Theoretical Computer Science 563 (2015) 43–74

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Robust reachability in timed automata and games: A game-based approach ☆

Patricia Bouyer [a,*], Nicolas Markey [a], Ocan Sankur [b]

[a] LSV, CNRS & ENS Cachan, Cachan, France
[b] Université Libre de Bruxelles, Brussels, Belgium

ARTICLE INFO

Article history:
Received 12 June 2013
Received in revised form 3 June 2014
Accepted 25 August 2014
Available online 28 August 2014
Communicated by D. Sannella

Keywords:
Timed automata
Model-checking
Robustness

ABSTRACT

Reachability checking is one of the most basic problems in verification. By solving this problem in a game, one can synthesize a strategy that dictates the actions to be performed for ensuring that the target location is reached. In this work, we are interested in synthesizing "robust" strategies for ensuring reachability of a location in timed automata. By robust, we mean that it must still ensure reachability even when the delays are perturbed by the environment. We model this perturbed semantics as a game between the controller and its environment, and solve the parameterized robust reachability problem: we show that the existence of an upper bound on the perturbations under which there is a strategy reaching a target location is EXPTIME-complete. We also extend our algorithm, with the same complexity, to turn-based timed games, where the successor state is entirely determined by the environment in some locations.

1. Introduction

Timed automata [4] are a timed extension of finite-state automata. They come with an automata-theoretic framework to design, model, verify and synthesize systems with timing constraints. One of the most basic problems in timed automata is the reachability problem: given a timed automaton and a target location, is there a path that leads to that location? This can be rephrased in the context of control as follows: is there a *strategy* that dictates how to choose time delays and edges to be taken so that a target location is reached? This problem has been solved long ago [4], generalized to timed games [6], and efficient algorithms have then been developed and implemented [29,38].

However, the abstract model of timed automata is an idealization of real timed systems. For instance, we assume in timed automata that strategies can choose the delays with arbitrary precision. In particular, the delays can be arbitrarily close to zero (the system is arbitrarily fast), and clock constraints can enforce exact delays (time can be measured exactly). Although these assumptions are natural in abstract models, they need to be justified after the design phase. Indeed the situation is different in real-world systems: digital systems have response times that may not be negligible, and control software cannot ensure timing constraints exactly, but only up to some error, caused by clock imprecisions, measurement errors, and communication delays. A good control software must be *robust*, i.e., it must ensure good behavior in spite of small imprecisions [23,28].

☆ Partly supported by the French ANR project ImpRo (ANR-10-BLAN-0317), by ERC Starting grants EQualIS (FP7-308087) and inVEST (FP7-279499), and by European FET project Cassting (FP7-601148).

* Corresponding author.
E-mail addresses: bouyer@lsv.ens-cachan.fr (P. Bouyer), markey@lsv.ens-cachan.fr (N. Markey), ocan.sankur@normalesup.org (O. Sankur).

http://dx.doi.org/10.1016/j.tcs.2014.08.014
0304-3975/

Consequently, there has been a recent effort to consider imprecisions inherent to real systems in the theory of timed systems. In particular, there have been several attempts to define convenient notions of robustness for timed automata. The approach initiated in [32,17,18] is the closest to our framework. It consists in modeling imprecisions by *enlarging* all clock constraints of the automaton by some parameter δ, that is transforming each constraint of the form $x \in [a, b]$ into $x \in [a - \delta, b + \delta]$, and in synthesizing $\delta > 0$ such that all runs of the enlarged automaton satisfy a given property. Several model-checking algorithms for timed automata were then re-visited and extended to this setting in [8,9,11,34] and symbolic algorithms were studied in [26,16]. This notion of robustness also corresponds to a concrete implementation semantics; in fact robustness implies implementability in a simplified model of a micro-processor, see [17].

In all these works, the robustness condition is satisfied if there exists an enlargement parameter for which *all* runs of the enlarged automaton satisfy a given property. In other terms, for any choice of the delays and edges, and any possible perturbation, the given property must hold. Although this is a convenient notion for e.g. safety properties, this does not capture the system's ability to adapt to perturbations that were observed earlier in a given run. In fact, if perturbations have accumulated and deviated the system from its course, a smart control software should be able to adapt its action to correct this. Thus, we believe a good notion of robustness can be defined by a game played between two players: *Controller* with a reachability objective, and *Perturbator* with the complementary objective.

Following this idea, we are interested in the synthesis of robust strategies in timed automata and games for reachability objectives, taking into account response times and imprecisions. In our semantics, which is parameterized by δ_P and δ_R with $0 < \delta_P \leq \delta_R$, Controller chooses to delay an amount $d \geq \delta_R$ after which the guard of a chosen edge is satisfied, and the system delays d' and takes the edge, where d' is chosen by Perturbator satisfying $|d - d'| \leq \delta_P$. Observe that the guard may not be satisfied after the delay d', but the chosen edge is taken whatever the perturbations. We say that a given location is *robustly reachable* if there exist parameters $0 < \delta_P \leq \delta_R$ such that Controller has a winning strategy ensuring that the location is reached against any strategy of Perturbator. To simplify the presentation, but w.l.o.g., we assume in this paper that $\delta = \delta_P = \delta_R$; our algorithm can easily be adapted to the general case (see Section 4).

