Information and Computation ••• (••••) •••–•••

Contents lists available at ScienceDirect

Information and Computation

www.elsevier.com/locate/yinco

Parametric Linear Dynamic Logic ☆

Peter Faymonville *, Martin Zimmermann

Reactive Systems Group, Saarland University, 66123 Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 15 April 2015
Available online xxxx

Keywords:
Linear Temporal Logic
Linear Dynamic Logic
Parametric Linear Temporal Logic
Model checking
Realizability

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 is an extension of Linear Temporal Logic (LTL) to all ω-regular specifications, while maintaining a translation into exponentially-sized non-deterministic Büchi automata. Since LDL cannot express timing constraints, we add parameterized operators and subsume parameterized extensions of LTL like Parametric LTL and PROMPT-LTL. Our contribution is a translation of PLDL into exponentially-sized non-deterministic Büchi automata via alternating automata. This yields PSPACE algorithms for model checking and assume-guarantee model checking and a 2EXPTIME realizability algorithm. The problems are complete for their complexity classes. We give tight bounds on optimal parameter values for model checking and realizability and present a PSPACE procedure for model checking optimization and a 3EXPTIME algorithm for realizability optimization. Our results show that these PLDL problems are no harder than their (parametric) LTL counterparts.

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.

☆ 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).

* Corresponding author.
E-mail addresses: faymonville@react.uni-saarland.de (P. Faymonville), zimmermann@react.uni-saarland.de (M. Zimmermann).

http://dx.doi.org/10.1016/j.ic.2016.07.009
0890-5401/

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 [9–11] 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:

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

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

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

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

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

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

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

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.

Please cite this article in press as: P. Faymonville, M. Zimmermann, Parametric Linear Dynamic Logic, Inf. Comput. (2016), http://dx.doi.org/10.1016/j.ic.2016.07.009