Size-Change Termination and Satisfiability for Linear-Time Temporal Logics

Martin Lange

School of Electr. Eng. and Computer Science, University of Kassel, Germany

Abstract. In the automata-theoretic framework, finite-state automata are used as a machine model to capture the operational content of temporal logics. Decision problems like satisfiability, subsumption, equivalence, etc. then translate into questions on automata like emptiness, inclusion, language equivalence, etc. Linear-time temporal logics like LTL, PSL and the linear-time μ-calculus have relatively simple translations into alternating parity automata, and this automaton model is closed under all Boolean operations with very simple constructions. Thus, the typical decision problems for such linear-time temporal logics reduce relatively simply to the emptiness problem for alternating parity automata. In this paper we present a method for decision this emptiness problem without going through intermediate automaton models like nondeterministic ones. The method is a direct adaptation of the size-change termination principle which was orgininally used to decide termination of abstract functional programs.

1 Introduction

Temporal logics are some of the most well-known and established tools for the specification of systems evolving in time. In computer science, they are mainly interesting as formal languages used to describe, reason about, analyse and verify program behaviour [7].

Temporal logics come in two different kinds depending on the nature of time underlying the models that they get interpreted about: linear-time and branching-time [20,21,25]. The viewpoint of linear time is that every moment in time has a unique successor, i.e. the future is determined. In branching time, a moment may have several successors. Here we only deal with linear time, namely we consider the well-known simple linear-time temporal logic LTL [16], as well as two of its extensions: a core of the industry standard property specification language PSL [2] which extends LTL with semi-regular expressions in order to remedy LTL's weakness of not being able to define all ω-regular properties; as well as the linear-time μ-calculus [22,3] which uses second-order quantification in the form of least and greatest fixpoints in order to obtain higher expressivity.

A prominent methodology for obtaining decision procedures is the automata-theoretic framework. It is particularly suitable for linear-time logics since their models can be seen as infinite words which immediately links logics and automata as two different specification formalisms for languages of such words. Logics are

C. Tinelli and V. Sofronie-Stokkermans (Eds.): FroCoS 2011, LNAI 6989, pp. 28–39, 2011.

often more natural to use as specification languages but automata are often closer to a decision procedure. Hence, translations from formulas to automata preserving their set of models are desirable, and they exist for the aforementioned logics.

It is known that every LTL formula can be translated into a nondeterministic Büchi automaton [26] with an expontial blow-up. The same holds for PSL [4] although the blow-up is in general doubly exponential. LT_μ can also be translated into such automata at a singly exponential blow-up only [26].

Translations into nondeterministic automata are particularly useful in order to decide satisfiability problems because satisfiability on the logical side corresponds to non-emptiness on the automata side, and non-emptiness problems for nondeterministic automata are usually solved via simple reachability problems on graphs. Other problems, however, in particular the universality and inclusion problem are as difficult for nondeterministic automata as the satisfiability problem for the corresponding temporal logics is. Note that on the logical side, problems like validity, subsumption and equivalence easily reduce to the satisfiability problem. This has led to the use of a richer automaton model: alternating automata. They typically enable a simple translation from formulas and on top of that a more difficult decision procedure as opposed to nondeterministic ones which usually come with a difficult translation and then simpler decision procedures. If "simple" means "polynomial" and "difficult" means "exponential" then the route via alternating automata may even be better in terms of complexity.

Translations from the linear-time temporal logics mentioned above into alternating automata are known [23,24,14,10]. In order to obtain decision procedures for these logics, one then only needs decision procedures for the corresponding problems on alternating automata. In fact, it suffices to be able to solve the emptiness problem for alternating automata just like it suffices to solve the satisfiability problem for temporal logics in order to solve all sorts of other problems through simple reductions.

The standard way to solve the emptiness problem for alternating automata has been using translations into nondeterministic automata. It may be the surprising simplicity of the Miyano-Hayashi construction [15] translating alternating Büchi into nondeterministic Büchi automata in comparison to the problem of turning a nondeterministic one into a deterministic one, that has put a brake onto research on different and possibly direct methods for the emptiness problem for alternating automata. This construction can be generalised to richer acceptance conditions like Streett automata [6], yet it still aims at translating into a nondeterministic model first.

Here we propose a different and direct method for the analysis of the emptiness problem for alternating parity automata. It originates from termination analysis for abstract functional programs and is called *size-change termination* (SCT) [13]. It is noted that the problem underlying this particular termination analysis can be solved by a reduction to the inclusion problem for nondeterministic Büchi automata but, since this requires complementation, SCT is proposed.

