Available online at www.sciencedirect.com

SCIENCE DIRECT®

Electronic Notes in Theoretical Computer Science 133 (2005) 41–60

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Analysis of Real-Time Systems with CTL Model Checkers

Mustapha Bourahla[1]

Computer Science Department, University of Biskra, Algeria

Mohamed Benmohamed[2]

Computer Science Department, University of Constantine, Algeria

Abstract

This paper presents a new method for model checking dense real-time systems. The dense real-time system is modeled by a timed automaton and the property is specified with the temporal logic TCTL. Specification of the TCTL property is reduced to CTL and its temporal constraints are captured in a new timed automaton. This timed automaton will be composed with the original timed automaton specifying the real-time system under analysis. Then, the product timed automaton will be abstracted using partition refinement of state space based on strong bi-simulation. The result is an untimed automaton modulo the TCTL property which represents an equivalent finite state system to be model checked using existing CTL model checking tools.

Keywords: Real-Time Systems, Model Checking, Timed Automaton, Strong Bi-simulation, Partition Refinement.

1 Introduction

Many formal frameworks that have been proposed to reason about real-time systems are based on timed automata [2]. These automata are equipped with clocks, variables used to measure time, ranging over the non negative real numbers (R^+). Consequently, the state space is infinite and cannot be explicitly represented by enumerating all states. Among the different description

[1] Email: mbourahla@hotmail.com
[2] Email: ibnm@yahoo.fr

doi:10.1016/j.entcs.2004.08.057

languages for specifying real-time requirements, we are particularly interested in the temporal logic TCTL [1,10]. Because of the introduction of dense time, which makes quantitative reasoning more complicated, real-time model-checking tools are less successful than CTL model-checking tools. Most published real-time model checking algorithms are based on backward/forward reachability analysis [4,12] which is implemented in several successful verification tools. Other algorithms are based on compositional techniques [11] where components of the real-time systems are gradually moved during the verification into the specification. Real-time model checking techniques based on partition refinement [17,19] build a symbolic state space that is as coarse as possible. Starting from some (implicit) initial partition, the partition is iteratively refined until the verification problem can be decided.

In this paper we propose a reduction technique from TCTL model-checking to CTL model-checking without changing the classic CTL model checking algorithm. Thus, the generated model can be translated to a language of one of the existing CTL model checkers to be checked directly. Given a timed automaton $\mathcal{A}$ and a TCTL formula ψ, we construct an equivalent CTL formula φ and a new timed automaton $\mathcal{A}^+$ augmented with behavior and specification clocks which are extracted from the formula ψ. We prove that $\mathcal{A}$ satisfies ψ if and only if $\mathcal{A}^+$ satisfies φ.

The transition system modeling the behavior of the constructed timed automaton $\mathcal{A}^+$ comprises two kinds of transitions, namely timeless actions representing the discrete evolutions of the system, and time lapses corresponding to the passage of time. Due to density of time, there are infinitely many time transitions. A finite model can be obtained by defining an appropriate equivalence relation inducing a finite number of equivalence classes. The main idea behind these relations is that they abstract away from the exact amount of time elapsed. An important problem consists in constructing the quotient of a labeled transition system (representing a timed automaton) with respect to an equivalence relation.

In this paper we have defined an equivalence relation based on strong bisimulation [13], which is used by our algorithm to generate the quotient graph. Each edge in the timed automaton represents a discrete transition which has information concerning the source and target states, the enabling condition and the set of clocks to be reset after making this transition. Initially, the timed automaton represents the states of timed system as blocks (zones) of states (called also, symbolic states). We call this the initial partition of states. We refine any source block of states if there is an outgoing edge with an enabling condition (which is a constraint) formula different from *true*, using the invariant of the block of states and the enabling condition of this transition.

The produced sub-blocs represent classes of equivalent states where each sub-block has new invariant that either satisfies or does not satisfy the enabling condition. The refinement process will terminate if there is no block of states to be refined. The produced finite graph abstracting temporal constraints can be too large. It is possible to reduce this finite graph using one approach of the equivalence based reductions [5,19].

Related work

A similar work can be found in [6], where the model checking is based on-the-fly exploration of a simulation graph. The simulation graph is the graph reachable, generated from the region graph [1] and from an initial region. Thus, because the nodes in the simulation graph are region sets and only discrete transitions are explicit, while time passes implicitly inside the nodes, the simulation graph is much smaller than the region graph. The simulation graph is used to solve the model checking problem for a proposed automata-based branching-time temporal logic ($TECTL^*_\exists$). The on-the-fly model checking procedure consists in solving the emptiness problem, that is, in checking whether an automaton (the automaton product of the system automaton and the property automaton) has an infinite execution sequence that satisfies a given acceptance condition. In our work, the property automaton capturing the temporal constraints, is automatically generated from the TCTL specification. Our quotient graph is produced directly from the initial automaton of timed system specification, which resembles the simulation graph, without passing by the region graph. The quotient graph is coarser than the initial automaton but finer, and therefore bigger, than the initial graph. Another algorithm that also combines the on-the-fly and the symbolic approaches has been proposed in [16]. In that work, a symbolic graph is dynamically constructed by the verification procedure, according to the formula (specified in an extended temporal logic of μ-Calculus) to be checked.

A similar reduction for a derivate of dense time TCTL (TCTL with freeze quantifiers [3]) is given in [9]. This approach augments the region graph used in [1] by a new atomic proposition and new transitions to handle the reset quantifier. Another related work can be found in [8], where verification is performed by translating TCTL (interpreted over discrete time) into CTL by adding an additional specification clock to the model. So, to model-check the augmented model, the CTL logic is extended, and thus the model-checker, too. The closest work to ours for the time abstraction based on equivalence can be found in [18]. Where the algorithm in [5] for minimal-model generation (which is an enhancement of the algorithm of Paige and Tarjan [15] to avoid refining unreachable classes) is adapted to infinite state space of timed automaton.

This new algorithm uses decision procedures for computing intersection, set difference and predecessors of classes, and testing whether a class is empty. Also, the TCTL specification is reduced to CTL logic extended with new atomic propositions to deal with the specification constraints. Then, a TCTL model checker has been developed based on techniques of the classic CTL model-checker. Other techniques are based on abstraction of the constraints specified in the system and in the property, using the framework of predicate abstractions as abstract interpretation [7,14].

The rest of the paper is organized as follows. In Section 2, we present the formalism of timed automata used to specify timed systems. The logic TCTL and our approach of transformation of the TCTL specifications to CTL specifications are presented in Section 3. Also, the proof of the transformation correctness is presented in this section. In Section 4, we present our algorithm for generating finite bi-similar graphs of the timed systems. Section 5 explains the method of using these graphs for CTL model checking and how the results can be projected back to original timed systems. At the end, a conclusion is given.

2 System Specification with Timed Automata

We model real-time systems by timed automata [2] which extend the automata formalism by adding clocks. Clocks are real-valued variables increasing uniformly with time. Several independent clocks may be defined for the same timed automaton. A timed automaton $\mathcal{A}$ is a tuple $\langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I}, \rangle$, where:

- $\mathcal{Q}$ is a finite set of locations. We denote by $q_0 \in \mathcal{Q}$ the initial location.
- $\mathcal{X}$ is a finite set of clocks. A valuation v is a function that assigns a non negative real-value $v(x) \in R^+$ to each clock $x \in \mathcal{X}$. The valuation $v[X := \delta]$ assigns the value δ to all clocks in the set X. The set of valuations is denoted $\mathcal{V}_\mathcal{X}$. For $\delta \in R^+$, $v + \delta$ denotes the valuation v' such that $v'(x) = v(x) + \delta$ for all $x \in \mathcal{X}$.
- $\mathcal{E}$ is a finite set of edges. Each edge $e \in \mathcal{E}$ is a tuple $\langle q, \theta, X, q' \rangle$ where
  - · $q, q' \in Q$ are the source and the target locations respectively,
  - · $\theta \in \Theta$ is an associated clock constraint which governs the triggering of the transition. It is called its enabling condition or its guard. We denote the set of constraints over $\mathcal{X}$ by Θ. A constraint is defined as a conjunction of atoms of the form $x \sim c$, where $x \in \mathcal{X}$, $\sim \in \{<, \leq, >, \geq, =\}$ and c is a natural constant.
  - · $X \subseteq \mathcal{X}$ is the set of clocks to be reset after making this transition.
- $\mathcal{L} : \mathcal{Q} \rightarrow 2^{AP}$ is a function that associates to each location a set of atomic

propositions from the set AP.

- $\mathcal{I}$ is a function that associates a condition $\mathcal{I}(q) \in \Theta$ to every location $q \in \mathcal{Q}$ called the invariant of q.

