How to Efficiently Translate Extensions of Temporal Logics into Alternating Automata

César Sánchez[1,2] and Julian Samborski-Forlese[1]

[1] IMDEA Software Institute, Madrid, Spain
[2] Institute for Applied Physics, CSIC, Madrid, Spain
{cesar.sanchez}@imdea.org

Abstract. This paper presents studies efficient and general translations of extensions of linear temporal logic (LTL) into alternating automata, which can be applied to improve algorithms for the automata-theoretic approach to model-checking. In particular, we introduce—using a game theoretic framework—a novel finer grain complementation theorem for the parity condition. This result enables simple and efficient translations of temporal operators into pairs of automata accepting complement languages, using up to 3 colors. Moreover, our results: (1) allows to translate directly operators from LTL and different extensions (2) that can be combined without restriction; and (3) does not require to eliminate negation upfront, or to start from formulas in negation normal form.

1 Introduction

We study the problem of temporal verification of reactive systems, that starts from a finite description of a system and a specification given in linear temporal logic [11, 9]. The problem consists in deciding whether all runs of the systems are accepted by the specification. The *automata-theoretic approach* to model checking [14, 15] allows to reduce this verification problem to automata constructions (like product and complementation) and automata decision problems (like non-emptiness and language containment). First, one builds an automaton on infinite words for the negation of the formula. Then, this automaton is composed synchronously with the system. Finally, an emptiness check is used to conclude whether the resulting product admits some trace (counter-example) or the systems is correct with respect to the specification.

In modern incarnations of the automata approach to model checking, specifications are translated into alternating automata. Their richer structure of alternating automata enables a more direct translation than non-deterministic automata, and allows to postpone a potentially exponential blow-up. Another advantage of alternation is the easy dualization (see Muller and Schupp [10]) provided by the availability of both conjunctive and disjunctive transition relations. However, to obtain an automaton accepting the complement language, one also needs to complement the acceptance condition (see, for example [13]).

In this paper we study complementation constructions for parity automata and applications to translate formulas from LTL and extensions into automata more efficiently. The parity acceptance condition allows a simple and well-known complementation construction: increment the color assigned to every state. This operation preserves the relative order between the colors of any two states, and inverts the parity of the maximum color in any given sequence of states. This way, accepting traces become non-accepting traces, and non-accepting traces become accepting traces. Even though this construction is simple and elegant, it suffers the drawback that the number of colors in the resulting automaton grows with every complementation step. If this construction is used to translate the logical negation operator, the total number of colors used in the resulting automaton can grow linearly in the size of the formula. The best known algorithms for alternating parity automata becomes less efficient as the number of colors grow. For example, translating an alternating parity automaton with n states and k colors into non-deterministic Büchi automaton [3] requires $O(2^{nk\ log\ nk})$. Hence, many researchers [7] have suggested translations of LTL into automata with weaker acceptance conditions enabled by an upfront manipulation of formulas in the logical level. Unfortunately, some extensions of LTL, particularly operators that use regular expressions, preclude the use of these simpler forms of automata.

In this paper we alleviate the problem of the inefficient translation into automata with the parity condition by exploiting the following intuition. The classical parity complementation construction complements all sequence of states in the automaton, while only a subset of these sequences can be exercise by the automaton. We show that the set of traces of an automaton and its dual are identical, and that to complement an automaton it is enough to provide a pair of parity assignment with opposite outcomes on these traces. The second contribution of this paper is a translation of temporal logics based on our complementation results. Each operator is translated into a pair of complement automata, starting from a pair of complement automata for the operands.

The rest of the of paper is structured as follows. Section 3 presents the notion of specular automata pairs, and show that they accept complement languages. Section 4 shows translations of some temporal operators from LTL and extensions into specular automata pairs. Finally, Section 5 concludes.

2 Preliminars

Positive Boolean Formulas: We use $\mathcal{B}^+(X)$ for the positive boolean formulas over a set of propositions X. These formulas are built from **true**, **false** and elements of X, combined using $\land$ and $\lor$. A model of a formula θ is a subset of X that makes θ true. A minimal model M of a formula θ is a model of θ such that no strict subset of M is a model of θ. For example, given the set $Q = \{q_0, q_1, q_2, q_3\}$, the formula $\theta_1 = (q_1 \land q_2) \lor q_3$ is a $\mathcal{B}^+(Q)$ formula. The sets $\{q_1, q_2\}$ and $\{q_3\}$ are the minimal models of θ_1. We use $Mod(\theta)$ for the set of models of θ and $mod(\theta)$ for the set of minimal models.

Every positive boolean formula can be expressed in disjunctive normal form, as disjunction of conjunctions of propositions. Given a positive boolean formula θ there is a dual formula $\widetilde{\theta}$ obtained by switching $\land$ and $\lor$, and switching **true** and **false**. Some easy properties of dual formulas are:

Proposition 1 (Duals). *For every θ and $\widetilde{\theta}$, and for every $M \in Mod(\theta)$:*

1. *For every $M' \in Mod(\widetilde{\theta})$, $M \cap M' \neq \varnothing$.*
2. *Let $q \in M$. There is an M' in $Mod(\widetilde{\theta})$ with $q \in M'$.*

For example, the dual of θ_1 above is $\widetilde{\theta_1} = (q_1 \vee q_2) \wedge q_3$, or equivalently in disjunctive normal form $\widetilde{\theta_1} = (q_1 \wedge q_3) \vee (q_2 \wedge q_3)$. The minimal models of $\widetilde{\theta_1}$ are $\{q_1, q_3\}$ and $\{q_2, q_3\}$.

