Compositional Specification of Timed Systems (Extended Abstract) * †

Joseph Sifakis and Sergio Yovine

VERIMAG-SPECTRE ‡

1 Introduction

Motivation

It is generally admitted that timed systems can be obtained as extensions of untimed ones by adding constructs that allow to manipulate time explicitly or implicitly. For instance, timed automata [1] are automata augmented with continuous variables, called clocks, that can be tested and modified at transitions. Timed process algebras are languages obtained by adding constructs such as delays, timeouts, watchdogs [8] to untimed languages. Finally, the various classes of timed Petri nets are obtained by adding timing interval constraints to Petri nets. All these formalisms rely on an assumption of *orthogonality* between discrete and continuous changes which drastically simplifies the underlying semantics: a run of a timed system is a sequence of steps where continuous time progress and timeless transitions alternate.

A subtle point in the definition of the semantics of timed systems is the manner in which discrete and continuous changes interact. This is done in timed automata by associating with control locations invariants characterizing the states reachable by letting time progress. Using invariants allows, in particular, the specification of hard deadline constraints (upper bound constraints): when a deadline is reached for an action the progress of time is blocked by the invariant and the action becomes urgent (its execution is forced if it is enabled). In timed process algebras and timed Petri nets the invariants are implicit.

Specifying the interaction between continuous and discrete changes is an important problem, especially for complex systems obtained by composition of sequential components. Clearly, the use of invariants or other equivalent mechanisms can lead to deadlocks. If for instance, two communication actions (say an input and an output) are submitted to local deadline constraints, a time deadlock may occur as a result of their composition. This is not surprising and can be interpreted as inconsistency of the specifications. However, the problem arises

* In *Proc. 13th Annual Symp. on Theoretical Aspects of Computer Science, STACS'96*, pages 347–359, Grenoble, France, February 1996. LNCS 1046, Springer-Verlag.

† This work has been partially supported by CNET contract # 95 7B.

‡ VERIMAG is a joint laboratory of CNRS, INPG, Université J. Fourier, and Verilog SA. SPECTRE is a joint CNRS-INRIA project. Address: Centre Equation, 2 Ave. de Vignate, 38610 Gières, France. Joseph.Sifakis@imag.fr, Sergio.Yovine@imag.fr.

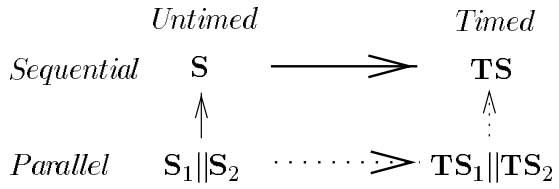


Fig. 1. Extending compositionality from untimed to timed specifications.

of how to write complex specifications and avoid such situations. An inherent difficulty of this problem is of course, the fact that usually timing constraints are global constraints (as time is a global variable) and it is not clear whether and how they can be obtained compositionally. We suggest a general approach for tackling this problem.

Compositionality

For an untimed description language with given coordination mechanisms we have a more or less clear insight concerning its appropriateness for the description of complex systems of a given type. However, this is not the case for timed systems. Given a complex global behavior we still do not know if the available coordination mechanisms allow a natural description. Consider a sequential untimed system **S** representing the behavior of a system obtained as the composition of two untimed components $\mathbf{S}_1$ and $\mathbf{S}_2$. Furthermore, suppose that a timed system **TS** is obtained from **S** by adding some global timing constraints (Figure 1). The question arises whether it is possible to find timed extensions $\mathbf{TS}_1$ and $\mathbf{TS}_2$ of $\mathbf{S}_1$ and $\mathbf{S}_2$, and a composition operator such that the composed system is equivalent to **TS**.

Clearly, this is a very general requirement for extending compositionality of untimed to compositionality of timed systems. However, under some reasonable restrictions it can be a useful criterion for evaluating the expressivity of timed specification languages.

Overview of this work

We propose a general framework for the compositional description of timed systems. We first start with a comparison of two different models for the specification of timed systems, namely timed automata and timed Petri nets.

The parallel composition of timed automata is defined as a timed automaton such that the invariant associated with a product state is the conjunction of the invariants associated with the component states [9, 4]. This rule is a consequence of the assumption of synchronous progress of time for all the components of a timed system.

Different classes of Timed Petri nets have been defined as timed extensions of Petri nets. Bounded Timed Petri nets can be considered as automata with

timing constraints and the question arises whether they can be obtained as the parallel composition of timed automata exactly as some classes of Petri nets can be obtained as the parallel composition of automata. This question has been partially investigated in [2] where the principle of a translation from safe timed Petri nets to a timed process algebra is proposed. We show that the parallel composition operations used for timed automata badly describe the synchronization mechanisms underlying Petri nets.

