From Temporal Logics to Automata via Alternation Elimination

Christian Dax

2010

DISS. ETH NO. 19200

From Temporal Logics to Automata via Alternation Elimination

A dissertation submitted to
ETH ZURICH
for the degree of
Doctor of Sciences

presented by
CHRISTIAN NIKOLAUS DAX
Dipl.-Inf., Ludwig-Maximilians-Universität München
born on December, 9th 1979
citizen of Germany

accepted on the recommendation of
Prof. David Basin, examiner
Prof. Kousha Etessami, co-examiner
Dr. Felix Klaedtke, co-examiner

2010

Acknowledgements

First, I would like to thank my supervisor David Basin for providing me such a great research environment and for all his scientific advices and moral support.

I also would like to thank my supervisor Felix Klaedtke who guided me through my PhD. He encouraged me to pursue my own ideas, helped me to develop them, and taught me how to present them on a high scientific level. Thank you for all your precious time and invaluable support!

Further, I would like to thank my co-examiner Kousha Etessami for helpful comments on my thesis. I also would like to thank Martin Lange who introduced me to the world of automata and temporal logics. He provided the key idea of the succinctness result between the logics PSL and PPSL presented in this thesis. I also like to thank Nir Piterman for the fruitful email correspondence that helped me to find the topic of my thesis.

I also like to thank all the information and system security group members for the pleasant and productive working atmosphere. I especially thank my office mates Matus Harvan and Mario Frank for their good company, Cas Cremers for all the scientific challenges at and outside of ETH, Alexander Pretschner for asking for the practical relevance of my work, Stefan Leue, Christoph Sprenger, Matthias Schmalz, Patrick Schaller, Simon Meier, Lukas Brügger, Benedikt Schmidt, Andreas Fürst, Eugen Zalinescu, Michèle Feltz, Mohammad Dashti, and Simone Frau for their helpful comments on my work and on my defense talk.

Most importantly, I would like to thank my parents, my two brothers, and Kathrin for all their patience and unconditional support.

Abstract

The automata-based approach for automated verification of finite-state systems and recursive state machines, i.e., finite-state systems with subprograms that can be called recursively, requires efficient translations from specification languages like LTL, PSL, and NWTL to nondeterministic automata. Since formulas of those specification languages can directly be translated into alternating automata, it suffices to solve the problem of removing alternation.

In this thesis, we present a construction scheme that reduces alternation elimination to the problem of complementing so-called existential automata. Existential automata are nondeterministic automata that inspect only a single path in their inputs. The presented alternation-elimination constructions are instances of our scheme. We obtain these instances by revisiting state-of-the-art complementation constructions and by providing novel ones for restricted classes of 1-way and 2-way existential automata. With these instances at hand, we correct, simplify, improve, and generalize previously proposed translations from temporal logics to nondeterministic automata. From some instances we obtain novel translations.

Moreover, we extend various temporal logics with past operators and utilize our new alternation-elimination constructions to obtain translations to nondeterministic automata. For instance, we extend the IEEE standard PSL by past operators. We call this logic PPSL and show that the additional cost for translating PPSL formulas to nondeterministic automata is rather small whereas PPSL is exponentially more succinct than PSL.

Zusammenfassung

Beim automatenbasierten Ansatz für das automatische Verifizieren von endlichen Zustandssystemen und von so genannten endlichen rekursiven Zustandssystemen, was endliche Zustandssysteme mit zusätzlichen rekursiv aufrufbaren Unterprogramme sind, werden effiziente Übersetzungen von Spezifikationssprachen wie LTL, PSL und NWTL in nichtdeterministische Automaten benötigt. Da Formeln dieser Spezifikationssprachen direkt in alternierende Automaten übersetzt werden können, reicht es das Problem der Alternierungselimination für alternierende Automaten zu lösen.

In dieser Arbeit stellen wir ein Konstruktionsschema vor, dass das Problem der Alternierungselimination auf das Problem der Komplementierung eines existentiellen Automaten reduziert. Ein existentieller Automat is ein nichtdeterministischer Automat, der nur einen einzigen Pfad in der Eingabe betrachtet. Die vorgestellten Konstruktionen zur Alternierungselimination sind Instanzen des Schemas. Wir erhalten diese Instanzen, indem wir neuste Komplementierungskonstruktionen untersuchen und zudem neue Komplementierungskonstruktionen für Unterklassen von 1-wege und 2-wege Automaten einführen. Mit Hilfe dieser Instanzen korrigieren, vereinfachen, verbessern und generalisieren wir bereits bekannte Übersetzungen von temporalen Logiken in nichtdeterministische Automaten. Zudem erhalten wir aus einigen dieser Instanzen neue Übersetzungen.

Des weiteren erweitern wir verschiedene Logiken mit Vergangenheitsoperatoren und nutzen die vorgestellten Konstruktionen zur Alternierungselimination, um Übersetzungen in nichtdeterministische Automaten zu erhalten. Ein Beispiel ist der IEEE Standard PSL, den wir mit Vergangenheitsoperatoren erweitern. Für die neue Logik, die wir PPSL nennen, zeigen wir, dass die zusätzlichen Kosten für die Übersetzung in nichtdeterministische Automaten relativ klein sind. Im Gegensatz dazu steht, dass PPSL Eigenschaften exponentiell kürzer beschreiben kann als PSL.

Contents

Chapter 1

Introduction

1.1 Scope, Motivation, and Results

Information and communication technology (ICT) systems play an important part in our daily lives. We use these systems, for instance, as banking applications, mobile phones, transportation systems, traffic control and alert systems, or medical applications. Since the significance of ICT systems increase steadily and we become more and more dependent on them, the reliability of these systems also becomes increasingly important. System errors may have substantial financial consequences. A bug in Intel's Pentium II floating-point division unit caused a loss of about 475 million US dollars. The opening of Denver's airport was delayed for nine months at a loss of about 1.1 million US dollars due to a software error in the automated baggage handling system. If a failure occurs in a safety-critical system, the cost can become unacceptably high. In 1996, the maiden flight of Ariane 5 rocket ended in a firework about forty seconds after its lift-off because of a malfunction in the control software. Similar bugs have been found in the Mars Pathfinder and the airplanes of the Airbus family. Between 1985 and 1987, six cancer patients died after receiving overdoses of radiation due to miscalculation of the control part of the radiation therapy machine Therac-25. For more examples, we refer to [BK08].

To ensure that critical ICT systems behave correctly, developers try to find failures through simulation or testing. However, systems tend to be large and too complex to be thoroughly tested. Subtle design errors resulting in unexpected behavior might be missed. In formal verification, we represent the system and its specification by mathematical models and prove that *every* system execution fulfills the given specification. Clarke and Emerson [CE82] and independently, Queille and Sifakis [QS82] introduced the *model-checking*

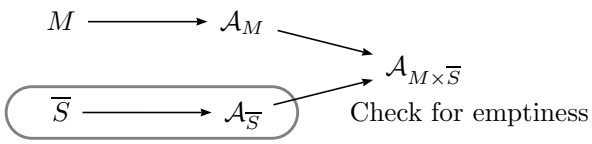


Figure 1.1: Automata-Theoretic Approach to Model Checking

problem that asks for an algorithm that automatically checks whether a mathematical model M of a system fulfills a specification S, see [CGP99]. In the context of automatic verification, the model M usually represents all possible system computations that are described by sequences of system configurations, and the specification S describes the allowed configuration sequences, which we refer to as *good computations*. We say M fulfills S if every system computation is also a good computation.

In [VW86], Vardi and Wolper introduce an *automata-theoretic approach* for solving the model checking problem. This approach assumes that the representation of the system behavior M and the representation of bad computations $\overline{S}$ can both be translated into nondeterministic finite-state automata $\mathcal{A}_M$ and $\mathcal{A}_{\overline{S}}$, respectively. Then, we can solve the model checking problem in two steps. First, we construct the product automaton $\mathcal{A}_{M\times\overline{S}}$ that represents the intersection of all system computations and all bad computations. Second, we check whether the set of computations represented by $\mathcal{A}_{M\times\overline{S}}$ is empty. This is the case if and only if M fulfills S. Figure 1.1 illustrates this approach. Since $\mathcal{A}_M$ and $\mathcal{A}_{\overline{S}}$ are nondeterministic automata, computing the intersection and checking for emptiness is efficiently solvable, see [Var07] for more details.

Model checking plays an important role in automatic verification and is increasingly used in hardware industry. Here, a system behavior is represented by a *labeled transition system* M and a specification is described by a temporal logic formula S. Several different approaches have been introduced to encode a labeled transition system M in a concise and intuitive way [Hol04, McM92, CGP99, BK08]. Since labeled transition systems can be viewed as nondeterministic automata, we can directly extract the representation $\mathcal{A}_M$ from those encodings. For specifying desired behavior, several temporal logics have been proposed that vary in their *expressiveness*, *succinctness*, and *implementability* [AFF+02]. A logic is succinct if we can easily read and write relevant specifications with this logic. For instance, adding past operators may not make a logic more expressive but may increase its succinctness.

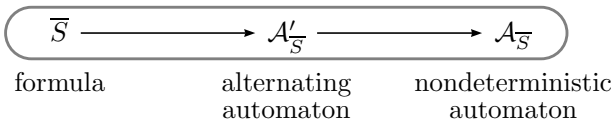


Figure 1.2: Translation from Logics to Automata

With implementability of a logic, we measure the cost for obtaining the nondeterministic automaton $\mathcal{A}_{\overline{S}}$ from a formula in the logic. The scope of this thesis is the design of temporal logics and the translations from these logics to nondeterministic automata. In the following, we expand on this part.

From Logics to Automata When translating temporal logics to nondeterministic automata, *alternating automata* play an essential role. They serve as a kind of glue between declarative specification languages like the logics LTL [Pnu77] and PSL [Psl05] and nondeterministic automata. Translations of declarative specification languages into alternating automata are usually rather direct and easy to establish due to the rich combinatorial structure of alternating automata, see for instance [VW07, EJ91, Boz07]. Translating an alternating automaton into a nondeterministic automaton remains to be a purely combinatorial problem. Figure 1.2 illustrates this two-step approach.

This *two-step approach* [Var98] is introduced by Vardi and has the following advantages over translations that do use alternating automata as intermediate step: (*i*) The two-step approach splits a direct translation from a formula into a nondeterministic automaton in two independent translations. The constructions of these two parts are easier to understand, to establish their correctness, and to implement than the direct construction. (*ii*) Alternating automata represent temporal logic formulas in an abstract and uniform way. Thus, alternation elimination is a mathematically elegant way to formalize translations from logics to nondeterministic automata. (*iii*) Another benefit is the fact that we can reuse alternation-elimination constructions, their correctness proofs, and their implementations for different logics. (*iv*) We can optimize an alternating automaton before we translate it into a nondeterministic automaton. Several automata-based techniques are applicable like simulation-based reduction techniques [GO01, EWS05, FW05, FW06].

Different classes of alternating automata are used for these kinds of translations depending on the expressive power of the specification language. For

instance, for temporal logics like LTL, CaRet [AEM04], NWLTL [AAB+08], or CTL* [EH86], restricted classes of alternating automata suffice, namely automata that are *very weak* [Roh97,LT00]. For LTL, these restrictions have been exploited to obtain efficient translators to nondeterministic Büchi automata, see [GO01, Fri03]. For fragments of the standardized property specification language PSL [Psl05] or RXPath [CGLV09], one uses alternating Büchi automata [BFH05]. For logics with explicit fixpoint operators like the linear-time μ-calculus μLTL [BB89, Var88], μNWTL [Boz07], or the μ-calculus [Koz82], one uses alternating parity automata. If the temporal specification language has future and past operators, one uses 2-way alternating automata instead of 1-way alternating automata, see for instance [Var98, KPV01, GO03, Boz07, DKL09, CGLV09].

Alternation-Elimination Scheme In this thesis, we present a general construction scheme for translating restricted classes of alternating automata into language-equivalent nondeterministic automata. In a nutshell, the general construction scheme shows that the problem of translating an alternating automaton into a language-equivalent nondeterministic automaton reduces to the problem of complementing a nondeterministic automaton whose acceptance condition is negated. We also show that the nondeterministic automaton that needs to be complemented inherits structural and semantic properties of the given alternating automaton. Using complementation constructions that exploit the inherited properties, we directly obtain instances of the scheme for various automata classes.

Furthermore, we instantiate the construction scheme to different classes of alternating automata. Some of the constructions that we obtain share similar technical details with previously proposed constructions such as, e.g., the ones described in [MH84,KV01,GO01,Boz07]. Some of them even produce the same nondeterministic Büchi automata modulo minor technical details. However, recasting these known constructions in such a way that they become instances of a general construction scheme increases their accessibility. In particular, correctness proofs become modular and simpler. Another benefit of utilizing the construction scheme is that differences and similarities between the translations for the different classes of alternating automata become apparent. We also present novel alternation-elimination constructions. These constructions are instances of our construction scheme and utilize new complementation

constructions for *eventually* 1-*way* nondeterministic co-Büchi automata. The novel technique used in these constructions enables new translations from important logics *with past operators* to nondeterministic automata whose worst-case sizes are surprisingly small. That is, compared to the worst-case sizes of the nondeterministic automata that we obtain from logics without past operators, their worst-sizes differ only by a small constant in the exponent. In the following paragraphs, we expand on two applications of these alternation-elimination constructions in more detail.

Past Operators for PSL One important application of the alternation-elimination constructions is the translation from PSL [Psl05] with past operators to nondeterministic Büchi automata (NBA). PSL is an IEEE standard and it is increasingly used in the hardware industry to formally express, validate, and verify requirements of circuit designs. The linear-time core of PSL[1] extends LTL with semi-extended regular expressions (SEREs), which are essentially regular expressions with an additional operator for expressing the intersection of languages. The prominence of PSL in industry over other specification languages like LTL [Pnu77], μLTL [BB89], and ETL [Wol83] is based on the fact that PSL balances well the competing needs of a specification language such as *expressiveness*, *succinctness*, and *implementability* [AFF+02]: PSL can describe all ω-regular properties, specifications are fairly easy to read and write in PSL in a concise way, and relevant verification problems such as model checking for PSL are automatically solvable in practice.

Although temporal operators that refer to the past have been found natural and useful when expressing temporal properties [LPZ85, KPV01, Mar03, CRS04, CRST06, SL10], the PSL standard supports temporal past operators only in a restrictive way. We define the logic PPSL as an extension of PSL with past operators. We also present examples that support the claim that the new past operators are natural and useful for describing properties that refers to the past. Moreover, we show that PPSL allows one to describe ω-regular languages more concisely than PSL. In particular, we define a family of ω-regular languages and prove that these languages can be described in PPSL exponentially more succinctly than in PSL. As a byproduct, we obtain a doubly exponential succinctness gap between PPSL and LTL, for the LTL-

[1] For the ease of exposition and similar to [BDBF+05, CRST06, PZ06], we identify PSL with its core. The core is *unclocked* and its semantics is only defined over infinite words.

expressible properties, that is, the ω-regular languages that are star-free (see, for example, [DG07]).

Taking all these benefits into account, one might ask for the reason why the PSL standard only supports past operators in a restrictive way. The design choice has already been made for the predecessor ForSpec [AFF+02] of PSL and has been justified by the argument that handling "arbitrary mixing of past and future operators results in nonnegligible implementation cost" [AFF+02]. One reason for this belief is that the best construction for translating PPSL to NBAs translates a formula of size n into an NBA with at most $\mathcal{O}(2^{4 \cdot 2^{4n} + 2^{2n}})$ states and is based on an alternation-elimination construction for 2-way Büchi automata, see [KPV01]. In contrast, the standard automata constructions for PSL translates a formula of size n into an NBA of size $\mathcal{O}(3^{2^{2n}})$ [BDBF+05, BFH05].

In this thesis, 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. We present a construction for PPSL that translates a formula of size n with m propositional variables into an NBA of size $\mathcal{O}(2^m \cdot 3^{2^{2n}})$. The difference between the upper bounds of the sizes of the resulting automata for PSL and PPSL is surprisingly small. Moreover, in symbolic model checkers like SMV [McM92] and its successors VIS [BHSV+96], RuleBase [BBDE+97], CadenceSMV [McM99], and NuSMV [CCG+02], the NBA is represented as labeled transition system. Here, we can adjust our construction to obtain the bound $\mathcal{O}(3^{2^{2n}})$, which matches exactly the bound that we obtain when translating PSL formulas into transition systems. That means that from a theoretical point of view, there is no reason for not supporting and using past operators.

We also show similar results for the logics DLTL [HT99] and RLTL [LS07] that are extensions of PSL. RLTL is of particular interest since every ω-regular expression can be translated into an RLTL formula of the same size, whereas the additional cost for the construction to NBAs is only a small factor in the exponent. We extend these logics with past operators, provide new translations constructions to NBAs, and show that the additional implementation costs are again small.

Nested-Word Logics Another application of our alternation-elimination construction scheme is the translation from the logic NWTL [AAB+08] to non-

deterministic nested-word automata (NWA). Nested words extend words by adding nested edges to the linearly ordered sequence of positions in the word. These nested edges connect *call positions* with *return positions* and are not allowed to cross. The data of many applications can be represented by nested words. For instance, in natural language processing, a sentence is viewed as a linear sequence of words while the underlying syntactic categories—the building blocks of sentences and the units of grammatical analysis—impart a nesting structure. In software verification, nested words model the control flow of sequential computations in typical programming languages with nested, and potentially recursive, invocations of program modules such as procedure calls. Another example is the representation of hierarchical data like XML documents as streams, that is, when viewing such a document as a linear sequence of characters, along with hierarchically nested edges connecting opening and closing tags.

For describing nested-word languages, Alur and Madhusudan use visibly-pushdown automata [AM04], a computational model whose access on the stack is input-driven. That is, a visibly-pushdown automaton or equivalently, nested-word automaton [AM09], is a finite automaton that accesses its stack only if the automaton processes a letter at a call or return position. In [AM04,AM09], Alur and Madhusudan show that this restricted class of pushdown automata enjoy similar properties as finite-state word automata such as being closed under union, intersection, and complementation. Furthermore, they provide efficient algorithms for deciding the membership, emptiness, and language inclusion of languages that are represented by nested-word automata. Due to these closure properties of NWAs, they call the class of languages definable by NWAs *regular*.

Apart from representing nested-word languages by computational models such as nested-word automata, the use of temporal logics to describe nested-word languages in a declarative way is often more natural. In [AEM04], Alur, Etessami, and Madhusudan introduce CaRet, a temporal logic over nested words that extends the well-known PLTL by operators such as *abstract next* or *abstract until* that are not only interpreted along the linear paths of a nested-word but also along paths that are implied by the additional nesting structure. The definition of CaRet is motivated by practice, namely, to describe requirements for recursive state machines. However, the theoretical question remains open whether CaRet is expressively complete with respect to first-order logic over nested word structures. Further development on the logic

CaRet led to the variants NWTL, NWTL$^+$, and CaRet+W in [AAB+08]. The authors show that all three logics are as expressively complete with respect to first-order logic over nested words. In [Boz07], Bozzelli presents the temporal logic μNWLTL, an extension of the well-known μLTL. She shows that this logic can describe all regular properties over nested words, and hence, the logic is even more expressive than any first-order logic over nested words. In subsequent papers, she also investigates variants of the logic CaRet [Boz08, Boz09]. For all of these logics, the authors provide translations to nested-word automata. So they reduce decision problems such as satisfiability and model checking to combinatorial problems for nested-word automata. However, the translations are non-trivial and hardly share any construction ideas.

We follow Vardi's two-step approach for translating logics over nested words to nested-word automata. We use alternating automata as an intermediate step and then apply our alternation-elimination scheme by providing several novel complementation constructions. Apart from the benefits we gain by using Vardi's two-step-approach, we also clarify and simplify many constructions. In particular, one of our constructions fixes a flaw in the translation from NWTL to NWAs given in [AAB+08]. Another construction provides an alternative, more modular translation from μNWLTL to NWAs than Bozelli's translation given in [Boz07]. Due to the modularity, our construction is simpler, easier to prove, and to implement. Moreover, for formulas with a restricted use of past operators, we can easily exchange the alternation-elimination part and obtain translations whose resulting NWAs are smaller than those obtained from Bozzelli's construction. Finally, we present a novel logic NWPSL that is more expressive than NWTL and show how to utilize our novel translations to easily obtain language-equivalent NWAs.

1.2 Contributions

We see our main contributions of this thesis as follows.

First, we improve and generalize Vardi's approach to eliminating alternation and formalize it as an alternation-elimination scheme. Our scheme is not only restricted to tree automata as in Vardi's case but also applies to graph automata and, in particular, to nested-word automata. Furthermore, instances of our scheme produce smaller worst-case results by an exponential factor in the size of the input.

Second, we provide new constructions for complementing restricted classes of 2-way automata. Together with the scheme, we obtain alternation-elimination constructions for several restricted classes of 2-way automata over words and nested words. The classes in consideration are important in the sense that they correspond to common temporal logics with past operators known from the literature and used in practice.

Third, we extend various temporal logics from the literature with past operators and present translations into alternating automata. Furthermore, we discuss the matter of succinctness that past operators provide. In particular, we extend the IEEE standardized temporal-logic PSL with past operators and show that—in contrary to a widely held belief—model checking systems with respect to this logic remains feasible. We also show that PSL with past operators is exponentially more succinct than PSL. We show similar results for the logics DLTL and RLTL. Moreover, we provide an alternative, mathematically clean way to translate NWTL into nested-word automata, correcting an error in a recently published construction. We also show how to extend NWTL by regular expressions, obtaining a new, more expressive logic and we provide constructions to nested-word automata. Furthermore, we present an alternative translation from μNWTL to nested-word automata. Our construction improves the state-of-the-art construction by a constant in the exponent. Furthermore, for formulas without past operators, our construction produces automata whose worst-case sizes improve upon the best-known construction by the factor $\log(n^2)/n^2$ in the exponent, where n is the size of the input.

1.3 Overview

This thesis is organized as follows.

In Chapter 2, we recall basic definitions. In Chapter 3, we present the alternation-elimination scheme and prove some properties of this scheme.

The next two chapters have a similar structure. In Chapter 4, we first develop complementation constructions for automata over words to obtaining alternation-elimination instances from our scheme. Then, we present extensions of temporal logics known from literature and present translations to nondeterministic automata, for these logics. Furthermore, we establish an exponential gap between certain logics and its extensions by past operators.

In Chapter 5, we first present novel complementation constructions that

translate nondeterministic automata into nondeterministic visibly-pushdown automata. Then, we present temporal logics known from literature and present translations to visibly-pushdown automata for these logics.

Chapter 2

Preliminaries

In this chapter, we fix notation and define the mathematical objects and the different automata classes that we use throughout this thesis. We also give a brief summary over some fundamental complementation constructions of nondeterministic automata that we use as running examples and as building blocks in the proofs in this thesis.

2.1 Graphs, Words, and Trees

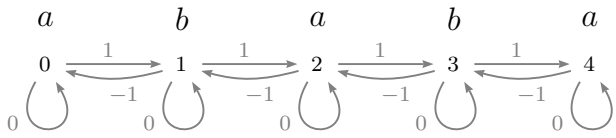
We write $\mathbb{N} := \{0, 1, 2, \ldots\}$ for the set of natural numbers and $\mathbb{N}_1 := \mathbb{N} \setminus \{0\}$ for the set of natural numbers that start with 1. For $n \in \mathbb{N}$, we write $[n]$ for the set $\{0, 1, \ldots, n-1\}$.

Graphs Let $\mathbb{D}$ be a finite set of *directions*. A *$\mathbb{D}$-graph* is an edge-labeled graph (V, E), where V is a set of *nodes* and $E = (E_d)_{d \in \mathbb{D}}$ is a family of *edges* $E_d \subseteq V \times V$ with label $d \in \mathbb{D}$. We denote the set of directions of the outgoing edges of a node $v \in V$ by $\mathbb{D}_v$, i.e., we write $\mathbb{D}_v := \{d \in \mathbb{D} \mid \text{there is a node } v' \text{ with } (v, v') \in E_d\}$. We write ε for the *empty graph*, i.e., the graph is of the form $(\emptyset, \emptyset)$. A graph is *finite* if the cardinality of the set of its nodes is finite. Otherwise, we call it *infinite*. A *pointed $\mathbb{D}$-graph* (G, v_I) is a $\mathbb{D}$-graph G with an *initial node* $v_I \in V$. A *$(\Sigma, \mathbb{D})$-graph* (V, E, v_I, λ) is pointed $\mathbb{D}$-graph (V, E, v_I) with a labeling $\lambda : V \to \Sigma$ of the nodes by elements from Σ. For a class of pointed $\mathbb{D}$-graphs $\mathcal{G}$, we write $\Sigma^{\mathcal{G}}$ for the set of all $(\Sigma, \mathbb{D})$-graphs (G, λ) with $G \in \mathcal{G}$.

Words A *word* over Σ is a $(\Sigma, \{-1, 0, 1\})$-graph of the form $(V, E, 0, w)$, where $V \in \{[n] \mid n \in \mathbb{N}\} \cup \{\mathbb{N}\}$, $E_1 = \{(i, j) \in V \times V \mid j = i + 1\}$, $E_{-1} =$

$\{(i,j) \in V \times V \mid j = i-1\}$, $E_0 = \{(i,i) \mid i \in V\}$. In the following, we denote a word over Σ by just giving its labeling function $w : V \to \Sigma$. We write Σ^* and Σ^ω for the set of all finite and infinite words over Σ, respectively. We denote the union of Σ^* and Σ^ω by Σ^∞. For a word w, we denote the ith letter by $w_i := w(i) \in \Sigma$, its *length* by $|w| := |V|$. We write $w_{i..}$ for the *suffix* $w_i w_{i+1} \dots$ of the word. We write $w_{i..j}$ for the *sub-word* $w_i w_{i+1} \dots w_j$. For convenience, $w_{i..j}$ denotes the empty graph ε if $j < i$. We write vw for the *concatenation* of the two words v and w.

Example 2.1 Consider the alphabet $\Sigma := \{a, b\}$ and the finite word $w : [5] \to \Sigma$ with $w_0 w_1 w_2 w_3 w_4 w_5 = ababa$. The following figure depicts the graph structure of this word.



Nested Words Nested words [AEM04, AM09] are words equipped with a hierarchical structure. This structure is imposed by letters that denote the start and the end of block structures. Prominent examples of nested words are, e.g., XML documents or source code of imperative programming languages with nested block structures. Formally, for an alphabet Σ, we define $\Sigma_i := \Sigma$ as the set of *internals*, $\Sigma_c := \{\langle a \mid a \in \Sigma\}$ as the set of *calls*, and $\Sigma_r := \{a\rangle \mid a \in \Sigma\}$ as the set of *returns*. The *tagged alphabet* of Σ is $\hat{\Sigma} := \Sigma_i \cup \Sigma_c \cup \Sigma_r$. We call a finite word $w \in \hat{\Sigma}^*$ *well-matched* if w is a word that can be build by the grammar $v ::= \varepsilon \mid av \mid cvrv$, where $a \in \Sigma_i$, $c \in \Sigma_c$, and $r \in \Sigma_r$.

A *nested word* over Σ is a $(\hat{\Sigma}, \{-2,-1,0,1,2\})$-graph of the form $(V, E, 0, w)$, where

1. $(V, (E_d)_{d \in \{-1,0,1\}}, 0, w)$ is a word,
2. $E_2 := \{(i,j) \mid w_i \in \Sigma_c, w_j \in \Sigma_r, w_{i..j}$ is well-matched, and any prefix $w_{i..k}$ is not well-matched, for any k with $i < k < k'\}$, and
3. $E_{-2} := \{(i,j) \mid (j,i) \in E_2\}$.

For convenience, we denote a nested word $(V, \rightsquigarrow, 0, w)$ by the tuple $(w, \rightsquigarrow)$ consisting of the word and the family of labeled edges.

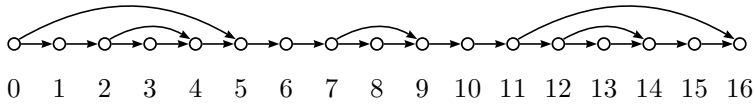
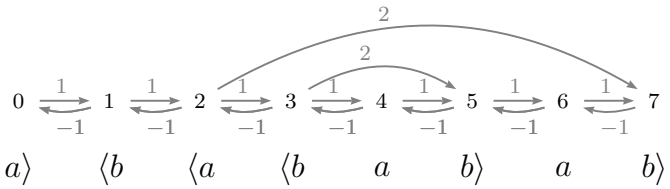


Figure 2.1: Graph structure of a nested word.

We call an element in $E_{-2} \cup E_2$ a *nested edge*. We call a position $i < |w|$ *internal*, *call*, and *return*, if w_i is a letter in Σ_i, Σ_c, and Σ_r, respectively. A call position i is *matched* if it has a *matching return* position j, i.e., there is an $j > i$ such that $(i, j) \in E_2$. Otherwise, the call position is *pending*. A return position j is matched if it has a *matching call* position i, i.e., there is an $i < j$ such that $(i, j) \in E_2$. Otherwise, the return position is *pending*. We call a position k *sync position* if the two sub-words $w_{0..k}$ and $w_{(k+1)..}$ of w are not connected by some edge in E_2. Formally, $k \in \mathbb{N}$ is a sync position if there is no $(i, j) \in E_2$ such that $i \leq k$ and $j > k$. For a position $i < |w|$, a *caller* of i is the greatest matched call position $j < i$ whose matching return position is after i. Formally, j is a caller for i if $j < i$ is a call and either i is a call and $w_{j..i-1}$ is well-matched, or i is not a call and $w_{j..i}$ is well-matched. We write $\hat{\Sigma}^*$ and $\hat{\Sigma}^\omega$ for the set of all finite and infinite nested words over Σ, respectively.

Example 2.2 Consider the alphabet $\Sigma := \{a, b\}$ and the finite nested word $(w, \rightsquigarrow)$ with $w : [8] \to \hat{\Sigma}$, where $w_{0..7} = a\rangle\langle b\langle a\langle bab\rangle ab\rangle$. The following figure depicts the graph structure of this nested word. For readability, we omit the 0-labeled self-loops. Note that position 0 and 1 are pending positions.



Example 2.3 Consider the alphabet $\Sigma := \{a\}$ and the nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^*$, where $w = \langle aa\langle aaa\rangle a\rangle a\rangle\langle aaa\rangle\langle a\langle a\langle aaa\rangle aa\rangle$. Figure 2.1 depicts the graph structure of w. For readability, we omit self-loops and edges with negative directions. The positions 1, 3, 8, 13, and 15 are internal positions. The positions 0, 2, 7, 10, 11, and 12 are call positions and 10 is a pending call. The positions 4, 5, 6, 9, 14, and 16 are return positions, where position 6 is pending. Position 11 is the caller position for position 12, 15, and 16. Finally, the positions 5, 6, 9, 10, and 16 are sync positions. □

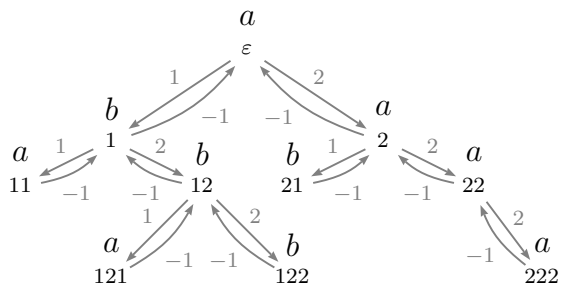


Figure 2.2: The graph structure of a tree.

Trees A *tree* over Σ is a $(\Sigma, \mathbb{N}_1 \cup \{0, -1\})$-graph (T, E, ε, t), where

1. $T \subseteq \mathbb{N}_1^*$ is *prefix-closed*, i.e., for every $x \in \mathbb{N}_1^*$ and $d \in \mathbb{N}_1$, if $xd \in T$ then $x \in T$,
2. $E_d \subseteq \{(x, xd) \mid x \in T \text{ and } xd \in T\}$, for $d \in \mathbb{N}_1$,
3. $E_0 := \{(x, x) \mid x \in T\}$, and
4. $E_{-1} := \{(x, y) \mid \text{there is a } d \in \mathbb{N} \text{ with } (y, x) \in E_d\}$.

The *root* of a tree is its initial node $\varepsilon \in T$. For an edge $(x, y) \in E_d$, for some $d \in \mathbb{N}_1$, the node y is the *child* of the node x. In the following, we will denote a tree over Σ by just giving its labeling function $t : T \to \Sigma$. A *path* in t is a sequence of nodes $\pi \in T^\infty$ such that $\pi_0 = \varepsilon$ and for every $i \in \mathbb{N}$, the node π_{i+1} is a child of π_i. We write $t(\pi)$ for the word $t(\pi_0)t(\pi_1)\ldots \in \Sigma^\infty$. For a set S, we denote all prefix-closed subsets of S^* by $S^\star$ and write $\Sigma^{\mathbb{N}_1^\star}$ for the set of all trees over Σ.

Example 2.4 Consider the alphabet $\Sigma := \{a, b\}$ and the prefix-closed set $T := \{x \in \mathbb{N}_1^* \mid |x| \leq 2\} \cup \{121, 122, 222\}$. We define the tree $t : T \to \Sigma$, where for every node $x \in T$, the labeling $t(x) = a$ if and only if the sum of the digits of x is even. Figure 2.2 depicts the graph structure of the tree t. For readability, we omit the 0-labeled self-loop of each node. The sequence of nodes $\pi := \varepsilon\ 1\ 12\ 122$ is a path in t. And the labeling of this path is $t(\pi) = abbb$.

2.2 Automata

Propositional Logic Let P be a finite set of atomic propositions. We denote the set of *Boolean formulas* over P by $\mathcal{B}(P)$, i.e., $\mathcal{B}(P)$ consists of all formulas that are inductively built by the grammar

$$\varphi ::= \mathsf{tt} \mid \mathsf{ff} \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi,$$

where tt and ff denote the Boolean constants tt and ff and p is a proposition in P. We write $\mathcal{B}^+(P)$ for the set of *positive Boolean formulas* that do not use the connective $\neg$. We write $\mathcal{B}^\vee(P)$ and $\mathcal{B}^\wedge(P)$ for the set of positive Boolean formulas that only use the connectives $\vee$ and $\wedge$, respectively. For a set $M \subseteq P$ and a formula $\varphi \in \mathcal{B}(P)$, we say that M *satisfies* φ and write $M \models \varphi$ if and only if φ evaluates to true when assigning true to the propositions in M and false to the propositions in $P \setminus M$. Moreover, we write $M \mid\!\equiv \varphi$ if and only if M is a *minimal model* of φ, i.e., $M \models \varphi$ and there is no $p \in M$ such that $M \setminus \{p\} \models b$. For a proposition $p \in P$ and a formula $\varphi \in \mathcal{B}^+(P)$, we say p *occurs in* φ if the formula φ has a minimal model M that contains p. If p occurs in φ, we also write $p \in \varphi$.

Automata In the following, we define $\mathbb{D}$-way alternating automata that operate over $(\Sigma, \mathbb{D})$-graphs. Intuitively, the automaton starts by reading the label of the initial node and then proceeds by moving its read-only head along the outgoing directions to the nodes whose labels are processed next. Formally, let $\mathbb{D}$ be a finite set of directions in which the read-only head of the automaton can move. A *$\mathbb{D}$-way alternating automaton* $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_I, A)$, where

- Q is a finite set of *states*,
- Σ is a finite input alphabet,
- $\delta = (\delta_D)_{D \subseteq \mathbb{D}}$ is a family of *transition functions* with $\delta_D : Q \times \Sigma \to \mathcal{B}^+(Q \times \mathbb{D})$, for $D \subseteq \mathbb{D}$,
- $q_I \in Q$ is an *initial state*, and
- $A \subseteq Q^\omega$ is an *acceptance condition*.

The *size* $|\mathcal{A}|$ of an automaton $\mathcal{A}$ is the number of its states.

Let $G := (V, E, v_I, \lambda)$ be a $(\Sigma, \mathbb{D})$-graph. We define $\mathbb{C} := Q \times V$ as the set of *configurations*. Intuitively, a configuration $(q, h) \in \mathbb{C}$ denotes the current state q and the position h of the read-only head in the input graph. A *run* of $\mathcal{A}$ on G is a tree $r : R \to \mathbb{C}$ such that $r(\varepsilon) = (q_I, v_I)$ and for every node $x \in R$ with $r(x) = (p, h)$, we have $M \models \delta_{\mathbb{D}_h}(p, \lambda(h))$, where

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

That is, the positive Boolean formula $\delta_{\mathbb{D}_h}(p, \lambda(h))$ specifies a constraint that has to be fulfilled by the successor states of the node denoted by the current head position h.

A path $\pi \in T^\infty$ in a run r with $r(\pi) = (q_0, h_0)(q_1, h_1) \dots$ is *accepting* if $q_0 q_1 \dots \in A$. The run r is *accepting* if every path in r is accepting. The *graph, word, finite-word*, and *nested word language* of $\mathcal{A}$ is, respectively, the set

$$\begin{aligned}
L(\mathcal{A}) &:= \{G \in \Sigma^{\mathcal{G}} \mid \text{there is an accepting run of } \mathcal{A} \text{ on } G\},\\
L^\omega(\mathcal{A}) &:= \{w \in \Sigma^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\},\\
L^*(\mathcal{A}) &:= \{w \in \Sigma^* \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\},\\
L^{\mathsf{nw}}(\mathcal{A}) &:= \{(w, \rightsquigarrow) \in \hat{\Sigma}^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } (w, \rightsquigarrow)\}.
\end{aligned}$$

We say that a graph, word, finite-word, and nested-word automaton $\mathcal{A}$ *accepts a language* L if its corresponding language $L(\mathcal{A})$, $L^\omega(\mathcal{A})$, $L^*(\mathcal{A})$, and $L^{\mathsf{nw}}(\mathcal{A})$ is equal to L, respectively. We call two automata $\mathcal{A}$ and $\mathcal{B}$ over graphs, words, finite-words, or nested-words *(language-)equivalent* if their corresponding languages are equal, i.e., $L(\mathcal{A}) = L(\mathcal{B})$, $L^\omega(\mathcal{A}) = L^\omega(\mathcal{B})$, $L^*(\mathcal{A}) = L^*(\mathcal{B})$, and $L^{\mathsf{nw}}(\mathcal{A}) = L^{\mathsf{nw}}(\mathcal{B})$, respectively.

In the following, we consider several special classes of automata. We start with the *directionality* of an automaton. Consider a $\mathbb{D}$-way alternating automaton $\mathcal{A}$ with $\mathbb{D} \subseteq \mathbb{Z}$. The automaton $\mathcal{A}$ is *1-way* if $\mathbb{D} \subseteq \mathbb{N}$. For these kind of automata, we abuse notation and omit writing the $\mathbb{D}$ component in the transitions and the runs. Otherwise, the automaton $\mathcal{A}$ is *2-way*. The automaton $\mathcal{A}$ is *locally 1-way* if for every state $q \in Q$, letter a, and set $D \subseteq \mathbb{D}$, we either have $\delta_D(q, a) \in \mathcal{B}^+(Q \times \mathbb{N})$ or $\delta_D(q, a) \in \mathcal{B}^+(Q \times (\mathbb{Z} \setminus \mathbb{N}))$. That is, in each processing step, the automaton moves its read-only head only forwards or only backwards.

Next, we distinguish between different *branching modes* of an automaton according to its transition function. We call $\mathcal{A}$ *existential* if $\delta_D : Q \times \Sigma \to$

$\mathcal{B}^{\vee}(Q \times \mathbb{D})$, for all $D \subseteq \mathbb{D}$. We call it *universal* if $\delta_D : Q \times \Sigma \to \mathcal{B}^{\wedge}(Q \times \mathbb{D})$, for all $D \subseteq \mathbb{D}$, for all $D \subseteq \mathbb{D}$. An automaton is *deterministic* if it is universal and existential. For $\mathbb{D} \subseteq \mathbb{Z}$, we call an automaton *nondeterministic* if for every state p, letter a, set $D \subseteq \mathbb{D}$, and minimal model M of $\delta_D(p, a)$, either (i) $|M| = 1$, (ii) for every $(q, d), (q', d') \in M$, we have $d > 0$ and $d \neq d'$, or (iii) for every $(q, d), (q', d') \in M$, we have $d < 0$ and $d \neq d'$. That is, the transition function cannot be satisfied by two different successor states along the same direction. Intuitively, a nondeterministic automaton annotates the nodes of the input by configurations while an existential automaton annotates only a path in the input by configurations.

Finally, we define the *type* of an automaton that specifies the acceptance condition of $\mathcal{A}$ in a finite way. Commonly used types of acceptance conditions are listed in Table 2.1. Here, $\mathsf{Inf}(\pi)$ is the set of states that occur infinitely often in an infinite path $\pi \in T^{\omega}$. The integer k is called *index* of $\mathcal{A}$. For instance, the acceptance condition A of a Büchi automaton $(Q, \Sigma, \delta, q_I, F, B)$ with $B \subseteq Q$ is defined as $A = \{\pi \in Q^{\omega} \mid \mathsf{Inf}(\pi) \cap B \neq \emptyset\}$.

Conventions For readability, we use the following standard abbreviations and conventions. Each four-letter acronym from the set

$$\{1, 2\} \times \{\text{A, E, U, N, D}\} \times \{\text{F, cF, B, C, G, cG, P, cP, R, S}\} \times \{\text{A}\}$$

refers to one particular automata class. The first letter corresponds to the directionality (1-way, 2-way), the second letter to the branching mode (alternating, existential, universal, nondeterministic, deterministic), and the third letter to the acceptance condition (finite, co-finite, Büchi, co-Büchi, generalized Büchi, generalized co-Büchi, parity, co-parity, Rabin, Streett). The fourth letter A stands for automaton. For instance, a 2URA is a 2-way universal Rabin automaton.

For existential and universal automata, we use the standard set notation to denote the disjunction or conjunction of all successors of a state, respectively. Formally, we write the transition function as $\delta_D : Q \times \Sigma \to 2^{Q \times \mathbb{D}}$, for all $D \subseteq \mathbb{D}$. Note that an empty set is regarded as ff for existential automata, whereas for universal automata, an empty set is regarded as tt. For a direction $d \in \mathbb{D}$, we write $\delta_D^d(R, a) := \bigcup_{r \in R} \{q \mid (q, d) \in \delta_D(r, a)\}$ for all *d-successors* of the set R, where $R \subseteq Q$, $D \subseteq \mathbb{D}$, and $a \in \Sigma$. For a transition function $\delta_D : Q \times \Sigma \to \mathcal{B}^{+}(Q \times \mathbb{D})$, for $D \subseteq \mathbb{D}$, the *dual transition function* is $\overline{\delta}_D : Q \times \Sigma \to \mathcal{B}^{+}(Q \times \mathbb{D})$,

type	finite description α, acceptance condition A
Finite co-finite	$\alpha = F \subseteq Q$ $A := \{\pi \in Q^* \mid \pi_{\lvert\pi\rvert-1} \in F\}$ $A := \{\pi \in Q^* \mid \pi_{\lvert\pi\rvert-1} \notin F\}$
Büchi co-Büchi	$\alpha = F \subseteq Q$ $A := Q^* \cup \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap F \neq \emptyset\}$ $A := Q^* \cup \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap F = \emptyset\}$
generalized Büchi generalized co-Büchi	$\alpha = \{F_0, \ldots, F_k\} \subseteq 2^Q$ $A := Q^* \cup \bigcap_{i\in[k]} \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap F_i = \emptyset\}$ $A := Q^* \cup \bigcup_{i\in[k]} \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap F_i \neq \emptyset\}$
parity co-parity	$\alpha = \{F_0, \ldots, F_{2k-1}\} \subseteq 2^Q$, where $F_0 \subseteq F_1 \subseteq \cdots \subseteq F_{2k-1}$ $A := Q^* \cup \{\pi \in Q^\omega \mid \mathsf{min}\{i \in [2k] \mid F_i \cap \mathsf{Inf}(\pi) \neq \emptyset\} \text{ is even}\}$ $A := Q^* \cup \{\pi \in Q^\omega \mid \mathsf{min}\{i \in [2k] \mid F_i \cap \mathsf{Inf}(\pi) \neq \emptyset\} \text{ is odd}\}$
Rabin Streett	$\alpha = \{(B_0, C_0), \ldots, (B_{k-1}, C_{k-1})\} \subseteq 2^Q \times 2^Q$ $A := Q^* \cup \bigcup_{i\in[k]} \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap B_i \neq \emptyset \text{ and } \mathsf{Inf}(\pi) \cap C_i = \emptyset\}$ $A := Q^* \cup \bigcap_{i\in[k]} \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) \cap B_i = \emptyset \text{ or } \mathsf{Inf}(\pi) \cap C_i \neq \emptyset\}$
Muller	$\alpha = \{M_0, \ldots, M_{k-1}\} \subseteq 2^Q$ $A := Q^* \cup \bigcup_{i\in[k]} \{\pi \in Q^\omega \mid \mathsf{Inf}(\pi) = M_i\}$

Table 2.1: Types of acceptance conditions.

where each $\overline{\delta}_D(q, a)$ is obtained from $\delta_D(q, a)$ by swapping all $\vee$ and $\wedge$ operators and the Boolean constants tt and ff, for all $q \in Q$, $D \subseteq \mathbb{D}$, and $a \in \Sigma$.