A *choice function* is a map f that chooses, for a model M of θ an element of M, i.e., $f : Mod(\theta) \to X$ such $f(M) \in M$. Some interesting properties of choice functions follow:

Proposition 2 (Choice Functions). *Let θ be a formula and $\widetilde{\theta}$ its dual. Then*

1. *If f is a choice function for θ, then $Img\ f \in Mod(\widetilde{\theta})$.*
2. *If $M \in mod(\theta)$ then there is a choice function f of $\widetilde{\theta}$ such that $Img\ f = M$.*

Clearly, not every choice function has a minimal model as image (2.1 states that it must be a model but not necessarily minimal). Those choice functions whose images are minimal models are called *proper choice functions*. We will later focus our attention to proper choice functions as strategies for players in certain classes of parity games.

3 Specular Automata Pairs

Alternating Frames We study now the layout of alternating automata. An automaton *frame*, or simply a frame, is a tuple $\mathcal{F} : \langle \Sigma, Q, \delta, I \rangle$ where Σ is an alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$ is the transition function, and $I \in \mathcal{B}^+(Q)$ is the initial condition of the frame. A frame determines which are the legal traces for a given automaton and input word. We will later introduce automata as frames equipped with an acceptance condition, which will determine which traces of the frame are good. A frame is *non-deterministic* whenever I, and $\delta(q, a)$ for all states q and input symbols a, have singleton sets as minimal models. In other words, I and $\delta(q, a)$ are equivalent to disjunctive formulas. A frame is called *universal* if I, and $\delta(q, a)$ for all states q and symbols a, have a unique minimal model. In other words, I and $\delta(q, a)$ are equivalent to conjunctive formulas. A frame is deterministic if it is both non-deterministic and universal, that is if both the initial condition and transition functions correspond to **true**, **false** or a single successor state. In general a frame is neither universal nor non-deterministic, but fully alternating.

Run and trace Given a word $w \in \Sigma^\omega$, a *run* of w on a frame $\mathcal{F} : \langle \Sigma, Q, \delta, I \rangle$ is a DAG (V, E) with nodes $V \subseteq Q \times \mathbb{N}$, such that:

1. The nodes $(m, 0) \in V$ form a minimal model of I.
2. for every (q, k) in V, there is a minimal model M of $\delta(q, w[k])$ such that E contains an edge $(q, k) \to (q', k + 1)$ for all q' in M.

A *trace* of a run is an infinite path in the run, following edges. A non-deterministic frame may admit multiple different runs for a given word, but each run contains a unique trace. A universal frame admits just one run for each word, but this run may contain multiple traces. In general a frame admits multiple runs each with multiple traces.

Given a frame $\mathcal{F} : \langle \Sigma, Q, \delta, I \rangle$, the *specular frame* is the frame $\widetilde{\mathcal{F}} : \langle \Sigma, Q, \widetilde{\delta}, \widetilde{I} \rangle$, where $\widetilde{I}$ is the dual formula of I and $\widetilde{\delta}$ is the dual transition function: $\widetilde{\delta}(q, a)$ is the dual formula of $\delta(q, a)$ for all states q and symbols a.

Frame Graphs We define the *graph* of a frame $\mathcal{F}$ as $(V_\mathcal{F}, E_\mathcal{F})$ where $V_\mathcal{F} = Q$ and there is an edge in $E_\mathcal{F}$ from $p \to q$ whenever q is in some minimal model of $\delta(p, a)$ for some symbol a. Since the union of all minimal models of a formula is the same set as the union of all minimal models of its dual formula, the edge relation is the same for the graph of a frame and its specular frame:

Proposition 3 (Frame Graphs). *The graph of a frame and the graph of its specular frame are identical.*

By construction, if $(p, k) \to (q, k + 1)$ is an edge in some run of a given frame, then $p \to q$ is an edge in the graph of the frame. Consequently, the set of traces of runs of a frame correspond to the set of walks in the graph, which in turn is also the set of traces of runs of the specular frame.

Automata A frame $\mathcal{F} : \langle \Sigma, Q, \delta, I \rangle$ can be enriched into an automaton $\mathcal{A} : \langle \Sigma, Q, \delta, I, F \rangle$ by adding an acceptance condition F. In this paper we will use the *parity* acceptance condition defined by a map $F : Q \to \{0 \dots d\}$. Let π be an infinite sequence of states $\pi : q_0, q_1, q_2 \dots$ and $\mathit{inf}(\pi)$ be those states that occur infinitely many times in π. The sequence π is accepting according to F, which we denote $\pi \in \mathit{acc}(F)$, whenever the maximum value that occurs infinitely often in $F(\pi)$ is even:

$$\mathit{max}\{F(q) \mid q \in \mathit{inf}(\pi)\} \text{ is even}$$

A *run* of a word w on an automaton $\mathcal{A}$ is a run on its frame. A run is called accepting whenever all its traces are accepting sequences. We say that a word w is in the language of automaton $\mathcal{A}$, and we write $w \in \mathcal{L}(\mathcal{A})$ whenever there is an accepting run for w on $\mathcal{A}$.

Definition 1 (Specular Automata). *Two automata $\mathcal{A} : \langle \Sigma, Q, \delta, I, F_A \rangle$ and $\mathcal{B} : \langle \Sigma, Q, \widetilde{\delta}, \widetilde{I}, F_B \rangle$ with specular frames are specular automata whenever for all paths π in the frame graph:*

$$\pi \in \mathit{acc}(F_A) \qquad \textit{if and only if} \qquad \pi \notin \mathit{acc}(F_B).$$

The standard construction for complementing alternating parity automata consists in creating the specular frame by dualizing the initial condition and transition functions, and making $F_B(q) = F_A(q) + 1$ for every q. However, in many cases it is possible to exploit the particular structure of $\mathcal{A}$ to define lower values for F_B.

