Preservation of timed properties during an incremental development by components

Jacques Julliand, Hassan Mountassir, Emilie Oudot

Preservation of timed properties during an incremental development by components

Jacques Julliand, Hassan Mountassir, and Emilie Oudot

LIFC - Laboratoire d'Informatique de l'Université de Franche-Comté
16, route de Gray, 25030 Besançon Cedex, France
Ph:+33 (0)3 81 66 66 51, Fax:+33 (0)3 81 66 64 50
{julliand,mountass,oudot}@lifc.univ-fcomte.fr

Abstract. We are interested in the preservation of local properties of timed components during their integration in a timed system. Timed components are modeled as timed automata or timed automata with deadlines. Properties considered are all safety and liveness properties which can be expressed with the timed linear logic MITL (Metric Interval Linear Logic), as well as non-zenoness and deadlock-freedom.
Integration of components is a kind of incremental development which consists in checking locally the properties of the components, before integrating them in the complete system, using some composition operator. Of course, established properties have to be preserved by this integration. Checking preservation can be achieved by means of the verification of timed τ-simulation relations. Composability, compatibility and compositionality of these relations w.r.t. composition operators are properties which allow to reduce the cost of this verification. We examine these properties when integration is achieved with two different timed composition operators: the classic operator usually taken for timed systems and which uses a CSP-like composition paradigm, and a non-blocking operator closer to the CCS paradigm.

Key-words. τ-simulations, component-based timed systems, integration of components, preservation of timed linear properties.

1 Introduction

Incremental development methods are a way to cope with the state space explosion problem of model-checking, which is increased in the case of timed systems due to the presence of timing constraints. In particular, for component-based systems, a way to develop incrementally is to use integration of components. This method is indicated for the verification of local properties of the components. It consists in checking the properties in isolation on the component before integrating it in its environment, with some parallel composition operator. Model-checking is there still applicable since the size of the components is generally small enough. Of course, this method is valid only if established properties of the component still hold after integration.

In [1], we defined two τ-simulation relations, adapted to timed systems, with preservation abilities: a timed τ-simulation preserving safety properties, and a divergence-sensitive and stability-respecting (DS) timed τ-simulation which preserves all properties which can be expressed in the linear timed logic MITL (Metric Interval Temporal Logic) [2], strong non-zenoness and deadlock-freedom. We also lead experiments in [3] to show that the cost, in terms of computation times, of local verification and preservation (via the verification of the simulations) is lower than the cost of a direct verification on the global model. The results are encouraging since they show that integration of components, using these relations to guarantee preservation, can speed up verification and that models that are too large to be verified in a whole can be checked in this way.

However, to increase the interest of the method, it would seem interesting to avoid a systematic verification of the simulation. It can be the case that integration automatically preserves properties, depending on the composition operator used to achieve this integration. Properties such as composability, compatibility and compositionality of the simulation relations are a way to reach this goal. Consider components A, B, C and D. Composability is a major property since it expresses that a component A simulates its composition with another component. The direct consequence is that properties of A (which are preserved by the simulation) are automatically preserved by composition. It is thus clearly essential for integration of components, or for the reuse of a component. The compatibility of the relation w.r.t. to composition operators is the classic notion of compatibility. Given some composition operator $\|$, it states that if A simulates B (and thus, properties of A also hold on B) then $A\|C$ simulates $B\|C$. During development, it is beneficial for instance in the case of the replacement in a system of the component A by the component B. Compositionality is a consequence of compatibility since it expresses that if A simulates B and C simulates D then, $A\|C$ simulates $B\|D$.

Therefore, in this paper, we study if the simulations we defined in [1] allow to benefit of these properties, in particular when integration is achieved with one of the two following operators: the classic parallel composition operator used for timed systems and a non-blocking operator defined in [4, 5]. The first operator uses a composition paradigm *a la CSP* [6]. The second one is closer to the paradigm of *CCS* [7] and uses a notion of priorities between actions to favour synchronizations. This analysis shows that the timed τ-simulation is well-adapted to both operators, since we benefit of the three properties without any assumptions. The DS timed τ-simulation is appropriate in the case of the non-blocking operator, on some conditions. This study of the properties of the simulations with respect to these composition operators is the contribution of the paper.

The structure of the paper is the following. In section 2, we recall some background on timed systems. We present timed automata which is the formalism we use to model timed systems and the two composition operators we consider for these automata. Section 3 recalls the simulation relations we defined for timed systems in [1], and their preservation abilities. Section 4 presents the contributions of this paper. We study whether the simulations have the composability, compatibility and compositionality properties w.r.t. the two composition operators. In section 5, we present some related works. Finally, section 6 presents a synthesis of the results obtained, as well as the consequences in terms of preservation during integration, and plans some future works.

2 Preliminary definitions

In this section, we recall some background on the model we consider for timed systems, i.e., timed automata and their variant timed automata with deadlines. We also present the two operators that we consider in this work for the composition of timed automata.

2.1 Timed automata

Timed automata were introduced in [8]. They are finite automata with real-valued variables, called clocks, which model the time elapsing.

Clock valuations and clock constraints. Let X be a set of clocks. A clock valuation over X is a function $v : X \to \mathbb{R}^+$ mapping to each clock in X a value in $\mathbb{R}^+$. We note $\mathbf{0}$ the valuation assigning 0 to each clock in X. Let v be a valuation over X and $t \in \mathbb{R}^+$, the valuation $v + t$ (respectively $v - t$) is obtained by adding (resp. substracting) t to the value of each clock. Given $Y \subseteq X$, the reset in v of the clocks in Y, written $[Y := 0]v$ is the valuation obtained from v by setting to zero all clocks in Y, and leaving the values of other clocks ($\in X \backslash Y$) unchanged. The set $\mathcal{C}_{df}(X)$ of diagonal-free clock constraints over X is defined as follows:

$$g ::= x \sim c \mid g \wedge g \mid true$$

where $x \in X$, $c \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. Diagonal-free constraints do not allow comparisons between clocks, of the form $x - y \sim c$. We say that a valuation v over X satisfies a constraint $x \sim c$, written $v \in x \sim c$, if $v(x) \sim c$. The satisfaction of other constraints is defined as usual. Note that a clock constraint over X defines a convex X-polyhedron. The reset operation defined on valuations can be directly extended to polyhedra. The *backward diagonal projection* of the X-polyhedron ζ defines an X-polyhedron $\swarrow\zeta$ such that $v' \in \swarrow\zeta$ if $\exists \delta \in \mathbb{R}^+ \cdot v' + \delta \in \zeta$. Similarly, the *forward diagonal projection* of ζ defines an X-polyhedron $\nearrow\zeta$ such that $v' \in \nearrow\zeta$ if $\exists \delta \in \mathbb{R}^+ \cdot v' - \delta \in \zeta$. All these operations preserve the convexity of polyhedra.