The main result of this paper is the following: We show that deciding the existence of $\delta > 0$, and of a strategy for the controller so as to ensure reachability of a given location in a turn-based timed game (whatever the imprecision, up to δ), is EXPTIME-complete. Moreover, if there is a strategy, we can compute a *uniform* one, which is parameterized by δ, using *shrunk difference bound matrices* (shrunk DBMs) that we introduced in [36]. In this case, our algorithm provides a bound $\delta_0 > 0$ such that the strategy is correct for all $\delta \in [0, \delta_0]$. Our strategies also give quantitative information on how perturbations accumulate or can compensate. Technically, our work extends shrunk DBMs by *constraints*, and establishes non-trivial algebraic properties of this data structure (Section 3). The main result is then obtained by transforming the infinite-state game into a finite abstraction, which we prove can be used to symbolically compute a winning strategy, if any (Section 4).

A variant of our semantics was studied in [15] for timed games with *fixed* parameters. In this variant, Controller can only suggest delays and edges whose guards are satisfied after any perturbation; thus, this is a *conservative* variant of our semantics. When δ is fixed, the semantics can be encoded by a usual timed game, and standard algorithms can be applied. Whether one can synthesize $\delta > 0$ for which the controller has a winning strategy was left as a challenging open problem. We solve this problem for reachability objectives in turn-based timed games. Parameterized reachability and safety in the conservative semantics of [15] are studied in [37]. See also Section 2.3 for notes on related work.

2. Robust reachability in timed automata

2.1. Timed automata and games, and robust reachability

Given a finite set of clocks $\mathcal{C}$, we call *valuations* the elements of $\mathbb{R}^{\mathcal{C}}_{\geq 0}$, which are nonnegative real vectors of dimension $|\mathcal{C}|$. For a subset $R \subseteq \mathcal{C}$ and a valuation v, $v[R \leftarrow 0]$ is the valuation defined by $v[R \leftarrow 0](x) = v(x)$ for $x \in \mathcal{C} \setminus R$ and $v[R \leftarrow 0](x) = 0$ for $x \in R$. Given $d \in \mathbb{R}_{\geq 0}$ and a valuation v, the valuation $v + d$ is defined by $(v + d)(x) = v(x) + d$ for all $x \in \mathcal{C}$. We extend these operations to sets of valuations in the obvious way. We write $\vec{0}$ for the valuation that assigns 0 to every clock.

An atomic clock constraint is a formula of the form $k \prec x \prec' l$ or $k \prec x - y \prec' l$ where $x, y \in \mathcal{C}$, $k, l \in \mathbb{Z} \cup \{-\infty, \infty\}$ and $\prec, \prec' \in \{<, \leq\}$. A *guard* is a conjunction of atomic clock constraints. A valuation v satisfies a guard g, denoted $v \models g$, if all constraints are satisfied when each $x \in \mathcal{C}$ is replaced with $v(x)$. We write $\Phi_{\mathcal{C}}$ for the set of guards built on $\mathcal{C}$.

Definition 2.1. A *(two-player) turn-based timed game* $\mathcal{A}$ is a tuple $(\mathcal{L}_C \cup \mathcal{L}_P, \mathcal{C}, \ell_0, E)$, where $\mathcal{L} = \mathcal{L}_C \cup \mathcal{L}_P$ is a finite set of locations satisfying $\mathcal{L}_C \cap \mathcal{L}_P = \emptyset$, $\mathcal{C}$ is a finite set of clocks, $E \subseteq \mathcal{L} \times \Phi_{\mathcal{C}} \times 2^{\mathcal{C}} \times \mathcal{L}$ is a set of edges, and $\ell_0 \in \mathcal{L}_C$ is the initial location. An edge $e = (\ell, g, R, \ell')$ is also written as $\ell \xrightarrow{g,R} \ell'$. A *timed automaton* is a turn-based timed game with $\mathcal{L}_P = \emptyset$.

Standard semantics of timed automata is usually given as a timed transition system [4]. To capture robustness, we define the semantics as a game where perturbations in delays are uncontrollable. The semantics extends naturally to turn-based timed games, where our game semantics simply gives control over perturbations to one of the players.

Given a turn-based timed game $\mathcal{A} = (\mathcal{L}_C \cup \mathcal{L}_P, \mathcal{C}, \ell_0, E)$ and $\delta > 0$, we define the *perturbation game* of $\mathcal{A}$ w.r.t. δ as a two-player turn-based game $\mathcal{G}_\delta(\mathcal{A})$ between players *Controller* and *Perturbator*. Intuitively the semantics is the following: At