Automata and Games We show now that specular automata accept complement languages, using game theory. From a given automaton $\mathcal{A}$ and a word w, we create a parity game called a *word game* as a tuple $\mathbf{G}(\mathcal{A}, w) : \langle V_A, V_P, E_A, E_P, f\rangle$ where:

$$\begin{aligned} V_A =& Q \times \omega \\ V_P =& \{(M, q, i) \mid M \in \mathit{Mod}(\delta(q, w[i]))\} \;\cup\; \{(M, \cdot, 0) \mid M \in \mathit{Mod}(I)\} \\ E_A =& (q, i) \to (M, q, i) \;\text{ for each } M \in \mathit{Mod}(\delta(q, w[i])) \\ E_P =& (M, q, i) \to (q', i + 1) \;\text{ for } q' \in M \end{aligned}$$

The game is played by two players: *Automaton* (A) and *Pathfinder* (P). The set of positions $V = V_A \cup V_P$ is partitioned into positions in which A plays and those in which P plays. The game begins by A choosing a model of I, which determines the initial position $(M, \cdot, 0)$ (here $\cdot$ represent an irrelevant state). The legal moves of the game are captured by the relation $E = E_A \cup E_P$ which correspond to A choosing a model from a V_A position, and P choosing the next successor from a given model from a V_P position. A *play* is an infinite sequence of positions $\pi : V_0 v_0 V_1 v_1 \ldots$ with V_0 being an initial position, v_i obtained from V_i by a P move, and V_{i+1} obtained from v_i by an A move. The map $f : V \to \{0 \ldots d\}$ determines the outcome of a play. We define the *trace* of a play $\pi : V_0 v_0 V_1 v_1 \ldots$ as the sequence of states $\mathit{trace}(\pi) : p_0 p_1 \ldots$ obtained by projecting the first component of the V_P positions of the play (i.e., $v_i = (p_i, i)$). The following follows directly from the definition:

Proposition 4. *Every trace of a play of* $\mathbf{G}(\mathcal{A}, w)$ *is also a trace of some run of* $\mathcal{A}$ *on* w.

As for parity automata the outcome of a play is determined by the highest color that is seen infinitely often in the play. Player A wins play π whenever:

$$\mathit{max}\{f(q) \mid q \in \mathit{inf}(\mathit{trace}(\pi))\} \text{ is even}$$

Otherwise, P wins play π. A strategy for player A is a map $\rho_A : (V^* V_A \cup \epsilon) \to V$, that maps histories of positions into moves. Here, ϵ denotes the empty sequence of positions, to let player A choose an initial state in the game. A memoryless strategy simply takes into account the last position: $\rho_A : V_A \cup \epsilon \to V$. Since parity games are memoryless determined [4] it is enough to consider memoryless strategies. Similarly, a strategy for player P is a map $\rho_P : V_P \to V$. A play $\pi : V_0 v_0 V_1 v_1 \ldots$ is played according to strategy ρ_A whenever the initial position is $V_0 = \rho_A(\epsilon)$ and all moves of A are played according to it $V_i = \rho_A(v_i)$. A strategy ρ_A is winning for player A whenever all plays played according to ρ_A are winning for A. Memoryless determinacy of parity games guarantees that either player A has a memoryless winning strategy or player P has a memoryless winning strategy. We say that π is a $G \cdot \rho_A$ play whenever π is played in G according to ρ_A.

We restrict our attention to strategies for A that choose minimal models, and strategies for P that are proper choice functions. This is not a drastic restriction.

Clearly, if there is a winning strategy for A that does not choose a minimal model, then any strategy that chooses a smaller minimal model is also winning. This is because the set of plays is reduced, and all plays in the unrestricted set are winning for A. Similarly, if ρ_P is a winning strategy for P, then restricting its moves to be a proper choice functions (by restricting the image) also gives a winning strategy. In both cases, the set of successor moves is restricted but still confined within winning regions. This lemma is essentially Prop. 2 from [13], where complementation of weak alternation automata by dualization is studied.

Lemma 1. *$w \in \mathcal{L}(\mathcal{A})$ if and only if A has a winning strategy in $\boldsymbol{G}(\mathcal{A}, w)$.*

Specular Pairs and Complementation We show in this section that specular automata accept complement languages. In the rest of the section we let $\mathcal{A}$ and $\widetilde{\mathcal{A}}$ be a specular automata pair, w be a word and $G : \mathbf{G}(\mathcal{A}, w)$ and $\widetilde{G} : \mathbf{G}(\widetilde{\mathcal{A}}, w)$ be the corresponding word games. First we need some preliminary definitions.

Definition 2. *We say that strategies ρ_A (for A in G) and $\widetilde{\rho}_P$ (for P in $\widetilde{G}$) are duals whenever both:*

- *for every $G \cdot \rho_A$ play π there is a $\widetilde{G} \cdot \widetilde{\rho}_P$ play $\widetilde{\pi}$ s.t. $trace(\widetilde{\pi}) = trace(\pi)$.*
- *for every $\widetilde{G} \cdot \widetilde{\rho}_P$ play $\widetilde{\pi}$ there is a $G \cdot \rho_A$ play π s.t. $trace(\widetilde{\pi}) = trace(\pi)$.*

Theorem 1 (Dual Strategies). *The following holds:*

(1) For every strategy ρ_A for A in G, there is a dual strategy $\widetilde{\rho}_P$ for P in $\widetilde{G}$.
(2) For every ρ_P for P in G, there is a dual strategy $\widetilde{\rho}_A$ for A in $\widetilde{G}$.

The following theorem follows directly from Lemma 1 and Theorem 1. This theorem allows to reason about complementation simply by reasoning about traces of two automata with dual frames.

Theorem 2. *Let $\mathcal{A}$ and $\widetilde{\mathcal{A}}$ be specular automata. Then $\mathcal{L}(\mathcal{A}) = \Sigma^\omega \setminus \mathcal{L}(\widetilde{\mathcal{A}})$.*

4 Temporal Logic to Specular Automata

We show in this section how the results in Section 3 can be used to translate temporal logic expressions into alternating parity automata. Most previous translations fix the logic first, and then show a translation from the whole expression into automata. Typically, these translations are preceded by a previous transformation of the expression into negation normal form, by pushing the negation operator to the propositional level. This transformation requires the logics to enjoy duality laws for all operators, or in other words, to admit a negation normal form. Using this transformation, the negation operator need not be considered in the translation into automata.

