On regular temporal logics with past

Acta Informatica (2010) 47:251–277
DOI 10.1007/s00236-010-0118-3

ORIGINAL ARTICLE

On regular temporal logics with past

Christian Dax · Felix Klaedtke · Martin Lange

Received: 12 August 2009 / Accepted: 8 April 2010 / Published online: 15 May 2010

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 article, we extend the common core of these specification languages with past operators. We name this extension PPSL. Although all ω-regular properties are expressible in PSL, SVA, and PPSL, past operators often allow one to specify properties more naturally and concisely. In fact, we show that PPSL is exponentially more succinct than the cores of PSL and SVA. On the star-free properties, PPSL is double exponentially more succinct than LTL. Furthermore, we present a translation of PPSL into language-equivalent nondeterministic Büchi automata, which is based on novel constructions for 2-way alternating automata. The upper bound on the size of the resulting nondeterministic Büchi automata obtained by our translation is almost the same as the upper bound for the nondeterministic Büchi automata obtained from existing translations for PSL and SVA. Consequently, the satisfiability problem and the model-checking problem for PPSL fall into the same complexity classes as the corresponding problems for PSL and SVA.

Keywords Linear-time Temporal Logics · Finite-state Model Checking · Alternating Automata · Regular/ω-regular Languages

C. Dax
Computer Science Department, ETH Zurich, CAB F 58.2, Universitätstr. 6, 8092 Zurich, Switzerland
e-mail: christian.dax@inf.ethz.ch

F. Klaedtke (✉)
Computer Science Department, ETH Zurich, CNB F 107.1, Universitätstr. 6, 8092 Zurich, Switzerland
e-mail: felix.klaedtke@inf.ethz.ch

M. Lange
Department of Electrical Engineering and Computer Science, University of Kassel,
Wilhelmshöher Allee 73, 34121 Kassel, Germany
e-mail: martin.lange@ifi.lmu.de

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.[1] The prominence of PSL and SVA in industry over other specification languages like LTL [26], μLTL [4], and ETL [34] is that PSL and SVA balance well the competing needs 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 (for example, 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 [9,10,19,23,24,29], 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 [33], 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 [19], 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 [25] an NBA with only $2^{\mathcal{O}(n)}$ states. Nevertheless, in this article, 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 article is as follows. We first propose an extension of PSL with past operators, which we name PPSL. PPSL 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. PPSL additionally contains the corresponding counterpart $\alpha \diamond\!\!\!\!\rightarrow \varphi$. This new operator 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]. See [17], for a linear-time variant of propositional dynamic logic. However, PSL/SVA uses SEREs over state predicates and in dynamic logic, the expressions are over program statements.

PSL, SVA, and PPSL have the same expressive power: they all describe the class of ω-regular languages. However, PPSL allows one to describe ω-regular languages more con-

[1] For the ease of exposition, we identify, similar to [5,7,9,27], 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.

cisely than PSL and SVA. To show this, we establish a lower bound on the succinctness of PPSL and PSL. We define a family of ω-regular languages and prove that these languages can be described in PPSL exponentially more succinctly than in PSL. For the LTL-expressible properties, that is, the ω-regular languages that are star-free (see, for example, [14]), we obtain as a byproduct that PPSL is double exponentially more succinct than LTL.

Furthermore, we investigate the additional computational cost for solving the satisfiability problem and the model-checking problem for PPSL. As for PSL and SVA, these problems are EXPSPACE-complete for PPSL. 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 $\mathcal{O}(3^{2^{2n}})$ [5,7]. We present a construction for PPSL that translates a PPSL formula of size n and m propositional variables into an NBA of size $\mathcal{O}(2^m \cdot 3^{2^{2n}})$. Since $m \leq n$ and hence $2^m \cdot 3^{2^{2n}} \leq 3^{0.631n+2^{2n}}$, the difference between these upper bounds of the sizes of the resulting automata for PSL/SVA and PPSL is surprisingly small. 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 article for the alternating automata that we obtain from our translation of PPSL formulas into alternating automata. We use this construction to translate a given PPSL formula into an initially equivalent SVA formula. The size of the resulting formula is quadruple exponentially larger, not quite matching the lower bound mentioned above. One of these four exponentials is due to the fact that the resulting SVA formula only contains semi-extended regular expressions without intersection operators.

We point out that our translation for PPSL 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 [19] 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 PPSL 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 article is organized as follows. In Sect. 2, we give preliminaries. In Sect. 3, we define PPSL and its fragments LTL, PSL, and SVA. In Sect. 4, we present the translation of PPSL formulas into language-equivalent NBAs and in Sect. 5, we draw some consequences from this translation. In Sect. 6, we show the succinctness gap between PPSL and PSL. Finally, in Sect. 7, we draw conclusions. The appendix contains additional proof details.

2 Preliminaries

We assume that the reader is familiar with automata theory over finite and infinite words. In the following, we recapitulate the needed background in this area and fix the notation and terminology that we use in the remainder of the text.

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 $w_i w_{i+1} \ldots$ and $w_{i..j}$ for the subword $w_i w_{i+1} \ldots w_j$.

The concatenation of the languages $L \subseteq \Sigma^*$ and $L' \subseteq \Sigma^*$ is $L ; L' := \{uv \mid u \in L \text{ and } v \in L'\}$ and the fusion is $L : L' := \{ubv \mid ub \in L \text{ and } bv \in L' \text{ with } b \in \Sigma\}$. Moreover, we define $L^* := \bigcup_{n\in\mathbb{N}} L^n(\beta)$, where $L^0 := \{\varepsilon\}$ and $L^{i+1} := L ; L^i$, for all $i \in \mathbb{N}$.

A (Σ-labeled) *tree* is a function $t : T \to \Sigma$, where $T \subseteq \mathbb{N}^*$ satisfies the conditions: (i) T is prefix-closed (this means, 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 $t(\varepsilon)t(\pi_0)t(\pi_0\pi_1)\ldots \in \Sigma^\omega$.

Propositional logic We denote the set of *Boolean formulas* over the set P of propositions by $\mathcal{B}(P)$, this means, $\mathcal{B}(P)$ 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 $P \backslash M$. We write $\mathcal{B}^+(P)$ for the set of *positive Boolean formulas* over P, this means, the set of 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 α over the proposition set P is inductively defined:

$$L(\alpha) := \begin{cases} \{\varepsilon\} & \text{if} \quad \alpha = \varepsilon, \\ \{w \in (2^P)^* \mid |w| = 1 \text{ and } w_0 \models \alpha\} & \text{if} \quad \alpha \in \mathcal{B}(P), \\ L(\beta) \star L(\gamma) & \text{if} \quad \alpha = \beta \star \gamma, \text{ where } \star \in \{\cup, \cap, ;, :\}, \text{ and} \\ (L(\beta))^* & \text{if} \quad \alpha = \beta^*. \end{cases}$$

The *size* of an SERE is defined as $\|\varepsilon\| := 1$, $\|b\| := 1$, for $b \in \mathcal{B}(P)$, $\|\beta \star \gamma\| := 1 + \|\beta\| + \|\gamma\|$, for $\star \in \{\cup, \cap, ;, :\}$, and $\|\beta^*\| := 1 + \|\beta\|$. Moreover, $Is(\alpha)$ is the number of intersection operators that occur in the SERE α. A *regular expression* (RE) α is an SERE with $Is(\alpha) = 0$.

Automata In the following, we define 2-way alternating Büchi automata, which scan input words letter by letter with their read-only head. The meaning of "2-way" and "alternating" is best illustrated by the example transition $\delta(p, a) = (q, -1) \vee ((r, 0) \wedge (s, 1))$ of such an automaton, where p, q, r, s are states, a is a letter of the input alphabet, and δ is the automaton's transition function. The second coordinate of the tuples $(q, -1)$, $(r, 0)$, $(s, 1)$ specifies in which direction the read-only head moves: -1 for left, 0 for not moving, and 1 for right. The transition above can be read as follows. When reading the letter a in state p, the automaton has two choices: (i) It goes to state q and moves the read-only head to the left. In this case, the automaton proceeds scanning the input word from the updated state and position. (ii) Alternatively, it can branch its computation by going to state r *and* to state s, where the read-only head is duplicated: the first copy proceeds scanning the input word from the state r, where the position of the read-only head is not altered; the second copy proceeds scanning the input word from the state s, where the read-only head is moved to the right. Note that the choices (i) and (ii) are given by the models of the example transition $\delta(p, a)$, which is a positive Boolean formula with propositions that are pairs of states and movements of the read-only head.

