Science of Computer Programming 111 (2015) 248–276

Contents lists available at ScienceDirect

Science of Computer Programming

www.elsevier.com/locate/scico

Fully symbolic TCTL model checking for complete and incomplete real-time systems ☆,☆☆

Georges Morbé *, Christoph Scholl *

Department of Computer Science, Georges–Köhler–Allee 51, 79110 Freiburg i. Br., Germany

ARTICLE INFO

Article history:
Received 7 June 2014
Received in revised form 30 July 2015
Accepted 3 August 2015
Available online 7 August 2015

Keywords:
Timed automata
Incomplete real-time systems
Full TCTL model checking

ABSTRACT

In this paper we present a fully symbolic TCTL model checking algorithm for real-time systems represented in a formal model called finite state machine with time (FSMT), which works on fully symbolic state sets containing both the clock values and the state variables. Our algorithm is able to verify TCTL properties on complete *and* incomplete FSMTs containing unknown components. For that purpose over-approximations of state sets fulfilling a TCTL property ϕ *for at least one* implementation of the unknown components and under-approximations of state sets fulfilling ϕ *for all* possible implementations of the unknown components are computed. We present two different methods to convert timed automata to FSMTs. In addition to FSMTs simulating pure interleaving behaviour of timed automata we can produce FSMTs with a parallelized interleaving behaviour which allows parallelism of conflict-free transitions. This can dramatically reduce the number of steps during verification. Our prototype implementation outperforms the state-of-the-art model checkers UPPAAL and RED on complete systems, and on incomplete systems our tool is able to prove interesting properties when parts of the system are unknown.

1. Introduction

Both the application areas and the complexity of real-time systems have grown with an enormous speed during the last decades. Moreover, in many applications the correct operation of real-time systems is safety-critical, which makes verification crucial. Timed automata [3,4] have become a standard for modelling real-time systems. They extend finite automata to the real-time domain by adding real-valued clock variables used to represent time. Verifying safety properties of timed automata can be reduced to the computation of all states from which unsafe states can be reached and checking whether some initial states are included in this set of states (backward model checking) or to the computation of all states which can be reached from the initial states and checking whether some unsafe states are included in this set of states (forward model checking).

Model checking approaches for timed automata can be classified into *semi-symbolic* and *fully symbolic* approaches. Semi-symbolic approaches represent discrete locations explicitly whereas sets of clock valuations are represented symbolically e.g. by *unions of clock zones*. Clock zones are convex regions that result from an intersection of clock constraints of the form

☆ This work was partly supported by the German Research Council (Deutsche Forschungsgemeinschaft – DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/).

☆☆ Parts of the article have been presented at CAV 2011 [1] and AVOCS 2013 [2].

* Corresponding authors.
E-mail addresses: morbe@informatik.uni-freiburg.de (G. Morbé), scholl@informatik.uni-freiburg.de (C. Scholl).

http://dx.doi.org/10.1016/j.scico.2015.08.002
0167-6423/

$x_i \sim d$ and $x_i - x_j \sim d$ where $d \in \mathbb{Q}$, $\sim \in \{<, \leq, =, \geq, >\}$ and x_i, x_j are clock variables. Fully symbolic approaches represent the complete state set (including valuations of both clocks and discrete variables) by a single data structure. In Section 3 we provide a more detailed review of data structures for semi-symbolic and symbolic representation of timed systems.

In this work, we present a fully symbolic model checking algorithm for a formal model for real-time systems, called finite state machines with time (FSMT), which represents real-time systems by symbolic transition functions and reset conditions. FSMTs have an elegant definition of parallel composition (where communication is performed by reading each other's state variables, reading shared input variables and shared clocks). In contrast to timed automata where parallel composition may lead to a blowup in the number of locations, the parallel composition of FSMTs just needs linear space due to the symbolic representation.

