A Game Approach to Determinize Timed Automata*

Amélie Stainer
Supervisors : Nathalie Bertrand and Thierry Jéron
Inria Rennes Bretagne Atlantique and Université Rennes 1, France
amelie.stainer@inria.fr

Abstract

Timed automata are a usual model for the specification of real time systems. Their determinization is a key issue for several problems such as implementability, diagnosis or test generation. There exist non-determinizable timed automata and, unfortunately, determinizability of timed automata is an undecidable problem. Recently, two approaches have been proposed: a determinization procedure [2] which terminates on some determinizable classes and an algorithm [6] that produces a deterministic over-approximation. In this paper, we propose a game approach covering both. This method is at once more general than the procedure of [2] and more precise than the approximation algorithm of [6].

1 Introduction

Timed automata, presented in [1], form a usual model for the specification of real-time systems. Their determinization is a key issue for several problems like implementability, diagnosis or test generation.The set of determinizable timed automata is a strict subclass of timed automata. Moreover, the problem of the determinizability of a timed automaton is undecidable [4]. Therefore, in order to determinize timed automata, there are two possible choices: either to restrict to determinizable classes like the procedure of [2] or to choose to ensure termination by accepting approximations like the algorithm of [6] which produces a deterministic over-approximation.

Our method combines both approaches. It is inspired by a game approach presented by Bouyer, Chevalier and D'Souza in [3] to decide diagnosability with fixed ressources of a timed automaton. It is based on the construction of a finite turn-based safety game between players Spoiler and Determinizator. If Determinizator has a winning strategy then it yields a deterministic timed automaton accepting exactly the same traces as the initial automaton, else it produces a deterministic over-approximation. Our approach is at once more general than the procedure of [2] and more precise than the algorithm of [6].

The plan of the paper is as follows. In section 2, after introducing notations and definitions, we will present the methods developed in [2] and [6]. Our approach is detailed in section 3 and compared to the other ones in section 4.

2 Definitions and related work

2.1 Definitions

Preliminaries. Given X a finite set of clocks, a clock *valuation* is a mapping $v : X \to \mathbb{R}_+$. We note $\overline{0}$ the valuation that assigns 0 to all clocks. If v is a valuation over X and $t \in \mathbb{R}_+$, then $v+t$ denotes the valuation which assigns to every clock $x \in X$ the value $v(x)+t$. For $Y \subseteq X$ we write $v_{[Y \leftarrow 0]}$ for the valuation equal to v on $X \setminus Y$ and to $\overline{0}$ on Y, and $v_{|Y}$ for the valuation v restricted to

*This work has been partially supported by ANR Testec.

Y. Given M a non-negative integer, an *M-bounded guard* (or guard) over X is a finite conjunction of constraints of the form $x \sim c$ where $x \in X$, $c \in [0,M] \cap \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$. We denote by $G_M(X)$ the set of M-bounded guards over X. Given v a valuation and g a guard we write $v \models g$ whenever v satisfies g. A *timed trace* over Σ is an element of $(\mathbb{R}_+ \times \Sigma)^*$. Given X and X' two finite sets of clocks, a *relation* beetween clocks of X and those of X' is a finite conjunction C of atomic constraints of the form $x - x' \sim c$ where $x \in X$, $x' \in X'$, $\sim \in \{<, =, >\}$ and $c \in \mathbb{N}$.

Timed automata. A *timed automaton* (TA for short) is a tuple $\mathscr{A} = (L, l_0, \Sigma, X, M, E)$ such that: (i) L is a finite set of locations, (ii) $l_0 \in L$ is the initial location, (iii) Σ is a finite alphabet, (iv) X is a finite set of clocks, (v) $M \in \mathbb{N}$ and (vi) $E \subseteq L \times G_M(X) \times \Sigma \times 2^X \times L$ is a finite set of edges. The constant M is called the maximal constant of $\mathscr{A}$. The semantics of $\mathscr{A}$ is given by a timed transition sytem $\mathscr{T}_\mathscr{A} = (S, s_0, (\mathbb{R}_+ \times \Sigma), \rightarrow)$ with set of states $S = L \times \mathbb{R}_+^X$, initial state $s_0 = (l_0, \overline{0})$ and transition relation $\rightarrow \subseteq S \times (\mathbb{R}_+ \times \Sigma) \times S$ composed of moves of the form $(l, v) \xrightarrow{\tau, a} (l', v')$ whenever there exists an edge $(l, g, a, Y, l') \in E$ such that $v + \tau \models g$ and $v' = (v + \tau)_{[Y \leftarrow 0]}$. A *run* ρ of $\mathscr{A}$ is a finite sequence of moves starting in s_0, i.e., $\rho = s_0 \xrightarrow{\tau_1, a_1} s_1 \cdots \xrightarrow{\tau_k, a_k} s_k$. The trace of ρ is $\tau_1, a_1 \ldots \tau_k, a_k$. We write $traces(\mathscr{A})$ for the set of traces of $\mathscr{A}$, that is the set of traces u such that there exists a run which reads u. A timed automaton $\mathscr{A}$ is *deterministic* (abbreviated DTA) whenever for every trace u, there is at most one run in $\mathscr{A}$ reading u.