We follow here a different approach. For each operator we construct a specular automata pair: one automaton is equivalent to the expression, and another equivalent to its complement. The construction for a given operator starts from a specular automata pair for each of the operands. This approach has two advantages. First, negation becomes trivial. Second, adding more constructs to a logic simply requires the addition of specific translations for the added operand. In this manner, operators from different logics can be easily combined. We present here a few examples of constructs from LTL and some extensions.

Linear Temporal Logic: Linear temporal Logic was introduced by Pnueli [11], see also [9]. We consider here the following operators

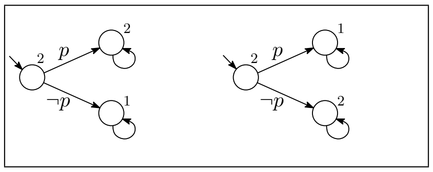
$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \Box\varphi \mid \Diamond\varphi \mid \varphi\,\mathcal{U}\,\varphi \mid \varphi\,\mathcal{R}\,\varphi \mid \varphi\,\mathcal{W}\,\varphi$$

This definition is not minimal but it serves to illustrate how to translate some of the operators into APW. The semantics of LTL expressions are defined using a binary relation $\vDash$ between pointed ω-words and LTL expressions:

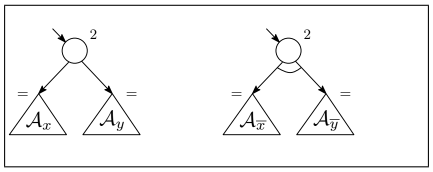
- $(w, i) \vDash p$ when $p \in w[j]$.
- $(w, i) \vDash \neg x$ when $(w, i) \nvDash x$.
- $(w, i) \vDash x \vee y$ when $(w, i) \vDash x$ or $(w, i) \vDash y$ or both.
- $(w, i) \vDash \bigcirc x$ when $(w, i+1) \vDash x$.
- $(w, i) \vDash \Diamond x$ when $(w, j) \vDash x$ for some $j \geq i$.
- $(w, i) \vDash \Box x$ when $(w, j) \vDash x$ for all $j \geq i$.
- $(w, i) \vDash x\,\mathcal{U}\,y$ when $(w, j) \vDash y$ for some $j \geq i$, and $(w, k) \vDash x$ for all $i \leq k < j$.
- $(w, i) \vDash x\,\mathcal{R}\,y$ when $(w, j) \vDash y$ for all $j \geq i$, or
  for some j, $(w, j) \vDash x$ and for all k in $i \leq k \leq j$, $(w, k) \vDash y$.
- $(w, i) \vDash x\,\mathcal{W}\,y$ when $(w, j) \vDash x$ for all $j \geq i$, or
  $(w, j) \vDash y$ for some $j \geq i$, and $(w, k) \vDash x$ for all $i \leq k < j$.

We now show the translation of each operator. We assume a pair of dual automata $(\mathcal{A}_x, \mathcal{A}_{\overline{x}})$ for each operand x.

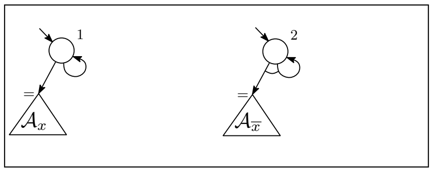
- p: The automaton $\mathcal{A}_p$ is $\langle Q, \delta, I, F\rangle$ such that $Q = \{q_0, q_1, q_2\}$, $I = q_0$, $F(q_0) = F(q_1) = 2$ and $F(q_2) = 1$. The transitions function is: $\delta(q_0, p) = q_1$, $\delta(q_0, \neg p) = q_2$, and $\delta(q_1, \cdot) = q_1$ and $\delta(q_2, \cdot) = q_2$ are self loops. The dual automaton $\mathcal{A}_{\overline{p}}$ has, as final condition, $F(q_0) = F(q_1) = 1$, $F(q_2) = 2$. Note, in particular, how the dualization of the acceptance condition is not performed by incrementing the color of each state, but by reasoning independently about the traces. The automata pair is shown graphically in Fig. 1(a).
- $\neg x$: the automaton for $\mathcal{A}_{\neg x}$ is $\mathcal{A}_{\overline{x}}$ and the automaton for $\mathcal{A}_{\overline{\neg x}}$ is $\mathcal{A}_x$.
- $x \vee y$: The automaton $\mathcal{A}_{x\vee y}$ is $\langle Q, \delta, I, F\rangle$ with $Q = Q_x \cup Q_y$, and $I = I_x \vee I_y$. The acceptance condition works as F_x for Q_x and as F_y for Q_y. For $\mathcal{A}_{\overline{x\vee y}}$, the automaton is build similarly, but from $\mathcal{A}_{\overline{x}}$ and $\mathcal{A}_{\overline{y}}$ with $I = I_{\overline{x}} \wedge I_{\overline{y}}$. The construction is depicted in Fig. 1(b)
- $\Box x$: The automaton $\mathcal{A}_{\Box x}$ has $Q = \{q_0\} \cup Q_x$, where q_0 is a fresh state. The initial condition is $I = \{q_0\}$. The acceptance condition works as F_x in Q_x, and assigns $F(q_0) = 1$. Finally, $\delta(q_0, a) = q_0 \wedge \delta_x(I_x, a)$. The dual automaton $\mathcal{A}_{\overline{\Box x}}$ is built analogously, from $\mathcal{A}_{\overline{x}}$, except: $F(q_0) = 2$ and $\delta(q_0, a) = q_0 \vee \delta_{\overline{x}}(I_{\overline{x}}, a)$.
- $\Diamond x$: The construction is exactly the dual as for $\Box x$. Hence, given $(\mathcal{A}_x, \mathcal{A}_{\overline{x}})$ the automata obtained for $\mathcal{A}_{\Diamond x}$ is identical to $\mathcal{A}_{\overline{\Box x}}$, and $\mathcal{A}_{\overline{\Diamond x}}$ is identical to $\mathcal{A}_{\Box x}$. This construction directly proves the duality of $\Diamond$ and $\Box$.