Restricted Automata Classes We introduce several restrictions on the definition of alternating automata that will be exploited in the automata constructions. Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, A)$ be a $\mathbb{D}$-way alternating automaton.

Weakness. The notion of weakness has been introduced in [MSS92]. Intuitively, weakness means that there is a partition on Q into either accepting or rejecting subsets $Q_0, \ldots, Q_{n-1}$, for some $n \in \mathbb{N}$, such that every path of any run gets trapped in exactly one Q_i, for some $i \in [n]$, and is accepting if and only if the partition Q_i is accepting.

Formally, we call a set of states $P \subseteq Q$ *accepting* if $\mathsf{Inf}(r(\pi)) \subseteq P$ implies $r(\pi) \in A$, for each run r and path π in r. We call a set $P \subseteq Q$ *rejecting* if $\mathsf{Inf}(r(\pi)) \subseteq P$ implies $r(\pi) \notin A$, for each run r and path π in r. The automaton

$\mathcal{A}$ is (inherently) *weak* if there is a partition on Q into sets $Q_0, \ldots, Q_{n-1}$, for some $n \in \mathbb{N}$, such that (a) for every $i \in [n]$, the set Q_i is either accepting or rejecting, and (b) for every $i, j \in [n]$, $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D \subseteq \mathbb{D}$, and $d \in \mathbb{D}$, if (q, d) occurs in $\delta_D(p, a)$ then $j \leq i$. The automaton is called *very weak* [GO01] (also known as 1-weak or linear) if additionally each Q_i, for every $i \in [n]$, is a singleton.

We append the letters W and V to the four-letter acronym of an automaton to characterize the automaton as weak and very weak, respectively. For instance, a V2ABA is a very weak 2ABA.

Eventually 1*-Way.* Let $\mathbb{D} \subseteq \mathbb{Z}$. Intuitively, being *eventually (strictly) 1-way* means that in every infinite path of any run, the automaton will eventually move its head position only forwards. Formally, the automaton $\mathcal{A}$ is eventually 1-way if there is a partition on Q into subsets $Q_0, Q_1, \ldots, Q_{n-1}$, for some $n \in \mathbb{N}$, such that for every $i, j \in [n]$, $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D \subseteq \mathbb{D}$, $d \in D$, and (q, d) occurs in $\delta_D(p, a)$, if (i is even and $d \leq 0$) or (i is odd and $d \geq 0$) then $j < i$. Intuitively, each subset Q_i represents states from which the automaton moves only forwards or only backwards. Since there are only finitely many such Q_is, the automaton may only change its direction finitely often and hence, it will eventually proceed only forwards.

Nested Word Automata

A *nested word automaton* (*NWA*) is a tuple $\mathcal{A} = (Q, S, \Sigma, \delta, q_I, F)$, where

- Q is a finite set of *states*,
- S is a finite set of *stack symbols* with $\bot \notin S$,
- Σ is a finite alphabet,
- $q_I \in Q$ is an *initial state*,
- $\delta = (\delta_x)_{x \in \{i,c,r\}}$ consists of three nondeterministic *transition functions* with
  1. $\delta_i : Q \times \Sigma_i \to 2^Q$,
  2. $\delta_c : Q \times \Sigma_c \to 2^{Q \times S}$,
  3. $\delta_r : Q \times (S \cup \{\bot\}) \times \Sigma_r \to 2^Q$,

- $F \subseteq Q$ is the set of *accepting states*.

A *run* of $\mathcal{A}$ on a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ is a sequence of configurations $(q_0, s_0)(q_1, s_1) \ldots \in (Q \times S)^\omega$ such that $q_0 = q_I$ and for every position $i \in \mathbb{N}$ in $(w, \rightsquigarrow)$,

1. if $w_i \in \Sigma_i$ then $q_{i+1} \in \delta_i(q_i, w_i)$,
2. if $w_i \in \Sigma_c$ then $(q_{i+1}, s_{i+1}) \in \delta_c(q_i, w_i)$,
3. if $w_i \in \Sigma_r$ and i is pending then $q_{i+1} \in \delta_r(q_i, \bot, w_i)$,
4. if $w_i \in \Sigma_r$ and i has a matching call j then $q_{i+1} \in \delta_r(q_i, s_{j+1}, w_i)$,

The run is *accepting* if some state $q \in F$ is visited infinitely often, i.e., we have $\mathsf{Inf}(q_0 q_1 \ldots) \cap F \neq \emptyset$. We denote the *nested word language* of the automaton $\mathcal{A}$ by

$$L^{\mathsf{nw}}(\mathcal{A}) := \{(w, \rightsquigarrow) \in \hat{\Sigma}^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } (w, \rightsquigarrow)\}.$$

Note that the NWA $\mathcal{A}$ implicitly uses a stack. Consider a run of the NWA $\mathcal{A}$. First, the stack is initialized with the bottom stack symbol $\bot$. Whenever $\mathcal{A}$ reads a letter in Σ_c it pushes a symbol $s \in S$ on the stack. Whenever $\mathcal{A}$ reads a letter from Σ_r it pops a symbol from the stack. In case the position is matched, it pops the top stack symbol that was previously pushed on the stack at the matching call position. In case the position is pending, it pops and pushes the bottom stack symbol $\bot$.

2.3 Complementation Constructions

In the following, we present complementation constructions from literature that we will use as basic building blocks throughout this thesis. Let $\mathbb{D} := \{-1, 0, 1\}$ be the set of direction.

The first construction is the standard *subset construction* [RS59,HU79], also known as powerset construction that complements the finite-word language of an 1EFA.

Theorem 2.5 *Let* $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ *be a 1EFA. Then, the 1DFA* $\mathcal{B} := (2^Q, \Sigma, \eta, \{q_I\}, 2^{Q \setminus F})$*, where*

$$\eta_D(R, a) := \delta_D^1(R, a)$$

for $R \in 2^Q$*,* $a \in \Sigma$*, and* $D \subseteq \mathbb{D}$*, accepts* $\Sigma^* \setminus L^*(\mathcal{A})$*.* □

Intuitively, the automaton $\mathcal{B}$ simulates each run of $\mathcal{A}$ simultaneously and accepts if all runs end in a non-accepting state. Observe that the theorem also holds if we replace the 1EFA and 1DFA by a 1EcFA and 1DcFA, respectively.

Another construction is the *2-way subset construction* [Var89] that complements the finite-word language of a 2-way EFA. In the next theorem, we present an improved version of Vardi's construction in terms of the resulting automata worst-cases sizes. We translate a 2EFA of size n into a 1EFA of size $(2^n)^2 + 1$ rather than $(2^n)^2 + 2^n$ as in Vardi's construction.

Theorem 2.6 *Let* $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ *be a 2EFA. Let* $\mathcal{B} := (2^Q \times 2^Q \cup \{q_I\}, \Sigma, \eta, q_I, G)$ *be a 1EFA, where* $G := (2^{Q \setminus F} \times 2^Q) \cup (\{q_I\} \cap (Q \setminus F))$*,*

$$\eta_D(q_I, a) := \{(R_0, R_1) \mid q_I \in R_0 \text{ and } \delta_D^d(R_0, a) \subseteq R_d, \text{ for all } d \in \{0, 1\}\},$$
$$\eta_D((R_{-1}, R_0), a) := \{(R_0, R_1) \mid \delta_D^d(R_0, a) \subseteq R_d, \text{ for all } d \in \mathbb{D}\},$$

for $R_{-1}, R_0, R_1 \in 2^Q$*,* $a \in \Sigma$*, and* $D \subseteq \mathbb{D}$*. Then,* $\mathcal{B}$ *accepts* $\Sigma^* \setminus L^*(\mathcal{A})$*.* □

Intuitively, the automaton $\mathcal{B}$ guesses a sequence $R_0 R_1 \dots R_n \in (2^Q)^*$ and checks that the sequence includes every run of $\mathcal{A}$ on the input word. That is, if the sequence $(q_0, h_0)(q_1, h_1) \dots (q_n, h_n) \in (Q \times \mathbb{N})^*$ is a run of $\mathcal{A}$ then $q_i \in R_{h_i}$, for all $i \in [n+1]$. Obviously, the sequence $R_0 R_1 \dots R_n$ contains every run, if the following *maximality condition* holds: for every position $i \in [n+1]$ and state $p \in R_i$, all d-successors are contained in the adjacent sets R_{i+d} (if they exists).

The following construction is the *breakpoint construction* [BJW05, MH84]. The construction is used to complement the infinite-word language of a 1ECA.

Theorem 2.7 *Let* $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ *be an 1ECA. Then, the 1DBA* $\mathcal{B} := (2^Q \times 2^F, \Sigma, \eta, q_I, 2^Q \times \{\emptyset\})$*, where*

$$\eta_D((R, \emptyset), a) := (\delta_D(R, a), \delta_D(R, a) \setminus F) \text{ and}$$
$$\eta_D((R, S), a) := (\delta_D(R, a), \delta_D(S, a)),$$

for $R \in 2^Q$*,* $S \in 2^F \setminus \{\emptyset\}$*,* $a \in \Sigma$*, and* $D \subseteq \mathbb{D}$ *accepts* $\Sigma^\omega \setminus L^\omega(\mathcal{A})$*.* □

Intuitively, the automaton $\mathcal{B}$ simulates each run of $\mathcal{A}$ simultaneously with its R component by using the subset construction. The S component of the state space is used to check that no run of $\mathcal{A}$ accepts by the co-Büchi condition. Roughly speaking, $\mathcal{B}$ stores states in S that owe a visit to F. A configuration with $S = \emptyset$ is a *breakpoint*. By filling up S with all states from R at breakpoints and removing F-states from the S component, in every processing step, the automaton $\mathcal{B}$ ensures that all states visit F before the next breakpoint is reached. Hence, $\mathcal{B}$ visits infinitely many breakpoints if and only if all runs of $\mathcal{A}$ visit F infinitely often.

Note that the 1DBA $\mathcal{B}$ has only 3^n states, where n is the size of the 1ECA $\mathcal{A}$. The reason is that a state (R, S) of $\mathcal{B}$ can be represented by a vector $v \in 3^n$, where every component of v corresponds to a state of $\mathcal{A}$ that is (i) contained in R and S, (ii) contained in R but not in S, or (iii) neither contained in R nor in S.

Chapter 3

Alternation-Elimination Scheme

In this chapter, we present a general construction scheme for removing alternation of $\mathbb{D}$-way alternating automata. The construction scheme is general in the sense that it can be instantiated for different classes of alternating automata. We provide such instances in the following chapters.

We proceed as follows. We first give an intuition of the construction scheme. Then, we show how to encode run trees of alternating automata as labeling functions of the input graphs of those automata. Afterwards, we present the construction scheme and some of its properties.

3.1 Overview on Construction Scheme

In this section, we introduce the idea of the alternation-elimination scheme. We present this from a game-theoretic point of view. Consider a $\mathbb{D}$-way alternating automaton $\mathcal{A}$, an input graph G, and the two players called *Automaton* and *Refuter*. Player Automaton claims that $\mathcal{A}$ accepts G, whereas player Refuter tries to refute this claim. They compete against each other in the following game.

1. First, player Automaton suggests a tree and claims that this tree is an accepting run of $\mathcal{A}$ on G. That is, the tree is a run of $\mathcal{A}$ on G and every infinite path in the tree is accepting.

2. Second, player Refuter examines Automaton's suggested tree and checks whether the claim is true. In particular, Refuter can refute the claim in two ways. Either, Refuter shows that the tree is not a run of $\mathcal{A}$ on G, i.e., Refuter picks a reachable node such that the labels of the node's

$$
\begin{aligned}
& \text{Alternating automaton } \mathcal{A} = (Q, \Sigma, \delta, q_I, A) \text{ accepts } G \\
\Leftrightarrow\ & \exists t : t \in \mathsf{runs}(\mathcal{A}, G) \wedge \forall \pi \in \mathsf{paths}(t) : t(\pi) \in A \\
\Leftrightarrow\ & \exists t : \neg(t \notin \mathsf{runs}(\mathcal{A}, G) \vee \exists \pi \in \mathsf{paths}(t) : t(\pi) \notin A) \\
\Leftrightarrow\ & \exists t : \neg\big(\text{the existential } \textit{refuter} \text{ automaton } \mathcal{B} \text{ accepts } (G, t)\big) \\
\Leftrightarrow\ & \exists t : \text{the nondeterministic automaton } \mathcal{C} \text{ accepts } (G, t) \\
\Leftrightarrow\ & \text{the nondeterministic automaton } \mathcal{D} \text{ accepts } G
\end{aligned}
$$

Figure 3.1: Overview of the alternation-elimination scheme.

children do not satisfy the transition function of $\mathcal{A}$. Or, Refuter identifies an infinite path in the tree that is not accepting.

Obviously, $\mathcal{A}$ accepts G if and only if player Automaton can win this game.

With this game in mind, we can describe the overall idea of the scheme as follows. First, we encode player Refuter's strategy by a existential automaton $\mathcal{B}$. The automaton $\mathcal{B}$ reads the input graph G and player Automaton's suggested tree t. The tricky part of this step is to represent t in such a way such that its representation has the same structure as the input graph G. The objective of the existential automaton $\mathcal{B}$ is twofold. It looks for a node in t that witnesses that t is not a run of $\mathcal{A}$ on G. Alternatively, it looks for an infinite path in t that is rejecting. In a second step, we translate $\mathcal{B}$ into a nondeterministic automaton $\mathcal{C}$ that accepts the complement of the language of $\mathcal{B}$. That is, $\mathcal{C}$ accepts a graph G with its suggested t if and only if t is an accepting run of $\mathcal{A}$ on G. Finally, we define $\mathcal{D}$ as the projection of $\mathcal{C}$ on G, which is a standard operation on nondeterministic automata. The resulting nondeterministic automaton $\mathcal{D}$ accepts G if and only if there is a tree t that is an accepting run of $\mathcal{A}$ on G. A sketch of this general overview is depicted in Table 3.1, where $\mathsf{runs}(\mathcal{A}, G)$ denotes the set of all runs of $\mathcal{A}$ on G and $\mathsf{paths}(t)$ denotes the set of all infinite paths in t.

In the following sections, we present this idea formally. Before going into the details of the construction scheme, we start with some preparatory work.

3.2 Memoryless Strategies as Input

In this section, we show how to represent a memoryless run of an alternating automaton in such a way such that the representation has the same structure as the input graph that is read by the alternating automaton. We need this kind of representation of a run as a prerequisite for the alternation-elimination scheme. We remark that the construction of the representation is based on constructions given in [Var88, Var98, KPV01]. In this thesis, we simplify the presentation of their representation of a memoryless run. A memoryless run is essentially a so-called positional strategy of player Automaton and we view the strategy as a function rather than as a relation. Furthermore, we generalize their representation to be able to represent memoryless runs on general graph inputs rather than just word or tree inputs.

We start with the formal definition. Let $\mathbb{D}$ be a set of directions, $\mathcal{A} = (Q, \Sigma, \delta, q_I, A)$ a $\mathbb{D}$-way alternating automaton, and $r : R \to Q \times V$ a run of $\mathcal{A}$ on a $(\Sigma, \mathbb{D})$-graph (V, E, v_I, λ). The run r is called *memoryless*[1] if all equally labeled nodes have isomorphic subtrees, i.e., for every $x, y \in R$ and $z \in \mathbb{N}_1^*$, if $r(x) = r(y)$ then

1. $xz \in R$ if and only if $yz \in R$, and
2. if $xz \in R$ then $r(xz) = r(yz)$.

We define

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

We call $\mathcal{A}$ *memoryless* if $L(\mathcal{A}) = M(\mathcal{A})$.

Obviously, we have $M(\mathcal{A}) \subseteq L(\mathcal{A})$. For a $\mathbb{D}$-way alternating automaton $\mathcal{A}$ with Büchi, co-Büchi, parity, co-parity, or Rabin acceptance condition, we also have the converse $M(\mathcal{A}) \supseteq L(\mathcal{A})$, see [EJ91, Jut97, Var98, Zie98] for more details. Note however that the converse does not hold in general.

In the following, we represent a memoryless run by a new labeling of the input graph. Since children of equally labeled nodes in a memoryless run

[1] The choice of the term "memoryless" becomes clear when viewing a run of an alternating automaton as a representation of a strategy of the first player in a two-person infinite game [MS87]. A memoryless run encodes a memoryless strategy (also known as a positional strategy) of the first player, i.e., a strategy that does not take the history of a play into account.

$r : R \to Q \times V$ are also equally labeled, we represent the memoryless run r by a successor function $\tau : (Q \times V) \to 2^{Q \times \mathbb{D}}$, where

$$\tau(q, v) := \big\{(q', d) \in Q \times \mathbb{D} \mid \text{there is a node } x \text{ with a child } y \text{ in } R \text{ such that } r(x) = (q, v), r(y) = (q', v'), \text{ and } (v, v') \in E_d\big\}.$$

That is, we map a configuration (q, v) to the set of labels of the children of some node that is labeled by (q, v). By "currying" the function τ, we obtain the function $\sigma : V \to (Q \to 2^{Q \times \mathbb{D}})$ that is isomorphic to τ. We call σ the *representation* of the memoryless run r. Note that this representation is unique, i.e., every representation of r is isomorphic to σ.

Example 3.1 We list some representations of memoryless runs for special cases of input graphs. Note that the representations are labeling functions of graphs that have exactly the same structure as the input graphs of the alternating automata.

- If the input of $\mathcal{A}$ is a finite word $w = w_0 \dots w_{n-1} \in \Sigma^*$ then a run is a tree $r : R \to Q \times [n]$. If r is memoryless, we represent the run by a finite word $\sigma : [n] \to (Q \to 2^{Q \times \mathbb{D}})$, where $\mathbb{D} = \{-1, 0, 1\}$ and for $i \in [n]$, the labeling $\sigma(i) : Q \to 2^{Q \times \mathbb{D}}$ is the function that maps a state q to the set that contains all tuples of the from (q', d) if and only if in the run r, the automaton $\mathcal{A}$ visits q having head position i and moves to state q' and head position $i + d$.

- If the input of $\mathcal{A}$ is an infinite word $w \in \Sigma^\omega$ then a run is a tree $r : R \to Q \times \mathbb{N}$. If r is memoryless, we represent the run by an infinite word $\sigma : \mathbb{N} \to (Q \to 2^{Q \times \mathbb{D}})$, where $\mathbb{D} = \{-1, 0, 1\}$.

- If the input of $\mathcal{A}$ is a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ then a run is a tree $r : R \to Q \times \mathbb{N}$. If r is memoryless, we represent the run by a nested word $(\sigma, \rightsquigarrow)$ with $\sigma : \mathbb{N} \to (Q \to 2^{Q \times \mathbb{D}})$, where $\mathbb{D} = \{-2, -1, 0, 1, 2\}$.

- If the input of $\mathcal{A}$ is an tree $t : T \to \Sigma$ then a run is a tree $r : R \to (Q \times T)$. If r is memoryless, we represent the run by a tree $\sigma : T \to (Q \to 2^{Q \times \mathbb{D}})$, where $\mathbb{D} = \{-1, 0, \dots, n\}$ and n is the greatest direction that occurs in the transition function of $\mathcal{A}$. □

3.3 Reduction to Complementation

Based on the representation of memoryless runs as input graphs, we formally introduce the alternation-elimination scheme.

Definition 3.2 Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, A)$ be a $\mathbb{D}$-way alternating automaton and $\Gamma := Q \to 2^{Q \times \mathbb{D}}$. From $\mathcal{A}$ we construct the $\mathbb{D}$-way existential *refuter automaton* $\mathcal{B}$ of $\mathcal{A}$ that is defined as

$$\mathcal{B} := (Q, \Sigma \times \Gamma, \eta, q_I, Q^* \cup (Q^\omega \setminus A)).$$

For $q \in Q$, $D \subseteq \mathbb{D}$, and $(a, g) \in \Sigma \times \Gamma$, we define the transition function as

$$\eta_D\big(q, (a, g)\big) := \begin{cases} g(q) & \text{if } g(q) \models \delta_D(q, a), \\ \mathsf{tt} & \text{otherwise.} \end{cases}$$

□

Intuitively, the existential automaton $\mathcal{B}$ from Definition 3.2 works as follows. It reads the input graph G and the run representation of the $\mathbb{D}$-way alternating automaton $\mathcal{A}$ and inspects one single path in the run representation. The refuter automaton $\mathcal{B}$ accepts the input if one of the following holds. (a) The inspected path is finite and leads to a node in the run that witnesses that the run is broken. That is, the labels of its children do not satisfy the transition function of $\mathcal{A}$. (b) The inspected path is infinite and its labeling yields a rejecting sequence with respect to the acceptance condition of $\mathcal{A}$.

Throughout this section, we fix the $\mathbb{D}$-way alternating automaton $\mathcal{A}$ and its refuter automaton $\mathcal{B}$ as defined in Definition 3.2. Moreover, we abbreviate the function space $Q \to 2^{Q \times \mathbb{D}}$ by Γ.

The next lemma is at the core of the results of this chapter. It states that the refuter automaton $\mathcal{B}$ rejects an input if and only if the input consists of a graph G and a representation of an accepting memoryless run of $\mathcal{A}$ on G. We use the following notation. For a graph $G \in (\Sigma \times \Gamma)^{\mathcal{G}}$, we write $G_\Sigma \in \Sigma^{\mathcal{G}}$ and $G_\Gamma \in \Gamma^{\mathcal{G}}$ to denote the graphs with the projections of their labeling of the nodes on Σ and Γ, respectively.

Lemma 3.3 *For every graph $G \in (\Sigma \times \Gamma)^{\mathcal{G}}$, we have*

$$G \notin L(\mathcal{B}) \quad \textit{iff} \quad G_\Gamma \textit{ is a representation of an accepting run of } \mathcal{A} \textit{ on } G_\Sigma.$$ □

Proof Let $G = (V, E, v_I, \lambda) \in (\Sigma \times \Gamma)^{\mathcal{G}}$ be a graph. We write λ_Σ and λ_Γ for the projection of λ on the alphabets Σ and Γ, respectively.

We first prove the *if* direction. Let $t : T \to Q \times V$ be an accepting memoryless run of $\mathcal{A}$ on G_Σ such that G_Γ is a representation of t. For the sake of contradiction, assume there is an accepting run r of $\mathcal{B}$ on G. We show that t contains a rejecting path π, which contradicts the fact that t is accepting. We consider the following two cases.

Case 1. Suppose that r is infinite. Let $r := (q_0, v_0)(q_1, v_1) \ldots \in (Q \times V)^\omega$ with $q_0 q_1 \ldots \notin A$. Moreover, we have $(q_0, v_0) = (q_I, v_I)$ and $\lambda_\Gamma(v_i)(q_i) \models\!\!\equiv \delta_{\mathbb{D}_{v_i}}(q_i, \lambda_\Sigma(v_i))$, for all $i \in \mathbb{N}$. We recursively construct the rejecting path $\pi \in V^\omega$ in T.

- We define $\pi_0 := \varepsilon$. By definition of a run of $\mathcal{A}$ on G_Σ, we have $t(\pi_0) = (q_I, v_I) = (q_0, v_0)$.
- For $i > 0$, we define π_i as some child of π_{i-1} that is labeled by $t(\pi_i) = (q_i, v_i)$. We show that the child π_i exists. Since r is a run of $\mathcal{B}$, we have $(q_i, v_i) \in \lambda_\Gamma(v_{i-1})(q_{i-1})$. Since G_Γ is a representation of the memoryless run t, the set $\lambda_\Gamma(v_{i-1})(q_{i-1})$ contains all labels of the children of any node $x \in T$ that is labeled by (q_{i-1}, v_{i-1}). Thus, there is a child $\pi_i \in T$ of π_{i-1} that is labeled by (q_i, v_i).

Since π is a path in T with $t(\pi) = r$ and $q_0 q_1 \ldots \notin A$, the run t is not accepting.

Case 2. Suppose that r is finite. Let $r := (q_0, v_0)(q_1, v_1) \ldots (q_{n-1}, v_{n-1}) \in (Q \times V)^*$. Moreover, we have $(q_0, v_0) = (q_I, v_I)$, $\lambda_\Gamma(v_i)(q_i) \models\!\!\equiv \delta_{\mathbb{D}_{v_i}}(q_i, \lambda_\Sigma(v_i))$, for all $i \in [n]$, and $\lambda_\Gamma(v_{n-1})(q_{n-1}) \not\models\!\!\equiv \delta_{\mathbb{D}_{v_{n-1}}}(q_{n-1}, \lambda_\Sigma(v_{n-1}))$. We recursively construct the rejecting path $\pi \in V^*$ in T. Using the construction from Case 1, we obtain a path $\pi = \pi_0 \pi_1 \ldots \pi_{n-1}$ in T with $t(\pi) = r$. Since G_Γ is a representation of the memoryless run t, the set $\lambda_\Gamma(v_{n-1})(q_{n-1})$ contains all labels of the children of the node $\pi_{n-1} \in T$ that is labeled by (q_{n-1}, v_{n-1}). Since $\lambda_\Gamma(v_{n-1})(q_{n-1}) \not\models\!\!\equiv \delta_{\mathbb{D}_{v_{n-1}}}(q_{n-1}, \lambda_\Sigma(v_{n-1}))$, the tree t is not a valid run of $\mathcal{A}$ on G_Σ.

Now, we prove the *only if* direction by contraposition. By assumption, G_Γ is not a representation of an accepting run $\mathcal{A}$ on G_Σ. We make a case distinction.

Case 1. Suppose G_Γ is not a representation of a run of $\mathcal{A}$ on G_Σ. Consider a tree $t : T \to Q \times V$ with $t(\varepsilon) = (q_I, v_I)$ such that G_Γ is a representation of t. It is easy to verify that such a tree exists and we omit this proof step. Since

t is not a run of $\mathcal{A}$ on G_Σ, there is a node $x \in T$ with label $t(x) = (q, v)$, for some $q \in Q$ and $v \in V$, such that the set

$$\{(q', d) \in Q \times \mathbb{D} \mid y \text{ is a child of } x, t(y) = (q', v'), \text{ and } (v, v') \in E_d\}$$

is not a minimal model of $\delta_{\mathbb{D}_v}(q, \lambda_\Sigma(v))$. Let $x \in T$ be chosen in that way such that $k := |x|$ is minimal. Now, we define an accepting run r of $\mathcal{B}$ on G. For $i \in [k]$, we define $r_i := (q_i, v_i) := t(x_{0..i})$. By the minimality of $|x|$, we have $q_{i+1} \in \lambda_\Gamma(v_i)(q_i)$ and $\lambda_\Gamma(v_i)(q_i) \models \delta_{\mathbb{D}_{v_i}}(q_i, \lambda_\Sigma(v_i))$, for every $i \in [k-1]$. Since $\lambda_\Gamma(v_{k-1})(q_{k-1}) \not\models \delta_{\mathbb{D}_{v_{i-1}}}(q_{k-1}, \lambda_\Sigma(v_{k-1}))$, we conclude that r is an accepting run of $\mathcal{B}$ on G.

Case 2. Suppose G_Γ is a representation of a run of $\mathcal{A}$ on G_Σ. Consider a run $t : T \to Q \times V$ of $\mathcal{A}$ on G_Σ such that G_Γ is a representation of t. By assumption, the run t is rejecting. Consider a rejecting path π in T. Note that the path π is infinite. We claim that $(q_0, v_0)(q_1, v_1) \ldots := t(\pi)$ is an accepting run of $\mathcal{B}$ on G. Since $t(\varepsilon) = (q_I, v_I)$ and $q_{i+1} \in \lambda_\Gamma(v_i)(q_i)$ and $\lambda_\Gamma(v_i)(q_i) \models \delta_{\mathbb{D}_{v_i}}(q_i, \lambda_\Sigma(v_i))$, for all $i \in \mathbb{N}$, the sequence $t(\pi)$ is a run of $\mathcal{A}$ on G_Σ. Since $q_0 q_1 \ldots \notin A$, the run $t(\pi)$ is accepting. ∎

The next theorem combines the results of this chapter. Whenever $G(\mathcal{A}) = M(\mathcal{A})$, the problem of eliminating the alternation of $\mathcal{A}$ (i.e., constructing a language-equivalent nondeterministic automaton) reduces to the problem of complementing the existential refuter automaton $\mathcal{B}$.

Theorem 3.4 *If* $L(\mathcal{A}) = M(\mathcal{A})$ *then* $L(\mathcal{A}) = \{G_\Sigma \mid G \notin L(\mathcal{B})\}$. □

Proof Consider a graph $G \in \Sigma^{\mathcal{G}}$.

	$G \in L(\mathcal{A})$
iff	$G \in M(\mathcal{A})$
iff	There is a graph $H \in (\Sigma \times \Gamma)^{\mathcal{G}}$ such that $H_\Sigma = G$ and H_Γ is the representation of some memoryless accepting run r of $\mathcal{A}$ on H_Σ.
iff	There is a graph $H \in (\Sigma \times \Gamma)^{\mathcal{G}}$ such that $H_\Sigma = G$ and $H \notin L(\mathcal{B})$.
iff	$G \in \{H_\Sigma \mid H \notin L(\mathcal{B})\}$

∎

Example 3.5 The following examples are simple instances of the scheme.

- Consider an AFA $\mathcal{A}$ and a construction $\tau : \text{EcFA} \to \text{NFA}$ that complements the word or tree language of an EcFA. According to the scheme, we can translate $\mathcal{A}$ into a language-equivalent NFA as follows. First, we construct the refuter EcFA $\mathcal{B} = (Q, \Sigma \times \Gamma, \eta, q_I, F)$ as described in Definition 3.2. Then, we project $\tau(\mathcal{B})$ on Σ and obtain an NFA that is language-equivalent to $\mathcal{A}$.

- Consider an ABA $\mathcal{A}$ and a construction $\tau : \text{ECA} \to \text{NBA}$ that complements the word or tree language of an ECA. According to the scheme, we can translate $\mathcal{A}$ into a language-equivalent NBA as follows. First, we construct the refuter NCA $\mathcal{B} = (Q, \Sigma \times \Gamma, \eta, q_I, F)$ as described in Definition 3.2. Then, we project $\tau(\mathcal{B})$ on Σ and obtain an NBA that is language-equivalent to $\mathcal{A}$.

- Consider an ABA $\mathcal{A}$ and a construction $\tau : \text{ECA} \to \text{NWA}$ that complements the nested word language of an ECA. According to the scheme, we can translate $\mathcal{A}$ into a language-equivalent NWA as follows. First, we construct the refuter ECA $\mathcal{B} = (Q, \Sigma \times \Gamma, \eta, q_I, F)$ as described in Definition 3.2. Then, we project $\tau(\mathcal{B})$ on Σ and obtain an NWA that is language-equivalent to $\mathcal{A}$. □

3.4 Inherited Properties

In the next lemma, we prove several properties that are inherited by the existential automaton $\mathcal{B}$ from the alternating automaton $\mathcal{A}$. We exploit these properties in the alternation-elimination construction. Intuitively, the properties discussed in Lemma 3.6 describe properties that are fulfilled by all runs of the given alternating automaton $\mathcal{A}$. By construction, a run of the nondeterministic automaton $\mathcal{B}$ corresponds to some path of some run of $\mathcal{A}$. So, every run of $\mathcal{B}$ also inherits these properties.

Lemma 3.6 *Let $\mathcal{A}$ be a $\mathbb{D}$-way alternating automaton and $\mathcal{B}$ the $\mathbb{D}$-way nondeterministic automaton defined in Definition 3.2. Then the following holds.*

1. *If $\mathcal{A}$ is weak then $\mathcal{B}$ is weak.*

2. *If $\mathcal{A}$ is very weak then $\mathcal{B}$ is very weak.*

3. *If $\mathcal{A}$ is locally 1-way then $\mathcal{B}$ is locally 1-way.*

4. *If $\mathcal{A}$ is eventually 1-way then $\mathcal{B}$ is eventually 1-way.* □

Proof Let $\mathcal{A} := (Q, \Sigma, \delta, q_I, A)$ and $\mathcal{B} := (Q, \Sigma \times \Gamma, \eta, q_I, Q^\omega \setminus A)$ be the two automata defined in Definition 3.2.

We prove that $\mathcal{B}$ inherits the weakness and very weakness property from $\mathcal{A}$. Let $Q_0, \ldots, Q_{n-1}$ be a partition of $\mathcal{A}$'s state space such that (a) for every $i \in [n]$, Q_i is either accepting or rejecting, and (b) for every $i, j \in [n]$, $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D \subseteq \mathbb{D}$, and $d \in \mathbb{D}$, if (q, d) occurs in $\delta_D(p, a)$ then $j \leq i$. We claim that $Q_0, \ldots, Q_{n-1}$ is a partition of $\mathcal{B}$'s state space such that for every $i, j \in [n]$, $p \in Q_i$, $q \in Q_j$, $(a, g) \in \Sigma \times (Q \to 2^{Q \times \mathbb{D}})$, $D \subseteq \mathbb{D}$, and $d \in \mathbb{D}$: if (q, d) occurs in $\eta_D(p, (a, g))$ then $j \leq i$. Let $i, j \in [n]$, $D \subseteq \mathbb{D}$, and $(a, g) \in \Sigma \times \Gamma$. Consider a tuple $(q, d) \in Q_j \times \mathbb{D}$ that occurs in $\eta_D(p, (a, g))$, for some $p \in Q_i$. By definition of the transition function of $\mathcal{B}$, we have $(q, d) \in g(p)$ and $g(p) \models \delta_D(p, a)$. Thus, (q, d) occurs in $\delta_D(p, a)$. Since $Q_0, \ldots, Q_{n-1}$ is a partition of $\mathcal{A}$'s state space, we obtain $j \leq i$. Note that the arguments in this proof are also valid if the Q_is are singletons.

Next, we show that $\mathcal{B}$ is locally 1-way if $\mathcal{A}$ is locally 1-way. Consider a transition $\eta_D(p, (a, g))$, for some $p \in Q$, $D \subseteq \mathbb{D}$, and $(a, g) \in \Sigma \times \Gamma$. We have two cases. (a) If $g(p) \not\models \delta_D(p, a)$ then $\eta_D(q, a) = \mathsf{tt} \in \mathbb{B}^+(Q \times \mathbb{N})$. (b) If $g(p) \models \delta_D(p, a)$ then $(q, d) \in g(p)$. Thus, (q, d) occurs in $\delta_D(p, a)$. We conclude that $\eta_D(q, a)$ is either in $\mathbb{B}^+(Q \times \mathbb{N})$ or in $\mathbb{B}^+(Q \times \mathbb{Z} \setminus \mathbb{N})$.

Finally, we prove that $\mathcal{B}$ is eventually 1-way if $\mathcal{A}$ is eventually 1-way. Let $Q_0, Q_1, \ldots, Q_{n-1}$, for some $n \in \mathbb{N}$, be a partitioning of the states of $\mathcal{A}$ such that for every $i, j \in \mathbb{N}$, $p \in Q_i$, $q \in Q_j$, $a \in \Sigma$, $D \subseteq \mathbb{D}$, and $d \leq 0$, if (q, d) occurs in $\delta_D(p, a)$ then $j < i$. We use the same partitioning to show that $\mathcal{B}$ is eventually 1-weak. Let $i, j \in \mathbb{N}$, $p \in Q_i$, $q \in Q_j$, $(a, g) \in \Sigma \times \Gamma$, $D \subseteq \mathbb{D}$, and $d \leq 0$. Assume (q, d) occurs in $\eta_D(p, (a, g))$. Then, $(q, d) \in g(p)$ and $g(p) \models \delta_D(p, a)$. Hence, (q, d) occurs in $\delta_D(p, a)$ and we conclude that $j < i$. ■

Chapter 4

Translating Logics over Words to Automata

In this chapter, we present translations from various classes of alternating automata over infinite words to 1-way nondeterministic Büchi automata. We obtain these translations from our alternation-elimination scheme by providing complementation constructions for the corresponding classes of nondeterministic automata over infinite words. We utilize these alternation-elimination constructions for translating various temporal logics over infinite words to 1-way nondeterministic Büchi automata.

We proceed as follows. First, we present complementation constructions for different classes of nondeterministic automata over infinite words. Then, we introduce the linear-time temporal logic PPSL, present a translation from PPSL to 1NBAs, and provide succinctness results. Finally, we introduce extensions of PPSL and show how to translate these to 1-way nondeterministic Büchi automata using our alternation-elimination scheme.

4.1 Complementation Constructions

In this section, we present several novel constructions for complementing the languages of nondeterministic automata over infinite words. The constructions translate various classes of nondeterministic automata into 1-way nondeterministic Büchi automata. Table 4.1 depicts the blow-ups of these constructions, where n is the size of the nondeterministic automaton and k its index. For instance, in Theorem 4.6, we present a complementation construction that translates an eventually 1-way very-weak nondeterministic co-Büchi automaton over infinite words into a 1-way nondeterministic Büchi automaton of size

	V2NCA	2NCA	2NPA
1-way	$\mathcal{O}(2^n n)$ Theorem 4.8	$\mathcal{O}(3^n)$ Theorem 2.7	$2^{\mathcal{O}(nk \log n)}$ Theorem 5.15
eventually and locally 1-way	$\mathcal{O}(\lvert\Sigma\rvert 2^n n)$ Theorem 4.7	$\mathcal{O}(\lvert\Sigma\rvert 3^n)$ Theorem 4.3	$2^{\mathcal{O}(nk \log n)}$ Theorem 5.15
eventually 1-way	$\mathcal{O}(2^{2n} n)$ Theorem 4.6	$\mathcal{O}(2^n 3^n)$ Theorem 4.2	$2^{\mathcal{O}(nk \log n)}$ Theorem 5.15
2-way	$\mathcal{O}(2^{3n} n)$ Theorem 4.6	$2^{\mathcal{O}(n^2)}$ Theorem 5.16	$2^{\mathcal{O}((nk)^2)}$ Theorem 5.16

Table 4.1: Sizes of 1NBAs obtained by the complementation constructions.

$\mathcal{O}(2^{2n} n)$. In the following sections, we write $\mathbb{D} := \{-1, 0, 1\}$ for the set of directions in an infinite word.

4.1.1 Complementing co-Büchi Automata

We start with a construction for complementing the word language of an eventually 1-way nondeterministic co-Büchi automaton. The construction can be seen as a combination of two known constructions. The first one is Vardi's 2-way subset-construction [Var89] for complementing the language of a 2-way nondeterministic automaton over *finite words*. The construction is given in Theorem 2.6. The second one is Boigelot, Jodogne and Wolpers' breakpoint construction [BJW05] for complementing the language of a 1-way nondeterministic co-Büchi automata over *infinite words*. This construction is given in Theorem 2.7. As remarked in [BJW05], this construction is one essential part in the Miyano-Hayashi alternation-elimination construction [MH84, KV01] for translating 1ABAs into language-equivalent 1NBAs.

Before presenting the construction, we give a characterization of the words that are rejected by an eventually 1-way 2NCA.

Lemma 4.1 *Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be an eventually 1-way 2NCA and $w \in \Sigma^\omega$. We have $w \notin L^\omega(\mathcal{A})$ if and only if there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ such that the following conditions hold.*

(1) $q_I \in R_0$.

(2) *For all* $d \in \mathbb{D}$ *and* $i \in \mathbb{N}$ *with* $i + d \geq 0$, *we have* $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_{i+d}$.

(3) *For no* $i \in \mathbb{N}$ *and* $q \in R_i$, *we have* $\emptyset \models \delta_{\mathbb{D}_i}(q, w_i)$.

(4) *For all* $i \in \mathbb{N}$, *we have* $\delta^d_{\mathbb{D}_i}(S_i, w_i) \setminus F \subseteq S_{i+1}$.

(5) *For infinitely many* $i \in \mathbb{N}$, *we have* $S_i = \emptyset$ *and* $S_{i+1} = R_{i+1} \setminus F$ □

Proof First, we prove the *only if* direction. Assume $w \notin L^\omega(\mathcal{A})$, i.e., every run of $\mathcal{A}$ on w visits a state in F infinitely often. We need the following definitions. We call a sequence of configurations $(q_0, h_0) \dots (q_n, h_n) \in (Q \times \mathbb{N})^*$ a *run segment* if $(q_{i+1}, h_{i+1} - h_i) \in \delta_{\mathbb{D}_i}(q_i, w_i)$, for all $i < n$. The run segment is *initial* if $(q_0, j_0) = (q_I, 0)$. For $i \in \mathbb{N}$, we define R_i as the set of states that can be reached by $\mathcal{A}$ when reading w and ending with head position i. Formally, for $i \in \mathbb{N}$, we define

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

We show that R fulfills the first two conditions of the lemma. R satisfies (1) since $(q_I, 0)$ is an initial run segment. To show that (2) holds, assume $i \in \mathbb{N}$, $p, q \in Q$, and $d \in \mathbb{D}$. If $p \in R_i$, $(q, d) \in \delta_{\mathbb{D}_i}(p, w_i)$, and $i + d \geq 0$ then there is an initial run segment $r_0 \dots r_n \in (Q \times \mathbb{N})^*$ such that $r_n = (p, i)$. Hence, $r_0 \dots r_n(q, i + d) \in (Q \times \mathbb{N})^*$ is also an initial run segment. Thus, $q \in R_{i+d}$. Condition (3) is fulfilled since otherwise, there is an $i \in \mathbb{N}$ and a state $q \in R_i$ such that there exists an accepting run $(q_0, h_0) \dots (q_n, h_n)$ with $(q_n, h_n) = (q, i)$ of $\mathcal{A}$ on w.

Now, we define $S \in (2^{Q \setminus F})^\omega$. In the following, we call a run segment $(q_0, h_0) \dots (q_n, h_n) \in (Q \times \mathbb{N})^*$ *F-avoiding* if $q_i \notin F$, for all $i \leq n$. For defining S inductively, it is convenient to use the auxiliary set $S_{-1} := \emptyset$.

Let $m \in \mathbb{N} \cup \{-1\}$ such that $S_m = \emptyset$. For every m, we define the word $T^m \in (Q \times \mathbb{N})^\omega$ as the set of F-avoiding run segments that start in $R_{m+1} \setminus F$. For brevity, we just write T instead of T^m. Formally, for $i \leq m$, we define $T_i = \emptyset$ and for $i > m$, we define

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

We show that there is a position $n > m$ such that $T_n = \emptyset$. Assume that such a position n does not exist. By König's Lemma, it is easy to see that T contains

an infinite F-avoiding run segment. Thus, there is an accepting infinite run of $\mathcal{A}$ on w. This contradicts the assumption $w \notin L^\omega(\mathcal{A})$. For the positions $i \in \mathbb{N}$ with $m < i \leq n$, we define $S_i := T_i$.

By construction of S, conditions (4) and (5) are fulfilled. This is shown by a similar argumentation as above, where we show that R fulfills condition (2).

