

Parametric Linear Dynamic Logic[☆]

Peter Faymonville, Martin Zimmermann

Reactive Systems Group, Saarland University, 66123 Saarbrücken, Germany

Abstract

We introduce Parametric Linear Dynamic Logic (PLDL), which extends Linear Dynamic Logic (LDL) by adding temporal operators equipped with parameters that bound their scope. LDL itself 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 intuitive syntax and semantics and a translation into non-deterministic Büchi automata of exponential size. However, LDL lacks capabilities to express timing constraints. By adding parameterized operators to LDL, we obtain a logic which is able to express all ω -regular properties and which 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 automata of exponential size via alternating automata. This yields polynomial space algorithms for model checking and assume-guarantee model checking and a realizability algorithm with doubly-exponential running time. All three problems are also shown to be complete for these complexity classes. Moreover, we give tight upper and lower bounds on optimal parameter values for model checking and realizability. Using these bounds, we present a polynomial space procedure for model checking optimization and an algorithm with triply-exponential running time for realizability optimization. Our results show that PLDL model checking, assume-guarantee model checking, and realizability are no harder than their respective (parametric) LTL counterparts.

Keywords: Linear Temporal Logic, Linear Dynamic Logic, Parametric Linear Temporal Logic, Model Checking, Realizability

2000 MSC: 03B44, 68Q60

[☆]A preliminary version of this work appeared in the proceedings of GandALF 2014 [1]. The research leading to this work was partially supported by the projects “TriCS” (ZI 1516/1-1) and “AVACS” (SFB/TR 14) of the German Research Foundation (DFG).

Email addresses: faymonville@react.uni-saarland.de (Peter Faymonville),
zimmermann@react.uni-saarland.de (Martin Zimmermann)

1. Introduction

Linear Temporal Logic (LTL) [2] is a popular specification language for the verification and synthesis of reactive systems and provides semantic foundations for industrial logics like PSL [3]. 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 semantics 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 [4] 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 polynomial space model checking algorithm and a doubly-exponential time 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 [3, 5]. Due to its complexity, realizability has not reached industrial acceptance (yet). First approaches relied on determinization of ω -automata, which is notoriously hard to implement efficiently [6]. More recent algorithms for realizability follow a safraless construction [7, 8], which avoids explicitly constructing the deterministic automaton, and show 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 is easily expressible by 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 to express modular verification properties, as in ForSpec [5]. Several extensions of LTL with regular expressions, finite automata, or grammar operators [9, 10, 11] 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 is the purpose of a long line of work in parametric temporal logics, e.g., parametric LTL (PLTL) [12], PROMPT-LTL [13] and parametric MITL [14]. These logics feature parameterized temporal operators to express time bounds, and either test the existence of a global bound, like PROMPT-LTL, or of individual bounds on the parameters, like PLTL.

Recently, the first drawback was revisited by De Giacomo and Vardi [15, 16] 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) [17], but the semantics follow linear-time logics. In

PDL and LDL, systems are expressed by regular expressions r with tests, and temporal requirements are specified by two basic modalities:

- $\langle r \rangle \varphi$, stating that φ should hold at some position where r matches, and
- $[r] \varphi$, stating that φ should hold at all positions where r matches.

The operators to build regular expressions from propositional formulas are as follows: sequential composition ($r_1 ; r_2$), non-deterministic 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 regular expressions, and are used to encode LTL into LDL.

For example, the program “**while** q **do** a ” with property p holding after termination 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 (every request should eventually be responded to) can be formalized as follows:

$$[\mathbf{tt}^*] (req \rightarrow \langle \mathbf{tt}^* \rangle resp).$$

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.

In PLDL, we are for example able to parameterize the eventuality of the request-response condition, denoted as

$$[\mathbf{tt}^*] (req \rightarrow \langle \mathbf{tt}^* \rangle_{\leq x} resp),$$

which states that every request has to be followed by a response within x steps.

Finally, the aforementioned property that is not expressible in LTL, (“ p holds on every even step”) can be expressed in PLDL as

$$[(\mathbf{tt} ; \mathbf{tt})^*] p.$$

Using the parameterized request-response property as the specification for a model checking problem entails determining whether there exists a valuation $\alpha(x)$ for x such that all paths of a given 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 therefore ensures the delivery of responses to requests in a timely manner.

The main result of this paper is the translation of PLDL into alternating Büchi automata of linear size. Using these automata and a generalization of the alternating color technique of [13], we obtain the following results.

First, we prove that PLDL model checking is PSPACE-complete by constructing a non-deterministic Büchi automaton of exponential size and using a modified on-the-fly non-emptiness test to obtain membership in PSPACE. PSPACE-hardness follows from the conversion of LTL to PLDL. Furthermore, we give a tight exponential bound on the satisfying valuation for model checking.

Second, we consider the PLDL assume-guarantee model checking problem and show it to be PSPACE-complete as well by extending the techniques used to show the similar result for model checking.

Third, we prove that PLDL realizability is 2EXPTIME-complete. Hardness again follows from the ability to express LTL, while membership is proven by solving a parity game constructed from a deterministic parity automaton of doubly-exponential size. Additionally, we give a tight doubly-exponential bound on the satisfying valuation for realizability.

Thus, the model checking, the assume-guarantee model checking, and the realizability problem are no harder than their corresponding variants for LTL. All three solutions to these problems are extensions of the ones for PROMPT-LTL [13].

Fourth, we investigate optimization problems for PLDL, i.e., determining the optimal valuation for a formula and a system or the tightest guarantee for realizability. While the model checking optimization problem is still solvable in polynomial space, we provide a triply-exponential time algorithm for the realizability optimization problem. This leaves an exponential gap to the decision variant, as for PLTL [18]. Both algorithms are based on exhaustive search through the bounded solution space induced by the upper bounds mentioned above.

Our translation into alternating automata is also applicable to LDL on infinite traces, while De Giacomo and Vardi [15] only considered LDL on finite traces. Furthermore, our construction differs conceptually, since we present a bottom-up procedure, while they gave a top-down construction.

2. PLDL

Let \mathcal{V} be a set of variables and let us fix a finite set P of atomic propositions which we use to build formulas and to label transition systems. 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

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

where $p \in P$, $z \in \mathcal{V}$, and where ϕ stands for arbitrary propositional formulas over P . We use the abbreviations $\mathbf{tt} = p \vee \neg p$ and $\mathbf{ff} = p \wedge \neg p$ 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. This is why both types of atoms are allowed.

The set of subformulas of φ is denoted by $\text{cl}(\varphi)$. Regular expressions are not subformulas, but the formulas appearing in the tests are, e.g., we have $\text{cl}(\langle p?; q \rangle_{\leq x} p') = \{p, p', \langle p?; q \rangle_{\leq x} p'\}$. The size $|\varphi|$ of φ is the sum of $|\text{cl}(\varphi)|$ and the sum of the lengths of the regular expressions appearing in φ (counted with multiplicity).

We define $\text{var}_{\diamond}(\varphi) = \{z \in \mathcal{V} \mid \langle r \rangle_{\leq z} \psi \in \text{cl}(\varphi)\}$ to be the set of variables parameterizing diamond-operators in φ , $\text{var}_{\square}(\varphi) = \{z \in \mathcal{V} \mid [r]_{\leq z} \psi \in \text{cl}(\varphi)\}$ to be the set of variables parameterizing box-operators in φ , and set $\text{var}(\varphi) = \text{var}_{\diamond}(\varphi) \cup \text{var}_{\square}(\varphi)$. Usually, we will denote variables in $\text{var}_{\diamond}(\varphi)$ by x and variables in $\text{var}_{\square}(\varphi)$ by y , if φ is clear from the context. A formula φ is variable-free, if $\text{var}(\varphi) = \emptyset$.

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

- $(w, n, \alpha) \models p$ if $p \in w_n$,
- $(w, n, \alpha) \models \neg p$ if $p \notin w_n$,
- $(w, n, \alpha) \models \varphi_0 \wedge \varphi_1$ if $(w, n, \alpha) \models \varphi_0$ and $(w, n, \alpha) \models \varphi_1$,
- $(w, n, \alpha) \models \varphi_0 \vee \varphi_1$ if $(w, n, \alpha) \models \varphi_0$ or $(w, n, \alpha) \models \varphi_1$,
- $(w, n, \alpha) \models \langle r \rangle \varphi$ if there exists $j \in \mathbb{N}$ s.t. $(n, n+j) \in \mathcal{R}(r, w, \alpha)$ and $(w, n+j, \alpha) \models \varphi$,
- $(w, n, \alpha) \models [r] \varphi$ if for all $j \in \mathbb{N}$ with $(n, n+j) \in \mathcal{R}(r, w, \alpha)$ we have $(w, n+j, \alpha) \models \varphi$,
- $(w, n, \alpha) \models \langle r \rangle_{\leq z} \varphi$ if there exists $0 \leq j \leq \alpha(z)$ s.t. $(n, n+j) \in \mathcal{R}(r, w, \alpha)$ and $(w, n+j, \alpha) \models \varphi$, and
- $(w, n, \alpha) \models [r]_{\leq z} \varphi$ if for all $0 \leq j \leq \alpha(z)$ with $(n, n+j) \in \mathcal{R}(r, w, \alpha)$ we have $(w, n+j, \alpha) \models \varphi$.

The relation $\mathcal{R}(r, w, \alpha) \subseteq \mathbb{N} \times \mathbb{N}$ contains all pairs (m, n) such that $w_m \cdots w_{n-1}$ matches r and is defined inductively by

- $\mathcal{R}(\phi, w, \alpha) = \{(n, n+1) \mid w_n \models \phi\}$ for propositional ϕ ,
- $\mathcal{R}(\theta?, w, \alpha) = \{(n, n) \mid (w, n, \alpha) \models \theta\}$,
- $\mathcal{R}(r_0 + r_1, w, \alpha) = \mathcal{R}(r_0, w, \alpha) \cup \mathcal{R}(r_1, w, \alpha)$,
- $\mathcal{R}(r_0; r_1, w, \alpha) = \{(n_0, n_2) \mid \exists n_1 \text{ s.t. } (n_0, n_1) \in \mathcal{R}(r_0, w, \alpha) \text{ and } (n_1, n_2) \in \mathcal{R}(r_1, w, \alpha)\}$, and

- $\mathcal{R}(r^*, w, \alpha) = \{(n, n) \mid n \in \mathbb{N}\} \cup \{(n_0, n_{k+1}) \mid \exists n_1, \dots, n_k \text{ s.t. } (n_j, n_{j+1}) \in \mathcal{R}(r, w, \alpha) \text{ for all } 0 \leq j \leq k\}$.

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 .

Example 1.

- The formula $\chi_{\infty p} := [\mathbf{tt}^*] \langle \mathbf{tt}^* \rangle p$ expresses that p holds true infinitely often.
- In general, every PTL formula [12] (and thus every LTL formula) can be translated into PLDL, e.g., $\mathbf{F}_{\leq x} p$ is expressible as $\langle \mathbf{tt}^* \rangle_{\leq x} p$ and $p \mathbf{U} q$ as $\langle p^* \rangle q$ or $\langle p^* q \rangle \mathbf{tt}$.
- The formula $[\mathbf{tt}^*](\text{req} \rightarrow \langle (\mathbf{tt}; \mathbf{tt})^* \rangle \text{resp})$ requires that every request (a position where req holds) is followed by a response (a position where resp holds) after an even number of steps. Note that the implication is not part of PLDL, but it can (here) be replaced by a disjunction.