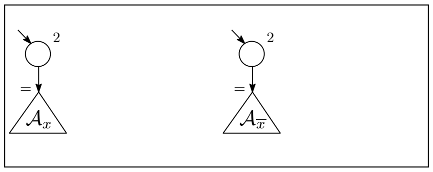
(a). Automata pair for p.



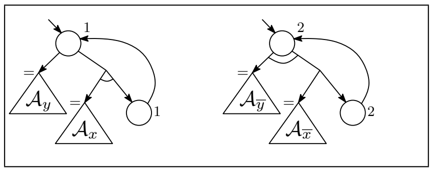
(b). Automata pair for $x \vee y$.



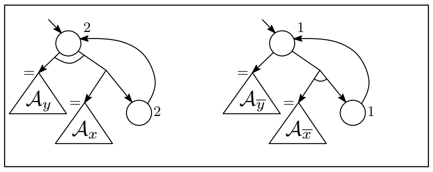
(c). Automata pair for $\Diamond x$.



(d). Automata pair for $\bigcirc x$.



(e). Automata pair for $x\,\mathcal{U}\,y$.



(f). Automata pair for $x\,\mathcal{R}\,y$.

Figure 1. Translations of LTL ito APW (1,2)

- $x\,\mathcal{U}\,y$: The automaton $\mathcal{A}_{x\mathcal{U}y}$ has $Q = \{q_0\} \cup Q_x \cup Q_y$ and $I = \{q_0\}$. The acceptance condition is such that $F(q_0) = 1$, and as F_x for states in Q_x, and F_y for states in Q_y. The transition function, maps $\delta(q_0, a) = \delta(I_y, a) \vee (\delta(I_x, a) \wedge q_0)$; for states in Q_x and Q_y, δ is as δ_x and δ_y. The dual automaton is constructed analogously, except that $\overline{F}(q_0) = 2$ and $\overline{\delta}(q_0, a) = \delta_{\overline{y}}(I_{\overline{y}}, a) \wedge (\delta_{\overline{x}}(I_{\overline{x}}, a) \vee q_1)$. This case illustrates again how colors need not be increased in the dualization. The only trace to be considered when incrementally proving the correctness (accepting complement languages) of $\mathcal{A}_{x\mathcal{U}y}$ and $\mathcal{A}_{\overline{x\mathcal{U}y}}$ is the infinite sequence $q_0q_0q_0\ldots$, which is accepting for $\mathcal{A}_{x\mathcal{U}y}$ and rejecting for $\mathcal{A}_{\overline{x\mathcal{U}y}}$. The other traces follow from the inductive construction.
- $x\,\mathcal{R}\,y$: The construction is exactly dual as for $x\,\mathcal{U}\,y$, which illustrates the duality between $\mathcal{U}$ and $\mathcal{R}$.
- $x\,\mathcal{W}\,y$: The construction is as for $\mathcal{U}$, except that $F(q_0) = F(q_1) = 2$ for $\mathcal{A}_{x\mathcal{W}y}$ and $\overline{F}(q_0) = \overline{F}(q_1) = 1$ for $\mathcal{A}_{\overline{x\mathcal{W}y}}$.

This translations are depicted graphically in Fig. 1. In all cases, the APW generated uses only two colors: 1 and 2. Every APW (1,2) automaton is a Büchi automaton: traces will be accepted if at least one 2 state is visited infinitely often. Also, by looking at the automata graph, we see that all even valued states can be assigned any even value, and all odd states can be assigned any odd value, because every trace will still have the same acceptance outcome. Hence, choosing 0 instead of 2 in all steps of the inductive construction will produce an APW with colors 0 and 1, which is a co-Büchi automaton. Our construction avoids the need to complement the Büchi automaton obtained from the formula for model-checking (or to negate the formula upfront and convert into negation normal form).

Regular Linear Temporal Logic We sketch here an incremental construction for operators of Regular Linear Temporal Logic RLTL [8, 12]. RLTL is a logic that fuses regular expressions and temporal operators in a single formalism. RLTL is defined in two stages: the first stage consists of a variation of regular expressions over finite words, using

$$\alpha ::= p \mid \alpha + \alpha \mid \alpha \,;\, \alpha \mid \alpha^{*}\alpha$$

We assume that a non-deterministic finite automaton of linear size is constructed from a given regular expression. The second stage defines temporal logic expressions that describe languages over infinite words, using regular expressions as building blocks. Since regular expressions are used to later build temporal expressions, the semantics for regular expressions are defined to accept *segments* of infinite words. Given an infinite word w and two positions i and j, the tuple (w, i, j) is called a segment of the word w. The syntax of RLTL expressions is defined by the following grammar:

$$\varphi ::= \varnothing \mid \varphi \vee \varphi \mid \neg\varphi \mid \alpha \,;\, \varphi \mid \varphi|\alpha\rangle\!\rangle\varphi \mid \varphi|\alpha\rangle\varphi$$

where α ranges over regular expressions. The symbol ; stands for the conventional concatenation of an expression over finite words followed by an expression over infinite words. The operator $\varnothing$ represents the empty language.

