the first subcase, $\mathcal{A}_\varphi$ stays in the state q_i, i.e., $q_i = q_{i+1} = \cdots = q_j$. By the definition of the transition function, $\mathcal{A}_\varphi$ moves the head position in one direction and thus, h_j is different from h_i. In the second subcase, $\mathcal{A}_\varphi$ eventually moves to a subformula q_k, for $i < k \leq j$, whose weight is smaller than that of q_i. Hence, q_j cannot be equal to q_i. For the other cases of $q_i \in Q_2$, observe that $\mathcal{A}_\varphi$ can only move to a state q_{i+1} with a smaller weight. This follows from the definition of the transition function and the function *weight*. Hence, q_j cannot be equal to q_i.

Case $q_i \in Q_3$. Since $\mathcal{A}_\varphi$ simulates a run of a 1-way automaton between configuration π_i and π_j, the head position h_j is different from h_i. □

A 2ABA $\mathcal{B} = (S, \Sigma, \eta, s_I, E)$ is *locally 1-way* if $\eta(s, b) \in \mathcal{B}^+(S \times \{0, 1\}) \cup \mathcal{B}^+(S \times \{-1, 0\})$, for every $s \in S$ and $b \in \Sigma$. Let us first make the following general statement that any 2ABA can be transformed into a language-equivalent 2ABA that is locally 1-way by doubling the state space.

Lemma 8. *For every 2ABA $\mathcal{B}$, there is a language-equivalent 2ABA $\mathcal{B}'$ that is locally 1-way and that has size at most $2\|\mathcal{B}\|$.*

Proof. Assume that $\mathcal{B} = (Q, \Sigma, \delta, q_I, F)$. We define $\mathcal{B}' = (Q \cup Q', \delta', q_I, F)$, where $Q' := \{q' \mid q \in Q\}$ and the transition function $\delta' : (Q \cup Q') \times \Sigma \to \mathcal{B}((Q \cup Q') \times \mathbb{D})$ is defined as follows. Let $q \in Q$ and $b \in \Sigma$. We define $\delta'(q', b) := (q, -1)$ and $\delta'(q, b)$ as the Boolean formula $\delta(q, b)$, where we replace the propositions $(p, -1)$ by $(p', 0)$, for each state $p \in Q$. The 2ABA $\mathcal{B}'$ works as follows. Whenever $\mathcal{B}$ moves its read-only head to the left and goes to state p, $\mathcal{B}'$ mimics this by first going to the state p' without moving the read-only head and in the next step $\mathcal{B}'$ goes from state p' to state p, where it also moves the read-only head to the left. Obviously, $\mathcal{B}'$ is locally 1-way and accepts the language $L(\mathcal{B})$. □

We remark that the above given transformation in Lemma 8 is not needed in our setting since the constructed 2ABA $\mathcal{A}_\varphi$ is already locally 1-way. This can be easily seen by inspecting $\mathcal{A}_\varphi$'s transition function.

Lemma 9. *The 2ABA $\mathcal{A}_\varphi$ is locally 1-way.*

4.2 From Loop-free and Locally 1-Way 2ABAs to NBAs

In the following, we show how the alternating automaton from the previous subsection for an RTL formula in negation normal form can be translated into an NBA. The presented construction is based on an improvement of an alternation-elimination construction from [12]. Here, we additionally exploit the fact that the given 2ABA is locally 1-way. Overall, for an RTL formula ψ, the resulting language-equivalent NBA has size $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$. With the construction in [12], we would obtain an NBA of size $\mathcal{O}(2^{4 \cdot 2^{2\|\psi\|}})$. Another advantage of the new construction is that it avoids the explicit representation of an extended alphabet, which is used in one of the intermediate construction steps in [12] and which is of exponential size. The presented construction also allows for a symbolic implementation [11], which can be used in tools like NuSMV [8] for satisfiability and finite-state model checking. See [6] for such implementations and an evaluation of constructions for the special case of 1-way alternating Büchi automata.

Theorem 10. *For a loop-free and locally 1-way 2ABA $\mathcal{A}$, there is a language-equivalent NBA $\mathcal{B}$ of size $\mathcal{O}(|\Sigma| \cdot 2^{2\|\mathcal{A}\|})$, where Σ is the alphabet of $\mathcal{A}$.*

Before we present the proof details of the automata construnction to prove this theorem, we give some intuition for the construction. For an input word w, the NBA $\mathcal{B}$ guesses a run r of $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ on w and checks whether this run is accepting. For this, as in [12, 28], $\mathcal{B}$ represents r as a sequence of state sets $R_0 R_1 \ldots \in (2^Q)^\omega$, where each R_i contains the state q iff there is a path in r that visits (q, i). In the case where $\mathcal{A}$ is 1-way, each R_i consists of the states that occur in the ith level of the run r. Note that in the general case where $\mathcal{A}$ is 2-way, R_i might contain states that occur in different levels of r. For instance, R_i contains the states q and q' from different levels if r contains a path of the form $(q_I, 0) \ldots (q, i) \ldots (q', i) \ldots$. Since $\mathcal{A}$ is locally 1-way, we can locally check whether such a sequence $R_0 R_1 \ldots$ represents a run of $\mathcal{A}$ on w. For doing so, $\mathcal{B}$ stores the set R_{i+1} and the letter w_{i+2} after reading the ith letter of w. For a state $q \in R_i$ with $\delta(q, w_i) \in \mathcal{B}^+(Q \times \{0, 1\})$, the set $(R_i \times \{0\}) \cup (R_{i+1} \times \{1\})$ must be a model of $\delta(q, w_i)$. $\mathcal{B}$ checks this when reading the letter w_i. For $\delta(q, w_i) \in \mathcal{B}^+(Q \times \{-1, 0\})$ and $i > 0$, $(R_{i-1} \times \{-1\}) \cup (R_i \times \{0\})$ must be a model of $\delta(q, w_i)$. $\mathcal{B}$ already checks this when it reads the $(i-1)$th input letter by using the guessed letter w_i. Additionally, $\mathcal{B}$ must check that every path in r visits configurations with an accepting state infinitely often. Since $\mathcal{A}$ is loop-free the run r is accepting iff there are indexes $i_0 < i_1 < \ldots$ such that each path in r that visits a configuration (q, i_j) visits a configuration with an accepting state before visiting (q', i_{j+1}), for every $j \in \mathbb{N}$. Similar to the alternation-elimination construction by Miyano and Hayashi [24] for 1-way alternating Büchi automata, $\mathcal{B}$ checks this property with an additional component in the state space and its set of accepting states.

Remark 11. We remark that for the sketched construction, a weaker but less intuitive condition for the given 2ABA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ than the condition of locally 1-way suffices. Namely, for every $q \in Q$, $b \in \Sigma$, and $M \subseteq Q \times \{-1, 0, 1\}$, it suffices to require that if $M \cup (Q \times \{1\}) \models \delta(q, b)$ and $M \cup (Q \times \{-1\}) \models$

$\delta(q,b)$ then $M \models \delta(q,b)$. It is easy to see that this property holds for locally 1-way 2ABAs. Note that we can check this weaker property by transforming the Boolean formulas of the automaton's transition function into CNF and checking whether each clause is in $\mathcal{B}^+(Q \times \{0,1\})$ or $\mathcal{B}^+(Q \times \{-1,0\})$. Furthermore, we note that this weaker property is of practical interest. We can exploit it to reduce the size of the 2ABA $\mathcal{A}_\varphi$ that we obtain from our construction for an RTL formula φ in negation normal form.

