On the Translation of Automata to Linear Temporal Logic*

Udi Boker[1], Karoliina Lehtinen[2,✉], and Salomon Sickert[3**]

[1] Reichman University, Herzliya, Israel
udiboker@idc.ac.il
[2] CNRS, Aix-Marseille University and University of Toulon, LIS, Marseille, France
lehtinen@lis-lab.fr
[3] The Hebrew University, Jerusalem, Israel
salomon.sickert@mail.huji.ac.il

Abstract While the complexity of translating future linear temporal logic (LTL) into automata on infinite words is well-understood, the size increase involved in turning automata back to LTL is not. In particular, there is no known elementary bound on the complexity of translating deterministic ω-regular automata to LTL.

Our first contribution consists of tight bounds for LTL over a unary alphabet: alternating, nondeterministic and deterministic automata can be exactly exponentially, quadratically and linearly more succinct, respectively, than any equivalent LTL formula. Our main contribution consists of a translation of general counter-free deterministic ω-regular automata into LTL formulas of double exponential temporal-nesting depth and triple exponential length, using an intermediate Krohn-Rhodes cascade decomposition of the automaton. To our knowledge, this is the first elementary bound on this translation. Furthermore, our translation preserves the acceptance condition of the automaton in the sense that it turns a looping, weak, Büchi, coBüchi or Muller automaton into a formula that belongs to the matching class of the syntactic future hierarchy. In particular, it can be used to translate an LTL formula recognising a safety language to a formula belonging to the safety fragment of LTL (over both finite and infinite words).

Keywords: Linear temporal logic · Automata · Cascade decomposition

1 Introduction

Linear Temporal Logic with only future temporal operators (from here on LTL) and ω-regular automata, whether deterministic, nondeterministic or alternating, are both well-established formalisms to describe properties of infinite-word languages. LTL is popular in formal verification and synthesis due to its simple

* The omitted proofs of this chapter can be found in the full version [5].

** Salomon Sickert is supported by the Deutsche Forschungsgemeinschaft (DFG) under project number 436811179.

P. Bouyer and L. Schröder (Eds.): FoSSaCS 2022, LNCS 13242, pp. 140–160, 2022.
https://doi.org/10.1007/978-3-030-99253-8_8

syntax and semantics. Yet, while properties might be convenient to define in LTL, most verification and synthesis algorithms eventually compile LTL formulas into ω-regular automata. The expressiveness of both these key formalisms, as well as translations from LTL to automata of various types, are well understood. Here, we consider the converse translations, which, in comparison, have received less attention: up till now, no elementary upper bound on the size blow-up of going from automata to LTL was known.

Regarding expressive power, deterministic Muller automata, nondeterministic Büchi automata, and weak alternating automata recognise all ω-regular languages [21,40]. LTL-definable languages (surveyed in [13]) are a strict subset thereof, also defined by first-order logic, star-free regular expressions, aperiodic monoids, counter-free automata, and very weak alternating automata. As for succinctness, nondeterministic and alternating automata can be exponentially and double-exponentially more succinct than deterministic automata, respectively. Determinisation in particular has precise bounds [32,35,24,36,12,3].

The succinctness of various representations of LTL-definable languages is less clear: effective translations between the different models are far from straightforward, and their complexity is sometimes uncertain. In particular, to the best of our knowledge, up to now there has been no elementary bound even on the translation of deterministic counter-free automata, arguably the simplest automata model for this class of languages, into LTL formulas. (Considering LTL with both future and past temporal operators, there is a double-exponential upper bound on the length of the formula [26][4].) The complexity of obtaining a deterministic counter-free automaton from a nondeterministic one is also, to the best of our knowledge, open.

We study the complexity of translating automata to LTL (equivalently, to very weak alternating automata), considering formula length, size, and nesting depth of temporal operators.

We begin (Section 3), as a warm-up, with the unary alphabet case on finite words. We show that the size-blow up involved in translating deterministic, non-deterministic and alternating automata to LTL, when possible, is linear, quadratic and exponential, respectively, and these bounds are tight. In contrast, going from LTL to alternating, nondeterministic and deterministic automata is linear, exponential and double-exponential, respectively [33,41,19].

The case of non-unary alphabets is much more difficult. We provide a translation of counter-free deterministic ω-regular automata (with any acceptance condition) into LTL formulas with double exponential depth and triple exponential length. Our translation uses an intermediate Krohn-Rhodes *reset cascade decomposition* (wreath product) of deterministic automata, which is a deterministic automaton built from simple components.

Our main technical contribution consists of a translation of a reset cascade into an LTL formula of depth linear and length singly exponential in the number of cascade configurations. Combining this with Eilenberg's Holonomy translation of a semigroup into a cascade [14, Corollary II.7.2] and Pnueli and Maler's adapt-

[4] See Remark 1 on whether the upper bound in [26] is single or double exponential.

ation of it to automata [26, Theorem 3] (see Remark 1), we obtain a translation of counter-free deterministic ω-regular automata into LTL formulas of double exponential depth and triple exponential length. Our construction preserves the acceptance condition of the automaton in the sense that it turns a Büchi-looping, coBüchi-looping, weak, Büchi or coBüchi automaton into a formula that belongs to the matching class of the syntactic future hierarchy (see Definition 1 and [8]).

Related work

Finite words. While LTL is usually interpreted over infinite words, it also admits finite-word semantics that coincide with the finite word version of the other equivalent formalisms. The equivalence between FO and star-free languages on finite words is due to McNaughton and Papert [31]. Cohen, Perrin and Pin [10] used the Krohn-Rhodes decomposition to characterise the expressive power of LTL with only **X** and **F** (eventually), but do not provide bounds on the size trade-off between the different models. Wilke [42] gives a double-exponential translation from counter-free DFA to LTL. More recently, Bojańczyk provided an algebraically flavoured adaptation of Wilke's proof [2, Section 2.2.2].

Infinite words. With substantial effort over several decades, the above techniques have been extended to infinite words using intricate tools with opaque complexities. Ladner [22] and Thomas [38,39] for example extended the equivalence of star-free regular expressions and FO to infinite words, while the ω-extension of the equivalence with aperiodic languages is due to Perrin [34]. The correspondence with LTL is due to Kamp [18] and Gabbay, Pnueli, Shelah and Stavi [16]. Diekert and Gastin's survey [13] provides an algebraic translation into LTL via ω-monoids while Cohen-Chesnot gives a direct algebraic proof of the equivalence of star-free ω-regular expressions and LTL [11]. Wilke takes an automata-theoretic approach, using backward deterministic automata [43,44]. However, none of the above address the complexity of the transformations. Zuck's dissertation [46] gives a translation of star-free regular expressions into LTL, with at least non-elementary complexity. Subsequently, Chang, Mana and Pneuli [8] use Zuck's results to show that the levels of their hierarchy of future temporal properties coincide with syntactic fragments of LTL. Sickert and Esparza [37] gave an exponential translation of any LTL formula into level Δ_2 of this hierarchy.

2 Preliminaries

Languages. An alphabet Σ, of size $|\Sigma|$, is a finite set of letters. Σ^*, Σ^+, and Σ^ω denote the sets of finite, nonempty finite, and infinite words over Σ, respectively. A language of finite or infinite words is a subset of Σ^* or Σ^ω, respectively. We write $[i..j]$ and $[i..j)$, with integers $i \leq j$, for the sets $\{i, i+1, \ldots, j\}$ and $\{i, i+1, \ldots, j-1\}$, respectively. For a word $w = \sigma_0 \cdot \sigma_1 \cdots$, we write $|w|$ for its length (∞ if w is infinite), $w[i]$ for σ_i, $w_{[i..j]}$ and $w_{[i..j)}$ for its corresponding infixes ($w_{[i..i)}$ is the empty word), and $w_{[i..]}$ for its (finite or infinite) suffix $\sigma_i \cdot \sigma_{i+1} \cdots$.

Linear Temporal Logic (LTL). Let AP be a finite set of atomic propositions. LTL formulas are constructed from the constant **true**, atomic propositions $a \in AP$, the connectives $\neg$ (negation) and $\wedge$ (and), and the temporal operators **U** (until) and **X** (next). Their semantics are given by a satisfiability relation $\models$ between finite or infinite words $w \in (2^{AP})^+ \cup (2^{AP})^\omega$, and a formula φ inductively as follows:

$$\begin{array}{llll} w \models \mathbf{true} & & w \models a & \textit{iff } a \in w[0] \\ w \models \neg\varphi & \textit{iff } w \not\models \varphi & w \models \varphi \wedge \psi & \textit{iff } w \models \varphi \textit{ and } w \models \psi \\ w \models \mathbf{X}\varphi & \textit{iff } |w| > 1 \textit{ and } w_{[1..]} \models \varphi & & \\ w \models \varphi\mathbf{U}\psi & \textit{iff } \exists i \in [0..|w|).\ w_{[i..]} \models \psi \textit{ and } \forall j \in [0..i).\ w_{[j..]} \models \varphi & & \end{array}$$

