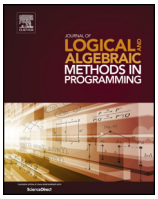
Journal of Logical and Algebraic Methods in Programming 99 (2018) 114–142

Contents lists available at ScienceDirect

Journal of Logical and Algebraic Methods in Programming

www.elsevier.com/locate/jlamp



On clock-aware LTL parameter synthesis of timed automata

Peter Bezděk *, Nikola Beneš, Ivana Černá, Jiří Barnat [1]

Faculty of Informatics, Masaryk University, Botanická 68a, 60200 Brno, Czech Republic

ARTICLE INFO

Article history:
Received 27 February 2017
Received in revised form 9 March 2018
Accepted 17 May 2018
Available online 21 May 2018

Keywords:
Parameter synthesis
Parametric timed automaton
Linear temporal logic
Clock-aware linear temporal logic

ABSTRACT

The parameter synthesis problem for timed automata is undecidable in general even for very simple reachability properties. In this paper we introduce restrictions on parameter valuations under which the parameter synthesis problem is decidable for Clock-Aware LTL properties. The investigated bounded integer parameter synthesis problem could be solved using an explicit enumeration of all possible parameter valuations. We propose an alternative symbolic zone-based method for this problem which can result in a faster computation. Our technique adapts the ideas of the automata-based approach to Clock-Aware LTL model checking of timed automata. In order to simplify the explanation of our method, we first introduce a parameter synthesis algorithm for timed automata, then we describe method for checking Clock-Aware LTL properties of timed automata and finally we combine these two methods together to provide general parameter synthesis algorithm for Clock-Aware LTL properties. To demonstrate the usefulness of our approach, we provide experimental evaluation and compare the proposed method with the explicit enumeration technique.

1. Introduction

Model checking [1] is a formal verification technique applied to check for logical correctness of discrete distributed systems. While it is often used to prove the unreachability of a bad state (such as an assertion violation in a piece of code), with a proper specification formalism, such as the *Linear Temporal Logic* (LTL), it can also check for many interesting liveness properties of systems, such as repeated guaranteed response, eventual stability, live-lock, etc.

Timed automata have been introduced in [2] and have emerged as a useful formalism for modelling time-critical systems as found in many embedded and cyber-physical systems. The formalism is built on top of the standard finite automata enriched with a set of real-time clocks and allowing the system actions to be guarded with respect to the clock valuations. In the general case, such a timed system exhibits infinite-state semantics (the clock domains are continuous). Nevertheless, when the guards are limited to comparing clock values with integers only, there exists a bisimilar finite state representation of the original infinite-state real-time system referred to as the region abstraction [3]. A practically efficient abstraction of the infinite-state space came with the so called zones [4]. The zone-based abstraction is much coarser and the number of zones *reachable* from the initial state can be significantly smaller. This in turn allows for an efficient implementation of verification tools for timed automata, see e.g. UPPAAL [5].

* Corresponding author.
E-mail addresses: bezdek@mail.muni.cz (P. Bezděk), xbenes3@fi.muni.cz (N. Beneš), cerna@fi.muni.cz (I. Černá), barnat@fi.muni.cz (J. Barnat).

[1] This work has been partially supported by the Czech Science Foundation grant No. 18-02177S.

https://doi.org/10.1016/j.jlamp.2018.05.004
2352-2208/

Very often the correctness of a time-critical system relates to proper timing, i.e. it does not only depend on the logical result of the computation, but also on the time at which the results are produced. To that end the designers are not only in the need of tools to verify correctness once the system is fully designed, but also in the need of tools that would help them derive proper time parameters of individual system actions that would make the system as a whole satisfy the required specification. After all this problem of *parameter synthesis* is more urgent in practice than the verification as such.

Related work. The problem of the existence of a parameter valuation for a reachability property of a parametric timed automaton in continuous time has been shown to be undecidable in [6,7] for a parametric timed automaton with as few as 3 clocks. This problem remains undecidable even for integer-valued parameters [8]. A partial solution for the parameter synthesis problem with reachability properties is presented in [9] where the authors provide a semi-decision algorithm which is not guaranteed to terminate in all cases. The authors also introduce a subclass of parametric timed automata, called L/U automata for which the emptiness problem is decidable. Decidability results for the class of L/U automata are further extended in [10]. In particular, the authors show that the emptiness, finiteness and universality problems of the set of parameter valuations for which there is an infinite accepting run are decidable. L/U automata introduce a significant structural restriction to the model of timed automata whereas our goal is to provide a parameter synthesis method for timed automata without any structural restrictions.