Now, we prove the *if* direction. Assume there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ fulfilling the conditions (1)–(5). Consider a run r of $\mathcal{A}$ on w that has the form $(q_0, h_0)(q_1, h_1) \cdots \in (Q \times \mathbb{N})^\omega$. Due to conditions (1) and (2), we have $q_i \in R_{h_i}$, for all $i \in \mathbb{N}$.

We show that r is rejecting. Suppose that r is accepting. Then, there is a $k \in \mathbb{N}$ such that $q_i \notin F$, for all $i > k$. Due to condition (5), there is a breakpoint $S_m = \emptyset$ with $m > h_k$ and $S_{m+1} = R_{m+1} \setminus F$. Since r is eventually 1-way, there is a position $n > k$ such that $h_n = m + 1$. Without loss of generality, we assume that n is maximal. Since r is eventually 1-way and the set Q is finite, such an n exists. We have $h_i > m + 1$, for all $i > n$.

Since $q_n \in R_{h_n}$ and $q_n \notin F$, we have $q_n \in S_{h_n}$. According to the condition (4), we infer that $q_i \in S_{h_i}$, for all $i > n$. Since r is eventually 1-way, there is no $m' > m$ such that $S_{m'} = \emptyset$. We obtain a contradiction to condition (5). ∎

The following theorem extends the breakpoint construction to 2-way automata that are eventually 1-way. We call it the *2-way breakpoint construction*. Roughly speaking, the constructed NBA $\mathcal{C}$ guesses a run that satisfies the conditions of Lemma 4.1 with respect to a given 2NCA.

Theorem 4.2 *For every eventually 1-way 2NCA $\mathcal{A}$ of size n, there is an 1NBA $\mathcal{B}$ that accepts the complement of $L^\omega(\mathcal{A})$ and has $\mathcal{O}(2^n 3^n)$ states.* □

Proof We construct a 1NBA that for an input word $w \in \Sigma^\omega$, guesses the words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ from Lemma 4.1. It locally checks the conditions (1)–(4). Using the breakpoint construction and its Büchi acceptance condition, it ensures that condition (5) is fulfilled. We first construct the 1NBA and then prove the correctness of the construction.

Construction Consider an eventually 1-way 2NCA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ with n states. We define the 1NBA $\mathcal{B} := (P, \Sigma, \eta, p_I, G)$ as follows.

- $P := (2^Q \times 2^{Q \setminus F} \times 2^Q) \cup \{p_I\}$.

- $G := 2^Q \times \{\emptyset\} \times 2^Q$.

The transition function η is defined as follows. For the initial state p_I, $D = \{0, 1\}$, and $a \in \Sigma$, we have $\eta_D(p_I, a) \ni (R_0, S_0, R_1)$ if and only if the following conditions hold.

1. $q_I \in R_0$.
2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.

For the states in $P \setminus \{p_I\}$, $D = \{-1, 0, 1\}$, and $a \in \Sigma$, the transition function $\eta_D\big((R_{-1}, S_{-1}, R_0), a\big)$ contains (R_0, S_0, R_1) if and only if the following conditions hold.

1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$,
3. For all $d \in \{0, 1\}$, we have $\delta_D^d(S_{-1}, a) \setminus F \subseteq S_{-1+d}$,
4. If $S_{-1} = \emptyset$ then $S_0 = R_0 \setminus F$.

Obviously, the size of $\mathcal{B}$ is in $\mathcal{O}(2^{3n})$. Observe that we can restrict P to the set of states $\{(R, S, R') \mid (R, S, R') \in P, S \subseteq R\} \cup \{p_I\}$. With this optimization and the argumentation as in Theorem 2.7, we obtain an automaton with $\mathcal{O}(2^n 3^n)$ states.

Correctness It remains to show that $L^\omega(\mathcal{B}) = \Sigma^\omega \setminus L^\omega(A)$. Consider a word $w \in \Sigma^\omega$.

Assume that $w \in L^\omega(B)$. Let $r := p_I(R_0, S_0, R_0')(R_1, S_1, R_1')\ldots$ be an accepting run of $\mathcal{C}$ on w. It suffices to show that the words $R := R_0 R_1 \ldots \in (2^Q)^\omega$ and $S := S_0 S_1 \ldots \in (2^{Q \setminus F})^\omega$ satisfy the conditions (1)–(5) of Lemma 4.1.

Since $\eta(p_I, w_0) \ni (R_0, S_0, R_0')$, we have $q_I \in R_0$. Thus, (1) holds.

By the definition of the transition function, for all $d \in \mathbb{D}$ and $i \in \mathbb{N}$ with $i + d \geq 0$, we have $\delta_{\mathbb{D}_i}^d(R_i, w_i) \subseteq R_{i+d}$. Thus, (2) holds. Similarly, we can show that (4) holds.

Condition (3) holds since by the definition of the transition function, we have $\mathtt{tt} \notin \delta_{\mathbb{D}_i}(p, w_i)$, for all $i \in \mathbb{N}$ and $p \in R_i$.

Since r is accepting, $S_m = \emptyset$ for infinitely many $m \in \mathbb{N}$. By definition of η, we also have that $S_{m+1} = R_{m+1} \setminus F$ whenever $S_m = \emptyset$, for all $m \in \mathbb{N}$. Thus, condition (5) holds.

For the other direction, assume that $w \notin L(\mathcal{A})$. By Lemma 4.1, there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that satisfy the conditions (1)–(5). We

define $r := p_I(R_0, S_0, R_1)(R_1, S_1, R_2)\ldots$ and show that r is an accepting run of $\mathcal{B}$ on w.

From the conditions (1)–(5), it follows that $\eta(p_I, w_0) \ni (R_0, S_0, R_1)$ and $\eta\big((R_{i-1}, S_{i-1}, R_i), w_i\big) \ni (R_i, S_i, R_{i+1})$, for all $i > 0$.

Since S_m is empty for infinitely many $m \in \mathbb{N}$, the run r is accepting. ∎

Now, consider the case where the eventually 1-way 2NCA whose language has to be complemented is locally 1-way. In this case, we can modify condition (2) of Lemma 4.1 since the automaton does not move its read-only head backwards and forwards at the same time. The following requirement must hold.

(2′) For all $d \in \{0, 1\}$ and $i \in \mathbb{N}$, we have $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_{i+d}$ and for all $d \in \{-1, 0\}$ and $i \in \mathbb{N}$ with $i + d \geq 0$, we have $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_{i+d}$.

From this observation, we directly obtain the following theorem as a special case of Theorem 4.2.

Theorem 4.3 *For every locally and eventually* 1*-way 2NCA* $\mathcal{A}$ *of size* n*, there is an NBA* $\mathcal{B}$ *that accepts the complement of* $L^\omega(\mathcal{A})$ *and has* $\mathcal{O}(|\Sigma| \cdot 3^n)$ *states.*□

Proof The construction is a special case of the construction given in Theorem 4.2. Consider a locally and eventually 1-way 2NCA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$. We define the 1NBA $\mathcal{B} := (P, \Sigma, \eta, p_I, G)$ as follows.

- We define $P := (\Sigma \times 2^Q \times 2^{Q \setminus F}) \cup \{p_I\}$. The automaton preserves the following invariant. Whenever $\mathcal{A}$ goes to the state (a, R, S) then a must be equal to the letter that will be read next. This invariant can be checked at every step when the next letter is actually read. Hence, if $\mathcal{A}$ moves from state (a, R, S) to a state (b, R', S') then it can check the *forward constraints* of condition (2′) for R and S and the *backward constraints* of condition (2′) for R' and S', locally. Then, one step later, after reading the next letter, the *forward constraints* of R' and S' will be checked.
- We define $G := \Sigma \times 2^Q \times \{\emptyset\}$.

The transition function η is defined as follows. For the initial state p_I, $D = \{0, 1\}$, and $a \in \Sigma$, we have $\eta_D(p_I, a) \ni (b, R_1, S_1)$ if and only if the following conditions hold. There is some set $R_0 \subseteq Q$ and some set $S_0 \subseteq Q \setminus F$ such that

1. $q_I \in R_0$.
2. For all $d \in \{0,1\}$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
3. For all $d \in \{-1,0\}$, we have $\delta_{\{-1,0,1\}}^d(R_1, b) \subseteq R_{1+d}$.
4. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
5. For all $d \in \{0,1\}$, we have $\delta_D^d(S_0, a) \setminus F \subseteq S_d$.

For states in $P \setminus \{p_I\}$, $D = \{-1,0,1\}$, and $a \in \Sigma$, we have $\eta_D\big((a', R_0, S_0), a\big) \ni (b, R_1, S_1)$ if and only if the following conditions hold.

1. $a' = a$.
2. For all $d \in \{0,1\}$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
3. For all $d \in \{-1,0\}$, we have $\delta_D^d(R_1, b) \subseteq R_{1+d}$.
4. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
5. For all $d \in \{0,1\}$, we have $\delta_D^d(S_0, a) \setminus F \subseteq S_d$.
6. If $S_0 = \emptyset$ then $S_1 = R_1 \setminus F$.

Intuitively, for an input word w, the automaton guesses the words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ from Lemma 4.1. In the first component, it guesses the next letter of the input word. With the second component of P, it checks the conditions (1)–(3). With the third component, it checks that (4) holds. Finally, the breakpoint construction and the Büchi acceptance condition ensure that the condition (5) is fulfilled. It easy to check that $\mathcal{B}$ accepts the complement of $L^\omega(\mathcal{A})$.

The size of $\mathcal{B}$ is in $\mathcal{O}(|\Sigma| \cdot 2^{2n})$. Observe that we can restrict P to the set of states $\{(a, R, S) \mid (a, R, S) \in P \text{ and } S \subseteq R\} \cup \{p_I\}$. Hence, the cardinality of the state space is $\mathcal{O}(|\Sigma| \cdot 3^n)$. ∎

Remark 4.4 Consider the automata $\mathcal{A}$ and $\mathcal{B}$ from Theorem 4.2 and assume that the alphabet Σ has the form $\Delta \times \Gamma$. Clearly, the worst-case size of $\mathcal{B}$ is in $\mathcal{O}(|\Delta \times \Sigma| \cdot 3^n)$. Consider now the automaton $\mathcal{C}$ that represents the projection of $\mathcal{B}$ on Δ. It is straightforward to see that we can restrict the state space of $\mathcal{C}$ on the set $\Delta \times 2^Q \times 2^{Q \setminus F} \cup \{p_I\}$. This restricted state space suffices since the first component of a state of $\mathcal{C}$ just guesses the letter that will be read next and checks in the successive step that this guess has been right. So, the overall worst-case size of the automaton $\mathcal{C}$ is in $\mathcal{O}(|\Delta| \cdot 3^n)$. □

For 1-way automata, we further optimize condition (2′) of Lemma 4.1 since the automaton does not move its read-only head backwards. The following requirement must hold.

($2''$) For all $d \in \{0,1\}$ and $i \in \mathbb{N}$, we have $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_{i+d}$.

From this observation, we directly obtain Boigelot, Jodogne, and Wolper's breakpoint construction given in Theorem 2.7.

4.1.2 Complementing Very-Weak Automata

In this section, we develop constructions for translating very weak 2NCAs into language-equivalent 1NBAs. The constructions can bee seen as special cases of the 2-way breakpoint construction presented in Theorem 4.2. In particular, we exploit the very-weakness property of an eventually 1-way 2NCA to optimize the construction in Theorem 4.2. The optimization is based on the following observation. Each infinite run of a very-weak automaton will eventually get trapped in a state with a self-loop. Thus, the conditions (4) and (5) from Lemma 4.1 can be simplified accordingly. The simpler conditions allow us to reduce the state space of the resulting 1NBA. Roughly speaking, instead of guessing the word $S \in (2^{Q \setminus F})^\omega$ from Lemma 4.1 and checking that S fulfills the conditions (4) and (5), the constructed 1NBA only checks that no run of the V2NCA gets trapped in a non-accepting state.

Additionally, for very-weak automata, we can easily extend the above construction such that it also translates V2NCAs that are not necessarily eventually 1-way. This extension is based on the observation that there are only two types of loops. A very-weak automaton loops if (a) it gets trapped in a state without moving the read-only head any more, or (b) it gets trapped in a state by alternately moving its the read-only head to the right and then to the left. Such kind of loops can be detected locally.

Based on these two observations, we simplify Lemma 4.1. The lemma characterizes words that are rejected by a given 2VNCA.

Lemma 4.5 *Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be a V2NBA and $w \in \Sigma^\omega$. We have $w \notin L^\omega(\mathcal{A})$ if and only if there is a word $R \in (2^Q)^\omega$ such that the conditions (1)–(3) of Lemma 4.1 and the following conditions hold.*

($5'$) *There is no $n \in \mathbb{N}$ and $q \in R_n \setminus F$ such that for all $i \geq n$, we have $(q, 1) \in \delta_{\mathbb{D}_n}(q, w_i)$.*

($5''$) *There is no $i \in \mathbb{N}$ and $q \in R_i \setminus F$ such that $(q, 0) \in \delta_{\mathbb{D}_n}(q, w_n)$ or such that $(q, 1) \in \delta_{\mathbb{D}_n}(q, w_i)$ and $(q, -1) \in \delta_{\mathbb{D}_n}(q, w_{i+1})$.*

Furthermore, if $\mathcal{A}$ is eventually 1-way, condition (5″) is not required. □

The optimized automaton construction to complement a very-weak 2NCA is given in the following theorem.

Theorem 4.6 *For every V2NCA $\mathcal{A}$ of size n, there is a 1NBA $\mathcal{B}$ that accepts the complement of $L^\omega(\mathcal{A})$ and has $\mathcal{O}(2^{3n}n)$ states. Moreover, if $\mathcal{A}$ is eventually 1-way then $\mathcal{B}$ has $\mathcal{O}(2^{2n}n)$ states.* □

Proof The proof is similar to the proof for Theorem 4.2. We construct a 1NBA that guesses the word $R \in (2^Q)^\omega$ from Lemma 4.5, locally checks that the conditions (1)–(3) are fulfilled, and uses a *focus* and its Büchi acceptance condition to check that condition (5″) holds. We present the construction of the 1NBA and then its correctness proof.

Construction Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be a V2NCA. Let $E := (Q \setminus F) \cup \{*\}$. Furthermore, let $<$ be a total ordering on the set E, where $*$ is the greatest element. The function $\mathsf{next} : E \to E$ maps the greatest element $*$ to the smallest one and each of the other elements to the next greater one. We define the NBA $\mathcal{B} := (P, \Sigma, \eta, p_I, G)$ as follows.

- $P := (2^Q \times 2^Q \times 2^{Q \setminus F} \times E) \cup \{p_I\}$. The component in E is called *focus*. It is used to find a state in $Q \setminus F$ that can get trapped in a self-loop.
- $G := 2^Q \times 2^Q \times 2^{Q \setminus F} \times \{*\}$.

The transition function η is defined as follows. For the initial state p_I, $D = \{0, 1\}$, and $a \in \Sigma$, we have $\eta_D(p_I, a) \ni (R_0, R_1, R'_0, *)$ if and only if the following conditions hold.

1. $q_I \in R_0$.
2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
4. For all $q \in R_0 \setminus F$, we have $(q, 0) \notin \delta_D(q, a)$.
5. $R'_0 = \{q \in R_0 \setminus F \mid (q, 1) \in \delta_D(q, a)\}$.

For the other states in $P \setminus \{p_I\}$, $D = \{-1, 0, 1\}$, and $a \in \Sigma$, the transition $\eta_D\big((R_{-1}, R_0, R'_{-1}, s), a\big)$ contains (R_0, R_1, R'_0, s') if and only if the following conditions hold.

1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.

2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
3. $s' = \begin{cases} s & \text{if } s \in R_0 \cap \delta_D^1(s, a), \\ \mathsf{next}(s) & \text{otherwise.} \end{cases}$
4. For all $q \in R_0 \setminus F$, we have $(q, 0) \notin \delta_D(q, a)$.
5. $R_0' = \{q \in R_0 \setminus F \mid (q, 1) \in \delta_D(q, a)\}$.
6. For all $q \in R_{-1}'$, we have $(q, -1) \notin \delta_D(q, a)$.

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

Assume that $w \in L^\omega(\mathcal{B})$. Let $r = p_0 p_1 \ldots \in P^\omega$ be an accepting run of the 1-way automaton $\mathcal{B}$ on w.

It suffices to construct a word $R \in (2^Q)^\omega$ that fulfills the conditions (1)–(3), (5′), and (5″) of Lemma 4.5. By definition of the transition function, $p_0 = p_I$ and for all $i > 0$, each p_i is a tuple of the form $(A_i, B_i, s_i) \in P$. Define $R_i := A_{i+1}$, for all $i \in \mathbb{N}$. By the same the arguments as in the proof of Theorem 4.2, it follows that R fulfills the conditions (1)–(3). Moreover, R fulfills condition (5″) since for every $i \in \mathbb{N}$ and $D = \{-1, 0, 1\}$, there is no state q in the set $\{q \in R_i \setminus F \mid (q, 1) \in \delta_D(q, w_i)\}$ such that $(q, -1) \in \delta_D(q, w_{i+1})$.

It remains to show that R fulfills condition (5′). For the sake of contradiction, assume (5′) does not hold. Then, there is an $n \in \mathbb{N}$ and a state $q \in R_n \setminus F$ such that $(q, 1) \in \delta_{\mathbb{D}_i}(q, w_i)$, for all $i \geq n$. Therefore, there is a position $k > n$ such that $s_k = q$. By the definition of the transition function, there is no $l > k$ such that $s_l \neq q$. Hence, the run r is not accepting since G is not visited infinitely often.

For the other direction, assume $w \notin L^\omega(\mathcal{A})$. Let $R \in (2^Q)^\omega$ be a word that fulfills the conditions (1)–(3), (5′), and (5″). We construct an accepting run of $\mathcal{B}$ on w. We need the following definitions of the sequences $R' \in (2^{Q \setminus F})^\omega$ and $s \in E^\omega$. For $i \in \mathbb{N}$, let $R_i' := \{q \in R_i \setminus F \mid (q, 1) \in \delta_{\mathbb{D}_i}(q, w_i)\}$. Furthermore, let $s_0 := *$ and for $i \in \mathbb{N}$, we define

$$s_i := \begin{cases} s_i & \text{if } s_i = R_i \cap \delta_{\mathbb{D}_i}^1(s_i, w_i), \\ \mathsf{next}(s_i) & \text{otherwise.} \end{cases}$$

We define the sequence $r := p_0 p_1 \in P^\omega$, where $p_0 := p_I$ and for $i > 0$, p_i is the tuple $(R_{i-1}, R_i, R_{i-1}', s_{i-1})$.

By construction, r is a run of $\mathcal{B}$ on w. We show that r is accepting. Assume the opposite, i.e., G is not visited infinitely often. Then, by definition of the

run, $\mathcal{B}$ gets trapped in a state $q \in Q \setminus F$ with the E component of its states. Therefore, there is a position $n \in \mathbb{N}$ such that $(q, 1) \in \delta_{\mathbb{D}_i}(q, w_i)$, for all $i \geq n$. Thus, condition (5′) holds.

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

If $\mathcal{A}$ is eventually 1-way, the automaton $\mathcal{B}$ does not have to check (5″). Hence, we can remove the third component from $\mathcal{B}$'s state space. ■

If the eventually 1-way V2NCA whose language has to be complemented is also locally 1-way, then we can further improve the construction from Theorem 4.6. In particular, we replace condition (2) of Lemma 4.5 by condition (2′) and then use the same construction technique as presented in Theorem 4.3. We directly obtain the following theorem.

Theorem 4.7 *For every locally and eventually* 1*-way V2NCA* $\mathcal{A}$ *with* n *states, there is an 1NBA* $\mathcal{B}$ *that accepts the complement of* $L^\omega(\mathcal{A})$ *and has* $\mathcal{O}(|\Sigma| \cdot 2^n n)$ *states.* □

Finally, consider the case, where the very-weak nondeterministic co-Büchi automaton whose language has to be complemented is 1-way. We further simplify the construction from Theorem 4.7. In particular, we replace condition (2′) of Lemma 4.5 by condition (2″) and then use the same construction technique as presented in Theorem 2.7.

We point out that the idea of this construction is implicitly used in the Gastin and Oddoux's alternation-elimination construction [GO01, BCPR07] that translates V1ABAs into NBAs, and in the Lange and Stirling's *focus approach* of the satisfiability checking algorithm for LTL formulas presented in [LS01, DL05].

Theorem 4.8 *For every V1NCA* $\mathcal{A}$ *with* n *states, there is a 1DBA* $\mathcal{B}$ *that accepts the complement of* $L^\omega(\mathcal{A})$ *and has* $\mathcal{O}(2^n n)$ *states.* □

Proof Consider a V1NCA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$. Let $E := (Q \setminus F) \cup \{*\}$ and let $<$ be a total ordering on the set E, where $*$ is the greatest element. The function $\mathsf{next} : E \to E$ maps the greatest element $*$ to the smallest one and all other elements to the next greater one.

We define the 1DBA $\mathcal{B} := (2^Q \times E, \Sigma, \eta, (\{q_I\}, *), 2^Q \times \{*\})$, where for $R \subseteq Q$, $s \in E$, $D \subseteq \mathbb{D}$, and $a \in \Sigma$, we have

$$\eta_D\big((R,s),a\big) := \begin{cases} \big(\delta(R,a), s\big) & \text{if } s \in R \cap \delta_D^1(s,a) \\ \big(\delta(R,a), \mathsf{next}(s)\big) & \text{otherwise.} \end{cases}$$

The V1NCA $\mathcal{A}$ accepts a word w if and only if there is a run that gets trapped in a state $q \notin F$. This is equivalent to the fact that the 1DBA $\mathcal{B}$ detects the existence of such a run with its E component and rejects. ∎

4.1.3 Transition Systems Instead of Automata

In this section, we investigate an alternative model for representing word languages, namely, transition systems, which are used in state-of-the-art model checkers that are based on the symbolic model checker SMV, see [McM92]. We show how to optimize the complementation constructions for locally and eventually 1-way nondeterministic co-Büchi automata when the resulting representation is a transition system rather than an automaton.

We start with the definition of a transition system. A *transition system* is a quintuple $\mathcal{T} = (Q, \Sigma, \Delta, I, F)$, where Q is the set of *states*, Σ is a finite, nonempty alphabet, $\Delta \subseteq (Q \times \Sigma) \times (Q \times \Sigma)$ is the *transition relation*, $I \subseteq (Q \times \Sigma)$ are the *initial locations*, and $F \subseteq Q$ is a Büchi acceptance condition. A *location* is a tuple in $Q \times \Sigma$. A *run* of a transition system on a word $w \in \Sigma^\omega$ is a sequence of locations $(q_0, w_0)(q_1, w_1) \ldots \in (Q \times \Sigma)^\omega$ such that for all $i \in \mathbb{N}$, we have $((q_i, w_i), (q_{i+1}, w_{i+1})) \in \Delta$. The run is *accepting* if the word $q_0 q_1 \ldots \in Q^\omega$ contains infinitely many states from F. The *word language* of a transition system $\mathcal{T}$ is $L^\omega(\mathcal{T}) := \{w \in \Sigma^\omega \mid \text{there is an accepting run of } \mathcal{T} \text{ on } w\}$. The *size* of $\mathcal{T}$ is the number of its states $|Q|$.

Remark 4.9 The definition of the size of a transition system is motivated from model checking with symbolic model checkers like SMV. Since the transition system of a (negated) specification S shares the alphabet of the transition system (without fairness constraints) of its system model M, the overall search space is just $(Q_M \times \Sigma_M) \times Q_S$, where Q_M is the state set of the system, Σ_M is the system alphabet, and Q_S is the state set of the specification S. □

Example 4.10 Consider the set of propositions $\mathcal{P} := \{a, b\}$ and the alphabet $\Sigma = 2^{\mathcal{P}}$. Figure 4.1 depicts a Büchi automaton and a transition system that

Figure 4.1: Eventually always proposition b.

Figure 4.2: Infinitely often proposition b.

accepts all words over Σ, where eventually always b occurs. We draw states that belong to the Büchi acceptance set by double lines.

Another example is presented in Figure 4.2. It depicts a Büchi automaton and a transition system that accepts all words over Σ that contain the proposition b at infinitely many positions.

Lemma 4.11 *We can translate every 1NBA into a language-equivalent transition system of the same size.* □

Proof Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be a 1NBA. We define the transition system $\mathcal{T} = (Q, \Sigma, \Delta, Q_I, F)$ with

- $Q_I := \{(q, a) \mid q \in \delta_{\{0,1\}}(q_I, a), \text{ for some } q \in Q, a \in \Sigma\}$ and
- $\Delta := \{((q, a), (q', a')) \mid q' \in \delta_{\{-1,0,1\}}(q, a'), \text{ for some } q, q' \in Q, a, a' \in \Sigma\}$.

We show that $L^\omega(\mathcal{T}) = L^\omega(\mathcal{A})$. Let $w \in \Sigma^\omega$ be a word. Consider an accepting run $(q_0, w_0)(q_1, w_1)\ldots$ of $\mathcal{T}$ on w. We show that $q_I q_0 q_1 \ldots$ is a run of $\mathcal{A}$ on w. Since $(q_0, w_0) \in Q_I$, we have $q_0 \in \delta_{\{0,1\}}(q_I, a)$. Now, consider an arbitrary position $i \in \mathbb{N}$. Since $((q_i, w_i), (q_{i+1}, w_{i+1})) \in \Delta$, we have $q_{i+1} \in \delta_{\{-1,0,1\}}(q_i, w_{i+1})$. Obviously $q_I q_0 q_1 \ldots$ is accepting. Hence, $w \in L^\omega(\mathcal{A})$.

We show the other direction. Let $q_0 q_1 \ldots$ be an accepting run of $\mathcal{A}$ on w. We show that $(q_1, w_0)(q_2, w_1)\ldots$ is an accepting run of $\mathcal{T}$ on w. Since $q_1 \in \delta_{\{0,1\}}(q_I, w_0)$, we have $(q_1, w_0) \in Q_I$. Next, consider an arbitrary position $i > 0$. Since $q_{i+1} \in \delta_{\{-1,0,1\}}(q_i, w_i)$, we have $((q_i, w_{i-1}), (q_{i+1}, w_i)) \in \Delta$. Obviously, $q_I q_0 q_1 \ldots$ is accepting. Hence, $w \in L^\omega(\mathcal{A})$. ■

In the remainder of this section, we show that the transition systems obtained from the constructions for complementing the languages of locally and eventually 1-way 2NCAs have the same worst-case sizes as the transition systems obtained from the constructions for complementing the languages of 1NCAs. Recall the complementation constructions from Theorem 4.7 and Theorem 4.3 that translate locally and eventually 1-way 2NCAs into 1NBAs. One component in the states of these 1NBAs is used to guess the letter that will occur after the letter that is currently read. When translating those 1NBAs into transition systems, this component is not needed any more since the transition systems always see the letter that will occur next in the transition relation. So, while a 1NBAs has to guess the correct letter in each step and additionally, has to store its guess in its state for checking afterwards that its guess has been correct, a transition system is able to directly move to the correctly labeled successor state. Hence, when translating the 1NBAs obtained from Theorem 4.7 and Theorem 4.3 into transition systems, we may dispose of the component in the state space that is used for storing the guessed letter. We state this observation in the next theorem.

Theorem 4.12 *For every locally and eventually* 1*-way 2NCA* $\mathcal{A}$ *of size* n*, there is a transition system* $\mathcal{T}$ *that accepts the complement of* $L^\omega(\mathcal{A})$ *and has* $\mathcal{O}(3^n)$ *states. If* $\mathcal{A}$ *is very weak then* $\mathcal{T}$ *has just* $\mathcal{O}(2^n n)$ *states.* □

State-of-the-art symbolic model checkers like NuSMV use transition systems for representing the specification. Theorem 4.12 states that the worst-case blow-ups for complementing 1NCAs and eventually 1-way 2NCAs are both in $\mathcal{O}(3^n)$, where n is the size of the input automata. It follows that alternation-elimination translations from 1ABAs and eventually 1-way ABAs to transition systems share the same worst-case bound. In the following section, we exploit this fact when translating logics with past operators to transition systems.

4.2 The Linear-Time Temporal Logic PPSL

In this section, we introduce the linear-time temporal logic PPSL. We first define the logic and illustrate its use by formalizing some properties in PPSL. Then, we present a translation from PPSL to nondeterministic Büchi automata. Finally, we discuss succinctness properties of PPSL with respect to other logics used in practice.

4.2.1 The Logic PPSL

PPSL extends the linear-time core of the IEEE standard *Property Specification Language* (PSL) [Psl05] by past operators. PSL is an descendant of the popular linear-time logic LTL [Pnu77] with operators that handle *semi-extended regular expressions* (SEREs), which are essentially regular expressions with an operator to represent the intersection of regular languages. As PSL, the logic PPSL consists of two layers: a (semi-extended) regular-expression layer and a logic layer that combines semi-extended regular expressions with temporal operators. In the following, we define each layer separately.

Semi-Extended Regular Expression Layer Let $\mathcal{P}$ be a set of propositions and $\Sigma := 2^{\mathcal{P}}$ a finite alphabet. A *semi-extended regular expression* (*SERE*) over $\mathcal{P}$ is given by the following grammar.

$$r ::= \varepsilon \mid b \mid r \cup r \mid r \cap r \mid r \,;\, r \mid r : r \mid r^{+},$$

where $b \in \mathcal{B}(\mathcal{P})$. We call the $\cap$ the *intersection* operator, $\cup$ the *union* operator, ; the *concatenation* operator, : the *fusion* operator, and $+$ the *plus* operator. Furthermore, for a SERE r, we define the *Kleene star* operator $r^* := \varepsilon \cup r^+$. A *regular expression* (*RE*) is a SERE that does not contain the intersection operator $\cap$.

The language of a SERE is inductively defined. Let $b \in \mathcal{B}(\mathcal{P})$ and r, s are SEREs.

$$\begin{aligned}
L^*(\varepsilon) &:= \{\varepsilon\}.\\
L^*(b) &:= \{w \in \Sigma^* \mid |w| = 1 \text{ and } w_0 \text{ fulfills } b\}.\\
L^*(r \cup s) &:= \{w \in \Sigma^* \mid w \in L^*(r) \text{ or } w \in L^*(s)\}.\\
L^*(r \cap s) &:= \{w \in \Sigma^* \mid w \in L^*(r) \text{ and } w \in L^*(s)\}.\\
L^*(r \,;\, s) &:= \{vw \in \Sigma^* \mid v \in L^*(r) \text{ and } w \in L^*(s)\}.\\
L^*(r : s) &:= \{vaw \in \Sigma^* \mid a \in \Sigma, va \in L^*(r), \text{ and } aw \in L^*(s)\}.\\
L^*(r^+) &:= \{w \in \Sigma^* \mid \exists n \in \mathbb{N} : w = v_0 \dots v_{n-1} \text{ and } \forall j \in [n] : v_i \in L^*(r)\}.
\end{aligned}$$

The *size* $|r|$ of a SERE r is its syntactic length.

Logic Layer Now, we define PPSL. The syntax of a *PPSL* formula over $\mathcal{P}$ is given by the following grammar.

$$\varphi ::= b \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \varphi \,\mathsf{S}\, \varphi \mid r \diamond\!\!\rightarrow \varphi \mid r \blacklozenge\!\!\rightarrow \varphi,$$

where $b \in \mathcal{B}(\mathcal{P})$ and r is a SERE. We denote the set of PPSL formulas whose SEREs are all REs by $PPSL^{\mathsf{re}}$. For a SERE r and formulas φ and ψ, we define the standard syntactic abbreviations $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi \mathsf{R} \psi := \neg(\neg\varphi \mathsf{U} \neg\psi)$, $\varphi \mathsf{T} \psi := \neg(\neg\varphi \mathsf{S} \neg\psi)$, $r \Box\!\!\rightarrow \varphi := \neg(r \Diamond\!\!\rightarrow \neg\varphi)$, and $r \boxminus\!\!\rightarrow \varphi := \neg(r \diamondminus\!\!\rightarrow \neg\varphi)$. Using these abbreviations, we can obviously translate any PPSL formula into *positive normal form*, i.e., negations occur only in front of propositional-logic formulas. Note that such a translation might double the size of the formula. Additionally, we introduce the following operators as syntactic sugar: $\mathsf{O}\varphi := \mathsf{tt}; \mathsf{tt} \Diamond\!\!\rightarrow \varphi$, $\mathsf{Y}\varphi := \mathsf{tt}; \mathsf{tt} \diamondminus\!\!\rightarrow \varphi$, $\mathsf{Z}\varphi := \neg\mathsf{Y}\mathsf{tt} \vee \mathsf{Y}\varphi$, $\mathsf{F}\varphi := \mathsf{tt}\ \mathsf{U}\ \varphi$, and $\mathsf{G}\varphi := \mathsf{ff}\ \mathsf{R}\ \varphi$.

A *PSL* formula is a PPSL formula that does not use the past operators S and $\diamondminus\!\!\rightarrow$. We denote the set of PSL formulas whose SEREs are all REs by PSL^{re} . Next, we define the well known linear-time temporal logic [Pnu77]. A *PLTL* formula is a PPSL formula that contains the operators $\Diamond\!\!\rightarrow$, $\Box\!\!\rightarrow$, $\diamondminus\!\!\rightarrow$, and $\boxminus\!\!\rightarrow$ only in a restricted way. Namely, only the SERE "$\mathsf{tt}\,;\,\mathsf{tt}$" is allowed. An *LTL* formula is a PLTL formula that has no Y, S and T operator.

We interpret PPSL formulas over infinite word over Σ. For a word $w \in \Sigma^\omega$ and a position $i \in \mathbb{N}$, we define the semantics of PPSL as follows.

$$
\begin{array}{lll}
(w,i) \models b & \text{iff} & w_i \text{ satisfies } b \\
(w,i) \models \varphi \vee \psi & \text{iff} & (w,i) \models \varphi \text{ or } (w,i) \models \psi \\
(w,i) \models \neg\varphi & \text{iff} & (w,i) \not\models \varphi \\
(w,i) \models \varphi\ \mathsf{U}\ \psi & \text{iff} & \exists k \geq i : (w,k) \models \psi \text{ and } \forall i \leq j < k : (w,j) \models \varphi \\
(w,i) \models \varphi\ \mathsf{S}\ \psi & \text{iff} & \exists k \leq i : (w,k) \models \psi \text{ and } \forall k < j \leq i : (w,j) \models \varphi \\
(w,i) \models r \Diamond\!\!\rightarrow \varphi & \text{iff} & \exists k \geq i : w_{i..k} \in L^*(r) \text{ and } (w,k) \models \varphi \\
(w,i) \models r \diamondminus\!\!\rightarrow \varphi & \text{iff} & \exists k \leq i : w_{k..i} \in L^*(r) \text{ and } (w,k) \models \varphi
\end{array}
$$

The language of a PPSL formula φ is $L^\omega(\varphi) := \{w \in \Sigma^\omega \mid (w,0) \models \varphi\}$. Two formulas φ and ψ are *initially equivalent* if $L^\omega(\varphi) = L^\omega(\psi)$. As for SEREs, the *size* $|\varphi|$ of a PPSL formula φ is its syntactic length. We write $\mathsf{Sub}(\varphi)$ for the set of sub-formulas of a PPSL formula φ.

Remark 4.13 In the PSL standard [Psl05], we also have atomic formulas of the form $\mathsf{ended}(r)$ and $\mathsf{prev}(r)$, where r is a SERE. For instance, the word $w \in \Sigma^\omega$ satisfies $\mathsf{ended}(r)$ at position i if and only if there is a subword u of w that ends at i and $u \in L^*(r)$. The operators ended and prev can be seen as restricted variants of the past operator $\diamondminus\!\!\rightarrow$. For instance, in PPSL, if $\varepsilon \notin L^*(r)$, $\mathsf{ended}(r)$ is syntactic sugar for $r \diamondminus\!\!\rightarrow \mathsf{tt}$, and tt otherwise. Observe that ended and prev can only be applied to SEREs, and, in contrast to $\diamondminus\!\!\rightarrow$, it is not possible

to define the classical past operators Y, H, and O with them. We also remark that the literature, for example, [BDBF+05, CRST06, Lan07, PZ06] 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 $\mathsf{ended}(r)$ and $\mathsf{prev}(r)$. Finally, we remark that the automata constructions [BDBF+05] for PSL cannot cope with the operators ended and prev, which are handled by our construction in Corollary 4.18 for PPSL. □

Example 4.14 A standard example for showing that the past operators of PLTL can lead to more intuitive specifications is $\mathsf{G}(\mathit{grant} \rightarrow \mathsf{O}\mathit{request})$, this means, every grant is preceded by a request [LPZ85]. An initially equivalent LTL formula is $\mathit{request}\,\mathsf{R}\,(\neg \mathit{grant} \vee \mathit{request})$. Let us now illustrate the beneficial use of SEREs and past operators. Suppose that a request is not a single event but a sequence of events, for example, a request consists of a *start* event that is later followed by an *end* event such that no *cancel* event happens between the *start* and the *end* event. Such sequences are naturally described by the SERE $(\mathit{start}\,;\mathsf{tt}^*\,;\mathit{end}) \cap (\neg \mathit{cancel})^*$. Using this SERE and the new past operator $\Diamond\!\!\!\!-\!\!\!\rightarrow$, we can easily express the property in PPSL that every grant is preceded by a request:

$$\mathsf{G}\big(\mathit{grant} \rightarrow \big(((\mathit{start}\,;\mathsf{tt}^*\,;\mathit{end}) \cap (\neg \mathit{cancel})^*)\,;\mathsf{tt}^* \Diamond\!\!\!\!-\!\!\!\rightarrow \mathsf{tt}\big)\big). \tag{4.1}$$

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

$$\mathsf{G}\big(\mathit{grant} \rightarrow \mathsf{O}\big(\mathit{end} \wedge \neg \mathit{cancel} \wedge \mathsf{Y}(\neg \mathit{cancel}\;\mathsf{S}\;(\mathit{start} \wedge \neg \mathit{cancel}))\big)\big). \tag{4.2}$$

Although debatable, we consider that the PPSL formula (4.1) is easier to understand than the PLTL formula (4.2). In PSL, we can express the property as $\mathit{norequest} \;\square\!\!\!\rightarrow \neg \mathit{grant}$, where the SERE *norequest* describes the complement of the language $L\big(\mathsf{tt}^*\,;((\mathit{start}\,;\mathsf{tt}^*\,;\mathit{end}) \cap (\neg \mathit{cancel})^*)\,;\mathsf{tt}^*\big)$, that is, *norequest* is the SERE

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

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 4.15 Let us give another example to illustrate the usefulness of past operators, in particular, the operator $\Diamond\!\!\!\!-\!\!\!\!\rightarrow$. For $n \geq 1$ and $i \in [n]$, consider the PPSL formula

$$\varphi_{n,i} := \mathsf{G}\big(send_i \rightarrow \big(switch_i \cap (init\,;(\neg init)^*) \Diamond\!\!\!\!-\!\!\!\!\rightarrow \mathsf{tt}\big)\big),$$

where $switch_i$ counts the number of $switch$ events modulo n, this means,

$$\begin{aligned} switch_i \;:=\; \Big(& \underbrace{(\neg switch)^*\,;\, switch\,;\ldots;(\neg switch)^*\,;\, switch}_{n \text{ times}}\Big)^*; \\ & \underbrace{(\neg switch)^*\,;\, switch\,;\ldots;(\neg switch)^*\,;\, switch}_{i \text{ times}}\,;(\neg switch)^*. \end{aligned} \tag{4.3}$$

Intuitively, $\varphi_{n,i}$ 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 if and only if 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, [DG07]).

The negation of the PSL formula

$$\big((\neg init)^* \Diamond\!\!\rightarrow send_i\big) \vee \mathsf{F}\big(init \wedge \big((\mathsf{tt}\,;(\neg init)^*) \cap \big(\textstyle\bigcup_{j\neq i} switch_j\big) \Diamond\!\!\rightarrow send_i\big)\big) \tag{4.4}$$

is initially equivalent to $\varphi_{n,i}$. Note that the size of the formula (4.4) is quadratic in n, whereas the size of the formula (4.3) is only linear in n. In Section 4.2.3, we prove that PPSL is exponentially more succinct than PSL. □

4.2.2 From PPSL to Automata

In this section, we translate PPSL formulas into language-equivalent Büchi automata. We first recall constructions for the regular layer. The following lemma summarizes standard constructions for translating SEREs and REs into NFAs and for computing the mirror language of an NFA, see [BDBF+05,HU79]. The *mirror language* of a word language $L \subseteq \Sigma^*$ is defined as the set $\{w_n \ldots w_0 \mid w_0 \ldots w_n \in L\}$.

Lemma 4.16 *Let s be a SERE, r a RE, and $\mathcal{A}$ a 1NFA of size n each.*

1. *We can construct an NFA $\mathcal{A}'$ of size 2^n with $L^*(\mathcal{A}') = L^*(s)$.*

2. *We can construct an NFA $\mathcal{A}'$ of size n with $L^*(\mathcal{A}') = L^*(r)$.*
3. *We can construct an NFA $\mathcal{A}'$ of size n that accepts the mirror language of $L^*(\mathcal{A})$.* □

The following theorem shows how we can translate a PPSL$^{\mathsf{re}}$ formula into a language-equivalent locally and eventually 1-way 2ABA.

Theorem 4.17 *We can translate every PPSL$^{\mathsf{re}}$ formula that is in positive normal form, and is of size n into a language-equivalent locally and eventually 1-way 2ABA with at most n states. Furthermore, for PSL$^{\mathsf{re}}$ formulas the 2ABA is 1-way and for PLTL formulas the 2ABA is very weak.* □

Proof Let φ be a PPSL$^{\mathsf{re}}$ formula. We translate this formula into a language-equivalent locally and eventually 1-way 2ABA.

For every RE r in φ, let $\mathcal{A}_r$ and $\mathcal{A}'_r$ be the corresponding automata constructed according to Lemma 4.16 such that $L(r) = L(\mathcal{A}_r)$ and $\mathcal{A}'_r$ accepts the mirror language of $L(r)$. We assume that the state sets of these automata are pairwise disjoint. In the following, we split the proof into several parts: construction of the 2ABA, correctness of the construction, proof of being eventually 1-way, and finally, translation to an 1NBA.

Construction Next, we define the 2ABA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$. We define the set of states as $Q := \mathsf{Sub}(\varphi) \cup \hat{Q}$, where

$$\begin{aligned}\hat{Q} :=& \{s \star\!\!\rightarrow \psi \mid \star\!\!\rightarrow \in \{\Diamond\!\!\rightarrow, \Box\!\!\rightarrow\}, r \star\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi), \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\ & \{s \star\!\!\rightarrow \psi \mid \star\!\!\rightarrow \in \{\diamondsuit\!\!\rightarrow, \boxminus\!\!\rightarrow\}, r \star\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi), \text{ and } s \text{ is a state in } \mathcal{A}'_r\}.\end{aligned}$$

The initial state $q_I := \varphi$. We define the set of accepting states as

$$\begin{aligned}F :=& \{\gamma \,\mathsf{R}\, \psi \mid \gamma \,\mathsf{R}\, \psi \in \mathsf{Sub}(\varphi)\} \cup \\ & \{s \Box\!\!\rightarrow \psi \mid r \Box\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\}.\end{aligned}$$

It remains to define the transition function δ. The following definitions are similar to the standard construction for translating LTL into alternating automata. Let $a \in \Sigma$ and $D \subseteq \mathbb{D}$.