Syntax. Let *Props* be a set of atomic propositions. A timed automaton (TA) is a tuple $A = \langle Q, q_0, \Sigma, X, T, \texttt{Invar}, L \rangle$, where Q is a finite set of locations, q_0 is the initial location, Σ is an alphabet of names of actions and X is the finite set of clocks. $\texttt{Invar} : Q \to \mathcal{C}_{df}(X)$ is a function which maps a clock constraint

to each location, called its invariant. $L : Q \rightarrow 2^{Props}$ is the labelling function for the locations, mapping a set of atomic propositions to each location. $T \subseteq Q \times \mathcal{C}_{df}(X) \times \Sigma \times 2^X \times Q$ is a finite set of edges. Each edge is a tuple $e = (q, g, a, r, q')$ where q and q' are respectively its source and target location, g is its guard, a is its label and r is a set of clocks to be reset by the edge.
Semantics. The semantics of a TA A is an infinite graph which states are pairs (q, v) where q is a location of A and v is a valuation over X such that $v \in \texttt{Invar}(q)$. The transitions of this graph can be either discrete transitions or time transitions. Consider a state (q, v). Given an edge $e = (q, g, a, r, q')$ of A, $(q, v) \overset{g,a,r}{\rightarrow} (q', v')$ is a discrete transition (where $v' = [r := 0]v$) if $v \in g$ and $v' \in \texttt{Invar}(q')$. We simply note $(q, v) \xrightarrow{a} (q', v')$ such a transition, when the other elements are irrelevant. Given $t \in \mathbb{R}^+$, the time transition $(q, v) \xrightarrow{t} (q, v + t)$ exists if $v + t \in \texttt{Invar}(q)$. Given a state $s = (q, v)$, the notation $s + t$ denotes the pair $(q, v + t)$. In the sequel, we say directly states and transitions of the TA A instead of states and transitions of the semantic graph of A.
A run of a TA is a path of its semantic graph. Thus, a run is a finite or infinite sequence $\rho = s_0 \xrightarrow{t_0} s_0 + t_0 \xrightarrow{a_0} s_1 \xrightarrow{t_1} s_1 + t_1 \xrightarrow{t_2} s_1 + t_1 + t_2 \xrightarrow{a_1} s_2 \cdots$. Note that we do not concatenate successive time transitions in a run. A run ρ is said non-zeno if time can diverge along the run, i.e., the total time elapsed along the run goes to infinity. A TA is said strongly non-zeno if all its runs are non-zeno.

A variant: timed automata with deadlines. Timed automata with deadlines (TAD) are a variant of TA introduced in [9]. The main difference lies in the fact that time-progress conditions are not given as invariants in locations (invariants mean that time can progress in the same way for each outgoing edge of the location), but are associated with the edges. This allows to express the urgency of an edge since the deadline represents the moment when time can not progress any more before taking the edge. The following three degrees of urgency are considered in [9]. Lazy edges, written λ, are edges which degree of urgency is the lowest since time is never stopped. Eager edges, written ϵ, become urgent at the moment they become enabled (an edge is enabled when its guard is true), i.e., they can not let time elapse once they are enabled. Delayable edges, written δ, are the most currently used: time can pass as long as the edge is enabled.
Formally, the syntax of TAD is the same than the one of TA, with no invariants. Edges are tuples (q, g, d, a, r, q') where q, g, a, r and q' are defined as for TA and $d \in \{\lambda, \varepsilon, \delta\}$ is the type of deadline of the edge. Deadlines are then translated into clock constraints computed from the guards. For an edge e, this clock constraint is denoted $\texttt{deadline}(e)$. The clock constraint is *false* for lazy edges, equal to the guard for eager edges and equal to the falling edge (i.e., the last moment when the guard is true) for delayable edges [5]. Note that invariants can be deduced from the deadlines, by the following formula: $\texttt{Invar}(q) = \neg \bigvee_{e \in \texttt{out}(q)} \texttt{deadline}(e)$, where $\texttt{out}(q)$ is the set of outgoing edges of the location q. Semantics of TAD is then defined as for TA. Discrete transitions are written $s \overset{g,d,a,r}{\rightarrow} s'$ (or, as for TA, simply $s \xrightarrow{a} s'$) where d represents the deadline of the corresponding edge.

2.2 Timed parallel composition operators

We consider in this paper two parallel composition operators which take into account the timing constraints of the components. The first one, which is the classic operator for TA, uses a composition paradigm close to the one of CSP, while the second, which we call non-blocking parallel composition operator, is closer to the paradigm of CCS and uses a notion of priorities between actions.

Classic parallel composition operator. This composition operator, written $\|$, operates between TA with disjoint sets of clocks. Intuitively, it is defined as a synchronized product, where actions with the same label synchronize, other actions interleave and time elapses synchronously between the components. Formally, consider two TA $A = \langle Q_A, q_{0_A}, \Sigma_A, X_A, T_A, \texttt{Invar}_A, L_A\rangle$ and $B = \langle Q_B, q_{0_B}, \Sigma_B, X_B, T_B, \texttt{Invar}_B, L_B\rangle$, such that $X_A \cap X_B = \emptyset$. The classic parallel composition of A and B, written $A\|B$, results in a TA which set of clocks is $X_A \cup X_B$ and which labels are in $\Sigma_A \cup \Sigma_B$. The set Q of locations is a subset of $Q_A \times Q_B$. The initial location is the pair (q_{0_A}, q_{0_B}). The invariant of a location (q_A, q_B) is $\texttt{Invar}(q_A) \wedge \texttt{Invar}(q_B)$ and its label is $L(q_A) \cup L(q_B)$. The set T of edges is defined by the three following rules:

$$\text{Interleaving:} \quad \frac{(q_A,q_B)\in Q \ , \ (q_A,g_A,a,r_A,q'_A) \in T_A \ , \ a\notin\Sigma_B}{((q_A,q_B),g_A,a,r_A,(q'_A,q_B)) \in T}$$

$$\frac{(q_A,q_B)\in Q \ , \ (q_B,g_B,b,r_B,q'_B) \in T_B \ , \ b\notin\Sigma_A}{((q_A,q_B),g_B,b,r_B,(q_A,q'_B)) \in T}$$

$$\text{Synchronization:} \quad \frac{\begin{array}{c}(q_A,q_B)\in Q,\ (q_A,g_A,a,r_A,q'_A) \in T_A \ ,\\ (q_B,g_B,a,r_B,q'_B) \in T_B\end{array}}{((q_A,q_B),g_A\wedge g_B,a,r_A\cup r_B,(q'_A,q'_B)) \in T}$$

The main drawback of this composition operator is that deadlocks are generally introduced during composition. For this reason, other operators have been defined, such as the following one.