Proof (Theorem 10). We give now the details of the construction with this weaker condition. For the loop-free, locally 1-way 2ABA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$, let $\mathcal{B} := (Q', \Sigma, \eta, p_I, E)$ be the NBA with $Q' := \{p_I\} \cup (\Sigma \times 2^Q \times 2^{Q \setminus F})$ and $E := \Sigma \times 2^Q \times \{\emptyset\}$. For $b \in \Sigma$, the transition function η is defined as follows. For the initial state p_I, $\eta(p_I, b)$ contains the state (c, R', S') iff $S' = R' \setminus F$ and there is some $R \subseteq Q$ such that $q_I \in R$ and

$$I \models \bigwedge_{q \in R} \delta(q,b) \qquad \text{and} \qquad I' \models \bigwedge_{q \in R'} \delta(q,c)\,,$$

where $I := \big(R \times \{0\}\big) \cup \big(R' \times \{1\}\big)$ and $I' := \big(R \times \{-1\}\big) \cup \big(R' \times \{0\}\big) \cup \big(Q \times \{1\}\big)$. For a state (a, R, S), the transition function η is defined as follows. If $a \neq b$ then $\eta\big((a,R,S),b\big) := \emptyset$. If $a = b$ then $\eta\big((a,R,S),b\big)$ contains the state (c, R', S') iff the following conditions for c, R', and S' are satisfied: First,

$$M \models \bigwedge_{q \in R} \delta(q,b) \qquad \text{and} \qquad M' \models \bigwedge_{q \in R'} \delta(q,c)\,,$$

where $M := \big(R \times \{0\}\big) \cup \big(R' \times \{1\}\big) \cup \big(Q \times \{-1\}\big)$ and $M' := \big(R \times \{-1\}\big) \cup \big(R' \times \{0\}\big) \cup \big(Q \times \{1\}\big)$. Second, $S' = R' \setminus F$ if $S = \emptyset$, and if $S \neq \emptyset$ then

$$N \models \bigwedge_{q \in S} \delta(q,b) \qquad \text{and} \qquad N' \models \bigwedge_{q \in S'} \delta(q,c)\,,$$

where $N := \big((S \cup F) \times \{0\}\big) \cup \big((S' \cup F) \times \{1\}\big) \cup \big(Q \times \{-1\}\big)$ and $N' := \big((S \cup F) \times \{-1\}\big) \cup \big((S' \cup F) \times \{0\}\big) \cup \big(Q \times \{1\}\big)$.

In the following, we prove that $L(\mathcal{A}) = L(\mathcal{B})$.

($\subseteq$) Assume that r is an accepting run of $\mathcal{A}$ on $w \in \Sigma^\omega$. We define a run ϱ of $\mathcal{B}$ on w as follows. Note that ϱ has to be a sequence of the form $p_I(a_1, R_1, S_1)(a_2, R_2, S_2)\ldots$ with $a_i \in \Sigma$, $R_i \subseteq Q$, and $S_i \subseteq Q \setminus F$, for all $i > 0$. We define the components a_i, R_i, and S_i separately.

We define $a_i := w_i$, for $i > 0$. The other components are defined as follows. For $i \geq 0$, we define

$$R_i := \{q \in Q \mid \text{there is a node } v \text{ in } r \text{ such that } r(v) = (q,i)\}\,.$$

The sets S_i are inductively defined:

$$S_0 := \emptyset \qquad \text{and} \qquad S_i := R_i \setminus F\,,$$

for $i > 0$ and $S_{i-1} = \emptyset$. For $i > 0$ and $S_{i-1} \neq \emptyset$, we define

$$\begin{aligned} S_i := \{q \in Q \setminus F \mid & \text{ there is a state } p \in S_{i-1} \text{ such that there is a path in } r \\ & \text{that is labeled by } (p, i-1)(q_0, i) \dots (q_n, i)(q, i) \text{ and} \\ & q_0, \dots, q_n \notin F\} \cup \\ \{q \in Q \setminus F \mid & \text{ there is a state } p \in S_{i-1} \text{ such that there is a path in } r \\ & \text{that is labeled by } (q, i)(q_0, i) \dots (q_n, i)(p, i-1) \text{ and} \\ & q_0, \dots, q_n \notin F\}\,. \end{aligned}$$

Let us first prove that for every $i \geq 0$, there is some $j \geq i$ such that $S_j = \emptyset$. Let $i \geq 0$ and let $S_i \neq \emptyset$. Assume that there is no $j \geq i$ with $S_j = \emptyset$. From the sets $S_i, S_{i-1}, \dots$, we obtain a directed graph G with the vertexes (p, j) with $p \in S_j$. The edges are according to $\mathcal{A}$'s transition function and the input word w. Observe that G is finitely branching and every vertex is reachable from some vertex of the form (p, i). Furthermore, G is infinite, since we assume that $S_j \neq \emptyset$, for all $j \geq i$. By König's Lemma, it follows that there is an infinite path in G starting from a vertex (p, i). This path never visit a vertex in which a state in F occurs. This contradicts the assumption that every path in the run r visits infinitely often a configuration in which a state in F occurs.

It remains to prove that $(a_{i+1}, R_{i+1}, S_{i+1}) \in \delta((a_i, R_i, S_i), w_i)$, for all $i > 0$, and that $(a_1, R_1, S_1) \in \delta(p_I, w_0)$.

- Let $i > 0$ and assume that $q \in R_i$. By definition, there is a node u in r with $r(u) = (q, i)$. Let $v_1, \dots, v_n$ be the children of u in r. Without loss of generality, we assume that $r(v_j) = (q_j, h_j)$ with $h_j \in \{i-1, i, i+1\}$, for all j with $1 \leq j \leq n$. Since r is a run, we have that $\{(q_1, h_1 - i), \dots, (q_n, h_n - i)\} \models \delta(q, w_i)$. For all j with $1 \leq j \leq n$, we have that $q_j \in R_i$ if $h_j = i$, and $q_j \in R_{i+1}$ if $h_j = i + 1$. Since $\delta(q, a_i)$ is a positive Boolean formula, $w_i = a_i$ by definition, and $M \models \delta(q, w_i)$ we have that

  $$\big(R_i \times \{0\}\big) \cup \big(R_{i+1} \times \{1\}\big) \cup \big(Q \times \{-1\}\big) \models \delta(q, a_i)\,.$$

  Assume that $q \in R_{i+1}$. By definition, there is a node u in r with $r(u) = (q, i+1)$. Let $v_1, \dots, v_n$ be the children of u in r. Without loss of generality, we assume that $r(v_j) = (q_j, h_j)$ with $h_j \in \{i, i+1, i+2\}$, for all j with $1 \leq j \leq n$. Since r is a run and by the definition of a_{i+1}, we have that $\{(q_1, h_1 - i - 1), \dots, (q_n, h_n - i - 1)\} \models \delta(q, a_{i+1})$. For all j with $1 \leq j \leq n$, we have that $q_j \in R_i$ if $h_j = i$, and $q_j \in R_{i+1}$ if $h_j = i + 1$. Since $\delta(q, a_{i+1})$ is a positive Boolean formula and $M \models \delta(q, a_{i+1})$ we have that

  $$\big(R_i \times \{-1\}\big) \cup \big(R_{i+1} \times \{0\}\big) \cup \big(Q \times \{1\}\big) \models \delta(q, a_{i+1})\,.$$

  If $S_i = \emptyset$ then $S_{i+1} = R_{i+1} \setminus F$ by definition. For the case $S_i \neq \emptyset$, the reasoning is similar as for the second components R_i and R_{i+1} of the states.
- Note that $S_1 = R_1 \setminus F$ by definition. Furthermore, we have that $q_I \in R_0$, since the root of r is labeled by the configuration $(q_I, 0)$. The reasoning is similar to the other case by setting i to 0.