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 (cf. [12]).

Definition 1. A PLDL formula φ is well-formed, if $\text{var}_{\diamond}(\varphi) \cap \text{var}_{\square}(\varphi) = \emptyset$.

In the following, we only consider well-formed formulas and drop the qualifier “well-formed” whenever possible.

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

Lemma 1. For every PLDL formula φ there exists an efficiently constructible (not necessarily well-formed) PLDL formula $\neg\varphi$ s.t.

1. $(w, n, \alpha) \models \varphi$ if and only if $(w, n, \alpha) \not\models \neg\varphi$, and
2. $|\neg\varphi| = |\varphi|$.

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

- | | |
|---|---|
| • $\neg(p) = \neg p$ | • $\neg(\neg p) = p$ |
| • $\neg(\varphi_0 \wedge \varphi_1) = (\neg\varphi_0) \vee (\neg\varphi_1)$ | • $\neg(\varphi_0 \vee \varphi_1) = (\neg\varphi_0) \wedge (\neg\varphi_1)$ |
| • $\neg(\langle r \rangle \varphi) = [r] \neg\varphi$ | • $\neg([r] \varphi) = \langle r \rangle \neg\varphi$ |
| • $\neg(\langle r \rangle_{\leq x} \varphi) = [r]_{\leq x} \neg\varphi$ | • $\neg([r]_{\leq y} \varphi) = \langle r \rangle_{\leq y} \neg\varphi$ |

The latter claim of Lemma 1 follows from the definition of $\neg\varphi$ while the first one can be shown by a straightforward structural induction over φ . \square

Note that negation does not necessarily preserve well-formedness, e.g., the negation of the well-formed formula $\varphi_{\boxplus} = [([p]_{\leq x} p)?)_{\leq x} p$ is $\langle ([p]_{\leq x} p)?)_{\leq x} \neg p$, which is not well-formed.

We consider the following fragments of PLDL. Let φ be a PLDL formula:

- φ is an LDL formula [15], if φ is variable-free,
- φ is a PLDL_{\diamond} formula, if $\text{var}_{\square}(\varphi) = \emptyset$, and
- φ is a PLDL_{\square} formula, if $\text{var}_{\diamond}(\varphi) = \emptyset$ and if $\neg\varphi$ is a PLDL_{\diamond} formula¹.
Note that this implies that a PLDL_{\square} formula cannot have parameterized subformulas in a test.

Every LDL, PLDL_{\diamond} , and PLDL_{\square} formula is well-formed by definition. As satisfaction of LDL formulas is independent of 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 [10], which expresses exactly the ω -regular languages, into LDL, and by a translation of LDL into Büchi automata.

Theorem 1 ([16]). *Let $L \subseteq (2^P)^{\omega}$. The following are effectively equivalent:*

1. L is ω -regular.
2. There exists an LDL formula φ such that $L = \{w \in (2^P)^{\omega} \mid w \models \varphi\}$.

A simple, but very useful property of PLDL is the monotonicity of the parameterized operators: increasing (decreasing) the values of parameters bounding diamond-operators (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 $x \in \text{var}_{\diamond}(\varphi)$ and $\beta(y) \leq \alpha(y)$ for every $y \in \text{var}_{\square}(\varphi)$. 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 PLDL_{\diamond} formula φ' whose size is at most the size of φ such that*

1. for every α there is an α' such that for all w : $(w, \alpha) \models \varphi$ implies $(w, \alpha') \models \varphi'$, and
2. for every β' there is a β such that for all w : $(w, \beta') \models \varphi'$ implies $(w, \beta) \models \varphi$.

¹The definition of PLDL_{\square} in the conference version [1] is slightly too inclusive, because it contains the formula φ_{\boxplus} . This is problematic, as we have to require the negation of a PLDL_{\square} formula to be a PLDL_{\diamond} formula.

Proof. For each r , we construct a test \hat{r} such that $\mathcal{R}(r, w, \alpha) \cap \{(n, n) \mid n \in \mathbb{N}\} = \mathcal{R}(\hat{r}, w, \alpha)$ for every w and every α . Then, $[r]_{\leq y} \psi$ and $[\hat{r}] \psi$ are equivalent, provided we have $\alpha(y) = 0$, which in combination with monotonicity is sufficient to prove our claim. We apply the following rewriting rules (in the given order) to r :

1. Replace every subexpression r'^* by $\mathbf{tt}?$, until no longer applicable.
2. Replace every subexpression $\phi; r'$ or $r'; \phi$ by $\mathbf{ff}?$ and replace every subexpression $\phi + r'$ or $r' + \phi$ by r' , where ϕ is a propositional formula, until no longer applicable.
3. Replace every subexpression $\theta_0? + \theta_1?$ by $(\theta_0 \vee \theta_1)?$ and replace every subexpression $\theta_0?; \theta_1?$ by $(\theta_0 \wedge \theta_1)?$, 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, denoted by \hat{r} . In the latter case, we define $\hat{r} = \mathbf{ff}?$.

Each rewriting step preserves the intersection $\mathcal{R}(r, w, \alpha) \cap \{(n, n) \mid n \in \mathbb{N}\}$. As \hat{r} is a test, we conclude $\mathcal{R}(r, w, \alpha) \cap \{(n, n) \mid n \in \mathbb{N}\} = \mathcal{R}(\hat{r}, w, \alpha)$ for every w and every α . Note that \hat{r} can be efficiently computed from r and its size is at most the size of 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 PLDL_{\diamond} formula that is efficiently constructible and has the desired size.

Given an α , we define α' by $\alpha'(z) = 0$ if $z \in \text{var}_{\square}(\varphi)$, and $\alpha'(z) = \alpha(z)$ otherwise. If $(w, \alpha) \models \varphi$, then $(w, \alpha') \models \varphi$ due to monotonicity. By construction of φ' , we also have $(w, \alpha') \models \varphi'$. On the other hand, if $(w, \beta') \models \varphi'$ for some β' , then $(w, \beta) \models \varphi'$ as well, where $\beta(z) = 0$, if $z \in \text{var}_{\square}(\varphi)$, and $\beta(z) = \beta'(z)$ otherwise. By construction of φ' , we conclude $(w, \beta) \models \varphi$. \square

2.1. The Alternating Color Technique and LDL_{cp}

In this subsection, we repeat the alternating color technique [13], 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 $P' = P \cup \{p\}$. 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 every 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:

- $(w, n, \alpha) \models \langle r \rangle_{cp} \psi$ if there exists a $j \in \mathbb{N}$ s.t. $(n, n + j) \in \mathcal{R}(r, w, \alpha)$, $w_n \cdots w_{n+j-1}$ 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 $(n, n + j) \in \mathcal{R}(r, w, \alpha)$ 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 relation \models and the relation \mathcal{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 PLDL_{\diamond} formula φ , let $\text{rel}(\varphi)$ be the formula obtained by inductively replacing every subformula $\langle r \rangle_{\leq x} \psi$ by $\langle \text{rel}(r) \rangle_{cp} \text{rel}(\psi)$, 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., $\text{rel}(r)$ is the regular expression obtained by applying the replacement to every test $\theta?$ in r .

Given a PLDL_{\diamond} formula φ let $c(\varphi) = \text{rel}(\varphi) \wedge \chi_{\infty p} \wedge \chi_{\infty \neg p}$ (cf. Example 1), which is an LDL_{cp} formula and only linearly larger than φ . On k -bounded and k -spaced colorings of w (for a suitable k) there is an equivalence between φ and $c(\varphi)$. The proof is similar to the original one for PROMPT-LTL [13].

Lemma 4 (cf. Lemma 2.1 of [13]). *Let φ be a 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 $k = \max_{x \in \text{var}(\varphi)} \alpha(x)$.*
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 LDL_{cp} to Alternating Büchi Automata

In this section, we show how to translate LDL_{cp} formulas into alternating Büchi word automata with linearly many states, but possibly with an exponential number of transitions, 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} . Since these problems are shown to be complete for the complexity classes PSPACE and 2EXPTIME, which allow us to construct the automata (on-the-fly), the potentially exponential number of transitions is not an issue.

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 $\delta: Q \times \Sigma \rightarrow \mathcal{B}^+(Q)$, and a set $F \subseteq Q$ of accepting states. Here, $\mathcal{B}^+(Q)$ 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 $\rho = (V, E)$ where $V \subseteq Q \times \mathbb{N}$ 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 $(q, n) \in V$: $\text{Succ}_\rho(q, n) \models \delta(q, w_n)$. Here $\text{Succ}_\rho(q, n)$ 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}_φ with linearly many states (in $|\varphi|$) and $L(\mathfrak{A}_\varphi) = \{w \in (2^{P'})^\omega \mid w \models \varphi\}$.*

To prove the theorem, we inductively construct automata \mathfrak{A}_ψ for every subformula $\psi \in \text{cl}(\varphi)$ satisfying $L(\mathfrak{A}_\psi) = \{w \in (2^{P'})^\omega \mid w \models \psi\}$.

The automata for atomic formulas are straightforward and depicted in Figures 1(a) and (b). To improve readability, we allow propositional formulas over P' as transition labels: a formula ϕ stands for all sets $A \in 2^{P'}$ with $A \models \phi$.

Furthermore, given automata \mathfrak{A}_{ψ_0} and \mathfrak{A}_{ψ_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}_{ψ_i} . The automaton $\mathfrak{A}_{\psi_0 \wedge \psi_1}$ is defined similarly, the only difference being $\delta(q_0, A) = \delta^0(q_0^0, A) \wedge \delta^1(q_0^1, A)$.

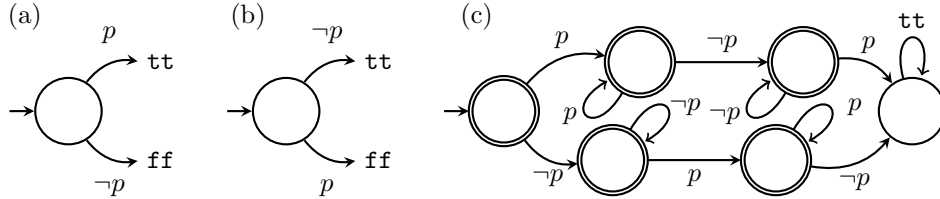


Figure 1: The automata \mathfrak{A}_p (a), $\mathfrak{A}_{\neg p}$ (b), and \mathfrak{A}_{cp} (c), which tracks changepoints.

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 defining the transition function, but we label states at which the test has to be executed, by this test. We adapt the Thompson construction [19] 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}_ψ and the automata $\mathfrak{A}_{\theta_1}, \dots, \mathfrak{A}_{\theta_k}$, where $\theta_1?, \dots, \theta_k?$ are the tests occurring in r . The ε -transitions introduced by the Thompson construction are removed during the construction, since alternating automata do not allow them. During this construction, we also ensure that the transition relation takes tests into account by introducing universal transitions that lead from a state marked with $\theta_j?$ into the corresponding automaton \mathfrak{A}_{θ_j} .

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 $\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$, 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} 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 $\Pi(q, q')$ and $m(\pi) = \{m(q_i) \mid 1 \leq i \leq k\}$ is the set of markings visited by π .

A run of \mathfrak{A} on $w_0 \cdots w_{n-1} \in \Sigma^*$ is a sequence $q_0 q_1 \cdots q_n$ of states such that for every i in the range $0 \leq i < n$ 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 from q_n via an ε -path π_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 $\theta_j?$ (denoted by labeling states with the test).