Starting from this fact, we propose a general framework for compositional description using a variant of timed automata called *timed automata with deadlines*. In this model invariants are replaced by *deadline conditions* associated with transitions. These are timing constraints specifying when a strict deadline is met for a transition that is, when the transition *must* be executed if it is enabled. We show that replacing invariants with deadline conditions associated with the transitions simplifies the problem of compositional description. However, invariants can be obtained from deadline conditions and a timed automaton with deadlines can be considered as a standard timed automaton.

On timed automata with deadlines we define two parallel composition operations: a *stiff* parallel composition which is close to usual parallel composition of timed automata and timed process algebras and a *flexible* parallel composition which allows to express synchronization of transitions in several classes of Timed Petri nets. These composition operators differ in the way deadlines of the composed system are obtained from the deadlines of its components.

The paper is organized as follows. In sections 2 and 3 we present timed automata and timed Petri nets and argue that timed automata and their associated parallel composition operator are not well adapted for the compositional description of timed Petri nets. Timed automata with deadlines are presented in section 4 where we present a compositional translation method from 1-safe timed Petri nets to this model. In section 5 we present basic ideas for a general compositional specification framework. Further and future work is discussed in section 6.

2 Timed Automata

Definition

Let $\mathcal{X}$ be a finite set of real-valued variables called *clocks*. A *valuation* $v \in \mathcal{V}$ of the clocks is a function that assigns a non-negative real-value $v(x) \in \mathbb{R}^+$ to each clock $x \in \mathcal{X}$. A *timing constraint* $\psi \in \Psi$ over $\mathcal{X}$ is a boolean combination of atoms of the form $x \# u$, where $x \in \mathcal{X}$, $u \in \mathbb{N}$, and $\# \in \{<, \leq, =, >, \geq\}$. We say that v satisfies ψ if $\psi(v)$ evaluates to true. For $v \in \mathcal{V}$ and $X \subseteq \mathcal{X}$, we define $v[X := 0]$ to be the valuation $v' \in \mathcal{V}$ such that $v'(x) = 0$ if $x \in X$, and $v'(x) = v(x)$ otherwise. For $\delta \in \mathbb{R}^+$, we define $v + \delta$ to be the valuation $v' \in \mathcal{V}$ such that $v'(x) = v(x) + \delta$ for all $x \in \mathcal{X}$.

Let $\mathcal{A}$ be a set of action names. An *automaton* $\mathbf{A}$ over $\mathcal{A}$ is a tuple $(\mathcal{S}, \mathcal{E})$ where $\mathcal{S}$ is a finite set of *locations* and $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{A} \times \mathcal{S}$ is a finite set of *edges*. A *timed* automaton $\mathbf{TA}$ is a tuple $((\mathcal{S}, \mathcal{E}), \mathcal{X}, G, R, I)$ where $(\mathcal{S}, \mathcal{E})$ is an automaton,

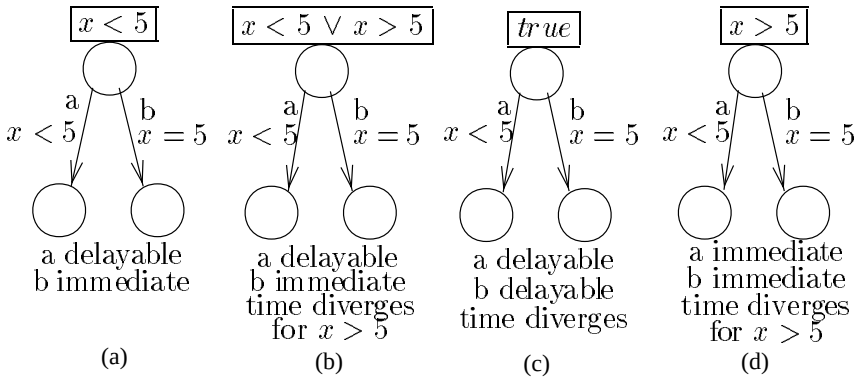


Fig. 2. The role of invariants.

$\mathcal{X}$ is a finite set of clocks, $G : \mathcal{E} \to \Psi$ associates with every edge $e \in \mathcal{E}$ a timing constraint called the *guard* of e, $R : \mathcal{E} \to \mathbf{2}^{\mathcal{X}}$ associates with every edge $e \in \mathcal{E}$ the set of clocks to be reset to zero, and $I : \mathcal{S} \to \Psi$ associates with every location $s \in \mathcal{S}$ a timing constraint called the *invariant* of s.