($\supseteq$) Assume that ϱ is an accepting run on $w \in \Sigma^\omega$. Without loss of generality, assume that ϱ has the form $p_I(a_1, R_1, S_1)(a_2, R_2, S_2)\ldots$ with $a_i \in \Sigma$, $R_i \subseteq Q$, and $S_i \in Q \setminus F$, for all $i > 0$. Furthermore, let $R_0 \subseteq Q$ be a set for which we require its existence in the definition of the transition function from state p_I.

We construct a run r of $\mathcal{A}$ on w inductively over the length of a node. An invariant of the construction is that if a node is labeled by (q, h) then $q \in R_h$. We label the root as $(q_I, 0)$. The construction invariant is obviously satisfied, since in the definition of the transitions from state p_I we require that $q_I \in R_0$. Let u be a node of r with $r(u) = (q, h)$. We have that $q \in R_h$. There are two cases.

- Assume that $h > 0$. By the definition of the transition function, we have that

  $$\big(R_h \times \{0\}\big) \cup \big(R_{h+1} \times \{1\}\big) \cup \big(Q \times \{-1\}\big) \models \delta(q, a_i)$$

  and

  $$\big(R_{h-1} \times \{-1\}\big) \cup \big(R_h \times \{0\}\big) \cup \big(Q \times \{1\}\big) \models \delta(q, a_i)\,.$$

  By the assumption that $\mathcal{A}$ is locally 1-way (or the more general assumption from Remark 11), we have that

  $$\big(R_{h-1} \times \{-1\}\big) \cup \big(R_h \times \{0\}\big) \cup \big(R_{h+1} \times \{1\}\big) \models \delta(q, a_i)\,.$$

  Let $M \subseteq \big(R_{h-1} \times \{-1\}\big) \cup \big(R_h \times \{0\}\big) \cup \big(R_{h+1} \times \{1\}\big)$ be a minimal model of $\delta(q, a_i)$. We define the children of u as follows: for each proposition $(p, d) \in M$, u has a child v that is labeled by $(p, h + d)$. This definition obviously satisfies the construction invariant.
- The construction step for $h = 0$ is similar. We omit it.

It is straightforward to see that r is a run of $\mathcal{A}$ on w. Note that from the definition of $\mathcal{B}$'s transition function, we have that $a_i = w_i$, for all $i > 0$.

It remains to show that r is accepting. Assume that π is a path in r with $r(\pi) = (p_0, h_0)(p_1, h_1)\ldots$. Since $\mathcal{A}$ is loop-free and we have always taken a minimal model in the construction of r, no configuration occurs twice in $r(\pi)$. For the sake of contradiction, assume that there is an $i \geq 0$ such that $p_j \notin F$, for all $j \geq i$. There is an integer $k \geq i$ such that $p_k \in S_k$. Since there is no configuration after position i in $r(\pi)$ with an accepting state, we have that $S_j \neq \emptyset$, for all $j \geq k$. This contradicts the assumption that ϱ is an accepting run of $\mathcal{B}$. □

We obtain the following result by putting the two constructions from Section 4.1 and Theorem 10 together.

Theorem 12. *For any RTL formula ψ, there is a language-equivalent NBA $\mathcal{C}$ of size $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$.*

Proof. First, we transform ψ into a logically equivalent formula ψ' that is in negation normal of size $2\|\psi\|$. Let $\mathcal{A}_{\psi'}$ be the 2ABA that we obtain from ψ' by the construction in Section 4.1. By the Lemmas 6, 7, and 9, $\mathcal{A}_{\psi'}$ is loop-free, locally 1-way, and $\|\mathcal{A}_{\psi'}\| \leq 4 + 2^{2\|\psi\|}$. By Lemma 5, $\mathcal{A}_{\psi'}$ accepts the language $\{\#w \mid w \in L(\psi)\}$. By Theorem 10, we translate $\mathcal{A}_{\psi'}$ into a language-equivalent NBA $\mathcal{B} = (Q, \{\#\} \cup 2^P, \delta, q_I, F)$ with $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$ states. Note that $|2^P| \leq 2^{\|\psi\|}$, where we assume without loss of generality that P contains only the propositions that occur in ψ. We define the NBA $\mathcal{C} = (Q, 2^P, \delta', q_I, F)$, where $\delta'(q, a) := \delta(q, a)$, for $q \in Q \setminus \{q_I\}$ and $a \in 2^P$, and $\delta'(q_I, a) := \{q' \mid q' \in \delta(q, a), \text{ for some } q \in \delta(q_I, \#)\}$. We have that $L(\mathcal{C}) = L(\psi)$ and $\|\mathcal{C}\| \in \mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$. □

Let us make the following remark on the size of the resulting NBA of the presented construction for RTL.

Remark 13. The size of the constructed 2ABA $\mathcal{A}_\varphi$ in Section 4.1 depends on the number of subformulas of the given RTL formula φ in negation normal form and the sizes of the automata for the SEREs in φ. First, we remark that the construction shares subformulas and the SEREs occurring in them, and that $|Sub(\varphi)| \leq \|\varphi\|$. Second, for a bounded number of intersection operators in the SEREs of the formula, we obtain a polynomial upper bound on the sizes of automata for the SEREs. In particular, for an SERE α with n intersection operators, there is a language-equivalent NFA of size $\|\alpha\|^{1+n}$. It follows that the size of the 2ABA $\mathcal{A}_\varphi$ is bounded by $4 + |Sub(\varphi)| + \sum_{\alpha \text{ SERE in } \varphi} \|\alpha\|^{1+Is(\alpha)}$ when φ is in negation normal form. With this new upper bound, we conclude that for an RTL formula ψ, there is a language-equivalent NBA of size $\mathcal{O}(2^{5(|Sub(\psi)| + \sum_{\alpha \text{ SERE in } \psi} \|\alpha\|^{1+Is(\alpha)})})$, which refines the upper bound in Theorem 12. Note that the transformation of ψ into an RTL formula in negation normal form doubles $|Sub(\psi)|$ and the number of SEREs in the subformulas in the worst case.

4.3 Consequences of the Translation

We conclude this section by proving some facts that follow from Theorem 12.

Since SVA can already express all ω-regular languages, we have that RTL describes exactly the ω-regular languages. Moreover, SVA, PSL, and RTL share the same computational complexity. In particular, the satisfiability and the model-checking problem for RTL are EXPSPACE-complete in general and PSPACE-complete for RTL formulas with a bounded number of intersection operators.

Corollary 14. *The satisfiability problem and model-checking problem for RTL are EXPSPACE-complete in general and PSPACE-complete for RTL formulas with a bounded number of intersection operators.*

Proof. Satisfiability of φ can be decided by checking the NBA $\mathcal{A}_\varphi$ according to Theorem 12 for emptiness. It is well known that the emptiness problem for NBAs is in NLOGSPACE, and therefore in NEXPSPACE for arbitrary RTL formulas. According to Savitch's Theorem, this equals EXPSPACE [27]. Moreover, if the

number intersection operators in the SEREs that occur in φ is bounded by some constant, then $\mathcal{A}_\varphi$ is only of size $\mathcal{O}(2^{poly(\|\varphi\|)})$ rather than doubly exponential. In this case, we can check emptiness of $\mathcal{A}_\varphi$ in NPSPACE, which equals PSPACE using Savitch's Theorem again.

Finally, the hardness results for the statisfiability and the model-checking problem for RTL follow directly from the hardness results for PSL and SVA, which are shown in [7, 19].

Not surprisingly, these bounds transfer to the model-checking problem (i.e., the question whether all paths in a given Kripke structure $\mathcal{K}$ from a state s of $\mathcal{K}$ satisfy a given RTL formula). Since RTL is closed under negation, one can build an NBA $\mathcal{A}_{\neg\varphi}$ and check whether the intersection of the automaton's language with the language of the Kripke structure $\mathcal{K}$ is empty. □

Another similarity between the logics is that they all have the small model property of doubly exponential size. In particular, there is a constant $c > 0$ such that a satisfiable RTL formula φ has a model of the form uv^ω with $|uv| \leq c \cdot 2^{3 \cdot 2^{2\|\varphi\|}}$.