The classical region construction. The regions form a partition of valuations over a given set of clocks. It allows to make abstractions in order to decide properties like the reachability of a location. We let X be a finite set of clocks, and $M \in \mathbb{N}$. We write $\lfloor t \rfloor$ and $\{t\}$ for the integer part and the fractional part of a real t, respectively. We define the equivalence relation $\equiv_{X,M}$ between valuations over X as follows: $v \equiv_{X,M} v'$ if (i) for every clock $x \in X$, $v(x) \leq M$ iff $v'(x) \leq M$; (ii) for every clock $x \in X$, if $v(x) \leq M$, then $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$ and (iii) for every pair of clocks $(x, y) \in X^2$ such that $v(x) \leq M$ and $v'(x) \leq M$, $\{v(x)\} \leq \{v(y)\}$ iff $\{v'(x)\} \leq \{v'(y)\}$. The equivalence relation is called the *region equivalence* for the set of clocks X w.r.t. M, and an equivalence class is called a *region*. A region r' is a time-successor of a region r if there is $v \in r$ and $t \in \mathbb{R}_+$ such that $v + t \in r'$.

2.2 Related work

Determinization procedure. In [2], the authors present an abstract determinization procedure. Given a timed automaton $\mathscr{A}$, this procedure produces a language-equivalent infinite timed tree by an unfolding. In this tree, a new clock is reset at each step: clocks of $\mathscr{A}$ are represented by these new clocks in a mapping. Then, the infinite tree is symbolically determinized: a node of this tree represents a set of nodes of the initial one. Next, the tree is folded up and the algorithm terminates provided there is a bound on the number of clocks needed in all mappings of the node. The clock-boundedness assumption is satisfied by unfoldings of TAs of several classes which are, as a consequence, determinizable. The resulting deterministic automaton is doubly exponential in the size of the initial automaton.

Deterministic over-approximation. In contrast, a procedure which always terminates is proposed in [6]. Given a timed automaton $\mathscr{A}$, their algorithm produces a deterministic over-approximation $\mathscr{B}$ of $\mathscr{A}$, that is a DTA $\mathscr{B}$ such that $traces(\mathscr{A}) \subseteq traces(\mathscr{B})$. This over-approximation is constructed by simulation of $\mathscr{A}$ with only the information carried by clocks of $\mathscr{B}$. Thus, a location of $\mathscr{B}$ is a state estimate of $\mathscr{A}$ consisting of a set of pairs (l, v) where l is a location of $\mathscr{A}$ and v a valuation over the set of clocks of $\mathscr{A}$ and $\mathscr{B}$. This method is based on the use of a fixed finite automaton which governs the behavior of the resets of the clocks of $\mathscr{B}$.

3 Our approach

3.1 Informal definition of the game

Given a timed automaton $\mathscr{A}$, we want to build a precise deterministic over-approximation of $\mathscr{A}$, an exact one if possible, with the ressources (k, M_Y), that is a deterministic timed automaton $\mathscr{B}$ with a set Y of k clocks and a maximal constant M_Y such that $traces(\mathscr{A}) \subseteq traces(\mathscr{B})$. The clocks in Y are used to estimate the values of the clocks of $\mathscr{A}$ in the guards of $\mathscr{B}$. Our approach is based on the construction of a finite turn-based safety game $\mathscr{G}_{\mathscr{A},(k,M_Y)}$ between players Spoiler and Determinizator, where Determinizator wants to avoid states marked in gray. Spoiler chooses an action and when to fire it: its choices are pairs (a, r) where a is an action and r a region over Y. Determinizator chooses to reset a clock of Y or not during this action. Then, any positional strategy for Determinizator yields a deterministic TA, where two successive moves of Spoiler and Determinizator together form a transition. A location of this timed automaton is a state of Spoiler in the game. A state of the game is a state estimate of $\mathscr{A}$ consisting in a set of configurations associating a location with a relation between clocks in Y and clocks of $\mathscr{A}$ together with a marking. A configuration is marked if it is obtained by an over-approximation. The state is marked in gray if all configurations are marked, meaning that as long as there is an unmarked configuration in a state, traces ending in this state are not over-approximated. We then have the two following properties:

- each strategy for Determinizator in the game $\mathscr{G}_{\mathscr{A},(k,M_Y)}$ yields a DTA with ressources (k, M_Y) which at least accepts the traces of $\mathscr{A}$,
- each winning strategy for Determinizator in the game $\mathscr{G}_{\mathscr{A},(k,M_Y)}$ yields a DTA with ressources (k, M_Y) which accepts exactly the same traces as $\mathscr{A}$.