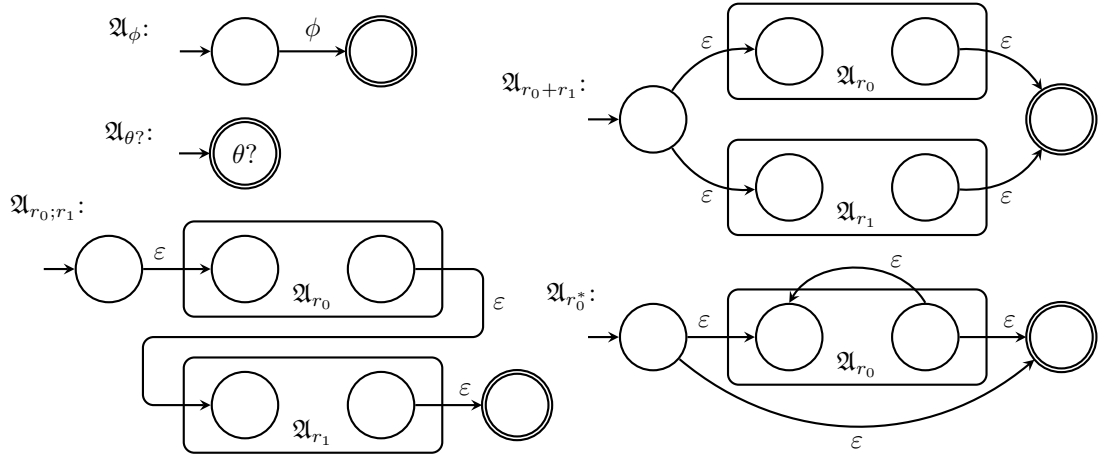


Figure 2: The inductive definition of \mathfrak{A}_r via the Thompson construction.

Lemma 5. *Let $w = w_0 w_1 w_2 \cdots \in (2^{P'})^\omega$ 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 on $w_0 \cdots w_{n-1}$ with ε -paths π_0, \dots, π_n such that $w_i w_{i+1} w_{i+2} \cdots \models \bigwedge m(\pi_i)$ for every i in the range $0 \leq i \leq n$.
2. $(0, n) \in \mathcal{R}(r, w)$.

Fix ψ and r (with tests $\theta_1?, \dots, \theta_k?$) 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 $\mathfrak{A}_{\theta_j} = (Q^j, 2^{P'}, q_0^j, \delta^j, F^j)$ 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 $\langle r \rangle \psi$ and define

$$\mathfrak{A}_{\langle r \rangle \psi} = (Q^r \cup Q' \cup Q^1 \cup \dots \cup Q^k, 2^{P'}, q_0^r, \delta, F' \cup F^1 \cup \dots \cup F^k)$$

with

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

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}_ψ and \mathfrak{A}_{θ_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 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 conjunctively add transitions that lead into the corresponding automata. The lower disjunct is similar, but ranges over ε -paths that end in a final state, which requires the A to be processed in \mathfrak{A}_ψ . Since we concatenate the automaton \mathfrak{A}_r with the automaton \mathfrak{A}_ψ , all edges leading into final states of \mathfrak{A}_r are rerouted to the successors of the initial state of \mathfrak{A}_ψ . The tests along the ε -paths 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 requires transitions that would lead \mathfrak{A}_r into a final state, we ensure the existence of a position n such that $(0, n) \in \mathcal{R}(r, w)$.

The definition of $\mathfrak{A}_{[r] \psi}$ is dual, which requires us to consider the negation of the tests: let $\mathfrak{A}_{\neg \theta_j} = (Q^j, 2^{P'}, q_0^j, \delta^j, F^j)$ for $j = 1, \dots, k$ be automata for the negation of the tests $\theta_1?, \dots, \theta_k?$ appearing in r . Recall that $\neg \theta_j$ always refers to the formula obtained by propagating the negation according to Lemma 1, and thus the automata for the negated tests can be obtained without using automata complementation. Furthermore, to construct $\mathfrak{A}_{[r] \psi}$, we remove ε -paths of \mathfrak{A}_r in a universal manner to account for the fact that the box-operator quantifies over all matches with r .

Formally, we define

$$\mathfrak{A}_{[r] \psi} = (Q^r \cup Q' \cup Q^1 \cup \dots \cup Q^k, 2^{P'}, q_0^r, \delta, Q^r \cup F' \cup F^1 \cup \dots \cup F^k)$$

where

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

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

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

- $\hat{Q}^r = Q^r \times Q^{cp}$,
- $\hat{q}_0^r = (q_0^r, q_0^{cp})$,
- $\hat{\delta}^r((q, q'), A) = \{(p, \delta^{cp}(q', A)) \mid p \in \delta^r(q, A)\}$ if $A \neq \varepsilon$, and $\hat{\delta}^r((q, q'), \varepsilon) = \{(p, q') \mid p \in \delta^r(q, A)\}$,
- $\hat{C}^r = C^r \times C^{cp}$, 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 , which restricts the matches with r recognized by \mathfrak{A}_r to those that are within at most one changepoint.

It remains to prove that the construction is correct.

Proof of Theorem 2. First, we determine the size of \mathfrak{A}_φ . 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}_{θ_j} to obtain the $\mathfrak{A}_{-\theta_j}$, instead we rely on Lemma 1. Hence, the size of \mathfrak{A}_φ is linear in the size of φ .

Thus, it remains to prove that \mathfrak{A}_φ recognizes the models of φ . We proceed by induction over the structure of φ . The induction starts for atomic formulas and the induction steps for disjunction and conjunction are trivial, hence it suffices to consider the temporal operators.

First, consider a subformula of the form $\langle r \rangle \psi$. If $w \models \langle r \rangle \psi$, then there exists a position n such that $w_n w_{n+1} w_{n+2} \dots \models \psi$ and $(0, n) \in \mathcal{R}(r, w)$. Hence,

due to Lemma 5, there is an accepting 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. Also, there is an accepting run of \mathfrak{A}_ψ on $w_n w_{n+1} w_{n+2} \cdots$, again by induction hypothesis. These runs can be “glued” together to obtain an accepting run of $\mathfrak{A}_{\langle r \rangle \psi}$ on w .

For the other direction, let ρ be an accepting run 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}_ψ and \mathfrak{A}_{θ_j} (the θ_j are the tests in r), 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 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 additionally satisfies the requirements formulated in Statement 1 of Lemma 5, due to the transitions into the test automata and applications of the induction hypothesis. Hence, we have $(0, n) \in \mathcal{R}(r, w)$. Also, from the remainder of ρ (levels greater or equal to n) we can extract an accepting run of \mathfrak{A}_ψ 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. So, we conclude $w \models \langle r \rangle \psi$.

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

The number of states of \mathfrak{A}_φ is linear in $|\varphi|$, but it is not clear that \mathfrak{A}_φ can be computed in polynomial time in $|\varphi|$, since, e.g., the transition functions of sub-automata of the form $\mathfrak{A}_{\langle r \rangle \psi}$ contain disjunctions that range over the set of ε -paths. Here, it suffices to consider simple paths, but even this restriction still allows for an exponential number of different paths. Fortunately, we do not need to compute \mathfrak{A}_φ in polynomial time. It suffices to construct it on-the-fly in polynomial space, as this is sufficient for our applications, which is clearly possible.

Furthermore, using standard constructions (e.g., [20, 21]), we can turn the alternating Büchi automaton \mathfrak{A}_φ into a non-deterministic Büchi automaton of exponential size and a deterministic parity automaton² of doubly-exponential size with exponentially many colors.

Finally, the automata we construct are weak [22], i.e., every strongly connected component either has only accepting or only non-accepting states, which allows for improved translations into non-deterministic automata: the automata for the atomic formulas are weak and taking the union or intersection of two weak automata preserves weakness. Thus, consider the automata constructed for the temporal operators: the states of the automaton for r are either all accepting or all rejecting, and once this set of states is left to some automaton checking a subformula, it is never reentered. Hence, as these sub-automata

²The states of a parity automaton are colored by $\Omega: Q \rightarrow \mathbb{N}$. 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.

are weak, the whole automaton is weak as well. However, our automata are not very weak [23, 24] (also known as linear), i.e., the automaton only has self-loops, but no non-trivial cycles, as the automata checking matches with r might have cycles of arbitrary length.

4. Model Checking

In this section, we consider the PLDL model checking problem. A (P -labeled) transition system $\mathcal{S} = (S, s_0, E, \ell)$ 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 $\ell: S \rightarrow 2^P$. An initial path through \mathcal{S} is a sequence $\pi = s_0 s_1 s_2 \dots$ of states satisfying $(s_n, s_{n+1}) \in E$ for every n . Its trace is defined as $\text{tr}(\pi) = \ell(s_0)\ell(s_1)\ell(s_2)\dots$. We say that \mathcal{S} satisfies a PLDL formula φ with respect to a variable valuation α , if we have $(\text{tr}(\pi), \alpha) \models \varphi$ 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 there exists a variable valuation α for which \mathcal{S} satisfies φ .

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 PLDL_\diamond formulas. Let φ and φ' be 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 [13] 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 change point 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 coloring function $\ell: V \rightarrow 2^{\{p\}}$ labeling vertices by p or not, and a set $F \subseteq V$ of accepting vertices. A path $v_0 v_1 v_2 \dots$ through G is pumpable, if all its blocks have at least one vertex 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 vertex.

Theorem 4 ([13]). *The pumpable non-emptiness problem for colored Büchi graphs is NLOGSPACE-complete.*

The following lemma reduces the PLDL_\diamond model checking problem to the pumpable non-emptiness problem for colored Büchi graphs of exponential size. Given a non-deterministic Büchi automaton $\mathfrak{A} = (Q, 2^{P'}, q_0, \delta, F)$ recognizing the models of $\neg \text{rel}(\varphi) \wedge \chi_{\infty p} \wedge \chi_{\infty \neg p}$ (note that $\text{rel}(\varphi)$ is negated) and a transition system $\mathcal{S} = (S, s_0, E, \ell)$, 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
- $\ell'(q, s, C) = C$.

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

Lemma 6 (cf. Theorem 4.2 of [13]). *\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 $(\text{tr}(\pi), \alpha) \not\models \varphi$. Pick α^* such that $\alpha^*(z) = 2 \cdot |Q| \cdot |S| + 2$ for every z and let π^* be the corresponding path. Applying Item 2 of Lemma 4 yields $w \not\models c(\varphi)$ for every $|Q| \cdot |S| + 1$ -bounded coloring w of $\text{tr}(\pi^*)$. Now, consider the unique $|Q| \cdot |S| + 1$ -bounded and $|Q| \cdot |S| + 1$ -spaced coloring w of $\text{tr}(\pi^*)$ that starts with p not holding true in the first position. As argued above, $w \not\models c(\varphi)$, and we have $w \models \chi_{\infty p} \wedge \chi_{\infty \neg p}$, as w is bounded. Hence, $w \models \neg \text{rel}(\varphi) \wedge \chi_{\infty p} \wedge \chi_{\infty \neg p}$, i.e., there is an accepting run $q_0 q_1 q_2 \cdots$ of \mathfrak{A} in w . This suffices to show that $(q_0, \pi_0^*, w_0 \cap \{p\})(q_1, \pi_1^*, w_1 \cap \{p\})(q_2, \pi_2^*, w_2 \cap \{p\}) \cdots$ is a pumpable fair path through $\mathfrak{A} \times \mathcal{S}$, since every block has length $|Q| \cdot |S| + 1$. This implies the existence of a repeated vertex in every block, since there are exactly $|Q| \cdot |S|$ vertices of each color.