SCT has not passed unnoticed in the world of temporal logics and automata on infinite words: it has first been used to decide validity for the linear-time μ-calculus [5] and then prosed as a method to decide universality and inclusion for nondeterministic Büchi automata [8,9,1]. In fact, the first real work in this area is Büchi's original complementation proof for nondeterministic Büchi automata since the decision problems based on SCT use the same techniques. They are often called *Ramsey-based* because their correctness proof usually relies on the famous combinatorial Ramsey Theorem [17].

This paper is organised as follows. In Sect. 2 we recall the three important temporal logics mentioned above: LTL, PSL and the linear-time μ-calculus. In Sect. 3 we recall alternating parity automata and various subclasses thereof, and sketch how their emptiness problems characterise typical problems like satisfiability, subsumption, etc. for the temporal logics at hand. In Sect.4 we describe an SCT based method to decide emptiness of alternating parity automata.

2 Linear-Time Temporal Logics

2.1 Infinite Words

Let $\mathcal{P} = \{p, q, \ldots\}$ be a countably infinite set of *atomic propositions*. Linear-time temporal logics are interpreted over ω-sequences of sets of such propositions: an *infinite word* w is an element of $(2^{\mathcal{P}})^\omega$. A *finite word* is a $v \in (2^{\mathcal{P}})^*$. We write ϵ for the empty word of length 0, and $|v|$ in general for the length of the finite word v.

If w is a word $A_0 A_1 \ldots$ for $A_i \subseteq \mathcal{P}$ then $w(i)$ is used to denote A_i. We write $w(i,j)$ to denote the finite subsequence $A_i \ldots A_j$. Note that $w(i,j) = \epsilon$ if $j < i$.

2.2 LTL

One of the simplest and most well-known linear-time temporal logics is LTL. Its formulas are built from atomic propositions using Boolean operators and two temporal operators: the *next state* operator $\bigcirc$, and the *until* operator $\mathtt{U}$. Formulas of LTL are given by the following grammar.

$$\varphi \ ::= \ q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi \,\mathtt{U}\, \varphi$$

Formulas of LTL are interpreted in positions i of an infinite word as follows.

$$\begin{array}{lll}
w, i \models q & \text{iff} & q \in w(i) \\
w, i \models \varphi \wedge \psi & \text{iff} & w, i \models \varphi \text{ and } w, i \models \psi \\
w, i \models \neg\varphi & \text{iff} & w, i \not\models \varphi \\
w, i \models \bigcirc \varphi & \text{iff} & w, i+1 \models \varphi \\
w, i \models \varphi \,\mathtt{U}\, \psi & \text{iff} & \text{there is } j \geq i \text{ s.t. } w, j \models \psi \text{ and for all } h \text{ with } i \leq h < j \\
 & & \quad \text{we have } w, h \models \varphi
\end{array}$$

Further Boolean operators like disjunction $\varphi \vee \psi$, implication $\varphi \rightarrow \psi$, are defined as abbreviations in the usual way. Other temporal operators can also be derived, for instance $\mathtt{F}\,\varphi := \mathtt{true}\,\mathtt{U}\,\varphi$ ("finally") where $\mathtt{true} := q \vee \neg q$ for some $q \in \mathcal{P}$; $\mathtt{G}\,\varphi := \neg\,\mathtt{F}\,\neg\varphi$ ("generally"), etc.

We write $|\varphi|$ for the size of the formula φ, measured in terms of its DAG representation or, equivalently, the number of different subformulas.

Example 1. LTL can easily express properties concerning infinite occurrences of some atomic proposition in a word. For example, $\mathtt{G}\,\mathtt{F}\,q \wedge \mathtt{F}\,\mathtt{G}\,\neg p$ expresses that q holds in infinitely many positions and p holds almost everywhere, i.e. in all but finitely many positions.

A *model* for φ is a $w \in (2^{\mathcal{P}})^{\omega}$ s.t. $w, 0 \models \varphi$. We write $L(\varphi)$ for the set of all models of φ. The *satisfiability problem* is: given a formula φ, decide whether or not there is a model for it, i.e. whether or not $L(\varphi) \neq \emptyset$.

Proposition 1 ([19]). *The satisfiability problem for* LTL *is PSPACE-complete.*

We also describe two extensions of LTL in the following. They differ in their syntax and their semantics need more technicalities, but central concepts like formula size as well as set of models are defined as they are for LTL. Hence, we will not repeat them explicitly anymore.

2.3 PSL – An Extension of LTL

It is known that LTL cannot express counting properties like "q holds in every second position of a word" [27]. Note that this is an ω-regular property. There are several ways to overcome this weakness, for instance by introducing quantification over positions, i.e. by extending LTL with stronger logical connectives. PSL extends LTL with tools from the domain of formal languages, namely semi-extended regular expressions.