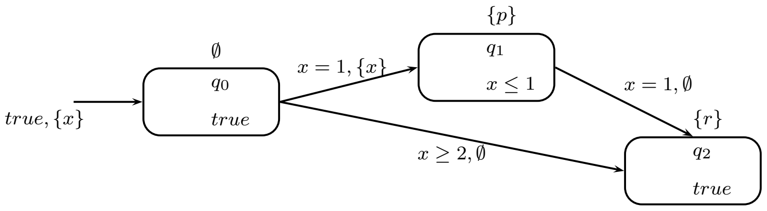


Fig. 1. Timed Automaton

Figure 1 shows an example of a timed automaton where $AP = \{p, r\}$ and $\mathcal{Q} = \{q_0, q_1, q_2\}$. A state of $\mathcal{A}$ is a pair $\langle q, v\rangle \in \mathcal{Q} \times \mathcal{V}_{\mathcal{X}}$ such that v satisfies $\mathcal{I}(q)$. The initial state is the pair $\langle q_0, v_0\rangle$ such that $v_0(x) = 0$ for all $x \in \mathcal{X}$. Let $\mathcal{S}$ denote the set of states of $\mathcal{A}$. We will refer to $\mathcal{L}(s)$ by $\mathcal{L}(q)$, for all $s \in \mathcal{S}$, where $s = \langle q, v\rangle$. The set $\mathcal{S}$ can be partitioned to zones (symbolic states). A zone $z = (q, \mathcal{V}_z)$ is a set of states from $\mathcal{S}$ which are associated with the same discrete state $q \in \mathcal{Q}$ and a convex set of valuations $\mathcal{V}_z = \{v \mid \exists\langle q, v\rangle \in \mathcal{S}\}$. The state of a timed system can be changed through an edge that changes the location and resets some of the clocks (discrete transition), or by letting time pass without changing the location (time transition).

Let $e = \langle q, \theta, X, q'\rangle \in \mathcal{E}$ be an edge. The state $\langle q, v\rangle$ has a discrete transition to $\langle q', v'\rangle$, denoted $\langle q, v\rangle \xrightarrow{e} \langle q', v'\rangle$, if v satisfies θ and $v' = v[X := 0]$ (we should note that the set of valuations respecting θ is always in the set of valuations respecting $\mathcal{I}(q)$). Let $\delta \in R^+$. The state $\langle q, v\rangle$ has a time transition to $\langle q, v + \delta\rangle$, denoted $\langle q, v\rangle \xrightarrow{\tau} \langle q, v + \delta\rangle$, if for all $\delta' \leq \delta$, $v + \delta'$ satisfies the invariant $\mathcal{I}(q)$. We note $\mathcal{M} = (\mathcal{S}, \Rightarrow, s_0)$ the transition system of $\mathcal{A}$, where $\Rightarrow$ is either a discrete transition or a time transition and s_0 is the initial state. A run r of $\mathcal{M}$ is an infinite sequence $s_0 \Rightarrow s_1 \Rightarrow \cdots$ of states and transitions. We denote $\mathfrak{R}$ the set of runs of $\mathcal{M}$. A run is divergent if $\sum_{i=0}^{\infty} \delta_i$ (the sum of all delays δ_i on this run) diverges. We denote $\mathfrak{R}_\infty$ the set of divergent runs of $\mathcal{M}$. In the following, we will consider timed automata with only divergent runs (if the automaton has non-divergent runs, called also zeno runs, it is possible to restrict the behavior to divergent runs [10]).

From the theoretical point of view, we represent the timed model as a labeled transition system (LTS), where each discrete transition has a label a (action). If the transition is taken, the action resets the set of clocks indicated by this transition. The time transitions have a particular label named τ denoting time elapse which is considered as an internal or hidden action. Let A be the set of actions and $A_\tau = A \cup \{\tau\}$. Given a labeled transition system

$LTS = (\mathcal{S}, A_\tau, \mathcal{T}, s_0)$, $\mathcal{S}$ is the set of reachable states from s_0 with respect to $\mathcal{T}$. $\mathcal{T} \subseteq \mathcal{S} \times A_\tau \times \mathcal{S}$ the (discrete or time) transition relation and s_0 the initial state. For each label a and each state s, we consider the image set $\mathcal{T}_a(s) = \{s' \in \mathcal{S} \mid (s, a, s') \in \mathcal{T}\}$. We extend this notation for sets of states: $\mathcal{T}_a(B) = \cup\{\mathcal{T}_a(s) \mid s \in B\}$. $\mathcal{T}^{-1}$ denotes the inverse relation.

3 Reduction of TCTL Specifications

Many important properties of timed systems find a natural expression in the real-time temporal logic TCTL, which extends the branching time logic CTL [1,3]. This extension either augments temporal operators with time bounds, or uses reset quantifiers. We use a version of TCTL with time bounds. The formulas ψ of the timed computation tree logic TCTL are defined inductively by the grammar

$$\psi ::= true \mid p \mid \neg\psi \mid \psi \wedge \psi \mid \psi \exists U_{\sim c} \psi \mid \psi \forall U_{\sim c} \psi.$$

where $p \in AP$ is an atomic proposition, for simplicity, $\sim$ is restricted to be in the set $\{\leq, \geq\}$ and $c \in N$ (N is the set of natural numbers). The temporal operators $\exists\Diamond_{\sim c}\psi$ and $\forall\Box_{\sim c}\psi$ stand for $true \exists U_{\sim c}\psi$ and $\neg\exists\Diamond_{\sim c}\neg\psi$, respectively, and the temporal operators $\forall\Diamond_{\sim c}\psi$ and $\exists\Box_{\sim c}\psi$ stand for $true\forall U_{\sim c}\psi$ and $\neg\forall\Diamond_{\sim c}\neg\psi$, respectively.

The formulas of TCTL are interpreted over the set of states of a timed automaton represented by a transition system $\mathcal{M}$. Let $\langle q, v\rangle \in \mathcal{S}$ be a state reachable in $\mathcal{M}$ and let a TCTL-formula ψ. The satisfaction relation, denoted by $\langle q, v\rangle \models_{\mathcal{M}} \psi$, is defined inductively on the syntax of ψ:

- $\langle q, v\rangle \models_{\mathcal{M}} true$
- $\langle q, v\rangle \models_{\mathcal{M}} p$ iff $p \in \mathcal{L}(q)$
- $\langle q, v\rangle \models_{\mathcal{M}} \neg\psi$ iff $\langle q, v\rangle \not\models_{\mathcal{M}} \psi$
- $\langle q, v\rangle \models_{\mathcal{M}} \psi' \wedge \psi''$ iff $\langle q, v\rangle \models_{\mathcal{M}} \psi' \wedge \langle q, v\rangle \models_{\mathcal{M}} \psi''$
- $\langle q, v\rangle \models_{\mathcal{M}} \psi'\exists U_{\sim c}\psi''$ iff $\exists r \in \mathfrak{R}_\infty$ and $r(0) = \langle q, v\rangle, \exists i.\Sigma_{j\leq i}\delta_j \sim c$ and $r(i) \models \psi''$ and $\forall j < i.r(j) \models_{\mathcal{M}} \psi'$
- $\langle q, v\rangle \models_{\mathcal{M}} \psi'\forall U_{\sim c}\psi''$ iff $\forall r \in \mathfrak{R}_\infty$ and $r(0) = \langle q, v\rangle, \exists i.\Sigma_{j\leq i}\delta_j \sim c$ and $r(i) \models \psi''$ and $\forall j < i.r(j) \models_{\mathcal{M}} \psi'$

3.1 Reduction Algorithm

Our objective is to reduce a TCTL formula ψ to a CTL formula φ. Any TCTL formula ψ will introduce a new set of specification clocks $\mathcal{X}_\psi$. This set of specification clocks do not control the behavior of any system under consideration. The algorithm defined below reduces the TCTL formula ψ recursively by decomposing ψ. At the end, it generates an equivalent CTL formula φ and a timed automaton $\mathcal{A}_\psi$, capturing the timed behavior specified in the TCTL formula ψ. If the formula does not contain temporal constraint (it is already a CTL formula), the algorithm returns this formula and an empty timed automaton.

On the other hand if the TCTL formula contains temporal constraints, these can be of one of the following two forms: $\psi'\exists U_{\leq c}\psi''$ (same thing for the operator $\forall U$). The second form is $\psi'\exists U_{\geq c}\psi''$. The constructed timed automaton $\mathcal{A}_\psi$, is the same for the two forms which has a set of one clock variable $\mathcal{X}_\psi = \{z\}$, two discrete states $Q_\psi = \{q_0^\psi, q_1^\psi\}$ with invariants $z \leq c$ and *true* respectively (these two discrete states are not labeled), and one edge $\mathcal{E}_\psi = \{(q_0^\psi, z = c, \emptyset, q_1^\psi)\}$. This timed automaton will be composed with the product of the two timed automata $\mathcal{A}_{\psi'}$ and $\mathcal{A}_{\psi''}$ constructed by the recursive call to the function $Untime(\psi')$ and $Untime(\psi'')$ respectively.

The produced CTL formula is of the form $Untime(\psi')\exists U\ (q_0^\psi \in Loc) \wedge Untime(\psi'')$ for the first form of ψ, or $Untime(\psi')\exists U(q_1^\psi \in Loc) \wedge Untime(\psi'')$ for the second form. $Loc :\rightarrow \mathcal{Q}$ is an operator defined to return the current discrete state of a timed system. Thus, if a timed system is composed of n components (automata), a call to *Loc* will return as a system current state a set of the components (composing the timed system) states $\{q_1, q_2, \cdots, q_n\}$ ($q = q_1 \times q_2 \times \cdots \times q_n$, where $q \in \mathcal{Q}$ and $q_i \in \mathcal{Q}_i$).