Non-blocking parallel composition operator. This operator was introduced in [4, 5] to operate between TAD with disjoint alphabets and sets of clocks. It is defined as a product in which all actions interleave and time elapses synchronously between components. Some actions also synchronize, according to a synchronization function $\imath\colon \Sigma_A \times \Sigma_B \to \Sigma_{sync} \cup \{\perp\}$, where Σ_{sync} is an alphabet disjoint from Σ_A and Σ_B. The function maps to each pair of labels (a, b) the label in Σ_{sync} of the action resulting of the synchronization of two actions a and b, or the special symbol $\perp$ if the two actions do not synchronize. Different synchronization modes can be used. The AND mode is the classic one, i.e., synchronization takes place between two actions if both can be taken. The MIN mode corresponds to a synchronization with interruption, i.e., the first enabled action causes the synchronization even if the other one is not yet enabled. Finally, the MAX mode corresponds to a synchronization with waiting, i.e., the first enabled action waits for the other to be enabled for synchronization to occur.

Formally, consider two TAD $A = \langle Q_A, q_{0_A}, \Sigma_A, X_A, T_A, L_A\rangle$ and $B = \langle Q_B, q_{0_B}, \Sigma_B, X_B, T_B, L_B\rangle$ such that $X_A \cap X_B = \emptyset$ and $\Sigma_A \cap \Sigma_B = \emptyset$. Given a synchronization function $\shortmid$, the non-blocking parallel composition of A and B, written $A|B$, results in a TAD which set of clocks is $X_A \cup X_B$. The set Q of locations is a subset of $Q_A \times Q_B$. The initial location is the pair (q_{0_A}, q_{0_B}). The label of a location (q_A, q_B) is $L(q_A) \cup L(q_B)$. The labels of the automaton are in $\Sigma_A \cup \Sigma_B \cup \Sigma_{sync}$ and the set T of edges is defined by the following rules:

$$\text{Interleaving:} \quad \frac{(q_A,q_B)\in Q \;,\; (q_A,g_A,d_A,a,r_A,q'_A) \in T_A}{((q_A,q_B),g_A,d_A,a,r_A,(q'_A,q_B)) \in T}$$

$$\frac{(q_A,q_B)\in Q \;,\; (q_B,g_B,d_B,b,r_B,q'_B) \in T_B}{((q_A,q_B),g_B,d_B,b,r_B,(q_A,q'_B)) \in T}$$

$$\text{Synchronization:} \quad \frac{\begin{array}{c}(q_A, q_B) \in Q,\; (q_A, g_A, d_A, a, r_A, q'_A) \in T_A\;,\\ (q_B, g_B, d_B, b, r_B, q'_B) \in T_B,\; a \shortmid b \neq \bot\end{array}}{((q_A, q_B), g', max(d_A, d_B), a \shortmid b, r_A \cup r_B, (q'_A, q'_B)) \in T}$$

where $max(d_A, d_B)$ is defined by $\lambda < \delta < \varepsilon$, and g' is computed as follows:

- $g' = g_A \wedge g_B$ for the AND mode,
- $g' = (g_A \wedge \swarrow g_B) \vee (\swarrow g_A \wedge g_B)$ for the MAX mode,
- $g' = (g_A \wedge \nearrow g_B) \vee (\nearrow g_A \wedge g_B)$ for the MIN mode.

Since all actions interleave, a priority order written $<^{sync}_{\infty}$ is used to favour synchronized actions rather than interleaving. An infinite priority is given to a synchronized action against the interleaving actions from which results this synchronization. This priority order induces a modification of the guard of the edges of lower priority. Given an action a_i and actions a_j leaving from the same location, such that a_i has a lower priority comparing to the a_j, the guard g_i of a_i is modified to g'_i, where

$$g'_i = g_i \wedge \bigwedge_{j \neq i, a_i <^{sync}_{\infty} a_j} \neg \nearrow g_j.$$

This priority order is applied as follows. If $a \in \Sigma_A$, $b \in \Sigma_B$ and $a \shortmid b \neq \bot$ then $a <^{sync}_{\infty} a \shortmid b$ and $b <^{sync}_{\infty} a \shortmid b$.

3 Timed τ-simulations to preserve properties

A way to ensure preservation of properties between two models is to compare the two models, i.e., the one on which properties are checked and the one on which they must be preserved. This comparison can be achieved by means of behavioral equivalences or preorders. Preorders are generally more adapted to incremental development and have already been used for this matter. For instance, in [10], the refinement of transition systems is formalized as a kind of simulation preorder which preserves LTL (Linear Temporal Logic) [11] properties.
In [1], we defined two kinds of simulation relations for timed automata. The first one, called timed τ-simulation, preserves safety properties. The second one, called divergence-sensitive and stability-respecting (DS) timed τ-simulation, preserves all properties expressed with the timed linear logic MITL (Metric Interval Temporal Logic) [2], strong non-zenoness and deadlock-freedom.

3.1 Definitions

Consider two TA A and B with respective alphabets Σ_A and Σ_B, s.t. $\Sigma_A \subseteq \Sigma_B$. In B, actions in $\Sigma_B \backslash \Sigma_A$ are called non-observable and renamed by τ. Other actions, in Σ_A, are called observable. The following conditions must hold in order that A simulates B w.r.t. the timed τ-simulation, and thus that safety properties of A are preserved on B. The first condition (clauses 1 and 2 in Def. 1) expresses that if B can make an observable action after some amount of time, then A could do the same observable action after the same amount of time. The second one (clause 3 in Def. 1) requires that non-observable actions stutter. This is a classic definition of τ-simulation that we extend to the timed framework.
Such a simulation only preserves safety properties. To deal with a wider range of properties, in particular liveness, additionnal conditions, called stability-respect and divergence-sensitivity, are necessary. Divergence-sensitivity (clause 4 of Def. 1) states that there are no non-zeno infinite sequences of non-observable actions in B, i.e., there are no non-zeno τ-cycles in B (a cycle in which discrete transitions are only labelled by τ is called a τ-cycle). Stability-respect (clause 5 of Def. 1) means that B must not contain deadlocks which do not exist in A. To express formally this condition, we use the predicate `free` defined in [12]. Given a location q, $\texttt{free}(q)$ is the set of all valuations (of states with q as discrete part) from which a discrete transition can be taken after some delay. Formally,

$$\texttt{free}(q) = \bigcup_{(q,g,a,r,q') \in T} \swarrow (g \cap ([r := 0]\texttt{Invar}(q'))).$$

The simulations are defined on the semantics of TA. In the sequel, we focus directly on the definition of the DS timed τ-simulation $\mathcal{S}_{ds}$. The definition of the timed τ-simulation, written $\mathcal{S}$, can be obtained by removing the clauses *divergence-sensitivity* and *stability-respect*.