We also use the common shortcuts $\mathbf{false} \coloneqq \neg\mathbf{true}$, $\varphi \vee \psi \coloneqq \neg((\neg\varphi) \wedge (\neg\psi))$, $\mathbf{F}\varphi \coloneqq \mathbf{true}\mathbf{U}\varphi$, $\mathbf{G}\varphi \coloneqq \neg\mathbf{F}\neg\varphi$, and $\psi_1\mathbf{R}\psi_2 \coloneqq \neg(\neg\psi_1)\mathbf{U}(\neg\psi_2)$. The language of finite words of φ is $L^{<\omega}(\varphi) \coloneqq \{w \in (2^{AP})^+ \mid w \models \varphi\}$, and the language of infinite words is $L(\varphi) \coloneqq \{w \in (2^{AP})^\omega \mid w \models \varphi\}$. Note that we omit the "$< \omega$" superscript if it is clear from the context which set is used. The *length* $|\varphi|$ of φ is the number of nodes in its syntax tree, the *size* of φ is the number of nodes in a DAG representing this syntax tree, and its *temporal nesting depth*, denoted by $\mathsf{depth}(\varphi)$, is defined by: $\mathsf{depth}(\mathbf{true}) = 0$; $\mathsf{depth}(a) = 0$ for an atomic proposition $a \in AP$; $\mathsf{depth}(\neg\psi) = \mathsf{depth}(\psi)$; $\mathsf{depth}(\psi_1 \wedge \psi_2) = \max(\mathsf{depth}(\psi_1), \mathsf{depth}(\psi_2))$; $\mathsf{depth}(\mathbf{X}\psi) = \mathsf{depth}(\psi) + 1$; and $\mathsf{depth}(\psi_1\mathbf{U}\psi_2) = \max(\mathsf{depth}(\psi_1), \mathsf{depth}(\psi_2)) + 1$. Chang, Manna, and Pnueli define in [8] a syntactic hierarchy for LTL formulas (over infinite words):

Definition 1 (LTL Syntactic future hierarchy [8] [5]).

- *$\Sigma_0 = \Pi_0 = \Delta_0$ is the least set containing all atomic propositions and their negations, and is closed under the application of conjunction and disjunction.*
- *Σ_{i+1} is the least set containing Π_i and negated formulas of Π_{i+1} closed under the application of conjunction, disjunction, and the* **X** *and* **U** *operators.*
- *Π_{i+1} is the least set containing Σ_i and negated formulas of Σ_{i+1} closed under the application of conjunction, disjunction, and the* **X** *and* **R** *operators.*
- *Δ_{i+1} is the least set containing Σ_{i+1} and Π_{i+1} that is closed under the application of conjunction, disjunction, and negation.*

Σ_1 is referred to as *syntactic co-safety* formulas, Π_1 as *syntactic safety* formulas.

Automata. A *deterministic semiautomaton* is a tuple $\mathcal{D} = (\Sigma, Q, \delta)$, where Σ is an alphabet; Q is a finite nonempty set of states; and $\delta \colon Q \times \Sigma \to Q$ is a transition function and we extend it to finite words in the usual way. A *path* of $\mathcal{D}$ on a word $w = \sigma_0 \cdot \sigma_1 \cdots$ is a sequence of states $q_0, q_1, \ldots$, such that for every $i < |w|$, we have $\delta(q_i, \sigma_i) = q_{i+1}$.

It is a *reset* semiautomaton if for every letter $\sigma \in \Sigma$, either i) for every state $q \in Q$ we have $\delta(q, \sigma) = q$, or ii) there exists a state $q' \in Q$, such that for every state $q \in Q$ we have $\delta(q, \sigma) = q'$.

[5] This extends [6,37] with negation, which can be removed via negation normal form.

It is *counter free* if for every state $q \in Q$, finite word $u \in \Sigma^+$, and number $n \in \mathbb{N} \setminus \{0\}$, there is a self loop of q on u^n iff there is a self loop of q on u.

A *deterministic automaton* is a tuple $\mathcal{D} = (\Sigma, Q, \iota, \delta, \alpha)$, where (Σ, Q, δ) is a deterministic semiautomaton, $\iota \in Q$ is an initial state; and α is some acceptance condition, as detailed below. A run of $\mathcal{D}$ on a word w is a path of $\mathcal{D}$ on w that starts in ι. It is a reset or counter-free automaton if its semiautomaton is.

The *acceptance condition* of an automaton on finite words is a set $F \subseteq Q$; a run is accepting if it ends in a state $q \in F$. The *acceptance condition* of an *ω-regular automaton*, on infinite words, is defined with respect to the set $\mathit{inf}(r)$ of states visited infinitely often along a run r. We define below several acceptance conditions that we use in the sequel; for other conditions, see, for example, [3].

The *Muller* condition is a set $\alpha = \{M_1, \ldots, M_k\}$ of sets $M_i \subseteq Q$ of states, and a run r is accepting if there exists a set M_i, such that $M_i = \mathit{inf}(r)$. The *Rabin* condition is a set $\alpha = \{(G_1, B_1), \ldots, (G_k, B_k)\}$ of pairs of sets of states, and r is accepting if there exists a pair (G_i, B_i), such that $G_i \cap \mathit{inf}(r) \neq \emptyset$ and $B_i \cap \mathit{inf}(r) = \emptyset$. The *Büchi* (resp. *coBüchi*) condition is a set $\alpha \subseteq Q$ of states, and r is accepting if $\alpha \cap \mathit{inf}(r) \neq \emptyset$ (resp. $\alpha \cap \mathit{inf}(r) = \emptyset$). A *weak* automaton is a Büchi automaton, in which every strongly connected component (SCC) contains only states in α or only states out of α. A *looping* automaton is a Büchi or coBüchi automaton, where all states are in α, except for a single sink state.

Deterministic automata of the above types correspond to the hierarchy of temporal properties [28]: Looping-Büchi, looping-coBüchi, weak, Büchi, coBüchi, and Rabin/Muller deterministic automata define respectively safety, guarantee (co-safety), obligation, recurrence, persistence, and reactivity languages. If the language is also LTL-definable, then there exists an equivalent LTL formula in Π_1, Σ_1, Δ_1, Π_2, Σ_2, and Δ_2, respectively [8]. Every deterministic ω-regular automaton is equivalent to deterministic Muller and Rabin automata, where the Muller (but not always Rabin) one can be defined on the same semiautomaton.

Nondeterministic and *alternating* automata (to which we only refer in Section 3, on finite words over a unary alphabet) extend deterministic automata by having a transition function $\delta\colon Q \times \Sigma \to 2^Q$ and $\delta\colon Q \times \Sigma \to$ (positive Boolean formulas over Q), respectively. (See, for example, [7] for formal definitions.)

3 Unary Alphabet

Kupferman, Ta-Shma and Vardi [20] compared the succinctness of different automata models when *counting*, that is, recognising the singleton language $\{a^k\}$ for some k over the singleton alphabet $\{a\}$. For the succinctness gap between automata and LTL, we study the task of recognising arbitrary languages over the unary alphabet, which can be seen as sets of integers, rather than a single integer.

For a unary alphabet, since there is only one infinite word, only languages on finite words are interesting. We thus consider LTL formulas over (no) atomic propositions $AP = \emptyset$, and automata on finite unary words over the corresponding alphabet $\Sigma = 2^{AP} = \{\emptyset\}$, where we use the shorthand $a = \emptyset$. The size of a deterministic automaton is the number of its states, of a nondeterministic automaton the number of its transitions, and of an alternating automaton the number of subformulas in its transition function.

We show that the size blow-up involved in translating deterministic, nondeterministic, and alternating automata to LTL, when possible, is linear, quadratic, and exponential, respectively.

In our analysis, we shall use the following folklore theorem, which extends Wolper's Theorem [45].

Proposition 1 (Extended Wolper's theorem, Folklore). *Consider an LTL formula* φ *with* $\mathsf{depth}(\varphi) = n$ *over the atomic propositions* AP*, and let* $\Sigma = 2^{AP}$*. Then for every words* $u \in \Sigma^*$*,* $v \in \Sigma^+$ *and* $t \in \Sigma^\omega$*, and numbers* $i, j > n$*,* φ *has the same truth value on the words* (uv^it) *and* (uv^jt)*.*

We use this to establish that unary LTL describes only finite and co-finite properties, and that there is a tight relation between the depth of LTL formulas and the length of words above which they are all in or all out of the language.

Proposition 2. *Given an LTL formula* φ *with* $\mathsf{depth}(\varphi) = n$ *on finite words over the unary alphabet* $\{a\}$*,* $a^i \in L(\varphi)$ *for all* $i > n$ *or* $a^i \notin L(\varphi)$ *for all* $i > n$*.*

Proposition 3. *Consider a language* $L \subseteq \{a\}^+$ *that agrees on all words of length over* n*, that is, has the same truth value on all such words. Then there is an LTL formula of size in* $O(n)$ *with language* L*.*

We now establish the trade-off between LTL and alternating automata (AFA) over unary alphabets. AFA are closed under (linear) complementation, so we use a pumping argument to bound the length after which all words have the same truth value, giving an upper bound on the LTL formula.

Lemma 1. *Every alternating automaton with* n *states that recognises an LTL-expressible language* $L \subseteq \{a\}^+$ *is equivalent to an LTL formula of size in* $O(2^n)$*.*

We show next that this upper bound is tight. Consider the language $\{a^{2^{n-1}}\}$, which, according to Proposition 2, is only recognised by LTL formulas of size at least 2^{n-1}. It is recognised by a weak alternating automaton with $2n$ states and size in $O(n)$, using an automaton based on Leiss's construction [23]. Intuitively, the alternating automaton represents an n-bit up-counter with two states for each bit, one for 1 and one for 0 (see Fig. 1), where the universal transitions enforce that nondeterministic transitions correctly update the counter.