Note that due to the undecidability of the problem of determinizability with fixed ressources [4, 7], there necessarily exist TAs which can be determinized with ressources (k, M_Y) such that the game $\mathscr{G}_{\mathscr{A},(k,M_Y)}$ admits no winning strategy for Determinizator.

3.2 An example

The DTA of Figure 2 is the result of our approach for the TA of Figure 1 with ressources $(1, 1)$ and Determinizator's winning strategy given with bold arrows on the game $\mathscr{G}_{\mathscr{A},(1,1)}$ of Figure 3.

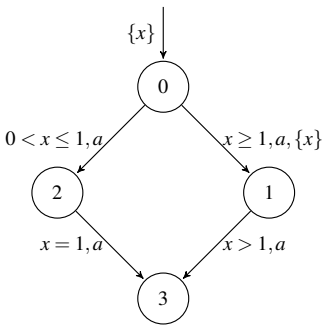

Figure 1: Example of a TA $\mathscr{A}$.

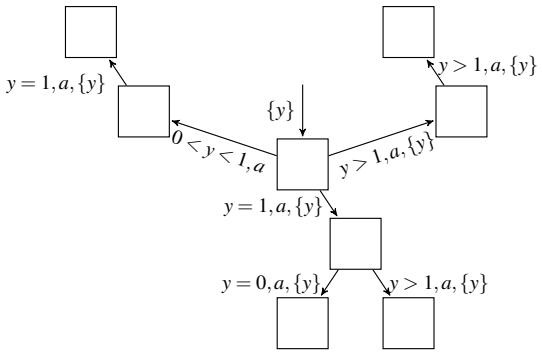

Figure 2: The DTA corresponding to the winning strategy for Determinizator represented in the Figure 3.

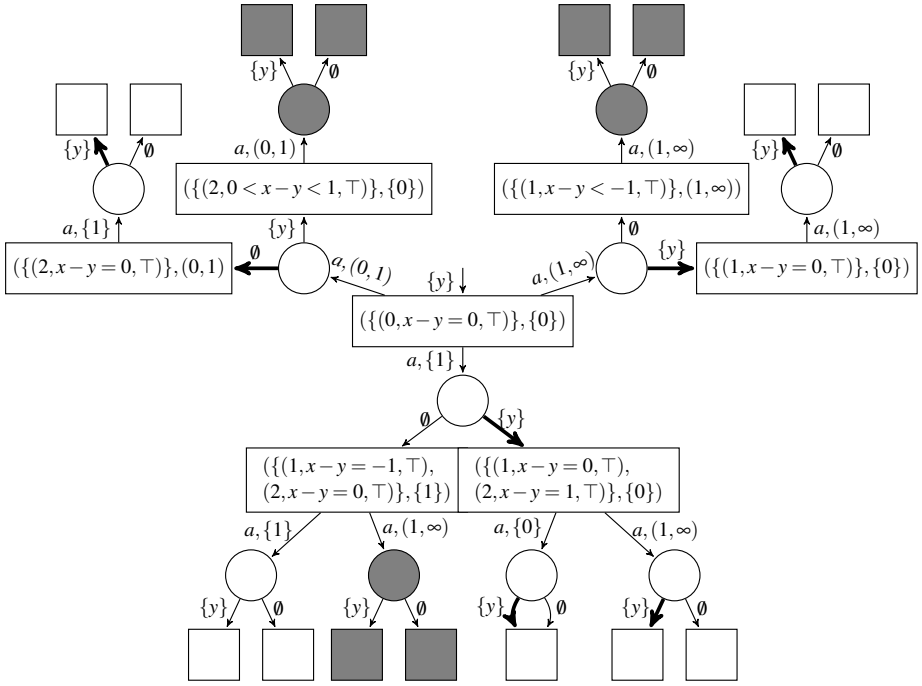

Figure 3: The game $\mathscr{G}_{\mathscr{A},(1,1)}$ and an example of winning strategy for Determinizator.