- For $b \in \mathcal{B}(\mathcal{P})$, we define

$$\delta_D(b, a) := \begin{cases} \mathsf{tt} & \text{if } a \text{ satisfies } b, \\ \mathsf{ff} & \text{otherwise.} \end{cases}$$

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

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

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

$$\begin{aligned}
\delta_D(\gamma \,\mathsf{U}\, \psi, a) &:= (\psi, 0) \vee \big((\gamma, 0) \wedge (\gamma \,\mathsf{U}\, \psi, 1)\big),\\
\delta_D(\gamma \,\mathsf{R}\, \psi, a) &:= (\psi, 0) \wedge \big((\gamma, 0) \vee (\gamma \,\mathsf{R}\, \psi, 1)\big),\\
\delta_D(\gamma \,\mathsf{S}\, \psi, a) &:= (\psi, 0) \vee \big((\gamma, 0) \wedge (\gamma \,\mathsf{S}\, \psi, -1)\big), \text{ and}\\
\delta_D(\gamma \,\mathsf{T}\, \psi, a) &:= \begin{cases} (\psi, 0) \wedge \big((\gamma, 0) \vee (\gamma \,\mathsf{T}\, \psi, -1)\big) & \text{if } -1 \in D,\\ (\psi, 0) & \text{otherwise.} \end{cases}
\end{aligned}$$

Now, we turn to the transitions for the subformulas with an RE. We follow the construction given in [BDBF+05] for PSL.

- The state $r \diamond\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi)$ is used to start a simulation of the NFA $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ 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 $\delta_D(r \diamond\!\!\rightarrow \psi, a) := (s_I \diamond\!\!\rightarrow \psi, 0)$ 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 $r \diamond\!\!\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi)$ are defined similarly. Instead of simulating the NFA $\mathcal{A}_r$, $\mathcal{A}_\varphi$ simulates the NFA $\mathcal{A}'_r$, where it moves the read-only head to the left instead of to the right.

- If the state is $\alpha \,\square\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi)$, the automaton $\mathcal{A}_\varphi$ simulates a run of the NFA $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ viewed as a universal automaton. Whenever the simulation reaches a final state, $\mathcal{A}_r$ has to proceed with the state ψ. Formally, we define $\delta_D(r \,\square\!\!\rightarrow \psi, a) := (s_I \,\square\!\!\rightarrow \psi, 0)$ and for $s \in S$,

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

  The transitions for a subformula $r \boxminus\!\!\rightarrow \psi \in \mathsf{Sub}(\varphi)$ are similarly defined. However, if the read-only head is at the beginning of the input word, $\mathcal{A}_r$

can stop the simulation. Formally, for the NFA $\mathcal{A}'_r = (S, \Sigma, \eta, s_I, E)$ and $s \in S$, we define $\delta_D(r \boxminus\!\!\rightarrow \psi, a) := (s_I \boxminus\!\!\rightarrow \psi, 0)$ and for $-1 \notin D$, we have

$$\delta_D(s \boxminus\!\!\rightarrow \psi, a) := \begin{cases} (\psi, 0) & \text{if } \eta(s,a) \cap E \neq \emptyset, \\ \mathsf{tt} & \text{otherwise.} \end{cases}$$

For $-1 \in D$, we have

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

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

Note that from the definition of the state set Q and Lemma 4.16, we directly obtain $|\mathcal{A}| \in \mathcal{O}(n)$. By inspecting $\mathcal{A}$'s transition function, we also see that $\mathcal{A}$ is locally 1-way.

Correctness In the remainder of the proof, we show the correctness of the given construction. In particular, we prove that for every word $w \in \Sigma^\omega$, subformula $\psi \in \mathsf{Sub}(\varphi)$, and position $i \in \mathbb{N}$, the following holds

$$(w, i) \models \psi \quad \text{if and only if} \quad \mathcal{A} \text{ accepts } w \text{ from configuration } (\psi, i).$$

This equivalence immediately implies $L^\omega(\mathcal{A}) = L^\omega(\varphi)$. We prove the equivalence by induction over the formula structure of ψ. Let $w \in \Sigma^\omega$.

Consider the base case $\psi = b$, for some $b \in \mathcal{B}(\mathcal{P})$. Let $i \in \mathbb{N}$. By definition, $(w, i) \models b$ is equivalent to "w_i satisfies b". By construction, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration (b, i).

Consider the case $\psi = \psi_1 \wedge \psi_2$. Let $i \in \mathbb{N}$. Assume $(w, i) \models \psi$, i.e, $(w, i) \models \psi_1$ and $(w, i) \models \psi_2$. By the induction hypothesis, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration (ψ_1, i) and from configuration (ψ_2, i). By construction, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration $(\psi_1 \wedge \psi_2, i)$. The step case for $\psi = \psi_1 \vee \psi_2$ is analogous.

Consider the case $\psi = \psi_1 \,\mathsf{U}\, \psi_2$. Let $i \in \mathbb{N}$. Assume $(w, i) \models \psi_1 \,\mathsf{U}\, \psi_2$, i.e., there is a $k \geq i$ such that $(w, k) \models \psi_2$ and for all $i \leq j < k$, we have

$(w, j) \models \psi_1$. By the induction hypothesis, this is equivalent to the fact that (i) there is a $k \geq i$ such that $\mathcal{A}$ accepts w from configuration (ψ_2, k) and $\mathcal{A}$ accepts w from configuration (ψ_1, j), for all $i \leq j < k$. We claim that this is equivalent to the fact that (ii) $\mathcal{A}$ accepts w from configuration $(\psi_1 \mathsf{U} \psi_2, i)$.
We first show the direction from left to right. Assume that (i) holds. By the induction hypothesis, $\mathcal{A}$ accepts w from configuration (ψ_2, k). Thus, by definition of the transition functions, $\mathcal{A}$ also accepts w from configuration $(\psi_1 \mathsf{U} \psi_2, k)$. Furthermore, by assumption and the induction hypothesis, $\mathcal{A}$ accepts w from configuration $(\psi_1, k-1)$. Thus, by the definition of the transition function, $\mathcal{A}$ also accepts w from configuration $(\psi_1 \mathsf{U} \psi_2, k-1)$. If we iterate this argumentation, we infer that $\mathcal{A}$ accepts w_j from configuration $(\psi_1 \mathsf{U} \psi_2, j)$, for all $i \leq j < k$. Thus, (ii) holds.
For the other direction, assume that the condition (ii) holds. Let r be an accepting run of $\mathcal{A}$ on w from configuration $(\psi_1 \mathsf{U} \psi_2, i)$. For the sake of contradiction, we additionally assume that (i) does not hold, i.e., we have $(\neg i)$: there is no $k \geq i$ such that $\mathcal{A}$ accepts w from configuration (ψ_2, k) and $\mathcal{A}$ accepts from configuration (ψ_1, j), for all $i \leq j < k$. From $(\neg i)$, it follows that $\mathcal{A}$ does not accept w from configuration (ψ_2, i). By assumption, $\mathcal{A}$ accepts w from configuration $(\psi_1 \mathsf{U} \psi_2, i)$. Hence, by construction of $\mathcal{A}$, it also accepts w from configurations (ψ_1, i) and w from configuration $(\psi_1 \mathsf{U} \psi_2, i+1)$. Again, since $(\neg i)$ holds and $\mathcal{A}$ does not accept w from configuration $(\psi_2, i+1)$, it must accept w from configuration $(\psi_2, i+1)$ and w from configuration $(\psi_1 \mathsf{U} \psi_2, i+2)$. If we repeat this argumentation, we obtain the following infinite rejecting path $(\psi_1 \mathsf{U} \psi_2, i)(\psi_1 \mathsf{U} \psi_2, i+1)(\psi_1 \mathsf{U} \psi_2, i+2)\ldots$ in the run r of $\mathcal{A}$ on w from configuration $(\psi_1 \mathsf{U} \psi_2, i)$. The existence of such a path is a contradiction to the fact that $\mathcal{A}$ accepts w from configuration $(\psi_1 \mathsf{U} \psi_2, i)$ by the run r. The case for $\psi = \psi_1 \mathsf{S} \psi_2$ is analogous.

Consider the case $\psi = \psi_1 \mathsf{R} \psi_2$. Let $i \in \mathbb{N}$. Assume $(w, i) \models \psi_1 \mathsf{R} \psi_2$, i.e., for all $k \geq i$, either $(w, k) \models \psi_2$ or there is a j with $i \leq j < k$ such that $(w, j) \models \psi_1$. By the induction hypothesis, this is equivalent to the fact that (i) for all $k \geq i$, either $\mathcal{A}$ accepts w from configuration (ψ_2, k) or there is a j with $i \leq j < k$ such that $\mathcal{A}$ accepts w from configuration (ψ_1, j). We claim that this is equivalent to the fact that (ii) $\mathcal{A}$ accepts w from configuration $(\psi_1 \mathsf{R} \psi_2, i)$.
We first show the direction from left to right. Assume (i) holds. It is easy to see that (i) is equivalent to the following statement. Either, (a) $\mathcal{A}$ accepts w from configuration (ψ_2, k), for all $k \geq i$, or (b) there is a $k \geq i$ such that

$\mathcal{A}$ accepts from (ψ_1, k) and for all j with $i \leq j \leq k$, we have $\mathcal{A}$ accepts from (ψ_2, j). Assume that the first case holds. We consider the run of $\mathcal{A}$ from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k)$, where $\mathcal{A}$ behaves as follows. Whenever $\mathcal{A}$ arrives in a configuration $(\psi_1 \,\mathsf{R}\, \psi_2, l)$, for $l \geq k$, it moves to configuration (ψ_2, l) and $(\psi_1 \,\mathsf{R}\, \psi_2, l+1)$ respecting the transition function. By assumption, $\mathcal{A}$ accepts from every configuration (ψ_2, l), for $l \geq k$. Thus, the run of $\mathcal{A}$ from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k)$ is accepting if the infinite path $(\psi_1 \,\mathsf{R}\, \psi_2, k)(\psi_1 \,\mathsf{R}\, \psi_2, k+1)\ldots$ is accepting, as well. This path is accepting since $\psi_1 \,\mathsf{R}\, \psi_2$ is an accepting state of $\mathcal{A}$. So, $\mathcal{A}$ accepts w from $(\psi_1 \,\mathsf{R}\, \psi_2, i)$. Assume that the second case holds. Let $k \geq i$ be a position such that $\mathcal{A}$ accepts w from configuration (ψ_1, k) and for all j with $i \leq j \leq k$, $\mathcal{A}$ accepts w from configuration (ψ_2, j). Since $\mathcal{A}$ accepts from (ψ_2, k) and from (ψ_1, k), it follows that by the definition of the transition function, $\mathcal{A}$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. Again, by assumption and the previous step, $\mathcal{A}$ accepts from $(\psi_2, k-1)$ and from $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. Thus, by the definition of the transition function, $\mathcal{A}$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, k-1)$. If we iterate this argumentation, we conclude that for all j with $i \leq j \leq k$, we have $\mathcal{A}$ accepts from $(\psi_1 \,\mathsf{R}\, \psi_2, j)$. Thus, $\mathcal{A}$ accepts w from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, i)$.
Now, we show the other direction by contraposition. Assume that (i) does not hold. That is, $(\neg i)$ there is a $k \geq i$ such that $\mathcal{A}$ does not accept from (ψ_2, k) and for all j with $i \leq j < k$ we have $\mathcal{A}$ does not accept (ψ_1, j). Let $k \geq i$ be the least number such that the $(\neg i)$ holds. In particular, $\mathcal{A}$ does not accept w from (ψ_2, k). By the definition of the transition function, $\mathcal{A}$ does not accept w from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k)$. By assumption, $\mathcal{A}$ does not accept w from configuration $(\psi_1, k-1)$. Thus, $\mathcal{A}$ does not accept w from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, k-1)$, too. If we repeat this argument, we infer that $\mathcal{A}$ does not accept w from configuration $(\psi_1 \,\mathsf{R}\, \psi_2, j)$, for all $i \leq j < k$. Thus (ii) does not hold, and we are done. The step case for $\psi = \psi_1 \,\mathsf{T}\, \psi_2$ is analogous.

Consider the case $\psi = r \diamond\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume $(w, i) \models \psi$, i.e, there is a position $k \geq i$ such that $w_{i..k} \in L(r)$ and $(w, k) \models \gamma$. By the induction hypothesis, this is equivalent to the fact that there is a $k \geq i$ such that $w_{i..k} \in L(\alpha)$ and $\mathcal{A}$ accepts w from configuration (γ, k). That is, $\mathcal{A}$ accepts from configuration $(r \diamond\!\!\rightarrow \gamma, i)$ if and only if there is a position k such that $\mathcal{A}$ has an accepting run on $w_{i..k}$ and $\mathcal{A}$ accepts from (γ, k). It is easy to see that by the definition of the transition function, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration $(r \diamond\!\!\rightarrow \gamma, i)$. The step case for $\psi = \alpha \diamond\!\!\!\!\rightarrow \gamma$ is analogous.

Consider the case $\psi = r \,\Box\!\!\rightarrow \gamma$. Let $i \in \mathbb{N}$. Assume $(w, i) \models \psi$, i.e, for all

positions $k \geq i$ such that $w_{i..k} \in L(\alpha)$, we have $(w, k) \models \gamma$. By the induction hypothesis, this is equivalent to the fact that for all positions $k \geq i$ such that $w_{i..k} \in L(\alpha)$, we have $\mathcal{A}$ accepts w from configuration (γ, k). This is equivalent to the fact that there exists a run of $\mathcal{A}$ on w from configuration $(r \,\Box\!\!\rightarrow \gamma, i)$ such that for every path in the run is labeled by $(q_0, i)(q_1, i+1)\ldots$ the following holds: for all $j \in \mathbb{N}$ such that $(q_0, i)\ldots(q_j, i+j)$ is an accepting run of $\mathcal{A}$ on $w_{i..j}$, the automaton $\mathcal{A}$ accepts w from $(q_j, i+j)$. That is equivalent to the fact that $\mathcal{A}$ accepts w from configuration $(r \,\Box\!\!\rightarrow \gamma, i)$. The step case $\psi = r \boxminus\!\!\rightarrow \gamma$ is analogous.

Eventually 1-Wayness We show that $\mathcal{A}$ is eventually 1-way. For the ease of exposition, we assume that the ε-moves of $\mathcal{A}$ from the states of the form $r \star \alpha$ are eliminated, where r is a SERE and $\star \in \{\Diamond\!\!\rightarrow, \diamondminus\!\!\rightarrow, \Box\!\!\rightarrow, \boxminus\!\!\rightarrow\}$. Let $Q^- := \{q \in \mathsf{Sub}(\varphi) \mid q \text{ is of the form } \alpha S \beta \text{ or } \alpha T \beta\} \cup \{q \in \hat{Q} \mid q \text{ is of the form } s \diamondminus\!\!\rightarrow \alpha \text{ or } s \boxminus\!\!\rightarrow \alpha\}$ denote the states that are built by past operators.

For defining the partitioning of the state set Q, we need the following function that assigns weights to states.

$$\mathsf{weight}(q) := \begin{cases} 2|\mathsf{Sub}(q)| + 1 & \text{if } q \in Q^-, \\ 2|\mathsf{Sub}(q)| & \text{otherwise.} \end{cases}$$

Let $n := 2|Q| + 1$. Let $(Q_i)_{i \leq n}$ be a partitioning of Q, where for $i \leq [n]$, we define $Q_i := \{q \mid \mathsf{weight}(q) = i\}$.

Let $p, q \in Q$, $D \subseteq \mathbb{D}$, $d \in D$, and $a \in \Sigma$ such that $(q, d) \in \delta_D(p, a)$. It suffices to show the following claim: if ($\mathsf{weight}(p)$ is even and $d \leq 0$) or ($\mathsf{weight}(p)$ is odd and $d \geq 0$) then $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q^-$. We have $\mathsf{weight}(p)$ is odd. Assume $d \geq 0$. By the definition of the transition function, $d \neq 1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q \setminus Q^-$. We have $\mathsf{weight}(p)$ is odd. Assume $d \leq 0$. By the definition of the transition function, $d \neq -1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$. ∎

Finally, we translate the locally and eventually 1-way 2ABA $\mathcal{A}$ obtained from the PPSL formula into a language-equivalent 1NBA. Using our scheme for removing alternation and appropriate complementation constructions from Section 4.1, we directly obtain the following result.

	PLTL	PPSL$^{\mathsf{re}}$	PPSL
no past operators	$\mathcal{O}(2^n n)$	$\mathcal{O}(3^n)$	$\mathcal{O}(3^{2^n})$
with past operators	$\mathcal{O}(2^m \cdot 2^n n)$	$\mathcal{O}(2^m \cdot 3^n)$	$\mathcal{O}(2^m \cdot 3^{2^n})$

Table 4.2: Sizes of 1NBAs obtained from PPSL formulas.

Corollary 4.18 *Let φ be a PPSL formula in positive normal form that has m propositions and is of size n. We can translate the formula into a language-equivalent 1NBA $\mathcal{A}$ with*

$$|\mathcal{A}| \in \begin{cases} \mathcal{O}(2^n n) & \text{if } \varphi \text{ is an LTL formula,} \\ \mathcal{O}(2^m \cdot 2^n n) & \text{if } \varphi \text{ is a PLTL formula,} \\ \mathcal{O}(3^n) & \text{if } \varphi \text{ is a PSL}^{\mathsf{re}} \text{ formula,} \\ \mathcal{O}(2^m \cdot 3^n) & \text{if } \varphi \text{ is a PPSL}^{\mathsf{re}} \text{ formula,} \\ \mathcal{O}(3^{2^n}) & \text{if } \varphi \text{ is a PSL formula,} \\ \mathcal{O}(2^m \cdot 3^{2^n}) & \text{if } \varphi \text{ is a PPSL formula.} \end{cases}$$

□

Table 4.2 summarizes the worst-case bounds of the 1NBAs for the corresponding formulas with m propositions and of size n that have no past operators, in the first line, and the bounds for formulas that have past operators, in the second line.

4.2.3 Succinctness Results

In this section, we examine several succinctness gaps between the logics that we have introduced in Section 4.2.1. Let L and L' be two logics from Section 4.2.1. We call L exponentially more *succinct* than L', if there is a family of L formulas $(\varphi_n)_{n>0}$ such that for every $i > 0$ and every L' formula ψ that is initially equivalent to φ_n, the size of ψ is exponential in the size of φ_n. The language L is double-exponentially more succinct than L' if ψ is double-exponential in the size of φ_n. Figure 4.3 summarizes the results of this section. For the sake of readability, we define $2_0^x := x$ and $2_k^x := 2^{2_{k-1}^x}$, for $k > 0$.

We proceed as follows. First, we recall a result by Markey to obtain succinctness gaps between PSL and PSL$^{\mathsf{re}}$ and between PPSL$^{\mathsf{re}}$ and PSL$^{\mathsf{re}}$. Then, we present a novel result to obtain succinctness gaps between PPSL and the

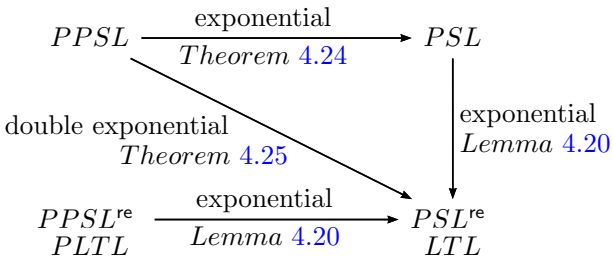


Figure 4.3: Succinctness gaps.

logics PSL, PSL$^{\mathsf{re}}$, and LTL. The proof technique for our new result has a similar flavor to Markey's proof in [Mar03]. However, our proof is more involved since we must take SEREs into account.

Gaps Inferred from Markey's Result In the following, let $n > 0$, $\mathcal{P}_n$ be the set of propositions $\{p_0, \ldots, p_n\}$, and Σ_n be the alphabet $2^{\mathcal{P}_n}$. Consider the following language that states that for any position, p_0's truth value is equal to its corresponding truth value at the initial position whenever the truth values of the propositions $p_1, \ldots, p_n$ are equal to the corresponding truth values at the initial position.

> M_n consists of all words $w \in \Sigma_n^\omega$ such that for every position $i \in \mathbb{N}$, we have $w_i \cap \{p_0\} = w_0 \cap \{p_0\}$ whenever $w_i \setminus \{p_0\} = w_0 \setminus \{p_0\}$.

The next theorem directly follows from Markey's proof in [Mar03].

Theorem 4.19 *Let L be a logic such that for all $n > 0$ there is an L formula of size $\mathcal{O}(n)$ that describes M_n. Let L' be a temporal logic such that*

1. *L' has no past operators,*
2. *L' has the generally operator G, and*
3. *every L' formula of size m can be translated into a language-equivalent 1NBA of size $2^{\mathcal{O}(m)}$.*

Then, the logic L is exponentially more succinct than the logic L'. □

Lemma 4.20 *PSL and PPSL$^{\mathsf{re}}$ are exponentially more succinct than PSL$^{\mathsf{re}}$ and LTL.* □

Proof Let $n > 0$ be a natural number. We present a PSL and a PLTL formula that are linear in n and describe the language M_n. The PSL formula is $r \,\square\!\!\rightarrow \mathsf{ff}$, where r is the SERE

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

The PLTL formula, which is also a PPSL$^{\mathsf{re}}$ formula, is

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

By Theorem 4.19, we obtain the succinctness gaps. ∎

Novel Construction and Gaps Let us now turn to the succinctness gaps between PPSL and the logics PSL, PSL$^{\mathsf{re}}$, and LTL. For this, we first introduce n-counting words, which can be defined in LTL by formulas of size $\mathcal{O}(n)$. In the following, let $n > 0$, $\mathcal{P}_n$ be the set $\{c_0, \ldots, c_{n-1}, p, q\}$ of propositions, and Σ_n the alphabet $2^{\mathcal{P}_n}$. The *n-value* of the letter $b \in \Sigma_n$ is

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

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

Lemma 4.21 *For every $n > 0$, there is an LTL formula count_n of size $\mathcal{O}(n)$ such that $L^\omega(\mathit{count}_n) \subseteq \Sigma_n^\omega$ is the language of n-counting words.* □

Proof Recall that the temporal operators G and X can easily be defined in PSL and PSL$^{\mathsf{re}}$ by using the operator $\Diamond\!\!\rightarrow$.

We define count_n as the LTL formula

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

Note that for i with $1 \le i < n$, the formula $c_i \leftrightarrow (c_{i-1} \rightarrow \mathsf{X} c_{i-1})$ is equivalent to the formula $\big(c_i \wedge \neg \mathit{carry}_{i-1}\big) \vee \big(\neg c_i \wedge \mathit{carry}_{i-1}\big)$, where $\mathit{carry}_{i-1} := c_{i-1} \rightarrow \mathsf{X} c_{i-1}$. It is easily checked that $w \in \Sigma_n^\omega$ is a model of count_n if and only if w is n-counting. ∎

An *n-segment* of a word $w \in \Sigma_n^\omega$ is a subword $v = w_i \dots w_{i+2^n-1}$ 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 two 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 4.22 *For every $n > 0$, there is a PPSL formula φ_n of size $\mathcal{O}(n)$ such that $L^\omega(\varphi_n) = L_n$.* □

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

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

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

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

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

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

For an n-counting word $w \in \Sigma_n^\omega$ and $i \in \mathbb{N}$, it holds that $w, i \models \mathit{back}_n^\psi$ if and only if $w, i \mod 2^n \models \psi$. Intuitively, back_n^ψ goes back in the word w until it reaches the position in the initial n-segment with same counter values as the

current position, and there it checks that ψ holds. Next, we define the SERE $\mathit{within}_n := (\neg c_{n-1})^* \,;\, (c_{n-1})^*$. We use it for checking if a larger position than the current position is still in the same n-segment of an n-counting word. Note that the highest bit c_{n-1} of the counter is only allowed to change its value from 0 to 1 once. The formula $\mathit{start}_n := \bigwedge_{0 \le i < n} \neg c_i$ checks that a position is the first one of an n-segment in an n-counting word.

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

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

The formula ψ_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 $L^\omega(\varphi_n) = L_n$. Furthermore, the size of the formula φ_n is in $\mathcal{O}(n)$. ∎

Lemma 4.23 *For every $n > 0$, if $\mathcal{A}$ is an NBA with $L^\omega(\mathcal{A}) = L'_n$ then $\|\mathcal{A}\| \geq 2_3^n$.* □

Proof Consider a natural number $n > 0$. Let m be the bound 2_2^n. Let $v^0, \ldots, v^{m-1} \in \{\emptyset, \{p\}\}^*$ be an enumeration of all pairwise different words of length 2^n. Let Σ be the alphabet $\Sigma_n \cup \{p, q\}$. We define the language $S_n \subseteq \Sigma^*$ such that for every word $w \in S_n$,

1. the projection of w on Σ_n is n-counting and
2. the projection of w on $\{\emptyset, \{p\}\}$ is the word $v^0 v^1 \ldots v^{m-1}$.

Note that S_n contains exactly $2^{m \cdot 2^n}$ different words that only differ in the distribution of the proposition q. Also, for every $w \in S_n$ we have $w^\omega \in L'_n$.

Suppose that $\|\mathcal{A}\| < |S_n|$. Then, by the pigeon hole principle, there are words $v, w \in S_n$ with $v \neq w$ such that $\mathcal{A}$ visits the same state s when reading the prefix v of the word v^ω and when reading the prefix w of the word w^ω. Hence, $\mathcal{A}$ also accepts the word vw^ω even though $vw^\omega \notin L'_n$. Thus, $\|\mathcal{A}\| \geq |S_n| > 2_3^n$. ∎

With the above lemmas we obtain our succinctness results.

Theorem 4.24 *PPSL is exponentially more succinct than PSL.* □

Proof Let $n > 0$ be a natural number and φ_n denote the PPSL formula from Lemma 4.22. Suppose that ψ is a PSL formula that is initially equivalent to φ_n. We define $\psi' := \mathit{count}_n \wedge \mathsf{G}(\neg c_0 \wedge \ldots \wedge \neg c_{n-1} \rightarrow \psi)$. Note that ψ' describes the language L'_n. By Theorem 4.18, there is an 1NBA $\mathcal{A}$ of size $2_2^{\mathcal{O}(\|\psi'\|)}$ and $L^\omega(\mathcal{A}) = L(\psi')$. By Lemma 4.23, we have $\|\mathcal{A}\| \geq 2_3^n$. It follows that $\|\psi'\| \in \Omega(2^{\|\varphi_n\|})$. Since ψ' is linear in the size of ψ, we conclude that $\|\psi\| \in \Omega(2^{\|\varphi_n\|})$. ∎

Note that L_n is a star-free language. This means, there is an LTL formula φ_n such that $L^\omega(\varphi_n) = L_n$. We can easily adapt the proof of Theorem 4.24 to obtain a double exponential succinctness gap between PPSL and the logics PSL$^{\mathsf{re}}$ and LTL.

Theorem 4.25 *PPSL is double-exponentially more succinct than PSL$^{\mathsf{re}}$ and LTL.* □

Proof Let $n > 0$ be a natural number. Let φ_n be the PPSL formula from Lemma 4.22. Suppose that ψ is an LTL formula that is initially equivalent to φ_n. Let $\psi' := \mathit{count}_n \wedge \mathsf{G}(\neg c_0 \wedge \ldots \wedge \neg c_{n-1} \rightarrow \psi)$. Note that ψ' describes the language L'_n. By Theorem 4.17, there is an 1NBA $\mathcal{A}$ of size $2^{\mathcal{O}(\|\psi'\|)}$ and $L^\omega(\mathcal{A}) = L(\psi')$. By Lemma 4.23, we have $\|\mathcal{A}\| \geq 2_3^n$. It follows that $\|\psi'\| \in \Omega(2_2^{\|\varphi_n\|})$. Since ψ' is linear in the size of φ_n, we conclude that $\|\psi\| \in \Omega(2_2^{\|\varphi_n\|})$. In case, ψ is a PSL$^{\mathsf{re}}$ formula, we obtain $\|\psi\| \in \Omega(2_2^{\|\varphi_n\|})$ by the same argumentation. ∎

Remark 4.26 We conclude this section by stating some open problems related to the presented succinctness gaps. First, it is open whether the exponential succinctness gap still holds between PPSL and extensions of PSL with restricted variants of the past operators like the ones discussed in Remark 4.13. We succeeded neither in proving such a gap nor in expressing the languages L_n concisely in such an extension. Second, it is open whether the succinctness gaps carry over to a fixed and finite proposition set. Note that the proposition sets P_n over which the PPSL formulas φ_n are defined grow linearly in n. As shown in [DS02], 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 4.22 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 PPSL and PSL. Note that for similar reasons, the adapted proof of the succinctness gap between PLTL and LTL in [Mar03, LMS02] for a fixed and finite proposition set also only shows that PLTL is superpolynomially more succinct than LTL. □

4.3 Translations for Extensions of PSL

The two linear-time temporal logics DLTL [HT99] and RLTL [LS07] extend the IEEE standard PSL by more general fix-point operators. We first present a translation from DLTL to 1NBAs that improves over the translation given in [HT99]. For a formula of size n, we obtain a 1NBA of size $\mathcal{O}(3^n)$ by our translation and a 1NBA of size $\mathcal{O}(3^n 2^{2n})$ by the translation in [HT99]. Furthermore, our translation is simpler since it is based on standard automata constructions. We also extend DLTL by past operators (PDLTL) and utilize an alternation-elimination construction from this chapter to obtain a translation from PDLTL to 1NBAs.

Second, we extend RTL by past operators (PRTL) and show that every PRTL formula can be translated into an alternating co-Büchi automaton. Utilizing our alternation-elimination scheme and the construction from Theorem 5.15, we obtain a translation from PRTL into 1NBAs whose worst-case sizes are the same as the worst-case sizes of the 1NBAs obtained from the translation from RTL to 1NBAs given in [LS07]. We remark that in [SL10], Leucker and Sánchez also extends RTL by past operators (pRTL). However, the linear-time temporal logic μLTL [BB89, Var88] generalizes over their suggested logic since all pRTL formulas can be rewritten as μLTL formulas with only a linear blow-up when using construction techniques from [Lan07]. Since the translation from pRTL to 1NBAs does not improve over Vardi's translation of μLTL to 1NBAs, we see no advantage of using pRTL over using μLTL. In contrast, our extension of RTL is as expressive as μLTL and the presented translation from PRTL to 1NBAs improve over the translation from μLTL to 1NBA.

We fix the set of atomic propositions $\mathcal{P}$ and the finite alphabet $\Sigma := 2^{\mathcal{P}}$.

4.3.1 Translations for the Logic DLTL

A variant of the logic PSL is *Dynamic Linear-Time Logic* (*DLTL*). In [HT99], Henriksen and Thiagarajan introduce this logic as an extension of LTL by a modified until operator U^r whose second argument must hold at the end of some sequence that is described by the regular expression r.

Remark 4.27 In dynamic logics [HKT00, HT99], we distinguish between sequences of *actions* A that describe possible executions of a program, and propositions $\mathcal{P}$ that may or may not hold after some program execution $a_0 a_1 \dots a_i \in A^*$, for $i \in \mathbb{N}$. Let $\Sigma = A \times 2^{\mathcal{P}}$ be a finite alphabet. For a word $w \in \Sigma^\omega$, we write $w^A \in A^\omega$ for the projection of w on A. The logic DLTL describes languages of the form $L \subseteq \Sigma^\omega$ such that for every two words $v, w \in L$ and $i \in \mathbb{N}$, if $v^A_{0..i} = w^A_{0..i}$ then $v_i = w_i$. In this thesis, we identify program actions and set of propositions. That is, we set $A \subseteq 2^{\mathcal{P}}$ and only consider words over $(2^{\mathcal{P}})^\omega$. □

In the following, we consider the logic *PDLTL* that is an extension of DLTL by the past operator S^r. The syntax of a PDLTL formula over $\mathcal{P}$ is given by the grammar

$$\varphi ::= b \mid \varphi \vee \varphi \mid \neg\varphi \mid \varphi\, \mathsf{U}^r\, \varphi \mid \varphi\, \mathsf{S}^r\, \varphi,$$

where $b \in \mathcal{B}(\mathcal{P})$ and r is a regular expression. The *size* $|\varphi|$ of a PDLTL formula φ is its syntactic length. We write $\mathsf{Sub}(\varphi)$ for the set of all sub-formulas of a PDLTL formula φ.

For a RE r and formulas φ and ψ, we define the following syntactic abbreviations $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\varphi\, \mathsf{R}^r\, \psi := \neg(\neg\varphi\, \mathsf{U}^r\, \neg\psi)$, $\varphi\, \mathsf{T}^r\, \psi := \neg(\neg\varphi\, \mathsf{S}^r\, \neg\psi)$. Using these abbreviations, we can obviously translate any PDLTL formula into *positive normal form*, i.e., negations occur only in front of propositional logic formulas. Note that such a translation might double the size of the formula.

We proceed with defining the semantics of PDLTL. Let Σ denote the set of propositions $2^{\mathcal{P}}$. Let $w \in \Sigma^\omega$ and $i \in \mathbb{N}$ be a position in w.

$$
\begin{array}{lll}
(w,i) \models b & \text{iff} & w_i \text{ fulfills } b. \\
(w,i) \models \varphi \vee \psi & \text{iff} & (w,i) \models \varphi \text{ or } (w,i) \models \psi. \\
(w,i) \models \neg\varphi & \text{iff} & (w,i) \not\models \varphi. \\
(w,i) \models \varphi\, \mathsf{U}^r\, \psi & \text{iff} & \text{there is a } k \geq i \text{ such that } (w,k) \models \psi, \\
 & & \quad L^*(r) \text{ contains } w_{i..k}, \text{ and} \\
 & & \quad \text{for all } j \text{ with } i \leq j < k, \text{ we have } (w,j) \models \varphi. \\
(w,i) \models \varphi\, \mathsf{S}^r\, \psi & \text{iff} & \text{there is a } k \leq i \text{ such that } (w,k) \models \psi, \\
 & & \quad L^*(r) \text{ contains } w_{i..k}, \text{ and} \\
 & & \quad \text{for all } j \text{ with } k < j \leq i, \text{ we have } (w,j) \models \varphi.
\end{array}
$$

For a PDLTL formula φ, we write $L^\omega(\varphi) := \{w \in \Sigma^\omega \mid (w,0) \models \varphi\}$ to denote its language, $|\varphi|$ for the size of φ that is defined as its syntactic length, and write $\mathsf{Sub}(\varphi)$ for the set of its sub-formulas.

In the next theorem, we present a construction to translate a PDLTL formula into a language-equivalent eventually 1-way 2ABA.

Theorem 4.28 *We can translate every PDLTL formula in positive-normal form of size n into a language-equivalent locally and eventually 1-way 2ABA of size $\mathcal{O}(n)$. For DLTL formulas, the resulting automaton is 1-way.* □

Proof Let φ be a PDLTL formula. We translate this formula into a language-equivalent locally and eventually 1-way 2ABA.

For every RE r in φ, let $\mathcal{A}_r$ and $\mathcal{A}'_r$ be the corresponding automata constructed according to Lemma 4.16 such that $L^*(r) = L^*(\mathcal{A}_r)$ and $\mathcal{A}'_r$ accepts the mirror language of $L^*(r)$. We assume that the state sets of these automata are pairwise disjoint. In the following, we (*i*) present a construction of the 2ABA, (*ii*) prove the correctness of the construction, and (*iii*) finally show that the 2ABA is eventually 1-way.

Construction We define the 2ABA $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$, where the set of states is $Q := \mathsf{Sub}(\varphi) \cup \hat{Q}$ with

$$
\begin{aligned}
\hat{Q} := &\{\alpha\, \mathsf{U}^s\, \beta \mid \alpha\, \mathsf{U}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\
&\{\alpha\, \mathsf{R}^s\, \beta \mid \alpha\, \mathsf{R}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\
&\{\alpha\, \mathsf{S}^s\, \beta \mid \alpha\, \mathsf{S}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}'_r\} \cup \\
&\{\alpha\, \mathsf{T}^s\, \beta \mid \alpha\, \mathsf{T}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}'_r\}.
\end{aligned}
$$

The initial state q_I is φ. The set of accepting states

$$F := Q \setminus \{\alpha \,\mathsf{U}^s\, \beta \mid \alpha \,\mathsf{U}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\}$$

contains all states except for the until formulas.

We proceed to define the transition function δ. Let $a \in \Sigma$ and $D \subseteq \mathbb{D}$. First, we define the transitions from states that are LTL formulas.

- For $b \in \mathcal{B}(\mathcal{P})$, we define

$$\delta_D(b, a) := \begin{cases} \mathsf{tt} & \text{if } a \text{ satisfies } b, \\ \mathsf{ff} & \text{otherwise.} \end{cases}$$

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

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

We now turn to the transitions for states that correspond to the temporal operators with the REs.

- The state $\alpha \,\mathsf{U}^r\, \beta \in \mathsf{Sub}(\varphi)$ is used to start a simulation of the NFA $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ 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 β. Furthermore, the simulation must visit state α at every step until the final state is reached. Formally, we define $\delta_D(\alpha \,\mathsf{U}^r\, \beta, a) := (\alpha \,\mathsf{U}^{s_I}\, \beta, 0)$ and for $s \in S$,

$$\delta_D(\alpha \,\mathsf{U}^s\, \beta, a) := \begin{cases} (\alpha, 0) \wedge \bigvee_{t \in \eta(s,a)} (\alpha \,\mathsf{U}^t\, \beta, 1) & \text{if } \eta(s, a) \cap E = \emptyset, \\ \big((\alpha, 0) \wedge \bigvee_{t \in \eta(s,a)} (\alpha \,\mathsf{U}^t\, \beta, 1)\big) \vee (\beta, 0) & \text{otherwise.} \end{cases}$$

  The transitions from the state $\alpha \,\mathsf{S}^r\, \beta \in \mathsf{Sub}(\varphi)$ are defined similarly. Instead of simulating the NFA $\mathcal{A}_r$, $\mathcal{A}_\varphi$ simulates the NFA $\mathcal{A}'_r$, where it moves the read-only head to the left instead of to the right.

- If the state is $\alpha \,\mathsf{R}^r\, \beta \in \mathsf{Sub}(\varphi)$, the automaton $\mathcal{A}_\varphi$ simulates a run of the NFA $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ viewed as a universal automaton. Whenever the simulation reaches a final state, $\mathcal{A}_r$ has to proceed with the state ψ.

Alternatively, the simulation may break up and proceed from state α. Formally, we define $\delta_D(\alpha\, \mathsf{R}^r\, \beta, a) := (\alpha\, \mathsf{R}^{s_I}\, \beta, 0)$ and for $s \in S$,

$$\delta_D(\alpha\, \mathsf{R}^s\, \beta, a) := \begin{cases} (\alpha, 0) \vee \bigwedge_{t \in \eta(s,a)} (\alpha\, \mathsf{R}^t\, \beta, 1) & \text{if } \eta(s,a) \cap E = \emptyset, \\ \big((\alpha, 0) \vee \bigwedge_{t \in \eta(s,a)} (\alpha\, \mathsf{R}^t\, \beta, 1)\big) \wedge (\beta, 0) & \text{otherwise.} \end{cases}$$

The transitions from state $\alpha\, \mathsf{T}^r\, \beta \in \mathsf{Sub}(\varphi)$ are similarly defined. However, if the read-only head is at the beginning of the input word, $\mathcal{A}_r$ may stop the simulation. Formally, for the NFA $\mathcal{A}'_r = (S, \Sigma, \eta, s_I, E)$ and $s \in S$, we define $\delta_D(\alpha\, \mathsf{T}^r\, \beta, a) := (\alpha\, \mathsf{R}^{s_I}\, \beta, 0)$ and for $-1 \notin D$, we have

$$\delta_D(\alpha\, \mathsf{T}^s\, \beta, a) := \begin{cases} \mathsf{tt} & \text{if } \eta(s,a) \cap E = \emptyset, \\ (\beta, 0) & \text{otherwise,} \end{cases}$$

and for $-1 \in D$, we have

$$\delta_D(\alpha\, \mathsf{T}^s\, \beta, a) := \begin{cases} (\alpha, 0) \vee \bigwedge_{t \in \eta(s,a)} (\alpha\, \mathsf{T}^t\, \beta, -1) & \text{if } \eta(s,a) \cap E = \emptyset, \\ \big((\alpha, 0) \vee \bigwedge_{t \in \eta(s,a)} (\alpha\, \mathsf{T}^t\, \beta, -1)\big) \wedge (\beta, 0) & \text{otherwise.} \end{cases}$$

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

Note that from the definition of the state set Q and Lemma 4.16, we directly obtain $|\mathcal{A}| \in \mathcal{O}(n)$. By inspecting $\mathcal{A}$'s transition function, we also see that $\mathcal{A}$ is locally 1-way.

Correctness In the remainder of the proof, we show the correctness of the given construction. In particular, we prove that for every word $w \in \Sigma^\omega$, subformula $\psi \in \mathsf{Sub}(\varphi)$, and position $i \in \mathbb{N}$, the following equivalence holds

$$(w, i) \models \psi \quad \text{if and only if} \quad \mathcal{A} \text{ accepts } w \text{ from configuration } (\psi, i).$$

This equivalence immediately implies $L^\omega(\mathcal{A}) = L^\omega(\varphi)$. We prove the equivalence by induction over the formula structure of ψ. Let $w \in \Sigma^\omega$ be an infinite word and $i \in \mathbb{N}$ a position in w.

Consider the base case $\psi = b$, for some $b \in \mathcal{B}(\mathcal{P})$. By definition, $(w, i) \models b$ holds if and only if w_i fulfills b. By construction, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration (b, i).

Consider the case $\psi = \alpha \wedge \beta$. Assume $(w, i) \models \psi$, i.e., $(w, i) \models \alpha$ and $(w, i) \models \beta$. By the induction hypothesis, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration (α, i) and from configuration (β, i). By construction, this is equivalent to the fact that $\mathcal{A}$ accepts w from configuration $(\alpha \wedge \beta, i)$. The step case for $\psi = \alpha \vee \beta$ is analogous.

Consider the case $\psi = \alpha \,\mathsf{U}^r\, \beta$. Let $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ be an 1NFA that accepts $L^*(r)$. Assume $(w, i) \models \alpha \,\mathsf{U}^r\, \beta$, i.e., there is a $k \geq i$ such that $(w, k) \models \beta$, $L^*(r)$ contains $w_{i..k}$, and for all j with $i \leq j < k$, we have $(w, j) \models \alpha$. By the induction hypothesis, this is equivalent to the fact that

> there is a $k \geq i$ such that $\mathcal{A}$ accepts w from configuration (β, k), $L^*(r)$ contains $w_{i..k}$, and $\mathcal{A}$ accepts w from configuration (α, j), for all $i \leq j < k$. (i)

We claim that this is equivalent to the fact that

> $\mathcal{A}$ accepts w from configuration $(\alpha \,\mathsf{U}^r\, \beta, i)$. (ii)

We first show the direction from left to right. Assume that (i) holds. Let $s_i s_{i+1} \dots s_{k+1} \in S^*$ be an accepting run of $\mathcal{A}_r$ on $w_{i..k}$. By assumption, $\mathcal{A}$ accepts w from configuration (β, k). Thus, by the definition of the transition functions, $\mathcal{A}$ also accepts w from configuration $(\alpha \,\mathsf{U}^{s_k}\, \beta, k)$. Furthermore, by assumption, $\mathcal{A}$ accepts w from configuration $(\alpha, k-1)$. Thus, by the definition of the transition function, $\mathcal{A}$ also accepts w from configuration $(\alpha \mathsf{U}^{s_{k-1}} \beta, k-1)$. If we iterate this argumentation, we infer that $\mathcal{A}$ accepts w_j from configuration $(\alpha \,\mathsf{U}^{s_j}\, \beta, j)$, for all $i \leq j < k$. Since $(\alpha \,\mathsf{U}^{s_i}\, \beta, i)$ is reached from $(\alpha \,\mathsf{U}^r\, \beta, i)$ by an ε-moves, we obtain (ii).