Let $\mathbb{D} := \{-1, 0, 1\}$ be the set of directions in which the read-only head can move. Formally, a *2-way alternating 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 set of accepting states. 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

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

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 π 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 a UBA as a function of the form $\delta : Q \times \Sigma \to 2^Q$. This means that clauses and monomials are written as sets. A 1-way automaton is *deterministic* if its transition function δ is nondeterministic and universal, that means, $\delta(q, a) \in Q$, for all states q and input letters a.

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. A *run* of the NFA $\mathcal{B} = (Q, \Sigma, \delta, q_I, F)$ on a finite word $w \in \Sigma^*$ is a sequence of $|w| + 1$ states $q_0 q_1 \ldots q_{|w|}$ 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 PPSL. The cores of the two industrial-standard property specification languages PSL [1] and SVA [2] are fragments of PPSL. The syntax of PPSL over the set P of propositions is given by the grammar

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

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

$$\begin{array}{lll}
w, i \models p & \text{iff} & p \in w_i \\
w, i \models \mathsf{cl}(\alpha) & \text{iff} & \exists k \geq i : w_{i..k} \in L(\alpha), \text{ or } \forall k \geq i : \exists v \in L(\alpha) : w_{i..k} \preceq v \\
w, i \models \varphi \wedge \psi & \text{iff} & w, i \models \varphi \text{ and } w, i \models \psi \\
w, i \models \neg\varphi & \text{iff} & w, i \not\models \varphi \\
w, i \models \mathsf{X}\varphi & \text{iff} & w, i + 1 \models \varphi \\
w, i \models \varphi \,\mathsf{U}\, \psi & \text{iff} & \exists k \geq i : w, k \models \psi \text{ and } \forall j : \text{if } i \leq j < k \text{ then } w, j \models \varphi \\
w, i \models \alpha \diamond\!\!\rightarrow \varphi & \text{iff} & \exists k \geq i : w_{i..k} \in L(\alpha) \text{ and } w, k \models \varphi
\end{array}$$

$$
\begin{array}{lll}
w, i \models \mathsf{Y}\varphi & \text{iff} & i > 0 \text{ and } w, i-1 \models \varphi \\
w, i \models \varphi \,\mathsf{S}\, \psi & \text{iff} & \exists k \leq i : w, k \models \psi \text{ and } \forall j : \text{if } k < j \leq i \text{ then } w, j \models \varphi \\
w, i \models \alpha \mathrel{\diamondsuit\!\!\rightarrow} \varphi & \text{iff} & \exists k \leq i : w_{k..i} \in L(\alpha) \text{ and } w, k \models \varphi
\end{array}
$$

A word $w \in (2^P)^\omega$ is a *model* of a PPSL formula φ if $w, 0 \models \varphi$. The *language* of a PPSL formula φ is $L(\varphi) := \{w \in (2^P)^\omega \mid w, 0 \models \varphi\}$. The PPSL 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 $w \in (2^P)^\omega$. For instance, let tt and ff be the usual abbreviations for the Boolean constants. Then Ytt and ff are initially equivalent since the former cannot be satisfied in state 0 of a model. However, they are not logically equivalent, since the former holds in any state $i > 0$ of any model whereas the latter holds in none. Clearly, logical equivalence implies initial equivalence and not vice versa. However, for formulas that do not use the past operators Y, S, $\mathrel{\diamondsuit\!\!\rightarrow}$ these two equivalences coincide.

As for SEREs, we define the *size* $\|\varphi\|$ of a PPSL formula φ as its syntactic length. That is, $\|p\| := 1$, $\|\mathsf{cl}(\alpha)\| := 1 + \|\alpha\|$, $\|\neg\varphi\| := \|\mathsf{X}\varphi\| := \|\mathsf{Y}\varphi\| := 1 + \|\varphi\|$, $\|\varphi \wedge \psi\| := \|\varphi \,\mathsf{U}\, \psi\| := \|\varphi \,\mathsf{S}\, \psi\| := 1 + \|\varphi\| + \|\psi\|$, and $\|\alpha \mathrel{\diamondsuit\!\!\rightarrow} \varphi\| := \|\alpha \mathrel{\diamondsuit\!\!\rightarrow} \varphi\| := 1 + \|\alpha\| + \|\varphi\|$.

We define the following fragments of PPSL. We call a PPSL formula a *PSL formula* if it does not contain the operators Y, S, and $\mathrel{\diamondsuit\!\!\rightarrow}$. An *LTL formula* is a PSL formula that does not contain the operators cl and $\mathrel{\diamond\!\!\rightarrow}$. 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.

We use standard syntactic sugar, like the Boolean constants and connectives ff, tt, $\vee$, $\rightarrow$, and we define $\varphi \,\mathsf{R}\, \psi := \neg(\neg\varphi \,\mathsf{U}\, \neg\psi)$, $\varphi \,\mathsf{T}\, \psi := \neg(\neg\varphi \,\mathsf{S}\, \neg\psi)$, $\mathsf{Z}\varphi := \mathsf{Ytt} \rightarrow \mathsf{Y}\varphi$. Moreover, for a PPSL formula φ and an SERE α, we write $\alpha \mathrel{\square\!\!\rightarrow} \varphi$ for $\neg(\alpha \mathrel{\diamondsuit\!\!\rightarrow} \neg\varphi)$ and $\alpha \mathrel{\boxminus\!\!\rightarrow} \varphi$ for $\neg(\alpha \mathrel{\diamondsuit\!\!\rightarrow} \neg\varphi)$. Note that the standard unary temporal operators can easily be defined in the respective fragment. For instance, in PLTL, we define the future operators *finally* $\mathsf{F}\varphi := \mathsf{tt}\,\mathsf{U}\varphi$ and *generally* $\mathsf{G} := \neg\mathsf{F}\neg\varphi$. Their past counterparts are *once* $\mathsf{O}\varphi := \mathsf{tt} \,\mathsf{S}\, \varphi$ and *historically* $\mathsf{H}\varphi := \neg\mathsf{O}\neg\varphi$, respectively. In PSVA, we define $\mathsf{G}\varphi := \mathsf{tt}^* \mathrel{\square\!\!\rightarrow} \varphi$, $\mathsf{F}\varphi := \mathsf{tt}^* \mathrel{\diamond\!\!\rightarrow} \varphi$, $\mathsf{H}\varphi := \mathsf{tt}^* \mathrel{\boxminus\!\!\rightarrow} \varphi$, and $\mathsf{O}\varphi := \mathsf{tt}^* \mathrel{\diamondsuit\!\!\rightarrow} \varphi$. Furthermore, in PSVA, we use $\mathsf{X}\varphi$ as syntactic sugar for $\mathsf{tt} \,;\, \mathsf{tt} \mathrel{\diamond\!\!\rightarrow} \varphi$ and $\mathsf{Y}\varphi$ for $\mathsf{tt} \,;\, \mathsf{tt} \mathrel{\diamondsuit\!\!\rightarrow} \varphi$.

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 $\mathrel{\diamondsuit\!\!\rightarrow}$. For instance, in PPSL, if $\varepsilon \notin L(\alpha)$, $\mathsf{ended}(\alpha)$ is syntactic sugar for $\alpha \mathrel{\diamondsuit\!\!\rightarrow} \mathsf{tt}$, and tt otherwise. Observe that ended and prev can only be applied to SEREs, and, in contrast to $\mathrel{\diamondsuit\!\!\rightarrow}$, it is not possible to define the classical past operators Y, H, and O with them. We also remark that the literature, for example, [5,7,9,20,27] usually considers the essential core of the PSL standard to which the operators ended and prev do not belong. We follow this convention, this means, the formulas in our fragment PSL of PPSL 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 Sect. 4 for PPSL.