3.3 How to choose a good strategy

The game is a finite turn-based safety game, therefore, if there is a winning strategy for Determinizator in the game, it can be computed by standard techniques [5]. But, in case there is no winning strategy, we want to choose a good losing one. To this aim, we define a natural partial order over the set of strategies of Determinizator. Given two strategies S_1 and S_2 for Determinizator, S_1 is better than S_2 if, against every strategy of Spoiler, either S_1 wins, or it loses later than S_2. With respect to this order, the set of optimal strategies is computable by a simple fixed point. Moreover, thanks to game theory and a contruction to remove resets of several clocks, we can show that for our criterion, allowing multiple clocks resets does not increase the power of Determinizator.

3.4 Elements of formalization

We give here some indications about the formalization of our approach. Given $\mathscr{A} = (L, l_0, X, M, E)$ a TA and ressources (k, M_Y), our method constructs a game $\mathscr{G}_{\mathscr{A},(k,M_Y)} = (V, \tilde{\Sigma}, \delta, v_0, \mathit{Gray})$ where:

- $V = V_S \sqcup V_D$ is a set of states such that $V_S \subseteq \mathscr{P}(L \times \mathscr{P}(\mathbb{R}_+^{X \cup Y}) \times \{\top, \bot\}) \times \mathit{Reg}_{M_Y}^Y$ and $V_D \subseteq \mathscr{P}(L \times \mathscr{P}(\mathbb{R}_+^{X \cup Y}) \times \{\top, \bot\}) \times \mathit{Reg}_{M_Y}^Y$ are the sets of states of Spoiler and of Determinizator,
- $\tilde{\Sigma} = (\Sigma \times \mathit{Reg}_{M_Y}^Y) \cup (\{\emptyset\} \cup \{\{y\} \mid y \in Y\})$ is the alphabet,
- $\delta \subseteq (V_S \times (\Sigma \times \mathit{Reg}_{M_Y}^Y) \times V_D) \cup (V_D \times (\{\emptyset\} \cup \{\{y\} \mid y \in Y\}) \times V_S)$ is the set of edges,
- $v_0 = (\{(l_0, \{\overline{0} + t \in \mathbb{R}^{X \cup Y} \mid t \in \mathbb{R}_+\}, \top)\}, \{0\}) \in V_S$ is the initial state,
- $\mathit{Gray} = \{(\{(l_j, C_j, \bot)_{j \in J}\}, r') \in V\}$ is the set of states which Determinizator wants to avoid.

A state of the game is of the form $(\{(l_j, C_j, b_j)_{j \in J}\}, r')$ where J is finite, $l_j \in L$, C_j is a relation between the clocks of X and Y, b_j is the marker of the configuration (l_j, C_j, b_j) and r' is a region over Y. *Gray* is the set of states such that every configuration (l_j, C_j, b_j) is marked ($b_j = \bot$).

Successors of a state of Spoiler. Given $q_S = (\{(l_j, C_j, b_j)_{j\in J}\}, r'_Y) \in V_S$, its successors are in V_D, and given a choice of Spoiler $(a, r_Y) \in \Sigma \times Reg^Y_{M_Y}$, $q_S \xrightarrow{a,r_Y} (\{(l_i, C_i, b_i)_{i\in I}\}, r_Y)$ if $\{(l_i, C_i, b_i)_{i\in I}\} = Succ(q_s, (a, r_Y))$, where $Succ(q_s, (a, r_Y))$ is the union of elementary successors by the choice (a, r_Y) of each configuration of q_S if r_Y is a time-successor of r'_Y, otherwise $Succ(q_s, (a, r_Y))$ is empty. More precisely, the set $Succ_e((l, C, b), (a, r_Y))$ of elementary successors of a configuration (l, C, b) by a choice of Spoiler (a, r_Y) is defined as follows:

$$Succ_e((l,C,b),(a,r_Y)) = \left\{ (l',C',b') \;\middle|\; \begin{array}{l} \exists l \xrightarrow{g,a,X'}_{\mathscr{A}} l' \wedge [r_Y \cap C]_{|X} \cap g \neq \emptyset, \\ C' = U_X(r_Y, C, X'), \\ b' = ([r_Y \cap C]_{|X} \cap \neg g = \emptyset) \wedge b \end{array} \right\}$$

where $U_X(r_Y, C, X')$ is the update of the relation between clocks caused by the reset of $X' \subseteq X$.