We now consider the other direction. Thus, assume $\mathfrak{A} \times \mathcal{S}$ contains 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 $k = \max_{x \in \text{var}_o(\varphi)} \alpha(x)$. 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 $\neg \text{rel}(\varphi)$. Furthermore, w'_k is k -spaced, since we pumped each repetition k times.

Towards a contradiction assume we have $(w_k, \alpha) \models \varphi$. Applying Item 1 of Lemma 4 yields $w'_k \models c(\varphi)$, which contradicts $w'_k \models \neg \text{rel}(\varphi)$. Hence, for every α we have constructed a path of \mathcal{S} whose trace does not satisfy φ with respect to α , i.e., \mathcal{S} does not satisfy φ with respect to any α . \square

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

Corollary 1. *If there is a valuation such that \mathcal{S} satisfies a PLDL $_{\diamond}$ formula φ , then there is also one that is bounded exponentially in $|\varphi|$ and linearly in $|\mathcal{S}|$.*

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

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 already holds true for PROMPT-LTL, as noted in [13].

Now, we are able 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 [25], as LTL is a fragment of PLDL.

The following is a polynomial space algorithm, which is correct due to Lemma 6: construct $\mathfrak{A} \times \mathcal{S}$ on-the-fly and check whether it contains a pumpable fair path. Since the search for such a path can be implemented on-the-fly without having to construct the full product [13], it can be implemented using polynomial space. \square

To conclude, we prove the dual of Corollary 1 for PLDL_{\square} formulas, which will be useful when we consider the model checking optimization problem.

Lemma 7. *Let φ be a PLDL_{\square} formula and let \mathcal{S} be a transition system. There is a variable valuation α^* that is bounded exponentially in $|\varphi|$ and linearly in $|\mathcal{S}|$ such that if \mathcal{S} satisfies φ with respect to α^* , then \mathcal{S} satisfies φ with respect to every valuation.*

Proof. We begin by defining α^* : let \mathfrak{A} be a Büchi automaton recognizing the models of $c(\neg\varphi)$, which is of exponential size in $|\varphi|$. Define $k^* = 4 \cdot |\mathfrak{A}| \cdot |\mathcal{S}| + 2$ and let α^* be the variable valuation mapping every variable to k^* . Now, we consider the contrapositive and show: if there is an α such that \mathcal{S} does not satisfy φ with respect to α , then \mathcal{S} does not satisfy φ with respect to α^* .

Thus, assume there is an α and a path π such that $(\text{tr}(\pi), \alpha) \models \neg\varphi$. Note that $\neg\varphi$ is a PLDL_{\diamond} -formula. Due to monotonicity, we can assume w.l.o.g. that α maps all variables to the same value, call it k .

We denote by $\text{tr}(\pi)'$ the unique k -bounded and k -spaced p -coloring of $\text{tr}(\pi)$ that starts with p not holding true in the first position. Applying Item 1 of Lemma 4 shows that $\text{tr}(\pi)'$ satisfies $c(\neg\varphi)$. Fix some accepting run of \mathfrak{A} on $\text{tr}(\pi)'$ and consider an arbitrary block of $\text{tr}(\pi)'$: if the run does not visit an accepting state during the block, we remove infixes of the block and the run where the run reaches the same state before and after the infix and where the state of \mathcal{S} at the beginning and the end of the infix are the same, until the block has length at most $|\mathfrak{A}| \cdot |\mathcal{S}|$.

On the other hand, assume the run visits at least one accepting state during the block. Fix one such position. Then, we can remove infixes as above between the beginning of the block and the position before the accepting state is visited and between the position after the accepting state is reached and before the end of the block. What remains is a block of length at most $2 \cdot |\mathfrak{A}| \cdot |\mathcal{S}| + 1$, at most $|\mathfrak{A}| \cdot |\mathcal{S}|$ many positions before the designated position, this position itself, and at most $|\mathfrak{A}| \cdot |\mathcal{S}|$ many after the designated position.

Thus, we have constructed a $2 \cdot |\mathfrak{A}| \cdot |\mathcal{S}| + 1$ -bounded coloring $\text{tr}(\hat{\pi})'$ of a trace $\text{tr}(\hat{\pi})$ for some path $\hat{\pi}$ of \mathcal{S} , as well as an accepting run of \mathfrak{A} on $\text{tr}(\hat{\pi})'$.

Hence, $\text{tr}(\hat{\pi})'$ is a model of $c(\neg\varphi)$ and applying Item 2 of Lemma 4 shows that $\text{tr}(\hat{\pi})$ is a model of $\neg\varphi$ with respect to the variable valuation mapping every variable to $2 \cdot (2 \cdot |\mathfrak{A}| \cdot |\mathcal{S}| + 1) = k^*$. Therefore, \mathcal{S} does not satisfy φ with respect to α^* . \square

5. Assume-guarantee Model Checking

After having solved the PLDL model checking problem, we turn our attention to the assume-guarantee model checking problem. An instance of this problem consists of a transition system \mathcal{S} and two specifications, an assumption φ_A and a guarantee φ_G . Intuitively, whenever the assumption φ_A is satisfied, then also the guarantee φ_G should be satisfied.

More formally, given two transition systems $\mathcal{S} = (S, s_0, E, \ell)$ and $\mathcal{S}' = (S', s'_0, E', \ell')$ with $\ell(s_0) = \ell'(s'_0)$, we define their parallel composition

$$\mathcal{S} \parallel \mathcal{S}' = (S'', s''_0, E'', \ell'')$$

where

- $S'' = \{(s, s') \in S \times S' \mid \ell(s) = \ell'(s')\}$,
- $s''_0 = (s_0, s'_0)$,
- $((s, s'), (t, t')) \in E''$ if and only if $(s, t) \in E$ and $(s', t') \in E'$, and
- $\ell''(s, s') = \ell(s) = \ell'(s')$.

Note that parallel composition as defined here amounts to taking the intersection of the trace languages of \mathcal{S} and \mathcal{S}' . In particular, we have the following property.

Remark 1. *Let $(v_0, v'_0)(v_1, v'_1)(v_2, v'_2) \cdots$ be a path through a parallel composition $\mathcal{S} \parallel \mathcal{S}'$. Then, $v_0 v_1 v_2 \cdots$ is a path through \mathcal{S} that has the same trace as $(v_0, v'_0)(v_1, v'_1)(v_2, v'_2) \cdots$.*

An assume-guarantee specification (φ_A, φ_G) consists of two PLDL formulas, an assumption φ_A and a guarantee φ_G . We say that a finite transition system \mathcal{S} satisfies the specification, denoted by $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$, if for every countably infinite³ transition system \mathcal{S}' , if $\mathcal{S} \parallel \mathcal{S}'$ is a model of φ_A with respect to some α , then $\mathcal{S} \parallel \mathcal{S}'$ is also a model of φ_G with respect to some β [26]. For LTL specifications, this boils down to model checking the implication $\varphi_A \rightarrow \varphi_G$, but the problem is more complex in the presence of parameterized operators, as already noticed by Kupferman et al. in the case of PROMPT-LTL [13]. This is due to the fact that the variable valuation β in the problem statement above may depend on \mathcal{S}' . In the following, we extend Kupferman et al.'s algorithm for the PROMPT-LTL assume-guarantee model checking problem to the PLDL one.

The main theorem of this section reads as follows.

³This is the only place where we allow infinite transition systems (see the discussion below the proof of Lemma 8).

Theorem 5. *The PLDL assume-guarantee model checking problem is PSPACE-complete.*

To begin with, we show that we can refute such an assume-guarantee specification using a single trace per valuation β , just like in the model checking problem where we are looking for a single counterexample. However, as we consider the satisfaction of two formulas, we have to deal with two variable valuations.

Lemma 8. *Let \mathcal{S} be a transition system and let (φ_A, φ_G) be a pair of PLDL formulas. Then, $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ does not hold if and only if there is a variable valuation α such that for every variable valuation β there is a path π_β through \mathcal{S} with $(\text{tr}(\pi_\beta), \alpha) \models \varphi_A$, but $(\text{tr}(\pi_\beta), \beta) \not\models \varphi_G$.*

Proof. Let $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ not hold, i.e., there is a transition system \mathcal{S}' such that $\mathcal{S} \parallel \mathcal{S}'$ is a model of φ_A with respect to some fixed α , but $\mathcal{S} \parallel \mathcal{S}'$ is not a model of φ_G with respect to any β . Thus, for every β , we find an initial path π'_β through $\mathcal{S} \parallel \mathcal{S}'$ with $(\text{tr}(\pi'_\beta), \beta) \not\models \varphi_G$. Furthermore, $\text{tr}(\pi'_\beta)$ satisfies φ_A with respect to α , as does every trace of $\mathcal{S} \parallel \mathcal{S}'$. To conclude this direction, we apply Remark 1 to show that there exists a path π_β over \mathcal{S} such that $\text{tr}(\pi_\beta) = \text{tr}(\pi'_\beta)$ is also a trace of \mathcal{S} .

Now, assume there is a variable valuation α such that for every variable valuation β there is a path π_β through \mathcal{S} with $(\text{tr}(\pi_\beta), \alpha) \models \varphi_A$, but $(\text{tr}(\pi_\beta), \beta) \not\models \varphi_G$. Let \mathcal{S}' be a possibly infinite transition system whose traces are exactly the traces of the paths π_β . By construction, the set of traces of $\mathcal{S} \parallel \mathcal{S}'$ is equal to the set of traces of \mathcal{S} . Furthermore, every trace of $\mathcal{S} \parallel \mathcal{S}'$ satisfies φ_A with respect to α . However, for every β the trace $\text{tr}(\pi_\beta)$ of $\mathcal{S} \parallel \mathcal{S}'$ does not satisfy φ_G with respect to β . Hence, there is no β such that $\mathcal{S} \parallel \mathcal{S}'$ satisfies φ_G with respect to β , i.e., \mathcal{S}' witnesses that $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ does not hold. \square

The PROMPT-LTL assume-guarantee model-checking problem as introduced in [13] only considers the product of the given transition system \mathcal{S} with finite transition systems \mathcal{S}' . However, in this setting, one can easily construct counterexamples to the analogue of Lemma 8. Indeed the transition system \mathcal{S}' we construct while proving the second implication above is necessarily infinite for the counterexamples. If one allows infinite systems \mathcal{S}' , then the analogue is still correct, using the same proof as above. The decidability of assume-guarantee model checking restricted to finite systems \mathcal{S}' is an open problem.

Next, we observe that we again can restrict ourselves to considering PLDL $_{\diamond}$ formulas, both as the assumption and as the guarantee. This follows from Lemma 3 and the fact that the variable valuations are quantified existentially in the problem statement.

As we have to deal with two variable valuations, we have to extend the alternating-color technique to two colors, one color p for α and one color q for β . We say that $w' \in (2^{P \cup \{p, q\}})^{\omega}$ is a coloring of $w \in (2^P)^{\omega}$, if $w'_n \cap P = w_n$ for every n . Furthermore, the notions of p -change points, p -blocks, and the analogues for q are defined as expected (cf. Subsection 2.1). Consequently, the

notions of k -boundedness and k -spacedness have to explicitly refer to the color under consideration. Lemma 4 still holds for each color separately.

