Parametric Linear Dynamic Logic*

Peter Faymonville Martin Zimmermann
Reactive Systems Group, Saarland University, 66123 Saarbrücken, Germany
{faymonville, zimmermann}@react.uni-saarland.de

We introduce Parametric Linear Dynamic Logic (PLDL), which extends Linear Dynamic Logic (LDL) by temporal operators equipped with parameters that bound their scope. LDL was proposed as an extension of Linear Temporal Logic (LTL) that is able to express all ω-regular specifications while still maintaining many of LTL's desirable properties like an intuitive syntax and a translation into non-deterministic Büchi automata of exponential size. But LDL lacks capabilities to express timing constraints. By adding parameterized operators to LDL, we obtain a logic that is able to express all ω-regular properties and that subsumes parameterized extensions of LTL like Parametric LTL and PROMPT-LTL.

Our main technical contribution is a translation of PLDL formulas into non-deterministic Büchi word automata of exponential size via alternating automata. This yields a PSPACE model checking algorithm and a realizability algorithm with doubly-exponential running time. Furthermore, we give tight upper and lower bounds on optimal parameter values for both problems. These results show that PLDL model checking and realizability are not harder than LTL model checking and realizability.

1 Introduction

Linear temporal logic (LTL) is a popular specification language for the verification and synthesis of reactive systems. It provides semantic foundations for industrial logics like PSL [5]. LTL has a number of desirable properties contributing to its ongoing popularity: it does not rely on the use of variables, it has an intuitive syntax and thus gives a way for practitioners to write declarative and concise specifications. Furthermore, it is expressively equivalent to first-order logic over the natural numbers with successor and order [10] and enjoys an exponential compilation property: one can efficiently construct a language-equivalent non-deterministic Büchi automaton of exponential size in the size of the specification. The exponential compilation property yields a **PSPACE** model checking algorithm and a **2EXPTIME** algorithm for realizability. Both problems are complete for the respective classes.

Model checking of properties described in LTL or its practical descendants is routinely applied in industrial-sized applications, especially for hardware systems [2, 5]. Due to its complexity, the realizability problem has not reached industrial acceptance (yet). First approaches used a determinization procedure for ω-automata, which is notoriously hard to implement efficiently [16]. More recent algorithms for realizability follow a safraless construction [6, 7], which avoids explicitly constructing the deterministic automaton, and are showing promise on small examples.

Despite the desirable properties, two drawbacks of LTL remain and are tackled by different approaches in the literature: first, LTL is not able to express all ω-regular properties. For example, the property "p holds on every even step" (but may or may not hold on odd steps) is not expressible in LTL, but easily expressible as an ω-regular expression. This drawback is a serious one, since the combination of regular properties and linear-time operators is common in hardware verification languages. Several

*This work was partially supported by the German Research Foundation (DFG) as part of SFB/TR 14 "AVACS".

Adriano Peron and Carla Piazza (Eds.):
Proceedings of the Fifth International Symposium on
Games, Automata, Logics and Formal Verification (GandALF 2014)
EPTCS 161, 2014, pp. 60–73, doi:10.4204/EPTCS.161.8

extensions of LTL [12, 20, 21] with regular expressions, finite automata, or grammar operators have been proposed as a remedy.

A second drawback of classic temporal logics like LTL is the inability to natively express timing constraints. The standard semantics are unable to enforce the fulfillment of eventualities within finite time bounds, e.g., it is impossible to require that requests are granted within a fixed, but arbitrary, amount of time. While it is possible to unroll an a-priori fixed bound for an eventuality into LTL, this requires prior knowledge of the system's granularity and incurs a blow-up when translated to automata, and is thus considered impractical. A more practical way of fixing this drawback has been the purpose of a long line of work in parametric temporal logics, such as parametric LTL [1], PROMPT–LTL [11] and parametric metric interval temporal logic [9]. All of them add parameters to the temporal operators to express time bounds, and either test the existence of a global time bound, like PROMPT–LTL, or of individual bounds on the parameters, like parametric LTL.

Recently, the first drawback was revisited by De Giacomo and Vardi [4, 19] by introducing an extension of LTL called linear dynamic logic (LDL), which is as expressive as ω-regular languages. The syntax of LDL is inspired by propositional dynamic logic (PDL) [8], but the semantics follow linear-time logics. In PDL and LDL, programs are expressed by regular expressions with tests, and temporal requirements are specified by two basic modalities: $\langle r \rangle \varphi$ and $[r]\varphi$, stating that φ should hold at some position where r matches, or at all positions where r matches, respectively. The operators to specify regular expressions from propositional formulas are as follows: sequential composition $(r_1 ; r_2)$, nondeterministic choice $(r_1 + r_2)$, repetition (r^*), and test $(\varphi?)$ of a temporal formula. On the level of the temporal operators, conjunction and disjunction are allowed. The tests allow to check temporal properties within programs, and are needed to encode LTL into LDL.

As an example, the program "**while** q **do** a" with property p holding after the execution of the loop is expressed in PDL/LDL as follows: $[(q?;a)^*;\neg q?]p$. Intuitively, the loop condition q is tested on every loop entry, the loop body a is executed/consumed until $\neg q$ holds, and then the post-condition p has to hold. A request-response property (i.e., every request should eventually be followed by a response) can be formalized as follows: $[\mathtt{tt}^*](\mathit{req} \rightarrow \langle \mathtt{tt}^* \rangle \mathit{resp})$.

Both aforementioned drawbacks of LTL, the inability to express all ω-regular properties and the missing capability to specify timing constraints, have been tackled individually in a successful way in previous work, but not at the same time. Here, we propose a logic called PLDL that combines the expressivity of LDL with the parametricity of PLTL on infinite traces.

In PLDL, we are for example able to parameterize the eventuality of the request-response condition, denoted as $[\mathtt{tt}^*](\mathit{req} \rightarrow \langle \mathtt{tt}^* \rangle_{\leq x} \mathit{resp})$, which states that every request has to be followed by a response within x steps. In the PLDL model checking problem, we determine whether there exists a valuation $\alpha(x)$ for x such that all paths of the system respond to requests within $\alpha(x)$ steps. If we take the property as a specification for the PLDL realizability problem, and define *req* as input, *resp* as output, we compute whether there exists a winning strategy that adheres to a valuation $\alpha(x)$ and is able to ensure the delivery of responses to requests in a timely manner.

The main result of this paper is the translation of PLDL to alternating Büchi automata. By an extension of the alternating color technique of [11], and by very similar algorithms, we obtain the following results: PLDL model checking is **PSPACE**-complete and realizability is **2EXPTIME**-complete. Thus, both problems are no harder than their corresponding variants for LTL. Finally, we give tight exponential and doubly-exponential bounds on satisfying valuations for model checking and realizability.

Our translation might also be of use for LDL on infinite traces, since De Giacomo and Vardi [4] only considered LDL on finite traces. Unlike the translation from logic into automata presented there, which is a top-down construction of an alternating automaton, we present a bottom-up approach.

2 PLDL

Let $\mathcal{V}$ be an infinite set of variables and let us fix a finite[1] set P of atomic propositions which we use to build our formulas and to label transition systems in which we evaluate them. For a subset $A \in 2^P$ and a propositional formula ϕ over P, we write $A \models \phi$, if the variable valuation mapping elements in A to true and elements not in A to false satisfies ϕ. The formulas of PLDL are given by the grammar

$$\varphi ::= p \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle r \rangle \varphi \mid [r]\varphi \mid \langle r \rangle_{\leq z} \varphi \mid [r]_{\leq z} \varphi$$
$$r ::= \phi \mid \varphi? \mid r + r \mid r\,;r \mid r^*$$

where $p \in P$, $z \in \mathcal{V}$, and where ϕ stands for arbitrary propositional formulas over P. We use the abbreviations $\mathtt{tt} = p \vee \neg p$ and $\mathtt{ff} = p \wedge \neg p$ for some atomic proposition p. The regular expressions have two types of atoms: propositional formulas ϕ over the atomic propositions and tests $\varphi?$, where φ is again a PLDL formula. Note that the semantics of the propositional atom ϕ differ from the semantics of the test $\phi?$: the former consumes an input letter, while tests do not make progress on the word. This is why both types of atoms are allowed.