For the other direction, assume that the (ii) holds. That is, $\mathcal{A}$ accepts w from configuration $(\alpha \,\mathsf{U}^r\, \beta, i)$. In particular, $\mathcal{A}$ accepts w from configuration $(\alpha \,\mathsf{U}^{s_I}\, \beta, i)$. For the sake of contradiction, we additionally assume that (i) does not hold, that is, we have

> there is no $k \geq i$ such that $\mathcal{A}$ accepts w from configuration (β, k), $L^*(r)$ contains $w_{i..k}$, and $\mathcal{A}$ accepts from configuration (α, j), for all $i \leq j < k$. $(\neg i)$

From ($\neg i$), it follows that $\mathcal{A}$ does not accept w from configuration (β, i). Hence, $\mathcal{A}$ accepts w from configurations (α, i) and there is also a successor s_1 of s_I such that $\mathcal{A}$ accepts w from configuration $(\alpha\, \mathsf{U}^{s_1}\, \beta, i+1)$. Again, since ($\neg i$) holds and $\mathcal{A}$ does not accept w from configuration $(\beta, i+1)$, it must accept w from configuration $(\alpha, i+1)$ and there is also a successor s_2 of s_1 such that $\mathcal{A}$ accepts w from configuration $(\alpha\, \mathsf{U}^{s_2}\, \beta, i+2)$. If we repeat this argumentation, we obtain an infinite sequence of states $s_I s_1 s_2 \ldots \in S^\omega$ and the following infinite rejecting path $(\alpha\, \mathsf{U}^{s_I}\, \beta, i)(\alpha\, \mathsf{U}^{s_1}\, \beta, i+1)(\alpha\, \mathsf{U}^{s_2}\, \beta, i+2)\ldots$ in the run of $\mathcal{A}$ on w from configuration $(\alpha\, \mathsf{U}^{r}\, \beta, i)$. The existence of such a path is a contradiction to the fact that $\mathcal{A}$ accepts w from configuration $(\alpha\, \mathsf{U}^{r}\, \beta, i)$. The case for $\psi = \alpha\, \mathsf{S}^r\, \beta$ is analogous.

Consider the case $\psi = \alpha\, \mathsf{R}^r\, \beta$. Let $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ be the NFA that accepts $L^*(r)$. Assume $(w, i) \models \alpha\, \mathsf{R}^r\, \beta$, i.e., for all $k \geq i$, if $L^*(r)$ contains $w_{i..k}$ then either $(w, k) \models \beta$ or there is a j with $i \leq j < k$ such that $(w, j) \models \alpha$. By the induction hypothesis, this is equivalent to the fact that

> for all $k \geq i$, if $L^*(r)$ contains $w_{i..k}$ then either $\mathcal{A}$ accepts w from configuration (β, k) or there is a j with $i \leq j < k$ such that $\mathcal{A}$ accepts w from configuration (α, j). (iv)

We claim that this is equivalent to the fact that

> $\mathcal{A}$ accepts w from configuration $(\alpha\, \mathsf{R}^r\, \beta, i)$. ($v$)

We first show the direction from left to right. Assume (iv) holds. It is easy to see that (iv) is equivalent to the following statement.

> Either for all $k \geq i$, if $L^*(r)$ contains $w_{i..k}$ then $\mathcal{A}$ accepts w from configuration (β, k), or (iv')
>
> there is a $k \geq i$ such that $\mathcal{A}$ accepts w from (α, k) and for all j with $i \leq j \leq k$, if $L^*(r)$ contains $w_{i..j}$ then $\mathcal{A}$ also accepts w from (β, j). (iv'')

Assume that (iv') holds. In the following, we just consider runs of $\mathcal{A}$ from configuration $(\alpha\, \mathsf{R}^{s_I}\, \beta, i)$, where $\mathcal{A}$ behaves as follows. Whenever $\mathcal{A}$ arrives in a configuration of the form $(\alpha\, \mathsf{R}^{s}\, \beta, j)$, for any $j \geq i$, it avoids moving to configuration (α, j). We show by contradiction that $\mathcal{A}$ still accepts from $(\alpha\, \mathsf{R}^{s_I}\, \beta, i)$ Suppose that $\mathcal{A}$ does not accept from $(\alpha\, \mathsf{R}^{s_I}\, \beta, i)$. Due to the

definition of the transition function, there are two cases. If $\eta(s_I, w_i) \cap E = \emptyset$ then there must be a state $s_1 \in \eta(s_I, w_i)$ such that $\mathcal{A}$ rejects from $(\alpha R^{s_1} \beta, i{+}1)$. Otherwise, if $\eta(s_I, w_i) \cap E \neq \emptyset$ then there must be a state $s' \in \eta(s_I, w_i)$ such $s_I s'$ is an accepting run of $\mathcal{A}_r$ on $w_{i..i+1}$. Thus, $\mathcal{A}$ accepts from $(\beta, i{+}1)$. Therefore, as in the first case, there must be a state $s_1 \in \eta(s_I, w_i)$ such that $\mathcal{A}$ rejects from $(\alpha R^{s_1} \beta, i{+}1)$. If we iterate this argument, the only possible way to reject from $(\alpha \mathsf{R}^{s_I} \beta, i)$ is by the path $(\alpha \mathsf{R}^{s_I} \beta, i)(\alpha \mathsf{R}^{s_1} \beta, i{+}1)(\alpha \mathsf{R}^{s_2} \beta, i{+}2) \ldots \in (Q \times \mathbb{N})^\omega$. Since this path is accepting, we conclude that $\mathcal{A}$ accept from $(\alpha \mathsf{R}^{s_I} \beta, i)$.
Assume that (*iv″*) holds. Let $k \geq i$ be the least position such that $\mathcal{A}$ accepts w from configuration (α, k) and for all j with $i \leq j \leq k$, if $L^*(r)$ contains $w_{i..j}$ then $\mathcal{A}$ accepts w from configuration (β, j). In the following, we just consider runs of $\mathcal{A}$ from configuration $(\alpha \mathsf{R}^{s_I} \beta, i)$, where $\mathcal{A}$ behaves as follows. Whenever $\mathcal{A}$ arrives in a configuration of the form $(\alpha \mathsf{R}^{s} \beta, j)$, for any $j \geq i$, it moves to configuration (α, k) if $j = k$ and otherwise, it avoids moving to configuration (α, k). We show by contradiction that $\mathcal{A}$ still accepts from $(\alpha \mathsf{R}^{s_I} \beta, i)$ Suppose that $\mathcal{A}$ does not accept from $(\alpha \mathsf{R}^{s_I} \beta, i)$. Due to the definition of the transition function, there are two cases. If $\eta(s_I, w_i) \cap E = \emptyset$ then there must be a state $s_1 \in \eta(s_I, w_i)$ such that $\mathcal{A}$ rejects from $(\alpha R^{s_1} \beta, i + 1)$. Otherwise, if $\eta(s_I, w_i) \cap E \neq \emptyset$ then there must be a state $s' \in \eta(s_I, w_i)$ such $s_I s'$ is an accepting run of $\mathcal{A}_r$ on $w_{i..i+1}$. Thus, $\mathcal{A}$ accepts from $(\beta, i + 1)$. Therefore, as in the first case, there must be a state $s_1 \in \eta(s_I, w_i)$ such that $\mathcal{A}$ rejects from $(\alpha R^{s_1} \beta, i + 1)$. If we iterate this argument, we obtain the only possible way to reject from $(\alpha \mathsf{R}^{s_I} \beta, i)$ is by a the path that starts with the prefix $(\alpha \mathsf{R}^{s_I} \beta, i)(\alpha \mathsf{R}^{s_1} \beta, i + 1)(\alpha \mathsf{R}^{s_2} \beta, i + 2) \ldots (\alpha \mathsf{R}^{s_j} \beta, i + j) \in (Q \times \mathbb{N})^*$, where $i{+}j = k$. But then, one of the following two cases holds. If $\eta(s_j, w_{i+j}) \cap E = \emptyset$ then $\mathcal{A}$ moves to (α, k) and accepts. Otherwise, if $\eta(s_i, w_{i+j}) \cap E \neq \emptyset$ then there must be a state $s' \in \eta(s_j, w_{i+j})$ such $s_I s_1 \ldots s_{i+j} s'$ is an accepting run of $\mathcal{A}_r$ on $w_{i..i+j}$. Thus, $\mathcal{A}$ accepts from $(\beta, i + j)$. Therefore, as in the first case, there $\mathcal{A}$ moves to (α, k) and accepts. Therefore, there is no rejecting path. We conclude that $\mathcal{A}$ accept from $(\alpha \mathsf{R}^{s_I} \beta, i)$.
Now, we show the other direction by contraposition. Assume that (*iv*) does not hold. That is,

> there is a $k \geq i$ such that $L^*(r)$ contains $w_{i..r}$, $\mathcal{A}$ does not accept w from (β, k) and for all j with $i \leq j < k$, $\mathcal{A}$ does not accept w from (α, j). ($\neg iv$)

Let $k \geq i$ be the least number such that the ($\neg iv$) holds. Furthermore, let

$s_i s_{i+1} \dots s_{k+1}$ be an accepting run of $\mathcal{A}_r$ on $w_{i..k}$. By assumption, $\mathcal{A}$ rejects w from (β, k). Hence, $\mathcal{A}$ rejects w from configuration $(\alpha\, \mathsf{R}^{s_k}\, \beta, k)$, too. By assumption, $\mathcal{A}$ rejects w from configuration $(\alpha, k-1)$. Thus, $\mathcal{A}$ also rejects w from configuration $(\alpha\, \mathsf{R}^{s_{k-1}}\, \beta, k-1)$, too. If we repeat this argument, we infer that $\mathcal{A}$ does not accept w from configuration $(\alpha\, \mathsf{R}^{s_j}\, \beta, j)$, for all $i \leq j < k$. Thus (*v*) does not hold, and we are done. The step case for $\psi = \alpha\, \mathsf{T}^r\, \beta$ is analogous.

Eventually 1-Wayness We show that $\mathcal{A}$ is eventually 1-way. For the ease of exposition, we assume that the ε-moves of $\mathcal{A}$ from the states of the form $\alpha \star^r \beta$ are eliminated, where r is an RE and $\star \in \{\mathsf{U}, \mathsf{R}, \mathsf{S}, \mathsf{T}\}$. Let $Q^- := \{q \in \mathsf{Sub}(\varphi) \mid q \text{ is of the form } \alpha S^r \beta \text{ or } \alpha T^r \beta\}$ denote the states that are built by past operators.

For defining the partitioning of the state set Q, we need the following function that assigns weights to states.

$$\mathsf{weight}(q) := \begin{cases} 2|\mathsf{Sub}(q)| + 1 & \text{if } q \in Q^-, \\ 2|\mathsf{Sub}(q)| & \text{otherwise.} \end{cases}$$

Let $n := 2|Q| + 1$. Let $(Q_i)_{i \leq n}$ be a partitioning of Q, where for $i \leq [n]$, we define $Q_i := \{q \mid \mathsf{weight}(q) = i\}$.

Let $p, q \in Q$, $D \subseteq \mathbb{D}$, $d \in D$, and $a \in \Sigma$ such that $(q, d) \in \delta_D(p, a)$. It suffices to show the following claim: if ($\mathsf{weight}(p)$ is even and $d \leq 0$) or ($\mathsf{weight}(p)$ is odd and $d \geq 0$) then $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q^-$. We have that $\mathsf{weight}(p)$ is odd. Assume $d \geq 0$. By the definition of the transition function, $d \neq 1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q \setminus Q^-$. We have that $\mathsf{weight}(p)$ is odd. Assume $d \leq 0$. By the definition of the transition function, $d \neq -1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$. ∎

Corollary 4.29 *We can translate every PDLTL formula in positive normal form of size n and having m propositions into a language-equivalent 1NBA of size $\mathcal{O}(2^m \cdot 3^n)$.* □

Corollary 4.30 *We can translate every DLTL formula in positive normal form of size n into a language-equivalent 1NBA of size $\mathcal{O}(3^n)$.* □

We remark that the translation in Corollary 4.30 improves over the translation from DLTL to 1NBAs given in [HT99]. For DLTL formulas in positive normal form of size n, the worst-case sizes of the 1NBAs obtained from Henriksen and Thiagarajan's translation is in $\mathcal{O}(3^n 2^{2n})$. Moreover, the constant hidden in the $\mathcal{O}$ notation is four times bigger than the constant in the bound of our translation.

4.3.2 Translations for the Logic PRLTL

Regular Linear-Time Temporal Logic (RLTL) [LS07] is an extension of a fragment of PSL$^{\mathsf{re}}$ by a variant of the until operator U^r that is equipped by a regular expression r. Leucker and Sánchez present a translation of RLTL formulas of size n into to 1NBAs of size $\mathcal{O}(3^n)$. In the follow-up paper [SL10], they extend RLTL with a negation operator in the regular layer and past operators in the regular-expression layer. We call this logic pRLTL. In their paper, Leucker and Sánchez also provide a translation of pRLTL formulas of size n and k nested negations into 1NBAs of size $2^{\mathcal{O}((nk)^2)}$. Although they mention that k can be bounded by the constant 3. However, no details are given.

In this section, we present an alternative extension of RLTL by the negation operator and past operators. We call the logic PRLTL. Based on our alternation-elimination construction in this chapter, we also provide a translation of PRLTL formulas of size n into 1NBAs of size $2^{\mathcal{O}(n \log n)}$.

The syntax of a PRLTL formula over $\mathcal{P}$ is defined as follows.

$$\varphi ::= b \mid \varphi \vee \varphi \mid \neg\varphi \mid \mathsf{E}^r\varphi \mid \varphi\, \mathsf{U}^r\, \varphi \mid \varphi\, \mathsf{S}^r\, \varphi,$$

where $b \in \mathcal{B}(\mathcal{P})$ and r is a RE over $\mathcal{P}$. For a RE r and formulas φ and ψ, we define the following syntactic abbreviations $\varphi \wedge \psi := \neg(\neg\varphi \vee \neg\psi)$, $\mathsf{A}^r\varphi := \neg(\mathsf{E}^r\neg\varphi)$, $\varphi\, \mathsf{R}^r\, \psi := \neg(\neg\varphi\, \mathsf{U}^r\, \neg\psi)$, $\varphi\, \mathsf{T}^r\, \psi := \neg(\neg\varphi\, \mathsf{S}^r\, \neg\psi)$. Using these abbreviations, we can obviously translate any PRLTL formula into positive normal form, i.e., negations occur only in front of propositional logic formulas. Note that such a translation might double the size of the formula.

Let us define the semantics of the operators. Let $w \in (2^{\mathcal{P}})^\omega$ and $i \in \mathbb{N}$.

$$
\begin{array}{lll}
(w,i) \models b & \text{iff} & w_i \text{ satisfies } b. \\
(w,i) \models \varphi \vee \psi & \text{iff} & (w,i) \models \varphi \text{ or } (w,i) \models \psi. \\
(w,i) \models \neg\varphi & \text{iff} & (w,i) \not\models \varphi. \\
(w,i) \models \mathsf{E}^r\varphi & \text{iff} & \exists k \in \mathbb{N}^\omega : k_0 = i \text{ and } \forall j \in \mathbb{N} : \\
 & & \quad k_{j+1} > k_j, w_{k_j..k_{j+1}} \in L^*(r), \text{ and } (w,k_j) \models \varphi. \\
(w,i) \models \varphi\,\mathsf{U}^r\,\psi & \text{iff} & \exists n \in \mathbb{N}, k \in \mathbb{N}^{n+1} : k_0 = i, (w,k_n) \models \psi, \text{ and } \forall j \in [n] : \\
 & & \quad k_{j+1} > k_j, w_{k_j..k_{j+1}} \in L^*(r), \text{ and } (w,k_j) \models \varphi. \\
(w,i) \models \varphi\,\mathsf{S}^r\,\psi & \text{iff} & \exists n \in \mathbb{N}, k \in \mathbb{N}^{n+1} : k_0 = i, (w,k_n) \models \psi, \text{ and } \forall j \in [n] : \\
 & & \quad k_{j+1} < k_j, w_{k_{j+1}..k_j} \in L^*(r), \text{ and } (w,k_j) \models \varphi.
\end{array}
$$

For a PRLTL formula φ, we write $L^\omega(\varphi) := \{w \in \Sigma^\omega \mid (w,0) \models \varphi\}$ to denote its language, $|\varphi|$ for the size of φ that is defined as its syntactic length, and write $\mathsf{Sub}(\varphi)$ for the set of its sub-formulas.

Next, we give a construction to translate PRLTL formulas into locally and eventually 1-way 2ABAs.

Theorem 4.31 *For every PRLTL formula in positive normal form of size n, there is a language-equivalent locally and eventually 1-way 2APA with three priorities and of size $\mathcal{O}(n)$. If the formula has no past operators then the 2APA is 1-way.* □

Proof Let φ be a PRLTL formula in positive normal form. For any regular expression r that occurs in a subformula of φ, let $\mathcal{A}_r$, $\mathcal{A}'_r$ be the NFAs for the language $L^*(r)$ and the mirror language of $L^*(r)$, respectively. We proceed as follows. First, we give the construction and prove its correctness. Then, we show that the constructed automaton is eventually 1-way.

Construction We define the 2APA $\mathcal{A} := (Q, \Sigma, \delta, q_I, \{F_0, F_1, F_2\})$. The set of states is $Q := \mathsf{Sub}(\varphi) \cup Q^+ \cup Q^-$, where

$$
\begin{aligned}
Q^+ := &\{s \Diamond\!\!\rightarrow \mathsf{E}^r\beta \mid \mathsf{E}^r\beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\
&\{s \Box\!\!\rightarrow \mathsf{A}^r\beta \mid \mathsf{A}^r\beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\
&\{s \Diamond\!\!\rightarrow \alpha\,\mathsf{U}^r\,\beta \mid \alpha\,\mathsf{U}^r\,\beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\} \cup \\
&\{s \Box\!\!\rightarrow \alpha\,\mathsf{R}^r\,\beta \mid \alpha\,\mathsf{R}^r\,\beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\}
\end{aligned}
$$

and

$$\begin{aligned} Q^- :=& \{s \diamondsuit\!\!\!\!\rightarrow \alpha\, \mathsf{S}^r\, \beta \mid \alpha\, \mathsf{S}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}'_r\} \cup \\ & \{s \boxplus\!\!\!\!\rightarrow \alpha\, \mathsf{T}^r\, \beta \mid \alpha\, \mathsf{T}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}'_r\}. \end{aligned}$$

The initial state q_I is the input formula φ. The parity acceptance condition is given by the following three sets.

$$\begin{aligned} F_0 :=& \{\mathsf{A}^r\beta \mid \mathsf{A}^r\beta \in \mathsf{Sub}(\varphi)\}, \\ F_1 :=& \{\alpha\, \mathsf{U}^r\, \beta \mid \alpha\, \mathsf{U}^r\, \beta \in \mathsf{Sub}(\varphi)\} \cup \\ & \{s \diamond\!\!\rightarrow \alpha\, \mathsf{U}^r\, \beta \mid \alpha\, \mathsf{U}^r\, \beta \in \mathsf{Sub}(\varphi) \text{ and } s \text{ is a state in } \mathcal{A}_r\}, \text{ and} \\ F_2 :=& Q \setminus (F_0 \cup F_1). \end{aligned}$$

We proceed to define the transition function δ. Consider the transition function δ' from the 2ABA in the proof of Theorem 4.17. For a state $q \in Q$, $D \subseteq \mathbb{D}$, and a letter $a \in \Sigma$, we define

$$\delta_D(q,a) := \begin{cases} (\beta,0) \wedge \delta'_D\big(r \diamond\!\!\rightarrow \mathsf{E}^r\beta, a\big) & \text{if } q \text{ is of the form } \mathsf{E}^r\beta, \\ (\beta,0) \vee \delta'_D\big(r \,\square\!\!\rightarrow \mathsf{A}^r\beta, a\big) & \text{if } q \text{ is of the form } \mathsf{A}^r\beta, \\ (\beta,0) \vee \big((\alpha,0) \wedge \delta'_D(r \diamond\!\!\rightarrow \alpha\, \mathsf{U}^r\, \beta, a)\big) & \text{if } q \text{ is of the form } \alpha\, \mathsf{U}^r\, \beta, \\ (\beta,0) \wedge \big((\alpha,0) \vee \delta'_D(r \,\square\!\!\rightarrow \alpha\, \mathsf{R}^r\, \beta, a)\big) & \text{if } q \text{ is of the form } \alpha\, \mathsf{R}^r\, \beta, \\ (\beta,0) \vee \big((\alpha,0) \wedge \delta'_D(r \diamondsuit\!\!\!\!\rightarrow \alpha\, \mathsf{S}^r\, \beta, a)\big) & \text{if } q \text{ is of the form } \alpha\, \mathsf{S}^r\, \beta, \\ (\beta,0) \wedge \big((\alpha,0) \vee \delta'_D(r \boxplus\!\!\!\!\rightarrow \alpha\, \mathsf{T}^r\, \beta, a)\big) & \text{if } q \text{ is of the form } \alpha\, \mathsf{T}^r\, \beta, \\ \delta'_D(q,a) & \text{otherwise.} \end{cases}$$

Note that by definition, $\mathcal{A}$ has at most $\mathcal{O}(n)$ states and is locally 1-way.

Correctness We show that $L^\omega(\mathcal{A}) = L^\omega(\varphi)$. In particular, we prove that for every word $w \in \Sigma^\omega$, subformula $\psi \in \mathsf{Sub}(\varphi)$, and position $i \in \mathbb{N}$, the following holds.

$$w, i \models \psi \quad \text{if and only if} \quad \mathcal{A} \text{ accepts } w \text{ from configuration } (\psi, i).$$

This equivalence immediately implies $L^\omega(\mathcal{A}) = L^\omega(\varphi)$. We prove the equivalence by induction over the formula structure of ψ. Let $w \in \Sigma^\omega$ and $i \in \mathbb{N}$ be a position in w. We only consider the cases that are different from the ones proven in Theorem 4.17.

Consider the case $\psi = \mathsf{E}^r\beta$. Let $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ denote the NFA that accepts $L^*(r)$. We first show the direction from left to right. We apply the induction hypothesis on the assumption and obtain

$$\text{there is an infinite path } k \in \mathbb{N}^\omega \text{ with } k_0 = i \text{ such that 1. } L^*(r) \text{ contains } w_{k_j..k_{j+1}}, \text{ for all } j \in \mathbb{N}, \text{ and 2. } \mathcal{A} \text{ accepts } w \text{ from } (\beta, k_j), \text{ for all } j \in \mathbb{N}. \tag{i}$$

Consider a sequence of positions $k \in \mathbb{N}^\omega$ that satisfies (i). For the sake of contradiction, suppose $\mathcal{A}$ rejects from (E^r, i). By the second condition of (i), $\mathcal{A}$ must reject from $(r \diamond\!\!\rightarrow \mathsf{E}^r\beta, i)$. By the first condition of (i), there is an accepting run $s_{k_0} \dots s_{k_1+1} \in S^*$ of $\mathcal{A}_r$ on $w_{k_0..k_1}$. Hence, $\mathcal{A}$ must reject from (E^r, k_1). If we iterate this argumentation, we obtain the infinite sequence $(\mathsf{E}^r, k_0)\dots(\mathsf{E}^r, k_1)\dots \in (Q \times \mathbb{N})^\omega$. Furthermore, $\mathcal{A}$ can only reject if this sequence is rejecting. Since this is not the case, we conclude that $\mathcal{A}$ accepts from (E^r, i).
Now, we show the direction from right to left. We construct an infinite path $k \in \mathbb{N}^\omega$ that satisfies condition (i). Define $k_0 := i$. By assumption $\mathcal{A}$ accepts from (E^r, k_0). Since $\mathcal{A}$ accepts from (β, k_0), we have $(w, k_0) \models \beta$. Since $\mathcal{A}$ also accepts from $(r \diamond\!\!\rightarrow \mathsf{E}^r\beta, k_0)$, there is a $k_1 \in \mathbb{N}$ such that $L^*(r)$ contains $w_{k_0..k_1}$. If we iterate this argumentation, we obtain an infinite sequence of positions $k_0k_1 \dots \in \mathbb{N}^\omega$ that fulfills condition (i). By the induction hypothesis, the left hand side holds.

Consider the case $\psi = \mathsf{A}^r\beta$. Let $\mathcal{A}_r = (S, \Sigma, \eta, s_I, E)$ denote the NFA that accepts $L^*(r)$. We first show the direction from left to right by contraposition. Let $k_0 := i$. So, assume $\mathcal{A}$ rejects w from $(\mathsf{A}^r\beta, k_0)$. Then, $\mathcal{A}$ rejects from (β, i). Since, states of the from $s \,\square\!\!\rightarrow \mathsf{A}^r\beta$, where $s \in S$, are accepting, there must be a position k_1 such that $\mathcal{A}_r$ accepts $w_{k_0..k_1}$ and $\mathcal{A}$ rejects from (A^r, k_1). If we iterate this argumentation, we obtain an infinite sequence of positions $k_0k_1 \dots \in \mathbb{N}^\omega$ such that $\mathcal{A}$ rejects (β, k_i), for all $i \in \mathbb{N}$, and $L^*(r)$ contains $w_{k_i..k_{i+1}}$, for all $i \in \mathbb{N}$. We apply the induction hypothesis and obtain $(w, k_0) \models \mathsf{E}^r\neg\beta$. Thus, $(w, k_0) \not\models \mathsf{A}^r\beta$ and the left-hand side does not hold, either.
Now, we show the direction from right to left. Let $k_0 := i$. Assume $\mathcal{A}$ accepts w from configuration $(\mathsf{A}^r\beta, k_0)$. We show by contradiction that the following

holds.

For all $k \in \mathbb{N}^\omega$ with $k_0 = i$, we either have
1. there is an $n \in \mathbb{N}$ with $(w, k_n) \models \beta$ and for all $j < n$, $L^*(r)$ contains $w_{k_j..k_{j+1}}$ and $(w, k_j) \models \beta$, or
2. there is an $n \in \mathbb{N}$ with $L^*(r)$ does not contain $w_{k_n..k_{n+1}}$ and for all $j < n$, $L^*(r)$ contains $w_{k_j..k_{j+1}}$ and $(w, k_j) \models \beta$.

(ii)

Assume (*ii*) does not hold. Then, we easily see that the following fact holds.

There is an infinite path $k \in \mathbb{N}^\omega$ with $k_0 = i$ such that 1. $L^*(r)$ contains $w_{k_j..k_{j+1}}$, for all $j \in \mathbb{N}$, and 2. $(w, k_j) \models \beta$, for all $j \in \mathbb{N}$. *(iii)*

Consider the path k from (*iii*). By assumption, $\mathcal{A}$ accepts from $(\mathsf{A}^r\beta, k_0)$. By (*iii*) and the induction hypothesis, $\mathcal{A}$ must also accept from $(r \Box\!\!\rightarrow \mathsf{A}^r, k_0)$. By (*iii*), $L^*(r)$ contains $w_{k_0..k_1}$ and so, $\mathcal{A}$ must also accept from (A^r, k_1). If we iterate this argumentation, we infer that $\mathcal{A}$ only accepts from $(\mathsf{A}^r\beta, k_0)$ if the infinite path $(\mathsf{A}^r\beta, k_0)\ldots(\mathsf{A}^r\beta, k_1)\ldots(\mathsf{A}^r\beta, k_2)\ldots \in (Q \times \mathbb{N})^\omega$ is accepting. This is not the case and thus, we infer that (*ii*) holds. It is easy to see that this is equivalent to the fact that $(w, i) \models \mathsf{A}^r\beta$.

Consider the case $\psi = \alpha\, \mathsf{U}^r\, \beta$. We first show the *only if* direction. Assume $w, i \models \alpha \mathsf{U}^r \beta$. By the definition and application of the induction hypothesis, we can fix a finite sequence $k = k_0 \ldots k_n$ such that $\mathcal{A}$ accepts w from configuration (β, k_n), $k_0 = i$, and for each $j \in [n]$, $\mathcal{A}$ accepts w from configuration (α, k_j), and $w_{k_j..k_{j+1}} \in L^*(r)$. From the fact that $\mathcal{A}$ accepts w from configuration (β, k_n), we infer that $\mathcal{A}$ accepts from $(\alpha\, \mathsf{U}^r\, \beta, k_n)$. Since $w_{k_{n-1}..k_n} \in L^*(r)$, we further infer that $\mathcal{A}$ from $(r \diamond\!\!\rightarrow \alpha\, \mathsf{U}^r\, \beta, k_{n-1})$. Since $\mathcal{A}$ accepts from (α, k_{n-1}), we conclude that it accepts from $(\alpha \wedge (r \diamond\!\!\rightarrow \alpha \mathsf{U}^r \beta), k_{n-1})$. Thus, it accepts from $(\alpha\, \mathsf{U}^r\, \beta, k_{n-1})$, by the definition of the transition function. If we iterate this argumentation, we conclude that $\mathcal{A}$ accepts from $(\alpha\, \mathsf{U}^r\, \beta, k_j)$, for all $j \in [n]$. Since $k_0 = i$, we are done.

Now, we show the *if* direction. Assume $\mathcal{A}$ accepts w from configuration $(\alpha\, \mathsf{U}^r\, \beta, i)$ by the run t. For the sake of contradiction, we assume that $w, i \not\models \alpha\, \mathsf{U}^r\, \beta$. By the definition and application of the induction hypothesis, this means the following. We cannot fix a finite sequence $k = k_0 \ldots k_n$ such that $\mathcal{A}$ accepts w from configuration (β, k_n), $k_0 = i$, and for each $j \in [n]$, $\mathcal{A}$ accepts w from configuration (α, k_j), and $w_{k_j..k_{j+1}} \in L^*(r)$. In particular, it means that $\mathcal{A}$ does not accept from (β, i). Thus, $\mathcal{A}$ must accept from (α, i) and

from $(r \diamond\!\!\rightarrow (\alpha\ \mathsf{U}^r\ \beta), i)$, by the definition of the transition function. Define $k_0 := i$. So, there must be a $k_1 \geq i$ such that $w_{k_0..k_1} \in L^*(r)$ and $\mathcal{A}$ accepts from $(\alpha\ \mathsf{U}^r\ \beta, k_1)$, by the definition of the transition function. From that, we infer that $\mathcal{A}$ does not accept from (β, k_1). If we iterate the argumentation, we obtain an infinite path $(\alpha\ \mathsf{U}^r\ \beta, k_0)\ldots(\alpha\ \mathsf{U}^r\ \beta, k_1)\ldots \in (Q \times \mathbb{N})^\omega$ in the run t such that every state in this path is rejecting. This contradicts the assumption that t is accepting. The case for $\psi = \alpha\ \mathsf{S}^r\ \beta$ is similar.

Consider the case $\psi = \alpha\ \mathsf{R}^r\ \beta$. We first show the *if* direction. Assume $w, i \models \alpha\ \mathsf{R}^r\ \beta$. By the definition and application of the induction hypothesis, this means the following. We cannot fix a finite sequence $k = k_0 \ldots k_n$ such that $\mathcal{A}$ accepts w from configuration (β, k_n), $k_0 = i$, and for each $j \in [n]$, $\mathcal{A}$ accepts w from configuration (α, k_j), $k_j \leq k_{j+1}$, and $w_{k_j..k_{j+1}} \in L^*(r)$. This is equivalent to the following statement: either

(*i*) there is an infinite sequence $k = k_0 k_1 \ldots \in \mathbb{N}^\omega$ such that $k_0 = i$, and for each $j \in \mathbb{N}$, $\mathcal{A}$ accepts w from configuration (β, k_j), and $w_{k_j..k_{j+1}} \in L^*(r)$,

(*ii*) there is a finite sequence $k = k_0 \ldots k_n \in \mathbb{N}^*$ such that there is no $h \in \mathbb{N}$ with $k_n \leq h$ and $w_{k_n..h} \in L^*(r)$, $k_0 = i$, $\mathcal{A}$ accepts w from configuration (β, k_n), and for each $j \in [n]$, $\mathcal{A}$ accepts w from configuration (β, k_j), and $w_{k_j..k_{j+1}} \in L^*(r)$, or

(*iii*) there is a finite sequence $k = k_0 \ldots k_n \in \mathbb{N}^*$ such that $\mathcal{A}$ accepts w from configuration (α, k_n) and from (β, k_n), $k_0 = i$, and for each $j \in [n]$, $\mathcal{A}$ accepts w from configuration (β, k_j), and $w_{k_j..k_{j+1}} \in L^*(r)$.

Let $k_0 := i$. We consider the case (i). We construct an accepting run of $\mathcal{A}$ on w from configuration $(\alpha \mathsf{R}^r \beta, i)$. Whenever, $\mathcal{A}$ arrives at configuration $(\alpha \mathsf{R}^r \beta, k_i)$, for $j \in \mathbb{N}$, it moves to configuration (β, k_j) and $(r \diamond\!\!\rightarrow \alpha\ \mathsf{R}^r\ \beta, k_j)$ respecting the transition function. Furthermore, from $(r \diamond\!\!\rightarrow \alpha\ \mathsf{R}^r\ \beta, k_j)$, for $j \in \mathbb{N}$, it moves to configuration $(\alpha\ \mathsf{R}^r\ \beta, k_{i+1})$ simulating the transition function of the NFA for r. By assumption, $\mathcal{A}$ accepts from (β, k_j), for all $j \in \mathbb{N}$. Thus, the constructed run is accepting if the infinite path $(\psi, k_0)\ldots(\psi, k_1)\ldots$ is accepting. This is the case since every state of a configuration in this path belongs to the set of accepting states.
We consider the case (ii). In particular, $\mathcal{A}$ accepts from (β, k_n). Since there is no h with $k_n \leq h$ and $w_{k_n..h} \in L^*(r)$, $\mathcal{A}$ accepts from $(r \Box\!\!\rightarrow \psi, k_n)$, by the definition of the transition function. Therefore, $\mathcal{A}$ accepts from (ψ, k_n) respecting

the transition function. If we iterate this argumentation, we conclude that $\mathcal{A}$ accepts from (ψ, k_j), for all $j \in [n]$ and we are done.
We consider the case (iii). In particular, $\mathcal{A}$ accepts from (β, k_n) and from (α, k_n). Thus, $\mathcal{A}$ accepts from (ψ, k_n) respecting the transition function. If we iterate this argumentation, we conclude that $\mathcal{A}$ accepts from (ψ, k_j), for all $j \in [n]$ and we are done. The case for $\psi = \alpha\, \mathsf{T}^r\, \beta$ is similar.

Eventually 1-Wayness We show that the automaton $\mathcal{A}$ is eventually 1-way. For a RE r and a state s in $\mathcal{A}_r$ or $\mathcal{A}'_r$, we define $\mathsf{Sub}'(s \diamond\!\!\rightarrow \alpha \mathsf{U}^r \beta) := \mathsf{Sub}(\alpha \mathsf{U}^r \beta)$. Similarly, we define Sub' for the all other states $q \in Q^+ \cup Q^-$. Furthermore, let $\hat{Q} := \{q \in \mathsf{Sub}(\varphi) \mid q \text{ is of the form } \alpha\, \mathsf{S}^r\, \beta \text{ or } \alpha\, \mathsf{T}^r\, \beta\}$.

For defining the partitioning of the state set Q, we need the following function that assigns weights to states.

$$\mathsf{weight}(q) := \begin{cases} 2|\mathsf{Sub}(q)| & \text{if } q \in \mathsf{Sub}(\varphi) \setminus \hat{Q}, \\ 2|\mathsf{Sub}(q)| + 1 & \text{if } q \in \mathsf{Sub}(\varphi) \cap \hat{Q}, \\ 2|\mathsf{Sub}(\alpha\, \mathsf{U}^r\, \beta)| & \text{if } q \in Q^+ \\ 2|\mathsf{Sub}(\alpha\, \mathsf{U}^r\, \beta)| + 1 & \text{if } q \in Q^-. \end{cases}$$

Let $n := 2|Q| + 1$. Let $(Q_i)_{i \leq n}$ be a partitioning of Q, where for $i \leq [n]$, we define $Q_i := \{q \mid \mathsf{weight}(q) = i\}$.

Let $p, q \in Q$, $D \subseteq \mathbb{D}$, $d \in D$, and $a \in \Sigma$ such that $(q, d) \in \delta_D(p, a)$. It suffices to show the following claim: if ($\mathsf{weight}(p)$ is even and $d \leq 0$) or ($\mathsf{weight}(p)$ is odd and $d \geq 0$) then $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in \mathsf{Sub}(\varphi) \setminus \hat{Q}$. We have that $\mathsf{weight}(p)$ is even. Assume $d \leq 0$. By the definition of the transition function, $d \neq -1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in \mathsf{Sub}(\varphi) \cap \hat{Q}$. We have that $\mathsf{weight}(p)$ is odd. Assume $d \geq 0$. By the definition of the transition function, $d \neq 1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q^+$. We have that $\mathsf{weight}(p)$ is even. Assume $d \leq 0$. By the definition of the transition function, $d \neq -1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case $p \in Q^-$. We have that $\mathsf{weight}(p)$ is odd. Assume $d \geq 0$. By the definition of the transition function, $d \neq 1$. It follows that $q \in \mathsf{Sub}(p)$ and hence $\mathsf{weight}(q) < \mathsf{weight}(p)$. ∎

Using our alternation-elimination scheme and the complementation constructions from Section 4.1, we obtain the following corollary.

Corollary 4.32 *For every PRLTL formula in positive normal form of size n, there is a language-equivalent 1NBA of size $2^{\mathcal{O}(n \log n)}$.* □

We remark that we can translate PRLTL formulas without any E or A operator, say PRLTL$^-$ formulas, into a 2APA with just two parities, or equivalently, into a 2ABA. So, we obtain the following corollary.

Corollary 4.33 *For every PRLTL$^-$ formula in positive normal form of size n and having m propositions, there is a language-equivalent 1NBA of size $\mathcal{O}(2^m \cdot 3^n)$. If the formula has no past operators then the size is in $\mathcal{O}(3^n)$.* □

Chapter 5

Translating Logics over Nested Words to Automata

In this chapter, we present translations from various classes of alternating automata over nested words to nested-word automata. We obtain these translations from our alternation-elimination scheme by providing complementation constructions for the corresponding classes of existential automata over nested words. We use these alternation-elimination constructions in turn to translate various temporal logics over nested words to nested-word automata.

We proceed as follows. In Section 5.1, we present complementation constructions for different classes of existential automata over nested words. In Section 5.2, we present several logics over nested words and show how to translate these logics into nested-word automata using instances of our alternation-elimination scheme.

5.1 Complementation Constructions

In this section, we present several novel constructions for complementing the languages of existential automata over nested words. The constructions translate various classes of existential automata into nested-word automata. Table 5.1 depicts the blow-ups of these constructions, where n is the size of the existential automaton and k its index. Furthermore, it references the theorems, where these constructions are given. For instance, we present a complementation construction that translates an eventually 1-way very weak existential co-Büchi automaton over nested words into a nested-word automaton of size $\mathcal{O}(2^{2n}n)$. This construction is given in Theorem 5.5. In the following sections, we write $\mathbb{D} := \{-2, -1, 0, 1, 2\}$ for the set of directions in a nested word.

	V2ECA	2ECA	2EPA
1-way	$\mathcal{O}(2^n n)$ Theorem 5.6	$\mathcal{O}(3^n)$ Theorem 5.3	$2^{\mathcal{O}(nk \log n)}$ Theorem 5.15
eventually 1-way	$\mathcal{O}(2^{2n} n)$ Theorem 5.5	$\mathcal{O}(2^n 3^n)$ Theorem 5.2	$2^{\mathcal{O}(nk \log n)}$ Theorem 5.15
2-way	$2^{\mathcal{O}(n^2)}$ Theorem 5.16	$2^{\mathcal{O}(n^2)}$ Theorem 5.16	$2^{\mathcal{O}((nk)^2)}$ Theorem 5.16

Table 5.1: Sizes of NWAs obtained by the complementation constructions.

5.1.1 Complementing co-Büchi Automata

In this section, we translate an eventually 1-way existential co-Büchi automaton $\mathcal{A}$ into a nested-word automaton $\mathcal{B}$ that accepts the complement of $L^{\mathsf{nw}}(\mathcal{A})$. We start this construction with a characterization of nested words that are not accepted by a given eventually 1-way 2ECA.

Lemma 5.1 *Let $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be an eventually 1-way 2ECA and $(w, \rightsquigarrow)$ a nested word in $\hat{\Sigma}^\omega$. We have $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$ if and only if there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ such that the following conditions hold.*

(1) $q_I \in R_0$.

(2) *For all $d \in \mathbb{D}$ and $(i, j) \in \rightsquigarrow_d$, $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_j$.*

(3) *There is no $i \in \mathbb{N}$ and $q \in R_i$ such that $\emptyset \models \delta_{\mathbb{D}_i}(q, w_i)$.*

(4) *For all $d \in \{1, 2\}$ and $(i, j) \in \rightsquigarrow_d$, $\delta^d_{\mathbb{D}_i}(S_i, w_i) \setminus F \subseteq S_j$.*

(5) *For infinitely many sync positions $k \in \mathbb{N}$, $S_k = \emptyset$ and $S_{k+1} = R_{k+1} \setminus F$.* □

Before presenting the proof, we give an intuition for the constraints of the lemma. The conditions (1) and (2) ensure that the word R represents all runs $(q_0, h_0)(q_1, h_1) \dots$ of the existential automaton $\mathcal{A}$ on the given input $(w, \rightsquigarrow)$, i.e., R_{h_i} contains q_i, for all $i \in \mathbb{N}$. The conditions (3) to (5) on the words R and S ensure that all the runs are rejecting. Recall that a nested word is rejected if it is not accepted by a finite run and every infinite run visits a state in F infinitely often. Condition (3) ensures that there is no finite accepting run. All the infinite runs are rejecting if the word R can be split into infinitely

many nonempty segments such that each run of the existential automaton that starts at the beginning of a segment will visit a state in F before reaching the end of the segment. The conditions (4) and (5) on the word S ensure the existence of such a splitting. In particular, the ks from condition (5) mark the end positions of the segments in the splitting.

Proof We first prove the *only if* direction. Assume $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$, i.e., every run of $\mathcal{A}$ on $(w, \rightsquigarrow)$ visits a state in F infinitely often.

For the constructions in this proof, we use the following definitions. A word $(q_0, h_0) \dots (q_n, h_n) \in (Q \times \mathbb{N})^*$ is a *run segment* if for all $i \in [n]$, there is a $d \in \mathbb{D}$ such that $(h_i, h_{i+1}) \in \rightsquigarrow_d$ and $(q_{i+1}, d) \in \delta_{\mathbb{D}_i}(q_i, w_i)$. The run segment is *initial* if $(q_0, h_0) = (q_I, 0)$. The run segment is *F-avoiding* if $q_i \notin F$, for all $i \leq n$.

We construct a word $R \in (2^Q)^\omega$ that satisfies the conditions (1) and (2). For $i \in \mathbb{N}$, we define R_i as

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

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

Condition (3) is also fulfilled since otherwise there is an initial run segment $(q_0, h_0) \dots (q_n, h_n)$, where $\mathtt{tt}$ occurs in $\delta_{\mathbb{D}_i}(q_n, w_{h_n})$. However, this means that $\mathcal{A}$ accepts $(w, \rightsquigarrow)$, which contradicts the assumption $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$.

Now, we define a word $S \in (2^{Q \setminus F})^\omega$ that satisfies the conditions (4)–(5). Let $b \in \{0,1\}^\omega$ be an infinite word such the bit $b_k = 0$ if and only if $k \in \mathbb{N}$ is a sync position. In the following, we define S inductively. For convenience, let $S_{-1} := \emptyset$ and $b_{-1} = 0$. Let $m \in \mathbb{N} \cup \{-1\}$ such that $S_m = \emptyset$ and $b_m = 0$. For every m, we define the word $T^m \in (Q \times \mathbb{N})^\omega$ as the set of F-avoiding run segments that start in $R_{m+1} \setminus F$. For brevity, we just write T instead of T^m. Formally, for $i \leq m$, we define $T_i := \emptyset$ and for $i > m$, we define

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

Next, we show that there is a position $n > m$ such that $T_n = \emptyset$ and $b_n = 0$. Intuitively, every run segment that starts in $R_{m+1} \setminus F$ and visits positions after n, (a) visits position n because of condition $b_n = 0$, and (b) visits an F-state before it reaches position n because of condition $T_n = \emptyset$. Let n be the smallest

position after m such that $T_n = \emptyset$ and $b_n = 0$. We argue that the position n exists. For the sake of contradiction, assume that this n does not exist.