Corollary 15. *Every satisfiable RTL formula φ has a model of the form uv^ω with $|uv| \in \mathcal{O}(2^{3 \cdot 2^{2\|\varphi\|}})$.*

Proof. It is well known that every NBA with n states that accepts a non-empty language accepts a word of the form uv^ω such that $|u| + |v| \leq n$. With this, the statement follows immediately from Theorem 12. □

Since PSL/SVA and RTL describe the same class of properties, the question arises of their relative succinctness. The next theorem states an upper bound on the translation from RTL to SVA. Roughly speaking, for the proof, we translate an RTL formula into an NBA and then into an ω-regular expression, which we finally translate into an SVA formula.

Theorem 16. *For any RTL formula φ, there is an initially equivalent SVA formula of size $2^{\mathcal{O}(2^{2^{2\|\varphi\|+2}})}$ and in which the intersection operator does not occur.*

Proof. According to Theorem 12, we construct an NBA $\mathcal{B} = (Q, \Sigma, \delta, q_I, F)$ with $L(\mathcal{B}) = L(\varphi)$ and $\|\mathcal{B}\| \in \mathcal{O}(2^{3 \cdot 2^{2\|\varphi\|}})$. For $s, t \in Q$, $\mathcal{B}^t_s$ denotes the NFA $(Q, \Sigma, \delta, s, \{t\})$, i.e., we view $\mathcal{B}$ as an automaton over finite words with the initial state s and the singleton acceptance set $\{t\}$. Note that the finite-word language $L(\mathcal{B}^t_s)$ can be expressed as an SERE (without the intersection operator) α^t_s of size $\mathcal{O}(2^{\|\mathcal{B}^t_s\|})$, see [17].

In the following, we write L^ω for the set of words that are obtained by an the infinite concatenation of words from the lanuage L of finite words.

The language of $\mathcal{B}$ can be described as follows in terms of the languages of the NFAs $\mathcal{B}^f_{q_I}$ and $\mathcal{B}^f_f$ with $f \in F$:

$$L(\mathcal{B}) = \bigcup_{f \in F} L(\mathcal{B}^f_{q_I}) \left(L(\mathcal{B}^f_f) \setminus \{\varepsilon\}\right)^\omega . \tag{5}$$

Without loss of generality, we assume in the following that $L(\mathcal{B}_f^f) \neq \emptyset$ and $L(\mathcal{B}_f^f) \neq \{\varepsilon\}$, for each $f \in F$. Furthermore, we assume that every final state in $\mathcal{B}$ is reachable from the initial state, i.e., $L(\mathcal{B}_{q_I}^f) \neq \emptyset$, for each $f \in F$. Observe that for an SERE α with $L(\alpha) \neq \emptyset$ and an RTL formula ψ, we have that

$$L(\alpha)L(\psi) = L(\alpha \diamond\!\!\rightarrow \mathsf{X}\psi) \cup \begin{cases} L(\psi) & \text{if } \varepsilon \in L(\alpha), \\ \emptyset & \text{otherwise,} \end{cases} \tag{6}$$

and if we additionally assume that $L(\alpha) \neq \{\varepsilon\}$ then it holds that

$$\big(L(\alpha) \setminus \{\varepsilon\}\big)^\omega = L\big(\alpha^* \,\square\!\!\rightarrow \mathsf{X}(\alpha \diamond\!\!\rightarrow \mathsf{tt})\big)\,. \tag{7}$$

With the equalities (5), (6), and (7) at hand, it is straightforward to see that the following PSL formula φ' is initially equivalent to φ:

$$\varphi' := \Big(\bigvee_{f \in F} \alpha_{q_I}^f \diamond\!\!\rightarrow \mathsf{X}\psi_f^f \Big) \vee \begin{cases} \psi_{q_I}^{q_I} & \text{if } q_I \in F, \\ \mathsf{ff} & \text{otherwise,} \end{cases}$$

where $\psi_s^s := (\alpha_s^s)^* \,\square\!\!\rightarrow \mathsf{X}(\alpha_s^s \diamond\!\!\rightarrow \mathsf{tt})$, for a state $s \in F$.

Obviously, we have that no intersection operator occurs in the SEREs of φ'. Furthermore, it holds that

$$\|\varphi'\| \in \mathcal{O}\big(\sum_{f \in F} (2^{\|\mathcal{B}_{q_I}^f\|} + 2 \cdot 2^{\|\mathcal{B}_f^f\|}) \big) \subseteq \mathcal{O}(\|\mathcal{B}\| \cdot 2^{\|\mathcal{B}\|+2}) \subseteq 2^{\mathcal{O}(2^{2^{2\|\varphi\|+2}})}\,.$$

□

It is fair to ask whether the upper bound in Theorem 16 is optimal, i.e., whether there is a family of RTL formulas such that every initially equivalent family of PSL formulas must be triply exponentially larger. The result on the small model property shows that such a lower bound cannot be proved by comparing the model sizes (see, e.g., the Gap Lemma in [20]). We were only able to establish an exponential lower bound. This result is presented in the next section.

5 Succinctness Gaps

In this section, we prove an exponential succinctness gap between RTL and PSL/SVA, i.e., there is a family $(\Phi_n)_{n>0}$ of RTL formulas such that for every family $(\Psi_n)_{n>0}$ of PSL or SVA formulas, if Ψ_n is initially equivalent to Φ_n for all $n > 0$, then $\|\Psi_n\|$ is exponential in $\|\Phi_n\|$. In fact, our result is stronger since the formulas Φ_n that we define are just PSVA formulas. The proof of this succinctness result can easily be adapted to show that PSVA and, hence, RTL, is double exponentially more succinct than PLTL on the star-free languages.

Our proof for the succinctness gap between PSVA and SVA has a similar flavor as the proof in [23], which shows that PLTL is exponentially more succinct

than LTL. However, our proof is more involved since we must take SEREs into account. In fact, the formulas in the family of PLTL formulas that is used in [23] are initially equivalent to SVA formulas of linear size. From this observation, we conclude that SVA is exponentially more succinct than LTL on the star-free languages.

Lemma 17. *For every $n > 0$, there is an SVA formula Θ_n such that for any LTL formula Ξ_n, if $L(\Xi_n) = L(\Theta_n)$ then $\|\Xi_n\| \in \Omega(2^{\|\Theta_n\|})$.*

Proof. Let P be the set $\{p_0, p_1, \ldots, p_n\}$ of propositions. We define Θ_n as the SVA formula $\alpha_n \boxdot\!\!\rightarrow \mathsf{ff}$, where α_n is the SERE

$$\big((p_0 \,;\, \mathsf{tt}^* \,;\, \neg p_0) \cup (\neg p_0 \,;\, \mathsf{tt}^* \,;\, p_0)\big) \cap \bigcap_{1 \leq i \leq n} \big((p_i \,;\, \mathsf{tt}^* \,;\, p_i) \cup (\neg p_i \,;\, \mathsf{tt}^* \,;\, \neg p_i)\big)\,.$$

It is easy to see that Θ_n is initially equivalent to the PLTL formula

$$\mathsf{G}\big(\bigwedge_{1 \leq i \leq n} (p_i \leftrightarrow \mathsf{OH} p_i) \rightarrow (p_0 \leftrightarrow \mathsf{OH} p_0)\big)\,.$$

From [23], it follows that any LTL formula Ξ_n that is initially equivalent to Θ_n is exponentially larger than Ξ_n. □

Let us now turn to the succinctness gap between PSVA and SVA. For this, we first introduce so-called n-counting words, which can be defined in SVA by formulas of size $\mathcal{O}(n)$. In the following, let $n > 0$, P_n be the set $\{c_0, \ldots, c_{n-1}, p, q\}$ of propositions, and Σ_n the alphabet 2^{P_n}. The *n-value* of the letter $b \in \Sigma_n$ is

$$\mathit{val}_n(b) := \sum_{0 \leq i < n} 2^{c_i'} \qquad \text{with} \qquad c_i' := \begin{cases} 1 & \text{if } c_i \in b, \\ 0 & \text{otherwise.} \end{cases}$$

In other words, the n-value of b is obtained by reading $c_0, \ldots, c_{n-1}$ as bits of a positive integer in binary representation. A word $w \in \Sigma_n^\omega$ is *n-counting* if $\mathit{val}_n(w_0) = 0$ and $\mathit{val}_n(w_{i+1}) \equiv \mathit{val}_n(w_i) + 1 \mod 2^n$, for all $i \in \mathbb{N}$.