The set of subformulas of φ is denoted by $\mathrm{cl}(\varphi)$. Note that regular expressions are not subformulas, but the formulas appearing in the tests are, e.g., we have $\mathrm{cl}(\langle p?\,;q\rangle_{\leq x} r) = \{p, r, \langle p?\,;q\rangle_{\leq x} r\}$. The size $|\varphi|$ of φ is the sum of $|\mathrm{cl}(\varphi)|$ and the sum of the lengths of the regular expressions appearing in φ (counted with multiplicity). We define $\mathrm{var}_\Diamond(\varphi) = \{z \in \mathcal{V} \mid \langle r \rangle_{\leq z} \psi \in \mathrm{cl}(\varphi)\}$ to be the set of variables parameterizing diamond operators in φ, $\mathrm{var}_\Box(\varphi) = \{z \in \mathcal{V} \mid [r]_{\leq z} \psi \in \mathrm{cl}(\varphi)\}$ to be the set of variables parameterizing box operators in φ, and set $\mathrm{var}(\varphi) = \mathrm{var}_\Diamond(\varphi) \cup \mathrm{var}_\Box(\varphi)$. Usually, we will denote variables in $\mathrm{var}_\Diamond(\varphi)$ by x and variables in $\mathrm{var}_\Box(\varphi)$ by y, if φ is clear from the context. A formula φ is variable-free, if $\mathrm{var}(\varphi) = \emptyset$.

The semantics of PLDL are defined inductively with respect to an ω-word $w = w_0 w_1 w_2 \cdots \in (2^P)^\omega$, a position $n \in \mathbb{N}$, and a variable valuation $\alpha \colon \mathcal{V} \to \mathbb{N}$ via

- $(w, n, \alpha) \models p$ if $p \in w_n$ and dually for $\neg p$,
- $(w, n, \alpha) \models \psi_0 \wedge \psi_1$ if $(w, n, \alpha) \models \psi_0$ and $(w, n, \alpha) \models \psi_1$,
- $(w, n, \alpha) \models \psi_0 \vee \psi_1$ if $(w, n, \alpha) \models \psi_0$ or $(w, n, \alpha) \models \psi_1$,
- $(w, n, \alpha) \models \langle r \rangle \psi$ if there exists $j \in \mathbb{N}$ s.t. $(n, n+j) \in \mathscr{R}(r, w, \alpha)$ and $(w, n+j, \alpha) \models \psi$,
- $(w, n, \alpha) \models [r]\psi$ if for all $j \in \mathbb{N}$ with $(n, n+j) \in \mathscr{R}(r, w, \alpha)$ we have $(w, n+j, \alpha) \models \psi$,
- $(w, n, \alpha) \models \langle r \rangle_{\leq z} \psi$ if there exists $0 \leq j \leq \alpha(z)$ s.t. $(n, n+j) \in \mathscr{R}(r, w, \alpha)$ and $(w, n+j, \alpha) \models \psi$,
- $(w, n, \alpha) \models [r]_{\leq z} \psi$ if for all $0 \leq j \leq \alpha(z)$ with $(n, n+j) \in \mathscr{R}(r, w, \alpha)$ we have $(w, n+j, \alpha) \models \psi$.

Here, the relation $\mathscr{R}(r, w, \alpha) \subseteq \mathbb{N} \times \mathbb{N}$ contains all pairs (m, n) such that $w_m \cdots w_{n-1}$ matches r (α is needed to evaluate tests in r, which might have parameterized subformulas) and is defined inductively by

- $\mathscr{R}(\phi, w, \alpha) = \{(n, n+1) \mid w_n \models \phi\}$ for propositional ϕ,
- $\mathscr{R}(\psi?, w, \alpha) = \{(n, n) \mid (w, n, \alpha) \models \psi\}$,
- $\mathscr{R}(r_0 + r_1, w, \alpha) = \mathscr{R}(r_0, w, \alpha) \cup \mathscr{R}(r_1, w, \alpha)$,
- $\mathscr{R}(r_0\,;r_1, w, \alpha) = \{(n_0, n_2) \mid \exists n_1 \text{ s.t. } (n_0, n_1) \in \mathscr{R}(r_0, w, \alpha) \text{ and } (n_1, n_2) \in \mathscr{R}(r_1, w, \alpha)\}$, and
- $\mathscr{R}(r^*, w, \alpha) = \{(n, n) \mid n \in \mathbb{N}\} \cup \{(n_0, n_{k+1}) \mid \exists n_1, \ldots, n_k \text{ s.t. } (n_j, n_{j+1}) \in \mathscr{R}(r, w, \alpha) \text{ for all } j \leq k\}$.

We write $(w, \alpha) \models \varphi$ for $(w, 0, \alpha) \models \varphi$ and say that w is a model of φ with respect to α.

[1] This greatly simplifies our notation and exposition when we translate formulas into automata, but is not essential.

Example 1.

- *The formula* $\theta_{\infty p} := [tt^*]\langle tt^*\rangle p$ *expresses that* p *holds true infinitely often.*
- *In general, every* PLTL *formula [1] (and thus every* LTL *formula) can be translated into* PLDL*, e.g.,* $\mathbf{F}_{\leq x}\, p$ *is expressible as* $\langle tt^*\rangle_{\leq x} p$ *and* $p\,\mathbf{U}\,q$ *as* $\langle p^*\rangle q$ *or* $\langle p^* q\rangle tt$*.*
- *The formula* $[tt^*](q \rightarrow \langle (tt;tt)^* p\rangle)$ *requires that every request (a position where* q *holds) is followed by a response (a position where* p *holds) after an even number of steps.*

As usual for parameterized temporal logics, the use of variables has to be restricted: bounding diamond and box operators by the same variable leads to an undecidable satisfiability problem (cp. [1]).

Definition 1. *A* PLDL *formula* φ *is well-formed, if* $\mathrm{var}_\Diamond(\varphi) \cap \mathrm{var}_\Box(\varphi) = \emptyset$.

In the following, we only consider well-formed formulas and drop the qualifier "well-formed". We consider the following fragments of PLDL. Let φ be a PLDL formula: φ is an LDL formula [4], if φ is variable-free, φ is a $\mathrm{PLDL}_\Diamond$ formula, if $\mathrm{var}_\Box(\varphi) = \emptyset$, and φ is a $\mathrm{PLDL}_\Box$ formula, if $\mathrm{var}_\Diamond(\varphi) = \emptyset$. Every LDL, $\mathrm{PLDL}_\Diamond$, and every $\mathrm{PLDL}_\Box$ formula is well-formed by definition. As satisfaction of LDL formulas is independent of variable valuations, we write $(w,n) \models \varphi$ and $w \models \varphi$ instead of $(w,n,\alpha) \models \varphi$ and $(w,\alpha) \models \varphi$, respectively, if φ is an LDL formula.

LDL is as expressive as ω-regular languages, which can be proven by a straightforward translation of ETL_f [20], which expresses exactly the ω-regular languages, into LDL.

Theorem 1 ([19])**.** *For every* ω*-regular language* $L \subseteq (2^P)^\omega$ *there exists an effectively constructible* LDL *formula* φ *such that* $L = \{w \in (2^P)^\omega \mid w \models \varphi\}$.

Note that we define PLDL formulas to be in negation normal form. Nevertheless, a negation can be pushed to the atomic propositions using dualities allowing us to define the negation of a formula.

Lemma 1. *For every* PLDL *formula* φ *there exists an efficiently constructible* PLDL *formula* $\neg\varphi$ *s.t.*

1. $(w,n,\alpha) \models \varphi$ *if and only if* $(w,n,\alpha) \not\models \neg\varphi$*,*
2. $|\neg\varphi| = |\varphi|$*.*
3. *If* φ *is well-formed, then so is* $\neg\varphi$*. and vice versa.*

Proof. We construct $\neg\varphi$ by structural induction over φ using the dualities of the operators:

- $\neg(p) = \neg p$
- $\neg(\neg p) = p$
- $\neg(\varphi \wedge \psi) = (\neg\varphi) \vee (\neg\psi)$
- $\neg(\varphi \vee \psi) = (\neg\varphi) \wedge (\neg\psi)$
- $\neg(\langle r\rangle\varphi) = [r]\neg\varphi$
- $\neg([r]\varphi) = \langle r\rangle\neg\varphi$
- $\neg(\langle r\rangle_{\leq x}\varphi) = [r]_{\leq x}\neg\varphi$
- $\neg([r]_{\leq y}\varphi) = \langle r\rangle_{\leq y}\neg\varphi$

The latter two claims of Lemma 1 follow from the definition of $\neg\varphi$ while the first one can be shown by a straightforward structural induction over φ. □

A simple, but very useful property of PLDL is the monotonicity of the parameterized operators: increasing (decreasing) the values of parameters bounding diamond (box) operators preserves satisfaction.

Lemma 2. *Let* φ *be a* PLDL *formula and let* α *and* β *be variable valuations satisfying* $\beta(x) \geq \alpha(x)$ *for every* $x \in \mathrm{var}_\Diamond(\varphi)$ *and* $\beta(y) \leq \alpha(y)$ *for every* $y \in \mathrm{var}_\Box(\varphi)$*. If* $(w,\alpha) \models \varphi$*, then* $(w,\beta) \models \varphi$*.*

The previous lemma allows us to eliminate parameterized box operators when asking for the existence of a variable valuation satisfying a formula.