The language of all *Boolean expressions* over atomic propositions as above is given by the following grammar.

$$\zeta \;::=\; q \mid \zeta \wedge \zeta \mid \neg\zeta$$

Other Boolean operators can be defined as abbreviations. The satisfaction relation between a set $A \subseteq \mathcal{P}$ and a Boolean expression ζ is defined straightforwardly, i.e. $A \models \zeta$ iff ζ evaluates to 1 under the usual rules for Boolean connectives when all atomic propositions in A are set to 1 and all in $\mathcal{P} \setminus A$ are set to 0.

Semi-extended regular expressions (SERE) over Boolean expressions are built according to the following grammar.

$$\alpha \;::=\; \zeta \mid \alpha \cup \alpha \mid \alpha \cap \alpha \mid \alpha;\alpha \mid \alpha^*$$

where ζ is a Boolean expression as above. We write α^+ to abbreviate $\alpha;\alpha^*$.

A SERE is interpreted in a finite subword of an infinite $w \in (2^{\mathcal{P}})^\omega$ as follows. Note that such a subword is uniquely identified by two positions $i, j \in \mathbb{N}$ with $i \leq j$.

$$\begin{array}{ll}
w,i,j \models \zeta & \text{iff } i = j \text{ and } w(i) \models \zeta \\
w,i,j \models \alpha \cup \beta & \text{iff } w,i,j \models \alpha \text{ or } w,i,j \models \beta \\
w,i,j \models \alpha \cap \beta & \text{iff } w,i,j \models \alpha \text{ and } w,i,j \models \beta \\
w,i,j \models \alpha;\beta & \text{iff there is } h \text{ s.t. } i \leq h \leq j \text{ and } w,i,h \models \alpha \text{ and } w,h,j \models \beta \\
w,i,j \models \alpha^* & \text{iff there are } n \geq 0 \text{ and } h_0,\ldots,h_n \text{ s.t. } h_0 = i, h_n = j \text{ and} \\
 & \quad w,h_k,h_{k+1} \models \alpha \text{ for all } k = 0,\ldots,n-1
\end{array}$$

Note that in the first clause, satisfaction of a SERE in a finite word is reduced to satisfaction of a Boolean expression in a symbol of that word.

Formulas of PSL are then built by extending the syntax of LTL with operators that make use of SERE. Note that the standard of PSL [2] describes many operators for the logic; here we concentrate on two of them only – the "and then" and the "closure" operator. The constructions to follow can easily be extended to cover other PSL operators as well though.

$$\varphi \;::=\; q \mid \varphi \wedge \varphi \mid \neg\varphi \mid \varphi \,\mathtt{U}\, \varphi \mid \alpha \diamond\!\!\rightarrow \varphi \mid \mathtt{Cl}\,\alpha$$

The interpretation in positions of a word $w \in (2^{\mathcal{P}})^\omega$ extends the one for LTL given above by two clauses.

$$\begin{array}{ll}
w,i \models \alpha \diamond\!\!\rightarrow \varphi & \text{iff there is } j \geq i \text{ s.t. } w,i,j \models \alpha \text{ and } w,j \models \varphi \\
w,i \models \mathtt{Cl}\,\alpha & \text{iff for all } j \geq i \text{ exists } v \in (2^{\mathcal{P}})^* \text{ s.t. } w(i,j)v \models \alpha
\end{array}$$

Example 2. The aforementioned language $L = \{w \in (2^{\mathcal{P}})^\omega \mid \forall i \in \mathbb{N} : q \in w(2i)\}$ of all words in which q holds in every even position can be defined in PSL by the formula $\mathtt{Cl}(q;\mathtt{true})^*$. Equally, it is defined by $q \wedge \neg\big((\mathtt{true};\mathtt{true})^+ \diamond\!\!\rightarrow \bigcirc\neg q\big)$.

Proposition 2 ([4,12]). *The satisfiability problem for* PSL *is EXPSPACE-complete.*

The exponential increase in complexity compared to LTL is owed to the use of the intersection operator in semi-extended regular expressions. Note that these can be translated into nondeterministic finite automata (NFA) at a blow-up that is polynomial in the size of such an SERE but exponential in the nesting depth of the intersection operator. The logic obtained by replacing SERE with NFA is closely related to LTL with automata connective which also has a PSPACE-complete satisfiability problem [11].

2.4 The Linear-Time μ-Calculus

The Linear-Time μ-Calculus LT_μ is not directly an extension of LTL. It obtains ω-regular expressive power by adding least (finite iteration) and greatest (infinite

iteration) fixpoint operators to the fragment of LTL without the until operator. Let $\mathcal{V} = \{X, Y, \ldots\}$ be a countably infinite set of variable. Formulas of LT_μ are constructed as follows.

$$\varphi \ ::= \ q \mid X \mid \varphi \wedge \varphi \mid \neg\varphi \mid \bigcirc \varphi \mid \mu X.\varphi$$

where $q \in \mathcal{P}$ and $X \in \mathcal{V}$. We require that each formula is well-formed in the sense that every variable is bound by a binder mu at most once, and in every subformula $\mu X.\psi$ every occurrence of X is in the scope of an even number of negation symbols. For instance, $\mu X.\neg\mu Y.(\neg p \vee \bigcirc \neg Y) \wedge \bigcirc \neg X$ is well-formed because both variable occurrences are under two (different) nested negation operators.