In order to verify timed automata (with additional integer variables in the state space) we present a method to convert a timed automaton into an FSMT. In addition to normal interleaving semantics (i.e. asynchronous semantics) for discrete steps of timed automata we give a symbolic representation of an FSMT simulating a *'parallelized interleaving'* behaviour, which allows parallelism of transitions causing no conflicts. This parallelized interleaving behaviour can dramatically reduce the number of steps during verification.

In contrast to [1], we do not consider invariants in timed automata or FSMTs. Invariants are a well-known means to enforce progress in timed automata. However, when considering parallel composition of several timed automata, invariants are a hidden way of communication between several components. By using invariants it is possible that a component *A enforces* that a synchronising transition in component *B* is taken without any time delay. By differentiating between *urgent* and *non-urgent* synchronisation actions we make this hidden communication mechanism explicit in the *interface* of the components.

The first part of the paper is dedicated to complete systems with possible non-determinism, but without any interaction with an environment, i.e. closed systems. We present a fully symbolic model checking algorithm for complete FSMTs able to verify complex TCTL properties. Our algorithm uses LinAIGs ('And-Inverter-Graphs with linear constraints') [5–7] to describe the state space. LinAIGs provide a fully symbolic representation both for the continuous part (i.e. the clock values) and the discrete part (i.e. the state variables). For state space compaction LinAIGs profit to a large extent from the enormous progress made in the area of SAT and SMT (SAT modulo theories) solving [8,9]. For the quantification of real-valued variables, LinAIGs make use of the Weispfenning–Loos test point method [10] which is especially suitable for LinAIG representations.

In the second part we extend our consideration to the verification of *incomplete* timed systems, i.e., timed systems that contain unknown components. Unknown components are called 'Black Boxes', whereas all known components are combined into the so-called 'White Box'. As for complete systems, there is no environment influencing the behaviour of an incomplete system. However, the white box interacts with the black box which plays a role similar to an environment of open systems. In contrast to an abstract 'environment' which enables or disables transitions synchronising with the environment, black boxes represent unknown component timed automata.

Our verification algorithm deals with different communication methods between the white box and the black box, namely shared integer variables and urgent and non-urgent synchronisation. Here we address two interesting questions: The question whether there exists a replacement of the black box such that a given property is satisfied ('realisability') and the question whether the property is satisfied for all possible replacements ('validity').

The verification of incomplete timed systems can provide three major benefits: (1) Certain verification steps can be performed at early stages of the design of a timed system, when parts of the overall system may not yet be finished, so that errors can be detected as early as possible. (2) Complex parts of a complete timed system can be abstracted away and just the relevant components for verifying a certain property are considered. (3) Finally, the location of design errors in timed systems not satisfying some property can be narrowed down by iteratively masking potentially erroneous components.

Our approach is not restricted to the verification of safety properties, but provides fully symbolic methods to do *full TCTL model checking* both for complete *and* incomplete timed systems. For incomplete systems we use over-approximations of state sets satisfying a TCTL property ϕ *for at least one* black box implementation and under-approximations of state sets satisfying ϕ *for all* possible black box implementations. Using these sets, we provide sound proofs of validity and non-realisability.

The paper is organised as follows. In Section 2 we give a brief review of timed automata, of TCTL, and LinAIGs. Here we also give more details on using urgent and non-urgent communication instead of invariants. In Section 3 we compare our approach to related work. Then we give a review of finite state machines with time (FSMT) in Section 4. In Section 5 we prepare the translation of timed automata into FSMTs by proposing two options for handling discrete steps: the optimised parallelized interleaving semantics for accelerating state space traversal and the pure interleaving semantics which corresponds to the standard asynchronous interleaving of several components. Then we present details on the translation of timed automata into FSMTs in Section 6. Our model checking algorithm for complete systems is given in Section 7. After introducing incomplete real-time systems in Section 8, we present a model checking approach for incomplete systems in Section 9, including a conversion of incomplete timed systems into incomplete FSMTs. We conclude the paper in Section 11 after presenting experimental results in Section 10.