Example 2 A standard example for showing that the past operators of PLTL can lead to more intuitive specifications is $\mathsf{G}(grant \rightarrow \mathsf{O}request)$, this means, every grant is preceded by a request [23]. An initially equivalent LTL formula is $request \,\mathsf{R}\, (\neg grant \vee request)$. 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, for example, a request consists of a *start* event that is later followed by an *end* event and no *cancel* event happens between the *start* and the *end*

event. Such sequences are naturally described by the SERE $(start\,;\mathsf{tt}^*\,;end)\cap(\neg cancel)^*$. Using this SERE and the new past operator $\Diamond\!\!\!\!-\!\!\!\!\rightarrow$, we can easily express in PPSL the property that every grant is preceded by a request:

$$\mathsf{G}\left(grant \rightarrow \left(((start\,;\mathsf{tt}^*\,;end)\cap(\neg cancel)^*)\,;\mathsf{tt}^* \Diamond\!\!\!\!-\!\!\!\!\rightarrow \mathsf{tt}\right)\right). \tag{1}$$

Note that according to the semantics of the operator $\Diamond\!\!\!\!-\!\!\!\!\rightarrow$, 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

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

Although debatable, we consider that the PPSL formula (1) is easier to understand than the PLTL formula (2). In SVA, we can express the property as $norequest \mathrel{\square\!\!\!\!-\!\!\!\!\rightarrow} \neg grant$, where the SERE *norequest* describes the complement of the language $L\big(\mathsf{tt}^*\,;((start\,;\mathsf{tt}^*\,;end)\cap(\neg cancel)^*)\,;\mathsf{tt}^*\big)$, that is, *norequest* is the SERE

$$\big((\neg start)\cup(start\wedge cancel)\cup(start;(\neg end)^*;cancel)\big)^*;(\varepsilon\cup(start\wedge end));(\neg end)^*.$$

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 $\Diamond\!\!\!\!-\!\!\!\!\rightarrow$. For $N \geq 1$ and $i \in \{0,\ldots,N-1\}$, consider the PPSL formula $\Phi_{N,i} := \mathsf{G}\,(send_i \rightarrow (switch_i \cap (init\,;(\neg init)^*) \Diamond\!\!\!\!-\!\!\!\!\rightarrow \mathsf{tt}))$, where $switch_i$ counts the number of *switch* events modulo N, this means,

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

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 k *switch* events with $k \equiv i \mod N$ occurred previously since the last *init* event. Note that this property is not expressible in LTL since it is not star-free (see, for example, [14]).

The negation of the PSL formula

$$((\neg init)^* \Diamond\!\!\rightarrow send_i) \vee \mathsf{F}\left(init \wedge \left((\mathsf{tt}\,;(\neg init)^*) \cap (\textstyle\bigcup_{j\neq i} switch_j) \Diamond\!\!\rightarrow send_i\right)\right) \tag{4}$$

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 Sect. 6, we prove that PPSL is exponentially more succinct than PSL.

In general, for writing specifications, PPSL 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 PPSL to nondeterministic automata

In this section, we present a translation from PPSL formulas into language-equivalent NBAs. Similar to the well-known translation for LTL formulas into NBAs, our translation comprises

two steps: for a given PPSL 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 PPSL to eventually and locally 1-way 2ABAs

In this subsection, we assume that φ is a PPSL formula over P and φ is in *negation normal form*, this means, the negation symbol $\neg$ only occurs directly in front of the atomic subformulas of φ. Note that every PPSL formula ψ can be rewritten into a logically equivalent PPSL 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 PPSL formula φ, we briefly highlight the similarities and the differences to the standard constructions for LTL, PLTL, SVA, and PSL [5,7,15,32]. 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 PPSL efficiently. As the standard construction for PSL [5], the state space of the 2ABA $\mathcal{A}_\varphi$ consists of the subformulas of the given PPSL formula and the states of the automata for the SEREs. We introduce an auxiliary 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. Some additional states are needed for such a check. The new operators $\boxminus\!\!\rightarrow$ and $\diamondsuit\!\!\rightarrow$ are then easily handled since $\mathcal{A}_\varphi$ is alternating and 2-way. In Sect. 4.2, we eliminate this additional symbol # when constructing from $\mathcal{A}_\varphi$ the NBA for the PPSL formula φ.

4.1.1 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 [18].

Lemma 4 *Let α be an 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 PPSL formula φ is defined as follows, where Γ is the alphabet $\{\#\} \cup 2^P$. 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 to that in the standard constructions which 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

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

The set of accepting states F is the set of states that are neither in the sets G_1, G_2, nor G_3, which are as follows. The set G_1 is the singleton $\{q_{rej}\}$. Similar to the standard constructions for translating LTL formulas to alternating 1-way automata, the set $G_2 := \{\psi \,\mathsf{U}\, \psi' \mid \psi \,\mathsf{U}\, \psi' \in \mathit{Sub}(\varphi)\}$ takes care of the least-fixpoint subformulas $\psi \,\mathsf{U}\, \psi' \in \mathit{Sub}(\varphi)$, that is, $\mathcal{A}_\varphi$ avoids infinite regeneration of these subformulas. The states in

$$\begin{aligned} G_3 := {} & \{\mathsf{cl}(s) \mid \mathsf{cl}(\alpha) \in \mathit{Sub}(\varphi) \text{ and } s \text{ is a rejecting state of } \mathcal{B}_\alpha\} \cup \\ & \{\neg\mathsf{cl}(s) \mid \neg\mathsf{cl}(\alpha) \in \mathit{Sub}(\varphi) \text{ and } s \text{ is a rejecting state of } \mathcal{B}'_\alpha\} \cup \\ & \{s \diamondsuit\!\!\rightarrow \psi \mid \alpha \diamondsuit\!\!\rightarrow \psi \in \mathit{Sub}(\varphi) \text{ and } s \text{ is a state of } \mathcal{A}_\alpha\} \end{aligned}$$

are obtained from the NBAs and UBAs that accept the languages of the formulas $\mathsf{cl}(\alpha)$, $\neg\mathsf{cl}(\alpha) \in \mathit{Sub}(\varphi)$, respectively, and from the states of the NFAs that correspond to the subformulas of the form $\alpha \diamondsuit\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$.

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

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

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 #, this means, we define $\delta(q, \#) := (q_{rej}, 1)$. For the remainder of the construction, let $a \in 2^P$.

The following definitions are similar to the standard constructions 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

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

– 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

$$\begin{aligned}\delta(\psi\ \mathsf{U}\ \psi', a) &:= (\psi', 0) \vee \big((\psi, 0) \wedge (\psi\ \mathsf{U}\ \psi', 1)\big)\ ,\\ \delta(\psi\ \mathsf{R}\ \psi', a) &:= (\psi', 0) \wedge \big((\psi, 0) \vee (\psi\ \mathsf{R}\ \psi', 1)\big)\ ,\\ \delta(\psi\ \mathsf{S}\ \psi', a) &:= (\psi', 0) \vee \big((\psi, 0) \wedge (\psi\ \mathsf{S}\ \psi', -1)\big)\ ,\end{aligned}$$

and

$$\delta(\psi\ \mathsf{T}\ \psi', a) := (\psi', 0) \wedge \big((\psi, 0) \vee (\psi\ \mathsf{T}\ \psi', -1) \vee (q_\#, -1)\big)\,.$$

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 $\alpha \Diamond\!\!\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$ 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 $\alpha \Box\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$, 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 $\delta(\alpha \Box\!\!\rightarrow \psi, a) := (s_I \Box\!\!\rightarrow \psi, 0)$ and for $s \in S$,

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

The transitions for a subformula $\alpha \boxminus\!\!\rightarrow \psi \in \mathit{Sub}(\varphi)$ are defined similarly. 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 (this means, the transitions of $\mathcal{A}_\varphi$ in which the read-only head does not move) can easily be eliminated by replacing a proposition $(s, 0)$ that occurs in $\delta(q, b)$ with $\delta(s, b)$, where $q, s \in Q$ and $b \in \Gamma$. Such a replacement does not alter $\mathcal{A}_\varphi$'s language, since the states in the configurations that $\mathcal{A}_\varphi$ visits by ε-transitions in a path of a run are not essential whether the run is accepting or not.

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 $4 + 2^{\|\varphi\|}$.*

4.1.2 Additional properties of the construction

The 2ABA $\mathcal{A}_\varphi$ has some additional properties, which we exploit in Sect. 4.2 for constructing the NBA. Namely, $\mathcal{A}_\varphi$ is eventually 1-way and locally 1-way.