Lemma 2 (Adaptation of [23, proof of Theorem 1]). *For every* $n \in \mathbb{N} \setminus \{0\}$*, there is a weak alternating automaton with* $2n$ *states and transition function of size in* $O(n)$ *recognising the language* $\{a^{2^{n-1}}\}$*.*

We continue to nondeterministic automata (NFAs), for which the arguments are more involved as they do not allow for linear complementation.

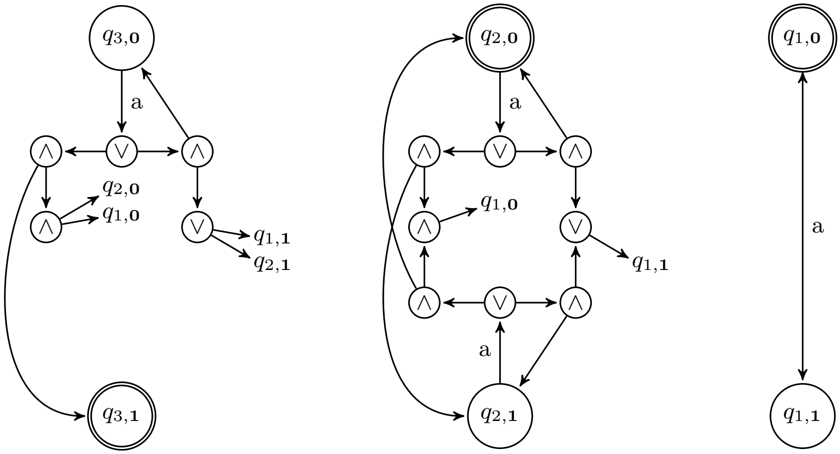

Figure 1. An alternating automaton of size in $O(n)$ recognising $\{a^{2^{n-1}}\}$; here with $n = 3$, where the initial configuration is $q_{1,\mathbf{0}} \wedge q_{2,\mathbf{0}} \wedge q_{3,\mathbf{0}}$.

Lemma 3. *Every nondeterministic automaton with n states recognising an LTL-expressible language $L \subseteq \{a\}^+$ is equivalent to an LTL formula of size in $O(n^2)$.*

Proof sketch. For finite L, by a pumping argument, $\mathcal{A}$ only accepts words up to length n, and by Proposition 3 we are done. We now consider a co-finite L.

We use 2-way deterministic automata, which are deterministic automata that process words of the form $\vdash w \dashv$, where $\vdash$ and $\dashv$ are start- and end-of-word markers respectively, and where transitions specify whether to read the letter to the right or to the left of the current position. They accept by reaching an end state, and reject by reaching a rejecting state or by failing to terminate [17], and every unary NFA $\mathcal{A}$ can be turned into a 2-way DFA $\mathcal{D}$ of size $O(n^2)$ [9].

We construct from an NFA $\mathcal{A}$ a 2-way DFA $\mathcal{D}$, and then a 2-way DFA $\mathcal{D}'$ of the same size that recognises $a^* \setminus \{a^k\}$, where a^k is the longest word not in L. We use the fact that a 2-way DFA of size m can be complemented into one of size $4m$ [17] to complement $\mathcal{D}'$ into $\mathcal{D}''$ that recognises $\{a^k\}$ and must therefore be of size at least $k + 2$ [1], so k, and by Proposition 2, an LTL formula for L, is in $O(n^2)$. □

We now show that this upper bound is tight. The previous lower bound ideas do not work with nondeterminism, since we need n states to recognise $\{a^n\}$ [20]. Yet, we need not count *exactly* to n for achieving a lower bound. We can use a variant of a language used in [4, pages 10–11]: For every positive integer k, define the set of positive integers $S_k = \{m > 0 \mid \exists i, j \in \mathbb{N}.\ m = ik + j(k + 1)\}$, and the language $V_k = \{a^m \mid m \in S_k\} \subseteq \{a\}^*$.

Proposition 4 (Folklore, [4, Theorem 3]). *For every $k \in \mathbb{N}$ the number $k^2 - k - 1$ is the maximal number not in S_k.*

Proposition 5 ([4, proof of Theorem 4]). *For every $n \in \mathbb{N}$, there is an NFA of size in $O(n)$ recognising a co-finite language $L \subseteq \{a\}^*$, such that a^{k^2-k-1} is not in L, while for every $t \geq k^2 - k$, we have that $a^t \in L$.*

Theorem 1. *The size blow-up involved in translating deterministic, nondeterministic, and alternating automata on finite unary words to LTL, when possible, is $\Theta(n)$, $\Theta(n^2)$, and $\Theta(2^n)$, respectively.*

4 General Alphabet

In this section we consider the more challenging task of turning counter-free ω-regular automata over arbitrary alphabets into LTL. We use the fact that these automata can be turned into reset cascade automata (Krohn-Rhodes-Holonomy decomposition), which we describe in Section 4.1. Our technical contribution is then the translation of reset cascade automata into LTL.

In brief, we build, in Section 4.2, a *parameterised LTL formula* that is satisfied by a word w iff the run of the cascade on w, starting in the parameter configuration S, reaches a parameter configuration T, such that the remaining suffix of w satisfies a parameter LTL formula τ. We then use this formula, in Section 4.4, to describe the automaton's acceptance condition.

When encoding the behavior of a cascade by an LTL formula, we need to overcome two major challenges: First, the cascade is a formalism that looks at the *past*, namely at the word read so far, to determine the next configuration, while an LTL formula obtains its value only from the future. Second, the cascade has an internal state, while an LTL formula does not. Our reachability formulas are therefore quite involved, built inductively over the number of levels in the cascade, and implicitly allowing to track the internal configuration of the cascade.

In Section 4.3 we analyse the length and depth of the resulting formulas.

4.1 Cascaded Automata

Cascades. A cascaded semiautomaton (analogous to the algebraic wreath product) over an alphabet Σ is a semiautomaton that can be described as a sequence of simple semiautomata, such that the alphabet of each of them is Σ together with the current state of each of the preceding semiautomata in the sequence. It is a reset cascade if it is a sequence of reset semiautomata. Formally, a *cascaded semiautomaton*, or just *cascade*, over alphabet Σ with n levels is a tuple $\mathcal{A} = \langle \Sigma, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \rangle$, such that $\mathcal{A}_i = (\Sigma_i, Q_i, \delta_i)$ is a semiautomaton for each level i, where $\Sigma_i = \Sigma \times Q_1 \times \cdots \times Q_{i-1}$. (So $\Sigma_1 = \Sigma$, $\Sigma_2 = \Sigma \times Q_1$, etc.). It is a *reset cascade* if all $\mathcal{A}_i$'s are reset semiautomata.

An i-configuration S of $\mathcal{A}$ is a tuple $\langle q_1, q_2, \ldots, q_i \rangle \in Q_1 \times \cdots \times Q_i$. If $q_{i+1} \in Q_{i+1}$ is a state of level $i+1$, we write $\langle S, q_{i+1} \rangle$ for the $(i+1)$-configuration $\langle q_1, \ldots, q_i, q_{i+1} \rangle$. Note that the 0-configuration is the empty tuple $\langle\rangle$. Further, we derive the transition relation for configurations by point-wise application of the respective δ_i's. We define $\delta_{\leq i}(\langle q_1, q_2, \ldots q_i \rangle, \sigma)$ as $\langle \delta_1(q_1, \langle \sigma \rangle), \delta_2(q_2, \langle \sigma, q_1 \rangle), \ldots \rangle$.

Note that we will omit the "$\leq i$"-subscript if it is clear from context, and by just writing "configuration", we mean an n-configuration.

Notice that $\mathcal{A}$ describes a standard semiautomaton $\mathcal{D}_{\mathcal{A}}$ over Σ, whose states are the configurations of $\mathcal{A}$ of level n, and its transition function is $\delta_{\leq n}$. If there are up to j states in each level of $\mathcal{A}$, there are up to j^n states in $\mathcal{D}_{\mathcal{A}}$. Observe that when $\mathcal{A}$ is a reset cascade, it can be translated to an equivalent reset cascade with up to $n \log j$ levels, and 2 states in each level [14, Ex. I.10.2].

For a state $q \in Q_i$ of level i of a reset cascade, we denote by $\mathsf{Enter}(q)$, $\mathsf{Stay}(q)$, and $\mathsf{Leave}(q) \subseteq \Sigma \times Q_1 \times \cdots \times Q_{i-1}$ the sets of (combined) letters that enter q, stay in it, and leave it, respectively. These are sets of pairs $\langle \sigma, S \rangle$, where S is an $(i-1)$-configuration and $\sigma \in \Sigma$. Notice that $\mathsf{Enter}(q) \subseteq \mathsf{Stay}(q)$, and that $\mathsf{Leave}(q)$ is the complement of $\mathsf{Stay}(q)$ (w.r.t. the relevant (combined) letters).

A semiautomaton (Σ, Q, δ) is *homomorphic* to a cascade $\langle \Sigma, \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ if there exists a partial surjective function $\varphi\colon Q_1 \times \cdots \times Q_n \to Q$, such that for every $\sigma \in \Sigma$ and $S \in Q_1 \times \cdots \times Q_n$, we have $\delta(\varphi(S), \sigma) = \varphi(\delta_{\leq n}(S, \sigma))$.