We first show that the b sequence has value 0 at infinitely many positions. Formally, for every position $i > m$, there is a position $j \geq i$ such that $b_j = 0$. Consider a position $i > m$. If $b_i = 0$, we are done. Otherwise, let h be the least matched call position such that its matching return position k is greater than i. Since edges in $\rightsquigarrow_2$ do not cross, we have $b_k = 0$.

Now, we consider the following graph. The vertices are elements of $\{(q, i) \in Q \times \mathbb{N} \mid i \in \mathbb{N}, q \in T_i, \text{ and } b_i = 0\}$. This set is infinite since by assumption, there are infinitely many positions $i \in \mathbb{N}$, where $b_i = 0$ and $T_i \neq \emptyset$. There is an edge from (q, h) to (q', h') if the automaton $\mathcal{A}$ can move from configuration (q, h) to (q', h'), i.e., there is a direction $d \in \mathbb{D}$ such that $(h, h') \in \rightsquigarrow_d$ and $(q', d) \in \delta_{\mathbb{D}_h}(q, w_h)$. Note that each node has only finitely many successors. Furthermore, every node is reachable by some node in the finite set $\{(q, m+1) \mid q \in T_{m+1}\}$. By König's Lemma, the graph contains an infinite path. Note that for each tuple (q, h) in that path, we have $q \notin F$. Thus, there is an accepting infinite run of $\mathcal{A}$ on $(w, \rightsquigarrow)$. This contradicts the assumption that $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$.

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

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

Now, we prove the *if* direction. Assume that there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that satisfy conditions (1)–(5).

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

Suppose that $r := (q_0, h_0)(q_1, h_1) \dots \in (Q \times \mathbb{N})^\omega$ is infinite. Note that the conditions (1) and (2) ensure that $q_i \in R_{h_i}$, for all $i \in \mathbb{N}$. Since $\mathcal{A}$ is eventually 1-way and r is accepting, there is an index $k \in \mathbb{N}$ such that for all $i \geq k$, we have $q_i \notin F$ and $(h_i, h_{i+1}) \in \rightsquigarrow_d$ with $d \in \{1, 2\}$. By condition (5), there is a breakpoint at position $m > h_k$ with $S_m = \emptyset$ and $S_{m+1} = R_{m+1} \setminus F$. Moreover, there is no $(i, j) \in \rightsquigarrow_2$ such that $i \leq m$ and $j > m$. It follows that r must visit

this breakpoint, i.e., there is an index l such that $h_l = m$ and thus, $q_l \in R_m$. By condition (5), the position m is not a matched call. Hence, $h_{l+1} = h_l + 1$. It follows that $q_{l+1} \in S_{m+1}$.

Now, we show that there cannot be a further breakpoint after position m, which contradicts condition (5). Assume there is a breakpoint at position $n > m$ with $S_n = \emptyset$ and $S_{n+1} = R_{n+1} \setminus F$. By condition (4), the fact that $q_{l+1} \in S_{m+1}$, and the fact that there is no $(i, j) \in \rightsquigarrow_2$ such that $i \leq n$ and $j > n$, the run r must visit S_n, i.e., there is an index l' such that $h_{l'} = n$ and $q_{l'} \in S_n$. However, this contradicts the fact that S_n is empty. ∎

Theorem 5.2 *For every eventually 1-way 2ECA $\mathcal{A}$ with n states, there is an NWA $\mathcal{B}$ with $\mathcal{O}(2^n 3^n)$ states and $\mathcal{O}(2^n 3^n)$ stack symbols that accepts the complement of $L^{\mathsf{nw}}(\mathcal{A})$.* □

Before presenting the proof, we give an overview on the construction of the automaton $\mathcal{B}$. Let $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be an eventually 1-way 2ECA. The automaton $\mathcal{B}$ guesses the words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ from Lemma 5.1 and locally checks that the conditions (1)–(4) hold. The construction is similar to the 2-way subset construction in Theorem 2.6. Besides, the automaton $\mathcal{B}$ additionally uses its stack to check the conditions for non-local moves of $\mathcal{A}$.

Let us elaborate on the construction for checking condition (5). The construction extends the 2-way breakpoint construction presented in Theorem 4.2. The automaton $\mathcal{B}$ guesses a sequence $b \in \{0, 1\}^\omega$ and checks that for every $k \in \mathbb{N}$, the bit b_k is 0 if and only if k is a sync position. For the check, the automaton $\mathcal{B}$ uses a bit in its states and a bit in its stack symbols. Initially, the bit in the state is assigned to 0. For every call position, $\mathcal{B}$ guesses whether the call is matched or not. In case the call is not matched, $\mathcal{B}$ pushes the special symbol pending on the stack. The automaton is not allowed to pop the symbol pending later on. In case the call is matched, the bit in the current state is stored on the stack and then assigned to 1. If the matched return is reached the bit in the state is restored from the stack. By using the bit sequence b and the breakpoint construction, $\mathcal{B}$ ensures that condition (5) holds.

Proof Let $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be an eventually 1-way 2ECA. The proof is structured as follows. We first formally define the NWA $\mathcal{B}$. Then, we prove the correctness of the construction.

Construction We define the NWA $\mathcal{B}$ as $(P, O, \Sigma, \eta, p_I, G)$, where P, O, and G are defined as follows.

- $P := O := (2^Q \times 2^{Q \setminus F} \times 2^Q \times \{0,1\}) \cup \{p_I\}$. A state can be a tuple or the initial state p_I. Consider the case where the automaton is in state (X, Y, X', z) and processes the ith input letter of $(w, \rightsquigarrow)$. Then, X and Y correspond to the guessed sets R_i and S_i of the words R and S, respectively. The component X' corresponds to the guessed sets R_{i+1} of the word R. The component z corresponds to the bit b_i of the guessed word b.

- $G := 2^Q \times \{\emptyset\} \times 2^Q \times \{0\}$. That is, at infinitely many positions $i \in \mathbb{N}$, $b_i = 0$ and $S_i = \emptyset$.

For brevity, we use pattern matching in the definition of the transition function. For instance, we write $\eta((R_{-1}, S_{-1}, R_0, b), a) \ni (R_0, S_0, R_1, 1)$ meaning $\eta((R_{-1}, S_{-1}, R_0, b), a) \ni (R'_0, S'_0, R_1, b')$, where the following three conditions hold: $R'_0 = R_0$ and $b' = 1$. For clarity, we write $\mathsf{pending}$ to denote the state p_I.

First, we define the transitions from the initial state p_I. Let $a \in \hat{\Sigma}$.

- For an internal position, we have $\eta_i(p_I, a) \ni (R_0, R_0 \setminus F, R_1, 0)$ iff the following conditions hold. Let $D := \{0, 1\}$.
  1. We have $q_I \in R_0$.
  2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.

- For a call position, we have $\eta_c(p_I, a) \ni ((R_0, R_0 \setminus F, R_1, 1), (R_0, R_0 \setminus F, R_2, 0))$ iff the following conditions hold. Let $D := \{0, 1, 2\}$.
  1. We have $q_I \in R_0$.
  2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.

  Furthermore, $\eta_c(p_I, a) \ni ((x, 0), \mathsf{pending})$ iff $\eta_i(p_I, a) \ni (x, b)$, for some $b \in \{0, 1\}$ and $x \in (2^Q \times 2^{Q \setminus F})^2$.

- For a return position, we have $\eta_r(p_I, o, a)$ contains $(R_0, S_0, R_1, 0)$ iff we have $\eta_i(p_I, a) \ni (R_0, S_0, R_1, 0)$, where $o \in O \cup \{\bot\}$ is some stack symbol.

Now, we define the transitions from states in $P \setminus \{p_I\}$. Let $a \in \hat{\Sigma}$.

- For an internal position, the transition function $\eta_i((R_{-1}, S_{-1}, R_0, b), a)$ contains (R_0, S_0, R_1, S_1, b) iff the following conditions hold. Let $D := \{-1, 0, 1\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0, 1\}$, we have $\delta_D^d(S_{-1}, a) \setminus F \subseteq S_{-1+d}$.
  4. If $S_{-1} = \emptyset$ and $b = 0$ then $S_0 = R_0 \setminus F$.

- For a call position, the transition function $\eta_c((R_{-1}, S_{-1}, R_0, b), a)$ contains $((R_0, S_0, R_1, 1), (R_0, S_0, R_2, b))$ iff the following conditions hold. Let $D := \{-1, 0, 1, 2\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0, 1, 2\}$, we have $\delta_D^d(S_{-1}, a) \setminus F \subseteq S_{-1+d}$.
  4. If $S_{-1} = \emptyset$ and $b = 0$ then $S_0 = R_0 \setminus F$.

  Furthermore, $\eta_c(p, a) \ni ((x, 0), \mathsf{pending})$ iff $\eta_i(p, a) \ni (x, b)$, for some $b \in \{0, 1\}$.

- For a return position, we have

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

  iff the following conditions hold. Let D be the set $\{-2, -1, 0, 1\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0, 1\}$, we have $\delta_D^d(S_{-1}, a) \setminus F \subseteq S_{-1+d}$.
  4. If $S_{-1} = \emptyset$ and $b = 0$ then $S_0 = R_0 \setminus F$.

  Furthermore, we have $\eta_r(p, \bot, a) = \eta_i(p, a)$, for $p \in P$. Note that there is no transition if the symbol $\mathsf{pending}$ is on the stack.

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

First, we prove $L^{\mathsf{nw}}(\mathcal{B}) \subseteq \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$. Assume that $\mathcal{B}$ accepts $(w, \rightsquigarrow)$ by the run $r := (p_0, o_0)(p_1, o_1) \ldots \in (P \times O)^\omega$. We show that there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that fulfill the conditions (1)–(5) of Lemma 5.1.

We construct the words R and S. By the definition of the transition function, $p_0 = p_I$ and for all $i > 0$, each p_i is a tuple of the from $(A_i, B_i, C_i, b_i) \in (2^Q \times 2^{Q \setminus F} \times 2^Q \times \{0, 1\})$. Define $R_i := A_{i+1}$ and $S_i := B_{i+1}$, for all $i \in \mathbb{N}$.

We show that the words R and S satisfy the conditions (1)–(5). By definition, conditions (1) and (3) are fulfilled. We show that condition (2) is fulfilled. Let $d \in \mathbb{D}$ with $(i,j) \in \rightsquigarrow_d$. We make a case distinction.

(a) Consider the case, where $i = 0$. Let $D := \mathbb{D}_i$. If i is an internal, a return or a pending call position then we have $\eta_i(p_I, w_0) \ni p_1$. We have $\delta_D^0(A_1, w_0) \subseteq A_1$ and hence, $\delta_D^0(R_0, w_0) \subseteq R_0$. We also have $\delta_D^1(A_1, w_0) \subseteq A_2$ since $C_1 = A_2$. So, $\delta_D^1(R_0, w_0) \subseteq R_1$. If i is a matched call position then we have $\eta_c(p_I, w_0) \ni (p_1, o_1)$. By the same arguments as above, we infer that $\delta_D^d(R_0, w_0) \subseteq R_d$, for $d \in \{0,1\}$. We have $\delta_D^0(A_1, w_0) \subseteq A_1$ and hence, $\delta_D^0(R_0, w_0) \subseteq R_0$. Let (A_1', B_1', C_1', b_1') denote the tuple o_1. We have $\delta_D^2(A_1, w_0) \subseteq A_{j+1}$ since j is the matching return position of i and $C_1' = A_{j+1}$. So, $\delta_D^2(R_0, w_0) \subseteq R_j$.

(b) The proof for the case, where $i > 0$ is similar.

The proof that condition (4) is fulfilled is analogous.

We show that condition (5) is fulfilled. Since G occurs infinitely often, there are infinitely many $i > 0$ such that $B_i = \emptyset$ and $b_i = 0$. Note that $b_{k+1} = 0$ if and only if for all $(i,j) \in \rightsquigarrow_2$, if $i \leq k$ then $j \leq k$. Since r is a run, $B_i = \emptyset$ and $b_i = 0$ implies that $B_{i+1} = R_{i+1} \setminus F$, for every $i > 0$. It follows that (5) holds.

Now, we prove $L^{\mathsf{nw}}(\mathcal{B}) \supseteq \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$. Assume $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$. Then, there are words $R \in (2^Q)^\omega$ and $S \in (2^{Q \setminus F})^\omega$ that fulfill the conditions (1)–(5) of Lemma 5.1. For convenience, we define the word $b \in \{0,1\}^\omega$, where $b_k = 0$ if and only if $k \in \mathbb{N}$ is a sync position. Without lost of generality, we assume that there is no position $i \in \mathbb{N}$ such that $S_i = S_{i+1} = \emptyset$ and $b_i = 0$. In other words, every S_i is filled up as soon as every non-pending call with position $j \leq i$ is matched before or at position i. For more details, see the construction of S in the proof of Lemma 5.1.

We construct an accepting run $r := (p_0, o_0)(p_1, o_1) \ldots \in (P \times O)^\omega$ of $\mathcal{B}$ on $(w, \rightsquigarrow)$. Let $p_0 := p_I$ and for all $i > 0$, we define $p_i := (R_{i-1}, S_{i-1}, R_i, b_{i-1})$. Next, we define the symbols that are pushed on the stack at call positions. For every pending call position i, we define $o_{i+1} := \mathsf{pending}$. For every $(i,j) \in \rightsquigarrow_2$, we define $o_{i+1} := (R_i, S_i, R_j, b_i)$.

In the following, we show that r is an accepting run. We show that for all $i \in \mathbb{N}$, there is a transition in $\mathcal{B}$ from r_i to r_{i+1} when reading the letter w_i. First, there is a transition from r_0 to r_1 when reading the letter w_0. Consider the case, where 0 is an internal or return position. By definition of r_0, the automaton is in state p_I, reads w_0 and goes to state $(R_0, S_0, R_1, 0)$. By definition of R

and S, all conditions for this transition are fulfilled. Consider the case, where $(0, j) \in \rightsquigarrow_2$, for some $j \in \mathbb{N}$. By definition of r_0, the automaton is in state p_I, reads w_0 and goes to state $p_1 = (R_0, S_0, R_1, 1)$ and pushes $o_1 = (R_0, S_0, R_j, 0)$ on the stack. By definition of R and S, all conditions for this transition are fulfilled. Finally, consider the case, where 0 is a pending call position. By definition of r_0, the automaton is in state p_I, reads w_0 and goes to state $(R_0, S_0, R_1, 1)$ and pushes pending on the stack. By definition of R and S, all conditions for this transition are fulfilled. The proof that there is a transition from r_i to r_{i+1} when reading the letter w_i, for any $i > 0$, is analogous.

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

Now, consider the case where the existential co-Büchi automaton whose language has to be complemented is 1-way. In this case, we can simplify condition (2) of Lemma 5.1 since the automaton does not move its read-only head backwards. The following requirement must hold.

(2′) For all $(i, j) \in \rightsquigarrow_d$ with $d \geq 0$, we have $\delta^d_{\mathbb{D}_i}(R_i, w_i) \subseteq R_j$.

From this observation, we directly obtain the following theorem as a special case of Theorem 5.2.

Theorem 5.3 *For every 1ECA $\mathcal{A}$ with n states, there is an NWA $\mathcal{B}$ with $\mathcal{O}(3^n)$ states and $\mathcal{O}(3^n)$ stack symbols that accepts the complement of $L^{\mathsf{nw}}(\mathcal{A})$.*□

Proof Let $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be an 1ECA. In a nutshell, the construction from Theorem 5.2 consists of three parts: a 2-way subset construction, a construction for guessing the sync positions, and a construction to check that all runs of $\mathcal{A}$ fulfill the co-Büchi acceptance condition. The construction of the NWA $\mathcal{B}$ follows the same line except for one part. The 2-way subset construction is replaced by the standard 1-way subset construction.

We formally define the NWA $\mathcal{A}$ as $(P, O, \Sigma, \eta, p_I, G)$, where P, O, p_I, and G are defined as follows.

- $P := O := (2^Q \times 2^Q \times \{0, 1\}) \cup \{p_I\}$ is the set of states and stack symbols, where p_I denotes a pending state.
- $p_I := (\{q_I\}, \emptyset, 0)$ is the initial state.
- $G := 2^Q \times \emptyset \times \{0\}$ is the set of accepting states.

Next, we define the transition function. Let $R_0, R_1 \in 2^Q$, $S_0, S_1 \in 2^Q \setminus F$, $b \in \{0,1\}$, and $a \in \hat{\Sigma}$.

- For an internal position, we have $\eta_i(R_0, S_0, b), a) \ni (R_1, S_1, b)$ iff the following conditions hold. Let $D := \{0,1\}$
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0,1\}$, we have $\delta_D^d(S_0, a) \setminus F \subseteq S_d$.
  4. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

- For a call position, we have $\eta_c((R_0, S_0, b), a) \ni ((R_1, S_1, 1), (R_2, S_2, b))$ iff the following conditions hold. Let $D := \{0,1,2\}$
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0,1,2\}$, we have $\delta_D^d(S_0, a) \setminus F \subseteq S_d$.
  4. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

  Furthermore, $\eta_c(p, a) \ni ((x, 0), \mathsf{pending})$ iff $\eta_i(p, a) \ni (x, b)$, for some $b \in \{0,1\}$.

- For a return position, we have $\eta_r((R_0, S_0, 1), (R_0, S_0, b), a) \ni (R_1, S_1, b)$ iff the following conditions hold. Let $D := \{0,1\}$
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{0,1\}$, we have $\delta_D^d(S_0, a) \setminus F \subseteq S_d$.
  4. If $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F$.

  Furthermore, we have $\eta_r(p, \bot, a) = \eta_i(p, a)$, for $p \in P$. Note that there is no transition if the symbol pending is on the stack.

The proof for $L^{\mathsf{nw}}(\mathcal{A}) = \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{B})$ is along the lines as the proof of Theorem 5.2. We omit it. ∎

5.1.2 Complementing Very-Weak Automata

In the following, we optimize our complementation construction for restricted classes of eventually 1-way 2ECAs. When $\mathcal{A}$ is very weak, we can characterize the language of nested words that is not accepted by $\mathcal{A}$ by similar conditions

as those given in Lemma 5.1. The existence of the S sequence together with condition (4) is not required anymore and condition (5) is replaced by the requirement that no run of the existential automaton gets trapped in some non-accepting state. We make this observation explicit, in the following lemma.

Lemma 5.4 *Let $\mathcal{A} := (Q, \hat{\Sigma}, \delta, q_I, F)$ be an eventually 1-way very weak 2ECA and $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$. We have $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$ if and only if there is a word $R \in (2^Q)^\omega$ such that the conditions (1)–(3) of Lemma 5.1 and the following condition hold.*

(5′) *There is no state $q \in Q \setminus F$ and no sequence of positions $h \in \mathbb{N}^\omega$ such that $q \in R_{h_0}$ and for all $i \in \mathbb{N}$, there is a direction $d > 0$ such that $(h_i, h_{i+1}) \in \rightsquigarrow_d$ and $q \in \delta^d_{\mathbb{D}_{h_i}}(q, w_{h_i})$.* □

Proof We first prove the *only if* direction. Assume $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$. As shown in Lemma 5.1, we construct the sequence $R \in (2^Q)^\omega$ that satisfies the conditions (1)–(3).

It remains to show that R fulfills condition (5′). Let $q \in Q \setminus F$ and $h \in \mathbb{N}^\omega$ such that $q \in R_{h_0}$ and for all $i \in \mathbb{N}$, there is a direction $d \in \{1, 2\}$ such that $(h_i, h_{i+1}) \in \rightsquigarrow_d$ and $q \in \delta^d_{\mathbb{D}_{h_i}}(q, w_{h_i})$. Since $q \in R_{h_0}$, there is an initial run segment $(q_0, k_0) \dots (q_n, k_n)$ with $(q_n, k_n) = (q, h_0)$. It follows that $(q_0, k_0) \dots (q_n, k_n)(q, h_1)(q, h_2) \dots$ is an accepting run of $\mathcal{A}$ on $(w, \rightsquigarrow)$. However, this contradicts the assumption $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$.

In the remainder of the proof, we show the *if* direction. Assume there is an $R \in (2^Q)^\omega$ such that the conditions (1)–(3) and (5′) hold. We show that any run of $\mathcal{A}$ on $(w, \rightsquigarrow)$ is rejecting. As shown in Lemma 5.1, we infer from the conditions (1)–(3) that every finite run must be rejecting.

Now, consider an infinite run $r := (q_0, h_0)(q_1, h_1) \dots \in (Q \times \mathbb{N})^\omega$. For the sake of contradiction, assume r is accepting. First, note that the conditions (1) and (2) ensure that $q_i \in R_{h_i}$, for all $i \in \mathbb{N}$. Second, note that there is an index $m \in \mathbb{N}$ and a state $q \in Q \setminus F$ such that $q_i = q$, for all $i \geq m$ since $\mathcal{A}$ is very weak. Third, note that there is an index $n \geq m$ such that for all $i \geq n$, we have $h_i < h_{i+1}$ since $\mathcal{A}$ is eventually 1-way. However, the existence of the infinite sequence $(q_n, h_n)(q_{n+1}, h_{n+1}) \dots$ contradicts condition (5′). ■

Theorem 5.5 *For every eventually 1-way V2ECA $\mathcal{A}$ with n states, there is an NWA $\mathcal{B}$ with $\mathcal{O}(2^{2n}n)$ states and $\mathcal{O}(2^{2n}n)$ stack symbols that accepts the complement of $L^{\mathsf{nw}}(\mathcal{A})$.* □

Proof Let $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ be an eventually 1-way V2ECA. We define $E := (Q \setminus F) \cup \{*\}$.

The proof is similar to the proof for Theorem 5.2. We construct an NWA $\mathcal{B}$ that guesses the word $R \in (2^Q)^\omega$ from Lemma 5.4 and locally checks the conditions (1)–(3) and condition (5′) with its acceptance condition.

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

- $P := S := (2^Q \times 2^Q \times E \times \{0,1\}) \cup \{p_I\}$. The component in E is called *focus*. It is used to find a state in $Q \setminus F$ that can get trapped in a self-loop.
- $G := 2^Q \times 2^Q \times \{*\} \times \{0\}$.

As in the proof of Theorem 5.2, we use pattern matching in the definition of the transition function. Let $<$ be a total ordering on the set E, where $*$ is the greatest element. Furthermore, let $\mathsf{next} : E \to E$ be a function that maps the greatest element $*$ to the smallest one and each of the other elements to the next greater one. We also use the synonym $\mathsf{pending} := p_I$.

First, we define the transitions from the initial state p_I. Let $a \in \hat{\Sigma}$.

- For an internal position, we have $\eta_i(p_I, a) \ni (R_0, R_1, *, 0)$ iff the following conditions hold. Let $D := \{0, 1\}$.
  1. We have $q_I \in R_0$.
  2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
- For a call position, we have $\eta_c(p_I, a) \ni ((R_0, R_1, *, 1), (R_0, R_2, *, 0))$, iff the following conditions hold. Let $D := \{0, 1, 2\}$.
  1. We have $q_I \in R_0$.
  2. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  3. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.

  Further, $\eta_c(p_I, a) \ni ((R_0, R_1, *, 0), \mathsf{pending})$ iff $\eta_i(p_I, a) \ni (R_0, R_1, *, 0)$.

- For a return position, we have $\eta_r(p_I, s, a) \ni (R_0, R_1, 0)$ iff $\eta_i(p_I, a) \ni (R_0, R_1, 0)$, where $s \in S \cup \{\bot\}$ is some stack symbol.

Second, we define the transitions from states in $P \setminus \{p_I\}$. Let $a \in \hat{\Sigma}$.

- For an internal position, we have $\eta_i((R_{-1}, R_0, s_0, b), a) \ni (R_0, R_1, s_1, b)$ iff the following conditions hold. Let $D := \{-1, 0, 1\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. $s_1 = \begin{cases} s_0 & \text{if } s_0 \in R_0 \cap \delta_D^1(s_0, a) \text{ or } (s_0 = * \text{ and } b = 1), \\ \mathsf{next}(s_0) & \text{otherwise.} \end{cases}$

- For a call position, the transition function $\eta_c((R_{-1}, R_0, s_0, b), a)$ contains $((R_0, R_1, s_1, 1), (R_0, R_2, s_2, b))$ iff the following conditions hold. Let $D := \{-1, 0, 1, 2\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For all $d \in \{1, 2\}$, we have

$$s_d = \begin{cases} s_0 & \text{if } s_0 \in R_0 \cap \delta_D^d(s_0, a) \text{ or } (s_0 = * \text{ and } b = 1), \\ \mathsf{next}(s_0) & \text{otherwise.} \end{cases}$$

  Furthermore, $\eta_c(p, a) \ni ((x, 0), \mathsf{pending})$ iff $\eta_i(p, a) \ni (x, b)$, for some $b \in \{0, 1\}$.

- For a return position, we have $\eta_r((R_{-1}, R_0, s_0, 1), (R_{-2}, R_0, s_0', b), a) \ni (R_0, R_1, s_1, b)$ iff the following conditions hold. Let $D := \{-2, -1, 0, 1\}$.
  1. For all $d \in D$, we have $\delta_D^d(R_0, a) \subseteq R_d$.
  2. For all $p \in R_0$, we have $\emptyset \not\models \delta_D(p, a)$.
  3. For $s := \mathsf{min}(s_0, s_0')$, we have

$$s_1 = \begin{cases} s & \text{if } s \in R_0 \cap \delta_D^1(s, a) \text{ or } (s = * \text{ and } b = 1), \\ \mathsf{next}(s) & \text{otherwise.} \end{cases}$$

  For a pending return position, we have $\eta_r(p, \bot, a) := \eta_i(p, a)$, for $p \in P$. Note that there is no transition if the symbol $\mathsf{pending}$ is on the stack.

Correctness The correctness proof of the construction is along the same lines as the correctness proof given in Theorem 5.2. We first prove that $L^{\mathsf{nw}}(\mathcal{B}) \subseteq \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$. Let $(w, \rightsquigarrow) \in L^{\mathsf{nw}}(\mathcal{B})$. Let $r := (p_0, s_0)(p_1, s_1) \ldots \in (P \times S)^\omega$ be an accepting run of $\mathcal{B}$ of $(w, \rightsquigarrow)$.

It suffices to construct a word $R \in (2^Q)^\omega$ that fulfills the conditions (1)–(3) and (5′) of Lemma 5.4. By definition of the transition function, $p_0 = p_I$ and for all $i > 0$, each p_i is a tuple of the form $(A_i, B_i, s_i, b_i) \in P$. Define $R_i := A_{i+1}$, for all $i \in \mathbb{N}$. By the same arguments as in the proof of Theorem 5.2, it follows that R fulfills the conditions (1)–(3).

It remains to show that R satisfies condition (5′). For the sake of contradiction, assume there is a state $q \in Q \setminus F$ and a sequence of positions $h \in \mathbb{N}^\omega$ such that $q \in R_{h_0}$ and for all $i \in \mathbb{N}$, there is a direction $d > 0$ such that $(h_i, h_{i+1}) \in \rightsquigarrow_d$ and $q \in \delta^d_{\mathbb{D}_{h_i}}(q, w_{h_i})$.

We show that the automaton $\mathcal{B}$ will eventually recognize this sequence of repeating qs with its third component of its states. We make this intuition formal. Let $m > h_0$ be a sync position with $s_m = *$. Let $n > m$ be the first sync position after m with $s_n = *$. Note that $b_m = b_n = 0$. The positions m and n exist since r is an accepting run. By the definition of the transition function, it follows that for any position $i \in \mathbb{N}$ with $m < i < n$ and $s_i < q$, there is a position $j \in \mathbb{N}$ with $i < j < n$ such that $s_j = q$. Note that s_{m+1} is the smallest element in E, by definition of the transition function. Therefore, there is a $k \in \mathbb{N}$ with $m < k < n$ such that $s_k = q$. Let k be the largest position such that $m < k < n$ and $s_k = q$.

We show that such a k does not exist. We make a case distinction.

(a) There is some $i \in \mathbb{N}$ such that $h_i = k$. Since $q_i = q$, we have $q \in R_k$. (i) If h_i is an internal, a pending call, or return position then $h_{i+1} = h_i + 1$ and $q \in R_{i_j+1}$. By definition of the transition function, we have $s_{k+1} = q$. That contradicts the maximality of k.

(ii) If h_i is a non-pending call position. Then h_{i+1} is the matching return position. By definition of the transition function, we have $s_{h_{i+1}} = q$. Furthermore, we have $b_j = 1$, for all $h_i < j \leq h_{i+1}$. It follows that $h_{i+1} < m$ since $b_m = 0$. However, the fact that $h_{i+1} > k$ contradicts the maximality of k.

(b) There is no $i \in \mathbb{N}$ such that $h_i = k$. Let $j \in \mathbb{N}$ such that $h_j < k$ and $h_{j+1} > k$. It follows that h_j is a call with matching return h_{j+1}. Moreover both positions h_j and h_{j+1} are between m and n. (i) If $s_{h_j} = q$ then by definition of the transition function, we have $s_{h_{j+1}} = q$. This contradicts the maximality

of k.
(ii) If $s_{h_j} > q$ then $s_k > q$, by definition of the transition function.
(iii) If $s_{h_j} < q$ then by definition of the transition function, there is an index $l \geq j+1$ such that $s_{h_l} = q$ and $h_l < n$. This contradicts the maximality of k.

Now, we show that $L^{\mathsf{nw}}(\mathcal{B}) \supseteq \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$. Assume $(w, \rightsquigarrow) \notin L^{\mathsf{nw}}(\mathcal{A})$. Let $R \in (2^Q)^\omega$ be a word that fulfills the conditions (1)–(3) and (5′). We construct an accepting run of $\mathcal{B}$ on $(w, \rightsquigarrow)$. The run $r := (p_0, s_0)(p_1, s_1) \ldots \in (P \times S)^\omega$ is defined as follows.

For defining the components of the run r, we need the following definitions. Let $b \in \{0,1\}^\omega$ be a word such that for every $i \in \mathbb{N}$, we have $b_i = 0$ if and only if i is a sync position. Furthermore, we define the sequences $u, v \in E^\omega$ recursively. Let $u_0 := *$ and $v_0 := *$. For $i \in \mathbb{N}$, we define u_{i+1} and v_{i+1} as follows. Let $D := \mathbb{D}_i$.

(a) If i is an internal, call, or pending-return position then

$$u_{i+1} := \begin{cases} u_i & \text{if } u_i \in R_i \cap \delta_D^1(u_i, w_i) \text{ or } (u_i = * \text{ and } b_i = 1), \\ \mathsf{next}(u_i) & \text{otherwise.} \end{cases}$$

and

$$v_{i+1} := \begin{cases} u_i & \text{if } u_i \in R_i \cap \delta_D^2(u_i, w_i) \text{ or } (u_i = * \text{ and } b_i = 1), \\ \mathsf{next}(u_i) & \text{otherwise.} \end{cases}$$

(b) If i is a non-pending return position with $(j, i) \in \rightsquigarrow_2$. Let m be the minimum of u_i and v_{j+1}.

$$u_{i+1} := v_{i+1} := \begin{cases} m & \text{if } m \in R_i \cap \delta_D^1(m, w_i) \text{ or } (m = * \text{ and } b_i = 1), \\ \mathsf{next}(m) & \text{otherwise.} \end{cases}$$

We define $p_0 := p_I$ and for $i > 0$, we define $p_i := (R_{i-1}, R_i, u_{i-1}, b_{i-1})$. Next, we define the symbols that are pushed on the stack at call positions. For every pending call $i \in \mathbb{N}$, we define $s_{i+1} := \mathsf{pending}$. For every non-pending call $i \in \mathbb{N}$ with $(i, j) \in \rightsquigarrow_2$, we define $s_{i+1} := (R_i, R_j, v_j, b_i)$.

By construction, r is a run. In the following, we show that r is accepting. Assume the opposite, i.e., G is not visited infinitely often. Note that each nested word has infinitely many sync positions. Therefore, the automaton $\mathcal{B}$ does not visit states of the form $(R, R', *, 1)$ with $R, R' \in 2^Q$. It follows that

the automaton $\mathcal{B}$ gets trapped in a non-F-state with its third component of its states. That is, there is a state $q \in Q \setminus F$ and a sequence of positions $h \in \mathbb{N}^\omega$ such that $q \in R_{h_0}$ and for all $i \in \mathbb{N}$, there is a $d > 0$ such that $(h_i, h_{i+1}) \in \rightsquigarrow_d$ and $q \in \delta^d_{\mathbb{D}_{h_i}}(q, w_{h_i})$. Thus, condition (5′) is violated. ∎

Now, consider the case where the existential very-weak co-Büchi automaton whose language has to be complemented is 1-way. In this case, we can simplify condition (2) of Lemma 5.4 by condition (2′) from Section 5.1.1 since the automaton does not move its read-only head backwards. We directly obtain the following theorem as a special case of Theorem 5.5.

Theorem 5.6 *For every V1ECA $\mathcal{A}$ with n states, there is an NWA $\mathcal{B}$ with $\mathcal{O}(2^n n)$ states, $\mathcal{O}(2^n n)$ stack symbols with $L^{\mathsf{nw}}(\mathcal{B}) = \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$.* □

Proof Let $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be a V1ECA. In a nutshell, the construction from Theorem 5.5 consists of three parts: a 2-way subset construction, a construction for guessing the sync positions, and a construction to check that every run of $\mathcal{A}$ does not get trapped in an non-F-state. The construction of the NWA $\mathcal{B}$ follows the same line except for one part. The 2-way subset construction is replaced by the standard 1-way subset construction.

Formally, let $E := (Q \setminus F) \cup \{*\}$. Furthermore, consider a total ordering on the set E and a function $\mathsf{next} : E \to E$ that maps the greatest element $*$ to the smallest one and each of the other elements to the next greater one. We define the NWA $\mathcal{B} := (P, S, \Sigma, \eta, p_I, G)$ as follows.

- $P := S := 2^Q \times E \times \{0, 1\} \cup \{\mathsf{pending}\}$.
- $p_I := (\{q_I\}, *, 0)$ is the initial state.
- $G := 2^Q \times \{*\} \times \{0\}$.

Now, we define the transition function. Let $a \in \hat{\Sigma}$.

- For an internal position, let $D := \{0, 1\}$. We have $\eta_i((R_0, s_0, b), a) \ni (R_1, s_1, b)$ iff the following conditions hold.
  1. For all $d \in D$, we have $R_d = \delta^d_D(R_0, a)$.
  2. For all $p \in R_0$, we have $\mathsf{tt} \notin \delta_D(p, a)$.
  3. $s_1 = \begin{cases} s_0 & \text{if } s_0 \in R \cap \delta^1_D(s_0, a) \text{ or } (s_0 = * \text{ and } b = 1) \\ \mathsf{next}(s_0) & \text{otherwise.} \end{cases}$

- For a call position, let $D := \{0, 1, 2\}$. We have $\eta_i((R_0, s_0, b), a)$ contains $((R_1, s_1, 1), (R_2, s_2, b))$ iff the following conditions hold.
  1. For all $d \in D$, we have $R_d = \delta_D^d(R_0, a)$.
  2. For all $p \in R_0$, we have $\mathsf{tt} \notin \delta_D(p, a)$.
  3. For all $d \in \{1, 2\}$, we have

$$s_d = \begin{cases} s_0 & \text{if } s_0 \in R_0 \cap \delta_D^d(s_0, a) \text{ or } (s_0 = * \text{ and } b = 1) \\ \mathsf{next}(s_0) & \text{otherwise.} \end{cases}$$

  Furthermore, $\eta_c(p, a) \ni ((x, 0), \mathsf{pending})$ iff $\eta_i(p, a) \ni (x, b)$, for some $b \in \{0, 1\}$.

- For a return position, let $D := \{0, 1\}$. We have $\eta_r((R_0, s_0, 1), (R_0, s_0', b), a)$ contains (R_1, s_1, b) iff the following conditions hold.
  1. For all $d \in D$, we have $R_d = \delta_D^d(R_0, a)$.
  2. For all $p \in R_0$, we have $\mathsf{tt} \notin \delta_D(p, a)$.
  3. For $s := \mathsf{min}(s_0, s_0')$, we have

$$s_1 = \begin{cases} s & \text{if } s \in R_0 \cap \delta_D^1(s, a) \text{ or } (s = * \text{ and } b = 1) \\ \mathsf{next}(s) & \text{otherwise.} \end{cases}$$

  For a pending return position, we have $\eta_r(p, \bot, a) := \eta_i(p, a)$, for $p \in P$. Note that there is no transition if the symbol $\mathsf{pending}$ is on the stack.

The proof for $L^{\mathsf{nw}}(\mathcal{B}) = \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$ is a special case of the proof given in Theorem 5.5. ∎

5.1.3 Complementing Parity Automata

In this section, we present a construction to complement the nested-word languages of eventually 1-way 2EPAs. It is a generalization of Kupferman and Vardi's construction from [KV05]: (a) The given automaton operates over nested words instead of just words. (b) The given automaton is not a 1-way automaton but a 2-way automaton that is eventually 1-way.

Theorem 5.7 *For an eventually 1-way 2EPA $\mathcal{A}$ with n states and index k, there is an NWA $\mathcal{B}$ with $2^{\mathcal{O}(nk \log n)}$ states, $2^{\mathcal{O}(nk \log n)}$ stack symbols, and $L^{\mathsf{nw}}(\mathcal{B})$ equals $\hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$.* □

In the remainder of this section, we prove this theorem. For translating the eventually 1-way 2EPA $\mathcal{A}$ into an NWA $\mathcal{B}$ with $L^{\mathsf{nw}}(\mathcal{B}) = \hat{\Sigma}^{\omega} \setminus L^{\mathsf{nw}}(\mathcal{A})$, we start with the following preparatory steps.

(a) We increase the EPA's parities by one to obtain a language-equivalent eventually 1-way 2EcPA.

(b) We translate the eventually 1-way 2EcPA into its dual automaton, i.e., we exchange the Boolean connectives $\wedge$ and $\vee$ and the Boolean constants tt and ff in the transition functions of the 2EcPA. Furthermore, we interpret the acceptance condition again as a parity acceptance condition. In [MS87], Muller and Schupp show that the dual automaton accepts the complement of the language of the original automaton.[1] That is, the dual automaton is an eventually 1-way 2UPA. Note that every universal automaton is memoryless.

(c) Finally, we view the eventually 1-way 2UPA as an eventually 1-way alternating 2ASA. Note that a parity acceptance condition is a special case of a Streett acceptance condition and a universal automaton is a special case of an alternating automaton.

In summary, the resulting 2ASA is memoryless, eventually 1-way, and accepts the complement of $L^{\mathsf{nw}}(\mathcal{A})$. Furthermore, the size and the index of the 2ASA are constant in the size and index of the 2EPA.

In the remainder of the proof, we show how to translate an eventually 1-way 2ASA that accepts by memoryless runs into a language-equivalent NWA. The outline of this translation is as follows. In step 1, we translate the eventually 1-way 2ASA into a language-equivalent eventually 1-way 2AGA that accepts by memoryless and *consistent* runs (see definition below). For this translation, we first show that an eventually 1-way 2ASA accepts a nested word if and only if it has a so-called *accepting Streett ranking* (see definition below). Then, we construct an eventually 1-way 2AGA that accepts a nested word if and only if the given 2ASA has an accepting Streett ranking. In step 2, we translate the 2AGA into a language-equivalent NWA. We remark that both steps are along the lines as Kupferman and Vardi's construction in [KV05].

[1] The result is proven for tree automata but canonically generalizes to automata over graphs.

Step 1: Streett Ranking In the following, let $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ be a nested word and $\mathcal{A} = (Q, \hat{\Sigma}, \delta, q_I, F)$ be an eventually 1-way 2ASA, where its acceptance condition F has the form $\{(C_0, B_0), \ldots, (C_{k-1}, B_{k-1})\}$. Furthermore, we require that $\mathcal{A}$ is memoryless, i.e., $L^\omega(\mathcal{A}) = M^\omega(\mathcal{A})$.

Consider a memoryless run $r : R \to Q \times \mathbb{N}$ of $\mathcal{A}$ on $(w, \rightsquigarrow)$. We define a directed graph $G := (V, E)$ that represents this run with

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

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

Consider a function $f : V \to [2n]^k$. For $i \in [k]$, we denote the projection of f on the ith component by $f_i : V \to [2n]$ and call it *i-rank*. The function f is an *Streett ranking* for G if the following two conditions hold.

(i) For all $v \in V$ and $i \in [k]$, if $f_i(v)$ is odd then v is not a C_i-vertex.

(ii) For all $(v, v') \in E$ and $i \in [k]$, either $f_i(v) \geq f_i(v')$ or v is a B_i-vertex.

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

Lemma 5.8 *Every infinite path in G fulfills the Streett acceptance condition F if and only if there is an accepting Streett ranking for G.* □

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

The *if direction* is easy to see. Assume that there is an accepting Streett ranking f for G. That is, every infinite path in G visits B-vertices infinitely

often or gets trapped in an odd rank. By definition, this means, every infinite path visits B-vertices infinitely often or eventually avoids visiting C-vertices. Hence, every infinite path in G fulfills the Streett acceptance condition F.

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

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

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

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

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

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

Lemma 5.9 *For every $i \in \mathbb{N}$, the graph G_{2i+1} is empty or has a C-free vertex from which an infinite path of C-free vertices starts.* □

Proof Let $i \in \mathbb{N}$. Consider the graph G_{2i}. We make a case distinction. If G_{2i} is finite then G_{2i+1} is empty and we are done. Otherwise, if G_{2i} is infinite then G_{2i+1} is infinite. For the sake of contradiction, assume that there is no C-free vertex in G_{2i+1}. Note that every vertex in G_{2i+1} has at least one successor. Consider some vertex v_1 in G_{2i+1}. Let v_1' be a successor of v_1. Since v_1' is not C-free, there is a C-vertex v_2 reachable from v_1'. Let v_2' be a successor of v_2. Since v_2' is not C-free, there is a C-vertex v_3 reachable from v_2'. Let v_3' be a

successor of v_3. If we continue this way, we can construct an infinite path that does not visit any B-vertex but visits C-vertices infinitely often. This path corresponds to a rejecting path in G, which contradicts the assumption that every infinite path in G fulfills the Streett acceptance condition. ∎

In the next lemma, we show that we obtain an empty graph from G in $2n$ steps by alternately removing infinite paths according to Lemma 5.9 and finite vertices from G. Let $H \subseteq \mathbb{N}$ be the set of head position h such that h is not strictly between a call position and its matching return position, i.e.,

$$H := \{h \in \mathbb{N} \mid \text{there are no } i, j \in \mathbb{N} \text{ such that } i < h < j \text{ and } (i, j) \in \rightsquigarrow_2\}.$$

Lemma 5.10 *For every $i \in \mathbb{N}$, either G_{2i+1} is empty or there is a position $n \in \mathbb{N}$ such that for every $h \in H$ with $h \geq n$, we have $|\{(q, h) \mid (q, h) \in V_{2i+2}\}| < |\{(q, h) \mid (q, h) \in V_{2i+1}\}|$.* □

Proof If G_{2i+1} is empty, we are done. Otherwise, consider an infinite path $\pi = (q_0, h_0)(q_1, h_1)\ldots$ in G_{2i+1} that consists of only C-free vertices. This path exists by Lemma 5.9. It suffices to show that for each $h \in H$ with $h > h_0$, there is a vertex (q, h) occurring in π. This is obvious since $\mathcal{A}$ is eventually 1-way and cannot jump over the positions in H on infinite paths of its run. ∎

Corollary 5.11 *G_{2n} is finite and G_{2n+1} is empty.* □

Proof First, observe that H is infinite. Second, note that G_{2n} does not contain any vertex (q, h) with $h \in H$, since G contains at most n vertices with head position h and from Lemma 5.10, it follows that for each $i \in [n]$, at least one vertex in G_{2i+1} with head position h gets removed.

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

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

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

Obviously, f fulfills condition (i) of the definition of a Streett ranking. Condition (ii) follows from the next lemma.