Alongside the *least fixpoint quantifier* μ we introduce the *greatest fixpoint quantifier* ν via $\nu X.\varphi := \mu X.\neg\varphi[\neg X/X]$ where $\varphi[\psi/X]$ denotes the formula that is obtained from φ by replacing every free occurrence of X with ψ. Then the formula above can be written entirely without negation symbols as $\mu X.\nu Y.(p \wedge \bigcirc Y) \vee \bigcirc X$.

In order to interpret an LT_μ formula with free variables in a position of a word $w \in (2^{\mathcal{P}})^\omega$ we need the help of environments $\rho : \mathcal{V} \to 2^{\mathbb{N}}$. We write $\rho[X \mapsto P]$ for the environment that maps X to P and behaves like ρ on all other arguments.

$$\begin{aligned}
&w, i, \rho \models q && \text{iff } q \in w(i)\\
&w, i, \rho \models \varphi \wedge \psi && \text{iff } w, i, \rho \models \varphi \text{ and } w, i, \rho \models \psi\\
&w, i, \rho \models \neg\varphi && \text{iff } w, i, \rho \not\models \varphi\\
&w, i, \rho \models \mu X.\varphi && \text{iff } i \in \bigcap \big\{P \subseteq \mathbb{N} \mid P \supseteq \{j \mid w, j, \rho[X \mapsto P] \models \varphi\}\big\}
\end{aligned}$$

Example 3. The language of all words containing q in every even position can easily be defined in LT_μ: $\nu X.q \wedge \bigcirc\bigcirc X$.

The formula $\mu X.\nu Y.(p \wedge \bigcirc Y) \vee \bigcirc X$ mentioned above states that p holds in almost all positions. I.e. it is equivalent to the LTL formula $\mathtt{F\,G}\,p$.

Proposition 3 ([22]). *The satisfiability problem for* LT_μ *is PSPACE-complete.*

The reason for introducing LT_μ is the fact that it subsumes the two important temporal logics LTL and PSL. While this is also trivially true for PSL, LT_μ provides a clean (albeit not necessarily intuitive) syntax which is advantageous for the further treatment of these logics.

Proposition 4. *There are equivalence-preserving translations from ...*

- LTL *into* LT_μ *that incur a linear blow-up,*
- PSL *into* LT_μ *that incur a blow-up which is polynomial in the size of the formula and exponential in the nesting depth of the intersection operators [12].*

The translation from LTL into LT_μ is realised by the fact that $\varphi\,\mathtt{U}\,\psi$ can be expressed as $\mu X.\psi \vee (\varphi \wedge \bigcirc X)$. The translation from PSL is more complicated and uses the fact that SERE can be translated into NFA, as well as a close

resemblance between LT_μ formulas and automata. This close resemblance will be used in the following where we introduce alternating parity automata, a model of finite automata operating on infinite words that easily captures LT_μ and allows several problems on other automata to be regarded as emptiness problems.

3 Automata on Infinite Words

3.1 Alternating Parity Automata

We introduce a very powerful automaton model which captures many well-known models of automata on infinite words, including (non)deterministic Büchi and co-Büchi automata.

For a set M let $\mathbb{B}^+(M)$ denote that set of all positive Boolean formulas over M, i.e. the least set that contains M and satisfies: if $\{\varphi, \psi\} \subseteq \mathbb{B}^+(M)$ then $\{\varphi \wedge \psi, \varphi \vee \psi\} \subseteq \mathbb{B}^+(M)$.

A *alternating parity automaton* (APA) is a tuple $\mathcal{A} = (Q, \mathrm{AP}, q_0, \delta, \Omega)$ where Q is a finite set of states, AP is a finite subset of $\mathcal{P}$ as used in the previous section, $q_0 \in Q$ is a designated starting state, $\delta : Q \to \mathbb{B}^+(Q \cup \mathrm{AP} \cup \neg\mathrm{AP})$ with $\neg\mathrm{AP} := \{\neg q \mid q \in \mathrm{AP}\}$ is the transition function, and $\Omega : Q \to \mathbb{N}$ assigns priorities to the states.

Here we measure the size of an automaton, $|\mathcal{A}|$, as the number of its states.