Eventually 1-way automata Intuitively speaking, eventually 1-way means that on every branch of a computation, the alternating automaton will eventually move its read-only head only forward. For the formal definition, we need the definition of a minimal run. The set $M \subseteq P$ is a *minimal model* of the positive Boolean formula $b \in \mathcal{B}^+(P)$ over the proposition set P if $M \models b$ and there is no $p \in M$ such that $M \backslash \{p\} \models b$. Let $\mathcal{B} = (S, \Sigma, \eta, s_I, E)$ be a 2ABA. A run $r : T \to Q \times \mathbb{N}$ of $\mathcal{B}$ on the word $w \in \Sigma^\omega$ is *minimal* if for every node $x \in T$ with $r(x) = (s, j)$, the set $\{(s, j' - j) \mid r(y) = (s', j')$, where y is a child of x in $r\}$ is a minimal model of $\eta(s, w_j)$. The 2ABA $\mathcal{B}$ is *eventually 1-way* if for every minimal run $r : T \to Q \times \mathbb{N}$ of $\mathcal{B}$ and every path $(q_0, h_0)(q_1, h_1) \ldots \in (Q \times \mathbb{N})^\omega$ in r, there is an integer $n \in \mathbb{N}$ such that for every $i \geq n$, we have $h_i < h_{i+1}$. That is, after position n, the read-only head only moves forward.

Lemma 7 *The 2ABA $\mathcal{A}_\varphi$ is eventually 1-way.*

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

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

Note that $weight(\alpha \bigstar\!\!\rightarrow \psi) > weight(s \bigstar\!\!\rightarrow \psi) > weight(\psi)$, since by definition $weight(\alpha \bigstar\!\!\rightarrow \psi) = 2|Sub(\alpha \bigstar\!\!\rightarrow \psi)| = 2|Sub(\psi)| + 2$, $weight(s \bigstar\!\!\rightarrow \psi) = 2|Sub(\psi)| + 1$, and $weight(\psi) = 2|Sub(\psi)|$. Let $r : T \to Q \times \mathbb{N}$ be a minimal run of $\mathcal{A}_\varphi$ on a word $w \in \Gamma^\omega$ and $(q_0, h_0)(q_1, h_1) \ldots \in (Q \times \mathbb{N})^\omega$ a path in r. By using the *weight* function, we now show that there is an integer $n \in \mathbb{N}$ such that for all $i \geq n$, we have $h_i < h_{i+1}$.

Observe that by the definition of the transition function, we have $weight(q') \leq weight(q)$, for all $q, q' \in Q$ whenever the proposition (q', d) occurs in $\delta(q, a)$, where $a \in \Gamma$ and $d \in \mathbb{D}$. Hence, for all positions $i, j \in \mathbb{N}$ with $i < j$, we have $weight(q_i) \geq weight(q_j)$. That is, the weights of the states of the configurations monotonically decrease along the path. Let $n \in \mathbb{N}$ be a position such that the weight becomes constant, that is, for all $i \geq n$, we have

$weight(q_i) = weight(q_n)$. We show that $h_i < h_{i+1}$, for all $i \geq n$ by considering the following cases.

Case $q_i \in Q_1$. Since from the initial state q_I we can only reach states whose weights are strictly smaller, q_i cannot be q_I. Assume that $q_i \in \{q_\#, q_{acc}, q_{rej}\}$. By the definition of $\mathcal{A}_\varphi$'s transition function, $q_{i+1} \in \{q_{acc}, q_{rej}\}$ and the position of the read-only head position increases by 1. Therefore, $h_i < h_{i+1}$.

Case $q_i \in Q_2$. Assume that q_i is a state of the form $\psi \,\mathsf{U}\, \psi'$ or $\psi \,\mathsf{R}\, \psi'$, for $\psi, \psi' \in Sub(\varphi)$. Since $weight(q_{i+1}) = weight(q_i)$, only the transition that stays in the same state and moves the read-only head forward is possible. Hence, $h_i < h_{i+1}$.

Assume that q_i is a state of the form $\psi \,\mathsf{S}\, \psi'$ or $\psi \,\mathsf{T}\, \psi'$, for $\psi, \psi' \in Sub(\varphi)$. Since $weight(q_j) = weight(q_i)$, from every configuration (q_j, h_j) with $j \geq i$, only the transition that stays in the same state and moves the read-only head backward is possible. That means, the second component decreases and eventually becomes negative, which is impossible.

Case $q_i \in Q_3$. The state q_{i+1} must be in Q_3, since otherwise the weight would decrease. Note that in the states Q_3, $\mathcal{A}_\varphi$ simulates a run of a 1-way automaton. If $\mathcal{A}_\varphi$ simulates a run of a 1-way automaton that always moves forward then clearly $h_i < h_{i+1}$. If $\mathcal{A}_\varphi$ simulates a run of a 1-way automaton in the reverse direction, that is, $\mathcal{A}_\varphi$ always moves backward then the position of the read-only head eventually becomes negative, which is impossible. □

Locally 1-way automata 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 $b \in \Sigma$. That is, the directions are limited in which $\mathcal{B}$ can move its read-only head while scanning the input word. Namely, from a configuration, the read-only head cannot move forward and backward when reading a letter. 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 PPSL does not have a temporal operator that simultaneously refers to the past and to the future, and hence, the constructed 2ABA $\mathcal{A}_\varphi$ is already locally 1-way.

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

Proof The claim directly follows from inspecting $\mathcal{A}_\varphi$'s transition function. □

4.2 From eventually and locally 1-way 2ABAs to NBAs

In the following, we show how the alternating automaton from the previous subsection for a PPSL formula in negation normal form can be translated into an NBA. Our construction is based on an alternation-elimination construction for so-called loop-free 2-way alternating

Büchi automata from [12]. Intuitively, loop freeness [12,15] means that on every branch of a computation of an 2-way alternating automaton, no configuration occurs twice. An automaton that is eventually 1-way is also loop-free. However, the converse does not hold in general.

The alternation-elimination construction presented in this subsection exploits the fact that the given alternating automaton is eventually 1-way and locally 1-way. Overall, for a PPSL formula ψ with m proposition, the resulting language-equivalent NBA has size $\mathcal{O}(2^m \cdot 3^{2^{2\|\psi\|}})$, which is included in $\mathcal{O}(3^{0.631m+2^{2\|\psi\|}})$. With the construction in [12], we would obtain an NBA of size $\mathcal{O}(3^{2\cdot 2^{2\|\psi\|}})$. Note that $m \leq \|\psi\|$. 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 an eventually 1-way and locally 1-way 2ABA $\mathcal{A}$, there is a language-equivalent NBA $\mathcal{B}$ of size $\mathcal{O}(|\Sigma| \cdot 3^{\|\mathcal{A}\|})$, where Σ is the alphabet of $\mathcal{A}$.*

Before we present the details of the automata construction 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 [12,31], $\mathcal{B}$ 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 $(q_I, 0) \ldots (q, i) \ldots (q', i) \ldots$. We can locally check whether such a sequence $R_0 R_1 \ldots$ represents a run of $\mathcal{A}$ on w. Since $\mathcal{A}$ is locally 1-way, $\mathcal{B}$ can do this as follows. It 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 infinitely often configurations with an accepting state. Similar to the alternation-elimination construction by Miyano and Hayashi [25] 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. Since $\mathcal{A}$ is eventually 1-way, $\mathcal{B}$ must only track transitions that move the position of $\mathcal{A}$'s read-only head forward.

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 a PPSL formula φ in negation normal form.