The operators $\varphi|\alpha\rangle\!\rangle\varphi$ and its weak version $\varphi|\alpha\rangle\varphi$ are the power operators. The power expressions $x|r\rangle\!\rangle y$ and $x|r\rangle y$ (read x *at* r *until* y, and, respectively, x *at* z *weak-until* y) are built from three elements: y (the *attempt*), x (the *obligation*) and r (the *delay*). Informally, for $x|r\rangle\!\rangle y$ to hold, either the attempt holds, or the obligation is met and the whole expression evaluates successfully after the delay; in particular, for a power expression to hold the obligation must be met after a finite number of delays. On the contrary, $x|r\rangle y$ does not require the obligation to be met after a finite number of delays. These two simple operators allow the construction of many other operators like $x\,\mathcal{U}\,y$ and r^{ω}, which make RLTL ω-complete. Also, for every LTL operator there is a RLTL operator with the same number of operands, and consequently LTL can be translated linearly into RLTL. The semantics of the new RLTL operands $\varnothing$, $r;x$, $r|x\rangle\!\rangle y$ and $r|x\rangle y$ is:

- $(w,i) \vDash \varnothing$ never holds.
- $(w,i) \vDash r\,;y$ when for some k, $(w,i,k) \vDash_{\mathrm{RE}} r$ and $(w,k) \vDash y$
- $(w,i) \vDash x|r\rangle\!\rangle y$ when $(w,i) \vDash y$ or for some $(i_0 = i, i_1, \ldots i_m)$, and for all $k < m$ $(w, i_k, i_{k+1}) \vDash_{\mathrm{RE}} r$ and $(w, i_k) \vDash x$, and $(w, i_m) \vDash y$
- $(w,i) \vDash x|r\rangle y$ when one of:
  (*i*) $(w,i) \vDash y$.
  (*ii*) for some $(i_0 = i, i_1, \ldots i_m)$, $(w, i_m) \vDash y$, and $(w, i_k, i_{k+1}) \vDash_{\mathrm{RE}} r$ and $(w, i_k) \vDash x$ for all $k < m$.
  (*iii*) for some inf. seq. $(i_0 = i, i_1, \ldots)$, $(w, i_k, i_{k+1}) \vDash_{\mathrm{RE}} r$ and $(w, i_k) \vDash x$

We show now the translations of RLTL into APW. The operators $\vee$ and $\neg$ can be reused from LTL. We assume again that we have the automata pair $(\mathcal{A}_x, \mathcal{A}_{\overline{x}})$ for all operands x, and a non-deterministic automaton N_r for each regular expression r.

- $\varnothing$: The automaton $\mathcal{A}_\varnothing$ consists of a single state $Q = \{q_0\}$ with $I = \{q_0\}$ and a self-loop $\delta(q_0, a) = q_0$. The accepting condition maps $F(q_0) = 1$. The dual automaton $\mathcal{A}_{\overline{\varnothing}}$ is identical except that $F(q_0) = 2$.
- $r;x$: The automaton for $\mathcal{A}_{r;x}$ consists of $Q = Q_r \cup Q_x$, and $I = I_r$. The transition function is as in $\mathcal{A}_x$ for states Q_x; for states q in Q_r:
  - if $\delta_r(q,a) \cap F_r = \emptyset$, then $\delta(q,a) = \bigvee \delta_r(q,a)$.
  - if $\delta_r(q,a) \cap F_r \neq \emptyset$, then $\delta(q,a) = \bigvee \delta_r(q,a) \vee I_x$.

  This allows δ to non-deterministically jump to x when an accepting segment is matched by r. Finally, the acceptance condition is $F(q) = F_x(q)$ for all states in Q_x and $F(q) = 1$ for all states in Q_r. Hence, a trace that remains in Q_r is a non-accepting trace.

  The automaton for $\mathcal{A}_{\overline{r;x}}$ is built from N_r and $\mathcal{A}_{\overline{x}}$: $Q = Q_r \cup Q_{\overline{x}}$. The transition function now interprets the transitions from states in Q_r universally:
  - if $\delta_r(q,a) \cap F_r = \emptyset$, then $\delta(q,a) = \bigwedge \delta_r(q,a)$.
  - if $\delta_r(q,a) \cap F_r \neq \emptyset$, then $\delta(q,a) = \bigwedge \delta_r(q,a) \wedge I_{\overline{x}}$.

  Finally, $F(q) = F_{\overline{x}}(q)$ for q in Q_x and $F(q) = 2$ for q in Q_r. Note how a trace that gets trapped in Q_r is now accepting, and how the frame corresponding to the regular expression r is universal.
- $x|r\rangle\rangle y$: The set of states is $Q = Q_x \cup Q_y \cup Q_r \cup \{q_0\}$. The initial condition is $I = \{q_0\}$. The transition function is as δ_x for states in Q_x, as δ_y for states in Q_y. For states q in Q_r:
  - if $\delta_r(q,a) \cap F_r = \emptyset$ then $\delta(q,a) = \bigvee \delta_r(q,a)$.
  - if $\delta_r(q,a) \cap F_r \neq \emptyset$ then $\delta(q,a) = \bigvee \delta_r(q,a) \vee q_0$.

  For q_0:
  - if $\delta_r(I_r,a) \cap F_r = \emptyset$ then $\delta(q_0,a) = \delta_y(I_y,a) \vee (\delta_x(I_x,a) \wedge \delta_r(I_r,a))$.
  - if $\delta_r(q,a) \cap F_r \neq \emptyset$ then $\delta(q_0,a) = \delta_y(I_y,a) \vee (\delta_x(I_x,a) \wedge (\delta_r(I_r,a) \vee q_0))$.

  The acceptance condition is $F(q) = F_x(q)$ for q in Q_x, $F(q) = F_y(q)$ for q in Q_y, and $F(q_0) = F(q) = 1$ for q in Q_r.

  The dual automaton $\mathcal{A}_{\overline{x|r\rangle\rangle y}}$ is built analogously. The transition function is dual of $\mathcal{A}_{x|r\rangle\rangle y}$. The acceptance condition is $F(q) = F_{\overline{x}}(q)$ for q in $Q_{\overline{x}}$, $F(q) = F_{\overline{y}}(q)$ for q in $Q_{\overline{y}}$, and $F(q_0) = F(q) = 2$ for q in $Q_{\overline{r}}$.
- $x|r\rangle y$: The set of states Q, the initial state I and the transition function δ are like in $x|r\rangle\rangle y$. The acceptance condition is $F(q) = F_x(q)$ for q in Q_x, $F(q) = F_y(q)$ for q in Q_y, and $F(q_0) = 2$, $F(q) = 1$ for q in Q_r. This makes traces that visit q_0 infinitely often accepting, but traces that get trapped in r rejecting.