Lemma 3. *For every* PLDL *formula* φ *there is an efficiently constructible* $\text{PLDL}_\Diamond$ *formula* φ' *of the same size as* φ *such that*

1. *for every* α *there is an* α' *such that for all* w*: if* $(w,\alpha) \models \varphi$ *then* $(w,\alpha') \models \varphi'$*, and*
2. *for every* α' *there is an* α *such that for all* w*: if* $(w,\alpha') \models \varphi'$ *then* $(w,\alpha) \models \varphi$*.*

Proof. We construct a single test $\hat{r}$ such that $\mathscr{R}(r,w,\alpha) \cap \{(n,n) \mid n \in \mathbb{N}\} = \mathscr{R}(\hat{r},w,\alpha)$ for every w and every α, which suffices to prove the equivalence of $[r]_{\leq y}\psi$ and $[\hat{r}]\psi$ provided we have $\alpha(y) = 0$, which is sufficient due to monotonicity. We apply the following rewriting rules (in the given order) to r:

1. Replace every subexpression of the form r'^* by $\mathtt{tt}?$, until no longer applicable.
2. Replace every subexpression of the form $\phi\,;r'$ or $r'\,;\phi$ by $\mathtt{ff}?$ and replace every subexpression of the form $\phi + r'$ or $r' + \phi$ by r', where ϕ is a propositional formula, until no longer applicable.
3. Replace every subexpression of the form $\psi_0? + \psi_1?$ by $(\psi_0 \vee \psi_1)?$ and replace every subexpression of the form $\psi_0?\,;\psi_1?$ by $(\psi_0 \wedge \psi_1)?$, until no longer applicable.

After step 2, r contains no iterations and no propositional atoms unless the expression itself is one. In the former case, applying the last two rules yields a regular expression which is a single test, which we denote by $\hat{r}$. In the latter case, we define $\hat{r} = \mathtt{ff}?$.

Each rewriting step preserves the intersection $\mathscr{R}(r,w,\alpha) \cap \{(n,n) \mid n \in \mathbb{N}\}$. As $\hat{r}$ is a test, we conclude $\mathscr{R}(r,w,\alpha) \cap \{(n,n) \mid n \in \mathbb{N}\} = \mathscr{R}(\hat{r},w,\alpha)$ for every w and every α. Note that $\hat{r}$ can be efficiently computed from r and is of the same size as r. Now, replace every subformula $[r]_{\leq y}\psi$ of φ by $[\hat{r}]\psi$ and denote the formula obtained by φ', which is a $\text{PLDL}_\Diamond$ formula that is efficiently constructible and of the same size.

Given an α, we define α_0 by $\alpha_0(z) = \alpha(z)$, if $z \in \text{var}_\Diamond(\varphi)$ and $\alpha_0(z) = 0$ otherwise. If $(w,\alpha) \models \varphi$, then $(w,\alpha_0) \models \varphi$ due to monotonicity. By construction of φ', we also have $(w,\alpha_0) \models \varphi'$. On the other hand, if $(w,\alpha') \models \varphi'$, then $(w,\alpha'_0) \models \varphi'$ as well, where α'_0 is defined as above. By construction of φ', we conclude $(w,\alpha_0) \models \varphi$. □

2.1 The Alternating Color Technique and LDL_{cp}

In this subsection, we repeat the alternating color technique, which was introduced by Kupferman et al. to solve the model checking and the realizability problem for PROMPT–LTL, amongst others. Let $p \notin P$ be a fresh proposition and define $P' = 2^{P \cup \{p\}}$. We think of words in $(2^{P'})^\omega$ as colorings of words in $(2^P)^\omega$, i.e., $w' \in (2^{P'})^\omega$ is a coloring of $w \in (2^P)^\omega$, if we have ${w_n}' \cap P = w_n$ for every position n. Furthermore, n is a changepoint, if $n = 0$ or if the truth value of p differs at positions $n-1$ and n. A block is a maximal infix that has exactly one changepoint, which is at the first position of the infix. By maximality, this implies that the first position after a block is a changepoint. Let $k \geq 1$. We say that w' is k-bounded, if every block has length at most k, which implies that w' has infinitely many changepoints. Dually, w' is k-spaced, if it has infinitely many changepoints and every block has length at least k.

The alternating color technique replaces a parameterized diamond operator $\langle r \rangle_{\leq x}\psi$ by an unparameterized one that requires the formula ψ to be satisfied within at most one color change. To this end, we introduce a changepoint-bounded variant $\langle\cdot\rangle_{cp}$ of the diamond operator. Since we need the dual operator $[\cdot]_{cp}$ to allow for negation via dualization, we introduce it here as well. We define

- $(w,n,\alpha) \models \langle r \rangle_{cp}\psi'$ if there exists a $j \in \mathbb{N}$ s.t. $(n,n+j) \in \mathscr{R}(r,w,\alpha)$, $w_n \cdots w_{n+j-1}$ contains at most one changepoint, and $(w,n+j,\alpha) \models \psi$, and
- $(w,n,\alpha) \models [r]_{cp}\psi'$ if for all $j \in \mathbb{N}$ with $(n,n+j) \in \mathscr{R}(r,w,\alpha)$ and where $w_n \cdots w_{n+j-1}$ contains at most one changepoint we have $(w,n+j,\alpha) \models \psi$.

We denote the logic obtained by disallowing parameterized operators, but allowing changepoint-bounded operators, by LDL_{cp}. Note that the semantics of LDL_{cp} formulas are independent of variable valuations. Hence, we drop them from our notation for the satisfaction relations $\models$ and $\mathscr{R}$. Also, Lemma 1 can be extended to LDL_{cp} by adding the rules $\neg(\langle r\rangle_{cp}\psi) = [r]_{cp}\neg\psi$ and $\neg([r]_{cp}\psi) = \langle r\rangle_{cp}\neg\psi$ to the proof.

Now, we are ready to introduce the alternating color technique. Given a $\mathrm{PLDL}_\Diamond$ formula φ, let $\mathrm{rel}(\varphi)$ be the formula obtained by inductively replacing every subformula $\langle r\rangle_{\leq x}\psi$ by $\langle \mathrm{rel}(r)\rangle_{cp}\mathrm{rel}(\psi)$, i.e., we replace the parameterized diamond operator by a changepoint-bounded one. Note that this replacement is also performed in the regular expressions, i.e., $\mathrm{rel}(r)$ is the regular expression obtained by applying the replacement to every test $\psi'?$ in r.

Given a $\mathrm{PLDL}_\Diamond$ formula φ let $c(\varphi) = \mathrm{rel}(\varphi) \wedge \theta_{\infty p} \wedge \theta_{\infty\neg p}$ (cf. Example 1), which is an LDL_{cp} formula and only linearly larger than φ. On k-bounded and k-spaced colorings of w there is an equivalence between φ and $c(\varphi)$. The proof is similar to the original one [11].

Lemma 4 (cp. Lemma 2.1 of [11])**.** *Let φ be a* $\mathrm{PLDL}_\Diamond$ *formula and let $w \in (2^P)^\omega$.*

1. *If $(w,\alpha) \models \varphi$, then $w' \models c(\varphi)$ for every k-spaced coloring w' of w, where $k = \max_{x\in\mathrm{var}(\varphi)} \alpha(x)$.*
2. *Let $k \in \mathbb{N}$. If w' is a k-bounded coloring of w with $w' \models c(\varphi)$, then $(w,\alpha) \models \varphi$, where $\alpha(x) = 2k$ for every x.*

3 From $\mathrm{LDL}_{\mathrm{cp}}$ to Alternating Büchi Automata

In this section, we show how to translate LDL_{cp} formulas into alternating Büchi word automata of linear size using an inductive bottom-up approach. These automata allow us to use automata-based constructions to solve the model checking and the realizability problem for PLDL via the alternating color technique which links PLDL and LDL_{cp}.

An alternating Büchi automaton $\mathfrak{A} = (Q,\Sigma,q_0,\delta,F)$ consists of a finite set Q of states, an alphabet Σ, an initial state $q_0 \in Q$, a transition function $\delta\colon Q\times\Sigma \to \mathscr{B}^+(Q)$, and a set $F \subseteq Q$ of accepting states. Here, $\mathscr{B}^+(Q)$ denotes the set of positive boolean combinations over Q, which contains in particular the formulas `tt` (true) and `ff` (false). A run of $\mathfrak{A}$ on $w = w_0w_1w_2\cdots \in \Sigma^\omega$ is a directed graph $\rho = (V,E)$ with $V \subseteq Q\times\mathbb{N}$ and $((q,n),(q',n')) \in E$ implies $n' = n+1$ such that the following two conditions are satisfied: $(q_0,0) \in V$ and for all $(q,n) \in V$: $\mathrm{Succ}_\rho(q,n) \models \delta(q,w_n)$. Here $\mathrm{Succ}_\rho(q,n)$ denotes the set of successors of (q,n) in ρ projected to Q. A run ρ is accepting if all infinite paths (projected to Q) through ρ visit F infinitely often. The language $L(\mathfrak{A})$ contains all $w \in \Sigma^\omega$ that have an accepting run of $\mathfrak{A}$.