Proof (Theorem 10) We now give the details of the construction with this weaker condition from Remark 11 and then prove the correctness of the construction. For the eventually 1-way and locally 1-way 2ABA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$, we define the NBA $\mathcal{B} := (Q', \Sigma, \eta, p_I, E)$ with $Q' := \{p_I\} \cup \{(a, R, S) \in \Sigma \times 2^Q \times 2^{Q \setminus F} \mid S \subseteq R\}$ and $E := \Sigma \times 2^Q \times \{\emptyset\}$. Note that $|Q'| \in \mathcal{O}(|\Sigma| \cdot 3^{|Q|})$ since we require that S is a subset of R for the states of the form (a, R, S). 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 $q_I \in R$,

$$\big(R \times \{0\}\big) \cup \big(R' \times \{1\}\big) \models \bigwedge_{q \in R} \delta(q, b)$$

and

$$\big(R \times \{-1\}\big) \cup \big(R' \times \{0\}\big) \cup \big(Q \times \{1\}\big) \models \bigwedge_{q \in R'} \delta(q, c).$$

For a state (a, R, S), the transition function η is defined as follows. If $a \neq b$ then $\eta((a, R, S), b) := \emptyset$. If $a = b$ then $\eta((a, R, S), b)$ contains the state (c, R', S') iff the following conditions for c, R', and S' are satisfied: First,

$$\big(Q \times \{-1\}\big) \cup \big(R \times \{0\}\big) \cup \big(R' \times \{1\}\big) \models \bigwedge_{q \in R} \delta(q, b)$$

and

$$\big(R \times \{-1\}\big) \cup \big(R' \times \{0\}\big) \cup \big(Q \times \{1\}\big) \models \bigwedge_{q \in R'} \delta(q, c).$$

Second, $S' = R' \setminus F$ if $S = \emptyset$, and if $S \neq \emptyset$ then $S' \subseteq R' \setminus F$ and

$$\big(S' \cup (F \cap R')\big) \times \{1\} \models \bigwedge_{q \in S} \delta(q, b).$$

In the remainder of the proof, we show 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 $S_i \subseteq R_i \setminus F$, for all $i > 0$. We define the components a_i, R_i, and S_i separately.

For $i > 0$, we define $a_i := w_i$ and for $i \geq 0$, we define

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

The sets S_i are inductively defined: $S_0 := \emptyset$ and for $i > 0$, we define

$$S_i := \begin{cases} R_i \setminus F & \text{if } S_{i-1} = \emptyset, \\ \{s' \in Q \setminus F \mid r \text{ has a node } u \text{ with a child } v \text{ such that} \\ \qquad r(u) = (s, i-1),\ s \in S_{i-1},\ \text{and } r(v) = (s', i)\} & \text{otherwise.} \end{cases}$$

Note that $S_i \subseteq R_i$, for all $i \in \mathbb{N}$.

We first prove that for every $i \geq 0$, there is some $j \geq i$ such that $S_j = \emptyset$. Let $i \geq 0$. Assume that there is no $j \geq i$ with $S_j = \emptyset$. From the sets $S_i, S_{i-1}, \ldots$, we obtain a directed graph G with the vertexes (p, j) with $p \in S_j$. The edges are according to the run r of

the automaton $\mathcal{A}$ on the 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 visits a vertex in which a state in F occurs. This contradicts the assumption that every path in the run r infinitely often visits a configuration in which a state in F occurs.

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

– We start with the second case. Let $i > 0$. First, consider some arbitrary $q \in R_i$. By definition, there is a node u of r with $r(u) = (q, i)$. Let $v_1, \ldots, v_n$ 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 and $a_i = w_i$, we have $\{(q_1, h_1 - i), \ldots, (q_n, h_n - i)\} \models \delta(q, a_i)$. For all j with $1 \leq j \leq n$, we have $q_j \in R_i$ 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, we obtain

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

  Second, consider some arbitrary $q \in R_{i+1}$. By definition, there is a node u of r with $r(u) = (q, i+1)$. Let $v_1, \ldots, v_n$ 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 $a_{i+1} = w_{i+1}$, we have $\{(q_1, h_1 - i - 1), \ldots, (q_n, h_n - i - 1)\} \models \delta(q, a_{i+1})$. For all j with $1 \leq j \leq n$, we have $q_j \in R_{i-1}$ if $h_j = i - 1$, and $q_j \in R_i$ if $h_j = i$. Since $\delta(q, a_{i+1})$ is a positive Boolean formula, we obtain

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

  If $S_i = \emptyset$ then $S_{i+1} = R_{i+1} \backslash F$ 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.
– The reasoning for the first case is similar to the second case by setting i to 0. We have $q_I \in R_0$, since the root of r is labeled by the configuration $(q_I, 0)$. Furthermore, by definition, we have $S_1 = R_1 \backslash F$.

($\supseteq$) 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 $S_i \subseteq R_i \backslash F$, 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, i) then $q \in R_i$, for all $q \in Q$ and $i \in \mathbb{N}$. We label the root of r by $(q_I, 0)$. 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 $r(u) = (q, i)$, for some $q \in Q$ and $i \in \mathbb{N}$. We have $q \in R_i$. There are two cases.

– Assume that $i = 0$. By definition of the transition function, we have

$$\big(R_0 \times \{0\}\big) \cup \big(R_1 \times \{1\}\big) \models \bigwedge_{q \in R_0} \delta(q, w_0).$$

  Let $M \subseteq (R_0 \times \{0\}) \cup (R_1 \times \{1\})$ be a minimal model of $\delta(q, w_0)$. 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, d). This definition obviously satisfies the construction invariant.

– Assume that $i > 0$. By the definition of the transition function, we have

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

and for $j := i - 1$,

$$\big(R_j \times \{-1\}\big) \cup \big(R_{j+1} \times \{0\}\big) \cup \big(Q \times \{1\}\big) \models \bigwedge_{q \in R_{j+1}} \delta(q, w_{j+1}).$$

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

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

Let $M \subseteq (R_{i-1} \times \{-1\}) \cup (R_i \times \{0\}) \cup (R_{i+1} \times \{1\})$ 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 $(p, i + d)$. This definition obviously satisfies the construction invariant.

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

It remains to show that r is accepting. For the sake of contradiction, assume that there is a rejecting path π in r with $r(\pi) = (p_0, h_0)(p_1, h_1) \ldots$. Since r is a minimal run of an eventually 1-way automaton, there is a position $n \in \mathbb{N}$ such that for all $i \geq n$, we have $h_i < h_{i+1}$ and $p_i \notin F$. By the construction invariant, we have $p_i \in R_i$, for all $i \in \mathbb{N}$. Consider a position $m \geq n$ with $S_m = \emptyset$. This position exists because ϱ is accepting. The state p_{m+1} is a member of S_{m+1} since ϱ is a run of $\mathcal{B}$ and $S_{m+1} = R_{m+1} \backslash F$. By induction, we infer that for all positions $i \geq m + 1$ we have $p_i \in S_i$ since ϱ obeys the transition function of $\mathcal{B}$. This contradicts the assumption that ϱ is an accepting run of $\mathcal{B}$. □

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

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

Proof First, we transform ψ into a logically equivalent formula ψ' that is in negation normal form of size $2\|\psi\|$. Let $\mathcal{A}_{\psi'}$ be the 2ABA that we obtain from ψ' by the construction in Sect. 4.1. By the Lemmas 6, 7, and 9, $\mathcal{A}_{\psi'}$ is eventually 1-way, locally 1-way, and its size is bounded by $4 + 2^{2\|\psi\|}$. 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 $\mathcal{O}(2^{|P|} \cdot 3^{2^{2\|\psi\|}})$ states. Note that 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 $q \in Q \backslash \{q_I\}$ 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 $L(\mathcal{C}) = L(\psi)$ and $\|\mathcal{C}\| \in \mathcal{O}(2^{|P|} \cdot 3^{2^{2\|\psi\|}})$. □

We make the following remark on the size of the resulting NBA of the presented construction for PPSL.

Remark 13 The size of the constructed 2ABA $\mathcal{A}_\varphi$ in Sect. 4.1 depends on the number of subformulas of the given PPSL 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

$$4 \;+\; |Sub(\varphi)| \;+\; \sum_{\alpha \star\!\!\rightarrow \psi \in Sub(\varphi)} \|\alpha\|^{1+Is(\alpha)} \;+\; \sum_{\mathsf{cl}(\alpha) \in Sub(\varphi)} \|\alpha\|^{1+Is(\alpha)} \;+\; \sum_{\neg \mathsf{cl}(\alpha) \in Sub(\varphi)} \|\alpha\|^{1+Is(\alpha)}$$

when φ is in negation normal form. Here, $\star\!\!\rightarrow$ ranges over the elements in the set $\{\Diamond\!\!\rightarrow, \Box\!\!\rightarrow, \Diamond\!\!\!\!\rightarrow, \Box\!\!\!\!\rightarrow\}$. With this new upper bound, we conclude that for a PPSL formula ψ with m propositions, there is a language-equivalent NBA of size

$$\mathcal{O}\left(2^m \cdot 3^{2\cdot|Sub(\psi)|+2\cdot \sum\limits_{\alpha \star\!\!\rightarrow \psi \in Sub(\varphi)} \|\alpha\|^{1+Is(\alpha)}+2\cdot \sum\limits_{\mathsf{cl}(\alpha) \in Sub(\varphi)} \|\alpha\|^{1+Is(\alpha)}}\right),$$

which refines the upper bound in Theorem 12. In particular, the size of the resulting NBA is only exponential in the size of ψ and not double exponential anymore when the number of intersection operators in the SEREs are bounded. Note that in the worst case the transformation of ψ into a PPSL formula in negation normal form doubles $|Sub(\psi)|$ and the number of SEREs in the subformulas.