A *run* of the APA $\mathcal{A}$ on a word $w \in (2^{\mathcal{P}})^\omega$ is a leveled DAG t whose nodes are labeled with states from Q, that has a single root on level 0, and the succesors of a node on level n are all on level $n + 1$. Furthermore, it obeys the following rules. We write $t(v)$ for the label of node v.

1. We have $t(v_0) = q_0$ for the root v_0.
2. Take any node v on some level n and let $u_1, \ldots, u_k$ be the set of its successors. Then we have $\{t(u_1), \ldots, t(u_k)\} \models \delta(t(v), w(n))$.

Such a run t is *accepting* if on every path through t the greatest priority of states that occur infinitely often is even. The *language* of $\mathcal{A}$ is $L(\mathcal{A}) = \{w \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$.

Example 4. Take the language $L = \{w \in (2^{\{p,q\}})^\omega \mid \text{if } q \text{ holds infinitely often, then } q \wedge p \text{ holds infinitely often in } w\}$. It is accepted by the APA $(\{0, 1, 2\}, \{p, q\}, 0, \delta, \Omega)$ where Ω is the identity function, and the transition function is defined for all three states i as

$$\delta(i) = (p \wedge q \wedge 2) \vee 1 \vee (\neg q \wedge 0)$$

It is also accepted by the APA $(\{0, 1, 2\}, \{p, q\}, 0, \delta, \Omega)$ where $\Omega(0) = \Omega(2) = 0$ and $\Omega(1) = 1$, and

$$\begin{aligned} \delta(0) &= (\neg q \vee p \vee 1) \wedge 0 \\ \delta(1) &= (q \wedge p \wedge 2) \vee 1 \\ \delta(2) &= 2 \end{aligned}$$

3.2 Subclasses of Alternating Parity Automata

Let $\mathcal{A} = (Q, \mathrm{AP}, q_0, \delta, \Omega)$ be an APA. Then $\mathcal{A}$ is an *alternating Büchi automaton* (ABA) if $\Omega : Q \to \{1, 2\}$. It is an *alternating co-Büchi automaton* (AcoBA) if $\Omega : Q \to \{0, 1\}$. I.e. Büchi acceptance is concerned with the infinite occurrence of some states whereas co-Büchi acceptance demands that certain states occur almost everywhere in a path of a run.

The literature contains different definitions of a weak automaton, sometimes constraining the graph structure of the automaton, sometimes weakening the acceptance condition by changing "occurs infinitely often" into "occurs" simply. These notions are equivalent though [14]. Here we consider the former. $\mathcal{A}$ is called *weak* (WAPA) if for all $q, q' \in Q$ s.t. q' occurs in $\delta(q)$ we have $\Omega(q') \leq \Omega(q)$. Hence, the priorities on every path in a run of a weak automaton are monotonically decreasing, and the largest priority that occurs infinitely often is automatically the one that occurs almost everywhere. Consequently, weak alternating automata can easily be defined as ABA or AcoBA.

$\mathcal{A}$ is *nondeterministic* if for all $q \in Q$ we have that $\delta(q)$ is a disjunction of minterms containing exactly one state, i.e. all other conjuncts are atomic propositions or negations thereof.

3.3 Constructions on Alternating Parity Automata

Alternating automata are closed under all Boolean operations.

Proposition 5. *Let $\mathcal{A}$ and $\mathcal{B}$ be two APA. There are APA*

1. *$\overline{\mathcal{A}}$ s.t. $L(\overline{\mathcal{A}}) = (2^{\mathcal{P}})^\omega \setminus L(\mathcal{A})$ and $|\overline{\mathcal{A}}| = |\mathcal{A}|$;*
2. *$\mathcal{A}_1$ s.t. $L(\mathcal{A}_1) = L(\mathcal{A}) \cup L(\mathcal{B})$ and $|\mathcal{A}_1| = \mathcal{O}(|\mathcal{A}| + |\mathcal{B}|)$;*
3. *$\mathcal{A}_2$ s.t. $L(\mathcal{A}_2) = L(\mathcal{A}) \cap L(\mathcal{B})$ and $|\mathcal{A}_2| = \mathcal{O}(|\mathcal{A}| + |\mathcal{B}|)$;*

Proof. (1) Let $\mathcal{A} = (Q, \mathrm{AP}, q_0, \delta, \Omega)$. We define $\overline{\mathcal{A}} = (Q, \mathrm{AP}, q_0, \overline{\delta}, \overline{\Omega})$ where $\overline{\Omega(q)} := \Omega(q) + 1$, and $\overline{\delta}(q) := \overline{\delta(q)}$ where $\overline{p} = \neg p$, $\overline{\neg p} = p$, $\overline{\zeta \vee \eta} = \overline{\zeta} \wedge \overline{\eta}$ and $\overline{\zeta \wedge \eta} = \overline{\zeta} \vee \overline{\eta}$. Clearly, the dual APA is not any bigger than its counterpart. A careful inspection reveals that it recognises the complement language.