Algorithm 1 $\{TimedAutomaton, CTL\}\ Untime(TCTL\ \psi)$

```
{
 $TimedAutomaton\ \mathcal{A}_{\psi'}, \mathcal{A}_{\psi''};$
 $CTL\ \varphi', \varphi'';$
 $TCTL \psi', \psi'';$

 $switch(\psi)\ \{$
    $case\ true\ :\ return\ \{\emptyset,\ true\};$
    $case\ p\ :\ return\ \{\emptyset,\ p\};$
    $case\ \neg\psi'\ :\ \{\mathcal{A}_{\psi'}, \varphi'\} \leftarrow Untime(\psi');\ return\ \{\mathcal{A}_{\psi'},\ \neg\varphi'\};$
    $case\ \psi' \wedge \psi''\ :\ \{\mathcal{A}_{\psi'}, \varphi'\} \leftarrow Untime(\psi');\ \{\mathcal{A}_{\psi''}, \varphi''\} \leftarrow Untime(\psi'');$
       $return\ \{\mathcal{A}_{\psi'} \oplus \mathcal{A}_{\psi''}, \varphi' \wedge \varphi''\};$
    $case\ \psi'\exists U_{\leq c(or\ \geq c)}\ (or\ \forall U_{\leq c(or\ \geq c)})\psi''\ :$
       $\{\mathcal{A}_{\psi'}, \varphi'\} \leftarrow Untime(\psi');\ \{\mathcal{A}_{\psi''}, \varphi''\} \leftarrow Untime(\psi'');$
       $Let\ z\ be\ the\ specification\ clock\ associated\ to\ this\ TCTL\ formula$
       $return\ \{(\mathcal{A}_{\psi'} \oplus \mathcal{A}_{\psi''}) \oplus \{\{q_0^\psi, q_1^\psi\}, \{z\}, \{q_0^\psi, z = c, \emptyset, q_1^\psi\},$
          $\{\mathcal{L}(q_0^\psi) = \mathcal{L}(q_1^\psi) = \emptyset\}, \{\mathcal{I}(q_0^\psi) = z \leq c, \mathcal{I}(q_1^\psi) = true\}\},$
```

```
            $\varphi' \exists U q_0^\psi \in Loc$ $(or$ $q_1^\psi \in Loc) \wedge \varphi''$
            $(or$ $\varphi' \forall U q_0^\psi \in Loc$ $(or$ $q_1^\psi \in Loc) \wedge \varphi'')\}$
 }
}
```

Example 3.1 For the TCTL formula $\psi = \forall\Diamond_{\geq 2} r$ (which is equivalent to $true \forall U_{\geq 2} r$), this algorithm generates the equivalent CTL formula φ and the timed automaton $\mathcal{A}_\psi$ shown in Figure 2.

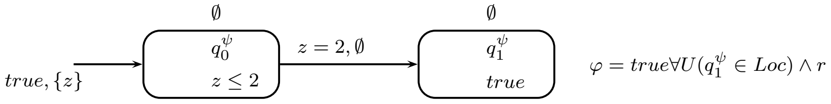


Fig. 2. Generated Timed Automaton $\mathcal{A}$ with CTL formula φ

The constructed timed automaton $\mathcal{A}_\psi$ will be composed with the original timed automaton $\mathcal{A}$ ($\mathcal{A} \otimes \mathcal{A}_\psi$). The parallel composition of two timed automata ($\otimes$) is defined as follows. Let $\mathcal{A}_i$ be $\langle \mathcal{Q}_i, \mathcal{X}_i, \mathcal{E}_i, \mathcal{L}_i, \mathcal{I}_i \rangle$, for $i = 1, 2$. We assume that $\mathcal{Q}_1 \cap \mathcal{Q}_2 = \emptyset$ and $\mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$. The product timed automaton $\mathcal{A} = \mathcal{A}_1 \otimes \mathcal{A}_2 = \langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I} \rangle$ is such that: $\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2$, $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$, $\mathcal{I}(\langle q_1, q_2 \rangle) = \mathcal{I}_1(q_1) \wedge \mathcal{I}_2(q_2)$, $\mathcal{L}(\langle q_1, q_2 \rangle) = \mathcal{L}_1(q_1) \cup \mathcal{L}_2(q_2)$, and the set $\mathcal{E}$ of edges is obtained as follows. Let $e_i \in \mathcal{E}_i$ of the form $\langle q_i, \theta_i, X_i, q_i' \rangle$, for $i = 1, 2$. Then $e \in \mathcal{E}$ can be either $e = \langle (q_1, q_2), \theta_1 \wedge \theta_2, X_1 \cup X_2, (q_1', q_2') \rangle$, $e = \langle (q_1, q_2), \theta_1, X_1, (q_1', q_2) \rangle$, or $e = \langle (q_1, q_2), \theta_2, X_2, (q_1, q_2') \rangle$.

The algorithm *Untime* is using the operator $\oplus$ to realize a parallel composition of two timed automata which is a particular case of the parallel composition $\otimes$. It is defined to compose the constructed timed automata $\mathcal{A}_{\psi'}$ and $\mathcal{A}_{\psi''}$, where ψ' and ψ'' are sub-formulas of ψ, to get only one timed automaton $\mathcal{A}_\psi$ with one clock variable z. Its difference from the operator $\otimes$ is in the construction of the set of edges $\mathcal{E}$, which is obtained as follows. Let $e_i \in \mathcal{E}_i$ of the form $\langle q_i, z_i = c_i, \emptyset, q_i' \rangle$, for $i = 1, 2$. Then, if $c_1 < c_2$, we add $e = \langle (q_1, q_2), z = c_1, \emptyset, (q_1', q_2) \rangle$ to $\mathcal{E}$, where $\mathcal{I}(\langle q_1, q_2 \rangle) = z \leq c_1$. We replace e_2 in $\mathcal{E}_2$ by $\langle q_2, z_2 = c_2 - c_1, \emptyset, q_2' \rangle$ and we remove e_1 from $\mathcal{E}_1$. Else, if $c_2 < c_1$, we add $e = \langle (q_1, q_2), z = c_2, \emptyset, (q_1, q_2') \rangle$ to $\mathcal{E}$, where $\mathcal{I}(\langle q_1, q_2 \rangle) = z \leq c_2$. We replace e_1 in $\mathcal{E}_1$ by $\langle q_1, z_1 = c_1 - c_2, \emptyset, q_1' \rangle$ and we remove e_2 from $\mathcal{E}_2$. Else, $e = \langle (q_1, q_2), z = c_1, \emptyset, (q_1', q_2') \rangle$, where $\mathcal{I}(\langle q_1, q_2 \rangle) = z \leq c_1$ and we remove e_1 from $\mathcal{E}_1$, e_2 from $\mathcal{E}_2$.

This process will continue until $\mathcal{E}_i = \emptyset$, for $i = 1, 2$ or one of the following two cases is satisfied. In the case where $\mathcal{E}_1 = \emptyset$ and $\mathcal{E}_2 = \{\langle q_2, z_2 = c_2, \emptyset, q_2' \rangle\}$, we assume that $q_1 \in \mathcal{Q}_1$ is the discrete state without outgoing edge. Then, we add the edge $e = \langle (q_1, q_2), z = c_2, \emptyset, (q_1, q_2') \rangle$ to $\mathcal{E}$, where $\mathcal{I}(\langle q_1, q_2 \rangle) = z \leq c_2$. In the other case where $\mathcal{E}_2 = \emptyset$ and $\mathcal{E}_1 = \{\langle q_1, z_1 = c_1, \emptyset, q_1' \rangle\}$, we assume that

$q_2 \in \mathcal{Q}_2$ is the discrete state without outgoing edge. Then, we add the edge $e = \langle (q_1, q_2), z = c_1, \emptyset, (q_1', q_2) \rangle$ to $\mathcal{E}$, where $\mathcal{I}(\langle q_1, q_2 \rangle) = z \leq c_1$. At the end, if there are discrete states in the produced timed automaton, without ingoing and outgoing edges, they will be removed from the set of discrete states $\mathcal{Q}$.

$\otimes$-Property 1: $\mathcal{A} \otimes \emptyset = \emptyset \otimes \mathcal{A} = \mathcal{A}$, $\emptyset$ (empty automaton is an automaton with no states and no transitions) is the identity. We note that $\emptyset$ is also an identity for $\oplus$

$\otimes$-Property 2: If the valuation functions v_1 and v_2 are defined as follows: $v_1 : \mathcal{X}_1 \rightarrow R^+$ and $v_2 : \mathcal{X}_2 \rightarrow R^+$, respectively. Then the valuation function $v : \mathcal{X} \rightarrow R^+$, where $\mathcal{X} = \mathcal{X}_1 \cup \mathcal{X}_2$, is defined as follows. If $x \in \mathcal{X}_1$ then $v(x) = v_1(x)$ else $v(x) = v_2(x)$.

$\otimes$-Property 3: If a TCTL-formula ψ is satisfied in a model $\mathcal{M}$ of the timed automaton $\mathcal{A}$, it is still satisfied in a model $\mathcal{M}^+$ of the timed automaton $\mathcal{A}^+$ which is the composition of $\mathcal{A}$ with any other timed automaton $\mathcal{A}'$ respecting the conditions mentioned above.

Proof. The proof proceeds by induction on the structure of ψ. The basis cases where ψ is of the form *true* or p are immediate. For example, if $\psi = p$, this formula is satisfied in the state $\langle q, v \rangle$ if and only if $p \in \mathcal{L}(q)$. By composition with a new timed automaton $\mathcal{A}'$, the state will have the form $\langle (q, q'), v \rangle$, where q' is one of the locations in $\mathcal{A}'$ and the set of atomic propositions marking this state is $\mathcal{L}(q) \cup \mathcal{L}'(q')$ which contains p.