5 Consequences of the translation

In this section, we prove some facts that follow from Theorem 12.

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

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

Proof We first show the EXPSPACE membership of the satisfiability problem for PPSL. Satisfiability of an instance φ can be checked by determining whether the language of the NBA $\mathcal{B}_\varphi$ according to Theorem 12 is nonempty. The emptiness check can be done by two simple reachability checks in the state graph of $\mathcal{B}_\varphi$: an accepting state needs to be found so that (1) it is reachable from the initial state and (2) it must be reachable from itself by a nonempty loop. Reachability can be decided in nondeterministic logarithmic space in the size of $\mathcal{B}_\varphi$, and therefore in nondeterministic exponential space in the size of φ. The crucial insight here is that the automaton can be constructed on-the-fly (see, e.g., [32]) whilst being checked for emptiness. Since NEXPSPACE equals EXPSPACE according to Savitch's Theorem [30], we conclude that the satisfiability problem for PPSL is in EXPSPACE.

If the number of intersection operators in the SEREs that occur in the instances φ is bounded by some constant, then the size of the NBA $\mathcal{B}_\varphi$ is bounded by $\mathcal{O}(2^{p(\|\varphi\|)})$, for some polynomial p (see Remark 13). In this case, we can check emptiness of $\mathcal{B}_\varphi$ in NPSPACE (measured in the size of φ), which equals PSPACE using Savitch's Theorem again.

Not surprisingly, these upper bounds transfer to the model-checking problem, that is, the question whether all paths in a given Kripke structure $\mathcal{K}$ from a state s of $\mathcal{K}$ satisfy a given PPSL formula φ. Since PPSL is closed under negation, we can build an NBA $\mathcal{B}_{\neg\varphi}$ and check whether the intersection of the automaton's language with the language of the Kripke structure is empty. As above, the construction of $\mathcal{B}_{\neg\varphi}$ is done on-the-fly while carrying out the emptiness check of the intersection of $\mathcal{B}_{\neg\varphi}$ and the Kripke structure $\mathcal{K}$.

The hardness results for the satisfiability and the model-checking problem for PPSL follow directly from the hardness results for PSL and SVA, which are shown in [7,20]. □

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 PPSL formula φ has a model of the form uv^ω with $|uv| \leq c \cdot 2^{\|\varphi\|} \cdot 3^{2^{2\|\varphi\|}}$.

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

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 PPSL describe the same class of properties, the question arises of their relative succinctness. The next theorem states an upper bound on the translation from PPSL to SVA. Roughly speaking, for the proof, we translate a PPSL formula into an NBA, which is in turn translated into a deterministic Muller automaton. From the Muller automaton we obtain an ω-regular expression, which we finally translate into an SVA formula. We remark that a similar translation using regular expressions and the connectives $\diamond\!\!\rightarrow$ and $\Box\!\!\rightarrow$ appears in [3] to show that the logic ForSpec is capable of describing all ω-regular languages. However, no upper bounds of the translation in [3] are given.

Recall that Muller automata are defined similar to Büchi automata except that their acceptance condition is given as a set of set of states. In particular, a run of a deterministic Muller automaton with the acceptance condition $\mathcal{F}$ is accepting if the set of states visited infinitely often is in $\mathcal{F}$. For the sake of readability, we define $2_0^x := x$ and $2_k^x := 2^{2_{k-1}^x}$, for $k > 0$.

Theorem 16 *For any PPSL formula φ, there is an initially equivalent SVA formula of size $2_4^{\mathcal{O}(\|\varphi\|)}$ and in which the intersection operator does not occur.*

Proof According to Theorem 12, we construct for the PPSL formula φ, an NBA $\mathcal{B}_\varphi$ with $L(\mathcal{B}_\varphi) = L(\varphi)$ and $\|\mathcal{B}_\varphi\| \in \mathcal{O}(2^{\|\varphi\|} \cdot 3^{2^{2\|\varphi\|}})$. From the NBA $\mathcal{B}_\varphi$, we obtain a language-equivalent deterministic Muller automaton $\mathcal{A} = (Q, \Sigma, \delta, q_I, \mathcal{F})$ by using Safra's construction [28]. The size of $\mathcal{A}$ is $2^{\mathcal{O}(\|\mathcal{B}_\varphi\| \cdot \log \|\mathcal{B}_\varphi\|)} \subseteq 2_3^{\mathcal{O}(\|\varphi\|)}$.

We describe the language of $L(\mathcal{A})$ as follows in terms of finite word languages. For $S \subseteq Q$ and $s, t \in S$, $\mathcal{A}_{s,t}^S$ denotes the NFA $(S, \Sigma, \delta', s, \{t\})$ with $\delta'(p, a) := \delta(p, a) \cap S$, for all $p \in S$ and $a \in \Sigma$. That means, we restrict $\mathcal{A}$'s state space to S and view it as an automaton over finite words with the initial state s and the singleton acceptance set $\{t\}$. Furthermore, L^ω denotes the language of infinite words that we obtain by an infinite concatenation of nonempty words from the language L of finite words. We have

$$L(\mathcal{A}) = \bigcup_{\substack{F \in \mathcal{F} \text{ with} \\ F = \{f_1, \ldots, f_n\}}} L(\mathcal{A}_{q_I, f_1}^Q) \,;\, \Big(L(\mathcal{A}_{f_1, f_2}^F) \,;\, \ldots \,;\, L(\mathcal{A}_{f_{n-1}, f_n}^F) \,;\, L(\mathcal{A}_{f_n, f_1}^F)\Big)^\omega. \tag{5}$$

In the remainder of the proof, we will use REs and the operators $\Diamond\!\!\rightarrow$ and $\Box\!\!\rightarrow$ to express the right-hand side of (5) as an SVA formula. Recall that for $S \subseteq Q$ and $r, s \in S$, the finite word language $L(\mathcal{A}^S_{s,t})$ can be expressed as an RE $\alpha^S_{s,t}$ of size $\mathcal{O}(2^{|S|})$, see [18]. Furthermore, observe that for an SERE α with $L(\alpha) \neq \emptyset$ and a PPSL formula ψ, we have

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

For an SERE α with $L(\alpha) \neq \emptyset$ and $L(\alpha) \neq \{\varepsilon\}$, we have

$$(L(\alpha))^\omega = L\left(\alpha \Diamond\!\!\rightarrow \mathsf{tt}\right) \cap L\left(\alpha^* \Box\!\!\rightarrow \mathsf{X}(\alpha \Diamond\!\!\rightarrow \mathsf{tt})\right) \tag{7}$$

whenever $L(\alpha)$ satisfies for all nonempty finite words u, v the condition: if $uv \in L(\alpha)$ and $u \in L(\alpha)$ then $v \in L(\alpha)$. Note that not every regular expression satisfies the equality (7). However, in our case, the equality holds, since the regular expression (see definition below) represents the words in a deterministic automaton that describe a loop from a given state.

For $F \in \mathcal{F}$ with $F = \{f_1, \ldots, f_n\}$, we define the RE

$$\beta^F_{f_1,\ldots,f_n} := \alpha^F_{f_1,f_2}\,;\,\ldots\,;\,\alpha^F_{f_{n-1},f_n}\,;\,\alpha^F_{f_n,f_1}.$$

Obviously, we have $L(\beta^F_{f_1,\ldots,f_n}) = L(\mathcal{A}^F_{f_1,f_2})\,;\,\ldots\,;\,L(\mathcal{A}^F_{f_{n-1},f_n})\,;\,L(\mathcal{A}^F_{f_n,f_1})$. Moreover, $L(\beta^F_{f_1,\ldots,f_n})$ satisfies the condition that a nonempty word v is in $L(\beta^F_{f_1,\ldots,f_n})$ whenever a nonempty word u and the word uv are both in $L(\beta^F_{f_1,\ldots,f_n})$.