Definition 1 (Divergence-sensitive and stability-respecting (DS) timed τ-simulation $\mathcal{S}_{ds}$). *Let* $A = \langle \mathrm{Q}_A, \mathrm{q}_{0_A}, \Sigma_A, \mathrm{X}_A, \mathrm{T}_A, \texttt{Invar}_A, \mathrm{L}_A\rangle$ *and* $B = \langle \mathrm{Q}_B, \mathrm{q}_{0_B}, \Sigma_A \cup \{\tau\}, \mathrm{X}_B, \mathrm{T}_B, \texttt{Invar}_B, \mathrm{L}_B\rangle$ *be two TA s.t.* $X_A \subseteq X_B$. *We call* S_A *and* S_B *the respective set of states of* A *and* B. *The DS timed* τ*-simulation* $\mathcal{S}_{ds}$ *is the greatest binary relation included in* $S_B \times S_A$. *Consider* $s_A = (q_A, v_A)$ *in* S_A *and* $s_B = (q_B, v_B)$ *in* S_B. *We say that* $s_B \mathcal{S}_{ds} s_A$ *if the following conditions hold:*

1. *Strict simulation:*
   $s_B \xrightarrow{a} s'_B \wedge a \in \Sigma_A \Rightarrow \exists s'_A \cdot (s_A \xrightarrow{a} s'_A \wedge s'_B\ \mathcal{S}_{ds}\ s'_A)$.
2. *Delays equality*[1]*:*
   $s_B \xrightarrow{t} s_B + t \Rightarrow \exists (s_A + t) \cdot (s_A \xrightarrow{t} s_A + t \wedge s_B + t\ \mathcal{S}_{ds}\ s_A + t)$.
3. *τ-transitions stuttering:*
   $s_B \xrightarrow{\tau} s'_B \Rightarrow s'_B\ \mathcal{S}_{ds}\ s_A$.
4. *Divergence-sensitivity:*
   B does not contain any non-zeno τ-cycles.
5. *Stability-respect:*
   $v_B \notin \texttt{free}(q_B) \Rightarrow v_A \notin \texttt{free}(q_A)$.

[1] Note that we do not intend to check this semantic definition directly. For algorithmic purpose, it is extended into a symbolic relation where this clause *delays equality* consists in polyhedra inclusion and thus, is decidable.

We extend this notion of simulation on TA. Given two TA A and B, and their respective initial state s_{0_A} and s_{0_B}, we say that A simulates B w.r.t. $\mathcal{S}_{ds}$ (respectively $\mathcal{S}$), written $B \preceq_{\mathcal{S}_{ds}} A$ (resp. $B \preceq_{\mathcal{S}} A$) if $s_{0_B} \mathcal{S}_{ds} s_{0_A}$ (resp. $s_{0_B} \mathcal{S} s_{0_A}$).

3.2 Preservation abilities

The timed τ-simulation preserves safety properties. This result for this kind of simulation is a classic result in the untimed case, which we extend to the timed framework. Adding divergence-sensitivity and stability-respect allows to preserve also liveness properties. More precisely, the DS timed τ-simulation preserves all properties expressed with the timed linear logic MITL, as well as strong non-zenoness and deadlock-freedom. A complete proof can be found in [1].

Theorem 1. *Let A and B be TA, and φ be a safety property. If $B \preceq_{\mathcal{S}} A$ and φ holds on A then φ holds on B.*

Theorem 2. *Let A and B be TA, and φ be a MITL formula. We have the following:*

- *If $B \preceq_{\mathcal{S}_{ds}} A$ and φ holds on A then φ holds on B.*
- *If $B \preceq_{\mathcal{S}_{ds}} A$ and A is strongly non-zeno then B is strongly non-zeno.*
- *If $B \preceq_{\mathcal{S}_{ds}} A$ and A is deadlock-free then B is deadlock-free.*

4 Properties of timed τ-simulations w.r.t. timed parallel composition

We implemented the verification of these simulations in a tool named VeSTA[2] (Verification of Simulations for Timed Automata). With this tool, we performed experimentations concerning the verification of local properties of components [3]. We compared the two following methods: first, direct verification of these local properties on the complete model, made up of all the components, and secondly local verification and integration of components, using the simulations as a way to guarantee the preservation. Even if the first experimental results are encouraging, it seems interesting to avoid a systematic verification of the relations. For this purpose, composability, compatibility and compositionality of the relations w.r.t. the composition operators used for integration of components are essential properties. Thus, in this section, we study these three properties for the timed τ-simulation and the DS one w.r.t. the two operators presented in section 2.2.

In the sequel, we use the following notations. Given a timed automaton A, we note S_A its set of states and Σ_A its alphabet. A state of A is simply written s_A or s'_A, which respectively represent the pairs (q_A, v_A) and (q'_A, v'_A). The initial state of A is written s_{0_A}.

[2] VeSTA is available at: `http://lifc.univ-fcomte.fr/~oudot/VeSTA`

4.1 Classic parallel composition

We first examine the properties with the timed τ-simulation.

Proposition 1 (Composability). *Let A and B be TA. We have: $A\|B \preceq_{\mathcal{S}} A$.*

Proof. By construction of $A\|B$, its initial state is the pair (s_{0_A}, s_{0_B}). To prove that $A\|B \preceq_{\mathcal{S}} A$, it is enough to prove that $(s_{0_A}, s_{0_B})\mathcal{S}s_{0_A}$. By definition, $\preceq_{\mathcal{S}}$ is the greatest relation included in $S_{A\|B} \times S_A$ which satisfies clauses 1 to 3 of Definition 1. Thus, each relation $R \subseteq S_{A\|B} \times S_A$ which satisfies these clauses is included in $\preceq_{\mathcal{S}}$. Consider a relation $R \subseteq S_{A\|B} \times S_A$ such that $\forall (s_A, s_B) \in S_{A\|B}$,

$$(s_A, s_B)R\ s'_A \text{ if } s_A = s'_A.$$

Consider $((s_A, s_B), s_A) \in R$.