Theorem 2. *For every* LDL_{cp} *formula φ, there is an alternating Büchi automaton $\mathfrak{A}_\varphi$ with linearly many states (in $|\varphi|$) such that $L(\mathfrak{A}_\varphi) = \{w \in (2^{P'})^\omega \mid w \models \varphi\}$.*

To prove the theorem, we inductively construct automata $\mathfrak{A}_\psi$ for every subformula $\psi \in \mathrm{cl}(\varphi)$ satisfying $L(\mathfrak{A}_\psi) = \{w \in (2^{P'})^\omega \mid w \models \psi\}$. The automata for atomic formulas are straightforward and depicted in Figure 1(a) and (b). To improve readability, we allow propositional formulas over P' as transition labels: the formula ϕ stands for all sets $A \in 2^{P'}$ with $A \models \phi$. Furthermore, given automata $\mathfrak{A}_{\psi_0}$ and $\mathfrak{A}_{\psi_1}$, using a standard construction, we can build the automaton $\mathfrak{A}_{\psi_0\vee\psi_1}$ by taking the disjoint union of the two automata, adding a new initial state q_0 with $\delta(q_0,A) = \delta^0(q_0^0,A) \vee \delta^1(q_0^1,A)$. Here, q_0^i is the initial state and δ^i is the transition function of $\mathfrak{A}_{\psi_i}$. The automaton $\mathfrak{A}_{\psi_0\wedge\psi_1}$ is defined similarly, the only difference being $\delta(q_0,A) = \delta^0(q_0^0,A) \wedge \delta^1(q_0^1,A)$.

It remains to consider temporal formulas, e.g., $\langle r\rangle\psi$. First, we turn the regular expression r into an automaton $\mathfrak{A}_r$. Recall that tests do not process input letters. Hence, we disregard the tests when

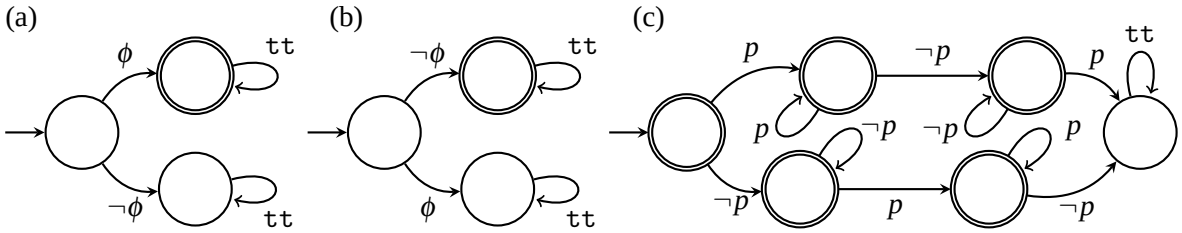

Figure 1: The automata $\mathfrak{A}_p$ (a), $\mathfrak{A}_{\neg p}$ (b), and $\mathfrak{A}_{cp}$ (c), which tracks color changepoints.

defining the transition function, but we label states at which the test has to be executed by this test. We use the Thompson construction [18] to turn r into $\mathfrak{A}_r$, i.e., we obtain an ε-NFA. Then, we show how to combine $\mathfrak{A}_r$ with the automaton $\mathfrak{A}_\psi$ and the automata $\mathfrak{A}_{\psi_1},\dots,\mathfrak{A}_{\psi_k}$, where $\psi_1?,\dots,\psi_k?$ are the test occurring in r. The ε-transitions introduced by the Thompson construction are then removed, since alternating automata do not allow them. During this process, we also ensure that the transition relation takes tests into account by introducing universal transitions that lead from a state marked with $\psi_j?$ into the corresponding automaton $\mathfrak{A}_{\psi_j}$.

Formally, an ε-NFA with markings $\mathfrak{A} = (Q,\Sigma,q_0,\delta,C,m)$ consists of a finite set Q of states, an alphabet Σ, an initial state $q_0 \in Q$, a transition function $\delta\colon Q\times\Sigma\cup\{\varepsilon\}\to 2^Q$, a set C of final states (C, since we use them to concatenate automata), and a partial marking function m, which assigns to some states $q\in Q$ an LDL_{cp} formula $m(q)$. We write $q\xrightarrow{a}q'$, if $q'\in\delta(q,a)$ for $a\in\Sigma\cup\{\varepsilon\}$. An ε-path π from q to q' in $\mathfrak{A}_r$ is a sequence $\pi = q_1\cdots q_k$ of $k\geq 1$ states with $q = q_1\xrightarrow{\varepsilon}\cdots\xrightarrow{\varepsilon}q_k = q'$. The set of all ε-paths from q to q' is denoted by $\Pi(q,q')$. Let $m(\pi) = \{m(q_i)\mid 1\leq i\leq k\}$ be the set of markings visited by π.

A run of $\mathfrak{A}$ on $w_0\cdots w_{n-1}\in\Sigma^*$ is a sequence $q_0q_1\cdots q_n$ such that for every i in the range $0\leq i\leq n-1$ there is a state q_i' reachable from q_i via an ε-path π_i and with $q_{i+1}\in\delta(q_i',w_i)$. The run is accepting if there is a $q_n'\in C$ reachable via an ε-path π_n from q_n. This slightly unusual definition (but equivalent to the standard one) simplifies our reasoning below. Also, the definition is oblivious to the marking.

We begin by defining the automaton $\mathfrak{A}_r$ by induction over the structure of r as depicted in Figure 2. Note that the automata we construct have no outgoing edges leaving the unique final state and that we mark some states with tests $\psi_j?$ (denoted by labeling states with the test).

Lemma 5. *Let $w = w_0w_1w_2\cdots\in(2^{P'})^\omega$ and let $w_0\cdots w_{n-1}$ be a (possibly empty, if $n=0$) prefix of w. The following two statements are equivalent:*

1. *$\mathfrak{A}_r$ has an accepting run $q_0q_1\cdots q_n$ on $w_0\cdots w_{n-1}$ with ε-paths π_i for i in the range $0\leq i\leq n$ such that $w_iw_{i+1}w_{i+2}\cdots\models\bigwedge m(\pi_i)$ for every i.*
2. *$(0,n)\in\mathscr{R}(r,w)$.*

Fix ψ and r (with tests $\psi_1?,\dots,\psi_k?$) and let $\mathfrak{A}_r = (Q^r,2^{P'},q_0^r,\delta^r,C^r,m)$, $\mathfrak{A}_\psi = (Q',2^{P'},q_0',\delta',F')$, and $\mathfrak{A}_{\psi_j} = (Q^j,2^{P'},q_0^j,\delta^j,F^j)$ for $j=1,\dots,k$ be the corresponding automata, which we assume to have pairwise disjoint sets of states. Next, we show how to construct $\mathfrak{A}_{\langle r\rangle\psi}$, $\mathfrak{A}_{[r]\psi}$, $\mathfrak{A}_{\langle r\rangle_{cp}\psi}$, and $\mathfrak{A}_{[r]_{cp}\psi}$.

We begin with $\langle r\rangle\psi$: we define $\mathfrak{A}_{\langle r\rangle\psi} = (Q^r\cup Q'\cup Q_1\cup\cdots\cup Q_k,2^{P'},q_0^r,\delta,F_1\cup\cdots\cup F_k)$ with

$$\delta(q,A) = \begin{cases} \delta'(q,A) & \text{if } q\in Q',\\ \delta^j(q,A) & \text{if } q\in Q^j,\\ \bigvee_{q'\in Q^r\setminus C^r}\bigvee_{\pi\in\Pi(q,q')}\bigvee_{p\in\delta^r(q',A)}\left(p\wedge\bigwedge_{\psi_j\in m(\pi)}\delta^j(q_0^j,A)\right) & \\ \qquad\qquad\vee & \text{if } q\in Q^r.\\ \bigvee_{q'\in C^r}\bigvee_{\pi\in\Pi(q,q')}\left(\delta'(q_0',A)\wedge\bigwedge_{\psi_j\in m(\pi)}\delta^j(q_0^j,A)\right) & \end{cases}$$