Proposition 6 (Part of the Krohn-Rhodes-Holonomy Decomposition [14, Corollary II.7.2], [26, Theorem 3]). *Every counter-free deterministic semiautomaton $\mathcal{D}$ with n states is homomorphic to a reset cascade $\mathcal{A}$ with up to 2^n levels and 2^n states in each level.*

Remark 1. The Krohn-Rhodes and Holonomy decomposition theorems consider also more general cascades and give results with respect to arbitrary semiautomata. The Holonomy decomposition in [14], as opposed to many other proofs of the Krohn-Rhodes decomposition, guarantees up to 2^n levels with up to 2^n states in each level. Yet, it shows that $\mathcal{A}$ *covers* $\mathcal{D}$, allowing $\mathcal{A}$ to operate over an alphabet different from that of $\mathcal{D}$. In [26,27,25], the algebraic proof of [14] is translated to an automata-theoretic one, providing the stated homomorphism. It is also stated in [26, Theorem 3.1], [27, Corollary 20], and [25, Corollary 2] that the number of configurations in $\mathcal{A}$ is singly exponential in n, but to the best of our understanding they do not provide an explicit proof for it.

Cascades with acceptance conditions. As a cascade $\mathcal{A}$ describes a standard semiautomaton (whose states are the configurations of $\mathcal{A}$), we can add to it an initial configuration and an acceptance condition to make it a standard deterministic automaton. We show below that the homomorphism between an automaton and a cascade can be extended to also transfer the same acceptance condition.

Proposition 7. *Let $\mathcal{D}$ be a deterministic Büchi, coBüchi or Rabin automaton, with a semiautomaton homomorphic to a cascade $\mathcal{A}$. There is respectively a deterministic Büchi, coBüchi or Rabin automaton $\mathcal{D}'$ equivalent to $\mathcal{D}$ with semiautomaton $\mathcal{A}$. For Rabin, $\mathcal{D}$ and $\mathcal{D}'$ have the same number of acceptance pairs.*

Proposition 8. *Consider a deterministic Muller automaton $\mathcal{D}$ with n states, whose semiautomaton is homomorphic to a reset cascade $\mathcal{A}$ with m configurations. Then there is a deterministic Muller automaton $\mathcal{D}'$ equivalent to $\mathcal{D}$, whose semiautomaton is $\mathcal{A}$ and its Muller condition has up to $2^{O(mn)}$ acceptance sets.*

4.2 Encoding Reachability within Reset Cascades by LTL Formulas

For the rest of this section, let us fix a set of atomic propositions AP, an alphabet $\Sigma = 2^{AP}$, and a reset cascade $\mathcal{A} = \langle \Sigma, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n \rangle$.

The main reachability formula. For every level i of $\mathcal{A}$, three configurations S, B and T of level i, and two LTL formulas β and τ, we will define the LTL formula $S \underset{\cancel{B(\beta)}}{\rightsquigarrow} T\,(\tau)$ with the intended semantics that it holds on a word $w \in \Sigma^\omega$ iff $\mathcal{A}$ goes from the 'starting' configuration S to the 'target' configuration T along some prefix u of w, such that the suffix of w after u satisfies τ and the path along u avoids the 'bad' configuration B with a suffix satisfying β.

Auxiliary reachability formulas. We will formally define the main reachability formula by induction on the level i of the involved configurations, and using four auxiliary formulas, whose intended semantics is described in Table 1. These formulas distinguish between the case that the top-level state is unchanged along the reachability path, denoted with a solid arrow $\longrightarrow$, and the case that it is changed, denoted by a dashed arrow $\dashrightarrow$. They also have dual, weak, versions.

Observe that intuitively $S \underset{\cancel{B(\beta)}}{\rightsquigarrow} T\,(\tau)$ is an extended *Until* operator, while its dual $S \underset{\cancel{B(\beta)}}{\overset{\text{weak}}{\rightsquigarrow}} T\,(\tau) = \neg(S \underset{\cancel{T(\tau)}}{\rightsquigarrow} B\,(\beta))$ is an extended *Weak until* (or *Release*) operator. We build the formulas so that for appropriate choices of β and τ, the (strong) reachability formulas 1, 3, and 5 (as numbered in Table 1) are syntactic co-safety and the weak formulas 2 and 4 are syntactic safety formulas.

Formulas 1 and 2. The main formula is simply defined as the union of two auxiliary formulas, corresponding to whether or not the top-level state changes, and its weak version is defined to be its dual.

$$S \underset{\cancel{B(\beta)}}{\rightsquigarrow} T\,(\tau) := \begin{cases} (\neg\beta)\mathbf{U}\tau & \text{if } S = \langle\rangle \\ S \xrightarrow[\cancel{B(\beta)}]{} T\,(\tau) \vee S \underset{\cancel{B(\beta)}}{\dashrightarrow} T\,(\tau) & \text{otherwise.} \end{cases}$$

$$S \underset{\cancel{B(\beta)}}{\overset{\text{weak}}{\rightsquigarrow}} T\,(\tau) := \neg\left(S \underset{\cancel{T(\tau)}}{\rightsquigarrow} B\,(\beta)\right)$$

Formula 3. Since the formula should ensure that the top-level state s is unchanged, we first distinguish between four cases, depending on which of the source configuration $\langle S, s\rangle$, bad configuration $\langle B, b\rangle$, and target configuration $\langle T, t\rangle$ are equal. The definitions of the four cases only differ in whether or not each of β and τ are satisfied in the first position of the word.

We define them using an intermediate common formula that is indifferent to the first position, which we mark by "> 0" on top of the arrow. We then define the "> 0" formula by using the main reachability formula with respect to a lower level, namely with respect to the configurations S and T instead of $\langle S, s\rangle$ and $\langle T, t\rangle$, and having corresponding disjunctions and conjunctions on all the combined letters of the top level that belong to $\mathsf{Stay}(s)$ and $\mathsf{Leave}(s)$.

Reachability formula φ	Intended semantics Intuitively: Reading a word w from the configuration S or $\langle S, s\rangle$ Formally: $w \models \varphi \iff$
1. $S \underset{\cancel{B(\beta)}}{\rightsquigarrow} T\,(\tau)$	not reaching $B(\beta)$ until reaching $T(\tau)$. $\exists i \geq 0.\ \delta(S, w_{[0..i)}) = T \wedge w_{[i..]} \models \tau$ $\wedge\ (\forall j \in [0..i).\ \delta(S, w_{[0..j)}) \neq B \vee w_{[j..]} \not\models \beta)$
2. $S \underset{\cancel{B(\beta)}}{\overset{\text{weak}}{\rightsquigarrow}} T\,(\tau)$	reaching $T(\tau)$ releases not reaching $B(\beta)$. $\forall i \geq 0.\ (\delta(S, w_{[0..i)}) = B \wedge w_{[i..]} \models \beta)$ $\rightarrow\ (\exists j \in [0..i).\ \delta(S, w_{[0..j)}) = T \wedge w_{[j..]} \models \tau)$
3. $\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\longrightarrow} \langle T, t\rangle\,(\tau)$	not reaching $\langle B, b\rangle(\beta)$ until reaching $\langle T, t\rangle(\tau)$, while staying in s. $\exists i \geq 0.\ \delta(\langle S, s\rangle, w_{[0..i)}) = \langle T, t\rangle \wedge w_{[i..]} \models \tau$ $\wedge\ (\forall j \in [0..i).\ \delta(\langle S, s\rangle, w_{[0..j)}) \neq \langle B, b\rangle \vee w_{[j..]} \not\models \beta)$ $\wedge\ (\forall j \in [0..i).\ \langle w[j], \delta(S, w_{[0..j)})\rangle \in \mathsf{Stay}(s))$
4. $\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{\text{weak}}{\longrightarrow}} \langle T, t\rangle\,(\tau)$	reaching $\langle T, t\rangle(\tau)$ releases not (reaching $\langle B, b\rangle(\beta)$ or leaving s). $\forall i \geq 0.\ \big((\delta(\langle S, s\rangle, w_{[0..i)}) = \langle B, b\rangle \wedge w_{[i..]} \models \beta)$ $\vee\ (i > 0 \wedge \langle w[i{-}1], \delta(S, w_{[0..i-1)})\rangle \in \mathsf{Leave}(s))\big)$ $\rightarrow\ (\exists j \in [0..i).\ \delta(\langle S, s\rangle, w_{[0..j)}) = \langle T, t\rangle \wedge w_{[j..]} \models \tau)$
5. $\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\dashrightarrow} \langle T, t\rangle\,(\tau)$	not reaching $\langle B, b\rangle(\beta)$ until reaching $\langle T, t\rangle(\tau)$ and leaving s. $\exists i_1, i_2 \geq 0.\ \delta(\langle S, s\rangle, w_{[0..i_1)}) = \langle T, t\rangle \wedge w_{[i_1..]} \models \tau$ $\wedge\ (\exists j_1 \in [0..i_1).\ \langle w[j_1], \delta(S, w_{[0..j_1)})\rangle \in \mathsf{Enter}(t))$ $\wedge\ \langle w[i_2], \delta(S, w_{[0..i_2)})\rangle \in \mathsf{Leave}(s)$ $\wedge\ (\forall j_2 \in [0..\max(i_1{-}1, i_2)].\ \delta(\langle S, s\rangle, w_{[0..j_2)}) \neq \langle B, b\rangle$ $\vee\ w_{[j_2..]} \not\models \beta)$