1. *Strict simulation*: let $(s_A, s_B) \xrightarrow{a} (s'_A, s'_B)$ in $A\|B$ such that $a \in \Sigma_A$. By construction of $A\|B$, a transition $s_A \xrightarrow{a} s'_A$ exists in A. By definition of R, $(s'_A, s'_B)R\ s'_A$ and R satisfies the *strict simulation.*
2. *Delays equality*: same arguments than those for strict simulation can be used to prove that this clause holds for R.
3. *τ-transitions stuttering*: consider a transition $(s_A, s_B) \xrightarrow{\tau} (s'_A, s'_B)$ in $A\|B$. Recall that τ-transitions represent non-observable actions initially labelled by an action in $\Sigma_B \backslash \Sigma_A$. By construction of $A\|B$, $s'_A = s_A$. Thus, $(s_A, s'_B)R\ s_A$ and R satisfies *τ-transitions stuttering.*

Proposition 2 (Compatibility). *Let A, B and C be TA. If $A \preceq_{\mathcal{S}} B$ then $A\|C \preceq_{\mathcal{S}} B\|C$.*

Proof. The structure of the proof is similar to the previous one. Consider a relation $R \subseteq S_{A\|C} \times S_{B\|C}$ such that $(s_A, s_C)R(s_B, s'_C)$ if $s_A\mathcal{S}s_B$ and $s_C = s'_C$. As previously, we prove that R satisfies clauses 1 to 3 of Definition 1. Let $((s_A, s_C), (s_B, s_C)) \in R$,

1. *Strict simulation* : the proof is naturally divided into three parts, since this clause concerns three types of transitions in $A\|C$:
   (i) Transitions of C which do not synchronize with an action of A,
   (ii) Transitions in $\Sigma_A \cap \Sigma_B$[3] which do not synchronize with an action of C,
   (iii) Transitions in C which synchronize with a transition in A (and which appear in $B\|C$ either as interleaving actions of C if the synchronization is done with an action of A which does not exist in B, or as an action of B synchronized with an action of C otherwise).

   Let us detail the three cases:
   (i) Consider a transition $(s_A, s_C) \xrightarrow{g,c,r} (s_A, s'_C)$ such that $c \in \Sigma_C \backslash \Sigma_A$. By construction of $A\|C$, a transition $s_C \xrightarrow{g,c,r} s'_C$ exists in C. Therefore, g only involves clocks of C and $v_C \in g$. Thus, by construction of $B\|C$, a transition $(s_B, s_C) \xrightarrow{g,c,r} (s_B, s'_C)$ exists in $B\|C$. Since $s_A\mathcal{S}s_B$, and by definition of R, we have $(s_A, s'_C)R(s_B, s'_C)$.

[3] These transitions exist since $A \preceq_{\mathcal{S}} B$ and thus $\Sigma_B \subseteq \Sigma_A$.

(ii) Consider a transition $(s_A, s_C) \overset{g,a,r}{\longrightarrow} (s'_A, s_C)$ such that $a \in (\Sigma_A \cap \Sigma_B) \backslash \Sigma_C$ (the state s_C is not modified). By definition of $\|$, the transition $s_A \overset{g,a,r}{\longrightarrow} s'_A$ exists in A. Since $s_A \mathcal{S} s_B$, there is a transition $s_B \overset{g',a,r'}{\longrightarrow} s'_B$ in B, such that $s'_A \mathcal{S} s'_B$. Thus, $v_B \in g'$. Since g' only involves clocks of B and that the set of clocks of B and C are disjoint (by hypothesis in the construction of $B\|C$), then $(v_B, v_C) \in g'$ and the transition $(s_B, s_C) \overset{g',a,r'}{\longrightarrow} (s'_B, s_C)$ exists in $B\|C$. Thus, $(s'_A, s_C) R (s'_B, s_C)$ by definition of R. Note that (s_B, s_C) is reachable. Indeed, (s_A, s_C) is reachable in $A\|C$. Thus, in C, there exists a prefix of a run ρ_C up to the state s_C which "synchronize" with the prefix of a run ρ_A of A up to the state s_A. As $s_A \mathcal{S} s_B$, this prefix of ρ_A is simulated by the prefix of a run ρ_B in B up to the state s_B. Comparing to ρ_B, the run ρ_A has the same ordering of observable actions with possibly non-observable actions inserted between them. Thus, the prefix of ρ_B can also "synchronize" with the prefix of ρ_C and therefore, the state (s_B, s_C) is reachable. This observation holds for the rest of the proof.

(iii) Consider a transition $(s_A, s_C) \overset{g,a,r}{\longrightarrow} (s'_A, s'_C)$ such that $a \in \Sigma_A \cap \Sigma_C$. By definition of $\|$, there is a transition $s_A \overset{g_1,a,r_1}{\longrightarrow} s'_A$ in A and a transition $s_C \overset{g_2,a,r_2}{\longrightarrow} s'_C$ in C. There are two cases: either $a \in \Sigma_A \cap \Sigma_B$ (a is an observable action of A comparing to B and exists in B), or $a \in \Sigma_A \backslash \Sigma_B$ (a is a non-observable action of A which does not exists in B).
In the first case, since $s_A \mathcal{S} s_B$, there is a transition $s_B \overset{g_3,a,r_3}{\longrightarrow} s'_B$ in B such that $s'_A \mathcal{S} s'_B$. Thus, there is a transition $(s_B, s_C) \overset{g',a,r'}{\longrightarrow} (s'_B, s'_C)$ in $B\|C$, such that $(s'_A, s'_C) R (s'_B, s'_C)$ by definition of R.
In the second case, $s'_A \mathcal{S} s_B$ since $s_A \mathcal{S} s_B$. The transition $(s_B, s_C) \overset{g_2,a,r_2}{\longrightarrow} (s_B, s'_C)$ exists in $B\|C$ since v_B does not involve clocks of C and $v_C \in g_2$. By definition of R, we have $(s'_A, s'_C) R (s_B, s'_C)$.

Thus, the relation R satisfies the *strict simulation*.

