2012 19th International Symposium on Temporal Representation and Reasoning

Compositional Refinement for Real-Time Systems with Priorities

Abdeldjalil Boudjadar, Jean-Paul Bodeveix, Mamoun Filali
IRIT-UPS, Université de Toulouse
Toulouse, France
{boudjada, bodeveix, filali}@irit.fr

***Abstract*—High-level requirements of real-time systems like as time constraints, communications and execution schedulability make the verification of real-time models arduous, where a system is the interaction of a possibly unbounded set of components. Priorities have been introduced to resolve execution conflicts, and by that, prevent the combinatorial explosion of state space. In this paper, we are interested in the composition and refinement of timed systems by considering static and dynamic priorities. Firstly, we propose a revised definition of the product of extended timed transition systems with static and dynamic priorities associated to individual transitions. Afterwards, we study the (compositional) refinement of compound extended timed systems. Without sacrificing compositionality, we instantiate this framework for the case of UPPAAL networks of timed automata with static priority *Committedness*, dynamic priority between channels and priority between processes. Moreover, we show how to associate an Extended Timed Transition System (ETTS) to timed automata (TA), where an `unique` generalized dynamic priority system of ETTS is derived from both dynamic priority orders: priority between channels and priority between processes.**

***Keywords*-Timed systems; composition; refinement; priorities.**

I. Introduction

The concurrency theory [25] is an extremely helpful concept whereby the design of complex systems becomes as far as tractable. It has by now established itself as an extensive research field for mastering the schedulability of executions. However, concurrency has a real impact on model checking of real-time systems, where conflicts and non-determinism of executions are greatly diverging together with the number of system (competing) processes, which often leads to combinatorial explosion of the state space, the reason for this being that time is considered as part of the state. To tackle non-determinism and execution conflicts, priorities have been introduced as a scheduling order. Composition, refinement and model-checking of timed systems with priorities have been intensively studied [9], [18], [22], [3], [17], [16], [13]. The ultimate goal of these work is to deal with details resulted from the use of clock variables and evolutive data structures, required by real-time applications. Abstraction refinement [19] plays a key role in the design and verification of real-time systems. It enables to abstract unbounded data structures and implementation details whereby, we are able to perform model checking on a system $S' = S'_1 \parallel ... \parallel S'_n$ instead of the original complex system $S = S_1 \parallel ... \parallel S_n$, where each S'_i is an abstraction of S_i. With the above fact, checking that S satisfies a property P becomes more tractable and simply consists of checking that each component S'_i satisfies a property P_i, where P is a composition of P_i. That is what happen in compositional verification [14] and abstraction-based verification [6]. Defining the parallel composition and finding a suitable refinement relation for real-time systems with communication, priorities and variables is a tough task. Real-time system properties are often formalized using time-constraints and priorities. To consider such concepts, we introduce real-time systems with a global space (variables and clocks) and priorities (static and dynamic). Similarly to process algebras, to deal with hierarchical design and specifications, real-time formalisms have known a large number of composition approaches. However, a compositional framework with high-level concepts like variables, communication and priorities is still lacking. Several works [11], [16], [22], [13] have focused on this subject by analyzing thoroughly the problem and criticizing existing solutions. In fact, this paper is a follow up of [12] where we have revisited the composition of timed systems, without priorities, and proposed a new communication mechanism for UPPAAL timed automata. In [12], we have defined an original composition operator endowed with good properties (associativity, refinement, etc.), and supporting communications via synchronization of actions and shared variables. Through the introduction of priority, we revisit the framework defined in [12] for reasoning about the composition of timed systems. Thereafter, our framework, defined with priorities, will be instantiated for UPPAAL timed automata with three priority orders : static priority, priority on channels and priority on processes, where we will analyze different priority relations, and give both operational semantics and refinement of timed automata TA and networks of timed automata NTA (in compositional way). The rest of our paper is organized as follows: Section 2 presents the existing related work. In Section 3, we present the formal basis of our work, where we introduce Communicating Labelled Transition Systems with location Invariants and Priorities (CLTSIP). We give a sufficient condition for the bisimilarity and define an associative product of CLTSIPs. Moreover, we define ETTSs as CLTSIPs which synchronize on time instants. Section 4 introduces UPPAAL TA and NTA with committed locations, priority on channels and priority on processes. Here, we show the compositionality of NTA semantics in terms of ETTSs, where the two corresponding ETTS-based semantics

1530-1311/12 $26.00 © 2012 IEEE
DOI 10.1109/TIME.2012.21

are equivalent: a direct one (an ETTS associated to the NTA) and the product of ETTSs, associated to individual timed automata, composed with a restriction. We also establish an interesting refinement property stating that: the refinement of a NTA consists in the refinements (separately) of its individual TA. Finally, we show an application of our framework to refine a version of the Alternating Bit Protocol [28], and conclude with future work.

II. Related Work

Composition and refinement of real-time systems with priorities have been intensively studied [18], [16], [13], [22], [9] during the last two decades. However, a compositional framework merging different priority systems is still relatively lacking. Several works of Sifakis [11], [18], [3], Cleaveland [9], [15] and UPPAAL team [16], [17] have focused on the modeling and synthesis of timed systems with priorities. They represent a common theoretical basis for the modeling with priorities.

The authors of [18] define a design framework for both safety and deadlock-freedom requirements. The framework consists of a priority system, where an action a_i has (dynamic) priority over another action a_j once a condition c_{ij} is satisfied together with the enabledness of actions. In the same way, [11] defines a dynamic priority on actions, where an action a_i has a priority on an action a_j during a certain time interval. In fact, such dynamic priority relations are a partial function because they are only applied under the satisfaction of an extra condition on comparable actions.

In [16], the authors define an extension of timed automata with a dynamic priority order between actions and another priority order between processes. They give an efficient algorithm to compute subtractions of DBMs (Difference Bounded Matrices). The authors define a non compositional semantics of networks of extended TA, in terms of timed transition systems. Under certain restrictions, they show how an unique generalized priority order can be derived from both action and process priority orders.

In [9], the authors describe a modeling framework for real-time systems, using dynamic priorities, which essentially extends CCS (Calculus of Communicating Systems) algebra [25] with dynamic priorities. Such a proposal reduces drastically the state space of systems and preserves their functional behavior. In fact, action priorities are not constant and may change when the system evolves. Formally, each action priority is inferred from delays preceding that action. Accordingly, the longer is the delay preceding an action, the lower is its priority.

In this paper, we present a compositional framework for the composition and refinement of timed systems with both static and dynamic priorities. To this end, we consider an extended structure of timed transition systems ETTS with variables, location invariants and communication where each transition possesses a static priority s and a dynamic one d. We define an associative parallel product of ETTSs together with a compositional refinement property. Moreover, we instantiate our framework for the case of UPPAAL networks of timed automata, with the static priority *committedness* and dynamic priority orders on channels and on processes, and establish their compositional semantics in terms of ETTSs.