The following proof extends the one for the model checking problem using colored Büchi graphs. To this end, we have to adapt the definition of such a graph to two colors. Formally, a colored Büchi graph of degree two is a tuple $(V, E, v_0, \ell, F_0, F_1)$ where (V, E) is a finite directed graph, $v_0 \in V$ is the initial vertex, $\ell: V \rightarrow 2^{\{p,q\}}$ is a vertex labeling by p and q , and $F_0, F_1 \subseteq V$ are two sets of accepting vertices.

A path $v_0 v_1 v_2 \dots$ through G is pumpable, if every q -block contains a vertex repetition such that there is a p -change point in between these vertices. More formally, we require the following condition to be satisfied: if i and i' are two adjacent q -change points, then there exist j, j', j'' with $i \leq j < j' < j'' < i'$ such that $v_j = v_{j''}$ and $\ell(v_j)$ and $\ell(v_{j'})$ differ in their p -label. Furthermore, the path is fair, if both F_0 and F_1 are visited infinitely often.

The pumpable non-emptiness problem for G asks whether there exists a pumpable fair path that starts in the initial vertex.

Theorem 6 ([13]). *The pumpable non-emptiness problem for colored Büchi graphs of degree two is NLOGSPACE-complete.*

Next, we show how to reduce the PLDL assume-guarantee model checking problem to the pumpable non-emptiness problem for colored Büchi graphs of degree two. Fix an instance $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ of the problem with $\mathcal{S} = (S, s_0, E, \ell)$ and two PLDL $_{\diamond}$ formulas φ_A and φ_G .

Now, let $\mathfrak{A}_A = (Q, 2^{P \cup \{p,q\}}, q_0, \delta, F)$ be a Büchi automaton recognizing the models of $\chi_{\infty p} \wedge \chi_{\infty \neg p} \wedge \text{rel}(\varphi_A)$, and let $\mathfrak{A}_G = (Q', 2^{P \cup \{p,q\}}, q'_0, \delta', F')$ be a Büchi automaton recognizing the models of $\chi_{\infty q} \wedge \chi_{\infty \neg q} \wedge \neg \text{rel}(\varphi_G)$. Note that we need to slightly adapt the construction of \mathfrak{A}_G , as we interpret the changepoint-bounded operators in φ_G w.r.t. color changes of q , not p . Hence, instead of using the automaton \mathfrak{A}_{cp} as depicted in Figure 1(c) with transition labels p and $\neg p$, we use the one with labels q and $\neg q$ to construct \mathfrak{A}_G .

Next, we define the colored Büchi graph of degree two

$$\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S} = (Q \times Q' \times S \times 2^{\{p,q\}}, E', (q_0, q'_0, s_0, \emptyset), \ell', F_0, F_1)$$

where

- $((q_1, q_2, s, C), (q'_1, q'_2, s', C')) \in E'$ if and only if $(s, s') \in E$, $q'_1 \in \delta(q_1, \ell(s) \cup C)$, and $q'_2 \in \delta'(q_2, \ell(s) \cup C)$,
- $\ell'(q_1, q_2, s, C) = C$,
- $F_0 = F \times Q' \times S \times 2^{\{p,q\}}$, and
- $F_1 = Q \times F' \times S \times 2^{\{p,q\}}$.

Lemma 9 (cf. Lemma 6.2 of [13]). *Let $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ and $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$ be defined as above. Then, $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ does not hold if and only if $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$ has a pumpable fair path.*

Proof. Recall that changepoint-bounded operators in φ_A are evaluated with respect to the color p while the ones in φ_G are evaluated with respect to q .

Let $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ not hold. Then, due to Lemma 8, there is a variable valuation α such that for every valuation β there is an initial path π_β of \mathcal{S} such that $(\text{tr}(\pi_\beta), \alpha) \models \varphi_A$, but $(\text{tr}(\pi_\beta), \beta) \not\models \varphi_G$.

Define $k_\alpha = \max_{x \in \text{var}(\varphi_A)} \alpha(x)$, $k_{\beta^*} = 2 \cdot |Q| \cdot |Q'| \cdot |S| \cdot k_\alpha + 1$, and let β^* be such that $\beta^*(x) = 2k_{\beta^*}$ for every x . Finally, let $w^* = \text{tr}(\pi_{\beta^*})$ be the corresponding trace as above.

Then, $(w^*, \alpha) \models \varphi_A$ and Item 1 of Lemma 4 imply that every k_α -bounded (with respect to p) coloring $w^{*'}$ of w^* satisfies $\chi_{\infty p} \wedge \chi_{\infty \neg p} \wedge \text{rel}(\varphi_A)$. Similarly, $(w^*, \beta^*) \not\models \varphi_G$ and Item 2 of Lemma 4 imply that every k_{β^*} -spaced (with respect to q) coloring $w^{*'}$ of w^* does not satisfy $\text{rel}(\varphi_G)$. Hence, every such $w^{*'}$ satisfies $\chi_{\infty q} \wedge \chi_{\infty \neg q} \wedge \neg \text{rel}(\varphi_G)$.

Now, consider the unique coloring $w^{*'}$ of w^* that is k_α -bounded and k_α -spaced with respect to p , k_{β^*} -bounded and k_{β^*} -spaced with respect to q , and begins with p and q not holding true. We have $w^{*'} \models \chi_{\infty p} \wedge \chi_{\infty \neg p} \wedge \text{rel}(\varphi_A)$ and $w^{*'} \models \chi_{\infty q} \wedge \chi_{\infty \neg q} \wedge \neg \text{rel}(\varphi_G)$. Hence, there are accepting runs $q_0 q_1 q_2 \dots$ of \mathfrak{A}_A and $q'_0 q'_1 q'_2 \dots$ of \mathfrak{A}_G on $w^{*'}$.

Consider the path

$$(q_0, q'_0, v_0, w_0^{*'} \cap \{p, q\}) (q_1, q'_1, v_1, w_1^{*'} \cap \{p, q\}) (q_2, q'_2, v_2, w_2^{*'} \cap \{p, q\}) \dots$$

through $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$. Here, $w_n^{*'}$ is the n -th letter of $w^{*'}$ and $v_0 v_1 v_2 \dots$ is the path through \mathcal{S} inducing the trace $w^{*'}$.

The path is fair, as the runs are both accepting. Furthermore, it is pumpable, as the p -blocks are of size k_α , but the q -blocks are of length $k_{\beta^*} = 2 \cdot |Q| \cdot |Q'| \cdot |S| \cdot k_\alpha + 1$ and there are only $2 \cdot |Q| \cdot |Q'| \cdot |S|$ many vertices with (and without) color q .

Now, we consider the converse: assume there is a pumpable fair path

$$(q_0, q'_0, v_0, C_0) (q_1, q'_1, v_1, C_1) (q_2, q'_2, v_2, C_2) \dots$$

in $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$. W.l.o.g., we can assume the path to be ultimately periodic [13]. Hence, the maximal length of a p -block in this path, call it k_α , is well-defined. Define α via $\alpha(x) = 2k_\alpha$ for every x , fix some arbitrary β , and let $k_\beta = \max_{x \in \text{var}(\varphi_G)} \beta(x)$.

Every q -block of the pumpable path contains a vertex repetition with a p -changepoint in between. Pumping each of these repetitions k_β times yields a new path through $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$ and thereby also a path π_β through \mathcal{S} as well as accepting runs of \mathfrak{A}_A and \mathfrak{A}_G on a coloring w' of $\text{tr}(\pi_\beta)$. Hence, $w' \models \chi_{\infty p} \wedge \chi_{\infty \neg p} \wedge \text{rel}(\varphi_A)$ and $w' \models \chi_{\infty q} \wedge \chi_{\infty \neg q} \wedge \neg \text{rel}(\varphi_G)$.

By construction, w' is k_α -bounded and k_β -spaced. Thus, applying both directions of Lemma 4 yields $(\text{tr}(\pi_\beta), \alpha) \models \varphi_A$ and $(\text{tr}(\pi_\beta), \beta) \not\models \varphi_G$. Hence, for every β we have constructed a path with the desired properties. Thus, due to Lemma 8, $\langle \varphi_A \rangle \mathcal{S} \langle \varphi_G \rangle$ does not hold. \square

Now, we are able to prove the main result of this section: PLDL assume-guarantee model checking is as hard as LTL assume-guarantee model checking, i.e., PSPACE-complete.

Proof of Theorem 5. Membership is obtained by solving the pumpable non-emptiness problem for the product $\mathfrak{A}_A \times \mathfrak{A}_G \times \mathcal{S}$, which can be done in polynomial space on-the-fly, as the product is of exponential size and the algorithm checking for pumpable non-emptiness runs in logarithmic space.

For the lower bound we use a reduction from the LTL model checking problem, which is PSPACE-complete: given a transition system \mathcal{S} and an LTL formula φ , we have $\mathcal{S} \models \varphi$ if and only if $\langle \text{tt} \rangle \mathcal{S} \langle \varphi \rangle$. \square

The solution to the assume-guarantee model checking problem also solves the implication problem for PLDL: given two PLDL formulas φ and ψ , decide whether for every, possibly countably infinite, transition system \mathcal{S} the following holds: if \mathcal{S} satisfies φ with respect to some α , then \mathcal{S} satisfies ψ with respect to some β .

Theorem 7. *The PLDL implication problem is PSPACE-complete.*

Proof. Hardness follows from hardness of the LTL satisfiability problem [25].

To prove membership, we reduce the problem to the assume-guarantee model checking problem: let \mathcal{U} be a universal transition system in the sense that it contains every trace over the propositions that appear in φ and ψ . It is straightforward to show that the implication between φ and ψ is satisfied, if and only if $\langle \varphi \rangle \mathcal{U} \langle \psi \rangle$ is satisfied, as $\mathcal{U} \parallel \mathcal{S}$ has exactly the traces of \mathcal{S} . The latter problem can be solved in PSPACE, although \mathcal{U} is of exponential size, since it can be constructed on-the-fly. \square

6. 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 $\sigma: (2^I)^+ \rightarrow 2^O$ 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 $\text{outcome}(\rho)$. We say that a strategy σ for Player O is winning with respect to a variable valuation α , if we have $(\text{outcome}(\rho), \alpha) \models \varphi$ 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., whether 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 (over (I, O)).

It is well-known that ω -regular specifications, and thus all LDL_{cp} specifications, are realizable by finite-state transducers (if they are realizable at all) [27]. A transducer $\mathcal{T} = (Q, \Sigma, \Gamma, q_0, \delta, \tau)$ consists of a finite set Q of states, an input alphabet Σ , an output alphabet Γ , an initial state q_0 , a transition function $\delta: Q \times \Sigma \rightarrow Q$, and an output function $\tau: Q \rightarrow \Gamma$. The function $f_{\mathcal{T}}: \Sigma^* \rightarrow \Gamma$ implemented by \mathcal{T} is defined as $f_{\mathcal{T}}(w) = \tau(\delta^*(w))$, 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 $\sigma = f_{\mathcal{T}}$ is finite-state. The size of σ is the number of states of \mathcal{T} . The following proof is analogous to the one for PROMPT-LTL [13].