(2+3) The APA $\mathcal{A}_1$ and $\mathcal{A}_2$ are obtained by taking the union of $\mathcal{A}$ and $\mathcal{B}$ and adding a new state q_0 with arbitrary priority and transitions obtained as the disjunction, resp. conjunction of the transitions of the two original starting states. □

This makes APA a rich model for various decision problems.

3.4 Decision Problems for Alternating Parity Automata

Important decision problems for automata are the following.

- Non-Emptiness: given an APA $\mathcal{A}$, is $L(\mathcal{A}) \neq \emptyset$?
- Universality: given an APA $\mathcal{A}$, is $L(\mathcal{A}) = (2^{\mathcal{P}})^\omega$?
- Subsumption: given APA $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) \subseteq L(\mathcal{B})$?
- Equivalence: given APA $\mathcal{A}$ and $\mathcal{B}$, is $L(\mathcal{A}) = L(\mathcal{B})$?

Thanks to Prop. 5 these problems are all interreducible in linear time. For instance, equivalence reduces to non-emptiness because $L(\mathcal{A}) \neq L(\mathcal{B})$ iff $L(\mathcal{C}) \neq \emptyset$ where $\mathcal{C}$ recognises $\big(L(\mathcal{A}) \cap L(\overline{\mathcal{B}})\big) \cup \big(L(\overline{\mathcal{A}}) \cap L(\mathcal{B})\big)$. Equally, non-emptiness and universality are interreducible because $L(\mathcal{A}) \neq \emptyset$ iff $L(\overline{\mathcal{A}}) \neq (2^{\mathcal{P}})^\omega$.

Note that these reductions need the full power of alternation as well as the full power of the parity acceptance condition unless they are weak. The dual of a Büchi automaton for instance is not a Büchi automaton but a co-Büchi automaton and vice-versa. Weakness is preserved by dualisation though. Also, the dual of a nondeterministic automaton is in general not a nondeterministic automaton anymore. However, regarding it as an alternating automaton enables easy dualisation.

3.5 Alternating Parity Automata and Temporal Logics

Many decision problems for linear-time temporal logics can be phrased as an emptiness problem for (a subclass) of alternating parity automata. The crucial ingredient for this is of course an equivalence-preserving translation from formulas to automata. With Prop. 4 above it suffices to check that LT_μ can be translated into alternating parity automata that way. However, translating logics like LTL and PSL separately can be benefitial because they may not need the full power of the parity acceptance condition.

Proposition 6 ([23,14,10,4]). *For every ...*

- LTL *formula* φ *there is a weak APA* $\mathcal{A}_\varphi$ *s.t.* $L(\mathcal{A}_\varphi) = L(\varphi)$ *and* $|\mathcal{A}_\varphi| = \mathcal{O}(|\varphi|)$*;*
- PSL *formula* φ *there is a weak APA* $\mathcal{A}_\varphi$ *s.t.* $L(\mathcal{A}_\varphi) = L(\varphi)$ *and* $|\mathcal{A}_\varphi| = 2^{\mathcal{O}(|\varphi|)}$*;*
- LT_μ *formula* φ *there is an APA* $\mathcal{A}_\varphi$ *s.t.* $L(\mathcal{A}_\varphi) = L(\varphi)$ *and* $|\mathcal{A}_\varphi| = \mathcal{O}(|\varphi|)$.

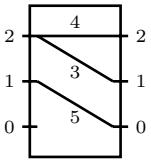
4 The Size-Change Termination Principle for Alternating Parity Automata

4.1 Boxes and Their Composition

We use $<$ to denote the usual total ordering on $\mathbb{N}$ or $\mathbb{Z}$, and introduce a second (non-well-founded) total ordering called *reward ordering*: $i \prec j$ iff $(-1)^i \cdot i < (-1)^j \cdot j$. I.e. we have $\ldots \prec 3 \prec 1 \prec 0 \prec 2 \prec \ldots$

For the remainder of this section we fix an APA $\mathcal{A} = (Q, \mathrm{AP}, q_0, \delta, \Omega)$. Let $P = \{\Omega(q) \mid q \in Q\}$ be the set of all priorities occurring in $\mathcal{A}$. We write $P_\perp$ for $P \cup \{\perp\}$ which will be used to model partial functions into P.