Table 1. The intended semantics of reachability formulas. Orange subformulas show the difference between the auxiliary formulas and the first or second (main) formula.

$$\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\longrightarrow} \langle T, t\rangle\,(\tau) \coloneqq$$

$$\begin{cases} \langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{>0}{\longrightarrow}} \langle T, t\rangle\,(\tau) & \text{if } \langle S, s\rangle \neq \langle B, b\rangle \text{ and } \langle S, s\rangle \neq \langle T, t\rangle \\ \langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{>0}{\longrightarrow}} \langle T, t\rangle\,(\tau) \vee \tau & \text{if } \langle S, s\rangle \neq \langle B, b\rangle \text{ and } \langle S, s\rangle = \langle T, t\rangle \\ \langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{>0}{\longrightarrow}} \langle T, t\rangle\,(\tau) \wedge \neg\beta & \text{if } \langle S, s\rangle = \langle B, b\rangle \text{ and } \langle S, s\rangle \neq \langle T, t\rangle \\ \left(\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{>0}{\longrightarrow}} \langle T, t\rangle\,(\tau) \wedge \neg\beta\right) \vee \tau & \text{if } \langle S, s\rangle = \langle B, b\rangle \text{ and } \langle S, s\rangle = \langle T, t\rangle \end{cases}$$

$$\text{where } \langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\overset{>0}{\longrightarrow}} \langle T, t\rangle\,(\tau) \coloneqq \bigvee_{\substack{\langle \sigma, T'\rangle \in \mathsf{Stay}(s) \\ \text{s.t. } \langle T', s\rangle \xrightarrow{\sigma} \langle T, t\rangle}} \left(S \underset{\cancel{S(\mathbf{false})}}{\rightsquigarrow} T'\,(\sigma \wedge \mathbf{X}\tau) \right)$$

$$\wedge \bigwedge_{\langle \eta, L\rangle \in \mathsf{Leave}(s)} S \overset{}{\underset{\not L(\eta)}{\rightsquigarrow}} T' \,(\sigma \wedge \mathbf{X}\tau) \quad \wedge \bigwedge_{\substack{\langle \rho, B'\rangle \in \mathsf{Stay}(s) \\ \text{s.t. } \langle B', s\rangle \xrightarrow{\rho} \langle B, b\rangle}} S \underset{\not B'(\rho \wedge \mathbf{X}\beta)}{\rightsquigarrow} T' \,(\sigma \wedge \mathbf{X}\tau) \Bigg)$$

Formula 4. Its intended semantics is also that the top-level state s is unchanged, but we weaken Formula 3 by not enforcing that the target configuration $\langle T, t\rangle$ is reached and τ is satisfied. Thus as long as the top-level state s stays unchanged and the bad configuration $\langle B, b\rangle$ is not reached while satisfying β, Formula 4 is also satisfied. Note that since both Formula 3 and Formula 4 need to ensure that the top-level state s is unchanged they cannot simply be defined as the dual of each other. However, they share the same construction principle:

$$\langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak}} \langle T, t\rangle\,(\tau) :=$$

$$\begin{cases} \langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak},>0} \langle T, t\rangle\,(\tau) & \text{if } \langle S, s\rangle \neq \langle B, b\rangle \text{ and } \langle S, s\rangle \neq \langle T, t\rangle \\ \langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak},>0} \langle T, t\rangle\,(\tau) \vee \tau & \text{if } \langle S, s\rangle \neq \langle B, b\rangle \text{ and } \langle S, s\rangle = \langle T, t\rangle \\ \langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak},>0} \langle T, t\rangle\,(\tau) \wedge \neg\beta & \text{if } \langle S, s\rangle = \langle B, b\rangle \text{ and } \langle S, s\rangle \neq \langle T, t\rangle \\ \left(\langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak},>0} \langle T, t\rangle\,(\tau) \vee \tau\right) \wedge \neg\beta & \text{if } \langle S, s\rangle = \langle B, b\rangle \text{ and } \langle S, s\rangle = \langle T, t\rangle \end{cases}$$

where

$$\langle S, s\rangle \xrightarrow[\not\langle B,b\rangle(\beta)]{\text{weak},>0} \langle T, t\rangle\,(\tau) :=$$

$$\bigvee_{\substack{\langle \sigma, T'\rangle \in \mathsf{Stay}(s) \\ \text{s.t. } \langle T', s\rangle \xrightarrow{\sigma} \langle T, t\rangle}} \Bigg(\bigwedge_{\langle \eta, L\rangle \in \mathsf{Leave}(s)} S \overset{\text{weak}}{\underset{\not L(\eta)}{\rightsquigarrow}} T' \,(\sigma \wedge \mathbf{X}\tau) \wedge \bigwedge_{\substack{\langle \rho, B'\rangle \in \mathsf{Stay}(s) \\ \text{s.t. } \langle B', s\rangle \xrightarrow{\rho} \langle B, b\rangle}} S \overset{\text{weak}}{\underset{\not B'(\rho \wedge \mathbf{X}\beta)}{\rightsquigarrow}} T' \,(\sigma \wedge \mathbf{X}\tau) \Bigg) \quad (1)$$

$$\vee \quad \Bigg(\bigwedge_{\langle \eta, L\rangle \in \mathsf{Leave}(s)} S \overset{\text{weak}}{\underset{\not L(\eta)}{\rightsquigarrow}} S\,(\mathbf{false}) \quad \wedge \bigwedge_{\substack{\langle \rho, B'\rangle \in \mathsf{Stay}(s) \\ \text{s.t. } \langle B', s\rangle \xrightarrow{\rho} \langle B, b\rangle}} S \overset{\text{weak}}{\underset{\not B'(\rho \wedge \mathbf{X}\beta)}{\rightsquigarrow}} S\,(\mathbf{false}) \Bigg) \quad (2)$$

Formula 5. The definition of the last reachability formula is the most challenging, since the top-level state changes ($s \neq t$), which prevents the direct usage of lower level configurations.

Intuitively, before reaching the target configuration $\langle T, t\rangle$, the run must see a combined letter $\langle \sigma, T'\rangle \in \mathsf{Enter}(t)$, after which the top-level state t is preserved and the bad situation $\langle B, b\rangle(\beta)$ is avoided. This is line (1) of the definition.

The run must also not see $\langle B, b\rangle(\beta)$ before reaching T', which is handled in line (2), whose difference from line (1) is the additional constraint on the path from S to T'. (Line (1) is required for the case that $\mathsf{Enter}(b)$ is empty.) We use Formula 4 for that constraint, rather than Formula 3 which could also be used, in order to ensure that Formula 5 can be a syntactic co-safety formula.

Lastly, line (3) ensures that the top-level state is indeed changed.

$$\langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\beta)}}{\dashrightarrow} \langle T, t\rangle\,(\tau) \coloneqq$$

$$\bigvee_{\substack{\langle \sigma, T'\rangle \in \\ \mathsf{Enter}(t)}} \left(S \underset{\cancel{S(\mathbf{false})}}{\rightsquigarrow} T' \left(\sigma \wedge \mathbf{X}\Big(\delta(\langle T', \cdot\rangle, \sigma) \xrightarrow[\cancel{\langle B,b\rangle(\beta)}]{} \langle T, t\rangle\,(\tau)\Big)\right)\right) \wedge \tag{1}$$

$$\bigwedge_{\substack{\langle \eta, R\rangle \in \\ \mathsf{Enter}(b)}} S \underset{\cancel{R(\eta\wedge\mathbf{X}(\delta(\langle R,\cdot\rangle,\eta)\xrightarrow[\cancel{\langle T,t\rangle(\tau)}]{\text{weak}}\langle B,b\rangle(\beta)))}}{\rightsquigarrow} T' \left(\sigma \wedge \mathbf{X}\Big(\delta(\langle T', \cdot\rangle, \sigma) \xrightarrow[\cancel{\langle B,b\rangle(\beta)}]{} \langle T, t\rangle\,(\tau)\Big)\right)\Bigg) \tag{2}$$

$$\wedge \bigvee_{\substack{\langle \sigma, L\rangle \in \\ \mathsf{Leave}(s)}} \langle S, s\rangle \xrightarrow[\cancel{\langle B,b\rangle(\beta)}]{} \langle L, s\rangle \left(\sigma \wedge \begin{cases} \neg\beta & \text{if } \langle L, s\rangle = \langle B, b\rangle \\ \mathbf{true} & \text{otherwise.} \end{cases}\right) \tag{3}$$

We prove the correctness of the above definitions with respect to the intended meaning of Table 1 by induction on the level of the involved configurations.

Lemma 4. *The intended semantics of Table 1 hold for all infinite words $w \in \Sigma^\omega = (2^{AP})^\omega$, configurations S, B, T of level $m \leq n$, states s, b, t in level $m+1$ (when $m < n$), and LTL formulas β and τ over AP.*

Using the same induction principle we prove that the reachability formulas stay within certain classes of the syntactic future hierarchy (Definition 1). We use $S \underset{\cancel{B(X)}}{\rightsquigarrow} T\,(Y) \in Z$ as a shorthand for saying that for every formulas $\beta \in X$ and $\tau \in Y$, the formula $S \underset{\cancel{B(\beta)}}{\rightsquigarrow} T\,(\tau)$ is in Z.