To obtain a decidable version of the parameter synthesis problem for timed automata it is sufficient to restrict parameter valuations to bounded integers. When modelling a real-time system, designers can usually provide practical bounds on the time parameters of individual system actions. Therefore, introducing a parameter synthesis method with such a restriction is still reasonable. In [11] the authors show that the problem of the existence of a bounded integer parameter value such that a given property is satisfied is PSPACE-complete for a significant number of properties, which include Timed Computational Tree Logic. They give symbolic algorithms for reachability and unavoidability properties only. While we work with the same model and restrictions as the authors in [11], the fundamental difference is in the supported class of properties where we allow the usage of the whole Clock-Aware LTL (CA-LTL) [12]. This usage of different class of properties requires us to apply a different approach.

Sometimes designers already have at least one correct instance of a parametrised system and they need to synthesise only the parameter valuations that are somewhat similar to the given one. For the given parametrised system and the parameter valuation v_0, the *inverse problem* [13] calls for the computation of a set of parameter valuations such that each valuation v from this set has the same behaviour (in terms of sets of traces) as the given valuation v_0. Algorithms solving the inverse problem are implemented in the tool IMITATOR [13]. Unlike the solution of the inverse problem, our technique targets the search for all correct parameter valuations without any previous knowledge of a correct system behaviour.

There is a plethora of derivatives of linear temporal logics for the specification of properties of real-time systems, timed automata in particular. To name at least some of them, we list TPTL [14], MTL [15], MITL [16], RTTL [17], XCTL [18], CLTL [19], LTLC [20], and CA-LTL. These logics employ various ways of expressing time aspects of underlying systems including one global time clock, time-bounded temporal operators, timing variables with quantifiers, and freeze operators. Some logics are defined with the use of time sampling semantics, which has been shown to be counter-intuitive [21]. The key aspect differentiating CA-LTL from the other logics mentioned above is the ability to properly and intuitively reason about clock values in the classical continuous-time semantics while still preserving practical efficiency of the model checking process.

Similar qualities are found in the branching time logic TCTL [22] a subset of which is actually supported by the UPPAAL tool. We stress that CA-LTL is able to reason about values of clocks in timed automata while still being practically simple enough to allow for an efficient model checking procedure as well as parameter synthesis. Note that the inclusion of time-bounded operators, such as the *until* operator of TCTL, would lead to the expressive power of at least MTL, model checking of which is considered computationally infeasible. CA-LTL can thus be seen as a practically motivated extension of LTL, which is powerful enough to express the same properties as can be expressed by the specification language of the world-wide leading timed automata verification tool UPPAAL.

Our main goal is thus to solve the parameter synthesis problem for CA-LTL properties. The corresponding solution is technically complicated. Therefore, to make the solution easier to comprehend, we explain the main ideas on less complex partial problems first, before combining them to solve our main problem. We first describe a solution to the parameter synthesis problem for LTL specifications [23] and then we continue with a description of the CA-LTL model checking technique [12] separately. Finally, we build the technically more complex solution to the parameter synthesis problem for CA-LTL specification using a combination of the previously explained ideas.

Contribution. This publication is an extended version of the SEFM 2016 conference paper [23] and also adapts techniques from [12]. The conference paper [23] solves the parameter synthesis problem for timed automata and LTL properties. The conference paper [12] introduces the logic CA-LTL and presents a CA-LTL model checking algorithm.

The main contribution of this publication is a symbolic algorithm that solves the parameter synthesis problem for timed automata with bounded integer parameters and CA-LTL properties. The algorithm is a novel and non-trivial combination of the parameter synthesis technique from [23] and the CA-LTL model checking from [12]. We introduce a new finite symbolic abstraction of timed automata with bounded integer parameters and provide an algorithm working over this abstraction. We provide experiments demonstrating the strength of our symbolic algorithm which may be faster by orders of magnitude in comparison with a naive solution.