Theorem 8. *The PLDL realizability problem is 2EXPTIME-complete.*

When proving membership in 2EXPTIME, we restrict ourselves w.l.o.g. to 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 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 10 (cf. Lemma 3.1 of [13]). *A PLDL_{\diamond} formula φ is realizable over (I, O) if and only if the LDL_{cp} formula $c(\varphi)$ is realizable over $(I, O \cup \{p\})$.*

Proof. Let φ be realizable, i.e., there is a winning strategy $\sigma: (2^I)^+ \rightarrow 2^O$ with respect to some α . Now, consider the strategy $\sigma': (2^I)^+ \rightarrow 2^{O \cup \{p\}}$ 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 $\text{outcome}(\rho')$ is a k -spaced p -coloring of $\text{outcome}(\rho)$, we deduce $\text{outcome}(\rho') \models c(\varphi)$ by applying Item 1 of Lemma 4. Hence, σ' realizes $c(\varphi)$.

Now, assume $c(\varphi)$ is realized by $\sigma': (2^I)^+ \rightarrow 2^{O \cup \{p\}}$, which we can assume to be finite-state, say it is implemented by \mathcal{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 $n+1$, e.g., between changepoints at positions i and j . Let $q_0 q_1 q_2 \cdots$ be the states reached during the run of \mathcal{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 \mathcal{T} . However, the output generated by this run has only finitely many changepoints,

since the output at the states $q_{i'}, \dots, q_{j'-1}$ coincides when restricted to $\{p\}$. This contradicts the fact that \mathcal{T} implements a winning strategy, which requires in particular that every output has infinitely many changepoints. Hence, ρ is $(n+1)$ -bounded.

Let $\sigma: (2^I)^+ \rightarrow 2^O$ be defined as $\sigma(i_0 \cdots i_{n-1}) = \sigma'(i_0 \cdots i_{n-1}) \cap O$. By definition, for every play ρ consistent with σ , there is an $(n+1)$ -bounded p -coloring of $\text{outcome}(\rho)$ that is the outcome of a play that is consistent with σ' . Hence, applying Item 2 of Lemma 4 yields $(\text{outcome}(\rho), \beta) \models \varphi$, where $\beta(x) = 2n+2$ for every x . Hence, σ realizes φ with respect to β . Note that σ is also finite-state and of the same size as σ' . \square

Now, we are able to prove the main result of this section.

Proof of Theorem 8. 2EXPTIME-hardness of the PLDL realizability problem follows immediately from the 2EXPTIME-hardness of the LTL realizability problem [28], as LTL is a fragment of PLDL.

Now, consider membership and recall that we have argued that it is sufficient to consider PLDL_\diamond formulas. Thus, let φ be a PLDL_\diamond formula. By Lemma 10, 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 \mathcal{G} such that Player 1 wins \mathcal{G} from some dedicated initial vertex if and only if $c(\varphi)$ is realizable. To this end, we define the arena $\mathcal{A} = (V, V_0, V_1, E)$ 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 $o \subseteq O \cup \{p\}$, which in turn triggers the (deterministic) update of the state stored in the vertices.

Finally, we define the coloring $\Omega_{\mathcal{A}}$ of the arena via $\Omega_{\mathcal{A}}(q) = \Omega_{\mathcal{A}}(q, i) = \Omega(q)$.

It is straightforward to show that Player O has a winning strategy from q_0 in the parity game $(\mathcal{A}, \Omega_{\mathcal{A}})$ if and only if $c(\varphi)$ (and thus φ) is realizable. Furthermore, if Player 1 has a winning strategy, then \mathcal{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 exponentially many colors, we can solve it in doubly-exponential time in the size of φ . \square

Also, we obtain a doubly-exponential upper bound on a valuation that allows to realize a given formula. A matching lower bound already holds for PLTL [18].

Corollary 2. *If a PLDL_\diamond formula φ is realizable with respect to some α , then it is realizable with respect to an 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 10, 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 8 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. \square

7. Optimal Variable Valuations for Model Checking and Realizability

In this section, we turn the model checking and the realizability problem into optimization problems, e.g., the model checking optimization problem asks for the *optimal* variable valuation such that a given system satisfies the specification with respect to this valuation. Similarly, the realizability optimization problem asks for an *optimal* variable valuation such that φ is realizable with respect to this valuation. Furthermore, we are interested in computing a winning strategy for Player O witnessing realizability with respect to an optimal valuation. The definition of optimality depends on the type of formula under consideration: for PLDL_\diamond formulas, we want to minimize the waiting times while for PLDL_\square formulas, we want to maximize satisfaction times. For formulas having both types of parameterized operators, the optimization problems are undefined.

In Subsection 7.1, we show how to solve the model checking optimization problem in polynomial space. Then, in Subsection 7.2, we explain how to adapt the approach to solve the realizability optimization problem in triply-exponential time. Thus, the model checking optimization problem is in polynomial space, just as the decision problem, but there is an exponential gap between the realizability optimization problem and its decision variant. Note that this gap already exists for PLTL [18].

Both our results rely on the existence of automata of a certain size that recognize the models of a given PLDL formula with respect to a fixed variable valuation. On the one hand, it suffices to translate formulas with a single variable; on the other hand, due to some technicalities, we have to consider formulas that might additionally contain changepoint-bounded operators. The semantics of such formulas are defined as expected.

Theorem 9. *Let φ be a PLDL formula with $\text{var}(\varphi) = \{z\}$ possibly having changepoint-bounded operators and let α be a variable valuation. Then, there exists a natural number $n \in (3 \cdot (\alpha(z) + 1))^{\mathcal{O}(|\varphi|)}$ and there exist*

1. *a non-deterministic Büchi automaton of size n and*
2. *a deterministic parity automaton of size $(n!)^2$ with $2n$ many colors*

that recognize the language $L(\varphi, \alpha) = \{w \in (2^{P'})^\omega \mid (w, \alpha) \models \varphi\}$, which are both effectively constructible.

The existence of such automata is proven in Subsection 7.3 by adapting the Breakpoint construction of Miyano and Hayashi [20].

7.1. The Model Checking Optimization Problem

In this subsection, we prove that the model checking optimization problem can be solved in polynomial space. As already mentioned above, we only consider PLDL_\diamond and PLDL_\square formulas. For PLDL_\diamond formulas, optimal variable valuations are as small as possible. To abstract a variable valuation to a single value, we can either take the minimal element among the variables, i.e. the shortest waiting time, or the maximal element among the variables, i.e. the longest waiting time. Both options will provide a total ordering among variable valuations. For PLDL_\square formulas, optimal variable valuations are as large as possible. For abstraction purposes, we may again either take the maximal element, i.e. the longest guarantee, or the minimal element among the variable, i.e. the shortest guarantee. Again, this results in a total order.

Theorem 10. *Let φ_\diamond be a PLDL_\diamond formula, let φ_\square be a PLDL_\square formula, and let \mathcal{S} be a transition system. The following values are computable in polynomial space:*

1. $\min_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\diamond \text{ w.r.t. } \alpha\}} \min_{x \in \text{var}(\varphi_\diamond)} \alpha(x)$.
2. $\min_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\diamond \text{ w.r.t. } \alpha\}} \max_{x \in \text{var}(\varphi_\diamond)} \alpha(x)$.
3. $\max_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\square \text{ w.r.t. } \alpha\}} \max_{y \in \text{var}(\varphi_\square)} \alpha(y)$.
4. $\max_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\square \text{ w.r.t. } \alpha\}} \min_{y \in \text{var}(\varphi_\square)} \alpha(y)$.

Note that all other combinations are trivial due to the monotonicity properties of PLDL. Furthermore, we can restrict our attention to formulas with at least one variable, as the optimization problem is trivial otherwise.

As a first step, we show that we can reduce all problems to ones with exactly one variable, but possibly with changepoint-bounded operators.

1. Fix some $x \in \text{var}(\varphi_\diamond)$ and apply the rewriting introduced for the alternating-color technique to every variable but x to obtain the formula φ_x , which has changepoint-bounded diamond-operators as well as diamond-operators parameterized by x . Applying both directions of Lemma 4 (which also holds if we do not replace all parameterized operators) yields

$$\begin{aligned} \min_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\diamond \text{ w.r.t. } \alpha\}} \min_{x \in \text{var}(\varphi_\diamond)} \alpha(x) &= \\ \min_{x \in \text{var}(\varphi_\diamond)} \min_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_x \text{ w.r.t. } \alpha\}} \alpha(x). \end{aligned}$$

Thus, we have reduced the problem to $|\text{var}(\varphi_\diamond)|$ many optimization problems for formulas φ_x with a single variable.

2. Rename every variable in φ_\diamond to z and call the resulting formula φ'_\diamond . Due to monotonicity, minimizing the maximal parameter value for φ_\diamond yields the same value as minimizing the value of z for φ'_\diamond .
3. Fix some $y \in \text{var}(\varphi_\square)$ and denote by φ_y the formula obtained from φ_\square by replacing every subformula $[r]_{\leq y'} \psi$ with $y' \neq y$ by $[\hat{r}] \psi$, where \hat{r} is defined as in the proof of Lemma 3. Intuitively, this sets the value for every $y' \neq y$ to zero. Due to monotonicity, we have

$$\begin{aligned} \max_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_\square \text{ w.r.t. } \alpha\}} \max_{y \in \text{var}(\varphi_\square)} \alpha(y) &= \\ \max_{y \in \text{var}(\varphi_\square)} \max_{\{\alpha | \mathcal{S} \text{ satisfies } \varphi_y \text{ w.r.t. } \alpha\}} \alpha(y), \end{aligned}$$

i.e., we have reduced the problem to $|\text{var}(\varphi_\square)|$ many optimization problems for formulas φ_y with a single variable.

4. Rename every variable in φ_\square to z and call the resulting formula φ'_\square . Due to monotonicity, maximizing the minimal parameter value for φ_\square yields the same value as maximizing the value of z for φ'_\square .

First, we consider the minimization problem for a formula φ_\diamond with a single variable $x \in \text{var}_\diamond(\varphi_\diamond)$ and possibly with changepoint-bounded operators. From Corollary 1, which can easily be shown to hold for such formulas, too, we obtain an upper bound (that is exponential in $|\varphi_\diamond|$ and linear in $|\mathcal{S}|$) on the value

$$\min_{\{\alpha \mid \mathcal{S} \text{ satisfies } \varphi_\diamond \text{ w.r.t. } \alpha\}} \alpha(x).$$

Dually, for a formula φ_\square with a single variable $y \in \text{var}_\square(\varphi_\square)$ and possibly with changepoint-bounded operators, Lemma 7, which holds for such formulas, too, yields a bound k_{\max} (that is exponential in $|\varphi_\square|$ and linear in $|\mathcal{S}|$) such that either

$$\max_{\{\alpha \mid \mathcal{S} \text{ satisfies } \varphi_\square \text{ w.r.t. } \alpha\}} \alpha(y) \leq k_{\max}$$

or the maximum is equal to ∞ . Thus, in both cases, we have an exponential search space for the optimal value.

Therefore, binary search yields the optimal value, if we can solve each query “does \mathcal{S} satisfy φ with respect to α ” in polynomial space, provided α is exponential in $|\varphi|$ and linear in $|\mathcal{S}|$. To this end, we use the non-deterministic Büchi automaton \mathfrak{A} recognizing $L(\neg\varphi, \alpha)$ as given by Item 1 of Theorem 9, whose size is exponential in $|\varphi|$ and linear in $|\mathcal{S}|$. Model checking \mathcal{S} against \mathfrak{A} answers the query and is possible in polynomial space by executing the emptiness test on-the-fly without constructing \mathfrak{A} completely [10].