III. Transition System Extensions

In this section, after introducing priorities, we give a brief recall of one of the fundamental models of concurrency, *transition systems*, originally introduced in [27] and since then studied extensively by [26] and others. Roughly speaking, transition systems are an elegant model for representing the behavioral aspects. Due to their safety properties verifiable using model-checking, transition systems have been intensively applied to the modelling of complex systems, as well as for giving semantics to synchronous languages and real-time formalisms. In what follows, we define composable symbolic transition systems (CLTSIP) as a modeling framework. Thereafter, we give their associative product and study the refinement of their parallel composition.

A. Priority Systems

Priorities [9], [15], [18], [3], [22], [13] have been introduced as a way to structure and control the usage of shared resources, by specifying that some actions or behavior are privileged over others. They offer a scheduling order to deal with non-determinism and execution conflicts. The BIP language [3] and ACSR algebra [13] provide a powerful mechanism to express different sort of priorities. Mainly, we distinguish static and dynamic priorities:

- Static priorities [12], [7], [4] define an order between transition executions regardless of their enabledness. With such priorities, non-enabled higher-priority transitions hide enabled lower-priority transitions, which often leads to a deadlock. In UPPAAL timed automata [4], the static priority is represented by the notion of *Committedness* (two priority levels) where committed transitions [7], those outgoing from a committed location, have priority over non-committed ones.
- Dynamic priorities [22], [18], [9], [16] state that an *enabled* transition hides a lower-priority transition. [18] introduces a conditional dynamic priority relation, where an enabled transition hides a lower-priority one if a given condition holds. Another class of priority relations [11] consists to restrict the applicability of priority to a given time interval. The semantics of priority relations [11], [18], defined over timed systems, is given by a model transformation where only dynamically higher-priority transitions are held.

Definition 1 (Priority system): A priority system $\mathcal{P}$ is a triplet $\langle P, \preceq, \triangle \rangle$ where $\langle P, \preceq \rangle$ is a join semi-lattice, and $\triangle : P \times P \to P$ is an associative and commutative operator defining the maximum of two values. We also use $\triangle_i \ p_i$ to represent the maximum of a finite non empty set of values.
In fact, the join semi-lattice $\langle P, \preceq \rangle$ represents a partially ordered set of priority values, where each subset of P has a least upper bound.

B. Labelled Transition Systems

Labelled transition systems [2] are the reference model used to express and compare behaviors through simulations. They offer a strong notion of equivalence that can be checked efficiently. Firstly, let us start with a brief recall of classical labelled transition systems (LTS).

Definition 2 (LTS): A labelled transition system (LTS) over an alphabet Σ is a tuple $\langle Q, Q^0, Tr, \alpha, \beta, \lambda \rangle$ where:

- Q is the state space such that $Q^0 \subseteq Q$ is the set of initial states,
- Tr is the set of transitions,
- $\alpha : Tr \to Q$ and $\beta : Tr \to Q$ are functions associating, respectively, source and target states to each transition,
- $\lambda : Tr \to \Sigma$ is a function associating to each transition a label.

For the sake of simplicity, we write $t : q \xrightarrow{l} q'$ for $t \in Tr$ with $\alpha(t) = q,\ \beta(t) = q',\ \lambda(t) = l$. If useless, the name of a transition will be omitted. Moreover, projection functions α, β and λ can be omitted and systematically inferred from the transition relation $\to$. By now, we give LTSs simulation to check that a concrete LTS implements an abstract one.

Definition 3 (Simulation): Given two labelled transition systems $\mathcal{T}_c = \langle Q_c, Q_c^0, Tr_c \rangle$ (concrete) and $\mathcal{T}_a = \langle Q_a, Q_a^0, Tr_a \rangle$ (abstract), $\mathcal{T}_a$ simulates $\mathcal{T}_c$ through a relation $R \subseteq Q_c \times Q_a$, denoted by $\mathcal{T}_c \sqsubseteq_R \mathcal{T}_a$, if:

- $\forall q_c \in Q_c^0$, there exists $q_a \in Q_a^0$ such that $R(q_c, q_a)$.
- $\forall q_c\ q_c'\ q_a\ l$, if $q_c \xrightarrow{l}_c q_c'$ and $R(q_c, q_a)$ there exists $q_a' \in Q_a$ such that $q_a \xrightarrow{l}_a q_a'$ and $R(q_c', q_a')$.

Accordingly, two LTSs $\mathcal{T}_i$ and $\mathcal{T}_j$ are bisimilar through a relation $R \subseteq Q_i \times Q_j$, denoted by $\mathcal{T}_i \sim_R \mathcal{T}_j$, if $\mathcal{T}_i \sqsubseteq_R \mathcal{T}_j$ and $\mathcal{T}_j \sqsubseteq_{R^{-1}} \mathcal{T}_i$.

C. Composable LTS with Location Invariants and Priorities

In this section, we define composable LTS with location invariants and priorities (CLTSIPs), as an extension of labelled transition systems with shared variables, communication, static priority, dynamic priority and location invariants. Moreover, we specialize both state space and alphabet to allow several communication protocols between transition systems:

- via a shared space: we distinguish local and global state spaces, and introduce abstract actions that update the global state space. These actions can be non-deterministic and blocking.
- via CCS-like channels [25]: we introduce a set C of send-receive channels, where two transitions synchronize if their actions are complementary. The resulting transition, of such a synchronization, corresponds to an internal transition in the composition.
- via CSP (Communicating Sequential Processes)-like synchronization [21]: we introduce a set M of many-to-many synchronization events, which enable to model a system transition where all processes perform a lock-step [21]. Throughout this paper, we use such a synchronization to model time-evolution transitions.

Furthermore, to reduce the non-determinism and execution conflicts, we consider the following priority mechanisms.

- Static priority expresses that a higher-priority transition hides a lower-priority one. The hiding is supposed to be static: a non-firable high-priority transition can hide a firable lower-priority transition.
- Dynamic priority states that an *enabled* priority transition hides lower-priority ones. An enabled transition is firable if it is not hidden by another higher-priority *enabled* transition.

Throughout this paper, the static priority is considered first. We have also introduced location invariants over the global space to restrict the set of states, by reducing the global space valuations.