2. *Delays equality*: consider a time transition $(s_A, s_C) \xrightarrow{t} (s'_A, s'_C)$. By definition of $\|$, the transitions $s_A \xrightarrow{t} s'_A$ and $s_C \xrightarrow{t} s'_C$ exist respectively in A and C. Since $s_A \mathcal{S} s_B$, then the transition $s_B \xrightarrow{t} s'_B$ exists in B and $s'_A \mathcal{S} s'_B$. The transition $(s_B, s_C) \xrightarrow{t} (s'_B, s'_C)$ exists also in $A\|C$ and, by definition of R, $(s'_A, s'_C) R (s'_B, s'_C)$.
3. *τ-transitions stuttering*: in $A\|C$, comparing to $B\|C$, τ-transitions are labelled in $\Sigma_A \backslash (\Sigma_B \cup \Sigma_C)$. Consider a transition $(s_A, s_C) \xrightarrow{\tau} (s'_A, s_C)$ (the state s_C is not modified since τ represents an action of $\Sigma_A \backslash (\Sigma_B \cup \Sigma_C)$). Since $s_A \mathcal{S} s_B$, we have $s'_A \mathcal{S} s_B$. It follows that $(s'_A, s_C) R (s_B, s_C)$.

Proposition 3 (Compositionality). *Let A, B, C and D be TA. If $A \preceq_{\mathcal{S}} B$ and $C \preceq_{\mathcal{S}} D$ then $A\|C \preceq_{\mathcal{S}} B\|D$.*

Proof. Immediate with Proposition 2. Since $A \preceq_{\mathcal{S}} B$, then $A\|C \preceq_{\mathcal{S}} B\|C$. Since $C \preceq_{\mathcal{S}} D$, then $B\|C \preceq_{\mathcal{S}} B\|D$. By transitivity of the relation $\preceq_{\mathcal{S}}$, we have $A\|C \preceq_{\mathcal{S}} B\|D$.

The timed τ-simulation allows to benefit of the three properties for free. This is not the case for the DS timed τ-simulation. Indeed, the operator $\|$ is known to introduce deadlocks during composition, which prevents the clause stability-respect of the DS timed τ-simulation from being established. However, this simulation allows to get the properties when using the non-blocking parallel composition operator, on some simple conditions.

4.2 Non-blocking parallel composition

First note that the non-blocking parallel composition operates between TAD. Thus, in this section, we extend the notations $\preceq_{\mathcal{S}}$ and $\preceq_{\mathcal{S}_{ds}}$, initially defined for TA, to TAD. This extension does not matter since the simulations are defined at a semantic level and that the semantics of TAD is given by an infinite graph of the same kind than for TA.

Consider two TAD A and A', and suppose that A' is obtained from A by integration of components using the non-blocking parallel composition operator, i.e., $A' = A|B$ for some automaton B. For a TAD A to simulates a TAD A', we imposed in the definition of the simulations that $\Sigma_A \subseteq \Sigma_{A'}$. Moreover, we suppose that observable actions in A' (and, in particular, synchronized actions) have the same label than in A. Recall that a synchronization function must be provided with the non-blocking parallel composition operator to describe the synchronizations. Without loss of generality, we consider here that the synchronization function is defined by $\shortmid\colon \Sigma_A \times \Sigma_B \rightarrow \Sigma_A \cup \{\bot\}$ such that, given $a \in \Sigma_A$ and $b \in \Sigma_B$, $a \shortmid b = a$ if the two actions synchronize. In other words, the label of the synchronized action is the same than the one of the action of A which takes part in the synchronization.

We focus directly on the DS timed τ-simulation. Indeed, as we will note at the end of the section, the results for the timed τ-simulation are the same than in the case of the classic operator. Recall also that different synchronization modes can be used with the non-blocking operator. We first consider the most frequently used mode, which is the AND one. First, the following result is necessary for composability.

Proposition 4 (Non τ-divergence preservation). *Let A and B be TAD. Actions in $\Sigma_B \backslash \Sigma_A$ are renamed by τ. If B does not contain any non-zeno τ-cycles, then $A|B$ does not contain any non-zeno τ-cycles.*

Proof. (by contradiction) Suppose that there are no non-zeno τ-cycles in B. Suppose also that there exists a non-zeno τ-cycle in $A|B$. If such a cycle exists in $A|B$, there is a non-zeno run which, from one point, only takes time transitions or τ-transitions. By construction of $A|B$, if there is a time transition $(s_A, s_B) \xrightarrow{t} (s'_A, s'_B)$ in $A|B$, then the transitions $s_A \xrightarrow{t} s'_A$ and $s_B \xrightarrow{t} s'_B$ respectively exist in A and B. In the same way, if a transition $(s_A, s_B) \xrightarrow{\tau} (s_A, s'_B)$ exists in $A|B$, then the transition $s_B \xrightarrow{\tau} s'_B$ exists in B. Thus, by construction of $A|B$, B must contain a non-zeno run which, from one point, only takes time transitions or τ-transitions, which leads to a contradiction with the assumption that B does not contain any non-zeno τ-cycles.

Proposition 5 (Composability). *Let A and B be TAD. Actions in $\Sigma_B \backslash \Sigma_A$ are renamed by τ in B. If B does not contain any non-zeno τ-cycles, we have: $A|B \preceq_{\mathcal{S}_{ds}} A$.*

Proof. We prove this proposition using the same method than for proposition 1. By construction of $A|B$, its initial state is the pair (s_{0_A}, s_{0_B}). As previously, we must prove that $(s_{0_A}, s_{0_B})\mathcal{S}_{ds}s_{0_A}$. Thus, we consider a relation $R \subseteq S_{A|B} \times S_A$, and prove that it is included in $\mathcal{S}_{ds}$ by showing that it satisfies the clauses of the definition of $\mathcal{S}_{ds}$. Consider a relation R such that $\forall (s_A, s_B) \in S_{A|B}$,

$$((s_A, s_B), s'_A) \in R \text{ if } s_A = s'_A.$$

Let $((s_A, s_B), s_A) \in R$.