  The dual automaton $\mathcal{A}_{\overline{x|r\rangle\rangle y}}$ is built with a dual frame and $F(q_0) = 1$, and $F(q) = 0$ for $q \in Q_r$. Color increasing was prevented by reasoning about traces independently.

Note how the automata obtained in this translation is now an APW (0,1,2). Still, the automata obtained has some particular structured. All strongly connected components (SCCs) are either labeled with 0 and 1, or labeled with 1 and 2. This is not a weak acceptance condition but can be potentially used to improved the translation into NBW further (this is ongoing work).

PSL operators The logic PSL [5] and its precursors ForSpec [1] and Sugar [2], also combine regular expressions with temporal operators. We illustrate here how to translate the PSL operator $r \mapsto x$, assuming that r is a regular expression as defined above. The semantics of $r \mapsto x$ and $r \Diamond\!\!\rightarrow x$ is:

– $(w, i) \vDash r \mapsto x$ when there is a j with $(w, i, j+1) \vDash r$ and $(w, j) \vDash x$.
– $(w, i) \vDash r \Diamond\!\!\rightarrow x$ when for all j with $(w, i, j+1) \vDash r$, then $(w, j) \vDash x$.

We sketch the translation from $r \mapsto x$ and $r \Diamond\!\!\rightarrow x$ into specular APW pairs:

– $r \mapsto x$: The automaton for $\mathcal{A}_{r \mapsto x}$ consists of $Q = Q_r \cup Q_x$, and $I = I_r$. The transition function is as in $\mathcal{A}_x$ for states Q_x; for states q in Q_r:
  - if $\delta_r(q, a) \cap F_r = \emptyset$, then $\delta(q, a) = \bigvee \delta_r(q, a)$.
  - if $\delta_r(q, a) \cap F_r \neq \emptyset$, then $\delta(q, a) = \bigvee \delta_r(q, a) \vee \delta_x(I_x, a)$.

  This allows δ to non-deterministically jump to x when an accepting segment is matched by r, overlapping the last state. Finally, the acceptance condition is $F(q) = F_x(q)$ for all states in Q_x and $F(q) = 1$ for all states in Q_r. Hence, a trace that remains in Q_r is a non-accepting trace.
  The automaton for $\mathcal{A}_{\overline{r \mapsto x}}$ is built dually. For the accepting condition: $F(q) = F_{\overline{x}}(q)$ for q in Q_x and $F(q) = 2$ for q in Q_r. Note how a trace that gets trapped in Q_r is now accepting, and how the frame corresponding to the regular expression r is universal.
– $r \Diamond\!\!\rightarrow x$ is dual of $r \mapsto x$.

Dynamic Linear Temporal Logic DLTL DLTL is defined as a dynamic logic in [6]. DLTL introduces a generalized until operator $x \, \mathcal{U}^r \, y$ that constraints those points at which the attempt y can be evaluated by successful matches of the regular expression r. In order for $x \, \mathcal{U}^r \, y$ to be satisfied, there must be a segment met by regular expression r after which y is satisfied, and x must be satisfied in all the positions until the successful match of r. More formally:

– $(w, i) \vDash x \, \mathcal{U}^r \, y$ when there is a j with $(w, i, j) \vDash r$ and $(w, j) \vDash y$,
  and for all k within $i \leq k < j$, $(w, k) \vDash x$.

The translation into APW is:

– $x \, \mathcal{U}^r \, y$: The set of states $Q = Q_x \cup Q_y \cup Q_r \cup \{q_0\}$. The initial state is $I = q_0$. The transition function is like δ_x for states in Q_x and like δ_y for states in Q_y. For q_0:
  - if $\delta_r(I_r, a) \cap F_r = \emptyset$ then $\delta(q_0, a) = \delta_y(I_y, a) \vee (\delta_x(I_x, a) \wedge \delta_r(I_r, a))$.
  - if $\delta_r(I_r, a) \cap F_r \neq \emptyset$ then $\delta(q_0, a) = \delta_y(I_y, a) \vee (\delta_x(I_x, a) \wedge (\delta_r(I_r, a) \vee q_0))$.

  For states $q \in Q_r$:
  - if $\delta_r(q, a) \cap F_r = \emptyset$ then $\delta(q, a) = (\delta_x(I_x, a) \wedge \delta_r(q, a)$.
  - if $\delta_r(q, a) \cap F_r \neq \emptyset$ then $\delta(q, a) = (\delta_x(I_x, a) \wedge (\delta_r(q, a) \vee q_0)$.

The acceptance condition is $F(q) = F_x(q)$ for q in Q_x, $F(q) = F_y(q)$ for q in Q_y, and $F(q_0) = 1$, $F(q) = 1$ for q in Q_r. The dual automaton $\mathcal{A}_{\overline{x\mathcal{U}^r y}}$ is built analogously, with the dual frame and $F(q_0) = 0$, and $F(q) = 0$ for $q \in Q_r$.

5 Conclusions

In this paper we have presented a finer grain complementation construction for alternating automata with the parity condition. This complementation allows to reason about walks in the graph of the specular automata pair, which are the only potential traces of runs. In turn, we showed how this results can be used to inductively translate temporal logic into APW. The translation of each operator produces a specular automata pair: one for the expression, one for its complement. This construction generates APW with few colors (2 for most expressions, 3 for the most sophisticated), which enables its efficient translation into non-deterministic Büchi Automata for model-checking. Future work includes the design of antichain algorithms directly for the APW generated from temporal logic expressions to alleviate even further the state explosion.