Lemma 5.12 *For all vertices $v, v' \in V$, we have $f(v') \leq f(v)$ if v' is reachable from v without visiting a B-vertex.* □

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

Assume that $f(v) = i$. We distinguish between three cases. (a) If $v' \notin V_i$ then $f(v') < f(v)$ by the fact from above. (b) Assume $v' \in V_i$ and i is even. Since v' is reachable from v in G without visiting a B-vertex, it follows that v' is reachable from v in V_i. Since v is finite, v' is also finite. Hence, $f(v') \leq f(v)$. (c) If $v' \in V_i$ and i is odd then v and v' are C-free in V_i, and hence, $f(v') \leq f(v)$. ■

Finally, we show that the Streett ranking f is accepting.

Lemma 5.13 *Every infinite path in G that does not visit B-vertices infinitely often gets trapped in an odd rank.* □

Proof Let $\pi = v_0 v_1 \ldots$ be an infinite path in G that avoids visiting B-vertices infinitely often. According to Lemma 5.12, there must be a position $k \in \mathbb{N}$ in π such that $f(v_m) = f(v_k)$, for all $m \geq k$. We show that $f(v_k)$ is odd. By the sake of contradiction, assume that $f(v_k)$ is even. Then, every vertex v in the path that is reachable from v_k is finite in $G_{f(v_k)}$. The path is infinite, and therefore, v_k cannot be finite in $G_{f(v_k)}$. ■

Step 2: Alternation Elimination We construct an eventually 1-way 2AGA $\mathcal{B}$ and show that the automaton is language-equivalent to the eventually 1-way 2ASA $\mathcal{A}$. Furthermore, we show that $\mathcal{B}$ accepts by runs having a special structure. This structure can be exploited in the alternation-elimination construction to avoid an overall double-exponential blow-up when translating $\mathcal{B}$ into a language-equivalent NWA.

We note that this special structure of runs of the 2ABA is observed by Kupferman and Vardi in [KV05] when the authors remove alternation from automata over words. Here, we make this property explicit and show that

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

$$\big\{(q, r) \mid \text{ there is a node } x \in T \text{ with } t(x) = ((q, r), h)\big\}$$

is consistent. For a node $x \in T$, we call the last component of $t(x)$ head position.

Theorem 5.14 *For a memoryless eventually 1-way 2ASA $\mathcal{A}$ with n states and index k, there is an eventually 1-way 2AGA $\mathcal{B}$ with $\mathcal{O}(n^k)$ states and $L^{\mathsf{nw}}(\mathcal{B}) = L^{\mathsf{nw}}(\mathcal{A})$. Furthermore, $\mathcal{B}$ accepts by memoryless and consistent runs.* □

Proof Let $\mathcal{A} = \big(Q, \hat{\Sigma}, \delta, q_I, \{(C_0, B_0), \ldots, (C_{k-1}, B_{k-1})\}\big)$ be an eventually 1-way 2ASA. Intuitively, the eventually 1-way 2AGA $\mathcal{B}$ that we construct from $\mathcal{A}$ works as follows. For a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$, it simulates a run of $\mathcal{A}$ on $(w, \rightsquigarrow)$ and annotates the run by a guessed Streett ranking. That is, each configuration of the run is annotated by a rank in $[2n]^k$. Then, the automaton $\mathcal{B}$ checks that the guessed Streett ranking fulfills the conditions (i) and (ii).

We need the following definition for constructing the check of condition (ii). For a state $q \in Q$ and two ranks $r, r' \in \mathbb{N}^k$, we write $r' \leq_q r$ if and only if for every $i \in [k]$, we either have $r'_i \leq r_i$ or $q \in B_i$. Intuitively, the automaton may move to all possible successor states whose rank is lower. For a given formula $\varphi \in \mathcal{B}^+(Q \times \mathbb{D})$, a state $q \in Q$ and a rank $r \in [2n]^k$, we define $\mathsf{rel}_q(\varphi, r)$ as the positive Boolean formula that we obtain by replacing each proposition (p, d) in φ by the disjunction $\bigvee_{r' \leq_q r}((p, r'), d)$.

Now, we formally define $\mathcal{B} := (Q \times [2n]^k, \hat{\Sigma}, \eta, p_I, \{F_0, \ldots, F_{k-1}\})$.

- The initial state is $p_I := (q_I, 2n, \ldots, 2n)$.
- For $q \in Q$, $r \in [2n]^k$, $D \subseteq \mathbb{D}$, and $a \in \hat{\Sigma}$, we define

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

  That is, $\mathcal{B}$ checks condition (i) and then moves to states with lower ranks.

- For $i \in [k]$, the acceptance set F_i contains the state (q, r) if and only if $q \in B_i$ or r_i is odd. That is, every path in a run either visits B_i infinitely often or its ith's component gets trapped in an odd rank.

It remains to prove that $\mathcal{B}$ accepts the same nested-word language as $\mathcal{A}$. We first show that $L^{\mathsf{nw}}(\mathcal{B}) \subseteq L^{\mathsf{nw}}(\mathcal{A})$. Let $(w, \rightsquigarrow) \in L^{\mathsf{nw}}(\mathcal{B})$ and let $t' : T \to (Q \times [2n]^k) \times \mathbb{N}$ be an accepting run of $\mathcal{B}$ on $(w, \rightsquigarrow)$. Consider the tree $t : T \to Q \times \mathbb{N}$ with $t(x) := (q, h)$, for every $x \in T$ with $t'(x) = (q, i, h)$. That is, t is the projection of t' on Q and $\mathbb{N}$. The tree t is a run of $\mathcal{A}$ on $(w, \rightsquigarrow)$ since the transitions of $\mathcal{B}$ just annotate the transitions of $\mathcal{A}$ by ranks. We show that t is accepting. Since t' is accepting, every path in t' visits some (q, r), where for every component r_i of r, for $i \in [k]$, we either have $q \in B_i$ or r_i is odd. Let $i \in [k]$. Consider an infinite path in t' that does not visit any state from B_i. By definition of the acceptance condition, the path gets trapped in the set $\{(q, r) \in Q \times [2n]^k \mid i \in [k] \text{ and } r_i \text{ is odd}\}$. Thus, by definition of η, the path eventually avoids visiting states from C_i. Consequently, the projection of the path on Q is an accepting path in r. Hence, r is accepting.

Now, we prove that $L^{\mathsf{nw}}(\mathcal{B}) \subseteq L^{\mathsf{nw}}(\mathcal{A})$. Let $(w, \rightsquigarrow) \in L^{\mathsf{nw}}(\mathcal{A})$ and $t : T \to Q \times \mathbb{N}$ be an accepting memoryless run of $\mathcal{A}$ on the nested word $(w, \rightsquigarrow)$. Let $G = (V, E)$ be the directed graph obtained from t. Furthermore, let $f : V \to [2n]^k$ be an accepting Streett ranking for G. We define a tree $t' : T \to (Q \times [2n]^k) \times \mathbb{N}$ and show that it is an accepting run of $\mathcal{B}$ on $(w, \rightsquigarrow)$. Let $t'(\varepsilon) := p_I$ and for a node $x \in T \setminus \{\varepsilon\}$ with $t(x) = (q, h)$, we define $t'(x) := ((q, f(q, h)), h)$. We show that t' is a run. By definition of t', we have $t'(\varepsilon) = p_I$. Consider a node $x \in T$ with $t'(x) = ((q, r), h)$. By conditions (i) and (ii) of the Streett ranking f, the set of labels of the successors of node x fulfill the transition function $\eta_D((q, r), w_h)$, for $D \subseteq \mathbb{D}$. Finally, by Lemma 5.13, every infinite path in G that does not visit B_i-vertices infinitely often, gets trapped in an odd i-rank, for every $i \in [k]$. Hence, by definition of the generalized Büchi acceptance condition $\{F_0, \ldots, F_{k-1}\}$, every infinite path in t' is accepting. Furthermore, note that t is a memoryless run and for every two configurations (q, h) and (q', h') in t with $(q, h) = (q', h')$, we have $f(q, h) = f(q', h')$. By definition of t', it follows that t' is also memoryless. Hence, $\mathcal{B}$ accepts by memoryless runs.

It remains to show that the run t' is consistent. By definition of t', for every $x \in T$ with $t'(x) = ((q, r), h)$, r is the rank of vertex (q, h) in G. Since every vertex in G has a unique rank, t' is consistent. ∎

Theorem 5.15 *For a memoryless eventually 1-way 2ESA $\mathcal{A}$ with n states and index k, there is an NWA $\mathcal{B}$ with $2^{\mathcal{O}(nk \log n)}$ states and $2^{\mathcal{O}(nk \log n)}$ stack symbols, and $L^{\mathsf{nw}}(\mathcal{B}) = L^{\mathsf{nw}}(\mathcal{A})$.* □

Proof First, we use Theorem 5.14 for translating the eventually 1-way 2ASA $\mathcal{A} := (Q, \hat{\Sigma}, \delta, q_I, F)$ into a language-equivalent eventually 1-way 2AGA $\mathcal{B}$ of size n^k that accepts by memoryless and consistent runs. Next, we use our alternation-elimination scheme to translate $\mathcal{B}$ into a language-equivalent NWA $\mathcal{C}$. That is, we have to complement an eventually 1-way 2EcGA of size n^k.

Recall Theorem 5.2 for complementing eventually 1-way 2ECAs. We adapt this construction for complementing an eventually 1-way 2EcGA $\mathcal{R}$ as follows. Let $\{F_0, \ldots, F_{k-1}\}$ be the acceptance condition of $\mathcal{R}$. We add a counter $i \in [k]$ to each state of the resulting NWA $\mathcal{C}$. The NWA $\mathcal{C}$ sequentially checks that there is no run of $\mathcal{R}$ that gets trapped in some F_i. Along the lines of the construction of Theorem 5.2, the NWA $\mathcal{C}$ guesses infinitely many sync positions $i_0, i_1, \ldots \in \mathbb{N}$ of the given input word $(w, \rightsquigarrow)$ such that for every such sync position i_j, for $j \in \mathbb{N}$, and every run of $\mathcal{R}$ that starts in a state that can be reached by reading the prefix $w_{0..i_j+1}$, visits the set $Q \setminus F_{(j \bmod k)}$ before reaching position i_{j+1} in the input word. This is accomplished by changing the transition function from non-initial states in the construction of Theorem 5.2. Namely, condition (4) is changed to: whenever $S_0 = \emptyset$ and $b = 0$ then $S_1 = R_1 \setminus F_i$ and $i' = (i + 1) \bmod k$, where i is the current counter value and i' is the next counter value.

Note that the NWA $\mathcal{C}$ has size $\mathcal{O}(k 2^{4n^k})$. We now show how to restrict the state space of the NWA $\mathcal{C}$. Let $\mathcal{R}$ be the refuter automaton for $\mathcal{B}$ according to our scheme. Consider a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ and a memoryless consistent run $t : T \to (Q \times [2n]^k) \times \mathbb{N}$ of $\mathcal{B}$ on $(w, \rightsquigarrow)$. We can represent this run by a nested word $(s, \rightsquigarrow) \in \hat{\Gamma}^\omega$, where $\Gamma := Q \to 2^{(Q \times [2n]^k) \times \mathbb{D}}$. Now, consider the nested word $(w \times s, \rightsquigarrow)$, where $w \times s \in (\hat{\Sigma} \times \hat{\Gamma})^\omega$ with $(w \times s)(i) := (w(i), s(i))$, for all $i \in \mathbb{N}$, and a tree $t' : T' \to (Q \times [2n]^k) \times \mathbb{N}$ that represents all runs of the refuter automaton $\mathcal{R}$ on $(w \times s, \rightsquigarrow)$, i.e., each path of the tree represents a run of the existential automaton $\mathcal{R}$. We show that this tree is consistent. Intuitively, the refuter automaton follows a path in the run given by $(s, \rightsquigarrow)$. So, the tree that represents all runs of the refuter automaton is only a sub-tree of the run t. Pick an arbitrary head position $h \in \mathbb{N}$. We have to show that the set

$$S' := \{t'(x) \mid t'(x) \text{ has head position } h, \text{ for some node } x \in T'\}$$

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

$$S := \{t(x) \mid t(x) \text{ has head position } h, \text{ for some node } x \in T'\},$$

which is consistent by assumption. Let $(q, h) \in S'$. Then there is a prefix $(q_0, h_0)(q_1, h_1)\dots(q_n, h_n) \in ((Q \times [2n]^k) \times \mathbb{N})^*$ of a run of $\mathcal{R}$ on a prefix of $(w, \rightsquigarrow)$ such that $(q_n, h_n) = (q, h)$. Since, $q_{i+1} \in s_i(q_i)$, for all $i \in [n]$, there is a node in r that is labeled by (q, h). Hence, $(q, h) \in S$.

Since t' is consistent, we can optimize the construction of the NWA $\mathcal{C}$ that complements the language of the automaton $\mathcal{R}$. Recall that the state space and space of the stack alphabet of $\mathcal{C}$ is $P = (2^{(Q\times[2n]^k)} \times 2^{(Q\times[2n]^k)} \times 2^{(Q\times[2n]^k)} \times 2^{(Q\times[2n]^k)} \times \{0,1\} \times [k]) \cup \{q_I\}$. Consider a state $(R, S, R', S', b, i) \in P$. The R and R' component correspond to labels in the tree t' having the same head position h and $h+1$, for some $h \in \mathbb{N}$, respectively. Hence, both sets are consistent. Since $S \subseteq R$ and $S' \subseteq R'$, we can restrict tuples in P to tuples whose components are consistent. That is, instead of using P, we can restrict the states space and the space of the stack alphabet to

$$P' := \{q_I\} \cup \{(R, S, R', S', b, i) \mid S \subseteq R, S' \subseteq R', b \in \{0,1\}, i \in [k] \text{ and } R, S, R', S' \text{ are consistent}\}.$$

An upper bound on the cardinality of P' is as follows. Consider a state $(R, S, R', S', b, i) \in P'$. We can represent (R, S, R', S') by a quadruple of functions in $Q \to [2n]^k$. Therefore, the size of P' is $\mathcal{O}(k((([2n]^k)^4)^n) = 2^{\mathcal{O}(nk \log n)}$. ∎

Complementing 2-way Existential Streett Automata By combining constructions—along the same lines as in [Var88, Var98]—we can generalize the above construction to 2EPAs. The first ingredient is Shepherdson's translation [She59, Var89] of a 2-way nondeterministic finite word automaton into a deterministic one, which generalizes to automata over nested words with only minor modifications. The second ingredient is a complementation construction for nondeterministic 1-way automata. Intuitively, the first construction eliminates the bi-directionality of runs of any 2-way word automaton. The second construction is then used to complement the resulting 1-way automaton. In our setting, we combine the generalized version of Shepherdson's translation with the complementation construction in Theorem 5.7. This combination results in the following theorem.

Theorem 5.16 *For a 1EPA $\mathcal{A}$ with n states and index k, there is an NWA $\mathcal{B}$ with $2^{\mathcal{O}((nk)^2)}$ states, $2^{\mathcal{O}((nk)^2)}$ stack symbols, and $L^{\mathsf{nw}}(\mathcal{B}) = \hat{\Sigma}^\omega \setminus L^{\mathsf{nw}}(\mathcal{A})$.* □

5.2 From Temporal Logics to Automata

In this section, we discuss an application of our alternation-elimination scheme. We show how to translate temporal logics over nested words into nondeterministic nested-word automata. In particular, we identify a flaw in a translation from the logic NWTL to NWAs given in [AAB+08] and provide a correct translation that is based on our automata constructions. Then, we introduce the new logic NWPSL based on the IEEE standard PSL. We show that NWPSL is more expressive than NWTL whereas the worst-case sizes of the NWAs obtained from the translations from the logics NWPSL and NWTL differ only by a small constant in the exponent. Finally, we present a new translation from the logic μNWTL [Boz07] to NWAs. For a relevant subclass of μNWTL that contains formulas without past operators, our construction improves over Bozelli's translation: the worst-case sizes of the resulting NWAs are bound by $2^{\mathcal{O}(nk \log n)}$ instead of $2^{\mathcal{O}((nk)^2)}$, where n is the size of the formula and k the alternation depth of the fix-point quantifiers.

For the remainder of this section, let $\mathcal{P}$ denote a finite, non-empty set of propositions and let $\Sigma := 2^{\mathcal{P}}$ denote the finite alphabet over $\mathcal{P}$. Furthermore, for a Boolean formula $\varphi \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$ and a set $M \subseteq \hat{\Sigma}$, we say that M *satisfies* the formula φ if and only if either (a) $M \in \Sigma_c$ and $M \cup \{\mathsf{call}\} \models \varphi$, (b) $M \in \Sigma_r$ and $M \cup \{\mathsf{ret}\} \models \varphi$, or (c) $M \in \Sigma_i$ and $M \models \varphi$.

5.2.1 Nested Word Temporal Logic

Nested Word Temporal Logic (NWTL) [AAB+08] is an extension of the well-known linear-time temporal logic (LTL) that is used for describing properties over nested words. In [AAB+08], the authors show that NWTL has the same expressive power as first-order logic over nested words. Furthermore, they give a translation of NWTL formulas into NWAs to obtain decision procedures for this logic. We remark that this translation has a flaw. Consider the formula $F^\sigma \mathsf{ff}$ that is equivalent to the formula ff. The NWA constructed from $F^\sigma \mathsf{ff}$ accepts the nested word $((\langle \emptyset\ \emptyset\ \emptyset \rangle)^\omega, \rightsquigarrow)$, where $\rightsquigarrow := \{(i, i+2) \in \mathbb{N}^2 \mid 3 \text{ divides } i\}$. We present an alternative translation from NWTL to NWA

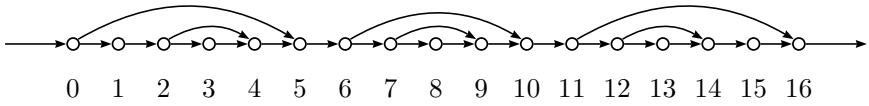


Figure 5.1: Paths in a nested word

based on our alternation-elimination scheme and the novel complementation construction from the Section 5.1.

In the following, we define the logic NWTL. The syntax of an NWTL formula over $\mathcal{P}$ is given by the following grammar.

$$\varphi ::= b \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{O}\varphi \mid \Theta\varphi \mid \mathsf{O}_\mu\varphi \mid \Theta_\mu\varphi \mid \varphi\, \mathsf{U}^\sigma\, \varphi \mid \varphi\, \mathsf{S}^\sigma\, \varphi$$

where $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$. We use the syntactic abbreviation $\mathsf{int} := \neg\mathsf{call} \wedge \neg\mathsf{ret}$.

We interpret NWTL formulas over nested words over Σ. For the definition of the fix-point operators, we need the following notion. A summary path is the shortest path between two positions. Formally, a *summary path* in a nested word $(w, \rightsquigarrow)$ is a finite sequence $i_0 \dots i_k \in \mathbb{N}^*$ of positions such that for all $j \leq k$, we have

$$i_{j+1} = \begin{cases} r & \text{if } (i_j, r) \in \rightsquigarrow_2 \text{ and } r \leq i_k, \\ i_j + 1 & \text{otherwise.} \end{cases}$$

For instance, the path $1, 2, 4, 5, 6, 10, 11, 12, 14, 15$ in Figure 5.1 is a summary path.

For a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ and a position $i \in \mathbb{N}$ in w, we define the semantics of NWTL as follows.

$$\begin{array}{lll}
(w, \rightsquigarrow, i) \models b & \text{iff} & w_i \text{ satisfies } b \\
(w, \rightsquigarrow, i) \models \neg\varphi & \text{iff} & (w, \rightsquigarrow, i) \not\models \varphi \\
(w, \rightsquigarrow, i) \models \varphi \vee \psi & \text{iff} & (w, \rightsquigarrow, i) \models \varphi \text{ or } (w, \rightsquigarrow, i) \models \psi \\
(w, \rightsquigarrow, i) \models \mathsf{O}\varphi & \text{iff} & (w, \rightsquigarrow, i+1) \models \varphi \\
(w, \rightsquigarrow, i) \models \Theta\varphi & \text{iff} & i > 0 \text{ and } (w, \rightsquigarrow, i-1) \models \varphi \\
(w, \rightsquigarrow, i) \models \mathsf{O}_\mu\varphi & \text{iff} & (i, j) \in \rightsquigarrow_2 \text{ and } (w, \rightsquigarrow, j) \models \varphi, \text{ for some } j \in \mathbb{N} \\
(w, \rightsquigarrow, i) \models \Theta_\mu\varphi & \text{iff} & (j, i) \in \rightsquigarrow_2 \text{ and } (w, \rightsquigarrow, j) \models \varphi, \text{ for some } j \in \mathbb{N} \\
(w, \rightsquigarrow, i) \models \varphi\, \mathsf{U}^\sigma\, \psi & \text{iff} & \text{there is a summary path } l_0 \dots l_k \text{ such that} \\
 & & l_0 = i, (w, \rightsquigarrow, l_k) \models \psi, \text{ and } \forall j < k : (w, \rightsquigarrow, l_j) \models \varphi \\
(w, \rightsquigarrow, i) \models \varphi\, \mathsf{S}^\sigma\, \psi & \text{iff} & \text{there is a summary path } l_0 \dots l_k \text{ such that} \\
 & & l_k = i, (w, \rightsquigarrow, l_0) \models \psi, \text{ and } \forall j > 0 : (w, \rightsquigarrow, l_j) \models \varphi,
\end{array}$$

where $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$. We denote the language of an NWTL formula φ by $L^{\mathsf{nw}}(\varphi) := \{(w, \rightsquigarrow) \in \hat{\Sigma}^\omega \mid (w, \rightsquigarrow, 0) \models \varphi\}$.

We use the following abbreviations to translate any NWTL formula into *positive normal form*, i.e., negations occur only in front of propositional-logic formulas. For the two NWTL formulas φ and ψ, we define the following operators.

- $\overline{\Theta}\varphi := \neg\Theta\neg\varphi$. That is, $\overline{\Theta}\varphi$ is equivalent to $\neg\Theta\mathsf{tt} \vee \Theta\varphi$, which intuitively means that either there is no previous position, or φ holds at the previous position.

- $\overline{\mathsf{O}}_\mu\varphi := \neg\mathsf{O}_\mu\neg\varphi$. That is, $\overline{\mathsf{O}}_\mu\varphi$ is equivalent to $\neg\mathsf{O}_\mu\mathsf{tt} \vee \mathsf{O}_\mu\varphi$, which intuitively means that either the current position is not a call with a matching return position, or at the matching return position φ holds.

- $\varphi\, \mathsf{R}^\sigma\, \psi := \neg\varphi\, \mathsf{U}^\sigma\, \neg\psi$. The operator R^σ corresponds to the release operator R known from LTL. While R is interpreted over linear paths, R^σ is interpreted over summary paths.

- $\varphi\, \mathsf{T}^\sigma\, \psi := \neg\varphi\, \mathsf{S}^\sigma\, \neg\psi$. The operator T^σ corresponds to the past operator T known from LTL.

For the translation of NWTL formulas into automata, we additionally need to substitute the fix-point connectives by operators that are defined over special kinds of summary paths. A summary-up path is a summary path, where every call may only be followed by its matched return. Formally, we call a summary path $p \in \mathbb{N}^*$ a *summary-up path* if for all $i < |p|$, if p_i is a call then p_{i+1} is its matched return. For instance, the path $1, 2, 4, 5, 6$ in Figure 5.1 is a summary-up path. A summary-down path is a summary path, where a return position can only be reached from its matching call position. Formally, we call a summary path $p \in \mathbb{N}^*$ a *summary-down path* if for all $i < |p|$, if p_{i+1} is a return position then p_i is its matching call position. For instance, the path $5, 6, 10, 11, 12, 14, 15$ in Figure 5.1 is a summary-down path. For a nested word

$(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ and a position $i \in \mathbb{N}$ in w, we define

$$
\begin{array}{lll}
(w, \rightsquigarrow, i) \models \varphi \,\mathsf{U}^{\sigma\uparrow}\, \psi & \text{iff} & \text{there is a summary-up path } l_0 \dots l_k \text{ such that} \\
 & & l_0 = i, (w, \rightsquigarrow, l_k) \models \psi, \text{ and } \forall j < k : (w, \rightsquigarrow, l_j) \models \varphi, \\
(w, \rightsquigarrow, i) \models \varphi \,\mathsf{U}^{\sigma\downarrow}\, \psi & \text{iff} & \text{there is a summary-down path } l_0 \dots l_k \text{ such that} \\
 & & l_0 = i, (w, \rightsquigarrow, l_k) \models \psi, \text{ and } \forall j < k : (w, \rightsquigarrow, l_j) \models \varphi, \\
(w, \rightsquigarrow, i) \models \varphi \,\mathsf{S}^{\sigma\uparrow}\, \psi & \text{iff} & \text{there is a summary-up path } l_0 \dots l_k \text{ such that} \\
 & & l_k = i, (w, \rightsquigarrow, l_0) \models \psi, \text{ and } \forall j > 0 : (w, \rightsquigarrow, l_j) \models \varphi, \\
(w, \rightsquigarrow, i) \models \varphi \,\mathsf{S}^{\sigma\downarrow}\, \psi & \text{iff} & \text{there is a summary-down path } l_0 \dots l_k \text{ such that} \\
 & & l_k = i, (w, \rightsquigarrow, l_0) \models \psi, \text{ and } \forall j > 0 : (w, \rightsquigarrow, l_j) \models \varphi.
\end{array}
$$

Note that the equivalences $\alpha \,\mathsf{U}^{\sigma}\, \beta \equiv \alpha \,\mathsf{U}^{\sigma\uparrow}\, (\alpha \,\mathsf{U}^{\sigma\downarrow}\, \beta)$ and $\alpha \,\mathsf{S}^{\sigma}\, \beta \equiv \alpha \,\mathsf{S}^{\sigma\uparrow}\, (\alpha \,\mathsf{S}^{\sigma\downarrow}\, \beta)$ hold. We define the abbreviations $\alpha \,\mathsf{R}^*\, \beta := \neg(\neg\alpha \,\mathsf{U}^*\, \neg\beta)$ and $\alpha \,\mathsf{T}^*\, \beta := \neg(\neg\alpha \,\mathsf{S}^*\, \neg\beta)$, for $* \in \{\sigma\uparrow, \sigma\downarrow\}$. A *normalized* NWTL formula is a formula of the form

$$
\begin{array}{rcl}
\varphi & ::= & b \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathsf{O}\varphi \mid \overline{\mathsf{O}}\varphi \mid \Theta\varphi \mid \overline{\Theta}\varphi \mid \mathsf{O}_\mu\varphi \mid \overline{\mathsf{O}}_\mu\varphi \mid \Theta_\mu\varphi \mid \overline{\Theta}_\mu\varphi \mid \\
 & & \varphi \,\mathsf{U}^*\, \varphi \mid \varphi \,\mathsf{R}^*\, \varphi \mid \varphi \,\mathsf{S}^*\, \varphi \mid \varphi \,\mathsf{T}^*\, \varphi,
\end{array}
$$

where $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$ and $* \in \{\sigma\uparrow, \sigma\downarrow\}$. Note that a translation of an NWTL formula into an equivalent normalized NWTL formula might exponentially increase the size of the formula. However, the number of subformulas remains linear in the size of the input formula.

The following theorem states that we can translate every NWTL formula into a language-equivalent NWA.

Theorem 5.17 *Every normalized NWTL formula of size n can be translated into a language-equivalent 2ABA of size $\mathcal{O}(n)$ and into a language-equivalent NWA of size $\mathcal{O}(2^{2n}n)$.* □

Proof Let φ be a normalized NWTL formula. We first start with some preparatory work. Note that the following equivalences hold for any normalized NWTL formulas α and β, see also [AAB+08] for more details.

$$
\begin{array}{l}
\alpha \,\mathsf{U}^{\sigma\uparrow}\, \beta \equiv \beta \vee \big(\alpha \wedge \mathsf{O}_\mu(\alpha \,\mathsf{U}^{\sigma\uparrow}\, \beta)\big) \vee \big(\alpha \wedge \neg\mathsf{call} \wedge \mathsf{O}(\alpha \,\mathsf{U}^{\sigma\uparrow}\, \beta)\big) \\
\alpha \,\mathsf{U}^{\sigma\downarrow}\, \beta \equiv \beta \vee \big(\alpha \wedge \mathsf{O}_\mu(\alpha \,\mathsf{U}^{\sigma\downarrow}\, \beta)\big) \vee \big(\alpha \wedge \neg\mathsf{ret} \wedge \mathsf{O}(\alpha \,\mathsf{U}^{\sigma\uparrow}\, \beta)\big) \\
\alpha \,\mathsf{S}^{\sigma\uparrow}\, \beta \equiv \beta \vee \big(\alpha \wedge \Theta_\mu(\alpha \,\mathsf{S}^{\sigma\uparrow}\, \beta)\big) \vee \big(\alpha \wedge \neg\mathsf{call} \wedge \Theta(\alpha \,\mathsf{S}^{\sigma\uparrow}\, \beta)\big) \\
\alpha \,\mathsf{S}^{\sigma\downarrow}\, \beta \equiv \beta \vee \big(\alpha \wedge \Theta_\mu(\alpha \,\mathsf{S}^{\sigma\downarrow}\, \beta)\big) \vee \big(\alpha \wedge \neg\mathsf{ret} \wedge \Theta(\alpha \,\mathsf{S}^{\sigma\uparrow}\, \beta)\big)
\end{array}
$$

$$\alpha \,\mathsf{R}^{\sigma\uparrow}\, \beta \equiv \beta \wedge \big(\alpha \vee \overline{\mathsf{O}}_\mu(\alpha \,\mathsf{R}^{\sigma\uparrow}\, \beta)\big) \wedge \big(\alpha \vee \mathsf{call} \vee \mathsf{O}(\alpha \,\mathsf{R}^{\sigma\uparrow}\, \beta)\big)$$
$$\alpha \,\mathsf{R}^{\sigma\downarrow}\, \beta \equiv \beta \wedge \big(\alpha \vee \overline{\mathsf{O}}_\mu(\alpha \,\mathsf{R}^{\sigma\downarrow}\, \beta)\big) \wedge \big(\alpha \vee \mathsf{ret} \vee \mathsf{O}(\alpha \,\mathsf{R}^{\sigma\uparrow}\, \beta)\big)$$
$$\alpha \,\mathsf{T}^{\sigma\uparrow}\, \beta \equiv \beta \wedge \big(\alpha \vee \overline{\Theta}_\mu(\alpha \,\mathsf{T}^{\sigma\uparrow}\, \beta)\big) \wedge \big(\alpha \vee \mathsf{call} \vee \overline{\Theta}(\alpha \,\mathsf{T}^{\sigma\uparrow}\, \beta)\big)$$
$$\alpha \,\mathsf{T}^{\sigma\downarrow}\, \beta \equiv \beta \wedge \big(\alpha \vee \Theta_\mu(\alpha \,\mathsf{T}^{\sigma\downarrow}\, \beta)\big) \wedge \big(\alpha \vee \mathsf{ret} \vee \overline{\Theta}(\alpha \,\mathsf{T}^{\sigma\uparrow}\, \beta)\big)$$

Let $\varphi \equiv \psi$ be one of the equations from above. We call $\mathsf{Unf}(\varphi) := \psi$ the *unfolding* of the formula φ.

Construction We define the 2ABA $\mathcal{A}_\varphi$ and show that this automaton is very weak, eventually 1-way, and language-equivalent to $L^{\mathsf{nw}}(\varphi)$.

Let $\mathcal{A}_\varphi := (Q, \hat{\Sigma}, \delta, \varphi, F)$, where $Q := \mathsf{Sub}(\varphi)$ is the set of sub-formulas of φ and $F := \{\alpha \,\mathsf{R}^*\, \beta \in \mathsf{Sub}(\varphi) \mid * \in \{\sigma\uparrow, \sigma\downarrow\}\}$ is the set of release sub-formulas of φ. The transition function is inductively defined over the formula structure. Let $a \in \hat{\Sigma}$ be a letter and $D \subseteq \mathbb{D}$. For $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\}) \cap \mathsf{Sub}(\varphi)$, we have

$$\delta_D(b, a) := \begin{cases} \mathsf{tt} & \text{if } a \text{ satisfies } b, \\ \mathsf{ff} & \text{otherwise.} \end{cases}$$

For $\alpha, \beta \in \mathcal{B}^+(\mathsf{Sub}(\varphi))$, we have

$$\delta_D(\alpha \vee \beta, a) := \delta(\alpha, a) \vee \delta(\beta, a),$$
$$\delta_D(\alpha \wedge \beta, a) := \delta(\alpha, a) \wedge \delta(\beta, a).$$

For $\alpha \in \mathsf{Sub}(\varphi)$, we have

$$\delta_D(\mathsf{O}\alpha, a) := (\alpha, 1),$$
$$\delta_D(\Theta\alpha, a) := (\alpha, -1),$$
$$\delta_D(\overline{\Theta}\alpha, a) := \begin{cases} \mathsf{tt} & \text{if } -1 \notin D, \\ (\alpha, -1) & \text{otherwise,} \end{cases}$$
$$\delta_D(\mathsf{O}_\mu\alpha, a) := (\alpha, 2),$$
$$\delta_D(\overline{\mathsf{O}}_\mu\alpha, a) := \begin{cases} \mathsf{tt} & \text{if } 2 \notin D, \\ (\alpha, 2) & \text{otherwise,} \end{cases}$$
$$\delta_D(\Theta_\mu\alpha, a) := (\alpha, -2),$$
$$\delta_D(\overline{\Theta}_\mu\alpha, a) := \begin{cases} \mathsf{tt} & \text{if } -2 \notin D \\ (\alpha, -2) & \text{otherwise.} \end{cases}$$

And finally, for a fix-point formula $\psi \in \mathsf{Sub}(\varphi)$ of the form $\alpha \circ^* \beta$ with $\circ \in \{\mathsf{U}, \mathsf{S}, \mathsf{R}, \mathsf{T}\}$ and $* \in \{\sigma\uparrow, \sigma\downarrow\}$, we have

$$\delta_D(\psi, a) := \delta_D(\mathsf{Unf}(\psi), a).$$

We show that $\mathcal{A}_\varphi$ is very weak. Consider a partitioning $q_0, q_1, \ldots, q_n \in Q$ of the state set such that for all $i, j \in [n]$, if $q_j \in \mathsf{Sub}(q_i)$ then $j \leq i$. By the definition of the transition function, for all $i, j \in [n]$ and letters a, $D \subseteq \mathbb{D}$, and directions d, if (q_j, d) occurs in $\delta_D(q_i, a)$ then $j \leq i$.

Correctness In the remainder of the proof, we show the correctness of our construction. Let $\mathcal{A}_{\psi,i}$, for $\psi \in \mathsf{Sub}(\varphi)$ and $i \in \mathbb{N}$, denote the automaton $\mathcal{A}_\psi$ that starts in configuration (ψ, i) instead of $(\psi, 0)$. By induction on the structure of φ, we show that for all $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$, $\psi \in \mathsf{Sub}(\varphi)$ and $i \in \mathbb{N}$ the following holds.

$$\mathcal{A}_{\psi,i} \text{ accepts } (w, \rightsquigarrow) \quad \text{ if and only if } \quad (w, \rightsquigarrow, i) \models \psi.$$

This equivalence immediately implies $L^{\mathsf{nw}}(\mathcal{A}_\varphi) = L^{\mathsf{nw}}(\varphi)$, for any normalized NWTL formula φ Let $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$ be a nested word, $i \in \mathbb{N}$, and φ, α, and β normalized NWTL formulas.

Consider the case $\varphi = b$, for some $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$. Then $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff w_i satisfies φ iff $(w, \rightsquigarrow, i) \models \varphi$.

Consider the case $\varphi = \alpha \vee \beta$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff $\mathcal{A}_{\alpha,i}$ accepts $(w, \rightsquigarrow)$ or $\mathcal{A}_{\beta,i}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. By the induction hypothesis, either $(w, \rightsquigarrow, i) \models \alpha$ or $(w, \rightsquigarrow, i) \models \beta$. This is equivalent to $(w, \rightsquigarrow, i) \models \alpha \vee \beta$. The case for $\varphi = \alpha \wedge \beta$ is similar.

Consider the case $\varphi = \mathsf{O}\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff $\mathcal{A}_{\alpha,i+1}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. By the induction hypothesis, $(w, \rightsquigarrow, i+1) \models \alpha$. This is equivalent to $(w, \rightsquigarrow, i) \models \mathsf{O}\alpha$.

Consider the case $\varphi = \Theta\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff $i > 0$ and $\mathcal{A}_{\alpha,i-1}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. By the induction hypothesis, $i > 0$ and $(w, \rightsquigarrow, i-1) \models \alpha$. This is equivalent to $(w, \rightsquigarrow, i) \models \Theta\alpha$.

Consider the case $\varphi = \overline{\Theta}\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff either $i = 0$ (there is no -1-direction edge from the initial position 0) or $\mathcal{A}_{\alpha,i-1}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. By the induction hypothesis, either $i = 0$ or $(w, \rightsquigarrow, i-1) \models \alpha$. This is equivalent to $(w, \rightsquigarrow, i) \models \overline{\Theta}\alpha$.

Consider the case $\varphi = \mathsf{O}_\mu\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff i is a call with matching return j and $\mathcal{A}_{\alpha,j}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. Applying the induction hypothesis, we obtain $(w, \rightsquigarrow, i) \models \varphi$.

Consider the case $\varphi = \overline{\mathsf{O}}_\mu\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff either i has no matching return (there is no 2-direction edge from position i), or i is a call with matching return j and $\mathcal{A}_{\alpha,j}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. Applying the induction hypothesis, we obtain $(w, \rightsquigarrow, i) \models \varphi$.

Consider the case $\varphi = \mathsf{\Theta}_\mu\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff i is a return with matching call j and $\mathcal{A}_{\alpha,j}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. Applying the induction hypothesis, we obtain $(w, \rightsquigarrow, i) \models \varphi$.

Consider the case $\varphi = \overline{\mathsf{\Theta}}_\mu\alpha$. Then, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ iff either i has no matching call (there is no -2-direction edge from position i), or i is a return with matching call j and $\mathcal{A}_{\alpha,j}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. Applying the induction hypothesis, we obtain $(w, \rightsquigarrow, i) \models \varphi$.

Consider the case $\varphi = \alpha\, \mathsf{U}^{\sigma\uparrow}\, \beta$. We first show the *only if* direction. Assume $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$. For the sake of contradiction, suppose $(w, nwto, i) \not\models \varphi$. That is,

$$\text{there is no summary-up path } l_0 \dots l_k \text{ such that } l_k = i,\ (w, \rightsquigarrow, l_0) \models \beta, \text{ and } \forall j < k : (w, \rightsquigarrow, l_j) \models \alpha. \tag{$*$}$$

From $(*)$, it follows that $\mathcal{A}_{\beta,i}$ does not accept $(w, \rightsquigarrow)$. Thus, either $\mathcal{A}_{\alpha,i}$ accepts $(w, \rightsquigarrow)$, i is a call with matching return j, and $\mathcal{A}_{\alpha \mathsf{U}^{\sigma\uparrow},j}$ accepts $(w, \rightsquigarrow)$, or we have $\mathcal{A}_{\alpha,i}$ accepts $(w, \rightsquigarrow)$, i is not a call, and $\mathcal{A}_{\alpha \mathsf{U}^{\sigma\uparrow},i+1}$ accepts $(w, \rightsquigarrow)$, by the definition of the transition function. Consider the first case. By the induction hypothesis and $(*)$, we infer that $\mathcal{A}_{\beta,j}$ does not accepts $(w, \rightsquigarrow)$. Now, consider the second case. By the induction hypothesis and $(*)$, we infer that $\mathcal{A}_{\beta,i+1}$ does not accepts $(w, \rightsquigarrow)$. If we repeat our argumentation, we obtain an infinite summary-up path $l_0 l_1 \ldots \in \mathbb{N}^\omega$ with $l_0 = i$ and an infinite path $(\varphi, l_0)(\varphi, l_1)\ldots$ in the run of $\mathcal{A}_{\varphi,i}$ on $(w, \rightsquigarrow)$. This contradict the fact that all paths in the run of $\mathcal{A}_{\varphi,i}$ are accepting.

For the other direction, assume that $(w, \rightsquigarrow, i) \models \varphi$. That is, there is a summary-up path $l_0 \dots l_k$ such that $l_k = i$, $(w, \rightsquigarrow, l_0) \models \beta$, and $\forall j < k : (w, \rightsquigarrow, l_j) \models \alpha$. By the induction hypothesis, $\mathcal{A}_{\beta,l_k}$ accepts $(w, \rightsquigarrow)$ and for every $j < k$, we have $\mathcal{A}_{\alpha,l_j}$ accepts $(w, \rightsquigarrow)$. By the definition of the transition function, we infer that for every $j < k$, we have $\mathcal{A}_{\varphi,l_j}$ accepts $(w, \rightsquigarrow)$. Hence, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$.

The cases where φ is of the form $\alpha\, \mathsf{U}^{\sigma\downarrow}\, \beta$, $\alpha\, \mathsf{S}^{\sigma\uparrow}\, \beta$, or $\alpha\, \mathsf{S}^{\sigma\downarrow}\, \beta$ are similarly proven.

Consider the case $\varphi = \alpha\, \mathsf{R}^{\sigma\uparrow}\, \beta$. Assume $(w, \rightsquigarrow, i) \models \varphi$. By the definition of the semantics of φ and the induction hypothesis, we have

> for all summary-up paths of the form $l_0 \dots l_k$ such that $l_0 = i$, we either have $\mathcal{A}_{\beta,l_k}$ accepts $(w, \rightsquigarrow)$, or there is some $j < k$ such that $\mathcal{A}_{\alpha,l_j}$ accepts $(w, \rightsquigarrow)$. (i)

We show that (i) is equivalent to the fact (ii), namely, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$.

We show that (i) implies (ii). First note that for every $k \in \mathbb{N}$, there is a unique summary-up path $l_0 \dots l_k \in \mathbb{N}^*$ since there is exactly one shortest path between two positions. From this observation, we obtain the property

> either for all summary-up paths of the form $l_0 \dots l_k$ such that $l_0 = i$, we have $\mathcal{A}_{\beta,l_k}$ accepts $(w, \rightsquigarrow)$, or there is a summary-up path of the form $l_0 \dots l_k$ such that $l_0 = i$, $\mathcal{A}_{\alpha,l_k}$ accepts $(w, \rightsquigarrow)$, and for all j with $i \leq j \leq k$, we have $\mathcal{A}_{\beta,l_j}$ accepts $(w, \rightsquigarrow)$ (i')

that is equivalent to (i). Assume that the first case holds. Then there is an infinite sequence $l_0 l_1 \dots \in \mathbb{N}^\omega$ of positions such that for every $i \in \mathbb{N}$, we have l_{j+1} is the matched return of l_j if l_j is a call and otherwise, if l_j is not a call then l_{j+1} is $l_j + 1$. Consider the following run of $\mathcal{A}_{\varphi,k}$. Whenever $\mathcal{A}_{\varphi,k}$ arrives in a configuration (φ, l_j), for $j \in \mathbb{N}$, it moves to the configuration (β, l_j) and (φ, l_{j+1}), respecting the transition function. Since $\mathcal{A}_{\beta,l_j}$ accepts $(w, \rightsquigarrow)$, for all $j \in \mathbb{N}$, we have $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$ if the infinite path $(\varphi, l_0)(\varphi, l_1) \dots$ is accepting, which is indeed the case. Thus, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$. Assume that the second case holds. Let $l_0 \dots l_k$ be a summary-up path such that $l_0 = i$, $\mathcal{A}_{\alpha,l_k}$ accepts $(w, \rightsquigarrow)$, and for all j with $i \leq j \leq k$, we have $\mathcal{A}_{\beta,l_j}$ accepts $(w, \rightsquigarrow)$. Since $\mathcal{A}_{\beta,l_k}$ accepts $(w, \rightsquigarrow)$ and $\mathcal{A}_{\alpha,l_{k-1}}$ accepts $(w, \rightsquigarrow)$, it follows that by the definition of the transition function, $\mathcal{A}_{\varphi,l_{k-1}}$ accepts $(w, \rightsquigarrow)$. If we iterate this argumentation, we conclude that for all j with $j \leq k$, we have $\mathcal{A}_{\varphi,l_j}$ accepts $(w, \rightsquigarrow)$. Thus, $\mathcal{A}_{\varphi,i}$ accepts $(w, \rightsquigarrow)$.