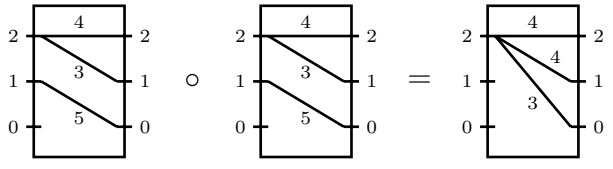
A *box* is an element of type $Q \times Q \to P_\perp$. The name suggest a particular visual representation of such a function. We regard the elements of Q as ports of a circuit, and a box has in- and out-ports which can be connected by an edge labeled with a number. For instance, if $Q = \{0, 1, 2\}$ and $P = \{3, 4, 5\}$ then the following is a box over Q.



We define a *composition* operation $\circ$ on two boxes f, g by

$$(f \circ g)(q, p) = \max_{\prec}\{\max_{<}\{f(q, q'), g(q', p)\} \mid q' \in Q\}$$

Here we use the important convention that $\bot$ is the *maximal* element w.r.t. $<$, but the *minimal* element w.r.t. $\prec$. Then box composition works as one would expect it from the graphical representation.



Box composition is lifted to sets of boxes in the natural way: $F \circ G := \{f \circ g \mid f \in F, g \in G\}$.

With every finite word $v \in (2^{\mathrm{AP}})^\omega$ we associate a set of boxes $[v]$ as follows: $[\epsilon]$ contains only a single box f which is defined by $f(q, q') = \Omega(q)$ if $q' = q$ and $f(q, q') = \bot$ otherwise. Furthermore, for every one-letter word $a \subseteq \mathrm{AP}$ we have that $f \in [a]$ if for all $q, q' \in Q$:

1. $f(q, q') \in \{\Omega(q), \bot\}$, and
2. $\{p \mid f(q, p) \neq \bot\} \cup a \models \delta(q)$

Intuitively, a box belongs to $[a]$ if it connects every in-port to all the out-ports in some model that agrees with a on the atomic propositions and their negations. The labels on the connections simply reflect the priority of the in-port.

Using composition it is easy to define $[v]$ for longer words v: $[av] := [a] \circ [v]$. We write $[\mathcal{A}^*]$ for $\{[v] \mid v \in (2^{\mathrm{AP}})^*$ and $[\mathcal{A}^+]$ for $\{[v] \mid v \in (2^{\mathrm{AP}})^+$. Note that these are finite sets.

A box f is called *idempotent* if $f \circ f = f$. It is called *good* w.r.t. some $Q' \subseteq Q$ if for all $q \in Q'$ we have that $f(q, q)$ is even.

4.2 Characterising the Emptiness Problem for Alternating Parity Automata

Theorem 1. *$L(\mathcal{A}) \neq \emptyset$ iff there are $f \in [\mathcal{A}^*]$ and $g \in [\mathcal{A}^+]$ s.t. g is idempotent and good w.r.t. $\{q \mid f(q_0, q) \neq \bot\}$.*

Proof. "$\Leftarrow$" Suppose such $f, g \in [\mathcal{A}]$ exist. Then there must be words $v_f = a_1 \dots a_n$ and $v_g = b_1 \dots b_m$ with $n \geq 0$, $m \geq 1$ s.t. $f \in [v_f]$ and $g \in [v_g]$. I.e. there must be $f_1 \in [a_1], \dots, f_n \in [a_n], g_1 \in [b_1], \dots, g_m \in [b_m]$ s.t. $f =$

$f_1 \circ \ldots \circ f_n$ and $g = g_1 \circ \ldots \circ g_m$. A run on $v_f(v_g)^\omega$ can easily be extracted from the sequence $f_1, \ldots, f_n, g_1, \ldots, g_m, g_1, \ldots$ by following all connections starting from q_0. Suppose this run was not accepting. Then it would contain a path on which the highest priority seen infinitely often is odd. Note that idempotency of g means that this path is compressed into a connection from some q' to itself in this box. Furthermore, q' must be reachable from q_0 in f. But then $g(q', q')$ must be odd which contradicts the assumption that g is good.

"$\Rightarrow$" Suppose that $L(\mathcal{A}) \neq \emptyset$, i.e. there is a $w \in (2^{\mathrm{AP}})^\omega$ s.t. $w \in L(\mathcal{A})$. Then there is an accepting run t of $\mathcal{A}$ on w. Let $w = a_0 a_1 \ldots$ The run t can easily be transformed into a sequence of boxes $f_0, f_1, \ldots$ by possibly adding nodes to each level s.t. every state is present on every level, and adding corresponding connections. Now consider a colouring of all ordered pairs of levels i, j with $i \leq j$, assigning to this pair the box $f_{i,j} := f_i \circ \ldots \circ f_j$. Since there are only finitely many boxes, Ramsey's Theorem [18] gives us an infinite sequence $j_0 < j_1 < \ldots$ of indices s.t. all pairs of indices from this sequence get assigned to the same box. In particular we have $f_{j_0,j_1} = f_{j_1,j_2} = f_{j_0,j_2} = f_{j_0,j_1} \circ f_{j_1,j_2}$ which shows that it is idempotent. Define the required boxes as $f := f_{0,j_0}$ and $g := f_{j_0,j_1}$. What remains to be seen is that g is good w.r.t. the set of all states that q_0 is connected to in f. As above, suppose it was not. Then the run would have contained an infinite path on which the highest priority occurring infinitely often was odd which would contradict the assumption that it was accepting. □