Lemma 18. *For every $n > 0$, there is an SVA formula count_n of size $\mathcal{O}(n)$ such that $L(\mathit{count}_n) \subseteq \Sigma_n^\omega$ is the language of n-counting words.*

Proof. Recall that the temporal operators G and X can easily be defined in SVA by using the operator $\diamond\!\!\rightarrow$.

We define count_n as the SVA formula

$$\big(\bigwedge_{0 \leq i < n} \neg c_i\big) \wedge \mathsf{G}\big(\neg \mathsf{X} c_0 \leftrightarrow c_0\big) \wedge \bigwedge_{1 \leq i < n} \mathsf{G}\big(\mathsf{X} c_i \leftrightarrow \big(c_i \leftrightarrow (c_{i-1} \rightarrow \mathsf{X} c_{i-1})\big)\big)\,.$$

It is easily checked that $w \in \Sigma_n^\omega$ is a model of count_n iff w is n-counting. □

An *n-segment* of a word $w \in \Sigma_n^\omega$ is a subword $v = w_i \ldots w_{i+2^n-1}$ such that $i \equiv 0 \mod 2^n$, for some $i \in \mathbb{N}$. The n-segment v is *initial* if $i = 0$. For a proposition $r \in P$, the words $u, v \in \Sigma_n^*$ are *r-equal* if $|u| = |v|$ and $r \in u_i \Leftrightarrow r \in v_i$, for all $i \in \mathbb{N}$ with $i < |v|$. In other words, the projection of two r-equal words onto r yields the same word. Let L_n and L_n' be the following languages:

- L_n consists of the n-counting words $w \in \Sigma_n^\omega$ such that if an n-segment of w is p-equal to the initial n-segment w then they are also q-equal.
- L'_n consists of the n-counting words $w \in \Sigma_n^\omega$ such that if the n-segments u and v of w are p-equal then they are also q-equal.

The languages L_n and L'_n have the following properties.

Lemma 19. *For every $n > 0$, there is a PSVA formula Φ_n of size $\mathcal{O}(n)$ such that $L(\Phi_n) = L_n$.*

Proof. First, we define the SERE $\mathit{samepos}_n$ such that for every subword $v \in \Sigma_n^*$ of an n-counting word $w \in \Sigma_n^\omega$, it holds that $v \in L(\mathit{samepos}_n)$ iff $v = w_{i..j}$, for some $i, j \in \mathbb{N}$ with $i < j$ and $i \equiv j \mod 2^n$. Note that since v is a finite subword of an n-counting word, one only has to assert that the n-values of the first and the last letter of v are equal. We define

$$\mathit{samepos}_n := \bigcap_{0 \le i < n} \big((c_i \,;\, \mathsf{tt}^* \,;\, c_i) \cup (\neg c_i \,;\, \mathsf{tt}^* \,;\, \neg c_i)\big)\,.$$

With the SERE $\mathit{samepos}_n$ at hand, we easily define an RTL formula that checks whether a position is in the initial n-segment of an n-counting word:

$$\mathit{initial}_n := \neg(\mathit{samepos}_n \mathrel{\Diamond\!\!\!\!-\!\!\!\!\rightarrow} \mathsf{tt})\,.$$

For an n-counting word $w \in \Sigma_n^\omega$ and $i \in \mathbb{N}$, we have that $w, i \models \mathit{initial}_n$ iff $i < 2^n$. Moreover, for an RTL formula ψ, we define

$$\mathit{back}_n^\psi := \mathit{samepos}_n \mathrel{\Diamond\!\!\!\!-\!\!\!\!\rightarrow} (\mathit{initial}_n \wedge \psi)\,.$$

For an n-counting word $w \in \Sigma_n^\omega$ and $i \in \mathbb{N}$, it holds that $w, i \models \mathit{back}_n^\psi$ iff $w, i \mod 2^n \models \psi$. Intuitively, back_n^ψ goes back in the word w until it reaches the position in the initial n-segment with same counter values as the current position, and there it checks whether ψ holds. Next, we define the SERE $\mathit{within}_n := (\neg c_{n-1})^* \,;\, (c_{n-1})^*$. We use it for checking if a larger position than the current position is still in the same n-segment of an n-counting word. Note that the highest bit c_{n-1} of the counter is only allowed to change its value from 0 to 1 once. The formula $\mathit{start}_n := \bigwedge_{0 \le i < n} \neg c_i$ checks whether a position is the first one of an n-segment in an n-counting word.

Finally, consider the RTL formula $\Phi_n := \mathit{count}_n \wedge \varphi_n$, where

$$\varphi_n := \mathsf{G}\Big(\mathit{start}_n \wedge \big(\mathit{within}_n \mathrel{\Box\!\!\!\!\rightarrow} (p \leftrightarrow \mathit{back}_n^p)\big) \rightarrow \big(\mathit{within}_n \mathrel{\Box\!\!\!\!\rightarrow} (q \leftrightarrow \mathit{back}_n^q)\big)\Big)\,.$$

The formula φ_n states that for any n-segment of an n-counting word, if the Boolean value of p at every position of that n-segment coincides with the Boolean value of p at the corresponding position of the initial n-segment, then the same holds for the Boolean values of q. Hence, we have that $L(\Phi_n) = L_n$. Furthermore, it is easy to see that $\|\Phi_n\| \in \mathcal{O}(n)$. □

Lemma 20. *For every $n > 0$, if $\mathcal{B}$ is an NBA with $L(\mathcal{B}) = L'_n$ then $\|\mathcal{B}\| \ge 2^{2^{2^n}}$.*

Proof. Throughout the proof, let $N := 2^{2^n}$. Note that there are N different n-segments with respect to the proposition p. Recall that an n-segment has length 2^n. Let $v_0, \ldots, v_{N-1} \in \{\emptyset, \{p\}\}^*$ be an enumeration of all these n-segments with $v_i = v_{i,0} \ldots v_{i,2^n-1}$. For $S \subseteq \{0, \ldots, N-1\} \times \{0, \ldots, 2^n - 1\}$ and $i \in \{0, \ldots, N-1\}$, we define $v_i^S := v_{i,0}^S \ldots v_{i,2^n-1}^S$, where

$$v_{i,j}^S := \begin{cases} v_{i,j} \cup w_j \cup \{q\} & \text{if } (i,j) \in S, \\ v_{i,j} \cup w_j & \text{otherwise,} \end{cases}$$

for $j \in \{0, \ldots, 2^n - 1\}$ and an n-counting word $w \in (2^{\{c_0, \ldots, c_{n-1}\}})^\omega$. Note that in the above definition we add the counter values to the n-segment v_i and the set S prescribes at which positions the proposition q should be added to the n-segments $v_0, \ldots, v_{N-1}$. Finally, we define the word $v^S := v_0^S \ldots v_{N-1}^S$. Observe that there are $M := 2^{N \cdot 2^n}$ different such sets S. Note that $M \geq 2^{2^{2^n}}$. Also, for every such S we have $(v^S)^\omega \in L(\mathcal{B})$.