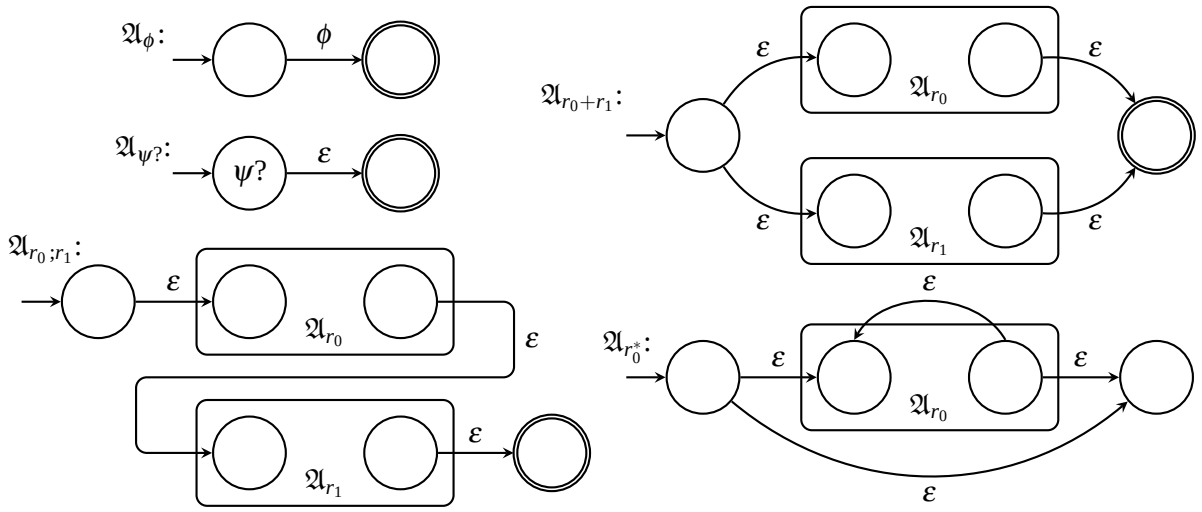

Figure 2: The inductive definition of $\mathfrak{A}_r$ via the Thompson construction.

So, $\mathfrak{A}_{\langle r\rangle\psi}$ is the union of the automata for the regular expression, the tests, and for ψ with a modified transition function. The transitions of the automata $\mathfrak{A}_\psi$ and $\mathfrak{A}_{\psi_j}$ are left unchanged and the transition function for states in Q^r is obtained by removing ε-transitions. First consider the upper disjunct: it ranges disjunctively over all non-final states p that are reachable via an initial ε-path and an A-transition in the end. To account for the tests visited during the ε-path (but not the test at p), we add conjunctively transitions that lead into the corresponding automata. The lower disjunct is similar, but ranges over paths that end in a final state. Since we concatenate the automaton $\mathfrak{A}_r$ with the automaton $\mathfrak{A}_\psi$, all edges leading into final states of $\mathfrak{A}_r$ are rerouted to the initial state of $\mathfrak{A}_\psi$. The tests along the ε-path are accounted for as in the first case. Finally, note that Q^r does not contain any (Büchi) accepting states, i.e., every accepting run on w has to leave Q^r after a finite number of transitions. Since this is only possible via transitions that would lead $\mathfrak{A}_r$ into a final state, this ensures the existence of a position n such that $(0,n) \in \mathscr{R}(r,w)$.

The definition of $\mathfrak{A}_{[r]\psi}$ is dual, i.e., we have to use automata $\mathfrak{A}_{\neg\psi_j} = (Q^j, 2^{P'}, q_0^j, \delta^j, F^j)$ for $j = 1,\dots,k$ for the negated tests and ε-transitions are removed in a universal manner. Formally, we define $\mathfrak{A}_{[r]\psi} = (Q^r \cup Q' \cup Q_1 \cup \dots \cup Q_k, 2^{P'}, q_0^r, \delta, Q^r \cup F_1 \cup \dots \cup F_k)$ where

$$\delta(q,A) = \begin{cases} \delta'(q,A) & \text{if } q \in Q', \\ \delta^j(q,A) & \text{if } q \in Q^j, \\ \bigwedge_{q' \in Q^r \setminus C^r} \bigwedge_{\pi \in \Pi(q,q')} \bigwedge_{p \in \delta^r(q',A)} (p \vee \bigvee_{\psi_j \in m(\pi)} \delta^j(q_0^j,A)) & \\ \qquad \wedge & \text{if } q \in Q^r. \\ \bigwedge_{q' \in C^r} \bigwedge_{\pi \in \Pi(q,q')} (\delta'(q_0',A) \vee \bigvee_{\psi_j \in m(\pi)} \delta^j(q_0^j,A)) & \end{cases}$$

Note that we add Q^r to the (Büchi) accepting states, since a run on w might stay in Q^r forever, as it has to consider all positions n with $(0,n) \in \mathscr{R}(r,w)$.

For the changepoint-bounded operators, we have to modify $\mathfrak{A}_r$ to make it count color changes. Let $\mathfrak{A}_{cp} = (Q^{cp}, 2^{P'}, q_0^{cp}, \delta^{cp}, C^{cp})$ be the DFA depicted in Figure 1(c). We define the product of $\mathfrak{A}_r$ and $\mathfrak{A}_{cp}$ as $\hat{\mathfrak{A}}_r = (\hat{Q}^r, 2^{P'}, \hat{q}_0^r, \hat{\delta}^r, \hat{C}^r, \hat{m})$ where $\hat{Q}^r = Q^r \times Q^{cp}$, $\hat{q}_0^r = (q_0^r, q_0^{cp})$,

$$\delta((q,q'),A) = \begin{cases} \{(p, \delta^{cp}(q',A)) \mid p \in \delta^r(q,A)\} & \text{if } A \neq \varepsilon, \\ \{(p,q') \mid p \in \delta^r(q,A)\} & \text{if } A = \varepsilon, \end{cases}$$

$\hat{C}^r = C^r \times C^{cp}$, and $\hat{m}(q,q') = m(q)$. Using this, we define $\mathfrak{A}_{\langle r\rangle_{cp}\psi}$ as we defined $\mathfrak{A}_{\langle r\rangle\psi}$, but using $\hat{\mathfrak{A}}_r$ instead of $\mathfrak{A}_r$. Similarly, $\mathfrak{A}_{[r]_{cp}\psi}$ is defined as $\mathfrak{A}_{[r]\psi}$, but using $\hat{\mathfrak{A}}_r$ instead of $\mathfrak{A}_r$.

Proof of Theorem 2. First, we consider the size of $\mathfrak{A}_\varphi$. Boolean operations add one state while a temporal operator with regular expression r adds a number of states that is linear in the size of r (which is its length), even when we take the intersection with the automaton checking for color changes. Note that we do not need to complement the automata $\mathfrak{A}_{\psi_j}$ to obtain $\mathfrak{A}_{\neg\psi_j}$, instead we rely on Lemma 1. Hence, the size of $\mathfrak{A}_\varphi$ is linear in the size of φ. It remains to prove $L(\mathfrak{A}_\varphi) = \{w \in (2^{P'})^\omega \mid w \models \varphi\}$ by induction over the structure of φ. The induction start for atomic formulas and the induction step for disjunction and conjunction are trivial, hence it remains to consider the temporal operators.

Consider $\langle r\rangle\psi$. If $w \models \langle r\rangle\psi$, then there exists a position n such that $w_n w_{n+1} w_{n+2} \cdots \models \psi$ and $(0,n) \in \mathscr{R}(r,w)$. Hence, there is a run of $\mathfrak{A}_r$ on $w_0 \cdots w_{n-1}$ such that the tests visited during the run are satisfied by the appropriate suffixes of w. Thus, applying the induction hypothesis yields accepting runs of the test automata on these suffixes. Furthermore, there is an accepting run of $\mathfrak{A}_\psi$ on $w_n w_{n+1} w_{n+2} \cdots$, again by induction hypothesis. These runs can be "glued" together to build an accepting run of $\mathfrak{A}_{\langle r\rangle\psi}$ on w.

For the other direction, consider an accepting run ρ of $\mathfrak{A}_{\langle r\rangle\psi}$ on w. Let $n \geq 0$ be the last level of ρ that contains a state from Q^r. Such a level has to exist since states in Q^r are not accepting and they have no incoming edges from states of the automata $\mathfrak{A}_\psi$ and $\mathfrak{A}_{\psi_j}$, but the initial state of $\mathfrak{A}_{\langle r\rangle\psi}$ is in Q^r. Furthermore, $\mathfrak{A}_{\langle r\rangle\psi}$ is non-deterministic and complete when restricted to states in $Q^r \setminus C^r$. Hence, we can extract an accepting run of $\mathfrak{A}_r$ from ρ on $w_0 \cdots w_{n-1}$ that satisfies additionally the requirements formulated in Statement 1 of Lemma 5, due to the transitions into the test automata and an application of the induction hypothesis. Hence, we have $(0,n) \in \mathscr{R}(r,w)$. Furthermore, from the remainder of ρ (levels greater or equal to n) we can extract an accepting run of $\mathfrak{A}_\psi$ on $w_n w_{n+1} w_{n+2} \cdots$. Hence, $w_n w_{n+1} w_{n+2} \cdots \models \psi$ by induction hypothesis. Altogether, we conclude $w \models \langle r\rangle\psi$.