Definition 4 (CLTSIP): Given two priority systems $\mathcal{P}_s = \langle P_s, \preceq_{\mathrm{s}}, \triangle^{\mathrm{s}} \rangle$ (static) and $\mathcal{P}_d = \langle P_d, \preceq_{\mathrm{d}}, \triangle^{\mathrm{d}} \rangle$ (dynamic), a composable LTS with location invariants and priorities (CLTSIP) over a shared space $\mathcal{G}$, an action language $\mathcal{A}$ [1], a set of one-to-one channels C and a set of synchronization events M is a tuple $\langle Q, q^0, G^0, I, Tr, \alpha, \beta, \lambda \rangle$ where:

- Q is the set of locations (local states),
- q^0 is the initial location,
- G^0 is the set of initial global states,
- $I : Q \to 2^{\mathcal{G}}$ associates an invariant to each location,
- Tr is the set of transitions,
- $\alpha : Tr \to Q$ and $\beta : Tr \to Q$ are functions associating, respectively, the source and target locations of each transition,
- $\lambda : Tr \to \mathcal{L} \times \mathcal{A} \times P_s \times P_d$ is a function associating to each transition the corresponding label, action, static priority value and dynamic one, where $\mathcal{L} = C? \cup C! \cup M \cup \{\tau\}$ is the set of labels. $C!$ and $C?$ correspond respectively to send and receive labels over channels C. M is the set of multiple (many to many) synchronization events and τ is the internal event.

Moreover, a CLTSIP must satisfy the wellformedness condition: n-ary synchronization transitions, with a label in M, are supposed to have the lowest static and dynamic priorities.

Here and elsewhere, we write $t : q \xrightarrow{l/a}_{s,d} q'$ for a transition $t \in Tr$ with $\alpha(t) = q,\ \beta(t) = q',\ \lambda(t) = \langle l, a, s, d \rangle$. If not needed, the name of a transition will be omitted. Again, the set of transitions Tr is often denoted by the transition relation $\to$, which enables to omit the projection functions $\alpha,\ \beta$ and λ. The semantics $[\![.]\!]$ of the action language $\mathcal{A}$ is given by $[\![.]\!] : \mathcal{A} \to 2^{\mathcal{G} \times \mathcal{G}}$. Let us consider the following predicates:

- A transition $t : q \xrightarrow{l/a}_{s,d} q'$ is said to be *enabled* in a global state G if $\exists G' \mid (G, G') \in [\![a]\!]$ and $G' \models I(q')$.
- A transition $t : q \xrightarrow{l/a}_{s,d} q'$ is said to be *statically hidden* if $\exists t' : q \xrightarrow{l'/a'}_{s',d'} q''$ such that $s \prec_{\mathrm{s}} s'$.
- A transition $t : q \xrightarrow{l/a}_{s,d} q'$ is said to be *dynamically hidden* if $\exists t' : q \xrightarrow{l'/a'}_{s',d'} q''$ enabled and non statically hidden such that $d \prec_{\mathrm{d}} d'$.

Accordingly, a transition is said to be *hidden* if it is statically and dynamically hidden.

[1] This action language is abstract here. It will be made more precise in section IV-B

Definition 5 (Semantics of a CLTSIP): Given a global space $\mathcal{G}$, a static priority system $\mathcal{P}_s = \langle P_s, \preceq_{\mathrm{s}}, \triangle^{\mathrm{s}} \rangle$, a dynamic priority system $\mathcal{P}_d = \langle P_d, \preceq_{\mathrm{d}}, \triangle^{\mathrm{d}} \rangle$ and an action language semantics $[\![.]\!] : \mathcal{A} \rightarrow 2^{\mathcal{G} \times \mathcal{G}}$. The semantics of the CLTSIP $\langle Q, q^0, G^0, I, Tr \rangle$ is the LTS:

- $Q \times \mathcal{G}$,
- $\{q^0\} \times (G^0 \cap I(q^0))$,
- $\{(q, G), l, (q', G') \mid \exists t : q \xrightarrow{l/a}_{s,d} q' \in Tr, G \models I(q), enabled(t, G),\ \neg statically\_hidden(t)$ and $\neg dynamically\_hidden(t)\}$

In fact, LTS states correspond to the product of both locations and space valuations of the CLTSIP. Namely, an enabled transition t, of CLTSIP, is held if it is not statically hidden by a higher-priority transition t', i.e. $s \not\prec_{\mathrm{s}} s'$, and if it is not again dynamically hidden by another enabled non-hidden transition, i.e $enabled(t', G) \wedge \neg statically\_hidden(t') \Rightarrow d \not\prec_{\mathrm{d}} d'$.

Definition 6 (Similarity): A CLTSIP $\mathcal{T}_i$ is said to be (bi)similar to CLTSIP $\mathcal{T}_j$ if their associated LTSs are (bi)similar.

The presence of both static and dynamic priorities makes the semantics of CLTSIPs rather complex. It would be much more readable if we could get rid of managing priorities during simulation proofs. To this end, we consider a sufficient condition for the refinement of CLTSIPs, expressed as the simulation of the corresponding LTSs. Firstly, we introduce the predicate $Ismax\_t_s(t : q \xrightarrow{l/a}_{s,d} q') \triangleq \forall t' : q \xrightarrow{l'/a'}_{s',d'} q''\ \neg(s \prec_{\mathrm{s}} s')$ defining a higher static priority transition outgoing from location q. In the same way, $Ismax\_t_d(t : q \xrightarrow{l/a}_{s,d} q') \triangleq \forall t' : q \xrightarrow{l'/a'}_{s',d'} q''\ \neg((d \prec_{\mathrm{d}} d') \wedge enabled(t', G))$, with $(G, G') \in [\![a']\!]$, is a predicate defining a higher dynamic priority transition outgoing from location q.

Theorem 1 (Refinement of CLTSIPs): Given two CLTSIPs $\mathcal{T}_1$ and $\mathcal{T}_2$ with their respective static and dynamic priority systems $(\mathcal{P}_s^1, \mathcal{P}_d^1)$ and $(\mathcal{P}_s^2, \mathcal{P}_d^2)$. $\mathcal{T}_1$ refines $\mathcal{T}_2$ through the refinement relations $R_l \subseteq Q_1 \times Q_2$ and $R_g \subseteq \mathcal{G}_1 \times \mathcal{G}_2$, denoted $\mathcal{T}_1 \sqsubseteq_{R_l, R_g} \mathcal{T}_2$, if

1) The associated LTSs satisfy the sufficient condition for simulation, i.e:
   - $R_l(q_1^0, q_2^0)$,
   - $\forall x \in G_1^0,\ \exists y \in G_2^0 \mid R_g(x, y)$,
   - $\forall t : q_1 \xrightarrow{l/a_1}_{s_1,d_1} q_1',\ \forall q_2\ x\ x'\ y$ such that $(x, x') \in [\![a_1]\!]$. If $R_l(q_1, q_2)$ and $R_g(x, y)$ then there exist $q_2' \in Q_2,\ a_2, y', s_2 \in P_s^2, d_2 \in P_d^2$ such that $q_2 \xrightarrow{l/a_2}_{s_2,d_2} q_2' \wedge (y, y') \in [\![a_2]\!] \wedge R_l(q_1', q_2') \wedge R_g(x', y')$,
   - $\forall q_1 \in Q_1\ q_2 \in Q_2, R_l(q_1, q_2) \wedge I_2(q_2) \Rightarrow I_1(q_1)$,
2) For all $t_1 \in Tr_1$ and $t_2 \in Tr_2$ such that $R_l(\alpha_1(t_1), \alpha_2(t_2))$ and $R_l(\beta_1(t_1), \beta_2(t_2))$, then $Ismax\_t_s(t_1) \Rightarrow Ismax\_t_s(t_2)$, $Ismax\_t_d(t_1) \Rightarrow Ismax\_t_d(t_2)$.