1. *Strict simulation*: let $(s_A, s_B) \overset{g,d,a,r}{\rightarrow} (s'_A, s'_B)$ be a transition such that $a \in \Sigma_A$, i.e., a is either an interleaving action of A, or an action of A which synchronizes with an action of B. In both cases, this transition results of an edge $((q_A, q_B), g, d, a, r, (q'_A, q'_B))$ of $A|B$ (in the case of an interleaving action, note that $s'_B = s_B$, but this observation does not change anything to what follows).
   If a is a synchronized action, then it results of the synchronization of an edge $(q_A, g_A, d_A, a, r_A, q'_A)$ of A with an edge $(q_B, g_B, d_B, b, r_B, q'_B)$ of B, such that the synchronization function defines $a \mid b = a$. By definition, $g = g_A \wedge g_B$ and $r = r_A \cup r_B$. Since the transition $(s_A, s_B) \overset{g,d,a,r}{\rightarrow} (s'_A, s'_B)$ exists in $A|B$, $(v_A, v_B) \in g$. As v_A and g_A only involve clocks of A, it follows directly that $v_A \in g_A$. Moreover, since $(v'_A \cup v'_B) = [r := 0](v_A, v_B)$, and that $r \cap X_A = r_A$, we have $v'_A = [r_A := 0]v_A$, and thus the transition $s_A \overset{g_A,d_A,a,r_A}{\rightarrow} s'_A$ exists in A. By definition of R, we have $((s'_A, s_B), s'_A) \in R$.
   If a is an interleaving action, an edge $(q_A, g_A, d_A, a, r_A, q'_A)$ exists in A, with $g = g_A$. The same reasoning than for synchronized actions applies to this case: the transition $s_A \overset{g_A,d_A,a,r_A}{\rightarrow} s'_A$ exists in the semantic graph of A, and by definition of R, we have $((s'_A, s_B), s'_A) \in R$.
   In both cases, R satisfies the clause *strict simulation*.
2. *Delays equality* : consider a transition $(s_A, s_B) \xrightarrow{t} (s_A + t, s_B + t)$ in $A|B$. This transition appears in $A|B$ if the invariant of the location (q_A, q_B) is satisfied by the valuations $(v_A, v_B) + t'$, $\forall t' < t$. Recall that invariants of $A|B$ are deduced from the deadlines of the outgoing edges of (q_A, q_B). This means that the previous transition exists if the valuations $(v_A, v_B) + t'$ do not satisfy any deadlines of the outgoing edges of (q_A, q_B). Edges leaving from (q_A, q_B) can be either only interleaving actions, or also synchronized actions. First consider that only interleaving actions leave from (q_A, q_B). If there are no edges of B leaving from (q_A, q_B), then time elapsing from (s_A, s_B) is only

dependant of the deadlines of the edges of A, and thus time can elapse as it did in A. It follows directly that if the transition $(s_A, s_B) \xrightarrow{t} (s_A+t, s_B+t)$ exists in $A|B$, then the transition $s_A \xrightarrow{t} s_A+t$ exists in A. If edges of B also leave from (q_A, q_B), and that their deadlines are reached *later* than the ones of the edges of A, time elapsing will still be dependant of the deadlines of A, which leads to a similar case than the previous one. Otherwise, time elasping is dependant of the deadlines of the edges of B, and since the deadline is stronger, time elapses less than in A, which is required since we impose than time elapses at most as in A. It follows that $s_A \xrightarrow{t} s_A+t$ exists in A.
The analysis for synchronized actions is similar to the previous one, since, by definition of $|$, the deadline of a synchronized action is the stronger deadline of the actions which synchronize.
Thus, if a transition $(s_A, s_B) \xrightarrow{t} (s_A+t, s_B+t)$ exists in $A|B$, then there is a transition $s_A \xrightarrow{t} s_A+t$ in A'. By definition of R, $((s_A+t, s_B+t), s_A+t) \in R$, and thus R satisfies the clause *delays equality.*

3. *τ-transitions stuttering* : consider a transition $(s_A, s_B) \xrightarrow{\tau} (s'_A, s'_B)$ in $A|B$. These τ-transitions are non-observable actions in $A|B$, i.e., actions of B which do not synchronize with an action of A. By definition of $|$, these actions occur in $A|B$ as interleaving actions. Thus, $s'_A = s_A$ and, by definition of R, $((s_A, s'_B), s_A) \in R$ and R satisfies the clause *τ-transitions stuttering.*
4. *Divergence-sensitivity* : immediate, with proposition 4, since B does not contain any non-zeno τ-cycles.
5. *Stability-respect* : immediate by definition of $|$ (all actions interleave).

Proposition 6 (Compatibility). *Let A, B and C be TAD. If $A \preceq_{\mathcal{S}_{ds}} B$ then $A|C \preceq_{\mathcal{S}_{ds}} B|C$.*

Proof. Similar arguments than in the proof of proposition 2 apply in this case for the clauses 1 to 3. The proof for stability-respect is immediate by definition of $|$ and the fact that this operator does not introduce deadlocks, due to a total interleaving of all the actions. Divergence-sensitivity is ensured since τ-transitions of $A|C$ are labelled with actions in $\Sigma_B \backslash \Sigma_A$ and since $A \preceq_{\mathcal{S}_{ds}} B$.

Proposition 7 (Compositionality). *Let A, B, C and D be TA. The internal actions of $A|C$ (i.e. in $(\Sigma_A \cup \Sigma_C) \backslash (\Sigma_B \cup \Sigma_D)$) are renamed by τ. If $A \preceq_{\mathcal{S}_{ds}} B$, $C \preceq_{\mathcal{S}_{ds}} D$ and $A|C$ does not contain any non-zeno τ-cycles then $A|C \preceq_{\mathcal{S}_{ds}} B|D$.*

Proof. Immediate with proposition 6.

Remark 1 (MIN and MAX modes). Recall that the MIN synchronization mode corresponds to a synchronization with interruption, and that the guard of the synchronized action is modified to take into account this paradigm. As for the AND mode, this change consists in strengthening the guard of the synchronized action, which implies that the concerned clauses (*strict simulation* and *delays equality*) hold. Therefore, propositions 5 to 7 also hold when using the MIN mode. The MAX synchronization mode corresponds to a synchronization with waiting, which means that when one action in the synchronization is enabled, it waits for the other to be enabled to synchronize. It follows immediatly that the propositions do not hold. For instance, composability does not hold since synchronized actions (which are observable actions) can be taken later in $A|B$ than they were in A. Similar arguments can be given in the case of compatibility and compositionality.

Remark 2 (Timed τ-simulation and $|$). We focused on the properties of the DS timed τ-simulation. The properties also hold, without assumptions, in the case of timed τ-simulation, when using AND and MIN modes. Indeed, the DS timed τ-simulation is obtained from the timed τ-simulation by adding divergence-sensitivity and stability-respect. As the three properties hold for the DS timed τ-simulation (modulo assumptions for divergence-sensitivity in the case of composability and compositionality), they also hold for the timed τ-simulation.

4.3 Synthesis

Table 1 gives a synthesis of the results presented in this section. The abbreviations *hyp. div. 1* and *hyp. div. 2* represent respectively the assumptions of propositions 5 and 7 for divergence-sensitivity. Table 2 gives an interpretation of these results in terms of properties preserved for integration of components.