References

1. A. Armando, S. Ranise, and M. Rusinowitch. A rewriting approach to satisfiability procedures. *Information and Computation*, 183(2):140–164, 2003.
2. I. Beer, S. Ben-David, C. Eisner, D. Fisman, A. Gringauze, and Y. Rodeh. The temporal logic Sugar. In *CAV'01*, pages 363–367. Springer, 2001.
3. C. Dax and F. Klaedtke. Alternation elimination by complementation. In *LPAR'08*, volume 5530 of *LNCS*, pages 214–229. Springer, 2008.
4. E. A. Emerson and C. S. Jutla. Tree automata, mu-calculus and determinacy. In *FOCS'91*, pages 368–377. IEEE Computer Society, 1991.
5. D. Fisman, C. Eisner, and J. Havlicek. *Formal syntax and Semantics of PSL: App. B of Accellera Property Language Ref. Manual, v1.1*, March 2004.
6. J. G. Henriksen and P. S. Thiagarajan. Dynamic linear time temporal logic. *Annals of Pure and Applied Logic*, 96(1–3):187–207, 1999.
7. O. Kupferman and M. Y. Vardi. Weak alternating automata are not that weak. *ACM Transactions on Computational Logic*, 2(3):408–429, 2001.
8. M. Leucker and C. Sánchez. Regular linear temporal logic. In *ICTAC'07*, volume 4711 of *LNCS*, pages 291–305. Springer, September 2007.
9. Z. Manna and A. Pnueli. *Temporal Verification of Reactive Systems*. Springer, 1995.
10. D. E. Muller and P. E. Schupp. Altenating automata on infinite trees. *TCS*, 54:267–276, 1987.
11. A. Pnueli. The temporal logic of programs. In *FOCS'77*, pages 46–67, 1977.
12. C. Sánchez and M. Leucker. Regular linear temporal logic with past. In *VMCAI'10*, volume 5944 of *LNCS*, pages 295–311. Springer, 2010.
13. W. Thomas. Complementation of Büchi automata revisited. In *Jewels are Forever*, pages 109–120. Springer, 1999.
14. M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *LICS'86*, pages 332–344. IEEE CS Press, 1986.
15. M. Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115:1–37, 1994.

A Missing Proofs

Lemma 1. *$w \in \mathcal{L}(\mathcal{A})$ if and only if A has a winning strategy in $\boldsymbol{G}(\mathcal{A}, w)$.*

Proof. Assume $w \in \mathcal{L}(\mathcal{A})$ and let $\sigma : (V_\sigma, E_\sigma)$ be a successful run of w on $\mathcal{A}$. We first build a strategy ρ_A for A on $\mathbf{G}(\mathcal{A}, w)$ and then show that ρ_A is winning:

$$\begin{aligned} \rho_A(\epsilon) &= (M_0, \cdot, 0) && \text{with } M_0 = \{q \mid (q, 0) \in V_\sigma\} \\ \rho_A(q, i) &= (M, q, i+1) && \text{with } M = \{q' \mid (q, i) \to (q', i+1) \in E_\sigma\} \end{aligned}$$

The set M in $(M, q, i+1)$ is a model of $\delta(q, i)$ because σ is a run. For positions (q, i) that do not appear in the run σ, the strategy $\rho_A(q, i) = (M, q, i+1)$ can assign any model M in $Mod(\delta(q, w[i]))$. This model is not relevant because no play played according to ρ_A will visit these states. Consider now an arbitrary play $\pi : V_0 v_0 V_1 v_1 \dots$ of $\mathbf{G}(\mathcal{A}, w)$ played according to ρ_A. We show by induction that $trace(\pi) : p_0 p_1 \dots$ is a trace of σ.

– *base*: By construction M_0 is the set of initial positions of σ. Since p_0, chosen by player P, is $v_0 \in M_0$, then v_0 is a prefix of a trace of run σ.
– *induction step*: assume $p_0 \dots p_i$ is a prefix of some trace in σ, so (p_i, i) is in V_σ. Hence, $\rho_A(p_i, i) = (M, p_i, i+1)$ for M being the set of successors of (p_i, i) in E_σ. Consequently $p_{i+1} = (q, i+1)$ for some $(p_i, i) \to (p_{i+1}, i+1)$ in E_σ, so $v_0 \dots v_i v_{i+1}$ is a longer prefix of a trace of run in σ.

This shows that $trace(\pi)$ is a trace of the run σ. Now, since σ is a successful run all its traces must be accepting, and then:

$$max\{F(q) \mid q \in inf(trace(\pi)\} \text{ is even,}$$

which shows that ρ_A is a winning strategy for $\mathbf{G}(\mathcal{A}, w)$.

We now show the other direction: we start from a winning strategy ρ_A for A in $\mathbf{G}(\mathcal{A}, w)$ and show that there is a successful run σ of w on $\mathcal{A}$. Let $(M, \cdot, 0) = \rho_A(\epsilon)$. Then we let V_σ contain $(q, 0)$ for all $q \in M$. Note that M is a minimal model of I. Now, consider an arbitrary position (q, i) and let $(M, q, i+1)$ be $\rho_A(q, i)$. We add to E_σ all pairs of the form $(q, i) \to (q', i+1)$ for all $q' \in M$. We have to show that σ is successful run. We show by induction that all traces of σ correspond to plays in $\mathbf{G}(\mathcal{A}, w)$ played according to ρ_A. For the base case $(q, 0)$ is the initial state of the trace. By construction $(q, 0) \in \rho_A(\epsilon)$ so $(q, 0)$ is a possible choice of player P, and consequently a play prefix. For the inductive case, assume that trace prefix $(q_0, 0) \dots (q_i, i)$ is a play prefix, and let $(q_i, i) \to (q_{i+1}, i+1)$ be in E_σ. By construction $\rho_A(q_i, i)$ contains position $(q_{i+1}, i+1)$ so player P can again move to it. This shows that the arbitrary trace of σ correspond to a play played according to ρ_A. □