7.2. The Realizability Optimization Problem

In this subsection, we show how to adapt the reasoning of the model checking case to give an algorithm for the realizability optimization problem with triply-exponential running time.

Theorem 11. *Let φ_\diamond be a PLDL $_\diamond$ formula and let φ_\square be a PLDL $_\square$ formula. The following values (and winning strategies witnessing them) can be computed in triply-exponential time:*

1. $\min_{\{\alpha \mid \varphi_\diamond \text{ realizable w.r.t. } \alpha\}} \min_{x \in \text{var}(\varphi_\diamond)} \alpha(x)$.
2. $\min_{\{\alpha \mid \varphi_\diamond \text{ realizable w.r.t. } \alpha\}} \max_{x \in \text{var}(\varphi_\diamond)} \alpha(x)$.
3. $\max_{\{\alpha \mid \varphi_\square \text{ realizable w.r.t. } \alpha\}} \max_{y \in \text{var}(\varphi_\square)} \alpha(y)$.
4. $\max_{\{\alpha \mid \varphi_\square \text{ realizable w.r.t. } \alpha\}} \min_{y \in \text{var}(\varphi_\square)} \alpha(y)$.

The reductions to optimization problems for formulas with a single variable remain valid in the realizability case. However, instead of proving bounds on the search space for both the PLDL $_\diamond$ case and the PLDL $_\square$ case, we rely on Corollary 2, which proves an upper bound for the former case, and on duality:

given a PLDL formula φ over $P = I \cup O$ and its negation $\neg\varphi$ as defined in Lemma 1, define $\bar{\varphi}$ to be the formula obtained from $\neg\varphi$ by replacing each atomic proposition $p \in I$ by $\langle \text{tt} \rangle p$ and each negated proposition $\neg p$ with $p \in I$ by $\langle \text{tt} \rangle \neg p$. Here, $\langle \text{tt} \rangle$ can be understood as the PLDL-equivalent of LTL's next-operator. The realizability problems for φ and $\bar{\varphi}$ are dual, i.e., we have swapped the roles of the players and negated the specification (and used the next-operator to account for the fact that Player I is always the first to move). The following lemma formalizes this fact and relies on determinacy of parity games [29, 30], to which the realizability problem is reduced to, as shown in Section 6.

Lemma 11. *Let φ be a PLDL formula and let α be a variable valuation. Then, φ is not realizable over (I, O) with respect to α if and only if $\bar{\varphi}$ is realizable over (O, I) with respect to α .*

Thus, applying Lemma 11 and monotonicity in the case of a PLDL $_{\square}$ formula φ_{\square} with a single variable y yields

$$\max_{\{\alpha \mid \varphi_{\square} \text{ realizable w.r.t. } \alpha\}} \alpha(y) = \min_{\{\alpha \mid \bar{\varphi}_{\square} \text{ realizable w.r.t. } \alpha\}} \alpha(y) - 1,$$

i.e., to solve the PLDL $_{\square}$ optimization problem for φ_{\square} we just have to solve the problem for $\bar{\varphi}_{\square}$ and subtract one.

Thus, it remains to consider a minimization problem for a formula φ_{\diamond} with a single variable $x \in \text{var}_{\diamond}(\varphi_{\diamond})$ and possibly with changepoint-bounded operators. From Corollary 2, which holds for such formulas, too, we obtain a doubly-exponential (in $|\varphi_{\diamond}|$) upper bound on $\min_{\{\alpha \mid \varphi_{\diamond} \text{ realizable w.r.t. } \alpha\}} \alpha(x)$.

Thus, we have a doubly-exponential search space for the optimal variable valuation. Recall that Item 2 of Theorem 9 gives us a deterministic parity automaton of triply-exponential size and with exponentially many colors (both in $|\varphi_{\diamond}|$) recognizing $L(\varphi_{\diamond}, \alpha)$, as $\alpha(x)$ is bounded doubly-exponentially. This allows us to construct a parity game of triply-exponential size with exponentially many colors that is won by Player O if and only if φ_{\diamond} is realizable with respect to α . The construction is similar to the one described in the proof of Theorem 8. Such a parity game can be solved in triply-exponential time. Thus, to solve the optimization problem, we perform binary search through the doubly-exponential search space where each query can be answered in triply-exponential time by solving a parity game. Thus, the overall running time is indeed triply-exponential.

Furthermore, as already described in the aforementioned proof, a winning strategy for the parity game can be turned into a transducer witnessing realizability of φ_{\diamond} . Finally, it is straightforward to show how to turn this transducer into one for the original specifications with potentially several variables. This is trivial for the cases not requiring an application of the alternating-color technique and requires the transformation described in the proof of Lemma 10 for the other cases. This finishes the proof of Theorem 11, save for the construction of a deterministic parity automaton with the desired properties.

7.3. Small Automata for PLDL

Fix a formula φ with a single variable $z \in \text{var}(\varphi)$ possibly having change-point-bounded operators and a variable valuation α . We show how to adapt the Breakpoint construction of Miyano and Hayashi [20] to construct a non-deterministic Büchi automaton of size $(3 \cdot (\alpha(z) + 1))^{\mathcal{O}(|\varphi|)}$ to prove Item 1 of Theorem 9. The deterministic automaton for Item 2 of Theorem 9 can then be obtained by applying Schewe’s determinization construction [21], which determinizes a Büchi automaton with n states into a parity automaton with $(n!)^2$ states and $2n$ colors.

Recall that φ has a single variable. In the following, we assume that it parameterizes diamond-operators, the case of box-operators is dual and discussed below. Thus, call the variable x and let $\langle r_1 \rangle_{\leq x} \psi_1, \dots, \langle r_k \rangle_{\leq x} \psi_k \in \text{cl}(\varphi)$ be the parameterized subformulas of φ . Furthermore, let φ' be the LDL_{cp} formula obtained by removing the parameters, i.e., by replacing each $\langle r_j \rangle_{\leq x} \psi_j$ by $\langle r_j \rangle \psi_j$, and let $\mathfrak{A}_{\varphi'} = (Q, 2^{P'}, q_0, \delta, F)$ be the equivalent alternating Büchi automaton given by Theorem 2. For $j \in \{1, \dots, k\}$, we denote the set of states of the automaton \mathfrak{A}_{r_j} checking for a match with r_j by Q^{r_j} , which is a subset of Q . Furthermore, we assume the Q^{r_j} to be pairwise disjoint.

Lemma 12. *Let φ and $\mathfrak{A}_{\varphi'}$ as above and let $w \in (2^{P'})^\omega$. Then, $(w, \alpha) \models \varphi$ if and only if $\mathfrak{A}_{\varphi'}$ has an accepting run ρ that satisfies the bounded-match property: every path $(q_n, n) \cdots (q_{n+\ell}, n+\ell)$ in ρ with $q_n, \dots, q_{n+\ell} \in Q^{r_j}$ satisfies $\ell \leq \alpha(x)$ for every $j \in \{1, \dots, k\}$.*

To prove this lemma, we first need to strengthen Lemma 5 to be able to deal with parameterized formulas in the tests. Fix a regular expression r with tests $\theta_1?, \dots, \theta_k?$, which might contain parameterized operators and let \mathfrak{A}_r the ε -NFA with markings obtained from the construction described above Lemma 5. Note that the markings are the formulas $\theta_1?, \dots, \theta_k?$. Furthermore, let $w = w_0 w_1 w_2 \cdots \in (2^{P'})^\omega$, and let $w_0 \cdots w_{n-1}$ be a (possibly empty, if $n = 0$) prefix of w . The following two statements are equivalent for every α :

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

Proof of Lemma 12. The following proof is a strengthening of the proof of Theorem 2. Again, we proceed by induction over the structure of φ .

First, we consider the direction from logic to automata. The induction starts for atomic formulas and the induction steps for disjunction and conjunction are straightforward. Hence, it remains to consider the temporal operators.

Consider $\langle r \rangle \psi$. If $(w, \alpha) \models \langle r \rangle \psi$, then there exists a position n such that $(w_n w_{n+1} w_{n+2} \cdots, \alpha) \models \psi$ and $(0, n) \in \mathcal{R}(r, w, \alpha)$. Hence, due to the strengthening of Lemma 5, there is an accepting run of \mathfrak{A}_r on $w_0 \cdots w_{n-1}$ such that the tests visited during the run are satisfied with respect to α by the appropriate suffixes of w . Thus, applying the induction hypothesis yields accepting runs of

the appropriate test automata $\mathfrak{A}_{\theta'_j}$ on these suffixes which satisfy the bounded-match property. Also, there is an accepting run of $\mathfrak{A}_{\psi'}$ on $w_n w_{n+1} w_{n+2} \dots$ which satisfies the bounded-match property, again by induction hypothesis. These runs can be “glued” together to build an accepting run of $\mathfrak{A}_{\langle r \rangle \psi}'$ on w satisfying the bounded-match property.

Now, consider $\langle r_j \rangle_{\leq x} \psi_j$. If $(w, \alpha) \models \langle r_j \rangle_{\leq x} \psi_j$, then there is a position $n \leq \alpha(x)$ such that $(w_n w_{n+1} w_{n+2} \dots, \alpha) \models \psi_j$ and $(0, n) \in \mathcal{R}(r, w, \alpha)$. Recall that we removed the parameter to obtain φ' . Thus, we can argue as in the previous case and obtain runs of \mathfrak{A}_{r_j} , of the appropriate test automata $\mathfrak{A}_{\theta'_j}$, and of $\mathfrak{A}_{\psi'}$, all satisfying the induction hypothesis. In particular, the run of \mathfrak{A}_{r_j} has length $n \leq \alpha(x)$ and therefore satisfies the bounded-match property. Thus, the glued run of $\mathfrak{A}_{\langle r_j \rangle_{\leq x} \psi_j}' = \mathfrak{A}_{\langle r_j \rangle \psi'_j}$ satisfies the bounded-match property as well.

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

Now, we consider the other direction, where the induction starts for atomic formulas and the induction steps for disjunction and conjunction are again straightforward.

We continue with formulas of the form $\langle r \rangle \psi$. Let ρ be an accepting run 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 $\mathfrak{A}_{\theta'_j}$ (the θ'_j are the tests in r), 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 when restricted to states in $Q^r \setminus C^r$. Hence, we can extract an accepting run of \mathfrak{A}_r from ρ on $w_0 \dots w_{n-1}$ that additionally satisfies the requirements formulated in the strengthening of Lemma 5, due to the transitions into the test automata and an application of the induction hypothesis. Hence, we have $(0, n) \in \mathcal{R}(r, w, \alpha)$. Also, from the remainder of ρ (levels greater or equal to n) we can extract an accepting run of $\mathfrak{A}_{\psi'}$ on $w_n w_{n+1} w_{n+2} \dots$ satisfying the bounded-match property. Hence, $(w_n w_{n+1} w_{n+2} \dots, \alpha) \models \psi$ by induction hypothesis. So, we conclude $(w, \alpha) \models \langle r \rangle \psi$.

In the case of $\langle r_j \rangle_{\leq x} \psi_j$, the reasoning is similar: we have removed the parameter to obtain φ' . Thus, we end up in an analogous situation as in the previous case, but the level n satisfies $n \leq \alpha(x)$ due to the bounded-match property. This implies $(w, \alpha) \models \langle r_j \rangle_{\leq x} \psi_j$.