The case for $[r]\psi$ is dual, while the cases for the changepoint-bounded operators $\langle r\rangle_{cp}\psi$ and $[r]_{cp}\psi$ are analogous, using the fact that $\mathfrak{A}_{cp}$ only accepts words which have at most one changepoint. □

Note that the size of $\mathfrak{A}_\varphi$ is linear in $|\varphi|$, but it is not clear that it can be computed in polynomial time in $|\varphi|$, since the transition functions of subautomata of the form $\mathfrak{A}_{\langle r\rangle\psi}$ contain disjunctions that range over the set of ε-paths. Here, it suffices to consider paths that do not contain a state twice, but even this restriction still allows for an exponential number of different paths. Fortunately, we do not need to compute $\mathfrak{A}_\varphi$ in polynomial time. It suffices to do it in polynomial space, which is sufficient for the applications in the next sections, which is clearly possible.

Furthermore, using standard constructions (e.g., [13, 15]), we can turn the alternating Büchi automaton $\mathfrak{A}_\varphi$ into a non-deterministic Büchi automaton of exponential size and a deterministic parity automaton[2] of doubly-exponential size with linearly many colors.

4 Model Checking

In this section, we consider the PLDL model checking problem. A (P-labeled) transition system $\mathscr{S} = (S, s_0, E, \ell)$ consists of a finite set S of states, an initial state s_0, a (left-)total edge relation $E \subseteq S \times S$, and a labeling $\ell\colon S \to 2^P$. An initial path through $\mathscr{S}$ is a sequence $\pi = s_0 s_1 s_2 \cdots$ of states satisfying $(s_n, s_{n+1}) \in E$ for every n. Its trace is defined as $\mathrm{tr}(\pi) = \ell(s_0)\ell(s_1)\ell(s_2)\cdots$. We say that $\mathscr{S}$ satisfies a

[2] The states of a parity automaton are colored by $\Omega\colon Q \to \mathbb{N}$. It accepts a word w, if it has a run $q_0 q_1 q_2 \cdots$ on w such that $\max\{\Omega(q) \mid q_i = q \text{ for infinitely many i}\}$ is even.

PLDL formula φ with respect to a variable valuation α, if we have $(\mathrm{tr}(\pi),\alpha) \models \varphi$ for every initial path π of $\mathcal{S}$. The model checking problem asks, given a transition system $\mathcal{S}$ and a formula φ, to determine whether $\mathcal{S}$ satisfies φ with respect to some variable valuation α.

Theorem 3. *The* PLDL *model checking problem is* **PSPACE**-*complete.*

To solve the PLDL model checking problem, we first notice that we can restrict ourselves to $\mathrm{PLDL}_\Diamond$ formulas. Let φ and φ' be due defined as in Lemma 3. Then, $\mathcal{S}$ satisfies φ with respect to some α if and only if $\mathcal{S}$ satisfies φ' with respect to some α'.

Our algorithm is similar to the one presented for PROMPT–LTL in [11] and uses the alternating color technique. Recall that $p \notin P$ is the fresh atomic proposition used to specify the coloring and induces the blocks, maximal infixes with its unique changepoint at the first position. Let $G = (V,E,v_0,\ell,F)$ denote a colored Büchi graph consisting of a finite directed graph (V,E), an initial vertex v_0, a labeling function $\ell\colon V \to 2^{\{p\}}$ labeling vertices by p or not, and a set $F \subseteq V$ of accepting states. A path $v_0v_1v_2\cdots$ through G is pumpable, if all its blocks have at least one state that appears twice in this block. Furthermore, the path is fair, if it visits F infinitely often. The pumpable non-emptiness problem asks, given a colored Büchi graph G, whether it has a pumpable fair path starting in the initial state.

Theorem 4 ([11])**.** *The pumpable non-emptiness problem for colored Büchi graphs is* NLOGSPACE-*complete and can be solved in linear time.*

The following lemma reduces the $\mathrm{PLDL}_\Diamond$ model checking problem to the pumpable non-emptiness problem for colored Büchi graphs of exponential size. Given a non-deterministic Büchi automaton $\mathfrak{A} = (Q,2^{P\cup\{p\}},q_0,\Delta,F)$ recognizing the models of $\neg\mathrm{rel}(\varphi) \wedge \theta_{\infty p} \wedge \theta_{\infty\neg p}$ (note that $\mathrm{rel}(\varphi)$ is negated) and a transition system $\mathcal{S} = (S,s_0,E,\ell)$, we define the product $\mathfrak{A}\times\mathcal{S}$ to be the colored Büchi graph

$$\mathfrak{A}\times\mathcal{S} = (Q\times S\times 2^{\{p\}},E',(q_0,s_0,\emptyset),\ell',F\times S\times 2^{\{p\}})$$

where $((q,s,C),(q',s',C')) \in E'$ if and only if $(s,s') \in E$ and $q' \in \delta(q,\ell(s)\cup C)$, and where $\ell'(q,s,C) = C$.

Each initial path $(q_0,s_0,C_0)(q_1,s_1,C_1)(q_2,s_2,C_2)\cdots$ through the product $\mathfrak{A}\times\mathcal{S}$ induces a coloring $(L(s_0)\cup C_0)(L(s_1)\cup C_1)(L(s_2)\cup C_2)\cdots$ of the trace of the path $s_0s_1s_2\cdots$ through $\mathcal{S}$. Furthermore, $q_0q_1q_2\cdots$ is a run of $\mathfrak{A}$ on the coloring.

Lemma 6 (cp. Lemma 4.2 of [11])**.** *$\mathcal{S}$ does not satisfy φ with respect to any α if and only if $\mathfrak{A}\times\mathcal{S}$ has a pumpable fair path.*

Proof. Let φ not be satisfied by $\mathcal{S}$ with respect to any α, i.e., for every α there exists an initial path π through $\mathcal{S}$ such that $(\mathrm{tr}(\pi),\alpha) \not\models \varphi$. Pick α^* such that $\alpha^*(x) = 2\cdot|Q|\cdot|S|+1$ and let π^* be the corresponding path. Applying Lemma 4.2 yields $w \not\models c(\varphi)$ for every $|Q|\cdot|S|$-bounded coloring of $\mathrm{tr}(\pi^*)$. Now, consider the unique $|Q|\cdot|S|$-bounded and $|Q|\cdot|S|$-spaced coloring w of $\mathrm{tr}(\pi^*)$ that starts with p not holding true in the first position. As argued above, $w \not\models c(\varphi)$, and we have $w \models \theta_{\infty p}\wedge\theta_{\infty\neg p}$, as w is bounded. Hence, $w \models \neg\mathrm{rel}(\varphi)\wedge\theta_{\infty p}\wedge\theta_{\infty\neg p}$, i.e., there is an accepting run $q_0q_1q_2\cdots$ of $\mathfrak{A}$ in w. This suffices to show that $(q_0,\pi_0,w_0\cap\{p\})(q_1,\pi_1,w_1\cap\{p\})(q_1,\pi_1,w_2\cap\{p\})\cdots$ is a pumpable fair path through $\mathfrak{A}\times\mathcal{S}$, since every block has length greater than $|Q|\cdot|S|$. This implies the existence of a repeated state in every block, since there are exactly $|Q|\cdot|S|$ vertices of each color.

Now, let $\mathfrak{A}\times\mathcal{S}$ contain a pumpable fair path $(q_0,s_0,C_0)(q_1,s_1,C_1)(q_2,s_2,C_2)\cdots$, fix some arbitrary α, and define $k = \max_{x\in\mathrm{var}_\Diamond\varphi}\alpha(x)$. There is a repetition of a vertex of $\mathfrak{A}\times\mathcal{S}$ in every block, each of which can be pumped k times. This path is still fair and induces a coloring w'_k of a trace w_k of an initial path of $\mathcal{S}$. Since the run encoded in the first components is an accepting one on w'_k, we conclude that the coloring w'_k satisfies $\neg c(\varphi)$. Furthermore, w'_k is k-spaced, since we pumped each repetition k times.

Towards a contradiction assume we have $(w,\alpha) \models \varphi$. Applying Lemma 4.1 yields $w' \models c(\varphi)$, which contradicts $\neg c(\varphi)$. Hence, for every α we have constructed a path of $\mathscr{S}$ whose trace does not satisfy φ with respect to α, i.e., $\mathscr{S}$ does not satisfy φ with respect to any α. □

We can deduce an upper bound on valuations that satisfy a formula in a given transition system.

Corollary 1. *If there is a variable valuation such that $\mathscr{S}$ satisfies φ, then there is also one that is bounded exponentially in $|\varphi|$ and linearly in the number of states of $\mathscr{S}$.*

Proof. Let $\mathscr{S}$ satisfy φ with respect to α, but not with the valuation α^* with $\alpha^*(x) = 2 \cdot |Q| \cdot |S| + 1$. In the preceding proof, we constructed a pumpable fair path in $\mathfrak{A} \times \mathscr{S}$ starting from this assumption. This contradicts Lemma 6, since $\mathscr{S}$ satisfying φ with respect to α is equivalent to $\mathfrak{A} \times \mathscr{S}$ not having a pumpable fair path. Since $2 \cdot |Q| \cdot |S| + 1$ is exponential in $|\varphi|$ and linear in $|S|$, the result follows. □

A matching lower bound of 2^n can be proven by implementing a binary counter with n bits using a formula of polynomial size in n. This holds already true for PROMPT–LTL, as noted in [11].