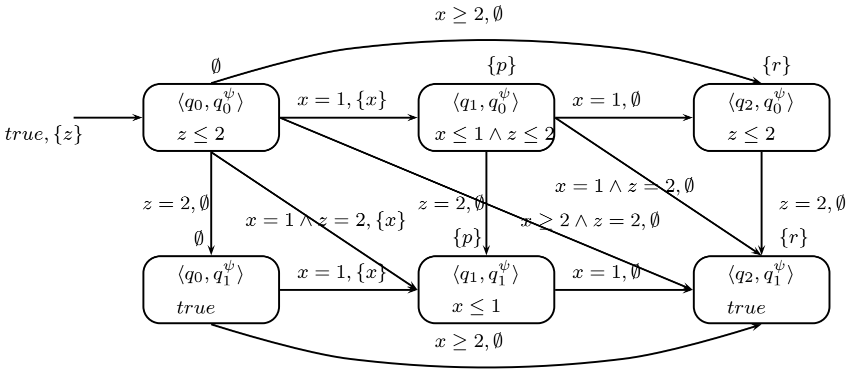
For $\psi = \neg\psi'$. By the semantics of TCTL, $\langle q, v \rangle \models_{\mathcal{M}} \neg\psi'$ if and only if $not(\langle q, v \rangle \models_{\mathcal{M}} \psi')$. The induction hypothesis is $\langle q, v \rangle \models_{\mathcal{M}} \psi' \Rightarrow \langle\langle (q, q'), v \rangle \models_{\mathcal{M}^+} \psi'$. Then, $\langle q, v \rangle \models_{\mathcal{M}} \psi' \Rightarrow not(\langle (q, q'), v \rangle \models_{\mathcal{M}^+} \neg\psi')$. By the semantics of TCTL $\langle (q, q'), v \rangle \models_{\mathcal{M}^+} \neg\psi')$.

For $\psi = \psi' \wedge \psi''$. By the semantics of TCTL, $\langle q, v \rangle \models_{\mathcal{M}} \psi' \wedge \psi''$ if and only if $\langle q, v \rangle \models_{\mathcal{M}} \psi'$ and $\langle q, v \rangle \models_{\mathcal{M}} \psi''$. The induction hypothesis is $\langle q, v \rangle \models_{\mathcal{M}} \psi' \Rightarrow \langle (q, q'), v \rangle \models_{\mathcal{M}^+} \psi'$ and $\langle q, v \rangle \models_{\mathcal{M}} \psi'' \Rightarrow \langle (q, q'), v \rangle \models_{\mathcal{M}^+} \psi''$. By the semantics of TCTL $\langle (q, q'), v \rangle \models_{\mathcal{M}^+} \psi' \wedge \psi''$.

For $\psi = \psi' \exists U_{\sim c} \psi''$. By the semantics of TCTL, $\exists r \in \mathfrak{R}_\infty$ where $r = \langle q_0, v_0 \rangle, \langle q_1, v_1 \rangle, \cdots, \langle q_j, v_j \rangle, \cdots, \langle q_i, v_i \rangle, \cdots$, such that $\Sigma_{j \leq i} \delta_j \sim c$ and $\langle q_i, v_i \rangle \models_{\mathcal{M}} \psi''$ and $\forall j < i. \langle q_j, v_j \rangle \models_{\mathcal{M}} \psi'$. By composition with other timed automaton, the run r will be $r' = \langle (q_0, q_0'), v_0 \rangle, \cdots, \langle (q_1, q_k'), v_1 \rangle, \cdots, \langle (q_j, q_m'), v_j \rangle, \cdots, \langle (q_i, q_p'), v_i \rangle, \cdots$. The induction hypothesis is $\langle q_i, v_i \rangle \models_{\mathcal{M}} \psi'' \Rightarrow \langle (q_i, q_p'), v_i \rangle \models_{\mathcal{M}^+} \psi''$ and $\langle q_j, v_j \rangle \models_{\mathcal{M}} \psi' \Rightarrow \langle (q_j, q_m'), v_j \rangle \models_{\mathcal{M}^+} \psi'$. By the semantics of TCTL, $\langle (q_0, q_0'), v_0 \rangle \models_{\mathcal{M}^+} \psi$.

By the same way, we prove the case $\psi = \psi' \forall U_{\sim c} \psi''$. Every run $r = \langle q_0, v_0 \rangle, \cdots, \langle q_j, v_j \rangle, \cdots, \langle q_i, v_i \rangle, \cdots \in \mathfrak{R}_\infty$, after composition with a new timed automaton, will have the form $r' = \langle(q_0, q'_0), v_0\rangle, \cdots, \langle(q_0, q'_k), v_k\rangle, \cdots, \langle(q_j, q'_0), v_m\rangle, \cdots, \langle(q_j, q'_p), v_j\rangle, \cdots, \langle(q_i, q'_m), v_n\rangle, \cdots, \langle(q_i, q'_l), v_i\rangle, \cdots$. The induction hypothesis is $\langle q_i, v_i\rangle \models_{\mathcal{M}} \psi'' \Rightarrow \langle(q_i, q'_l), v_i\rangle \models_{\mathcal{M}^+} \psi''$ and $\langle q_j, v_j\rangle \models_{\mathcal{M}} \psi' \Rightarrow \langle(q_j, q'_p), v_j\rangle \models_{\mathcal{M}^+} \psi'$. By the semantics of TCTL, $\langle(q_0, q'_0), v_0\rangle \models_{\mathcal{M}^+} \psi'\forall U_{\sim c}\psi''$. □

Example 3.2 The product timed automaton of the two timed automata shown in Figure 1 and Figure 2 is presented in Figure 3.



$x \geq 2, \emptyset$
$\emptyset$
$\{p\}$
$\{r\}$
$\langle q_0, q_0^\psi\rangle$
$z \leq 2$
$x = 1, \{x\}$
$\langle q_1, q_0^\psi\rangle$
$x \leq 1 \wedge z \leq 2$
$x = 1, \emptyset$
$\langle q_2, q_0^\psi\rangle$
$z \leq 2$
$true, \{z\}$
$x = 1 \wedge z = 2, \emptyset$
$z = 2, \emptyset$
$x = 1 \wedge z = 2, \{x\}$
$z = 2, \emptyset$
$x \geq 2 \wedge z = 2, \emptyset$
$z = 2, \emptyset$
$\emptyset$
$\{p\}$
$\{r\}$
$\langle q_0, q_1^\psi\rangle$
$true$
$x = 1, \{x\}$
$\langle q_1, q_1^\psi\rangle$
$x \leq 1$
$x = 1, \emptyset$
$\langle q_2, q_1^\psi\rangle$
$true$
$x \geq 2, \emptyset$

Fig. 3. The product timed automaton

3.2 Correctness

By the reduction algorithm *Untime*, we produce a CTL formula φ and a timed automaton $\mathcal{A}_\psi$ from a TCTL formula ψ. Let $\mathcal{M}$ be the transition system of the timed automaton $\mathcal{A}$ modeling a real-time system. Along the proof, we use $\langle q, v\rangle \models_{\mathcal{M}} \psi$ to indicate that the state $\langle q, v\rangle$ satisfies the TCTL formula ψ. We use $\langle q, v\rangle \models_{\mathcal{M}^+} \varphi$ to indicate that the state $\langle q, v\rangle$ satisfies the CTL formula φ ($\mathcal{M}^+$ is the transition system of $\mathcal{A}^+ = \mathcal{A} \otimes \mathcal{A}_\psi$).

Theorem 3.3 $\langle q, v\rangle \models_{\mathcal{M}} \psi \Leftrightarrow \langle q, v\rangle \models_{\mathcal{M}^+} \varphi$

Proof. The proof proceeds by induction on the structure of ψ. The basis cases where ψ is of the form *true* or p are immediate. In these basis cases $\mathcal{A} = \emptyset$; and $\psi = \varphi$, using $\otimes$-Property 1, this means $\langle q, v\rangle \models_{\mathcal{M}} \psi \Leftrightarrow \langle q, v\rangle \models_{\mathcal{M}^+} \varphi$.

For $\psi = \neg\psi'$, we prove $\langle q, v\rangle \models_{\mathcal{M}} \psi \Leftrightarrow \langle q, v\rangle \models_{\mathcal{M}^+} \varphi$, where $\mathcal{M}^+$ and φ are constructed by the call to *Untime*(ψ). The induction hypothesis is $\langle q, v\rangle \models_{\mathcal{M}} \psi' \Leftrightarrow \langle q, v\rangle \models_{\mathcal{M}^+_{\psi'}} \varphi'$, where $\mathcal{M}^+_{\psi'}$ and φ' are constructed by *Untime*(ψ'). According to the algorithm *Untime*, $\varphi = \neg\varphi'$ and $\mathcal{M}^+ = \mathcal{M}^+_{\psi'}$. By the semantics of TCTL, $\langle q, v\rangle \models_{\mathcal{M}} \neg\psi'$ if and only if not ($\langle q, v\rangle \models_{\mathcal{M}} \psi'$). By the induction hypothesis, $\langle q, v\rangle \models_{\mathcal{M}} \neg\psi'$ if and only if not ($\langle q, v\rangle \models_{\mathcal{M}^+_{\psi'}} \varphi'$).

Now, by the TCTL semantics, we can conclude $\langle q,v\rangle \models_{\mathcal{M}} \neg\psi'$ if and only if $\langle q,v\rangle \models_{\mathcal{M}^+_{\psi'}} \neg\varphi'$. This means that $\langle q,v\rangle \models_{\mathcal{M}} \psi$ if and only if $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi$.