Again, the case for $[r] \psi$ is dual to the one for $\langle r \rangle \psi$ and the cases for the changepoint-bounded operators $\langle r \rangle_{cp} \psi$ and $[r]_{cp} \psi$ rely on the fact that \mathfrak{A}_{cp} only accepts words which have at most one changepoint. \square

Next, we show that words having runs as described in Lemma 12 can be recognized by a non-deterministic Büchi automaton: the following lemma concludes the proof of Item 1 of Theorem 9. To this end, we extend the classical Breakpoint construction [20] by counters that check the bounded-match property: the original construction yields an automaton that guesses an accepting

run of a given alternating Büchi automaton level by level, which are represented as the set of states they contain. We employ the counters γ to keep track of the length of paths in Q^{r_j} in the guessed run. If the bound $\alpha(x)$ is exceeded, then the guessed run is discarded.

Lemma 13. *There exists a non-deterministic Büchi automaton of size $(3 \cdot (\alpha(x) + 1))^{\mathcal{O}(|\varphi|)}$ that accepts $w \in (2^{P'})^\omega$ if and only if $\mathfrak{A}_{\varphi'}$ has an accepting run on w satisfying the bounded-match property.*

Proof. Let $\mathfrak{A}_{\varphi'} = (Q, 2^{P'}, q_0, \delta, F)$ be as above and recall that $Q^{r_j} \subseteq Q$ for $j \in \{1, \dots, k\}$ is the set of states which has to be left after at most $\alpha(x)$ steps in order to satisfy the bounded-match property. Define the Büchi automaton $\mathfrak{A}' = (Q', 2^{P'}, q'_0, \delta', F')$ with

- $Q' = \{(T, O, \gamma) \mid Q \supseteq T \supseteq O \text{ and } \gamma \in \{0, 1, \dots, \alpha(x)\}^{T \cap \bigcup_{j=1}^k Q^{r_j}}\},$
- $q'_0 = (\{q_0\}, \emptyset, \gamma),$ where $\gamma(q_0) = \alpha(x)$ if $q_0 \in \bigcup_{j=1}^k Q^{r_j},$
- $F' = \{(T, \emptyset, \gamma) \mid (T, \emptyset, \gamma) \in Q'\},$ and
- $\delta'((T, O, \gamma), A)$ is equal to

$$\{(T', T' \setminus F, \text{upd}(\gamma, G)) \mid \text{exists graph } G = (T \cup T', E) \text{ with } E \subseteq T \times T' \\ \text{s.t. } \text{Succ}_G(q) \models \delta(q, A) \text{ for every } q \in T\} \cap Q'$$

if $O = \emptyset$, and equal to

$$\{(T', O' \setminus F, \text{upd}(\gamma, G)) \mid O' \subseteq T' \text{ and there} \\ \text{exists graph } G = (T \cup T', E) \text{ with } E \subseteq T \times T' \\ \text{s.t. } \text{Succ}_G(q) \models \delta(q, A) \text{ for every } q \in T, \text{ and} \\ \text{Succ}_{G \upharpoonright (O \cup O')}(q) \models \delta(q, A) \text{ for every } q \in O\} \cap Q'$$

if $O \neq \emptyset$. Here, $\text{Succ}_G(q)$ denotes the set of successors of q in G , $G \upharpoonright (O \cup O')$ is the restriction of G to $O \cup O'$, and $\text{upd}(\gamma, G)$ is defined via

$$\text{upd}(\gamma, G)(q') = \min\{\alpha(x), \gamma(q) - 1 \mid (q, q') \in E \text{ and } q, q' \in Q^{r_j} \text{ for some } j\}.$$

Note that we might have $\text{upd}(\gamma, G)(q') < 0$, which implies that $\text{upd}(\gamma, G)$ is not the third component of a state of \mathfrak{A}' and explains the intersection with Q' in the definition of δ' . Thus, the counter γ prevents the simulation of runs of $\mathfrak{A}_{\varphi'}$ that violate the bounded-match property by blocking transitions.

Intuitively, the graphs used to define the transition relation δ' are building blocks for runs of $\mathfrak{A}_{\varphi'}$ that contain two levels of a run as well as the edges between them that witness the satisfaction of the transition relation δ . As already explained, the counter γ ensures that every path through some Q^{r_j} is of length $\alpha(x)$ or less.

In the first two components of \mathfrak{A}' , we implement the Breakpoint construction while we use the third component to implement a counter that checks the bounded-match property. The correctness of this construction follows directly from the correctness of the Breakpoint construction [20]. \square

In case φ has parameterized box-operators, e.g., with variable y , Lemma 12 reads as follows:

$(w, \alpha) \models \varphi$ if and only if $\mathfrak{A}_{\varphi'}$ has an accepting run ρ where every path of the form $(q_n, n) \cdots (q_{n+\ell}, n + \ell)$ in ρ with $q_n, \dots, q_{n+\ell} \in Q^{r_j}$ for some $j \in \{1, \dots, k\}$ satisfying $\ell > \alpha(x)$ may end in a terminal vertex $(q_{n+\ell}, n + \ell)$.

As before, adapting the Breakpoint construction by adding a counter mapping states in $T \cap \bigcup_{j=1}^k Q^{r_j}$ to $\{0, 1, \dots, \alpha(x)\}$ yields a non-deterministic Büchi automaton that accepts exactly those words having a run that satisfies the bounded-match property for formulas with parameterized box-operators.

8. 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 and PLDL assume-guarantee model checking to be PSPACE-complete and PLDL realizability to be 2EXPTIME-complete, 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.

With respect to the computation of optimal valuations, we have shown this to be possible in polynomial space for model checking and in triply-exponential time for realizability, which is similar to the situation for PLTL [12, 18]. Note that it is an open question whether optimal valuations for PLTL realizability can be determined in doubly-exponential time. Recently, a step towards this goal was made by giving an $\frac{1}{2}$ -approximation algorithm with doubly-exponential running time [31].

Acknowledgements. We thank the journal referees for their detailed comments, which improved the paper significantly.

References

- [1] P. Faymonville, M. Zimmermann, Parametric linear dynamic logic, in: A. Peron, C. Piazza (Eds.), GandALF 2014, volume 161 of *EPTCS*, pp. 60–73.
- [2] A. Pnueli, The temporal logic of programs, in: *FOCS 1977, IEEE, 1977*, pp. 46–57.

- [3] C. Eisner, D. Fisman, *A Practical Introduction to PSL, Integrated Circuits and Systems*, Springer, 2006.
- [4] H. W. Kamp, *Tense Logic and the Theory of Linear Order*, Ph.D. thesis, Computer Science Department, University of California at Los Angeles, USA, 1968.
- [5] R. Armoni, L. Fix, A. Flaisher, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, E. Singerman, A. Tiemeyer, M. Y. Vardi, Y. Zbar, The ForSpec temporal logic: A new temporal property-specification language, in: J.-P. Katoen, P. Stevens (Eds.), *TACAS 2002*, volume 2280 of *LNCS*, Springer, 2002, pp. 296–311.
- [6] C. Schulte Althoff, W. Thomas, N. Wallmeier, Observations on determinization of Büchi automata, *Theor. Comput. Sci.* 363 (2006) 224 – 233.
- [7] E. Filiot, N. Jin, J.-F. Raskin, Antichains and compositional algorithms for LTL synthesis, *Formal Methods in System Design* 39 (2011) 261–296.
- [8] B. Finkbeiner, S. Schewe, Bounded synthesis, *STTT* 15 (2013) 519–539.
- [9] M. Leucker, C. Sánchez, Regular linear temporal logic, in: C. Jones, Z. Liu, J. Woodcock (Eds.), *ICTAC 2007*, volume 4711 of *LNCS*, Springer-Verlag, Macau, China, 2007, pp. 291–305.
- [10] M. Y. Vardi, P. Wolper, Reasoning about infinite computations, *Inf. Comput.* 115 (1994) 1–37.
- [11] P. Wolper, Temporal logic can be more expressive, *Information and Control* 56 (1983) 72 – 99.
- [12] R. Alur, K. Etessami, S. L. Torre, D. Peled, Parametric temporal logic for “model measuring”, *ACM Trans. Comput. Log.* 2 (2001) 388–407.
- [13] O. Kupferman, N. Piterman, M. Y. Vardi, From liveness to promptness, *Formal Methods in System Design* 34 (2009) 83–103.
- [14] B. D. Giampaolo, S. La Torre, M. Napoli, Parametric metric interval temporal logic, *Theor. Comput. Sci.* 564 (2015) 131–148.
- [15] G. De Giacomo, M. Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: F. Rossi (Ed.), *IJCAI, IJCAI/AAAI*, 2013.
- [16] M. Y. Vardi, The rise and fall of LTL, in: G. D’Agostino, S. L. Torre (Eds.), *GandALF 2011*, volume 54 of *EPTCS*.
- [17] M. J. Fischer, R. E. Ladner, Propositional dynamic logic of regular programs, *Journal of Computer and System Sciences* 18 (1979) 194 – 211.
- [18] M. Zimmermann, Optimal bounds in parametric LTL games, *Theor. Comput. Sci.* 493 (2013) 30–45.

- [19] K. Thompson, Programming techniques: Regular expression search algorithm, *Commun. ACM* 11 (1968) 419–422.
- [20] S. Miyano, T. Hayashi, Alternating finite automata on ω -words, *Theor. Comput. Sci.* 32 (1984) 321–330.
- [21] S. Schewe, Tighter bounds for the determinisation of Büchi automata, in: L. de Alfaro (Ed.), *FOSSACS 2009*, volume 5504 of *LNCS*, Springer, 2009, pp. 167–181.
- [22] D. E. Muller, A. Saoudi, P. E. Schupp, Alternating automata, the weak monadic theory of trees and its complexity, *Theor. Comput. Sci.* 97 (1992) 233–244.
- [23] G. S. Rohde, Alternating Automata and the Temporal Logic of Ordinals, Ph.D. thesis, University of Illinois at Urbana-Champaign, Champaign, IL, USA, 1997.
- [24] P. Gastin, D. Oddoux, Fast LTL to Büchi automata translation, in: G. Berry, H. Comon, A. Finkel (Eds.), *CAV 2001*, volume 2102 of *LNCS*, Springer, 2001, pp. 53–65.
- [25] A. P. Sistla, E. M. Clarke, The complexity of propositional linear temporal logics, *J. ACM* 32 (1985) 733–749.
- [26] A. Pnueli, In transition from global to modular temporal reasoning about programs, in: K. R. Apt (Ed.), *Logics and Models of Concurrent Systems*, volume 13 of *NATO ASI Series*, Springer, 1985, pp. 123–144.
- [27] J. R. Büchi, L. H. Landweber, Solving sequential conditions by finite-state strategies, *Trans. Amer. Math. Soc.* 138 (1969) pp. 295–311.
- [28] A. Pnueli, R. Rosner, On the synthesis of an asynchronous reactive module, in: G. Ausiello, M. Dezani-Ciancaglini, S. R. D. Rocca (Eds.), *ICALP 1989*, volume 372 of *LNCS*, Springer, 1989, pp. 652–671.
- [29] E. A. Emerson, C. S. Jutla, Tree automata, mu-calculus and determinacy (extended abstract), in: *FOCS 1991*, IEEE, 1991, pp. 368–377.
- [30] A. Mostowski, Games with Forbidden Positions, Technical Report 78, University of Gdańsk, 1991.
- [31] L. Tentrup, A. Weinert, M. Zimmermann, Approximating optimal bounds in Prompt-LTL realizability in doubly-exponential time, 2015. ArXiv:1511.09450.