It remains to prove the main result of this section: PLDL model checking is **PSPACE**-complete.

Proof of Theorem 3. **PSPACE**-hardness follows directly from the **PSPACE**-hardness of the LTL model checking problem [17], as LTL is a fragment of PLDL.

The following is a **PSPACE** algorithm: construct $\mathfrak{A} \times \mathscr{S}$ and check whether it contains a pumpable fair path, which is correct due to Lemma 6. Since the search for such a path can be implemented on-the-fly without having to construct the full product [11], it can be implemented using polynomial space. □

5 Realizability

In this section, we consider the realizability problem for PLDL. Throughout the section, we fix a partition (I,O) of the set of atomic propositions P. An instance of the PLDL realizability problem is given by a PLDL formula φ (over P) and the problem is to decide whether Player O has a winning strategy in the following game, played in rounds $n \in \mathbb{N}$: in each round n, Player I picks a subset $i_n \subseteq I$ and then Player O picks a subset $o_n \subseteq O$. Player O wins the play with respect to a variable valuation α, if $((i_0 \cup o_0)(i_1 \cup o_1)(i_2 \cup o_2)\cdots, \alpha) \models \varphi$.

Formally, a strategy for Player O is a mapping $\sigma\colon (2^I)^* \to 2^O$ and a play $\rho = i_0 o_0 i_1 o_1 i_2 o_2 \cdots$ is consistent with σ, if we have $o_n = \sigma(i_0 \cdots i_n)$ for every n. We call $(i_0 \cup o_0)(i_1 \cup o_1)(i_2 \cup o_2)\cdots$ the outcome of ρ, denoted by $\mathrm{outcome}(\rho)$. We say that a strategy σ for Player I is winning with respect to a variable valuation α, if we have $(\mathrm{outcome}(\rho), \alpha) \models \varphi$ for every play ρ that is consistent with σ. The PLDL realizability problem asks for a given PLDL formula φ, whether Player O has a winning strategy with respect to some variable valuation, i.e., there is a single α such that every outcome satisfies φ with respect to α. If this is the case, then we say that σ realizes φ and thus that φ is realizable.

We show the PLDL realizability problem to be **2EXPTIME**-complete: hardness follows easily from the **2EXPTIME**-completeness of the LTL realizability problem, which is a special case of the PLDL realizability problem. Membership in **2EXPTIME** on the other hand is shown by a reduction to the realizability problem for ω-regular specifications.

It is well-known that ω-regular specifications are realizable by finite-state transducers (if they are realizable at all) [3]. A transducer $\mathscr{T} = (Q, \Sigma, \Gamma, q_0, \delta, \tau)$ consists of a finite set Q of states, an input alphabet Σ, an output alphabet Γ, an initial state q_0, a transition function $\delta\colon Q \times \Sigma \to Q$, and a output function $\tau\colon Q \to \Gamma$. The function $f_{\mathscr{T}}\colon \Sigma^* \to \Gamma$ implemented by $\mathscr{T}$ is defined as $f_{\mathscr{T}}(w) = \tau(\delta^*(w))$,

where δ^* is defined as usual: $\delta^*(\varepsilon) = q_0$ and $\delta^*(wv) = \delta(\delta^*(w), v)$. To implement a strategy by a transducer, we use $\Sigma = 2^I$ and $\Gamma = 2^O$. Then, we say that the strategy $\sigma = f_{\mathscr{T}}$ is finite-state. The size of σ is the number of states of $\mathscr{T}$. The following proof is analogous to the one for PROMPT–LTL [11].

Theorem 5. *The* PLDL *realizability problem is* **2EXPTIME***-complete.*

When proving membership in **2EXPTIME**, we restrict ourselves without loss of generality to $\text{PLDL}_\Diamond$ formulas, as this special case is sufficient as shown in Lemma 3. First, we use the alternating color technique to show that the $\text{PLDL}_\Diamond$ realizability problem is reducible to the realizability problem for specifications in LDL_{cp}. When considering the LDL_{cp} realizability problem, we add the fresh proposition p used to specify the coloring to O, i.e., Player O is in charge of determining the color of each position.

Lemma 7 (cp. Lemma 3.1 of [11])**.** *A* $\text{PLDL}_\Diamond$ *formula* φ *over* I *and* O *is realizable if and only if the* LDL_{cp} *formula* $c(\varphi)$ *over* I *and* $O \cup \{p\}$ *is realizable.*

Proof. Let φ be realizable, i.e., there is a winning strategy $\sigma\colon (2^I)^+ \to 2^O$ for Player O with respect to some α. Now, consider the strategy $\sigma'\colon (2^I)^+ \to 2^{O\cup\{p\}}$ defined by

$$\sigma'(i_0 \cdots i_{n-1}) = \begin{cases} \sigma(i_0 \cdots i_{n-1}) & \text{if } n \bmod 2k < k, \\ \sigma(i_0 \cdots i_{n-1}) \cup \{p\} & \text{otherwise,} \end{cases}$$

where $k = \max_{x \in \text{var}_\Diamond(\varphi)} \alpha(x)$. We show that σ' realizes $c(\varphi)$. To this end, let $\rho' = i_0 o_0 i_1 o_1 i_2 o_2 \cdots$ be a play that is consistent with σ'. Then, $\rho = i_0(o_0 \setminus \{p\})i_1(o_1 \setminus \{p\})i_2(o_2 \setminus \{p\}) \cdots$ is by construction consistent with σ, i.e., $(\text{outcome}(\rho), \alpha) \models \varphi$. As ρ' is a k-spaced p-coloring of ρ, we deduce $\rho' \models c(\varphi)$ by applying Lemma 4.1. Hence, σ' realizes $c(\varphi)$.

Now, assume $c(\varphi)$ is realized by $\sigma'\colon (2^I)^+ \to 2^{O\cup\{p\}}$, which we can assume to be finite-state, say it is implemented by $\mathscr{T}$ with n states. We first show that every outcome that is consistent with σ' is $n+1$-bounded. Such an outcome satisfies $c(\varphi)$ and has therefore infinitely many changepoints. Now, assume it has a block of length strictly greater than $n+1$, say between changepoints at positions i and j. Let $q_0 q_1 q_2 \cdots$ be the states reached during the run of $\mathscr{T}$ on the projection of ρ to 2^I. Then, there are two positions i' and j' satisfying $i \leq i' < j' < j$ in the block such that $q_{i'} = q_{j'}$. Hence, $q_0 \cdots q_{i'-1}(q_{i'} \cdots q_{j'-1})^\omega$ is also a run of $\mathscr{T}$. However, the output generated by this run has only finitely many changepoints, since the output at the states $q_{i'}, \ldots, q_{j'-1}$ coincides when restricted to $\{p\}$. This contradicts the fact that $\mathscr{T}$ implements a winning strategy, which implies in particular that every output has infinitely many changepoints, as required by the conjunct $\theta_{\infty p} \wedge \theta_{\infty \neg p}$ of $c(\varphi)$. Hence, ρ is $(n+1)$-bounded.

Now, consider the strategy $\sigma\colon (2^I)^+ \to 2^O$ defined by $\sigma(i_0 \cdots i_{n-1}) = \sigma'(i_0 \cdots i_{n-1}) \cap O$. By definition, for every play ρ consistent with σ, there is a $(n+1)$-bounded p-coloring of ρ that is consistent with σ'. Hence, applying Lemma 4.2 yields $(\rho, \beta) \models \rho$, where $\beta(x) = 2n+2$. Hence, σ realizes φ with respect to β. Note that σ is also finite-state and of the same size as σ'. □

Proof of Theorem 5. As already mentioned above, **2EXPTIME**-hardness of the LDL realizability problem follows immediately from the **2EXPTIME**-hardness of the LTL realizability problem [14], as LTL is a fragment of PLDL.

Now, consider membership and recall that we have argued that it is sufficient to consider $\text{PLDL}_\Diamond$. Thus, let φ be a $\text{PLDL}_\Diamond$ formula. By Lemma 7 we know that it is sufficient to consider the realizability of $c(\varphi)$. Let $\mathfrak{A} = (Q, 2^{I \cup O \cup \{p\}}, q_0, \delta, \Omega)$ be a deterministic parity automaton recognizing the models of $c(\varphi)$. We turn $\mathfrak{A}$ into a parity game $\mathscr{G}$ such that Player 1 wins $\mathscr{G}$ from some dedicated initial vertex if and only if $c(\varphi)$ is realizable. To this end, we define the arena (V, V_0, V_1, E) with $V = Q \cup (Q \times 2^I)$, $V_0 = Q$, $V_1 = Q \times 2^I$, and $E = \{(q,(q,i)) \mid i \subseteq I\} \cup \{(q,i), \delta(q, i \cup o) \mid o \subseteq O \cup \{p\}\}$, i.e., Player 0 picks a subset

$i \subseteq I$ and Player O picks a subset $o \subseteq O$, which in turn triggers the (deterministic) update of the state stored in the vertices. Finally, we define the coloring $\Omega_{\mathscr{A}}$ of the arena via $\Omega_{\mathscr{A}}(q) = \Omega_{\mathscr{A}}(q,i) = \Omega(q)$.