Roughly speaking, the refinement consists of establishing a mapping between the transitions of refining and refined CLTSIPs. In fact, from each location q of the concrete CLTSIP $\mathcal{T}_1$, the presence of a transition t_1, with a maximal priority, states the presence of a maximal-priority transition outgoing from a location corresponding to q in the abstract CLTSIP $\mathcal{T}_2$. The universal quantifier given in Item (2) dissociates the condition on priorities from that of refinement and makes, by that, the proofs further simpler than that of the LTS-based refinement.

Proof. Straightforward.

1) Restriction of a CLTSIP: The restriction [25] of a CLTSIP, over a set of channels, is a CLTSIP where transitions composable over these channels have been removed together with transitions of lower static priority.

Definition 7 (CLTSIPs restriction): Given a CLTSIP $\mathcal{T} = \langle Q, q^0, G^0, I, \longrightarrow \rangle$ over a shared space $\mathcal{G}$ and a set of channels C. Let $C' \subseteq C$, we define the restriction of $\mathcal{T}$ over C', denoted by $\mathcal{T}\backslash C'$, to be the CLTSIP $\langle Q, q^0, G^0, I, \{t : q \xrightarrow{l/a}_{s,d} q' \mid l \notin C' \ \wedge\ \forall t' : q \xrightarrow{l'/a'}_{s',d'} q'',\ l' \in C' \Rightarrow s' \not\succ_{\mathrm{s}} s\}\rangle$.

In fact, from each location, a transition is held if it is neither labelled by a communication $l \in C'$, nor statically hidden by another communicating transition (over C') outgoing from that location.

Theorem 2 (Refinement and restriction): Let $\mathcal{T}_c, \mathcal{T}_a$ be two CLTSIPs defined on the same set of channels C, then $\mathcal{T}_c \sqsubseteq \mathcal{T}_a \Rightarrow \mathcal{T}_c\backslash C \sqsubseteq \mathcal{T}_a\backslash C$.

Proof. It consists to show that each non-hidden concrete transition of $\mathcal{T}_c\backslash C$, labelled by τ or $m \in M$, has a corresponding abstract transition non-hidden in $\mathcal{T}_a\backslash C$.

2) Product of CLTSIPs: In what follows, we define an associative n-ary product of CLTSIPs, where locations of composition are simply obtained by the product of individual CLTSIP locations. Moreover, our product is parameterized by two internal operations defined on the action language:

- $a_1 \triangleright a_2$ is used to compose actions associated to send-receive communication.
- $a_1 \odot a_2$ is used to compose actions associated to global synchronizations (lock-step). This operation is supposed to be commutative, respectively associative, in order to establish the commutativity, respectively associativity, of the product.

Definition 8 (N-ary product of a family of CLTSIPs): Given an indexed family $\mathcal{T}_i = \langle Q_i, q_i^0, G_i^0, I_i, \rightarrow_i \rangle_{1..n}$ of n CLTSIPs defined over the same shared space $\mathcal{G}$, action language $\mathcal{A}$, static priority system $\mathcal{P}_s$ and dynamic priority system $\mathcal{P}_d$, their product $\Pi_{1..n}\mathcal{T}_i$ is defined by the CLTSIP $\langle \bigotimes_i Q_i, \langle q_1^0, \ldots, q_n^0 \rangle, \bigcap_i G_i^0, I, \rightarrow \rangle$ over $\mathcal{G}$, $\mathcal{P}_s$ and $\mathcal{P}_d$ where $I(q) = \bigcap_i I_i(q_i)$ and $\rightarrow$ is the smallest relation such that:

$$\frac{t_i : q_i \xrightarrow{l/a}_{s,d} q_i' \quad l \in C! \cup C? \cup \{\tau\}}{q \xrightarrow{l/a}_{s,d} q[i \leftarrow q_i']}\ \mathrm{Async}(t_i)$$

$$\frac{(\forall i)\ q_i \xrightarrow{m/a_i}_{s_i,d_i} q_i'\ m \in M}{q \xrightarrow{m/\bigodot_i a_i}_{\triangle_i^{\mathrm{s}} s_i, \triangle_i^{\mathrm{d}} d} q'}\ \mathrm{Sync}$$

$$\frac{t_i : q_i \xrightarrow{c!/a_i}_{s_i,d_i} q_i' \quad t_j : q_j \xrightarrow{c?/a_j}_{s_j,d_j} q_j' \quad (PC) \quad i \neq j}{q \xrightarrow{\tau/a_i \triangleright a_j}_{s_i \triangle^{\mathrm{s}} s_j, d_i \triangle^{\mathrm{d}} d_j} q[i \leftarrow q_i', j \leftarrow q_j']}\ \mathrm{SR}(t_i, t_j)$$

where (PC) is a priority condition stating that if static, respectively dynamic, priorities of transitions t_i and t_j are increased upto the maximum $s_i \triangle^{\mathrm{s}} s_j$, respectively $d_i \triangle^{\mathrm{d}} d_j$, no new hiding may occur. For example, the static priority condition for CLTSIP $\mathcal{T}_i$ can be formally expressed as $\nexists t : q_i \xrightarrow{l/a}_{s,d} q_i'' \mid (s \succ_{\mathrm{s}} s_i) \wedge (s \not\succ_{\mathrm{s}} s_i \triangle^{\mathrm{s}} s_j)$.

The notation $q[i \leftarrow q_i']$ states the replacement of the ith location of vector q by location q_i'. If we consider UPPAAL TA, in which transition priorities are assigned to channels, then $d_i = d_j = d_i \triangle^{\mathrm{d}} d_j$. About transition rules, $Async(t_i)$ represents internal transitions and potential synchronizations that a CLTSIP may be willing to engage in with its environment. Rule $Sync$ defines a n-ary synchronization of a set of transitions on the same event m, which will be instantiated by a time-transition in the ETTS. Rule $SR(t_i, t_j)$, for send/receive, corresponds to a synchronized communication of both $\mathcal{T}_i$ and $\mathcal{T}_j$ on compatible events through a channel $c \in C$. Let us mention that n-ary synchronization transitions, labelled by m, cannot block or be blocked. One may remark that our product is syntactical, whereby, all of the CLTSIP non-composable transitions are held.