References

1. Abdulla, P.A., Chen, Y.-F., Clemente, L., Holík, L., Hong, C.-D., Mayr, R., Vojnar, T.: Simulation subsumption in ramsey-based büchi automata universality and inclusion testing. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 132–147. Springer, Heidelberg (2010)
2. Inc. Accellera Organization. Formal semantics of Accellera property specification language (2004), In Appendix B of `http://www.eda.org/vfv/docs/PSL-v1.1.pdf`
3. Banieqbal, B., Barringer, H.: Temporal logic with fixed points. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 62–73. Springer, Heidelberg (1989)
4. Bustan, D., Fisman, D., Havlicek, J.: Automata constructions for PSL. Technical Report MCS05-04, The Weizmann Institute of Science (2005)
5. Dax, C., Hofmann, M., Lange, M.: A proof system for the linear time μ-calculus. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 274–285. Springer, Heidelberg (2006)
6. Dax, C., Klaedtke, F.: Alternation elimination by complementation (Extended abstract). In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 214–229. Springer, Heidelberg (2008)
7. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science. Formal Models and Semantics, vol. B, ch. 16, pp. 996–1072. Elsevier, MIT Press, New York, USA (1990)
8. Fogarty, S., Vardi, M.Y.: Büchi complementation and size-change termination. In: Kowalewski, S., Philippou, A. (eds.) TACAS 2009. LNCS, vol. 5505, pp. 16–30. Springer, Heidelberg (2009)

9. Fogarty, S., Vardi, M.Y.: Efficient büchi universality checking. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 205–220. Springer, Heidelberg (2010)
10. Kaivola, R.: Using Automata to Characterise Fixed Point Temporal Logics. PhD thesis, LFCS, Division of Informatics, The University of Edinburgh, Tech. Rep. ECS-LFCS-97-356 (1997)
11. Kupferman, O., Piterman, N., Vardi, M.Y.: Extended temporal logic revisited. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 519–535. Springer, Heidelberg (2001)
12. Lange, M.: Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 90–104. Springer, Heidelberg (2007)
13. Lee, C.S., Jones, N.D., Ben-Amram, A.M.: The size-change principle for program termination. ACM SIGPLAN Notices 36(3), 81–92 (2001)
14. Löding, C., Thomas, W.: Alternating automata and logics over infinite words. In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) TCS 2000. LNCS, vol. 1872, pp. 521–535. Springer, Heidelberg (2000)
15. Miyano, S., Hayashi, T.: Alternating finite automata on omega-words. TCS 32(3), 321–330 (1984)
16. Pnueli, A.: The temporal logic of programs. In: Proc. 18th Symp. on Foundations of Computer Science, FOCS 1977, pp. 46–57. IEEE, Providence (1977)
17. Ramsey, F.P.: On a problem of formal logic. Proc. London Mathematical Society, Series 2 30(4), 338–384 (1928)
18. Ramsey, F.P.: On a problem in formal logic. Proc. London Math. Soc. 30(3), 264–286 (1930)
19. Sistla, A.P., Clarke, E.M.: The complexity of propositional linear temporal logics. Journal of the Association for Computing Machinery 32(3), 733–749 (1985)
20. Stirling, C.: Comparing linear and branching time temporal logics. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 1–20. Springer, Heidelberg (1989)
21. Vardi.: Linear vs. branching time: A complexity-theoretic perspective. In: LICS: IEEE Symposium on Logic in Computer Science (1998)
22. Vardi, M.Y.: A temporal fixpoint calculus. In: ACM (ed.) Proc. Conf. on Principles of Programming Languages, POPL 1988, pp. 250–259. ACM Press, New York (1988)
23. Vardi, M.Y.: Alternating automata and program verification. In: van Leeuwen, J. (ed.) Computer Science Today. LNCS, vol. 1000, pp. 471–485. Springer, Heidelberg (1995)
24. Vardi, M.Y.: An Automata-Theoretic Approach to Linear Temporal Logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)
25. Vardi, M.Y.: Branching vs. Linear time: Final showdown. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 1–22. Springer, Heidelberg (2001)
26. Vardi, M.Y., Wolper, P.: Reasoning about infinite computations. Information and Computation 115(1), 1–37 (1994)
27. Wolper, P.: Temporal logic can be more expressive. Information and Control 56, 72–99 (1983)