It is straightforward to show that Player O has a winning strategy from q_0 in the parity game $(\mathscr{A}, \Omega_{\mathscr{A}})$ if and only if $c(\varphi)$ (and thus φ) is realizable. Furthermore, if Player 1 has a winning strategy, then $\mathscr{A}$ can be turned into a transducer implementing a strategy that realizes $c(\varphi)$ using V as set of states. Note that $|V|$ is doubly-exponential in $|\varphi|$, if we assume that I and O are restricted to propositions appearing in φ. As the parity game is of doubly-exponential size and has linearly many colors, we can solve it in doubly-exponential time in the size of φ. This concludes the proof. □

Also, we obtain a doubly-exponential upper bound on a variable valuation that allows to realize a given formula. A matching lower bound already holds for PLTL [22].

Corollary 2. *If a* PLDL$_\Diamond$ *formula* φ *is realizable with respect to some* α*, then it is realizable with respect to some* α *that is bounded doubly-exponentially in* $|\varphi|$.

Proof. If φ is realizable, then so is $c(\varphi)$. Using the construction proving the right-to-left implication of Lemma 7, we obtain that φ is realizable with respect to some α that is bounded by $2n+2$, where n is the size of a transducer implementing the strategy that realizes $c(\varphi)$. We have seen in the proof of Theorem 5 that the size of such a transducer is at most doubly-exponential in $|c(\varphi)|$, which is only linearly larger than $|\varphi|$. The result follows. □

6 Conclusion

We introduced Parametric Linear Dynamic Logic, which extends Linear Dynamic Logic by temporal operators equipped with parameters that bound their scope, similarly to Parametric Linear Temporal Logic, which extends Linear Temporal Logic by parameterized temporal operators. Here, the model checking problem asks for a valuation of the parameters such that the formula is satisfied with respect to this valuation on every path of the transition system. Realizability is defined in the same spirit.

We showed PLDL model checking to be complete for **PSPACE** and the realizability problem to be complete for **2EXPTIME**, just as for LTL. Thus, in a sense, PLDL is not harder than LTL. Finally, we were able to give tight exponential respectively doubly-exponential bounds on the optimal valuations for model checking and realizability.

We did not consider the assume-guarantee model checking problem here, but the algorithm solving the problem for PROMPT–LTL presented in [11] should be adaptable to PLDL as well. Another open problem concerns the computation of optimal valuations for PLDL$_\Diamond$ and PLDL$_\Box$ formulas. By exhaustive search within the bounds mentioned above, one can determine the optima. We expect this to be possible in polynomial space for model checking and in triply exponential space for realizability, which is similar to the situation for PLTL [1, 22]. Note that it is an open question whether optimal valuations for PLTL realizability can be determined in doubly-exponential time.

References

[1] Rajeev Alur, Kousha Etessami, Salvatore La Torre & Doron Peled (2001): *Parametric Temporal Logic for "Model Measuring"*. *ACM Trans. Comput. Log.* 2(3), pp. 388–407, doi:10.1145/377978.377990.

[2] Roy Armoni, Limor Fix, Alon Flaisher, Rob Gerth, Boris Ginsburg, Tomer Kanza, Avner Landver, Sela Mador-Haim, Eli Singerman, Andreas Tiemeyer, Moshe Y. Vardi & Yael Zbar (2002): *The ForSpec Temporal*

Logic: A New Temporal Property-Specification Language. In Joost-Pieter Katoen & Perdita Stevens, editors: *TACAS*, *LNCS* 2280, Springer, pp. 296–211, doi:10.1007/3-540-46002-0_21.

[3] J. Richard Büchi & Lawrence H. Landweber (1969): *Solving Sequential Conditions by Finite-State Strategies*. *Trans. Amer. Math. Soc.* 138, pp. pp. 295–311, doi:10.2307/1994916.

[4] Giuseppe De Giacomo & Moshe Y. Vardi (2013): *Linear Temporal Logic and Linear Dynamic Logic on Finite Traces*. In Francesca Rossi, editor: *IJCAI*, IJCAI/AAAI. Available at http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

[5] C. Eisner & D. Fisman (2006): *A Practical Introduction to PSL*. Integrated Circuits and Systems, Springer, doi:10.1007/978-0-387-36123-9.

[6] Emmanuel Filiot, Naiyong Jin & Jean-François Raskin (2011): *Antichains and compositional algorithms for LTL synthesis*. *Formal Methods in System Design* 39(3), pp. 261–296, doi:10.1007/s10703-011-0115-3.

[7] Bernd Finkbeiner & Sven Schewe (2013): *Bounded Synthesis*. *STTT* 15(5-6), pp. 519–539, doi:10.1007/s10009-012-0228-z.

[8] Michael J. Fischer & Richard E. Ladner (1979): *Propositional Dynamic Logic of Regular Programs*. *Journal of Computer and System Sciences* 18(2), pp. 194 – 211, doi:10.1016/0022-0000(79)90046-1.

[9] Barbara Di Giampaolo, Salvatore La Torre & Margherita Napoli (2010): *Parametric Metric Interval Temporal Logic*. In Adrian Horia Dediu, Henning Fernau & Carlos Martín-Vide, editors: *LATA*, *LNCS* 6031, Springer, pp. 249–260, doi:10.1007/978-3-642-13089-2_21.

[10] Hans W. Kamp (1968): *Tense Logic and the Theory of Linear Order*. Ph.D. thesis, Computer Science Department, University of California at Los Angeles, USA.

[11] Orna Kupferman, Nir Piterman & Moshe Y. Vardi (2009): *From Liveness to Promptness*. *Formal Methods in System Design* 34(2), pp. 83–103, doi:10.1007/s10703-009-0067-z.

[12] Martin Leucker & César Sánchez (2007): *Regular Linear Temporal Logic*. In Cliff Jones, Zhiming Liu & Jim Woodcock, editors: *ICTAC'07*, *LNCS* 4711, Springer-Verlag, Macau, China, pp. 291–305, doi:10.1007/978-3-540-75292-9_20.

[13] Satoru Miyano & Takeshi Hayashi (1984): *Alternating Finite Automata on ω-Words*. *Theor. Comput. Sci.* 32, pp. 321–330, doi:10.1016/0304-3975(84)90049-5.

[14] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of an Asynchronous Reactive Module*. In Giorgio Ausiello, Mariangiola Dezani-Ciancaglini & Simona Ronchi Della Rocca, editors: *ICALP*, *LNCS* 372, Springer, pp. 652–671, doi:10.1007/BFb0035790.

[15] Sven Schewe (2009): *Tighter Bounds for the Determinisation of Büchi Automata*. In Luca de Alfaro, editor: *FOSSACS*, *LNCS* 5504, Springer, pp. 167–181, doi:10.1007/978-3-642-00596-1_13.

[16] Christoph Schulte Althoff, Wolfgang Thomas & Nico Wallmeier (2006): *Observations on Determinization of Büchi Automata*. *Theor. Comput. Sci.* 363(2), pp. 224 – 233, doi:10.1016/j.tcs.2006.07.026.

[17] A. Prasad Sistla & Edmund M. Clarke (1985): *The Complexity of Propositional Linear Temporal Logics*. *J. ACM* 32(3), pp. 733–749, doi:10.1145/3828.3837.

[18] Ken Thompson (1968): *Programming Techniques: Regular Expression Search Algorithm*. *Commun. ACM* 11(6), pp. 419–422, doi:10.1145/363347.363387.

[19] Moshe Y. Vardi (2011): *The Rise and Fall of LTL*. In Giovanna D'Agostino & Salvatore La Torre, editors: *GandALF*, *EPTCS* 54, doi:10.4204/EPTCS.54. Invited presentation.

[20] Moshe Y. Vardi & Pierre Wolper (1994): *Reasoning About Infinite Computations*. *Inf. Comput.* 115(1), pp. 1–37, doi:10.1006/inco.1994.1092.

[21] Pierre Wolper (1983): *Temporal Logic Can be More Expressive*. *Information and Control* 56(1–2), pp. 72 – 99, doi:10.1016/S0019-9958(83)80051-5.

[22] Martin Zimmermann (2013): *Optimal Bounds in Parametric LTL Games*. *Theor. Comput. Sci.* 493, pp. 30–45, doi:10.1016/j.tcs.2012.07.039.