Theorem 3 (Generalized associativity): If $\odot$ is associative, i.e. $\odot_{i\in\mathcal{I}} \odot_{j\in\mathcal{J}_i} a_{i,j} = \odot_{i\in\mathcal{I},j\in\mathcal{J}_i} a_{i,j}$, the product of CLTSIPs is associative, i.e.: $\Pi_{i\in\mathcal{I}}(\Pi_{j\in\mathcal{J}_i}\mathcal{T}_{i,j}) \sim \Pi_{i\in\mathcal{I},j\in\mathcal{J}_i}\mathcal{T}_{i,j}$

Proof. It essentially consists of defining an isomorphism between the two structures, state space and transitions, preserving labels and priorities. □

3) Compositional Refinement of CLTSIPs Product: Model-checking of real-time systems suffers from the state explosion problem, the reason for this being that time is considered as part of the state, leading then to a widely large or even infinite state space of the system. Abstraction refinement plays a key role in the model-checking of complex systems where unbounded data structures can be abstracted. However, for compound systems, defining the refinement of the whole system is an arduous task. In what follows, we show how the (compositional) refinement [19] of CLTSIPs product has been brought to a set of simpler refinements of individual CLTSIPs.

Theorem 4 (Compositional refinement): Given two products of CLTSIPs $\mathcal{T}_1 \parallel ... \parallel \mathcal{T}_n$ and $\mathcal{T}_1' \parallel ... \parallel \mathcal{T}_n'$ defined on the same priority systems (with total orders). $\mathcal{T}_1' \parallel ... \parallel \mathcal{T}_n'$ refines $\mathcal{T}_1 \parallel ... \parallel \mathcal{T}_n$, denoted by $\mathcal{T}_1' \parallel ... \parallel \mathcal{T}_n' \sqsubseteq_{\otimes_i R_l^i, R_g} \mathcal{T}_1 \parallel ... \parallel \mathcal{T}_n$ where $R_l^i \otimes R_l^j(q_i, q_j)(q_i', q_j') = R_l^i(q_i, q_i') \wedge R_l^j(q_j, q_j')$, if:

- $\forall i\ \mathcal{T}_i' \sqsubseteq_{R_l^i, R_g} \mathcal{T}_i$,
- each concrete transition and its corresponding abstract one have the same priorities (morphism),
- refinement preserves deadlock-freeness, where deadlock-freeness is defined by the existence of firable transitions.

Through this theorem, we are able to perform model checking on the composition of CLTSIPs, $\mathcal{T}_1' \parallel ... \parallel \mathcal{T}_n'$, instead of the original composition $\mathcal{T}_1 \parallel ... \parallel \mathcal{T}_n$. In fact, through the last two conditions of this theorem, we may reduce the refinement of CLTSIPs to classical refinement relations of transition systems, and by that, the proof would be much more tractable (note that we do not claim that composition preserves deadlock freeness).

Proof. Given a transition of the concrete product, it is either asynchronous from some $\mathcal{T}_i'$ and has a corresponding abstract transition in $\mathcal{T}_i$ with the same priority, which is in turn present in the abstract product, or a synchronization SR of two transitions from $\mathcal{T}_i'$ and $\mathcal{T}_j'$ that have abstractions in $\mathcal{T}_i$ and $\mathcal{T}_j$ which synchronize in the abstract product with the same priority, or again an n-ary synchronization $Sync$ which, as previously stated, leads to an n-ary synchronization in the abstract product. Moreover, if a concrete transition has priority over ready [2] transitions, then the corresponding abstract one has also priority. □

D. Timed Transition System Extensions

Timed transition systems [20] are the reference model to define the semantics of real-time formalisms such as time Petri nets and timed automata. Basically, a Timed Transition System (TTS) is a labeled transition system where labels can be events or durations. In this section, we define extended timed transition systems (ETTS) as CLTSIPs which synchronize on time. ETTS actions are considered as a pair (*guard, assignment*) of CLTSIPs. Furthermore, we consider the global state space structured as valued variables.

Definition 9 (ETTS): An Extended Timed Transition System (ETTS) on a set of variables $\mathcal{V}$ valued over a domain $\mathcal{D}$, a static priority system $\mathcal{P}_s$, a dynamic priority system $\mathcal{P}_d$ and a set of channels C is a CLTSIP over the global space $\mathcal{G} = \mathcal{D}^{\mathcal{V}}$ where the synchronization events $m \in M$ are time instants of $\Delta = \mathbb{R}_{\geq 0}$. Its action language is defined as the set of pairs (guard, assignment), where a guard is a predicate over variables of $\mathcal{V}$ and an assignment is a partial function mapping variables to expressions built on $\mathcal{V}$.

The semantics of an ETTS depends on its action language semantics. Here, we have chosen the following definition for the (guard, assignment) pairs:

Action	language	Semantics
a:=	$g/\ \parallel_{v\in V} v := e_v$	$[\![g/\ \parallel_{v\in V} v := e_v]\!](x, x') = g(x) \wedge \bigwedge_{v\in V} x'(v) = [\![e_v]\!](x)$
	$\mid a \triangleright a$	
	$\mid \odot_{j\in J} a_j$	

The notation $g/\ \parallel_{v\in V}$ states the parallel update of variables of V as an assignment guarded by g. Both action composition operators $\odot$ and $\triangleright$ are left undefined. Their semantics will be chosen to conform with the semantics of timed automata action composition.

IV. Instantiation for UPPAAL

UPPAAL [4] is an integrated tool environment for modeling, validation and model checking of real-time systems modeled as networks of timed automata. The tool has been used successfully and routinely for many industrial case studies. In this section, we consider UPPAAL timed automata (TA) [4] as an instantiation of CLTSIPs. The UPPAAL language [24] considers 3 priority orders: a static binary priority so-called *Committedness* associated to locations, a dynamic priority order on channels and another dynamic priority order between

[2] Transitions outgoing from the same state and, for dynamic priority, enabled.

processes. In fact, location committedness is a high level mechanism defining two priority levels, where transitions outgoing from committed locations have priority over transitions outgoing from non-committed locations, independently of their enabledness.

- In UPPAAL [4], committedness is associated to states where systems cannot delay if the current state is committed.
- In order to define a compositional semantics of timed automata composition, using a products of TTSs (Timed Transition Systems), [7] proposes a restriction on UPPAAL so that a committed state has always a firable outgoing transition.
- With respect to our previous work [10], we do not require such a restriction but we use a slightly modified structure of TTSs. Location committedness is considered as a static priority system with two priority values $\{true, false\}$.

Moreover, the dynamic priority order on channels [11], [18], [16] states that a synchronizing transition t on a channel c, which has priority over a channel c', has priority over transitions composable on c' if it is firable, i.e. the guard of t is satisfied. We also consider the priority order on TA processes, which is a dynamic relation stating that the executions of a timed automaton have priority over the executions of other TA. In what follows, we introduce UPPAAL TA with the three priority orders (committedness, priority on channels and priority on TA) and define their ETTS-based semantics, where an `unique` generalized dynamic priority system is derived from both priority on channels and priority on TA. To this end, on a composition, static priority is checked first and if the conflict is not solved we may refer to the priority order on channels. Again, if the choice of a transition from a conflict cannot be made, we compare then the dynamic priorities associated to the involved TA.