It remains to show that (ii) implies (i). We show this implication by contraposition. Assume (i) does not hold, i.e.,

> there is a summary-up path of the form $l_0 \dots l_k$ with $l_0 = i$ such that $\mathcal{A}_{\beta,l_k}$ rejects $(w, \rightsquigarrow)$ and for all $j < k$, we have $\mathcal{A}_{\alpha,l_j}$ rejects $(w, \rightsquigarrow)$. $(\neg i)$

Let k be the smallest number such that $(\neg i)$ holds. So, $\mathcal{A}_{\beta,l_k}$ rejects $(w, \rightsquigarrow)$. By the definition of the transition function, $\mathcal{A}_{\varphi,l_k}$ rejects $(w, \rightsquigarrow)$. By $(\neg i)$, $\mathcal{A}_{\alpha,l_{k-1}}$ rejects $(w, \rightsquigarrow)$. Thus, $\mathcal{A}_{\varphi,l_{k-1}}$ rejects $(w, \rightsquigarrow)$, as well. If we repeat this argument, we infer that $\mathcal{A}_{\varphi,l_j}$ rejects $(w, \rightsquigarrow)$, for every $i \leq j < k$. Therefore, (ii) does not hold.

The cases where φ is of the form $\alpha\, \mathsf{R}^{\sigma\downarrow}\, \beta$, $\alpha\, \mathsf{T}^{\sigma\uparrow}\, \beta$, or $\alpha\, \mathsf{T}^{\sigma\downarrow}\, \beta$ are similarly proven.

Eventually 1-Wayness We show that $\mathcal{A}$ is eventually 1-way. For defining the partitioning of the state set Q, we need the following function that assigns weights to states.

$$\mathsf{weight}(q) := \begin{cases} 2|\mathsf{Sub}(q)| & \text{if } q \text{ is a future formula ,} \\ 2|\mathsf{Sub}(q)| + 1 & \text{otherwise.} \end{cases}$$

Let $n := 2|Q| + 1$. Let $(Q_i)_{i \leq n}$ be a partitioning of Q, where for $i \leq [n]$, we define $Q_i := \{q \mid \mathsf{weight}(q) = i\}$.

Let $p, q \in Q$, $D \subseteq \mathbb{D}$, $d \in D$, and $a \in \Sigma$ such that $(q, d) \in \delta_D(p, a)$. It suffices to show the following claim: if ($\mathsf{weight}(p)$ is even and $d \leq 0$) or ($\mathsf{weight}(p)$ is odd and $d \geq 0$) then $\mathsf{weight}(q) < \mathsf{weight}(p)$.

Consider the case, where p is a future formula. We have $\mathsf{weight}(p)$ is even. By the definition of the transition function, $d \geq 0$.

Consider the case p is not a future formula. We have $\mathsf{weight}(p)$ is odd. By the definition of the transition function, $d \leq 0$.

Finally, we translate the eventually 1-way V2ABA into an NWA using the alternation-elimination scheme from Chapter 3 with the complementation construction given in Theorem 5.5. ∎

5.2.2 Extensions of Nested Word Temporal Logic

In this section, we introduce the new logic NWPSL that extends NWTL by a regular expression layer. We show that NWPSL is more expressive than NWTL and present a translation to nested-word automata.

In [AAB+08], the authors show that NWTL has the same expressive power as first-order logic over nested words. Hence, not all regular properties over nested words can be expressed in NWTL. For instance, counting is not possible in NWTL. That is, specifications that require that exactly at each nth step, for

$n > 1$, some property should hold, cannot be expressed in NWTL. We make the intuition explicit in the next theorem that is based on Wolper's result for LTL in [Wol83].

Theorem 5.18 *For an atomic proposition p, the property "p holds at every even position" is not expressible in NWTL.* □

Proof We first show that NWTL over words without nesting edges is as expressive as LTL. Formally, let $\Sigma := 2^{\mathcal{P}}$ be an alphabet, where $\mathcal{P}$ denote the set of atomic propositions. For convenience, we write $[\varphi] := \{(w, i) \mid w \in \Sigma^\omega, i \in \mathbb{N}, \text{ and } (w, \rightsquigarrow, i) \models \varphi\}$ for the set of word models without nested edges of a NWTL formula φ. Note that an LTL formula is a special kind of a NWTL formula. We define the translation τ that translates an NWTL formula φ into an LTL formula $\tau(\varphi)$ such that $[\tau(\varphi)] = [\varphi]$.

The proof proceeds by induction on the structure of φ.

- φ is a propositional formula. We define $\tau(\varphi)$ as the formula obtained from φ by replacing the constants call and ret by ff. Obviously, $[\tau(\varphi)] = [\varphi]$.

- φ is of the form $\alpha \vee \beta$. We define $\tau(\varphi) := \tau(\alpha) \vee \tau(\beta)$. By induction hypothesis, we obtain $[\tau(\varphi)] = [\varphi]$. The case for $\alpha \wedge \beta$ and $\neg\alpha$ is similar.

- φ is of the form $\mathsf{O}\alpha$. We define $\tau(\varphi) := \mathsf{O}\tau(\alpha)$. By induction hypothesis, we obtain $[\tau(\varphi)] = [\mathsf{O}\tau(\alpha)] = \{(w, i) \mid w, i+1 \models \tau(\alpha)\} = \{(w, i) \mid w, i+1 \models \alpha\} = [\mathsf{O}\alpha]$. The case for $\Theta\alpha$ is similar.

- φ is of the form $\mathsf{O}_\mu\alpha$. We define $\tau(\varphi) := \mathsf{ff}$. By definition of the nested next operator, we have $[\varphi] = \emptyset = [\tau(\varphi)]$. The case for $\Theta_\mu\alpha$ is similar.

- φ is of the form $\alpha \,\mathsf{U}\, \beta$. We define $\tau(\varphi) := \tau(\alpha) \,\mathsf{U}\, \tau(\beta)$. By definition of the until operator and induction hypothesis, we have $[\tau(\varphi)] = \{(w, i) \mid w, i \models \tau(\varphi)\} = \{(w, i) \mid \exists k \geq i : w, k \models \tau(\beta) \text{ and } \forall 0 \leq j < k : w, j \models \tau(\alpha)\} = \{(w, i) \mid w, i \models \alpha \,\mathsf{U}\, \beta\} = [\varphi]$. The case for $\alpha \,\mathsf{S}\, \beta$ is similar.

- φ is of the form $\alpha \,\mathsf{U}^\sigma\, \beta$. Note that for words without nesting edges, $[\alpha \,\mathsf{U}^\sigma\, \beta] = [\alpha \,\mathsf{U}\, \beta]$, for any NWTL formula α and β. Hence, the case for $\alpha \,\mathsf{U}^\sigma\, \beta$ reduces to the case for $\alpha \,\mathsf{U}\, \beta$. The case for $\alpha \,\mathsf{S}\, \beta$ is similar.

In [Wol83], Wolper shows that LTL cannot express the property "p is true at every even position" in a word. Assume that there is a NWTL formula φ such that for every nested word $(w, \rightsquigarrow)$, we have $(w, \rightsquigarrow) \in L^{\mathsf{nw}}(\varphi)$ if and only if "p is true at every even position". Since each model of φ has no nested edges, we have $[\varphi] = [\tau(\varphi)]$. Note that $\tau(\varphi)$ is an LTL formula and describes all the words, where p holds at every even position. This contradicts Wolper's result. ∎

In the following, we present the new logic NWPSL that adds a regular layer to the logic NWTL. The syntax of an NWPSL formula over $\mathcal{P}$ is given by the following grammar.

$$\varphi ::= b \mid \ \diamond \varphi \mid \varphi \circ \varphi \mid r \diamondsuit\!\!\rightarrow \varphi \mid r \diamondsuit\!\!\!\!-\!\!\rightarrow \varphi \mid r \diamondsuit\!\!\rightarrow^{\sigma} \varphi \mid r \diamondsuit\!\!\!\!-\!\!\rightarrow^{\sigma} \varphi$$

where $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$, $\diamond$ is a unary NWTL operator, $\circ$ is a binary NWTL operator, and r is a regular expression (RE) as defined in Section 4.2.1.

For a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^{\omega}$ and a position $i \in \mathbb{N}$ in w, we define the semantics of the new operators as follows.

$$\begin{array}{lll}
(w, \rightsquigarrow, i) \models r \diamondsuit\!\!\rightarrow \varphi & \text{iff} & \exists k : w_{i..k} \models r \text{ and } (w, \rightsquigarrow, k) \models \varphi \\
(w, \rightsquigarrow, i) \models r \diamondsuit\!\!\!\!-\!\!\rightarrow \varphi & \text{iff} & \exists k : w_{k..i} \models r \text{ and } (w, \rightsquigarrow, k) \models \varphi \\
(w, \rightsquigarrow, i) \models r \diamondsuit\!\!\rightarrow^{\sigma} \psi & \text{iff} & \text{there is a summary path } l_0 \dots l_k \text{ such that} \\
 & & l_0 = i, w_{l_0} \dots w_{l_k} \models r, \text{ and } (w, \rightsquigarrow, l_k) \models \varphi \\
(w, \rightsquigarrow, i) \models r \diamondsuit\!\!\!\!-\!\!\rightarrow^{\sigma} \psi & \text{iff} & \text{there is a summary path } l_0 \dots l_k \text{ such that} \\
 & & l_k = i, w_{l_0} \dots w_{l_k} \models r, \text{ and } (w, \rightsquigarrow, l_0) \models \varphi
\end{array}$$

Note that we can express the property "p holds on every even position" by the NWPSL formula $p \wedge (p \,\square\!\!\rightarrow (p; \mathsf{tt}; p \diamondsuit\!\!\rightarrow \mathsf{tt}))$. So, NWPSL is more expressive than NWTL.

Next, we define new operators such that each NWPSL formula can be translated into *positive normal form*. For the two NWPSL formulas φ and ψ, we define the following operators.

- $r \,\square\!\!\rightarrow \varphi := \neg(r \diamondsuit\!\!\rightarrow \neg\varphi)$. Intuitively, the regular expression *triggers* the formula φ. That is, whenever the pattern r matches then the formula φ must hold.
- $r \boxminus\!\!\rightarrow \varphi := \neg(r \diamondsuit\!\!\!\!-\!\!\rightarrow \neg\varphi)$.

- $r \,\Box\!\!\rightarrow^{\sigma} \varphi := \neg(r \,\Diamond\!\!\rightarrow^{\sigma} \neg\varphi)$.
- $r \,\boxminus\!\!\rightarrow^{\sigma} \varphi := \neg(r \,\diamondminus\!\!\rightarrow^{\sigma} \neg\varphi)$.

For the translation of an NWPSL formula into an NWA, we transform an NWPSL formula into a *normalized* NWPSL formula, i.e., the NWPSL formula is in positive normal form and every fix-point operator over summary paths is replaced by the corresponding fix-point operators over summary-up and summary-down paths as described in Section 5.2.1.

We use our alternation-elimination scheme with a novel complementation construction from Section 5.1 to translate a normalized NWPSL formula into a nested-word automaton.

Theorem 5.19 *Every normalized NWPSL formula of size n can be translated into a language-equivalent 2ABA of size $\mathcal{O}(2n)$ and into a language-equivalent NWA of size $\mathcal{O}(2^{8n})$.* □

Proof Let φ be a NWPSL formula of size n. First, we translate this formula into a language-equivalent 2ABA. The construction combines the construction from normalized NWTL formulas to alternating automata given in Theorem 5.17 and the construction for the new operators $\Diamond\!\!\rightarrow$, $\diamondminus\!\!\rightarrow$, $\Box\!\!\rightarrow$, and $\boxminus\!\!\rightarrow$.

In this proof, we describe the construction for the new operators $\Diamond\!\!\rightarrow^{\sigma}$, $\diamondminus\!\!\rightarrow^{\sigma}$, $\Box\!\!\rightarrow^{\sigma}$, and $\boxminus\!\!\rightarrow^{\sigma}$. The construction is along the lines as in the proof for Theorem 4.17. In the following, we describe the construction of the automaton that is obtained from a formula of the form $r \,\Diamond\!\!\rightarrow^{\sigma} \psi \in \mathsf{Sub}(\varphi)$. It suffices to construct an NFA $\hat{\mathcal{A}}_r$ that accepts a finite nested word if the regular expression r matches the labels along the unique summary path between the first and the last position of the word. The NFA $\hat{\mathcal{A}}_r$ is built from two NFAs, $\mathcal{B}$ and $\mathcal{A}_r$.

First, we start with a construction of the NFA $\mathcal{B}$ that follows only summary paths in an input word. Let $\mathcal{B} := (\{p, q\}, \hat{\Sigma}, \eta, p, \{p, q\})$, where η is defined as depicted in Fig 5.2. Formally, for every $D \subseteq Q$ and $a \in \hat{\Sigma}$, we have

$$\eta_D(p, a) := \begin{cases} (p, 2) \vee (q, 2) \vee (q, 0) & \text{if } 2 \in D, \\ (p, 1) \vee (q, 1) \vee (q, 0) & \text{otherwise,} \end{cases}$$
$$\eta_D(q, a) := \{(q, 2) \mid 2 \in D\} \cup \{(q, 1) \mid -2 \notin D\}.$$

Intuitively, $\mathcal{B}$ stays in state p as long as it follows a summary-up path. When it changes to state q, it follows a summary-down path.

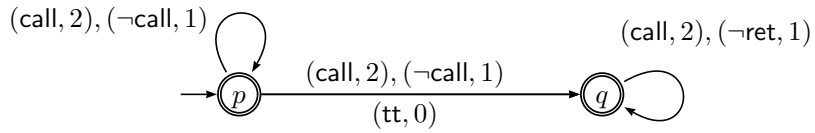


Figure 5.2: NFA that only follows summary paths.

Second, we construct an NFA $\mathcal{A}_r = (Q, \Sigma, \delta, q_I, F)$ such that $L^*(\mathcal{A}_r) = L^*(r)$. Then, we add new transitions along the 2-direction, for each transition along the 1-direction. Formally, we define $\mathcal{A}'_r := (Q, \hat{\Sigma}, \delta', q_I, F)$, where $\delta'(p, a) := \delta'(p, a) \cup \{(q, 2) \mid (q, 1) \in \delta'(p, a)\}$. That is, $\mathcal{A}'_r$ may use any path in the input word to obtain an accepting run. Finally, we define $\hat{\mathcal{A}}_r$ as the intersection of $\mathcal{A}'_r$ and the NFA $\mathcal{B}$ that only follows summary paths in the input word. Observe that the size of the automaton $\hat{\mathcal{A}}_r$ might be twice as big as the size of the regular expression r. The constructions for the operators $\diamondsuit\!\!\rightarrow^{\sigma}$, $\square\!\!\rightarrow^{\sigma}$, and $\boxminus\!\!\rightarrow^{\sigma}$ are analogous.

Overall, we obtain a 2ABA $\mathcal{A}_\varphi$ with at most $2n$ states that is language-equivalent to the normalized NWPSL formula φ. Furthermore, the 2ABA $\mathcal{A}_\varphi$ is eventually 1-way. This is shown the same way as is it shown for the 2ABA that is constructed in Theorem 4.17. Using our alternation-elimination scheme from Chapter 3 with the complementation construction from Theorem 5.2, we translate the 2ABA $\mathcal{A}_\varphi$ into a language-equivalent NWA with $\mathcal{O}(2^{8n})$ states. ∎

5.2.3 Linear-Time μ-Calculus

In this section we present new translations from the linear-time μ-calculus (μNWTL) over nested words to nested-word automata. The translations are based on our alternation-elimination constructions and are more modular than the constructions known from literature. Further, if the formula do not contain past operators or only in a restricted way, our translations yield smaller nested word automata. The logic μNWTL is introduced by Bozzelli in [Boz07] and—roughly speaking—extends the logic NWTL by a least and greatest fix-point operator. We can use this logic to specify any ω-regular property over nested words since any nested-word automaton can be translated into a language-equivalent μNWTL formula [Boz07].

We define the logic μNWTL. Let $\mathcal{V} = \{X, Y, \ldots\}$ be a set of monadic second-

order variables. The syntax of a μNWTL formula over the set of propositions $\mathcal{P}$ is given by the following grammar.

$$\varphi ::= b \mid X \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \mathsf{O}_d\varphi \mid \neg\mathsf{O}_d\mathsf{tt} \mid \mu X.\varphi \mid \nu X.\varphi,$$

where $b \in \mathcal{B}(\mathcal{P} \cup \{\mathsf{call}, \mathsf{ret}\})$, $X \in \mathcal{V}$, and $d \in \{-1, 1, c, \mu\}$ denote the directions of the O operator, namely, the previous position, next position, previous caller position, next matched position, respectively. In the following, we assume that in every μNWTL formula, every fixpoint quantifier binds a different variable. For a μNWTL formula φ, we write $\mathsf{Sub}(\varphi)$ for the set of sub-formulas of φ. For a variable $X \in \mathsf{Sub}(\varphi)$, we denote the fixpoint formula that defines X by $\mathsf{fp}(X)$. The *alternation depth* of a formula φ is the length of the longest sequence of variables $X_0X_1 \dots X_n \in (\mathsf{Sub}(\varphi) \cap \mathcal{V})^*$ such that for all $i < n$, we have (a) X_{i+1} occurs free in $\mathsf{fp}(X_i)$, and (b) $\mathsf{fp}(X_{i+1})$ and $\mathsf{fp}(X_i)$ have different fixpoint types.

To define the semantics of a μNWTL formula, we need the following definition. We call the function $\nu : \mathcal{V} \to 2^{\mathbb{N}}$ that maps a variable to a set of positions a *valuation*. For $X \in \mathcal{V}$ and $N \subseteq \mathbb{N}$, we write $\nu[X \mapsto N]$ for the valuation that maps X to the set N and behaves like ν on all other variables.

The semantics of a μNWTL formula is interpreted with respect to a nested word $(w, \rightsquigarrow) \in \hat{\Sigma}^\omega$, a valuation ν, and a position $i \in \mathbb{N}$.

$$\begin{array}{lcl}
[\![b]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid w_i \text{ satisfies } b\} \\
[\![X]\!]_\nu^{(w,\rightsquigarrow)} & := & \nu(X) \\
[\![\varphi \vee \psi]\!]_\nu^{(w,\rightsquigarrow)} & := & [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)} \cup [\![\psi]\!]_\nu^{(w,\rightsquigarrow)} \\
[\![\varphi \wedge \psi]\!]_\nu^{(w,\rightsquigarrow)} & := & [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)} \cap [\![\psi]\!]_\nu^{(w,\rightsquigarrow)} \\
[\![\mathsf{O}_1\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid i+1 \in [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)}\} \\
[\![\mathsf{O}_{-1}\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid i > 0 \text{ and } i-1 \in [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)}\} \\
[\![\mathsf{O}_c\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid j \text{ is the caller of } i \text{ and } j \in [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)}\} \\
[\![\mathsf{O}_\mu\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid (i,j) \in \rightsquigarrow_2 \text{ and } j \in [\![\varphi]\!]_\nu^{(w,\rightsquigarrow)}, \text{ for some } j \in \mathbb{N}\} \\
[\![\neg\mathsf{O}_1\mathsf{tt}]\!]_\nu^{(w,\rightsquigarrow)} & := & \mathbb{N} \\
[\![\neg\mathsf{O}_{-1}\mathsf{tt}]\!]_\nu^{(w,\rightsquigarrow)} & := & \mathbb{N} \setminus \{0\} \\
[\![\neg\mathsf{O}_c\mathsf{tt}]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid \text{there is no caller of } i\} \\
[\![\neg\mathsf{O}_\mu\mathsf{tt}]\!]_\nu^{(w,\rightsquigarrow)} & := & \{i \in \mathbb{N} \mid \text{there is no } j \in \mathbb{N} \text{ with } (i,j) \in \rightsquigarrow_2\} \\
[\![\mu X.\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \bigcap\{M \subseteq \mathbb{N} \mid [\![\varphi]\!]_{\nu[X\mapsto M]}^{(w,\rightsquigarrow)} \subseteq M\} \\
[\![\nu X.\varphi]\!]_\nu^{(w,\rightsquigarrow)} & := & \bigcup\{M \subseteq \mathbb{N} \mid [\![\varphi]\!]_{\nu[X\mapsto M]}^{(w,\rightsquigarrow)} \supseteq M\}
\end{array}$$

If φ is *closed*, i.e., every variable is bound then $[\![\varphi]\!]_\nu^{(w,\rightsquigarrow)}$ does not depend on the valuation ν. In that case, we just write $[\![\varphi]\!]^{(w,\rightsquigarrow)}$. The nested-word language of a closed μNWTL formula over $\mathcal{P}$ is $L^{\mathsf{nw}}(\varphi) := \{(w, \rightsquigarrow) \in \hat{\Sigma}^\omega \mid 0 \in [\![\varphi]\!]^{(w,\rightsquigarrow)}\}$.

Next, we will show how to translate a μNWTL formula into a nested word automaton. We need the following definition. An *extended nested word* is a nested word with an explicit relation for the caller edges. Formally, an extended nested word $(w, (\rightsquigarrow_d)_{d\in(\mathbb{D}\cup\{c\})})$ is a tuple such that $(w, (\rightsquigarrow_d)_{d\in\mathbb{D}}) \in \hat{\Sigma}^\omega$ is a nested word and $\rightsquigarrow_c = \{(i, j) \in \mathbb{N}^2 \mid i \text{ is a caller of } j\}$. We write $\tilde{\Sigma}^\omega$ for the set of all extended nested words. For an automaton $\mathcal{A}$, we write

$$L^{\mathsf{xnw}}(\mathcal{A}) := \{(w, \rightsquigarrow) \in \tilde{\Sigma}^\omega \mid \text{ there is an accepting run of } \mathcal{A} \text{ on } (w, \rightsquigarrow)\}.$$

As shown in [Boz07], every μNWTL formula can be translated into a 2APA over extended nested words that accepts the language that is described by the formula.

Theorem 5.20 *Every μNWTL formula φ of size n and alternation depth k can be translated into a 2APA of size $\mathcal{O}(n)$ and index k and $\{(w, (\rightsquigarrow_d)_{d\in\mathbb{D}}) \mid (w, \rightsquigarrow) \in L^{\mathsf{xnw}}(\mathcal{A})\} = L^{\mathsf{nw}}(\varphi)$. If φ contains no past operators, the automaton is 1-way.* □

It is straight-forward to simulate the behavior of a 2APA $\mathcal{A}$ over extended nested words by a 2APA $\mathcal{B}$ over nested words such that $\{(w, (\rightsquigarrow_d)_{d\in\mathbb{D}}) \mid (w, \rightsquigarrow) \in L^{\mathsf{xnw}}(\mathcal{A})\} = L^{\mathsf{nw}}(\mathcal{B})$. Consider a transition $\delta_D(p, a) \in \mathcal{B}^+(Q \times \{c, -2, -1, 0, 1, 2\})$ of an 2APA $\mathcal{A}$ that contains the tuple (q, c), for $D \subseteq \mathbb{D} \cup \{c\})$ and $a \in \hat{\Sigma}$. That is, if the automaton $\mathcal{A}$ is in state p and goes to state q, it moves its read-only head back to the caller position of the current position. According to the definition of caller positions, we can simulate this move in $\mathcal{B}$ by using one extra states q' for every tuple (q, c) that occurs in some transition of $\mathcal{A}$. Instead of directly moving to state q, the automaton moves to state q' and changes its read-only head position depending on the current read letter. If the current letter is a call, it decreases its head position by one. Otherwise, it remains at the same head position. Being in state q', the automaton $\mathcal{B}$ moves back along a well-matched sub-word until it reaches the caller position. The simulation is depicted in Figure 5.3. To sum up, we obtain the following theorem.

Theorem 5.21 *Every μNWTL formula φ of size n and fix-point depth k can be translated into a language-equivalent 2APA of size $\mathcal{O}(n)$ and index k. If φ contains no past operators, the automaton is 1-way.* □

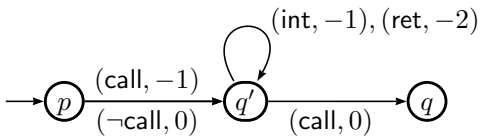


Figure 5.3: Simulation of the direct *back-to-caller* move

Consider a 2APA $\mathcal{A}_\varphi$ that represents the nested-word language of a μNWTL formula φ. Our alternation-elimination scheme with the complementation constructions from Section 5.1.3 provides three novel constructions to translate $\mathcal{A}_\varphi$ into a language-equivalent NWA depending on whether $\mathcal{A}_\varphi$ is 1-way, eventually 1-way, or none of both.

Theorem 5.22 *Every μNWTL formula φ of size n and alternation depth k can be translated into a language-equivalent NWA of size $2^{\mathcal{O}((nk)^2)}$. If the automaton $\mathcal{A}_\varphi$ from Theorem 5.21 is eventually 1-way then the NWA is of size $2^{\mathcal{O}(nk \log n)}$.* □

Note that the construction in Theorem 5.22 improves over the state-of-the-art construction given in [Boz07] to translate μNWTL formulas to NWAs. Consider a μNWTL formula φ. If the formula φ has no past operators at all, the automaton $\mathcal{A}_\varphi$ from Theorem 5.21 is 1-way and thus, the size of the resulting NWA is bound by $2^{\mathcal{O}(nk \log n)}$ instead of $2^{\mathcal{O}((nk)^2)}$ when using Bozzelli's construction. If the formula φ has only restricted past operators like those known from NWTL or NWPSL, the automaton $\mathcal{A}_\varphi$ from Theorem 5.21 is eventually 1-way and thus, the size of the resulting NWA is also bound by $2^{\mathcal{O}(nk \log n)}$. To check whether $\mathcal{A}_\varphi = (Q, \hat{\Sigma}, \delta, q_I, F)$ is eventually 1-way, we can check that no component in the strongly-connected component graph of the graph

$$(Q, \{(p, d, q) \mid (q, d) \text{ occurs in } \delta_D(p, a), \text{ for some } D \subseteq \mathbb{D}, \text{ and } a \in \hat{\Sigma}\})$$

has not both, a negatively and a positively labeled edge. Finally, if $\mathcal{A}_\varphi$ is not eventually 1-way, we obtain the same bound as in Bozzelli's construction. However, our construction is more modular and hence, easier to understand, to implement, and to optimize.

Chapter 6

Conclusion

We have presented an alternation-elimination scheme that reduces the problem of removing the alternation of an alternating automaton to the problem of complementing an existential automaton. We point out that the existential automaton inspects just a single path in the input graph and its construction is remarkably simple. We have also provided instances of this construction scheme by presenting complementation constructions for various classes of existential automata. Some of these instances clarify, simplify, or improve existing ones; some of these instances are novel. We observe that our presented complementation constructions consist of two main building blocks and the constructions are composed by canonically combining the appropriate constructions for each building block. In the first building block, we use the well-known (1-way or 2-way) subset construction to represent all runs of the input automaton. For the second one, we use a simple (focus, breakpoint, or ranking) construction to recognize a run of the input automaton that does not fulfill its acceptance condition. From this perspective, translations for 2-way alternating automata, which are considered to be difficult to understand, become as simple as common, standard constructions.

Moreover, we have extended various temporal logics by adding past operators to their corresponding future counterparts and have shown how to translate these logics into nondeterministic automata using novel instances of our alternation-elimination scheme. Our translations show that the additional cost for handling the past operators is surprisingly small. For common symbolic model checkers like NuSMV, the resulting nondeterministic automata are encoded as transition systems and in that case, there is no additional cost at all.

Let us conclude with an outlook on future work that is based on the results

in this thesis. It remains an open question to which extent our scheme generalizes to automata that process more general graph structures. Since our automata models are restricted to a finite alphabet and a finite set of discreet transitions, they are only used for representing properties over graphs whose nodes are labeled by a finite alphabet and whose set of edges between two nodes are finite. These kind of properties might not suffice to represent specifications of more general systems. For instance, continuous behavior of timed systems [Dil89, AD90] or linear-hybrid systems [ACHH93] are represented by graphs whose nodes are connected by infinitely many edges. Another example is the behavior of systems that should be monitored. Representing those behavior by graphs over a finite alphabet might be a limiting factor for describing relevant system properties. When considering more general graph structures, we are also interested in the question how the classes of alternating automata over those extended graphs look like and how corresponding logics over those graph structures can be defined in a reasonable way. For instance, it should be possible to give an automata-theoretic framework for the logic and monitoring technique presented in [HV09].

We have presented several complementation constructions for automata over words and nested words and have provided upper bounds. However, for many of these constructions, we do not provide tight lower bounds that would establish the optimality of our constructions.

We have introduced the temporal logic NWPSL and showed that it is more expressive than NWLTL whereas formulas of size n in this logic can be translated into NWAs of size $2^{\mathcal{O}(n)}$. However, it is still unknown whether NWPSL suffices to express all regular properties over nested words. If NWPSL cannot describe all regular properties, we would like to know how to extend NWPSL such that it becomes expressive enough while preserving the property that formulas of size n can be translated into NWAs of size $2^{\mathcal{O}(n)}$.

Finally, we have proposed the temporal logic PPSL, which extends the IEEE standard PSL with past operators. We analyzed its complexity and our results show that PPSL and PSL are similarly related as PLTL and LTL with respect to expressiveness, succinctness, and the computational complexity of the satisfiability and the model checking problems. It remains to be seen whether the advantages of PPSL over PSL pay off in practice. The presented translation for PPSL into 1NBAs shows that the additional cost for handling past operators is small and should not be a burden in implementing PPSL in system verification. For using PPSL with the explicit model checker

SPIN [Hol04], techniques like formula rewriting and simulation-based reductions have to be implemented to optimize the sizes of the resulting 1NBAs, see [CRT07, EWS05, FW05, EH00, SB00]. Our preliminary experience with a prototype implementation for the symbolic model checker NuSMV are promising.

Bibliography

[AAB+08] Rajeev Alur, Marcelo Arenas, Pablo Barceló, Kousha Etessami, Neil Immerman, and Leonid Libkin. First-order and temporal logics for nested words. *Logical Methods in Computer Science*, 4(4), 2008.

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AEM04] Rajeev Alur, Kousha Etessami, and P. Madhusudan. A temporal logic of nested calls and returns. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lecture Notes in Computer Science*, pages 467–481. Springer, 2004.

[AFF+02] Roy Armoni, Limor Fix, Alon Flaisher, Rob Gerth, Boris Ginsburg, Tomer Kanza, Avner Landver, Sela Mador-Haim, Eli Singerman, Andreas Tiemeyer, Moshe Y. Vardi, and Yael Zbar. The ForSpec temporal logic: A new temporal property-specification language. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2280, pages 296–211, 2002.

[AM04] Rajeev Alur and P. Madhusudan. Visibly pushdown languages. In

Symposium on the Theory of Computing (STOC), pages 202–211. ACM, 2004.

[AM09] Rajeev Alur and P. Madhusudan. Adding nesting structure to words. *Journal of the ACM*, 56(3):1–43, 2009.

[BB89] Behnam Banieqbal and Howard Barringer. Temporal logic with fixed points. In *Temporal Logic in Specification*, volume 398 of *Lecture Notes in Computer Science*, pages 62–74. Springer, 1989.

[BBDE+97] Ilan Beer, Shoham Ben-David, Cindy Eisner, Daniel Geist, Leonid Gluhovsky, Tamir Heyman, Avner Landver, P. Paanah, Yoav Rodeh, G. Ronin, and Yaron Wolfsthal. RuleBase: Model checking at IBM. In *Computer Aided Verification (CAV)*, volume 1254 of *Lecture Notes in Computer Science*, pages 480–483. Springer, 1997.

[BCPR07] Roderick Bloem, Alessandro Cimatti, Ingo Pill, and Marco Roveri. Symbolic implementation of alternating automata. *International Journal of Foundations of Computer Science*, 18(4):727–743, 2007.

[BDBF+05] Shoham Ben-David, Roderick Bloem, Dana Fisman, Andreas Griesmayer, Ingo Pill, and Sitvanit Ruah. Automata construction algorithms optimized for PSL. Technical report, The Prosyd Project, http://www.prosyd.org, 2005.

[BFH05] D. Bustan, D. Fisman, and J. Havlicek. Automata construction for PSL. Technical report, Computer Science and Applied Mathematics, The Weizmann Institute of Science, Israel, 2005.

[BHSV+96] Robert K. Brayton, Gary D. Hachtel, Alberto L. Sangiovanni-Vincentelli, Fabio Somenzi, Adnan Aziz, Szu-Tsung Cheng, Stephen A. Edwards, Sunil P. Khatri, Yuji Kukimoto, Abelardo Pardo, Shaz Qadeer, Rajeev K. Ranjan, Shaker Sarwary, Thomas R. Shiple, Gitanjali Swamy, and Tiziano Villa. VIS: A system for verification and synthesis. In *Computer Aided Verification (CAV)*, volume 1102 of *Lecture Notes in Computer Science*, pages 428–432. Springer, 1996.

[BJW05] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(3):614–633, 2005.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[Boz07] Laura Bozzelli. Alternating automata and a temporal fixpoint calculus for visibly pushdown languages. In *International Conference on Concurrency Theory (CONCUR)*, volume 4703 of *Lecture Notes in Computer Science*, pages 476–491. Springer, 2007.

[Boz08] Laura Bozzelli. The complexity of CaRet + chop. In *TIME Symposium / Workshop*, pages 23–31. IEEE Computer Society, 2008.

[Boz09] Laura Bozzelli. CaRet with forgettable past. *Electronic Notes in Theoretical Computer Science*, 231:343–361, 2009.

[CCG+02] Alessandro Cimatti, Edmund M. Clarke, Enrico Giunchiglia, Fausto Giunchiglia, Marco Pistore, Marco Roveri, Roberto Sebastiani, and Armando Tacchella. NuSMV 2: An opensource tool for symbolic model checking. In *Computer Aided Verification (CAV)*, volume 2404 of *Lecture Notes in Computer Science*, pages 359–364. Springer, 2002.

[CE82] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1982.

[CGLV09] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. An automata-theoretic approach to regular XPath. In *International Workshop on Database Programming Languages*, volume 5708 of *Lecture Notes In Computer Science*, pages 18–35. Springer, 2009.

[CGP99] E. M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[CRS04] Alessandro Cimatti, Marco Roveri, and Daniel Sheridan. Bounded verification of Past LTL. In *Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, volume 3312, pages 245–259, 2004.

[CRST06] Alessandro Cimatti, Marco Roveri, Simone Semprini, and Stefano Tonetta. From PSL to NBA: a modular symbolic encoding. In *Proceedings of the 6th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 125–133. IEEE Computer Society, 2006.

[CRT07] Alessandro Cimatti, Marco Roveri, and Stefano Tonetta. Syntactic optimizations for PSL verification. In *Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 4424 of *Lecture Notes in Computer Science*, pages 505–518. Springer, 2007.

[DG07] Volker Diekert and Paul Gastin. First-order definable languages. In J. Flum, E. Grädel, and T. Wilke, editors, *Logic and Automata: History and Perspectives*, volume 2 of *Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2007.

[Dil89] David L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.

[DKL09] Christian Dax, Felix Klaedtke, and Martin Lange. On regular temporal logics with past. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 5556 of *Lecture Notes In Computer Science*, pages 175–187. Springer, 2009.

[DL05] Christian Dax and Martin Lange. Game over: The foci approach to LTL satisfiability and model checking. In *Electronic Notes in Theoretical Computer Science (ENTCS)*, volume 119 of *Electronic Notes in Theoretical Computer Science*, pages 33–49. Elsevier, 2005.

[DS02] Stéphane Demri and Ph. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. *Information and Computation*, 174(1):84–103, 2002.

[EH86] E. Allen Emerson and Joseph Y. Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151–178, 1986.

[EH00] Kousha Etessami and Gerard J. Holzmann. Optimizing Büchi automata. In *International Conference on Concurrency Theory (CONCUR)*, volume 1877 of *Lecture Notes In Computer Science*, pages 153–167. Springer, 2000.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 368–377. IEEE Computer Society, 1991.

[EWS05] Kousha Etessami, Thomas Wilke, and Rebecca A. Schuller. Fair simulation relations, parity games, and state space reduction for Büchi automata. *SIAM Journal on Computing*, 34(5):1159–1175, 2005.

[Fri03] Carsten Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating büchi automata. In *Conference on Implementation and Application of Automata (CIAA)*, volume 2759 of *Lecture Notes in Computer Science*, pages 35–48. Springer, 2003.

[FW05] Carsten Fritz and Thomas Wilke. Simulation relations for alternating Büchi automata. *Theoretical Computer Science*, 338(1-3):275–314, 2005.

[FW06] Carsten Fritz and Thomas Wilke. Simulation relations for alternating parity automata and parity games. In *Developments in Language Theory*, volume 4036 of *Lecture Notes in Computer Science*, pages 59–70. Springer, 2006.

[GO01] Paul Gastin and Denis Oddoux. Fast LTL to Büchi automata translation. In *Computer Aided Verification (CAV)*, volume 2102

of *Lecture Notes in Computer Science*, pages 53–65. Springer, 2001.

[GO03] Paul Gastin and Denis Oddoux. LTL with past and two-way very-weak alternating automata. In *Mathematical Foundations of Computer Science (MFCS)*, volume 2747 of *Lecture Notes in Computer Science*, pages 439–448. Springer, 2003.

[HKT00] David Harel, Dexter Kozen, and Jerzy Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Hol04] Gerard J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison-Wesley, 2004.

[HT99] Jesper G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1-3):187–207, 1999.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

[HV09] Sylvain Hallé and Roger Villemaire. Browser-based enforcement of interface contracts in web applications with BeepBeep. In *Computer Aided Verification (CAV)*, volume 5643 of *Lecture Notes In Computer Science*, pages 648–653. Springer, 2009.

[Jut97] Charanjit S. Jutla. Determinization and memoryless winning strategies. *Information and Computation*, 133(2):117–134, 1997.

[Koz82] Dexter Kozen. Results on the propositional μ-calculus. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 348–359, 1982.

[KPV01] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. Extended temporal logic revisited. In *International Conference on Concurrency Theory (CONCUR)*, volume 2154 of *Lecture Notes in Computer Science*, pages 519–535. Springer, 2001.

[KV01] Orna Kupferman and Moshe Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.

[KV05] Orna Kupferman and Moshe Y. Vardi. Complementation constructions for nondeterministic automata on infinite words. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 3440, pages 206–221. Springer, 2005.

[Lan07] Martin Lange. Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In *Proceedings of the 18th International Conference on Concurrency Theory (CONCUR)*, volume 4703, pages 90–104, 2007.

[LMS02] François Laroussinie, Nicolas Markey, and Ph. Schnoebelen. Temporal logic with forgettable past. In *Logic in Computer Science (LICS)*, pages 383–392. IEEE Computer Society, 2002.

[LPZ85] Orna Lichtenstein, Amir Pnueli, and Lenore D. Zuck. The glory of the past. In *Logic of Programs*, volume 193 of *Lecture Notes in Computer Science*, pages 196–218. Springer, 1985.

[LS01] Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *Logic in Computer Science (LICS)*, pages 357–365. IEEE Computer Society, 2001.

[LS07] Martin Leucker and César Sánchez. Regular linear temporal logic. In *International Colloquium on Theoretical Aspects of Computing*, volume 5944 of *Lecture Notes in Computer Science*, pages 291–305. Springer, 2007.

[LT00] Christof Löding and Wolfgang Thomas. Alternating automata and logics over infinite words. In *IFIP International Conference on Theoretical Computer Science (IFIP TCS)*, volume 1872 of *Lecture Notes in Computer Science*, pages 521–535. Springer, 2000.

[Mar03] Nicolas Markey. Temporal logic with past is exponentially more succinct. *Bulletin of the EATCS*, 79:122–128, 2003.

[McM92] Kenneth L. McMillan. *Symbolic Model Checking*. PhD thesis, Carnegie Mellon University, 1992.

[McM99] Kenneth L. McMillan. The SMV language. Technical report, Cadence Berkeley Labs, March 1999.

[MH84] Satoru Miyano and Takeshi Hayashi. Alternating finite automata on omega-words. *Theoretical Computer Science*, 32:321–330, 1984.

[MS87] D. Muller and P. Schupp. Alternating automata on infinite trees. *Theoretical Computer Science*, 54:267–276, 1987.

[MSS92] David E. Muller, Ahmed Saoudi, and Paul E. Schupp. Alternating automata, the weak monadic theory of trees and its complexity. *Theoretical Computer Science*, 97(2):233–244, 1992.

[Pnu77] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57. IEEE Computer Society, 1977.

[Psl05] IEEE standard for property specification language (PSL). IEEE Std 1850TM, Oct. 2005.

[PZ06] Amir Pnueli and Aleksandr Zaks. PSL model checking and run-time verification via testers. In *Proceedings of the 14th International Symposium on Formal Methods (FM)*, volume 4085, pages 573–586, 2006.

[QS82] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in CESAR. In *Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer, 1982.

[Roh97] Gareth Scott Rohde. *Alternating automata and the temporal logic of ordinals*. PhD thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1997.

[RS59] M. O. Rabbin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[SB00] Fabio Somenzi and Roderick Bloem. Efficient Büchi automata from LTL formulae. In *Computer Aided Verification (CAV)*, volume 1855 of *Lecture Notes in Computer Science*, pages 248–263. Springer, 2000.

[She59] John C. Shepherdson. The reduction of two-way automata to one-way automata. *IBM Journal of Research and Development*, 3:198–200, April 1959.

[SL10] César Sánchez and Martin Leucker. Regular linear temporal logic with past. In *Verification, Model Checking and Abstract Interpretation (VMCAI)*, volume 5944 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2010.

[Var88] Moshe Y. Vardi. A temporal fixpoint calculus. In *Symposium on Principles of Programming Languages (POPL)*, pages 250–259. ACM, 1988.

[Var89] Moshe Y. Vardi. A note on the reduction of two-way automata to one-way automata. *Information Processing Letters*, 30(5):261–264, 1989.

[Var98] Moshe Y. Vardi. Reasoning about the past with two-way automata. In *International Colloquium on Automata, Languages and Programming (ICALP)*, volume 1443 of *Lecture Notes in Computer Science*, pages 628–641. Springer, 1998.

[Var07] M. Vardi. Automata-theoretic model checking revisited. In *Proceedings of the 8th International Conference on erification, Model Checking, and Abstract Interpretation (VMCAI)*, volume 4349, pages 137–150. Springer, 2007.

[VW86] Moshe Y. Vardi and Pierre Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Logic in Computer Science (LICS)*, pages 332–344. IEEE Computer Society, 1986.

[VW07] Moshe Y. Vardi and Thomas Wilke. Automata: From logics to algorithms. In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata: History and Perspectives*, number 2 in

Texts in Logic and Games, pages 629–736. Amsterdam University Press, Amsterdam, 2007.

[Wol83] Pierre Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1/2):72–99, 1983.

[Zie98] Wieslaw Zielonka. Infinite games on finitely coloured graphs with applications to automata on infinite trees. *Theoretical Computer Science*, 200(1-2):135–183, 1998.

Index

Curriculum Vitae

	Born on 9th of December 1979 in Aachen, Germany.
09/1990–07/1999	Abitur, Ignaz–Günther–Gymnasium Rosenheim, Germany
08/1999–06/2000	Civil service at Caritas Bad Aibling, Germany
10/2000–03/2006	Diplom-Informatiker Univ., Ludwig–Maximilians–Universität München, Germany
10/2003–05/2004	Visiting student at University of Edinburgh, Scotland
04/2006–08/2010	Ph.D. student in the Information Security Group, Department of Computer Science, ETH Zurich, Switzerland