For $\psi = \psi' \wedge \psi''$. First we prove $\langle q,v\rangle \models_{\mathcal{M}} \psi \Rightarrow \langle q,v\rangle \models_{\mathcal{M}^+} \varphi$. By the semantics of TCTL, $\langle q,v\rangle \models_{\mathcal{M}} \psi' \wedge \psi''$ if and only if $\langle q,v\rangle \models_{\mathcal{M}} \psi'$ and $\langle q,v\rangle \models_{\mathcal{M}} \psi''$. The induction hypothesis is $\langle q,v\rangle \models_{\mathcal{M}} \psi' \Leftrightarrow \langle q,v\rangle \models_{\mathcal{M}^+_{\psi'}} \varphi'$ and $\langle q,v\rangle \models_{\mathcal{M}} \psi'' \Leftrightarrow \langle q,v\rangle \models_{\mathcal{M}^+_{\psi''}} \varphi''$, where φ' (with $\mathcal{A}_{\psi'}$) and φ'' (with $\mathcal{A}_{\psi''}$) are the CTL formulas (timed automata) constructed from the TCTL sub-formulas ψ' and ψ'' respectively, $\mathcal{M}^+_{\psi'}$ and $\mathcal{M}^+_{\psi''}$ are the transition systems of $\mathcal{A}\otimes\mathcal{A}_{\psi'}$ and $\mathcal{A}\otimes\mathcal{A}_{\psi''}$, respectively. By $\otimes$-Property 3 of automata composition, we have $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi'$ and $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi''$. Then, we can conclude by using the semantics of TCTL $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi' \wedge \varphi''$ ($\mathcal{M}^+$ is the transition system of $\mathcal{A} \otimes \mathcal{A}_{\psi'} \otimes \mathcal{A}_{\psi''}$).

Now, we proof $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi \Rightarrow \langle q,v\rangle \models_{\mathcal{M}} \psi$. We know that φ is constructed from ψ and as ψ is of the form $\psi' \wedge \psi''$, then according to the algorithm *Untime*, φ is of the form $\varphi' \wedge \varphi''$. The sub-formula φ' and the automaton $\mathcal{A}_{\psi'}$ are both constructed from the sub-formula ψ' (the same for φ'' and $\mathcal{A}_{\psi''}$ are constructed from ψ''). By the induction hypothesis ($\langle q,v\rangle \models_{\mathcal{M}^+_{\psi'}} \varphi' \Leftrightarrow \langle q,v\rangle \models_{\mathcal{M}} \psi'$ and $\langle q,v\rangle \models_{\mathcal{M}^+_{\psi''}} \varphi'' \Leftrightarrow \langle q,v\rangle \models_{\mathcal{M}} \psi''$), using the composition property $\otimes$-Property 3 and the semantics of TCTL, we can conclude that $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi' \wedge \varphi'' \Rightarrow \langle q,v\rangle \models_{\mathcal{M}} \psi' \wedge \psi''$.

For $\psi = \psi'\exists U_{\leq c}\psi''$. Consider a state $\langle q,v\rangle$ in $\mathcal{M}$. Assume that $\langle q,v\rangle \models_{\mathcal{M}} \psi$. Then, by the semantics of TCTL, there is a run $r = \langle q_0,v_0\rangle, \langle q_1,v_1\rangle, \cdots, \langle q_i,v_i\rangle, \cdots \in \mathfrak{R}_\infty$ with $\langle q_0,v_0\rangle = \langle q,v\rangle$, where $i \geq 0$ such that $\Sigma_{k=0}^{i}\delta_k \leq c$ and $\langle q_i,v_i\rangle \models_{\mathcal{M}} \psi''$, and for all $0 \leq j < i$ we have $\langle q_j,v_j\rangle \models_{\mathcal{M}} \psi'$. By the induction hypothesis, $\langle q_i,v_i\rangle \models_{\mathcal{M}} \psi'' \Leftrightarrow \langle (q_i,q_{\psi''}),v_i\rangle \models_{\mathcal{M}^+_{\psi''}} \varphi''$ ($\mathcal{M}^+_{\psi''}$ is the model of $\mathcal{A} \otimes \mathcal{A}_{\psi''}$) and $\langle q_j,v_j\rangle \models_{\mathcal{M}} \psi' \Leftrightarrow \langle (q_j,q_{\psi'}),v_j\rangle \models_{\mathcal{M}^+_{\psi'}} \varphi'$ ($\mathcal{M}^+_{\psi'}$ is the model of $\mathcal{A} \otimes \mathcal{A}_{\psi'}$). It is clear from the composition property $\otimes$-Property 3 that $\langle (q_i,q_{\psi'},q_{\psi''}),v_i\rangle \models_{\mathcal{M}^+_{\psi'\psi''}} \varphi''$ and $\langle (q_j,q_{\psi'},q_{\psi''}),v_j\rangle \models_{\mathcal{M}^+_{\psi'\psi''}} \varphi'$, where $\mathcal{M}^+_{\psi'\psi''}$ is the model of $\mathcal{A} \otimes (\mathcal{A}_{\psi'} \oplus \mathcal{A}_{\psi''})$. If we add a specification clock z to the set of clocks and if we compose the automaton $\mathcal{A}\otimes(\mathcal{A}_{\psi'}\oplus\mathcal{A}_{\psi''})$ with a new automaton $\mathcal{A}_\psi$ composed of two states $\{q_0^\psi, q_1^\psi\}$ with $\mathcal{I}(q_0^\psi) = z \leq c$ and edge $e^\psi = (q_0^\psi, z = c, \emptyset, q_1^\psi)$. It is clear that $v_i(z) \leq c$ and $\langle (q_i,q_{\psi'},q_{\psi''},q_0^\psi),v_i\rangle \models_{\mathcal{M}^+} \varphi''$ and for all $0 \leq j < i$ we have $\langle (q_j,q_{\psi'},q_{\psi''},q_0^\psi),v_j\rangle \models_{\mathcal{M}^+} \varphi'$, where $\mathcal{M}^+$ is the model of $\mathcal{A} \otimes (\mathcal{A}_{\psi'} \oplus \mathcal{A}_{\psi''} \oplus \mathcal{A}_\psi)$. Thus, $v_i(z) \leq c$ and $\langle (q_i,q_{\psi'},q_{\psi''},q_0^\psi),v_i\rangle \models_{\mathcal{M}^+} \varphi''$ if and only if $\langle q_i,v_i\rangle \models_{\mathcal{M}^+} \varphi'' \wedge q_0^\psi \in Loc$. By the semantics of TCTL, we have $\langle q,v\rangle \models_{\mathcal{M}^+} \varphi'\exists U\varphi'' \wedge q_0^\psi \in Loc$. We proceed by the same way for $\psi = \psi'\forall U_{\leq c}\psi''$.

By the same way we can prove the case $\psi = \psi'\exists U_{\geq c}\psi''$, where $\langle (q_j,q_{\psi'},q_{\psi''},q_1^\psi),$

$v_j\rangle \models_{\mathcal{M}^+} \varphi'$. Thus, $v_i(z) \geq c$ and $\langle(q_i, q_{\psi'}, q_{\psi''}, q_1^{\psi}), v_i\rangle \models_{\mathcal{M}^+} \varphi''$ if and only if $\langle q_i, v_i\rangle \models_{\mathcal{M}^+} \varphi'' \wedge q_1^{\psi} \in Loc$. By the semantics of TCTL, we have $\langle q, v\rangle \models_{\mathcal{M}^+} \varphi' \exists U \varphi'' \wedge q_1^{\psi} \in Loc$. We proceed by the same way for $\psi = \psi' \forall U_{\geq c} \psi''$. □

We denote the size of a timed automaton $\mathcal{A} = \langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I}\rangle$ by the pair $(|\mathcal{Q}|, |\mathcal{E}|)$, where $|\mathcal{Q}|$ is the number of discrete states and $|\mathcal{E}|$ is the number of edges. For a TCTL formula ψ with n temporal constraints, the algorithm *Untime* generates a timed automaton $\mathcal{A}_\psi$ with one clock variable, $|\mathcal{Q}_\psi| \leq n+1$ and $|\mathcal{E}_\psi| \leq n$. The size of $\mathcal{A} \otimes \mathcal{A}_\psi$ is at most $(|\mathcal{Q}| \times |\mathcal{Q}_\psi|, |\mathcal{E}| \times |\mathcal{E}_\psi| + |\mathcal{E}| + |\mathcal{E}_\psi| - 1)$.

4 Generating Bi-similar Finite System

The model of a timed automaton is an infinite transition-state system due to dense time. Then, it is not possible to perform a model checking. In this section we present our algorithm that generates a strongly bi-similar finite system based on a defined equivalence where exact delays are abstracted away while information on the discrete changes of the system is retained.

4.1 Strong Bi-simulation

For a labeled transition system $\mathcal{M} = (\mathcal{S}, A_\tau, \mathcal{T}, s_0)$, a partition $\wp$ (or equivalence relation on $\mathcal{S}$) of the elements of $\mathcal{S}$ is a set of disjoint blocks $\{B_i \mid i \in N\}$ such that $\cup_{i \in N} B_i = \mathcal{S}$. Let $\wp$ and $\wp'$ be partitions of $\mathcal{S}$. $\wp'$ is a refinement of $\wp$ ($\wp' \subseteq \wp$) if and only if $\forall B' \in \wp' : \exists B \in \wp : (B' \subseteq B)$. Intuitively, two states s_1 and s_2 are bi-similar if for each state s_1' reachable from s_1 by execution of an action $a \in A_\tau$ (see Section 2) there is a state s_2', reachable from s_2 by execution of the action a such that s_1' and s_2' are bi-similar.