To simplify matters, we assume in the following, without loss of generality, that every $F \in \mathcal{F}$ is reachable from the initial state, this means, $L(\mathcal{A}^Q_{q_I,f}) \neq \emptyset$, for each $f \in F$. We also assume for each $F \in \mathcal{F}$ that $q_I \notin F$ and that $L(\mathcal{A}^F_{f,f'}) \neq \emptyset$ and $L(\mathcal{A}^F_{f,f'}) \neq \{\varepsilon\}$, for all $f, f' \in F$.

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

$$\varphi' := \bigvee_{\substack{F \in \mathcal{F} \text{ with} \\ F = \{f_1,\ldots,f_n\}}} \alpha^Q_{q_I,f_1} \Diamond\!\!\rightarrow \mathsf{X}\psi^F_{f_1,\ldots,f_n}\,,$$

where $\psi^F_{f_1,\ldots,f_n} := (\beta^F_{f_1,\ldots,f_n} \Diamond\!\!\rightarrow \mathsf{tt}) \wedge ((\beta^F_{f_1,\ldots,f_n})^* \Box\!\!\rightarrow \mathsf{X}(\beta^F_{f_1,\ldots,f_n} \Diamond\!\!\rightarrow \mathsf{tt}))$. Obviously, no intersection operator occurs in the SEREs of φ'. An upper bound on the length of φ' is

$$\|\varphi'\| \in \mathcal{O}\left(\sum_{F\in\mathcal{F}} (2^{|Q|} + 3 \cdot |F| \cdot 2^{|F|})\right) \subseteq \mathcal{O}(2^{|Q|} \cdot 2^{3\cdot|Q|}) \subseteq 2_4^{\mathcal{O}(\|\varphi\|)}.$$

□

It is fair to ask whether the upper bound in Theorem 16 is optimal, this means, whether there is a family of PPSL 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, for example, the Gap Lemma in [21]). We are only able to establish an exponential lower bound. This result is presented in the next section.

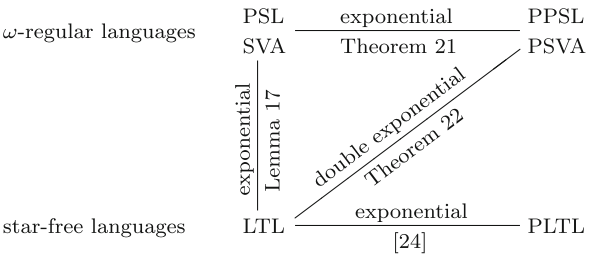

Fig. 1 Succinctness gaps

6 Succinctness gaps

In this section, we prove an exponential succinctness gap between PPSL and PSL/SVA, this means, there is a family $(\Phi_n)_{n>0}$ of PPSL formulas such that for every family $(\Psi_n)_{n>0}$ of PSL (and therefore also 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, PPSL, is double exponentially more succinct than LTL on the star-free languages. Figure 1 summarizes the results of this section.

Our proof for the succinctness gap between PSVA and PSL has a similar flavor as the proof in [24], 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 [24] 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 Υ_n such that for any LTL formula Ξ_n, if $L(\Xi_n) = L(\Upsilon_n)$ then $\|\Xi_n\| \in \Omega(2^{\|\Upsilon_n\|})$.*

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

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

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

$$\mathsf{G}\left(\bigwedge_{1 \le i \le n} (p_i \leftrightarrow \mathsf{OH} p_i) \rightarrow (p_0 \leftrightarrow \mathsf{OH} p_0)\right).$$

Literally, the SVA formula Υ_n and the above PLTL formula state that for any position, p_0's truth value is equal the corresponding truth value at the initial position whenever the truth values of the propositions $p_1, \ldots, p_n$ at that position are equal to the corresponding truth values at the initial position.

From [24], it follows that any LTL formula Ξ_n that is initially equivalent to Υ_n is exponentially larger than Ξ_n. □

Let us now turn to the succinctness gap between PSVA and PSL. 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

$$val_n(b) := \sum_{0 \le i < n} 2^{c_i'} \quad \text{with} \quad c_i' := \begin{cases} 1 & \text{if } c_i \in b, \\ 0 & \text{otherwise.} \end{cases}$$

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 $val_n(w_0) = 0$ and $val_n(w_{i+1}) \equiv val_n(w_i) + 1 \mod 2^n$, 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 $L(count_n) \subseteq \Sigma_n^\omega$ 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

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

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 $r \in \{p, q\}$, 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 of 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 $samepos_n$ such that for every subword $v \in \Sigma_n^*$ of an n-counting word $w \in \Sigma_n^\omega$, it holds that $v \in L(samepos_n)$ 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

$$samepos_n := \bigcap_{0 \le i < n} \left((c_i \,;\, \mathsf{tt}^* \,;\, c_i) \cup (\neg c_i \,;\, \mathsf{tt}^* \,;\, \neg c_i) \right).$$

With the SERE $samepos_n$ at hand, we easily define a PPSL formula that checks whether a position is in the initial n-segment of an n-counting word:

$$initial_n := \neg(samepos_n \Diamond\!\!\rightarrow \mathsf{tt}).$$

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

$$back_n^\psi := samepos_n \Diamond\!\!\rightarrow (initial_n \wedge \psi).$$

For an n-counting word $w \in \Sigma_n^\omega$ and $i \in \mathbb{N}$, it holds that $w, i \models back_n^\psi$ iff $w, i \mod 2^n \models \psi$. Intuitively, $back_n^\psi$ 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 $within_n := (\neg c_{n-1})^* ; (c_{n-1})^*$. 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 $start_n := \bigwedge_{0 \le i < n} \neg c_i$ checks whether a position is the first one of an n-segment in an n-counting word.

Finally, consider the PPSL formula $\Phi_n := count_n \wedge \varphi_n$, where

$$\varphi_n := \mathsf{G}\left(start_n \wedge \left(within_n \Box\!\!\rightarrow (p \leftrightarrow back_n^p)\right) \rightarrow \left(within_n \Box\!\!\rightarrow (q \leftrightarrow back_n^q)\right)\right).$$

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 $L(\Phi_n) = L_n$. 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 $\|\mathcal{B}\| \geq 2_3^n$.*

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 the words $v_0, \ldots, v_{N-1} \in \{\emptyset, \{p\}\}^*$ 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 $M \geq 2_3^n$. 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 $(v^{S'})^\omega$ visits the same state s after $N \cdot 2^n$ many steps, this means, 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 PSL.

Theorem 21 *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for every PSL 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 a PSL formula that is initially equivalent to Φ_n. Let $\Psi'_n := count_n \wedge \mathsf{G}(\neg c_0 \wedge \cdots \wedge \neg c_{n-1} \rightarrow \Psi_n)$. 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, this means, $L(\Psi'_n) = L'_n$. 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 $\|\mathcal{B}\| \geq 2_3^n$. 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, this means, there is an LTL formula φ_n such that $L(\varphi_n) = L_n$. We now adapt the proof of Theorem 21 to obtain a double exponential succinctness gap between PSVA and LTL.

Theorem 22 *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for any LTL 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 LTL 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, this means, $L(\Xi'_n) = L'_n$. 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 $\|\mathcal{B}\| \geq 2_3^n$. 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 PPSL 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 $\mathit{samepos}_n$ 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 [22,24] for a fixed and finite proposition set also only shows that PLTL is superpolynomially more succinct than LTL.

7 Conclusion

In this article, we have proposed the temporal logic PPSL, which extends PSL and SVA with past operators. We have analyzed its complexity and our results show that PPSL 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 PPSL over PSL and SVA pay off in practice. The presented translation for PPSL into NBAs shows that the additional cost for handling past operators is small and should not be a burden in implementing PPSL in system verification. Our preliminary experience with a prototype implementation for the model checker NuSMV are promising.[2]

Acknowledgments The work was partly supported by the Swiss National Science Foundation (SNF). We thank the reviewers for their valuable comments.