Suppose that $\|\mathcal{B}\| < M$. Then, by the pigeon hole principle, there are sets $S, S' \subseteq \{0, \ldots, N-1\} \times \{0, \ldots, 2^n - 1\}$ with $S \neq S'$ such that an accepting run π of $\mathcal{B}$ on $(v^S)^\omega$ and an accepting run π' of $\mathcal{B}$ on $(v^{S'})^\omega$ visit the same state s after $N \cdot 2^n$ many steps, i.e., after reading the prefixes v^S and $v^{S'}$ respectively. The suffixes of these runs could be interchanged which would create accepting runs on $(v^S)(v^{S'})^\omega$ for example, even though $(v^S)(v^{S'})^\omega \notin L_n'$. □

With the above lemmas we obtain our succinctness result for PSVA and SVA.

Theorem 21. *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for every SVA formula Ψ_n, if $L(\Psi_n) = L_n$ then $\|\Psi_n\| \in \Omega(2^{\|\Phi_n\|})$.*

Proof. For a given $n > 0$, take the PSVA formula Φ_n from Lemma 19. Suppose that Ψ_n is an SVA formula that is initially equivalent to Φ_n. Let $\Psi_n' := \mathit{count}_n \wedge \mathsf{G}(\neg c_0 \wedge \cdots \wedge \neg c_{n-1} \rightarrow \Psi_n)$. Note that Ψ_n' expresses that a model is n-counting and each two p-equal n-segments in a model are also q-equal, i.e., $L(\Psi_n') = L_n'$. By Theorem 12, there is an NBA $\mathcal{B}$ of size $2^{2^{\mathcal{O}(\|\Psi_n'\|)}}$ and $L(\mathcal{B}) = L(\Psi_n')$. By Lemma 20, we have that $\|\mathcal{B}\| \geq 2^{2^{2^n}}$. It follows that $\|\Psi_n'\| \in \Omega(2^{\|\Phi_n\|})$. Since Ψ_n' is linear in the size of Ψ_n, we conclude that $\|\Psi_n\| \in \Omega(2^{\|\Phi_n\|})$. □

Note that L_n is a star-free language, i.e., there is an LTL formula φ_n such that $L(\varphi_n) = L_n$. We can easily adapt the proof of Theorem 21 to obtain a double exponential succinctness gap between PSVA and PLTL.

Corollary 22. *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for any PLTL formula Ξ_n, if $L(\Xi_n) = L_n$ then $\|\Xi_n\| \in \Omega(2^{2^{\|\Phi_n\|}})$.*

Proof. For a given $n > 0$, take the PSVA formula Φ_n from Lemma 19. Suppose that Ξ_n is a PLTL formula that is initially equivalent to Φ_n. Let $\Xi_n' := \mathit{count}_n \wedge \mathsf{G}(\neg c_0 \wedge \cdots \wedge \neg c_{n-1} \rightarrow \Xi_n)$. Observe that we can adapt Lemma 18 so that count_n is an LTL formula. We remark that Ξ_n' expresses that a model is n-counting and each two p-equal n-segments in a model are also q-equal, i.e., $L(\Xi_n') = L_n'$. By

Theorem 12 and Remark 13, there is an NBA $\mathcal{B}$ of size $2^{\mathcal{O}(\|\Xi'_n\|)}$ and $L(\mathcal{B}) = L(\Xi'_n)$. By Lemma 20, we have that $\|\mathcal{B}\| \geq 2^{2^{2^n}}$. It follows that $\|\Xi'_n\| \in \Omega(2^{2^{\|\Phi_n\|}})$. Since Ξ'_n is linear in the size of Ξ_n, we conclude that $\|\Xi_n\| \in \Omega(2^{2^{\|\Phi_n\|}})$. □

Remark 23. We conclude this section by stating some open problems related to the presented succinctness gaps. First, it remains open whether the exponential succinctness gap still holds between RTL and extensions of PSL/SVA with restricted variants of the past operators like the ones discussed in Remark 1. We did not succeeded in proving such a gap, neither did we succeed in expressing the languages L_n concisely in such an extension. Second, it remains open whether the succinctness gaps carry over to a fixed and finite proposition set. Note that the proposition sets P_n over which the PSVA formulas Φ_n are defined grow linearly in n. As shown in [13], we can encode any number of propositions by a single proposition. However, the sizes of the adapted formulas for Φ_n are no longer linear in n. In particular, the sizes of the adapted SEREs $samepos_n$ in Lemma 19 are quadratic in n. It is not obvious how to adapt these SEREs so that their sizes remain linear in n. Therefore, for a fixed and finite proposition set, we only obtain a superpolynomial succinctness gap between PSVA and SVA. Note that for similar reasons, the adapted proof of the succinctness gap between PLTL and LTL in [21, 23] for a fixed and finite proposition set also only shows that PLTL is superpolynomially more succinct than LTL.

6 Conclusion

In this paper, we have proposed the temporal logic RTL, which extends PSL and SVA with past operators. We have analyzed its complexity and our results show that RTL and PSL/SVA are similarly related as PLTL and LTL with respect to expressiveness, succinctness, and the computational complexities of the satisfiability and the model-checking problem. It remains to be seen whether the advantages of RTL over PSL and SVA pay off in practice. The presented translation for RTL into NBAs shows that the additional cost for handling past operators is small and should not be a burden in implementing RTL in system verification. Our preliminary experience with a prototype implementation for the model checker NuSMV are promising.[4]

References

1. IEEE standard for Property Specification Language (PSL). IEEE Std 1850TM, Oct. 2005. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1524461`.
2. IEEE standard for SystemVerilog—unified hardware design, specification, and verification language. IEEE Std 1800TM, Nov. 2005. `http://ieeexplore.ieee.org/xpls/abs_all.jsp?tp=&isnumber=33132&arnumber=1560791`.

[4] See `www.inf.ethz.ch/~daxc/rtl2ba` for the most recent version of our tool.

3. R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M. Y. Vardi, and Y. Zbar. The ForSpec temporal logic: A new temporal property-specification language. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2280 of *Lect. Notes Comput. Sci.*, pages 296–211. Springer-Verlag, 2002.
4. B. Banieqbal and H. Barringer. Temporal logic with fixed points. In *Proceedings of Temporal Logic in Specification 1987*, volume 398 of *Lect. Notes Comput. Sci.*, pages 62–74. Springer-Verlag, 1989.
5. 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.
6. R. Bloem, A. Cimatti, I. Pill, and M. Roveri. Symbolic implementation of alternating automata. *Int. J. Found. Comput. Sci.*, 18(4):727–743, 2007.
7. D. Bustan and J. Havlicek. Some complexity results for SystemVerilog assertions. In *Proceedings of the 18th International Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lect. Notes Comput. Sci.*, pages 205–218. Springer-Verlag, 2006.
8. A. Cimatti, E. M. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification (CAV)*, volume 2404 of *Lect. Notes Comput. Sci.*, pages 359–364. Springer-Verlag, 2002.
9. A. Cimatti, M. Roveri, S. Semprini, and S. Tonetta. From PSL to NBA: a modular symbolic encoding. In *Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 125–133. IEEE Computer Society Press, 2006.
10. A. Cimatti, M. Roveri, and D. Sheridan. Bounded verification of Past LTL. In *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, volume 3312 of *Lect. Notes Comput. Sci.*, pages 245–259. Springer-Verlag, 2004.
11. E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *Form. Method. Syst. Des.*, 10(1):47–71, 1997.
12. C. Dax and F. Klaedtke. Alternation elimination by complementation. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*, volume 5530 of *Lect. Notes Comput. Sci.*, pages 214–229. Springer-Verlag, 2008.
13. S. Demri and P. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Inf. Comput.*, 174(1):84–103, 2002.
14. V. Diekert and P. Gastin. First-order definable languages. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2007.
15. P. Gastin and D. Oddoux. LTL with past and two-way very-weak alternating automata. In *Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2747 of *Lect. Notes Comput. Sci.*, pages 439–448. Springer-Verlag, 2003.
16. D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
17. J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 3rd edition, 2006.