Lemma 5. *Let S, B, T be configurations of level $m \leq n$, and let s, b, t be states in level $m+1$ (when $m < n$). Then for $i \geq 1$ it holds that:*

- $S \underset{\cancel{B(\Pi_i)}}{\rightsquigarrow} T\,(\Sigma_i)\,,\ \langle S, s\rangle \xrightarrow[\cancel{\langle B,b\rangle(\Pi_i)}]{} \langle T, t\rangle\,(\Sigma_i)\,,\ \langle S, s\rangle \underset{\cancel{\langle B,b\rangle(\Pi_i)}}{\dashrightarrow} \langle T, t\rangle\,(\Sigma_i) \ \in\ \Sigma_i$
- $S \underset{\cancel{B(\Sigma_i)}}{\overset{weak}{\rightsquigarrow}} T\,(\Pi_i)\,,\ \langle S, s\rangle \xrightarrow[\cancel{\langle B,b\rangle(\Sigma_i)}]{weak} \langle T, t\rangle\,(\Pi_i) \ \in\ \Pi_i$

4.3 Depth and Length Analysis

We analyze the length and temporal-nesting depth of the LTL reachability formulas defined in Section 4.2. Notice that both measures are of independent interest, as there might be a non-elementary gap between the depth and length of LTL formulas [15, Theorem 6]. Since we provide upper bounds, the bound on the length of formulas obviously gives also a bound on their size.

We consider a reset cascade $\mathcal{A}$ with n levels, as in Section 4.2, and further assume for the length and depth analysis that it has up to n states in each level. (This assumption holds in the reset cascades that result from the Krohn-Rohdes decomposition as per Proposition 6.)

We define for each of the five reachability formulas a *depth function* $\mathsf{D}_x(i, d)$ and a *length function* $\mathsf{L}_x(i, l)$, where x refers to the number of the reachability formula, to bound the depth and length of the formulas. These depend on the level i of its input configurations S, B and T, and the maximal depth d and length l of its input formulas β and τ. For the main (first) reachability formula, we also use D and L, standing for D_1 and L_1. For example, the length of the first formula $S \underset{\not B(\beta)}{\rightsquigarrow} T\,(\tau)$ over configurations S, B and T of level 7 and formulas β and τ of length up to 77 is bounded by the value of $\mathsf{L}_1(7, 77)$.

For simplicity, we consider the LTL representation of an alphabet letter $\sigma \in \Sigma$ to be of length 1, while its actual length is $3\log_2|\Sigma|$. This increase is due to the need to encode an alphabet letter $\sigma \in \Sigma = 2^{AP}$ as a conjunction of atomic propositions in AP. The representation length can be multiplied by the total length of the final relevant formula (e.g., a formula equivalent to the entire reset cascade), since it remains constant along all steps of our inductive computation.

We provide in Table 2 upper bounds on the depth and length functions, relative to values of other depth and length functions with respect to configurations of the same or lower-by-one level. The table is constructed by following the syntactic definitions of the reachability formulas, and applying basic simplifications to the resulting expressions. For example, $\mathsf{L}_1(0, l) = 2+2l$ standing for the length of $(\neg\beta)\mathbf{U}\tau$. In Lemma 6 we will use Table 2 to bound the absolute depth and length of the main reachability formula.

Depth Analysis. The temporal nesting depth of the main reachability formula $S \underset{\not B(\beta)}{\rightsquigarrow} T\,(\tau)$ is intuitively exponential in the number n of levels of the reset cascade (linear in the number of configurations), since it is defined inductively along these levels, and the depth of a level-$(i+1)$ formula is about twice the depth of a level-i formula. The parameters of the reachability formula are both the configurations S, B and T of level i, and the formulas β and τ; yet, the depth of the reachability formula only linearly depends on the depth of β and τ.

Length Analysis. Intuitively, the overall length of the main reachability formula $S \underset{\not B(\beta)}{\rightsquigarrow} T\,(\tau)$ with respect to configurations of the top level is doubly exponential in the number n of levels of the reset cascade (and thus singly exponential in the number of configurations), since the formula is defined inductively along these levels, and the length $\mathsf{L}(i, l)$ is roughly $\mathsf{L}(i-1, l) \cdot \mathsf{L}(i-1, l)$. More precisely, $\mathsf{L}(i, l) = l \cdot f(i)$ for some doubly exponential function $f(i)$.

Now, why is $\mathsf{L}(i, l)$ roughly equal to $\mathsf{L}(i-1, l) \cdot \mathsf{L}(i-1, l)$? The dominant component of the level-i reachability formula is line (2) in the definition of $\langle S, s\rangle \underset{\not\langle B, b\rangle(\beta)}{\dashrightarrow} \langle T, t\rangle\,(\tau)$. It is a level-$(i-1)$ reachability formula whose formula-parameters are themselves auxiliary reachability formulas of level i with formula parameters of length l. The length of an auxiliary reachability formula of level i is roughly as of the main reachability formula of level $i-1$, implying that the length of $L_i(l)$ is roughly $L_{i-1}(L_{i-1}(l))$. By the inductive proof that $L_{i-1}(l) = l \cdot f(i-1)$, we get that $L_i(l) = L_{i-1}(L_{i-1}(l)) = L_{i-1}(l) \cdot f(i-1) = l \cdot f(i-1) \cdot f(i-1)$.

As for the many disjunctions and conjunctions that appear in the formulas, observe that the number of disjuncts and conjuncts does not depend on the

Reachability formula φ	Bounds on $\mathsf{depth}(\varphi)$ and length $\lvert\varphi\rvert$
1. $S \underset{\not B(\beta)}{\rightsquigarrow} T\,(\tau)$	$\mathsf{D}_1(i,d) \le \begin{cases} d+1 & \text{if } i=0 \\ \max(\mathsf{D}_3(i,d)\,,\,\mathsf{D}_5(i,d)) & \text{otherwise.}\end{cases}$ $\mathsf{L}_1(i,l) \le \begin{cases} 2+2l & \text{if } i=0 \\ 1+\mathsf{L}_3(i,l)+\mathsf{L}_5(i,l) & \text{otherwise.}\end{cases}$
2. $S \overset{\text{weak}}{\underset{\not B(\beta)}{\rightsquigarrow}} T\,(\tau)$	$\mathsf{D}_2(i,d) = \mathsf{D}_1(i,d)$ $\mathsf{L}_2(i,l) = 1+\mathsf{L}_1(i,l)$
3. $\langle S,s\rangle \underset{\not\langle B,b\rangle(\beta)}{\longrightarrow} \langle T,t\rangle\,(\tau)$	$\mathsf{D}_3(i,d) \le \mathsf{D}_1(i{-}1,d+1)$ $\mathsf{L}_3(i,l) \le 3{+}2l + \lvert\Sigma\rvert n^{i-1}\big(1{+}\mathsf{L}_1(i{-}1,3{+}l){+}$ $1+\lvert\Sigma\rvert n^{i-1}(\mathsf{L}_1(i{-}1,3{+}l)+1){+}$ $1+\lvert\Sigma\rvert n^{i-1}(\mathsf{L}_1(i{-}1,3{+}l)+1)\big)$ $\le 3+2l+4\lvert\Sigma\rvert^2 n^{2(i-1)}\mathsf{L}_1(i{-}1,l{+}3)$
4. $\langle S,s\rangle \overset{\text{weak}}{\underset{\not\langle B,b\rangle(\beta)}{\longrightarrow}} \langle T,t\rangle\,(\tau)$	$\mathsf{D}_4(i,d) \le \mathsf{D}_2(i{-}1,d+1) = \mathsf{D}_1(i{-}1,d+1)$ $\mathsf{L}_4(i,l) \le 3+2l+(1+\lvert\Sigma\rvert n^{i-1})\big(1+\lvert\Sigma\rvert n^{i-1}(1+\mathsf{L}_2(i{-}1,l{+}3))\big)$ $\le 3+2l+4\lvert\Sigma\rvert^2 n^{2(i-1)}\mathsf{L}_1(i{-}1,l+3)$
5. $\langle S,s\rangle \underset{\not\langle B,b\rangle(\beta)}{\dashrightarrow} \langle T,t\rangle\,(\tau)$	$\mathsf{D}_5(i,d) \le \mathsf{D}_1(i{-}1,\max(1+\mathsf{D}_3(i,d)\,,\,1+\mathsf{D}_4(i,d)))$ $\mathsf{L}_5(i,l) \le \lvert\Sigma\rvert n^{i-1}\cdot\big(\mathsf{L}_1(i-1,3+\mathsf{L}_3(i,l))+2{+}$ $\lvert\Sigma\rvert n^{i-1}\cdot\big(\mathsf{L}_1(i-1,\max(3+\mathsf{L}_3(i,l),3+\mathsf{L}_4(i,l)))+1\big)$ $+1+\lvert\Sigma\rvert n^{i-1}\cdot(1+\mathsf{L}_3(i,3+l))\big)$

Table 2. The relative depths and lengths of the reachability formulas over configurations of level i, and LTL formulas β and τ of depth at most d and length at most l. For the first two reachability formulas, we consider $i \ge 0$ and for the other formulas $i \ge 1$.

formula-parameters β and τ, but only the level i of the configurations S, B, and T. Hence, they do not dominate the growth rate of the overall formula length.