[2] Our tool is publicly available at www.infsec.ethz.ch/research/Software/#PSL2BA.

A Additional Proof Details

In this appendix, we prove Lemma 5 about the accepted language of the 2ABA $\mathcal{A}_\varphi$ constructed in Sect. 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 $p \in w_i$ 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 $w, i \models \mathsf{cl}(\alpha)$ 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 that $w, i \models \psi$, this means, $w, i \models \psi_1$ 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 that $w, i \models \mathsf{X}\gamma$, this means, $w, i+1 \models \gamma$. 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 that $w, i \models \mathsf{Z}\gamma$, this means, $i = 0$ 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 that $w, i \models \psi_1 \,\mathsf{U}\, \psi_2$, this means, 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 that the left-hand side holds. Then, $\mathcal{A}_\varphi$ accepts $\#w$ from the 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 the 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, this means, 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 that $w, i \models \psi_1 \mathsf{R} \psi_2$, this means, 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 that 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 the 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 the 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 the 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 the 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 the configuration $(\psi_1, k+1)$ and for all j with $i \leq j \leq k$, $\mathcal{A}_\varphi$ accepts $\#w$ from the 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 $(\psi_1 \mathsf{R} \psi_2, k-1)$. By iterating 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 the configuration $(\psi_1 \mathsf{R} \psi_2, i+1)$.

Now, we show the other direction. Assume that the right-hand side holds, this means, $\mathcal{A}_\varphi$ 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, this means, there is an integer $k \geq i$ 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 obtain that $\mathcal{A}_\varphi$ 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 $\psi = \alpha \diamondsuit\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume that $w, i \models \psi$, this means, 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 the configuration $(\gamma, k+1)$. That is, $\mathcal{A}_\varphi$ accepts from the configuration $(\alpha \diamondsuit\!\!\rightarrow \gamma, i+1)$ 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 the configuration $(\alpha \diamondsuit\!\!\rightarrow \gamma, i+1)$.

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

Step Case $\psi = \alpha \square\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume that $w, i \models \psi$, this means, 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 the configuration $(\gamma, k+1)$. This is equivalent to the fact that there exists a run of $\mathcal{A}_\varphi$ on $\#w$ from the configuration $(\alpha \square\!\!\rightarrow \gamma, i+1)$ 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 the configuration $(\alpha \square\!\!\rightarrow \gamma, i+1)$.

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

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
3. Armoni, R., Fix, L., Flaisher, A., Gerth, R., Ginsburg, B., Kanza, T., Landver, A., Mador-Haim, S., Singerman, E., Tiemeyer, A., Vardi, M.Y., Zbar, Y.: 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). Lecture Notes in Computer Science, vol. 2280, pp. 296–211. Springer (2002)
4. Banieqbal, B., Barringer, H.: Temporal logic with fixed points. In: Proceedings of Temporal Logic in Specification 1987. Lecture Notes in Computer Science, vol. 398, pp. 62–74. Springer (1989)
5. Ben-David, S., Bloem, R., Fisman, D., Griesmayer, A., Pill, I., Ruah, S.: Automata construction algorithms optimized for PSL. Technical Report, The Prosyd Project, http://www.prosyd.org (2005)
6. Bloem, R., Cimatti, A., Pill, I., Roveri, M.: Symbolic implementation of alternating automata. Int. J. Found. Comput. Sci. **18**(4), 727–743 (2007)
7. Bustan, D., Havlicek, J.: Some complexity results for SystemVerilog assertions. In: Proceedings of the 18th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 4144, pp. 205–218. Springer (2006)
8. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An opensource tool for symbolic model checking. In: Proceedings of the 14th International Conference on Computer Aided Verification (CAV). Lecture Notes in Computer Science, vol. 2404, pp. 359–364. Springer (2002)
9. Cimatti, A., Roveri, M., Semprini, S., Tonetta, S.: From PSL to NBA: a modular symbolic encoding. In: Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design (FMCAD), pp. 125–133. IEEE Computer Society Press (2006)
10. Cimatti, A., Roveri, M., Sheridan, D.: Bounded verification of Past LTL. In: Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD). Lecture Notes in Computer Science, vol. 3312, pp. 245–259. Springer (2004)

11. Clarke, E.M., Grumberg, O., Hamaguchi, K.: Another look at LTL model checking. Form. Method Syst. Des. **10**(1), 47–71 (1997)
12. Dax, C., Klaedtke, F.: Alternation elimination by complementation. In: Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR). Lecture Notes in Computer Science, vol. 5530, pp. 214–229. Springer (2008)
13. Demri, S., Schnoebelen, P.: The complexity of propositional linear temporal logics in simple cases. Inf. Comput. **174**(1), 84–103 (2002)
14. Diekert, V., Gastin, P. : First-order definable languages. In: Flum, J., Grädel, E., Wilke, T. (eds.) Logic and Automata: History and Perspectives, Texts in Logic and Games, vol. 2, pp. 261–306. Amsterdam University Press, Amsterdam (2007)
15. Gastin, P., Oddoux, D.: LTL with past and two-way very-weak alternating automata. In: Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS). Lecture Notes in Computer Science, vol. 2747, pp. 439–448. Springer (2003)
16. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
17. Henriksen, J.G., Thiagarajan, P.S.: Dynamic linear time temporal logic. Ann. Pure Appl. Logic **96**(1–3), 187–207 (1999)
18. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. 3rd edn. Addison-Wesley, Reading (2006)
19. Kupferman, O., Piterman, N., Vardi, M.Y.: Extended temporal logic revisited. In: Proceedings of the 12th International Conference on Concurrency Theory (CONCUR). Lecture Notes in Computer Science, vol. 2154, pp. 519–535. Springer (2001)
20. Lange, M.: Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In: Proceedings of the 18th International Conference on Concurrency Theory (CONCUR). Lecture Notes in Computer Science, vol. 4703, pp. 90–104. Springer (2007)
21. Lange, M.: A purely model-theoretic proof of the exponential succinctness gap between CTL^+ and CTL. Inf. Process. Lett. **108**(5), 308–312 (2008)
22. Laroussinie, F., Markey, N., Schnoebelen, P.: Temporal logic with forgettable past. In: Proceedings of the 17th IEEE Symposium on Logic in Computer Science (LICS), pp. 383–392. IEEE Computer Society Press (2002)
23. Lichtenstein, O., Pnueli, A., Zuck, L.D.: The glory of the past. In: Proceedings of the Conference on Logics of Programs 1985. Lecture Notes in Computer Science, vol. 193, pp. 196–218. Springer (1985)
24. Markey, N.: Temporal logic with past is exponentially more succinct. Bull. EATCS **79**, 122–128 (2003)
25. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theor. Comput. Sci. **32**(3), 321–330 (1984)
26. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS), pp. 46–57. IEEE Computer Society Press (1977)
27. Pnueli, A., Zaks, A.: PSL model checking and run-time verification via testers. In: Proceedings of the 14th International Symposium on Formal Methods (FM). Lecture Notes in Computer Science, vol. 4085, pp. 573–586. Springer (2006)
28. Safra, S.: On the complexity of ω-automata. In: Proceedings of the 29th Annual Symposium on Foundations of Computer Science (FOCS), pp. 319–327. IEEE Computer Society Press (1988)
29. Sánchez C., Leucker, M.: Regular linear temporal logic with past. In: Proceedings of the 11th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI). Lecture Notes in Computer Science, vol. 5944, pp. 295–311. Springer (2010)
30. Savitch, W.J.: Relationships between nondeterministic and deterministic tape complexities. J. Comput. Syst. Sci. **4**(2), 177–192 (1970)
31. Vardi, M.Y.: A note on the reduction of two-way automata to one-way automata. Inf. Process. Lett. **30**(5), 261–264 (1989)
32. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Proceedings of the 8th Banff Higher Order Workshop on Logics for Concurrency 1995. Lecture Notes in Computer Science, vol. 1043, pp. 238–266. Springer (1996)
33. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification (preliminary report). In: Proceedings of the 1st Symposium on Logic in Computer Science (LICS), pp. 332–344. IEEE Computer Society Press (1986)
34. Wolper, P.: Temporal logic can be more expressive. Inf. Control **56**(1/2), 72–99 (1983)