18. O. Kupferman, N. Piterman, and M. Y. Vardi. Extended temporal logic revisited. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR)*, volume 2154 of *Lect. Notes Comput. Sci.*, pages 519–535. Springer-Verlag, 2001.
19. M. Lange. Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR)*, volume 4703 of *Lect. Notes Comput. Sci.*, pages 90–104. Springer-Verlag, 2007.
20. M. Lange. A purely model-theoretic proof of the exponential succinctness gap between CTL^+ and CTL. *Inform. Process. Lett.*, 108(5):308–312, 2008.
21. F. Laroussinie, N. Markey, and P. Schnoebelen. Temporal logic with forgettable past. In *Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS)*, pages 383–392. IEEE Computer Society Press, 2002.
22. O. Lichtenstein, A. Pnueli, and L. D. Zuck. The glory of the past. In *Proceedings of the Conference on Logics of Programs 1985*, volume 193 of *Lect. Notes Comput. Sci.*, pages 196–218. Springer-Verlag, 1985.
23. N. Markey. Temporal logic with past is exponentially more succinct. *Bulletin of the EATCS*, 79:122–128, 2003.
24. S. Miyano and T. Hayashi. Alternating finite automata on ω-words. *Theoret. Comput. Sci.*, 32(3):321–330, 1984.
25. A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57. IEEE Computer Society Press, 1977.
26. A. Pnueli and A. Zaks. PSL model checking and run-time verification via testers. In *Proceedings of the 14th International Symposium on Formal Methods (FM)*, volume 4085 of *Lect. Notes Comput. Sci.*, pages 573–586. Springer-Verlag, 2006.
27. W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *J. Comput. Syst. Sci.*, 4(2):177–192, 1970.
28. M. Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Inform. Process. Lett.*, 30(5):261–264, 1989.
29. M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proceedings of the 8th Banff Higher Order Workshop 1995*, volume 1043 of *Lect. Notes Comput. Sci.*, pages 238–266. Springer-Verlag, 1996.
30. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proceedings of the 1st Symposium on Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society Press, 1986.
31. P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.

A Further Details

In this appendix, we prove Lemma 5 about the accepted langauge of the 2ABA $\mathcal{A}_\varphi$ constructed in Section 4.1.

It suffices to prove that for every word $w \in (2^P)^\omega$, $\psi \in \mathit{Sub}(\varphi)$, and $i \in \mathbb{N}$, it holds that

$$w, i \models \psi \quad \text{iff} \quad \mathcal{A}_\varphi \text{ accepts } \#w \text{ from the configuration } (\psi, i+1)\,.$$

From this equivalence it immediately follows that $L(\mathcal{A}_\varphi) = \{\#w \mid w \in L(\varphi)\}$. Observe that $\mathcal{A}_\varphi$ ensures from its initial state q_I that exactly the letter at position 0 of an input word is # and that $\mathcal{A}_\varphi$ makes a transition from q_I so that it starts scanning the input word from the configuration $(\varphi, 1)$. We prove the above equivalence by induction over the formula structure of ψ. Let $w \in (2^P)^\omega$.

Base Case $\psi = p$, for some $p \in P$. Let $i \in \mathbb{N}$. By definition, $w, i \models p$ is equivalent to $p \in w_i$. By construction, we have that $p \in w_i$ iff $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(p, i+1)$ by reading the letter w_i and moving to the state q_{acc}. The base case for $\psi = \neg p$ is analogous.

Base Case $\psi = \mathsf{cl}(\alpha)$, for some SERE α. Let $i \in \mathbb{N}$. By construction of the NBA $\mathcal{B}_\alpha$, we have that $w, i \models \mathsf{cl}(\alpha)$ iff $\mathcal{B}_\alpha$ accepts $w_{i..}$. By construction of $\mathcal{A}_\varphi$, this is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\mathsf{cl}(\alpha), i+1)$. The base case for $\psi = \neg\mathsf{cl}(\alpha)$ is analogous.

Step Case $\psi = \psi_1 \wedge \psi_2$. Let $i \in \mathbb{N}$. Assume $w, i \models \psi$, i.e, $w, i \models \psi_1$ and $w, i \models \psi_2$. By induction hypothesis, this is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_k, i+1)$, for every $k \in \{1, 2\}$. From the construction of $\mathcal{A}_\varphi$, we conclude that $w, i \models \psi$ iff $\mathcal{A}_\varphi$ accepts $\#w$ from $(\psi, i+1)$. The step case for $\psi = \psi_1 \vee \psi_2$ is analogous.

Step Case $\psi = \mathsf{X}\gamma$. Let $i \in \mathbb{N}$. Assume $w, i \models \mathsf{X}\gamma$, i.e., $w, i+1 \models \gamma$. By induction hypothesis, we obtain the equivalent fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\gamma, i+2)$, which is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\mathsf{X}\gamma, i+1)$ by the construction of $\mathcal{A}_\varphi$. The step case for $\psi = \mathsf{Y}\gamma$ is analogous.

Step Case $\psi = \mathsf{Z}\gamma$. Let $i \in \mathbb{N}$. Assume $w, i \models \mathsf{Z}\gamma$, i.e., $i = 0$ or $i > 0$ and $w, i-1 \models \gamma$. By construction of $\mathcal{A}_\varphi$, the first disjunct is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(q_\#, 0)$. By induction hypothesis, the second disjunct is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\gamma, i-1)$, if $i > 0$. From the construction of $\mathcal{A}_\varphi$, we conclude that $w, i \models \mathsf{Z}\gamma$ iff $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\mathsf{Z}\gamma, i+1)$.

Step Case $\psi = \psi_1 \,\mathsf{U}\, \psi_2$. Let $i \in \mathbb{N}$. Assume $w, i \models \psi_1 \,\mathsf{U}\, \psi_2$, i.e., there is a $k \geq i$ such that $w, k \models \psi_2$ and $w, j \models \psi_1$, for all j with $i \leq j < k$. By induction hypothesis, this is equivalent to the fact that there is a $k \geq i$ such that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_2, k+1)$ and $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1, j+1)$, for all j with $i \leq j < k$. We claim that this is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)$.

We first show the direction from left to right. Assume the left-hand side holds. Then, $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_1 \,\mathsf{U}\, \psi_2, k-1)$ since it accepts

from the configuration (ψ_2, k). It follows that $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_1 \,\mathsf{U}\, \psi_2, k-2)$ since it accepts from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, k-1)$ and from the configuration $(\psi_1, k-1)$ by assumption. Similarly, $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, j+1)$, for all $i \leq j < k$. Thus, the right-hand side holds.

For the other direction, assume that the right-hand side holds. Let r be an accepting run of $\mathcal{A}_\varphi$ on $\#w$ from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)$. For the sake of contradiction, we additionally assume that the left-hand side does not hold, i.e., we have the property $(*)$: there is no $k \geq i$ such that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_2, k+1)$ and $\mathcal{A}_\varphi$ accepts from the configuration $(\psi_1, j+1)$, for all j with $i \leq j < k$. From $(*)$, it follows that $\mathcal{A}_\varphi$ does not accept $\#w$ from the configuration $(\psi_2, i+1)$. By assumption, $\mathcal{A}_\varphi$ accepts from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)$. Hence, by the construction of $\mathcal{A}_\varphi$, it must accept from the configurations $(\psi_1, i+1)$ and $(\psi_1 \,\mathsf{U}\, \psi_2, i+2)$. Again, since $(*)$ holds and $\mathcal{A}_\varphi$ does not accepts from the configuration $(\psi_2, i+1)$, it cannot accept from the configuration $(\psi_2, i+2)$. So, it must accept from the configurations $(\psi_1, i+2)$ and $(\psi_1 \,\mathsf{U}\, \psi_2, i+3)$. If we repeat this argumentation, we obtain the following infinite rejecting path $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)(\psi_1 \,\mathsf{U}\, \psi_2, i+2)(\psi_1 \,\mathsf{U}\, \psi_2, i+3)\ldots$ in the run r of $\mathcal{A}_\varphi$ on $\#w$ from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)$. The existence of such a path is a contradiction to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1 \,\mathsf{U}\, \psi_2, i+1)$ by the run r.