Lemma 6. *Consider a reset cascade $\mathcal{A}$ with n levels and up to n states in each level, and a formula $\zeta = S \underset{\not B(\beta)}{\rightsquigarrow} T\,(\tau)$ with configurations S, B and T of $\mathcal{A}$ of level $i \le n$. Let $d = \max(\mathsf{depth}(\beta), \mathsf{depth}(\tau))$ and let $l = \max(\lvert\beta\rvert, \lvert\tau\rvert)$. Then:*

$$(a)\ \mathsf{depth}(\zeta) \le d + 3^i \quad \text{and} \quad (b)\ \lvert\zeta\rvert \le l \cdot (10\lvert\Sigma\rvert^2 n)^{4^i}$$

Lemma 6 is proven by induction on i and the details of this proof can be found in the full version [5].

4.4 Translating Deterministic Counter-Free Automata to LTL

We use the reachability formulas of Section 4.2 to translate a reset cascade $\mathcal{A}$ to an equivalent LTL formula. Our LTL formulation of $\mathcal{A}$'s acceptance condition is based on an LTL formulation of "C is visited finitely/infinitely often along a run of $\mathcal{A}$ on a word w", for a given configuration C of $\mathcal{A}$. It thus applies to every ω-regular acceptance condition and by Propositions 6 and 8 to every deterministic counter-free ω-regular automaton. We introduce two shorthands to the main reachability formula: the first is satisfied if we reach T from S without any side constraints (which is always satisfied in the case that $S = T$), and the second requires that we reach it along a nonempty prefix.

$$S \rightsquigarrow T := S \underset{\cancel{T(\mathbf{false})}}{\rightsquigarrow} T\,(\mathbf{true}) \qquad S \overset{>0}{\rightsquigarrow} T := \bigvee_{\sigma \in \Sigma} \Big(\sigma \wedge \mathbf{X}(\delta(S,\sigma) \rightsquigarrow T)\Big)$$

With Lemmas 4 and 5 we then obtain (the proof can be found in [5]):

Lemma 7. *Consider a reset cascade $\mathcal{A} = \langle 2^{AP}, \mathcal{A}_1, \ldots, \mathcal{A}_n \rangle$ together with an initial configuration ι and some configuration C. Then for a word $w \in (2^{AP})^\omega$, the run of $\mathcal{A}$ on w starting in ι visits C finitely often iff w satisfies the formula $Fin(C) := \neg(\iota \rightsquigarrow C) \vee \iota \rightsquigarrow C(\neg(C \overset{>0}{\rightsquigarrow} C))$. Furthermore, $Fin(C) \in \Sigma_2$.*

We are now in position to give our main result.

Theorem 2. *Every counter-free deterministic ω-regular automaton $\mathcal{D}$ over alphabet 2^{AP} with n states (and any acceptance condition) is equivalent to an LTL formula φ over atomic propositions AP of double-exponential temporal-nesting depth (in $O(2^{2^n})$) and triple-exponential length (in $2^{2^{O(2^n)}}$). If $\mathcal{D}$ is a looping-Büchi, looping-coBüchi, weak, Büchi, coBüchi, or Muller automaton then φ is respectively in the $\Pi_1, \Sigma_1, \Delta_1, \Pi_2, \Sigma_2$, or Δ_2 syntactic fragment of LTL.*

Proof. We first prove the general result, w.r.t. an arbitrary counter-free deterministic automaton $\mathcal{D}$, and then take into account $\mathcal{D}$'s acceptance condition, to establish the last part of the theorem.

Consider a counter-free deterministic ω-regular automaton $\mathcal{D}$ with some acceptance condition and n states. Recall that there is a Muller automaton $\mathcal{D}'$ equivalent to $\mathcal{D}$ over the semiautomaton of $\mathcal{D}$. By Propositions 6 and 8, $\mathcal{D}'$ is equivalent to a deterministic Muller automaton $\mathcal{D}''$ that is described by a reset cascade $\mathcal{A}$ with up to $m = 2^n$ levels and m states in each level (and thus up to m^m configurations), and whose acceptance condition has up to $k \in 2^{O(m^m n)} = 2^{O(m^m)}$ acceptance sets. An LTL formula φ equivalent to $\mathcal{D}$ can be defined by formulating the acceptance condition of $\mathcal{D}'$ along Lemma 7.

Recall that the Muller condition is a k-elements disjunction, where each disjunct M is a conjunction of requirements to visit infinitely often every configuration from some set G and finitely often every configuration not in G. Observe that M can be formulated as a disjunction over all the configurations in $\mathcal{D}''$ (at most m^m), having for each configuration C the LTL formula $Fin(C)$ or $\neg Fin(C)$, as defined in Lemma 7, depending on whether or not $C \in G$. Hence, the overall formula φ is a combination of disjunctions and conjunctions of up to $k \cdot m^m$ subformulas of the form $Fin(C)$ or $\neg Fin(C)$. Therefore, the depth of φ is the same as of $Fin(C)$, while $|\varphi| \in O(km^m|Fin(C)|) \leq 2^{O(m^m)}|Fin(C)|$. For calculating $\mathsf{depth}(Fin(C))$ and $|Fin(C)|$, we use Lemma 6 bottom up over the subformulas of $Fin(C)$.

Depth.

$\mathsf{depth}(\iota \rightsquigarrow C) \leq 3^m$; $\mathsf{depth}(C \overset{>0}{\rightsquigarrow} C) \leq 3^m + 1$

$\mathsf{depth}(\iota \rightsquigarrow C(\neg(C \overset{>0}{\rightsquigarrow} C))) \leq 2 \cdot 3^m + 1$

$\mathsf{depth}(\mathit{Fin}(C)) = \max(3^m, 2 \cdot 3^m + 1) \in O(3^m) = O(2^{2^n})$,
implying $\mathsf{depth}(\varphi) \in O(2^{2^n})$.

Length.

$|\iota \rightsquigarrow C| \leq (10|\Sigma|^2 m)^{4^m}$; $|C \overset{>0}{\rightsquigarrow} C| \leq (4|\Sigma|) \cdot (10|\Sigma|^2 m)^{4^m}$

$|\iota \rightsquigarrow C(\neg(C \overset{>0}{\rightsquigarrow} C))| \leq (4|\Sigma|(10|\Sigma|^2 m)^{4^m}+1)(10|\Sigma|^2 m)^{4^m} \in (|\Sigma| m)^{2^{O(m)}}$

$|\mathit{Fin}(C)| \in 2 + (10|\Sigma|^2 m)^{4^m} + (|\Sigma| m)^{2^{O(m)}} \in (|\Sigma| m)^{2^{O(m)}}$.

Therefore, $|\varphi| \in 2^{O(m^m)} \cdot (m^m) \cdot ((|\Sigma| m)^{2^{O(m)}}) = |\Sigma|^{2^{O(m)}}$.

Expressing the length of φ with respect to the number n of states in the automaton $\mathcal{D}$, and taking into account the fact that the alphabet Σ has at most n^n different letters (any additional letter must have the same behavior as another letter), we have: $|\varphi| \in |\Sigma|^{2^{O(2^n)}} \leq (2^n)^{2^{O(2^n)}} = 2^{2^{O(2^n)}}$.

We now sketch the second part of the theorem connecting the syntactic hierarchy and the different acceptance conditions of $\mathcal{D}$. We only consider the cases in which $\mathcal{D}$ is either a Muller or a coBüchi automaton. The complete analysis is given in the full version [5]. If $\mathcal{D}$ is a Muller automaton, then the overall formula φ is in Δ_2, since it is a Boolean combination of $\mathit{Fin}(C)$ formulas, which by Lemma 7 belong to Σ_2. If $\mathcal{D}$ is a coBüchi automaton, then we construct the formula φ directly from the coBüchi condition α: φ is a conjunction of $\mathit{Fin}(C)$ formulas over all configurations C that are mapped to states in α. As $\mathit{Fin}(C)$ belongs to Σ_2, so does φ. □

Observe that by Theorem 2, we get the following result, extending the result of [39, Theorem 3.2] that only considers Rabin automata.

Corollary 1. *Every counter-free deterministic ω-regular automaton (with any acceptance condition) recognises an LTL-definable language.*

Proof. Recall that every deterministic ω-regular automaton is equivalent to a deterministic Muller automaton over the same semiautomaton (see, e.g., [3]). The claim is then a direct consequence of Theorem 2. □

Remark 2. Theorem 2 can be adapted to the finite-word setting. While on infinite words, the ne**X**t operator is self-dual, i.e., $\neg\mathbf{X}\psi$ is equivalent to $\mathbf{X}\neg\psi$, over finite words, this equivalence does not hold on words of length 1. Thus $\mathbf{X}$ gains a dual *weak next*, defined as $\tilde{\mathbf{X}}\psi := \neg\mathbf{X}\neg\psi$. In the finite word case, syntactic cosafety (safety) formulas are constructed from **true**, **false**, a, $\neg a$, $\vee$, $\wedge$, and the temporal operators $\mathbf{U}$ and $\mathbf{X}$ ($\mathbf{R}$ and $\tilde{\mathbf{X}}$). Observe that $\mathbf{X}$ and $\tilde{\mathbf{X}}$ differ only on words of length 1, and thus the only required change in our translation scheme is to replace some $\mathbf{X}$s with $\tilde{\mathbf{X}}$s in the reachability formula 4. For finite words a translation of a counter-free DFA to an LTL formula with only a double exponential size blow-up is known [42]; however, unlike our translation, it does not guarantee syntactic safety (cosafety) formulas for safety (cosafety) languages.

Lastly, we provide a corollary on looping automata, using Theorem 2 and the following known result.