Definition 4.1 Given a labeled transition system $\mathcal{M} = (\mathcal{S}, A_\tau, \mathcal{T}, s_0)$, a binary relation $\wp \subseteq \mathcal{S} \times \mathcal{S}$ is a strong bi-simulation if and only if the following conditions hold $\forall (s_1, s_2) \in \wp$ and $\forall a \in A_\tau$:

(i) $\mathcal{L}(s_1) = \mathcal{L}(s_2)$,

(ii) $\forall s_3 (s_3 = \mathcal{T}_a(s_1) \Rightarrow \exists s_4 (s_4 = \mathcal{T}_a(s_2) \wedge (s_3, s_4) \in \wp))$ and

(iii) $\forall s_4 (s_4 = \mathcal{T}_a(s_2) \Rightarrow \exists s_3 (s_3 = \mathcal{T}_a(s_1) \wedge (s_3, s_4) \in \wp))$.

The set of bi-simulations on $\mathcal{S}$, ordered by inclusion has a minimal element which is the identity relation denoted by $\wp_0$ and it has a maximal element denoted by $\wp_{max}$ which is an equivalence relation on (or a partition of) $\mathcal{S}$. We will be interested in the maximal element which induces the smallest number of equivalence classes in terms of relation inclusion. $\wp_{max}$ (which is unique) may be obtained as the limit of a decreasing sequence of relations $\wp_i$.

Most algorithms used to solve the bi-simulation problem are based on some form of partition refinement, i.e. they perform successive iterations in which blocks of the current partition are split into smaller blocks, until no block can be split anymore. While splitting a block, states that cannot be distinguished are kept in the same block. Two states can be distinguished if one of the states allows a transition with a certain label to a state in a certain block and the other state does not have a transition with the same label to a state in the same block.

Let $\wp$ be a partition of $\mathcal{S}$. $\wp$ is compatible with $\mathcal{T}$ (it is called also stable) if and only if the following property $\mathcal{P}$ holds:

$$P(\wp) = \forall a \in A_\tau : \forall B, B' \in \wp : (B' \subseteq \mathcal{T}_a^{-1}(B) \vee B' \cap T_a^{-1}(B) = \emptyset).$$

Correctness of a partition refinement algorithm follows from two facts. First, a stable partition is a bi-simulation relation (states are equivalent if they are in the same block). Second, each computed partition by the refinement of the previous one, respects the property $\mathcal{P}$.

Definition 4.2 Let $\mathcal{M} = (\mathcal{S}, A_\tau, \mathcal{T}, s_0)$ be a labeled transition system and $\wp$ an equivalence relation which is a strong bi-simulation, the quotient labeled transition system denoted by $\mathcal{M}/\wp$ is defined as follows: $\mathcal{M}/\wp = (\mathcal{S}/\wp, A_\tau, \mathcal{T}/\wp, s_0)$ where:

- $\mathcal{S}/\wp$ is the set of equivalence classes noted $\mathcal{C}$, $\mathcal{C} = \{B \subseteq \mathcal{S} \mid \forall s_1, s_2 \in B : (s_1, s_2) \in \wp)$
- $((B = \mathcal{T}_a(B')) \in \mathcal{T}/\wp)$ if and only if $\mathcal{T}_a^{-1}(B) \cap B' \neq \emptyset$
- $\mathcal{C}_0 = [s_0]$ is the equivalence class of s_0.

$\mathcal{M}/\wp_{max}$ is the normal form of $\mathcal{M}$ with respect to $\wp_{max}$. We present below a partition-refinement algorithm based on strong bi-simulation. We start from an initial partition of the state space in zones. Each time a zone Z is to be refined, it is split with respect to all its discrete successors by some edge e. We can prove that if all successors are zones, then the result of the split is also a set of zones, that is, convexity is preserved by the split operation.

4.2 Partition-Refinement Algorithm

Consider a timed automaton $\mathcal{A} = \langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I} \rangle$ and let $e \in \mathcal{E}$ be an edge such that its guard is θ different from *true*. We will refine the block of source states (q, v) of e represented as a convex zone $Z = (q, \mathcal{V}_Z)$. The objective of refinement is to abstract the quantitative aspect of time needed to measure the constraint θ. So, this block of states (zone) is refined into sub-zones. The invariant of one of these sub-zones satisfies the constraint θ. But, the invariants of the other sub-zones don't satisfy this constraint. This process of refinement will continue until there are no blocks to refine. The operators over temporal constraints, used in the algorithm of partition refinement are defined as follows.

(i) $Var(\theta)$ is the set of clock variables in the formula θ.

(ii) $\theta\rfloor_x$ is the constraint θ reduced to a constraint defined only on the clock variable x (e.g. $x = 1 \wedge y < 2 \wedge z \leq 3\rfloor_x \equiv x = 1$)

(iii) $\theta\rceil^x$ is the constraint θ reduced to a constraint defined without the clock variable x (e.g. $x = 1 \wedge y < 2 \wedge z \leq 3\rceil^x \equiv y < 2 \wedge z \leq 3$)

(iv) $\cap$ is the intersection operator (e.g. $x \leq 2 \cap x \geq 2 \equiv x = 2$). $\theta_1 \cap \theta_2 = \emptyset$ if $Var(\theta_1) \cap Var(\theta_2) = \emptyset$.

(v) $\setminus$ is the set difference operator (e.g. $x \leq 2 \setminus x = 1 \equiv x < 1 \vee (x > 1 \wedge x \leq 2)$, it is not convex).

(vi) $\lfloor\theta\rfloor$ if θ is convex then this operator will return θ itself else it returns the constraint representing the lower convex valuations. The constraint θ is defined on one clock variable (e.g $\lfloor x < 1 \vee (x > 1 \wedge x \leq 2)\rfloor \equiv x < 1$).

(vii) $\lceil\theta\rceil$ if θ is convex then this operator will return $\emptyset$ else it returns the constraint representing the upper convex valuations. The constraint θ is defined on one clock variable (e.g $\lceil x < 1 \vee (x > 1 \wedge x \leq 2)\rceil \equiv x > 1 \wedge x \leq 2$).

The algorithm below refines a zone that is a source of an edge $e = \langle q, \theta, X, q'\rangle$, taken arbitrarily from the set $\mathcal{E}$ of the current partition, where $\theta \neq true$. The refinement is based on a clock variable x taken also arbitrarily from the set of clock variables in the constraint θ. The zone is split into at most three sub-zones. These sub-zones have the same location q, but with different invariants. Their union equals to $\mathcal{I}(q)$. Because their invariants are different and for algorithm simplicity, we will denote their location q differently to distinguish them, this will not have any effect on the algorithm results.

The first sub-zone (with discrete state q_x) has the invariant $\mathcal{I}(q_x) = \theta\rfloor_x \wedge \mathcal{I}(q)\rceil^x$ and an outgoing edge $\langle q_x, \theta\rceil^x, \emptyset, q'\rangle$.

If $\lfloor\mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x\rfloor \neq \emptyset$, we have a second sub-zone with a discrete state q_l with an invariant $\mathcal{I}(q_l) = \lfloor\mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x\rfloor \wedge \mathcal{I}(q)\rceil^x$. This sub-zone has an outgoing edge $\langle q_l, true, \emptyset, q_x\rangle$. If $\lceil\mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x\rceil \neq \emptyset$ then, we have a third sub-zone with a discrete state q_u and an invariant

$\mathcal{I}(q_u) = \lceil\mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x\rceil \wedge \mathcal{I}(q)\rceil^x$. This sub-zone has an ingoing edge $\langle q_x, true, \emptyset, q_u\rangle$. The three new sub-zones will be marked by the same set of atomic propositions $\mathcal{L}(q)$.

At the end of this iteration, the edge e and the zone Z will be removed and replaced by the new edges and the new sub-zones. The other outgoing and incoming edges from and to the zone Z will be updated according to the new partition, see the algorithm for more details. The non-zenoness of the timed automaton and the convexity of its constraints guarantee that the produced partition has zones preserving the convexity and the non-zenoness. Moreover, the algorithm terminates.