A. TA with Committed Locations and Priorities

Timed automata [1] have been introduced as a modeling framework to support the description and analysis of timed systems. In fact, a timed automaton is structured as a finite-state machine extended with a finite collection of real-valued clocks initialized to zero and increased synchronously.

Definition 10 (TA with committedness and priorities): Given a set of clocks χ, a set of channels C, a priority order $\preceq_c$ on channels of C and a priority order $\preceq_{ta}$ on TA, a timed automaton with committedness and priorities is a tuple $\langle Q, q^0, K, I, \rightarrow\rangle$ where:

- Q is a set of locations where $q^0 \in Q$ is the initial location,
- $K \subseteq Q$ is a set of committed locations,
- $I : Q \rightarrow 2^{\chi \rightarrow \Delta}$ associates a clock invariant (a set of clock valuations) to each location,
- $\rightarrow \subseteq Q \times 2^{\chi \rightarrow \Delta} \times 2^{\chi} \times \Sigma \times Q$ is the transition relation defined with a clock guard, a reset set and an event $l \in \Sigma$. $\Sigma = C? \cup C! \cup \{\tau\}$ is the set of transition labels.

We write $q \xrightarrow{g/l/r} q'$ for $(q, g, r, l, q') \in \rightarrow$. Different semantics of TA with priorities in terms of TTS have been proposed [19], [23], [29], [5]. Here, we define the semantics of TA with priorities in terms of ETTSs where committedness is an instance of the static priority system with two values. Moreover, the unique dynamic priority system of ETTS semantics is derived from a merge of both priority orders $\preceq_c$ and $\preceq_{ta}$.

Definition 11 (Priority systems corresponding to TA): Given a timed automaton $\mathcal{T} \in \mathtt{TA}$ defined on a total priority order $\preceq_c$ on channels and a total priority order $\preceq_{ta}$ on elements of `TA`, where `TA` is the set of timed automata names, the static priority system associated to $\mathcal{T}$ is defined by $\mathcal{P}_s = \langle \{\bot, \top\}, \Rightarrow, \bigvee \rangle$ and the dynamic one is defined by $\mathcal{P}_d = \langle (\{\epsilon, \mathtt{default}\} \cup C) \times \mathtt{TA}, \preceq_d, \triangle^{\mathtt{d}} \rangle$ where $(x, y) \preceq_{\mathtt{d}} (x', y') \triangleq x \prec_c x' \vee (x = x' \wedge y \preceq_{ta} y')$.
Since the orders $\preceq_c$ and $\preceq_{ta}$ are total in UPPAAL, the priority order $\preceq_{\mathtt{d}}$ is total. `default` is the UPPAAL priority level assigned to τ-transitions and ϵ is the lowest priority level. In fact, the static priority system is straightforward, whereas the dynamic priority system consists in checking first the priority on channels and if the choice of a transition cannot be made, we refer then to priorities of the corresponding TA. By now, we give the semantics of TA in terms of ETTSs where ETTS locations are TA locations.

Definition 12 (ETTS of a TA): Given a set of channels C, a priority order $\preceq_c$ on channels, a priority order $\preceq_{ta}$ on TA and a set χ of clocks. The semantics of a timed automaton with committed locations and priorities $\mathcal{T} = \langle Q, q^0, K, I, \rightarrow_{ta}\rangle$ is defined by the ETTS $\langle Q, q^0, G^0, I, \rightarrow\rangle$ over the global space $\chi \rightarrow \Delta$, static priority system $\mathcal{P}_s$ and dynamic priority system $\mathcal{P}_d$ where $G^0 = \chi \times \{0\}$ and $\rightarrow$ is the smallest relation such that:

$$\frac{q \xrightarrow{g/\tau/r}_{ta} q'}{q \xrightarrow{g/\tau/\|_{x\in r} x:=0}_{q\in K, (\mathtt{default}, \mathcal{T})} q'} \text{ Tau}$$

$$\frac{q \xrightarrow{g/l/r}_{ta} q' \quad l \in C}{q \xrightarrow{g/l/\|_{x\in r} x:=0}_{q\in K, (l, \mathcal{T})} q'} \text{ Com}$$

$$\frac{q \in K}{q \xrightarrow{\bot/\tau/skip}_{\top, (\mathtt{default}, \mathcal{T})} q} \text{ Empty}$$

$$\frac{q \notin K}{q \xrightarrow{[\|_{x\in\chi} x:=x+\delta] I(q)/\delta/\|_{x\in\chi} x:=x+\delta}_{\bot, (\epsilon, \mathcal{T})} q} \text{ Time}$$

For transition rules Tau and Com, both guards and labels of $\mathcal{T}$ transitions are still unchanged in the corresponding ETTS transitions. The semantics of a reset r consists in a parallel reinitialization of clocks $x \in r$. Moreover, the static priority of each ETTS transition corresponds to the committedness of the TA source location q, whereas the dynamic priority is the pair of label l for a communication, respectively `default` for internal and *Empty* transitions and ϵ for *Time* transitions, and the name of the automaton $\mathcal{T}$. *Empty* transitions are not firable and especially introduced to hold the committedness of TA locations, when that locations do not have outgoing transitions. From each non committed location q, we may perform a *Time*-transition adding an amount δ, respecting the invariant of q, to each clock $x \in \chi$. One may distinguish that no transition can be blocked by *Time*-transitions because they have the lowest static and dynamic priority values.

Simulation: a timed automaton $\mathcal{T}_c$ refines another TA

$\mathcal{T}_a$ if the simulation relation holds between their associated ETTSs: $\mathcal{T}_c \sqsubseteq \mathcal{T}_a \triangleq ETTS(\mathcal{T}_c) \sqsubseteq ETTS(\mathcal{T}_a)$.

B. Networks of TA with Committedness and Priorities

In order to model concurrency and communication, TA have been extended with parallel compositions, giving rise to the NTA. Several semantics for TA composition have been studied [4], [7], [16], [23], [29] and various parallel composition operators have been proposed, the well known ones are those of CCS [25] and CSP [21]. The UPPAAL language [24] has adopted the CCS parallel composition which allows interleaving of actions as well as hand-shake synchronization. In this section, to compare NTA through simulation and bisimulation relations, we define their semantics in terms of ETTS and establish a compositionality result.

Definition 13 (Networks of timed automata): A network of timed automata with committed locations, priority on channels $\preceq_c$ and priority on TA $\preceq_{ta}$ is a finite collection of TA.
First, let us choose the following action language and its underlying semantics for ETTS.