Proposition 9 (Rephrased Theorem 13 from [29]). *Let $\mathcal{D}$ be a deterministic looping-Büchi automaton with n states that recognises an LTL-definable language. Then there exists an equivalent counter-free deterministic looping-Büchi automaton $\mathcal{D}'$ with at most n states.*

Corollary 2. *Every deterministic looping-Büchi (looping-coBüchi) automaton with n states that recognises an LTL-definable language is equivalent to an LTL formula $\varphi \in \Pi_1$ (Σ_1) of temporal nesting depth in $O(2^{2^n})$ and length in $2^{2^{O(2^n)}}$.*

This is an elementary upper bound for two constructions for which either the upper bound was unknown or non-elementary: the liveness-safety decomposition of LTL [29] and the translation of semantic safety LTL to syntactic safety LTL.

5 Conclusions

We have studied the size trade-offs between LTL and automata. Over a unary alphabet, the situation is straightforward and we provided tight complexity bounds. The general case of infinite words over an arbitrary alphabet is more complex. We gave to our knowledge the first elementary complexity bound on the translation of counter-free deterministic ω-regular automata into LTL formulas.

Every ω-regular automaton recognising an LTL-definable language can be translated to a counter-free deterministic automaton [39, Theorem 3.2]. Yet, we are unaware of a bound on the size blow-up involved in such a translation. Once established, it can be combined with our translation to get a general bound on the translation of automata to LTL. It will also provide a (currently unknown[6]) elementary upper bound on the translation of LTL with both future and past operators to LTL with only future operators (which is the version of LTL that we have considered), as (both version of) LTL can be translated to nondeterministic Büchi automata with a single exponential size blow-up [41, Theorem 2.1].

While going from non-elementary to double-exponential depth and triple-exponential length is an improvement, these upper bounds might not be tight—there is currently no known non-linear lower bound! Closing this gap is a challenging open problem, which might require new lower bound techniques for alternating automata, as LTL formulas are an inherently alternating model.

Acknowledgements. We thank Moshe Vardi and Orna Kupferman for suggesting studying the succinctness gap between semantic and syntactic safe formulas, and Mikołaj Bojańczyk for answering our questions on algebraic automata theory.

[6] In consultation with the author of [30], we have confirmed that while the lower bound provided in that paper holds, the stated upper bound is erroneous.

References

1. Birget, J.C.: Two-way automata and length-preserving homomorphisms. Mathematical Systems Theory **29**(3), 191–226 (1996)
2. Bojańczyk, M.: Languages recognised by finite semigroups, and their generalisations to objects such as trees and graphs, with an emphasis on definability in monadic second-order logic (2020)
3. Boker, U.: Why these automata types? In: Proc. of LPAR. pp. 143–163 (2018)
4. Boker, U., Kupferman, O.: The quest for a tight translation of Büchi to co-Büchi automata. In: Fields of Logic and Computation, pp. 147–164. Springer (2010)
5. Boker, U., Lehtinen, K., Sickert, S.: On the translation of automata to linear temporal logic (2022), https://arxiv.org/abs/2201.10267, full version
6. Cerná, I., Pelánek, R.: Relating hierarchy of temporal properties to model checking. In: MFCS. Lecture Notes in Computer Science, vol. 2747, pp. 318–327. Springer (2003)
7. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. J. ACM **28**(1), 114–133 (Jan 1981)
8. Chang, E.Y., Manna, Z., Pnueli, A.: Characterization of temporal property classes. In: Kuich, W. (ed.) Automata, Languages and Programming, 19th International Colloquium, ICALP92, Vienna, Austria, July 13-17, 1992, Proceedings. Lecture Notes in Computer Science, vol. 623, pp. 474–486. Springer (1992)
9. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science **47**, 149–158 (1986)
10. Cohen, J., Perrin, D., Pin, J.E.: On the expressive power of temporal logic. Journal of computer and System Sciences **46**(3), 271–294 (1993)
11. Cohen-Chesnot, J.: On the expressive power of temporal logic for infinite words. Theoretical Computer Science **83**(2), 301–312 (1991)
12. Colcombet, T., Zdanowski, K.: A tight lower bound for determinization of transition labeled Büchi automata. In: International Colloquium on Automata, Languages, and Programming. pp. 151–162. Springer (2009)
13. Diekert, V., Gastin, P.: First-order definable languages. In: Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]. Texts in Logic and Games, vol. 2, pp. 261–306 (2008)
14. Eilenberg, S.: Automata, Languages, and Machines Volume B. Academic Press, Inc., USA (1976)
15. Etessami, K., Vardi, M.Y., Wilke, T.: First-order logic with two variables and unary temporal logic. Inf. Comput. **179**(2), 279–295 (2002)
16. Gabbay, D., Pnueli, A., Shelah, S., Stavi, J.: On the temporal analysis of fairness. In: Proc. of POPL. p. 163–173. New York, NY, USA (1980)
17. Geffert, V., Mereghetti, C., Pighizzini, G.: Complementing two-way finite automata. Information and Computation **205**(8), 1173–1187 (2007)
18. Kamp, J.A.W.: Tense logic and the theory of linear order. University of California, Los Angeles (1968)
19. Kupferman, O., Rosenberg, A.: The blowup in translating LTL to deterministic automata. In: Proc. of Model Checking and Artificial Intelligence. pp. 85–94 (2010)
20. Kupferman, O., Ta-Shma, A., Vardi, M.Y.: Counting with automata. In: Proc. of LICS (1999)
21. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. ACM Transactions on Computational Logic (TOCL) **2**(3), 408–429 (2001)

22. Ladner, R.E.: Application of model theoretic games to discrete linear orders and finite automata. Information and Control **33**(4), 281–303 (1977)
23. Leiss, E.: Succinct representation of regular languages by boolean automata. Theoretical computer science **13**(3), 323–330 (1981)
24. Löding, C.: Optimal bounds for transformations of ω-automata. In: Rangan, C.P., Raman, V., Ramanujam, R. (eds.) Foundations of Software Technology and Theoretical Computer Science. pp. 97–109. Springer Berlin Heidelberg, Berlin, Heidelberg (1999)
25. Maler, O.: On the Krohn-Rhodes cascaded decomposition theorem. In: Time for Verification, Essays in Memory of Amir Pnueli. Lecture Notes in Computer Science, vol. 6200, pp. 260–278. Springer (2010)
26. Maler, O., Pnueli, A.: Tight bounds on the complexity of cascaded decomposition of automata. In: Proc. of FOCS. pp. 672–682 (1990)
27. Maler, O., Pnueli, A.: On the cascaded decomposition of automata, its complexity and its application to logic. Unpublished. Available at: http://www-verimag.imag.fr/~maler/Papers/decomp.pdf (1994)
28. Manna, Z., Pnueli, A.: A hierarchy of temporal properties. In: PODC. pp. 377–410. ACM (1990)
29. Maretic, G.P., Dashti, M.T., Basin, D.A.: LTL is closed under topological closure. Inf. Process. Lett. **114**(8), 408–413 (2014)
30. Markey, N.: Temporal logic with past is exponentially more succinct. Bull. EATCS **79**, 122–128 (2003)
31. McNaughton, R., Papert, S.A.: Counter-Free Automata (MIT research monograph no. 65). The MIT Press (1971)
32. Michel, M.: Complementation is more difficult with automata on infinite words. CNET, Paris **15** (1988)
33. Muller, D.E., Saoudi, A., Schupp, P.E.: Weak alternating automata give a simple explanation of why most temporal and dynamic logics are decidable in exponential time. In: Proceedings Third Annual Symposium on Logic in Computer Science. pp. 422–423. IEEE Computer Society (1988)
34. Perrin, D.: Recent results on automata and infinite words. In: International Symposium on Mathematical Foundations of Computer Science. pp. 134–148. Springer (1984)
35. Safra, S.: Complexity of automata on infinite objects. Ph.D. thesis, Weizmann Institute, Rehovot, Israel (1989)
36. Schewe, S.: Büchi Complementation Made Tight. In: Albers, S., Marion, J.Y. (eds.) Proc. of 26th International STACS. Leibniz International Proceedings in Informatics (LIPIcs), vol. 3, pp. 661–672 (2009)
37. Sickert, S., Esparza, J.: An efficient normalisation procedure for linear temporal logic and very weak alternating automata. In: LICS. pp. 831–844. ACM (2020)
38. Thomas, W.: Star-free regular sets of ω-sequences. Information and Control **42**(2), 148–156 (1979)
39. Thomas, W.: A combinatorial approach to the theory of ω-automata. Information and Control **48**(3), 261–283 (1981)
40. Thomas, W.: Automata on infinite objects. In: Formal Models and Semantics, pp. 133–191. Elsevier (1990)
41. Vardi, M., Wolper, P.: An automata-theoretic approach to automatic program verification. In: Proc. of LICS. pp. 332–344 (1986)
42. Wilke, T.: Classifying discrete temporal properties. In: Annual symposium on theoretical aspects of computer science. pp. 32–46. Springer (1999)

43. Wilke, T.: Past, present, and infinite future. In: 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2016)
44. Wilke, T.: Backward deterministic Büchi automata on infinite words. In: 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2018)
45. Wolper, P.: Temporal logic can be more expressive **56**(1–2), 72–99 (1983)
46. Zuck, L.D.: Past Temporal Logic. Ph.D. thesis, The Weizmann Institute of Science, Israel (Aug 1986)