Algorithm 2 *partition-refinement($\mathcal{A} = \langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I} \rangle$)*

```
{
 if ($\exists e \in \mathcal{E} \mid e = \langle q, \theta, X, q' \rangle \wedge \theta \neq true$) {
   Let $x$ be a clock variable in the constraint $\theta$.
   $\mathcal{E}_{Old} \leftarrow \{e\}$;
   $C_q \leftarrow q_x$ with $\mathcal{I}(q_x) = \theta\rfloor_x \wedge \mathcal{I}(q)\rceil^x$ and $\mathcal{L}(q_x) = \mathcal{L}(q)$;
   $\mathcal{E}_{New} \leftarrow \{\langle q_x, \theta\rceil^x, \emptyset, q' \rangle\}$;
   if ($\lfloor \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rfloor \neq \emptyset$) {
     $C_q \leftarrow C_q \cup \{q_l\}$ with $\mathcal{I}(q_l) = \lfloor \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rfloor \wedge \mathcal{I}(q)\rceil^x$ and $\mathcal{L}(q_l) = \mathcal{L}(q)$;
     $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q_l, true, \emptyset, q_x \rangle\}$
   }
   if ($\lceil \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rceil \neq \emptyset$) {
     $C_q \leftarrow C_q \cup \{q_u\}$ with $\mathcal{I}(q_u) = \lceil \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rceil \wedge \mathcal{I}(q)\rceil^x$ and $\mathcal{L}(q_u) = \mathcal{L}(q)$;
     $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q_x, true, \emptyset, q_u \rangle\}$
   }
   for each edge $e' = \langle q, \theta', X', q' \rangle \in \mathcal{E}$ do //Outgoing edges
     $\mathcal{E}_{Old} \leftarrow \mathcal{E}_{Old} \cup \{e'\}$
     if ($\lfloor \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rfloor \neq \emptyset \wedge \theta' \cap \mathcal{I}(q_l) \neq \emptyset$) $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q_l, \theta' \cap \mathcal{I}(q_l), X', q' \rangle\}$
     if ($\theta' \cap \mathcal{I}(q_x) \neq \emptyset$) $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q_x, \theta' \cap \mathcal{I}(q_x), X', q' \rangle\}$
     if ($\lceil \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rceil \neq \emptyset \wedge \theta' \cap \mathcal{I}(q_u) \neq \emptyset$) $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q_u, \theta' \cap \mathcal{I}(q_u), X', q' \rangle\}$
   end for
   for each edge $e'' = \langle q'', \theta'', X'', q \rangle \in \mathcal{E}$ do //Incoming edges
     $\mathcal{E}_{Old} \leftarrow \mathcal{E}_{Old} \cup \{e''\}$
     if ($\lfloor \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rfloor \neq \emptyset \wedge (\mathcal{I}(q_l) \cap \theta'' \neq \emptyset \vee Var(\mathcal{I}(q_l)) \cap Var(X'') \neq \emptyset)$)
       $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q'', \theta'', X'', q_l \rangle\}$
     else
       if ($\lceil \mathcal{I}(q)\rfloor_x \setminus \theta\rfloor_x \rceil = \emptyset \vee \theta'' \cap \mathcal{I}(q_x) \neq \emptyset \vee Var(\mathcal{I}(q_x)) \cap Var(X'') \neq \emptyset$)
         $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q'', \theta'', X'', q_x \rangle\}$
       else
         $\mathcal{E}_{New} \leftarrow \mathcal{E}_{New} \cup \{\langle q'', \theta'', X'', q_u \rangle\}$
   end for
   return partition-refinement($\langle \mathcal{Q} \setminus \{q\} \cup C_q, \mathcal{X}, (\mathcal{E} \setminus \mathcal{E}_{Old}) \cup \mathcal{E}_{New}, \mathcal{L}, \mathcal{I} \rangle$)
 } else {
   for each $q \in \mathcal{Q} \mid \mathcal{I}(q)$ is bounded only from below or $\nexists e = \langle q, true, \emptyset, q' \rangle$
     $\mathcal{E} \leftarrow \mathcal{E} \cup \{\langle q, true, \emptyset, q \rangle\}$
   return $\langle \mathcal{Q}, \mathcal{X}, \mathcal{E}, \mathcal{L}, \mathcal{I} \rangle$
 }
}
```

4.3 Quotient Graph

The partition-refinement algorithm generates a stable partition $\wp_{max}$ which is the coarsest. Each block in this partition is characterized by an invariant and a unique discrete state. These blocks are reachable and their invariants are convex. The edges of this partition, are of the form $\langle q, true, \emptyset, q' \rangle$. This partition can be easily represented by a graph, we call it the quotient graph

$G_{\wp_{max}}$. The set $\mathcal{C}$ of nodes of $G_{\wp_{max}}$ is the set of the partition blocks. Thus, a node corresponding to block B_i is denoted C_i. The edges of $G_{\wp_{max}}$ are the edges in the partition $\wp_{max}$ between the different blocks in addition to edges of the form $\langle q, true, \emptyset, q\rangle$ if the invariant $\mathcal{I}(q)$ is bounded only from below. We redefine the function $\mathcal{L}$ over $\mathcal{C}$ as follows. If the effective discrete state (it is the discrete state of the partitioned original block in the initial partition) of block B_i is q then, $\mathcal{L}(C_i) = \mathcal{L}(q)$. Then, different nodes of this graph can refer to one location from $\mathcal{Q}$. It is necessary to keep this trace for verification purpose by using the operator *Loc* defined in Section 3. For example, if the current state of the transition system is the node C_1 and C_1 has inherited the effective discrete state q_1, then $Loc = \{q_1\}$. The strong bi-simulation quotient graph ($G_{\wp_{max}}$) generated by the algorithm of partition refinement and as it is defined, has the following properties:

$G_{\wp_{max}}$**-Property 1**: $G_{\wp_{max}}$ is stable which means that $\forall C_1, C_2 \in G_{\wp_{max}}$, then by definition, if $C_1 \xrightarrow{\tau} C_2$ then $\forall s_1 \in C_1$ there exists $s_2 \in C_2$, such that $s_1 \xrightarrow{\delta} s_2$, for some $\delta \in R^+$ and if $C_1 \xrightarrow{e} C_2$, for some edge e, then $\forall s_1 \in C_1$ there exists $s_2 \in C_2$, such that $s_1 \xrightarrow{e} s_2$.

$G_{\wp_{max}}$**-Property 2**: Given a path $\rho = C_1 \Rightarrow C_2 \Rightarrow \cdots$ of $G_{\wp_{max}}$ and a run $r = s_1 \Rightarrow s_2 \Rightarrow \cdots$, we say that r is inscribed in ρ if for all $i \geq 1 : s_i \in C_i$ and, if $C_i \xrightarrow{\tau} C_{i+1}$ then there exists $\delta > 0$ such that $s_i \xrightarrow{\delta} s_{i+1}$, if $C_i \xrightarrow{e} C_{i+1}$ then $s_i \xrightarrow{e} s_{i+1}$. It is easy to conclude that every run r is inscribed in a unique path ρ in $G_{\wp_{max}}$. And inversely, if $\rho = C_1 \Rightarrow C_2 \Rightarrow \cdots$ is a path in $G_{\wp_{max}}$ then for all $s_1 \in C_1$ there exists a run r starting from s_1 and inscribed in ρ.

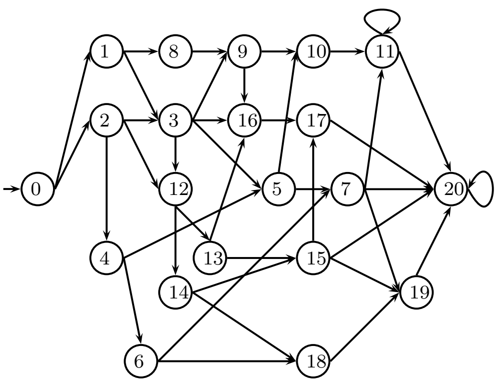
$G_{\wp_{max}}$**-Property 3**: Any time transition traverses a unique (finite) set of classes. Also, if $(s, s') \in \wp_{max}$ then for any time transition $s \xrightarrow{\delta} s + \delta$, there exists a time transition $s' \xrightarrow{\delta'} s' + \delta'$ such that $(s + \delta, s' + \delta') \in \wp_{max}$ and the two transitions traverse the same classes.

The size of the quotient graph $G_{\wp_{max}}$ is defined by the pair of the number of its nodes and number of its edges, which are at most $(3 \times |\mathcal{E}|, |\mathcal{Q}| + 3 \times |\mathcal{E}|)$.

Example 4.3 The strong bi-simulation quotient graph of the system described in Example 2 is shown in Figure 4, whose nodes are the symbolic states (zones) shown on the right with detailed information.

5 CTL Model-Checking

In this section we show that the strong bi-simulation $\wp_{max}$ preserves the CTL properties. The timed automaton model-checking can be reduced to model-



$\langle 0, q_0 \times q_0^{\psi}, x < 1 \wedge z < 2, \emptyset \rangle$
$\langle 1, q_0 \times q_0^{\psi}, x < 1 \wedge z = 2, \emptyset \rangle$
$\langle 2, q_0 \times q_0^{\psi}, x = 1 \wedge z < 2, \emptyset \rangle$
$\langle 3, q_0 \times q_0^{\psi}, x = 1 \wedge z = 2, \emptyset \rangle$
$\langle 4, q_0 \times q_0^{\psi}, 1 < x < 2 \wedge z < 2, \emptyset \rangle$
$\langle 5, q_0 \times q_0^{\psi}, 1 < x < 2 \wedge z = 2, \emptyset \rangle$
$\langle 6, q_0 \times q_0^{\psi}, x \geq 2 \wedge z < 2, \emptyset \rangle$
$\langle 7, q_0 \times q_0^{\psi}, x \geq 2 \wedge z = 2, \emptyset \rangle$
$\langle 8, q_0 \times q_1^{\psi}, x < 1, \emptyset \rangle$
$\langle 9, q_0 \times q_1^{\psi}, x = 1, \emptyset \rangle$
$\langle 10, q_0 \times q_1^{\psi}, 1 < x < 2, \emptyset \rangle$
$\langle 11, q_0 \times q_1^{\psi}, x \geq 2, \emptyset \rangle$
$\langle 12, q_1 \times q_0^{\psi}, x < 1 \wedge z < 2, \{p\} \rangle$
$\langle 13, q_1 \times q_0^{\psi}, x < 1 \wedge z = 2, \{p\} \rangle$
$\langle 14, q_1 \times q_0^{\psi}, x = 1 \wedge z < 2, \{p\} \rangle$
$\langle 15, q_1 \times q_0^{\psi}, x = 1 \wedge z = 2, \{p\} \rangle$
$\langle 16, q_1 \times q_1^{\psi}, x < 1, \{p\} \rangle$
$\langle 17, q_1 \times q_1^{\psi}, x = 1, \{p\} \rangle$
$\langle 18, q_2 \times q_0^{\psi}, z < 2, \{r\} \rangle$
$\langle 19, q_2 \times q_0^{\psi}, z = 2, \{r\} \rangle$
$\langle 20, q_2 \times q_1^{\psi}, true, \{r\} \rangle$

Fig. 4. Quotient Graph

checking a finite graph, the strong bi-simulation quotient graph (G_{max}) generated by the algorithm of partition refinement.