Successors of a state of Determinizator. Given $q_D = (\{(l_i, C_i, b_i)_{i\in I}\}, r_Y) \in V_D$ its successors are in V_S and given a reset choice of Determinizator $p \in (\{\emptyset\} \cup \{\{y\} \mid y \in Y\})$, $q_D \xrightarrow{p} (\{(l_i, C'_i, b_i)_{i\in I}\}, r'_Y)$ if $((\forall i \in I,\, C'_i = U_Y(r_Y, C_i, p)) \wedge r'_Y = r_{Y[p\leftarrow 0]})$ where $U_Y(r_Y, C_i, p)$ is the update of the relation between clocks caused by the reset of the clock in p if $p \neq \emptyset$, and $U_Y(r_Y, C_i, p) = C_i$ if $p = \emptyset$.

4 Comparison with other methods

4.1 More precise than the deterministic over-approximation of [6]

Krichen et Tripakis proposed a method to compute deterministic over-approximations of TA based on the use of a finite automaton, called *skeleton*, which governs the behavior of the clocks used to estimate clocks of $\mathscr{A}$ [6]. First, choosing a good skeleton is an issue; moreover one can easily observe that finite skeletons are not sufficient to preserve determinism: some deterministic timed automata are strictly over-approximated by this method. On the contrary, with sufficient ressources, our approach preserves determinism. In fact, any strategy of Determinizator in our game can be seen as a timed skeleton. As a consequence, solving our game is a way to obtain a good timed skeleton.

4.2 More flexible than the determinization of [2]

The method of [2] for an automaton $\mathscr{A}$ is based on a mapping from $\mathscr{A}$'s clocks to the clocks of the construction. First of all, for any automaton $\mathscr{A}$ which can be determinized by this method into a timed automaton with ressources (k, M_Y), Determinizator has a winning strategy in the game $\mathscr{G}_{\mathscr{A},(k,M_Y)}$, and our method preserves the doubly exponential upperbound. As a consequence, our method is more general than the one of [2]. Moreover, our approach is better in two main aspects. Firstly, our precise marking of configurations allows to treat some cases of traces inclusions, for instance when traces of a non-determinizable branch are included in a determinizable branch. Thus, some trivial counter-examples of the method [2] are easily treated by ours. Secondly, our relations are much more flexible than the mapping of [2]. More precisely, a relation of the form $x - y = c$ where c is an integer, allows to express exactly the guards of $\mathscr{A}$ and it permits to treat some cases where there is an explosion of the possible values of a clock in a state but all with the same fractional part. This second point allows us to slightly improve several determinizability results of [2]. In fact, the aim of these results is to find decidable sufficient conditions to ensure that an automaton is determinizable. If the unfolding of an automaton is such that the number of clocks needed in locations is bounded by a constant γ, [2] terminates over this automaton.

More generaly, if we can limit the number of sets of clocks which have same fractional parts in locations of the unfolding, our approach yields a DTA accepting exactly the same traces as $\mathscr{A}$. Of course, these hypotheses are undecidable, so we look for stronger decidable conditions. Several ones are presented in [2], thanks to our approach, they can be slightly extended while preserving determinizability. For example, ressources needed for the determinization of TAs with integer resets are reduced: our approach determinizes these TAs with a single clock, the same maximal constant and the same size.

5 Conclusion

Our approach improves two existing methods of determinization or deterministic over-approximation and includes their respective advantages: exactness if possible and termination. Moreover, it treats some cases of traces inclusion between two branches of the timed automaton which we want to determinize. Our method produces an over-approximation, but one can similarly construct an under-approximation which can be useful for other problems like languages inclusion. Krichen and Tripakis use their deterministic over-approximation for test generation with respect to a time conformance relation under the strong assumption of input enabledness. We plan to adapt our method in an over-approximation in outputs and under-approximation in inputs which would respect the time conformance relation without the input enabledness assumption.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] Christel Baier, Nathalie Bertrand, Patricia Bouyer, and Thomas Brihaye. When are timed automata determinizable? In *International Colloquium on Automata, Languages and Programming*, pages 43–54, 2009.

[3] Patricia Bouyer, Fabrice Chevalier, and Deepak D'Souza. Fault diagnosis using timed automata. In *Foundations of Software Science and Computational Structures*, pages 219–233, 2005.

[4] Olivier Finkel. Undecidable problems about timed automata. In *Formal Modeling and Analysis of Timed Systems*, pages 187–199, 2006.

[5] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research.*

[6] Moez Krichen and Stavros Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.

[7] Stavros Tripakis. Folk theorems on the determinization and minimization of timed automata. *Information Processing Letters*, 99(6):222–226, 2006.