Action language
a:= a $\triangleright$ a $\mid \odot_{j\in J}\, a_j$ $\mid (g/r)$ $\mid (g/c := c+d)$ $\mid (g/skip)$
Semantics
a $\triangleright$ a is still unspecified $[\![\odot_j a_j]\!](x,x') = \bigwedge_j [\![a_j]\!](x,x')$ $[\![g/r]\!](x,x') = g(x) \wedge \bigwedge_{c\in r} x'(c) = 0 \wedge \bigwedge_{c\notin r} x'(c) = x(c)$ $[\![g/c := c+d]\!](x,x') = g(x) \wedge x'(c) = x(c) + d$ $[\![g/skip]\!](x,x') = g(x) \wedge (x' = x)$

The semantics of NTA is parameterized by the way guarded actions are composed on a send/receive synchronization, i.e. $(g_s/a_s) \triangleright (g_r/a_r)$. The semantics $[\![a \triangleright a]\!]$, depends on the semantics chosen for TA composition, is still unspecified.

Definition 14 (NTA semantics): Given a set of clocks χ, a set of channels C, a priority order $\preceq_c$ on C and a priority order $\preceq_{ta}$ on timed automata, the semantics of an NTA $\mathcal{N} = \langle Q_i, q_i^0, K_i, I_i, \rightarrow_i\rangle_{1..n}$ is defined by the ETTS $\langle \bigotimes_i Q_i, \langle q_1^0, \ldots, q_n^0\rangle, \chi \times \{0\}, I, \rightarrow\ \rangle$ over the global space $\chi \rightarrow \Delta$, static and dynamic priority systems $\mathcal{P}_s$ and $\mathcal{P}_d$ given in *Definition 11* where $I(q) = \bigcap_i I_i(q_i)$ and $\rightarrow$ is the smallest relation such that:

$$\frac{q_i \xrightarrow{g/\tau/r}_i q_i' \quad \bigvee_j q_j \in K_j \Rightarrow q_i \in K_i}{q \xrightarrow{g/\tau/\|_{x\in r} x:=0}_{q_i\in K_i,(default,\mathcal{T}_i)} q[i \leftarrow q_i']}\ \text{Tau}_i$$

$$\frac{\begin{array}{c} q_i \xrightarrow{g_i/c!/r_i}_i q_i' \quad q_j \xrightarrow{g_j/c?/r_j}_j q_j' \quad i \neq j \\ g/r = g_i/r_i \triangleright g_j/r_j \\ \bigvee_k q_k \in K_k \Rightarrow (q_i \in K_i \vee q_j \in K_j) \end{array}}{q \xrightarrow{g/\tau/r}_{q_i\in K_i \vee q_j \in K_j,(c,\max(\mathcal{T}_i,\mathcal{T}_j))} q'}\ \text{SR}_{i,j}(c)$$

$$\frac{q_i \in K_i}{q \xrightarrow{\bot/\tau/skip}_{\top,(default,\mathcal{T}_i)} q}\ \text{Empty}$$

$$\frac{\bigwedge_i q_i \notin K_i}{q \xrightarrow{g/\delta/\odot_i\, x_i:=x_i+\delta}_{\bot,(\epsilon,\max(\{\mathcal{T}_i|i\in 1..n\})} q}\ \text{Time}$$

where, for rule *SR*$_{i,j}(c)$, $q' = q[i \leftarrow q_i', j \leftarrow q_j']$ and for *Time* transitions, the guard $g = [\odot_i\, x_i := x_i + \delta]I(q)$ states that the location invariant $I(q)$ should be respected after updating clocks. Such an invariant corresponds to the intersection of individual TA location invariants. Through rule *Tau*, an internal (statically) non-hidden transition of a TA corresponds to an internal transition of the ETTS semantics. Hiding is not local, i.e not only attached to a given TA. In fact, from each TA location, after checking that such a location of the NTA state vector does not have the weakest committedness, we check then whether the current TA transition is not hidden by another transition outgoing from that location. Rule $SR_{i,j}(c)$ describes the synchronization of TA $\mathcal{T}_i$ and $\mathcal{T}_j$ on a channel c, where the input transition guard is only checked after taking into account the effect of the output transition action. Such a synchronization is held if either the send or the receive transition is not statically hidden. The dynamic priority of the resulting transition corresponds to that of channel c and the maximum process priority of both $\mathcal{T}_i$ and $\mathcal{T}_j$. Finally, both *Empty* and *Time* rules have been earlier explained.

Definition 15 (NTA refinement): The refinement between networks of timed automata is defined as the refinement between their associated ETTSs. Formally, given two NTA $\mathcal{N}_c$ and $\mathcal{N}_a$; then: $\mathcal{N}_c \sqsubseteq \mathcal{N}_a \triangleq ETTS(\mathcal{N}_c) \sqsubseteq ETTS(\mathcal{N}_a)$. Accordingly, we establish the following theorems:

Theorem 5 (Compositionality of NTA semantics): The ETTS of a NTA is bisimilar to the restriction to *Time* and *Tau*-transitions of the product of ETTSs associated to individual TA, i.e. $ETTS(NTA) \sim \Pi_i\ ETTS(TA_i)\backslash C$.

Proof. It is direct because we have the same composition rules in both sides. The difference resides in the occurrence of unmatched communication transitions in the ETTSs product, but these transitions will be suppressed by the restriction. This proof has been formalized and validated using the Coq theorem prover. □

Theorem 6 (Refinement and parallel composition): Given 2 networks of TA $N = \langle \mathcal{T}_1, .., \mathcal{T}_n\rangle$ and $N' = \langle \mathcal{T}_1', .., \mathcal{T}_n'\rangle$ defined on the same set of channels, N refines N' if:

- $\forall i\ \mathcal{T}_i' \sqsubseteq_{R_l^i,R_g}\ \mathcal{T}_i$,
- channel priority orders of N and N' are equivalent,
- each concrete process and its corresponding abstract one have the same priority (morphism),
- $\forall q\ q', R_l(q,q') \Rightarrow Comm(q) \Leftrightarrow Comm(q')$,
- refinement preserves deadlock-freeness.

Proof. It follows from theorems 4 and 5 together with the restriction theorem 2.