The step case for $\psi = \psi_1 \,\mathsf{S}\, \psi_2$ is analogous.

Step Case $\psi = \psi_1 \,\mathsf{R}\, \psi_2$. Let $i \in \mathbb{N}$. Assume $w, i \models \psi_1 \,\mathsf{R}\, \psi_2$, i.e., for all $k \geq i$, it holds that $w, k \models \psi_2$ or there is a j with $i \leq j < k$ such that $w, j \models \psi_1$. By induction hypothesis, this is equivalent to the fact for all $k \geq i$, it holds that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_2, k+1)$ or there is a j with $i \leq j < k$ such that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1, j+1)$. We claim that this is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from the configuration $(\psi_1 \,\mathsf{R}\, \psi_2, i+1)$.

We first show the direction from left to right. It is easy to see that the left-hand side is equivalent to the following statement: either, (i) $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_2, k+1)$, for all $k \geq i$, or (ii) there is a $k \geq i$ such that $\mathcal{A}_\varphi$ accepts from $(\psi_1, k+1)$ and for all j with $i \leq j \leq k$, we have $\mathcal{A}_\varphi$ accepts from (ψ_2, j). Assume that the first case holds. We consider the run of $\mathcal{A}_\varphi$ from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k+1)$, where $\mathcal{A}_\varphi$ behaves as follows. Whenever $\mathcal{A}_\varphi$ arrives in a configuration $(\psi_1 \,\mathsf{R}\, \psi_2, l)$, for $l \geq k+1$, it moves to configuration (ψ_2, l) and $(\psi_1 \,\mathsf{R}\, \psi_2, l+1)$ respecting the transition function. By assumption, $\mathcal{A}_\varphi$ accepts from every configuration (ψ_2, l), for $l \geq k+1$. Thus, the run of $\mathcal{A}_\varphi$ from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k+1)$ is accepting if the infinite path $(\psi_1 \,\mathsf{R}\, \psi_2, k+1)(\psi_1 \,\mathsf{R}\, \psi_2, k+2)\ldots$ is accepting, as well. This path is accepting since $\psi_1 \,\mathsf{R}\, \psi_2$ is an accepting state of $\mathcal{A}_\varphi$. So, $\mathcal{A}_\varphi$ accepts $\#w$ from $(\psi_1 \,\mathsf{R}\, \psi_2, i+1)$. Assume that the second case holds. Let $k \geq i$ be a position such that $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_1, k+1)$ and for all j with $i \leq j \leq k$, $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_2, j+1)$. Since $\mathcal{A}_\varphi$ accepts from $(\psi_2, k+1)$ and from (ψ_1, k), it follows that by definition of the transition function, $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. Again, by assumption and the previous step, $\mathcal{A}_\varphi$ accepts from $(\psi_2, k-1)$ and from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. Thus, by definition of the transition function, $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k-1)$. Iteration

this argumentation, we conclude that for all j with $i \leq j \leq k$, it holds that $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, j+1)$. Thus, $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, i+1)$.

Now, we show the other direction. Assume that the right-hand side holds, i.e., $\mathcal{A}_\varphi$ accepts from the configuration $(\psi_1 \,\mathsf{R}\, \psi_2, i+1)$. For the sake of contradiction, we additionally assume that the left-hand side does not hold, i.e., we have that there is a $k \geq i$ such that $\mathcal{A}_\varphi$ does not accept from $(\psi_2, k+1)$ and for all j with $i \leq j < k$ we have $\mathcal{A}_\varphi$ does not accept $(\psi_1, j+1)$. We refer to these assumptions by the first and second assumption, respectively. Let $k \geq i$ be the least number such that the second assumption holds. In particular, we have $\mathcal{A}_\varphi$ does not accept from (ψ_2, k). For the sake of contradiction, we show that $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. By the first assumption we have that $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, i+1)$. Hence, by the definition of the transition function and acceptance definition of a run, $\mathcal{A}_\varphi$ also accepts from $(\psi_2, i+1)$ and either from $(\psi_1, i+1)$ or $(\psi_1 \,\mathsf{R}\, \psi_2, i+2)$. From the second assumption, it follows that $\mathcal{A}_\varphi$ does not accept from $(\psi_1, i+1)$. Therefore, $\mathcal{A}_\varphi$ accepts from $(\psi_2, i+1)$ and from $(\psi_1 \,\mathsf{R}\, \psi_2, i+2)$. Repeating this argumentation, we can show that for all j with $i \leq j < k$ we have $\mathcal{A}_\varphi$ accepts from (ψ_2, j) and from $(\psi_1 \,\mathsf{R}\, \psi_2, j+1)$. Thus, $\mathcal{A}_\varphi$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$ contradicting the choice of k.

The step case for $\psi = \psi_1 \,\mathsf{T}\, \psi_2$ is analogous.

Step Case $\psi = \alpha \diamond\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume $w, i \models \psi$, i.e, there is a position $k \geq i$ such that $w_{i..k} \in L(\alpha)$ and $w, k \models \gamma$. By induction hypothesis, this is equivalent to the fact that there is $k \geq i$ such that $w_{i..k} \in L(\alpha)$ and $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\gamma, k+1)$. That is, $\mathcal{A}_\varphi$ accepts from configuration $(\alpha \diamond\!\!\rightarrow \gamma, i+1)$ iff there is a position k such that $\mathcal{A}_\alpha$ has an accepting run on $\#w_{i+1..k+1}$ and $\mathcal{A}_\varphi$ accepts from $(\gamma, k+1)$. It is easy to see that by definition of the transition function, this is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\alpha \diamond\!\!\rightarrow \gamma, i+1)$.

The step case for $\psi = \alpha \diamond\!\!\!\!\!-\!\!\rightarrow \gamma$ is analogous.

Step Case $\psi = \alpha \,\Box\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume $w, i \models \psi$, i.e, for all positions $k \geq i$ such that $w_{i..k} \in L(\alpha)$, it holds that $w, k \models \gamma$. By induction hypothesis, this is equivalent to the fact that for all positions $k \geq i$ such that $w_{i..k} \in L(\alpha)$, it holds that $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\gamma, k+1)$. This is equivalent to the fact that there exists a run of $\mathcal{A}_\varphi$ on $\#w$ from the configuration $(\alpha \,\Box\!\!\rightarrow \gamma, i+1)$ such that for every path in the run labeled by $(q_0, i+1)(q_1, i+2)\ldots$ the following holds: for all $j \in \mathbb{N}$ such that $(q_0, i+1)\ldots(q_j, i+1+j)$ is an accepting run of $\mathcal{A}_\alpha$ on $w_{i..j}$, the automaton $\mathcal{A}_\varphi$ accepts $\#w$ from $(q_j, i+1+j)$. That is equivalent to the fact that $\mathcal{A}_\varphi$ accepts $\#w$ from configuration $(\alpha \,\Box\!\!\rightarrow \gamma, i+1)$.

The step case $\psi = \alpha \,\boxminus\!\!\rightarrow \gamma$ is analogous.