Consider a transition system $\mathcal{M} = (\mathcal{S}, \Rightarrow, s_0)$ modeling a strongly non-zeno timed automaton $\mathcal{A}$ and a CTL formula φ. We want to check whether $\mathcal{M}$ satisfies φ. Let $\wp_{max}$ be a strong bi-simulation on $\mathcal{M}$. From $G_{\wp_{max}}$-Property 3 of $G_{\wp_{max}}$, we can conclude that for any CTL formula φ and any pair of states $(s, s') \in \wp_{max}$, $s \models_{\mathcal{M}} \varphi$ if and only if $s' \models_{\mathcal{M}} \varphi$.

A formula is said to hold in a node C of $G_{\wp_{max}}$ if it is satisfied in some state of C (this implies that the formula is satisfied in any state of C). Now, the problem of verifying if a state $s \in \mathcal{S}$ satisfies the CTL formula φ ($s \models_{\mathcal{M}} \varphi$) is reduced to checking if the node $C \in \mathcal{C}$ containing the state s satisfies the formula φ ($C \models \varphi$). The following lemma gives the correctness of the model checking.

Lemma 5.1 *Let $\mathcal{M} = (\mathcal{S}, \Rightarrow, s_0)$ be a transition system modeling a strongly non-zeno automaton, $\mathcal{L}$ is a labeling function associating to each discrete state a set of atomic propositions from AP. Let $\wp_{max}$ be a strong bi-simulation on $\mathcal{M}$ and $G_{\wp_{max}}$ is its quotient graph with the set of nodes $\mathcal{C}$. Let C be in $\mathcal{C}$ and φ a CTL formula. $C \models \varphi$ if and only if $\forall s \in C$, $s \models_{\mathcal{M}} \varphi$.*

Proof. The proof is by induction on the syntax of φ. The basis (φ is an atomic proposition) comes from the fact that $\wp_{max}$ respects $\mathcal{L}$. The case for $\varphi_1 \wedge \varphi_2$ is trivial.

Consider the case where φ is of the form $\varphi_1 \exists U \varphi_2$. Assume that $C' \models \varphi_2$

and $C \models \varphi_1$. If $C \models \varphi$, by the semantics of CTL, there is a path $\rho = C \Rightarrow \cdots \Rightarrow C'$. By $G_{\wp_{max}}$-Property 2 of $G_{\wp_{max}}$, from any state $s \in C$ there is a run inscribed in the path ρ which satisfies the formula φ. This means that $\forall s \in C$, $s \models_{\mathcal{M}} \varphi$.

If $\forall s \in C$, $s \models_{\mathcal{M}} \varphi$. By the semantics of CTL, there is a run $r = s \Rightarrow \cdots \Rightarrow s'$, where $s \models_{\mathcal{M}} \varphi_1$ and $s' \models_{\mathcal{M}} \varphi_2$. We know by $G_{\wp_{max}}$-Property 2 that from any state s there is a run inscribed in a unique path from C ($\rho = C \Rightarrow \cdots \Rightarrow C'$). Then $s' \in C'$. By induction hypothesis, $C \models \varphi_1$ and $C' \models \varphi_2$. By the semantics of CTL, this means $C \models \varphi_1 \exists U \varphi_2$.

The case where φ is of the form $\varphi_1 \forall U \varphi_2$ can be proved by the fact that if $C \not\models \varphi$, we can extract a run r which falsifies φ, from the path ρ starting from the node C using the property $G_{\wp_{max}}$-Property 2. □

Example 5.2 The TCTL model checking of the problem $\langle q_0, x = 0\rangle \models \forall\Diamond_{\geq 2} r$ on the model of the timed automaton of Figure 1, is then reduced to CTL model checking of $C_0 \models \mathit{true} \forall U (q_1^{\psi} \in \mathit{Loc}) \wedge r$ on the model represented by the graph in Figure 4. Where $C_0 = [\langle q_0, x = 0\rangle]$. This CTL formula is not satisfied and the model checking returns a trace $t = C_0 \rightarrow C_1 \rightarrow C_8 \rightarrow C_9 \rightarrow C_{10} \rightarrow C_{11} \rightarrow \cdots$ as a counterexample. By mapping to the concrete timed automaton, the discrete states of the nodes (classes) $C_0, C_1, C_8, C_9, C_{10}, C_{11}$ are q_0. Thus, the concrete trace is $\langle q_0, x = 0\rangle \xrightarrow{\delta_0} \langle q_0, x = \delta_0\rangle \xrightarrow{\delta_1} \langle q_0, x = \delta_0 + \delta_1\rangle \xrightarrow{\delta_2} \cdots$.

6 Conclusion

In this paper, we have presented a technique for model checking dense real-time systems. This method is based on the reduction of TCTL specifications to CTL. The timed behavior of the TCTL specification is captured and represented as a timed automaton. This timed automaton is composed with the original timed automaton modeling the timed system. Then, a time abstraction technique based on strong bi-simulation, is used to generate a finite graph modulo the TCTL specification. A number of branching-time verification tools can be used for performing CTL model-checking on this graph. The advantage of this technique is that there is no need to extend the logic CTL or its model-checking algorithm. Our algorithm of time abstraction is very simple where complicated operations on sets are avoided. Thus, it can be implemented easily.

Effectively, we have implemented a preliminary version of a tool to test this technique for model checking dense real-time systems. During the implementation, we tried to avoid the generation of large quotient graphs by model checking the TCTL formula gradually. Based on the structure of the TCTL formula, we decompose it into sub-formulas and each sub-formula will be checked separately. The overall result is the combination of the partial results.

References

[1] Alur R., Courcoubetis C., and Dill D. L., *Model checking in dense real time*, Information and Computation, **104(1)** (1993), 2–34.

[2] Alur R., and Dill D., *Automaton for modeling real-time systems*, In 17th ICALP, number 443 in Lecture Notes in Computer Science, Springer-Verlag, 1990.

[3] Alur R., and Henzinger T. A., *Logics and models of real time: a survey*, In Real Time: Theory in Practice, number 600 in Lecture Notes in Computer Science, 74–106, Springer-Verlag, 1992.

[4] Alur R., Henzinger T. A., and Ho P. H., *Automatic symbolic verification of embedded systems*, IEEE Transactions on Software Engineering, **22** (1996), 181–201.

[5] Bouajjani A., Fernandez J. C., Halbwachs N., Raymond P., and Ratel C., *Minimal state graph generation*, Science of Computer Programming, **18** (1992), 247–269.

[6] Bouajjani A., Tripakis S., and Yovine S., *On-the-fly symbolic model-checking for real-time systems*, IEEE RTSS'97, IEEE Computer Society Press, 1997.

[7] Bourahla M., and Benmohamed M., *Verification of real-time systems by abstraction of time constraints*, In IPDPS(FMPPTA), IEEE Computer Society, 2003.

[8] Brockmeyer U., and Wittich G., *Real-time verification of statemate designs*, In Computer Aided Verification, number 1427 in Lecture Notes in Computer Science, 537–541, Springer-Verlag, 1998.

[9] Henzinger T. A., and Kupferman O., *From quantity to quality*, In Workshop on Hybrid and Real-Time Systems (HART97), number 1201 in Lecture Notes in Computer Science, 48–62, Springer-Verlag, 1997.

[10] Henzinger T. A., Nicollin X., Sifakis J., and Yovine S., *Symbolic model checking for real-time systems*, Information and Computation, **111(2)** (1994), 193–244.

[11] Larsen K. G., Petterson P., and Yi W., *Compositional and symbolic model checking of real-time systems*, In IEEE Real-Time Systems Symposium, 1995.

[12] Larsen K. G., Petterson P., and Yi W., *Model checking for real-time systems*, In Fundamentals of Computation Theory, number 965 in Lecture Notes in Computer Science, 62–88, Springer-Verlag, 1995.

[13] Milner R., *A calculus of communicating systems*, number 92 in Lecture Notes in Computer Science, Springer-Verlag, 1980.

[14] Moller M. O., Rueb H., and Sorea M., *Predicate abstraction for dense real-time systems*, Electronic Notes in Theoretical Computer Science, **65** (2002).

[15] Paige R., and Tarjan R., *Three partition refinement algorithms*, SIAM Journal on Computing, **16(6)** (1987).

[16] Sokolsky O., and Smolka S. A., *Local Model Checking for Real-Time Systems*, In CAV'95, number 939 in LNCS, Springer-Verlag, 1995.

[17] Spelberg R., Toetenel H., and Ammerlaan M., *Partition refinement in real-time model checking*, In Formal Techniques in Real-Time and Fault-Tolerant Systems, number 1486 in Lecture Notes in Computer Science, Springer-Verlag, 1998.

[18] Tripakis S., and Yovine S., *Analysis of timed systems using time-abstracting bi-simulations*, Formal Methods in System Design, **18** (2001) 25–68.

[19] Yannakakis M., and Lee D., *An efficient algorithm for minimizing real-time transition systems*, In Computer Aided Verification, number 697 in Lecture Notes in Computer Science, 210–224, Springer-Verlag, 1993.