	Classic parallel composition	Non-blocking parallel composition	
		AND / MIN	MAX
	timed τ-simulation		
Composability	OK	OK	nOK
Compatibility	OK	OK	nOK
Compositionality	OK	OK	nOK
	DS timed τ-simulation		
Composability	nOK	OK (hyp. div. 1)	nOK
Compatibility	nOK	OK	nOK
Compositionality	nOK	OK (hyp. div. 2)	nOK

Table 1. Properties of τ-simulations w.r.t. composition operators

	Classic parallel composition	Non-blocking parallel composition	
		AND / MIN	MAX
Properties preserved during integration of components	safety	Mitl, deadlock-freedom, strong non-zenoness (hyp. div. 1 and 2)	none

Table 2. Synthesis on the preservation of properties during integration of components

5 Related Works

Several works have been devoted to the definition of simulation relations for timed systems. A time-abstracting simulation has been studied in [13], but does not preserve timed properties. A timed simulation is defined in [14]. This relation has the properties of composability, compatibility and compositionality w.r.t. to a totally synchronous composition operator. However, non-observable actions are not considered and no criteria is added to take into account liveness.
The closest notion of simulation regarding to the one we defined is the timed ready simulation of [15]. Non-observable actions are also taken into account. The definition of the relation is almost the same than our timed τ-simulation and thus, preserves safety properties. But, it was not extended to preserve liveness properties. Composability, compatibility and compositionality are ensured w.r.t. a composition operator using a composition paradigm close to the one of the classic operator we consider. However, an assumption is made concerning the absence of internal activity in the automata to guarantee the properties. In particular, compositionality is expressed as follows. If $A \preceq B$ and $C \preceq D$, and B and D do not contain internal activity, then $A \| C \preceq B \| D$, where $\preceq$ is the timed ready simulation and $\|$ the composition operator considered.

6 Conclusion and Future works

In previous works, we defined τ-simulation relations for timed systems, with preservation abilities. Checking these relations is a way to guarantee the preservation of properties during incremental development of timed systems, in particular during integration of components. However, we wish to avoid the verification of the simulations, while still benefiting of their preservation abilities.
For this purpose, in this paper, we studied the properties of composability, compatibility and compositionality of the relations w.r.t. two composition operators for timed systems: the classic operator, and a non-blocking one with three different synchronization modes. It turns out that the properties hold for the timed τ-simulation with both operators (except when using the MAX synchronization mode with the non-blocking one). This means that the preservation of safety properties is ensured for free when using these operators for integration of components. The divergence-sensitive and stability-respecting timed τ-simulation has the properties only with the non-blocking operator (except with the MAX mode), on some conditions for divergence-sensitivity. Thus, MITL properties, deadlock-freedom and strong non-zenoness, are preserved (on the conditions expressed) during integration of components with this operator. This is not the case when using the classic operator. The reason is that this operator does not prevent from introducing deadlocks during composition, which makes the *stability-respecting* part of the simulation not guaranteed during integration of components.
Thus, in this case, the verification of the DS timed τ-simulation is necessary. In this perspective, we are working on reducing the cost in practice of the algorithmic verification of the simulation. The classic algorithm for the verification of the simulation consists in a joint depth-first exploration of the two models

to compare, i.e., the one on which verification has been done and the one on which preservation must be ensured. Stability-respect is checked at each step of the exploration (divergence-sensitivity consists in detecting non-zeno τ-cycles, thus it is checked independently with classic algorithms for cycles detection). The verification of this clause has a high cost in practice. Thus, we intend to verify it only on the paths on which preservation has to be ensured, i.e., the paths which concern the properties to be preserved. Of course, this partial check is done accordingly to the properties which must be preserved, and therefore only ensures the preservation of these properties. This partial verification has already been implemented in our tool VeSTA for response properties of the form $\Box(p \Rightarrow \Diamond q)$ (details can be found in [16]). As a future work, we intend to extend this partial verification to other patterns of properties.

References

1. Bellegarde, F., Julliand, J., Mountassir, H., Oudot, E.: On the contribution of a τ-simulation in the incremental modeling of timed systems. In: Proc. of FACS'05. Volume 160 of ENTCS., Macao, Macao, Elsevier (2005) 97–111
2. Alur, R., Feder, T., Henzinger, T.: The benefits of relaxing punctuality. Journal of the ACM **43** (1996) 116–146
3. Bellegarde, F., Julliand, J., Mountassir, H., Oudot, E.: Experiments in the use of τ-simulations for the components-verification of real-time systems. In: Proc. of SAVCBS'06, Portland, Oregon, USA (2006) Also available on ACM Digital Library.
4. Bornot, S., Sifakis, J.: An Algebraic Framework for Urgency. Information and Computation **163** (2000) 172–202
5. Bornot, S., Sifakis, J., Tripakis, S.: Modeling Urgency in Timed Systems. In: COMPOS'97. Volume 1536 of LNCS., Springer-Verlag (1997)
6. Hoare, C.: Communicating Sequential Processes. Prentice Hall (1985)
7. Milner, R.: Communication and Concurrency. Prentice Hall (1989)
8. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
9. Sifakis, J., Yovine, S.: Compositional Specification of Timed Systems. In: Proc. of STACS'96 - Invited Paper. Volume 1046 of LNCS. (1996) 347–359
10. Bellegarde, F., Julliand, J., Kouchnarenko, O.: Ready-simulation is not Ready to Express a Modular Refinement Relation. In: Proc. of FASE'00. Volume 1783 of LNCS., Berlin, Germany, Springer-Verlag (2000) 266–283
11. Pnueli, A.: The temporal semantics of concurrent programs. Theoretical Computer Science **13** (1981) 1–20
12. Tripakis, S.: The analysis of timed systems in practice. PhD thesis, Universite Joseph Fourier, Grenoble, France (1998)
13. Henzinger, M., Henzinger, T., Kopke, P.: Computing simulations on finite and infinite graphs. In: Proc. of FOCS'95. (1995) 453–462
14. Tasiran, S., Alur, R., Kurshan, R., Brayton, R.: Verifying Abstractions of Timed Systems. In: CONCUR'96. Volume 1119 of LNCS., Pisa, Italy (1996) 546–562
15. Jensen, H., Larsen, K., Skou, A.: Scaling up UPPAAL : Automatic verification of real-time systems using compositionnality and abstraction. In: Proc. of FTRTFT'00, London, UK, Springer-Verlag (2000) 19–30
16. Julliand, J., Mountassir, H., Oudot, E.: VeSTA : A tool to verify the correct integration of a component in a composite timed system. In: Proc. of ICFEM'07, Boca Raton, Florida, USA (2007) To appear.