C. Experiments

In this section, we give an application of our compositional framework to refine a version of the well known Alternating

Bit Protocol [28] (ABP). In fact, we aim to verify that *the number of correctly received messages is less than the number of correctly sent ones, and their difference is bounded by one*. To send a new message, the TA *Sender* synchronizes with the TA *Mmedium* on channel *send* and increments the shared variable s. The TA *Receiver* receives the sent message via a synchronization with the *Mmedium* on channel *receive* and updates the shared variable r by adding one. The intended property is represented by the auxiliary Boolean variable $Ok \triangleq s == r \vee s == (r+1)$. This property cannot be verified using the UPPAAL toolbox on the NTA formed by *Sender, Receiver, Mmedium, Amedium* because of the unboundedness[3] of s and r. To deal with the unboundedness, a clever idea consists in the replacement of both *Sender* and *Receiver* automata by finite abstractions, where the evolutions of variables s and r are respectively modeled by two shared Boolean variables $b_1 = (s == r)$, initialized to $true$, and $b_2 = (s == r+1)$, initialized to $false$, with a slight modification of the corresponding involved transitions. The abstraction refinement of processes *Mmedium* and *Amedium* is the identity relation. Accordingly, the new property $Ok = b_1 \vee b_2$ can be checked. The abstraction refinement of each TA by its corresponding finite abstraction has been checked using a manual proof of the refinement $refin$ between their ETTS-based semantics, where both local and global spaces have been considered. The local space refinement R_l consists in: (a) the correspondence of local states; (b) the identity of local variables values. However, for the global space, the refinement R_g consists to: (a) match the values of b_1 and b_2 of the abstract system to the corresponding expressions computed through s and r in the concrete system; (b) check the identity between the other shared variables values. We may write then $refin = R_l^{Sender} \wedge R_l^{Receiver} \wedge R_l^{Mmedium} \wedge R_l^{Amedium} \wedge R_g$. This result has been checked using UPPAAL and a manual proof of refinement using observers.

V. Conclusion

In this paper, we have studied the refinement and composition of different timed systems with priorities. For the parallel composition operator we have defined, we give a corresponding (compositional) refinement relation. As a semantic model, our compositional framework has been successfully instantiated to define a compositional semantics of networks of timed automata, in which an unique generalized dynamic priority system of ETTS is defined from both NTA priority orders (channels, TA) with an important refinement property stating that: if each individual TA of an NTA refines another individual TA of another NTA, then the ETTS corresponding to the semantics of the first NTA refines the ETTS corresponding to the semantics of the second NTA. Furthermore, the theorems established within our framework have been proved [4]. In the future, we wish to extend our framework by other concepts, like assume-guarantee properties, in order to model and to analyze the semantics of the Fiacre language [8].

[3]In fact, it is checked upto the size of UPPAAL integers, then UPPAAL throws an out of range exception.

[4]Proofs will be available in the forthcoming PhD thesis of Abdeldjalil Boudjadar.

Acknowledgment. We wish to thank the anonymous referee for his scrutinous reading and valuable remarks.

References

[1] R. Alur. Timed automata. In *11th International Conference on Computer Aided Verification*, volume 1633 of *LNCS*, pages 8–22. 1999.
[2] A. Arnold. *Finite transition systems: semantics of communicating systems*. Prentice Hall International Ltd., Hertfordshire, UK, 1994.
[3] A. Basu, M. Bozga, and J. Sifakis. Modeling heterogeneous real-time components in BIP. In *SEFM'06*, pages 3–12. IEEE Computer Society, 2006.
[4] G. Behrmann, A. David, and K. G. Larsen. A Tutorial on Uppaal. Department of computer science, Aalborg university, 2004.
[5] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets*, pages 87–124. volume 3098 of LNCS, Springer-Verlag, 2004.
[6] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification - Model-Checking Techniques and Tools*. Springer-Verlag Berlin, 2001.
[7] J. Berendsen and F. Vaandrager. Compositional Abstraction in Real-Time Model Checking. In *FORMATS'08*, pages 233–249, 2008.
[8] B. Berthomieu, J. Bodeveix, M. Filali, H. Garavel, F. Lang, F. Peres, R. Saad, J. Stoecker, F. Vernadat, P. Gaufillet, and F. Lang. The syntax and semantics of FIACRE. Technical report, 2009.
[9] G. Bhat, R. Cleaveland, and G. Lüttgen. Dynamic priorities for modeling real-time. In *FORTE X/PSTV XVII '97, Osaka*, pages 321–336. Chapman and Hall, 1996.
[10] J. P. Bodeveix, A. Boudjadar, and M. Filali. An alternative definition for timed automata composition. In *ATVA'11*, pages 105 –119, Taiwan, 2011. LNCS 6996.
[11] S. Bornot, G. Gössler, and J. Sifakis. On the construction of live timed systems. In *TACAS'00*, pages 109–126, 2000.
[12] A. Boudjadar, J.-P. Bodeveix, and M. Filali. Revising and extending the UPPAAL communication mechanism. In *SC'12*, volume LNCS 7306, pages 114 – 131. Springer Heidelberg, 2012.
[13] P. Brémond-Grégoire, I. Lee, and R. Gerber. ACSR: An algebra of communicating shared resources with dense time and priorities. In *CONCUR*, pages 417–431, 1993.
[14] E. M. Clarke, D. E. Long, and K. L. McMillan. Compositional model checking. In *LICS'89*, pages 353–362, 1989.
[15] R. Cleaveland, G. Lüttgen, and V. Natarajan. Priority in process algebras. *Information and Computation*, 87:58–77, 1999.
[16] A. David, J. Hakansson, K. G. Larsen, and P. Pettersson. Model checking Timed Automata with Priorities Using DBM Subtraction. In *FORMAT'06*, pages 128–142. LNCS volume 4202, 2006.
[17] E. Fersman, P. Pettersson, and W. Yi. Timed Automata with asynchronous processes: Schedulability and decidability. In *TACAS'02*, pages 67–82. Volume LNCS 2280, Springer-Verlag, 2002.
[18] G. Gössler and J. Sifakis. Priority systems. In *FMCO'03*, pages 314–329, 2003.
[19] F. He, H. Zhu, W. Hung, X. Song, and M. Gu. Compositional abstraction refinement for timed systems. In *TASE'10*, pages 168 –176, 2010.
[20] T. A. Henzinger, Z. Manna, and A. Pnueli. Timed transition systems. 1992.
[21] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
[22] P.-A. Hsiung and S.-W. Lin. Model checking timed systems with priorities. In *RTCSA'05*, pages 539–544, USA, 2005.
[23] H. E. Jensen, K. G. Larsen, and A. Skou. Scaling up Uppaal: Automatic Verification of Real-Time Systems using Compositionality and Abstraction. In *FTRTFT'00*, pages 19–30, 2000.
[24] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. In *Journal on software tools for technology transfert*, 1997.
[25] R. Milner. *Communication and Concurrency*. Prentice Hall Ltd., 1989.
[26] M. Nielsen, G. Rozenberg, and P. Thiagarajan. Transition-systems, event structures, and unfoldings. *Information and Computation*, 118:191 – 207, 1995.
[27] M. Nielsen, G. Rozenberg, and P. S. Thiagarajan. Elementary transition systems. *Theoritical Computer Science*, pages 3–33, 1992.
[28] G. J. Veltink and S. Mauw. *Algebraic Specification of Communication Protocols*. Cambridge Tracts in Theoretical Computer Science No. 36, 2008.
[29] S. Yovine. Model checking Timed Automata. In *Lectures on Embedded Systems*. Lecture Notes in Computer Science 1494, Springer-Verlag, 1998.