A *state* of a timed automaton is a pair (s, v) defined by a location and a clock valuation. At any state, **TA** can evolve either by a *discrete* state change corresponding to a move through an edge that may change the location and reset some of the clocks, or by a *continuous* state change due to the progress of time at a location. For $a \in \mathcal{A}$ and for $\delta \in \mathbb{R}^+$ we define the relations $\xrightarrow{a} \subseteq (\mathcal{S} \times \mathcal{V})^2$ and $\xrightarrow{\delta} \subseteq (\mathcal{S} \times \mathcal{V})^2$ characterizing respectively the discrete and the continuous state changes as follows:

$$\frac{(s, a, s') \in \mathcal{E},\ G(e)(v)}{(s, v) \xrightarrow{a} (s', v[R(e) := 0])} \qquad \frac{\forall \delta' \in \mathbb{R}^+, \delta' < \delta.\ I(s)(v + \delta')}{(s, v) \xrightarrow{\delta} (s, v + \delta)}$$

The role of invariants. As time progresses, the values of the clocks increase provided the state satisfies the invariant. For states that do not satisfy the invariant, the progress of time is "stopped". This mechanism allows the specification of hard deadlines: when for some action a deadline specified by the invariant is reached, the continuous flow of time is interrupted. Therefore, the action becomes urgent and it is "forced" to occur if it is enabled.

Figure 2 illustrates the use of invariants to express urgency of actions. An action is said to be *delayable* if whenever it is enabled its execution can be postponed by letting time progress. Clearly, delaying an action may cause its disabling. An action is called *immediate* if it cannot be delayed. We say that time diverges from a given state (s, v) if $I(s)(v + \delta)$ for all $\delta \in \mathbb{R}^+$.

Notice that our rule for the progress of time follows [5]. This definition is slightly different from the one given in [9, 4] where time can progress from state (s, v) by some $\delta \in \mathbb{R}^+$ if for all $\delta' \in \mathbb{R}^+$, $\delta' \leq \delta$, $v + \delta'$ satisfies $I(s)$. It is easy to check that the latter definition does not allow to characterize situations such as

the one illustrated in Figure 2(b) where there is a strict deadline for $x = 5$ but time diverges for $x > 5$.

Parallel composition

Let $\mathbf{TA}_i$ be $((\mathcal{S}_i, \mathcal{E}_i), \mathcal{X}_i, G_i, R_i, I_i)$, for $i = 1, 2$, with disjoint sets of locations and clocks, and $A \subseteq \mathcal{A}$ be a set of *communication* actions. The parallel composition $\mathbf{TA}_1 \parallel_A \mathbf{TA}_2$ is the timed automaton $((\mathcal{S}, \mathcal{E}), \mathcal{X}, G, R, I)$ where the set of locations $\mathcal{S}$ is $\mathcal{S}_1 \times \mathcal{S}_2$, the set of clocks $\mathcal{X}$ is $\mathcal{X}_1 \cup \mathcal{X}_2$, for all $(s_1, s_2) \in \mathcal{S}$, the invariant $I(s_1, s_2)$ is the predicate $I_1(s_1) \wedge I_2(s_2)$, and $\mathcal{E}$, G and R are defined by the following rules (the second rule is applied symmetrically to the other component):

$$\frac{e_1 = (s_1, a, s_1') \in \mathcal{E}_1,\ e_2 = (s_2, a, s_2') \in \mathcal{E}_2,\ a \in A}{e = ((s_1, s_2), a, (s_1', s_2')) \in \mathcal{E},\ \boxed{G(e) = G_1(e_1) \wedge G_2(e_2),\ R(e) = R_1(e_1) \cup R_2(e_2)}}$$

$$\frac{e_1 = (s_1, a_1, s_1') \in \mathcal{E}_1,\ a_1 \notin A}{e = ((s_1, s_2), a_1, (s_1', s_2)) \in \mathcal{E},\ \boxed{G(e) = G_1(e_1),\ R(e) = R_1(e_1)}}$$

Notice that $\parallel_A$ is the extension of the parallel composition operator $||_A$ of CSP [3] to timed automata. The timed automaton $\mathbf{TA}$ is obtained by adding timing constraints to the automaton $(\mathcal{S}, \mathcal{E})$ corresponding to the parallel composition $(\mathcal{S}_1, \mathcal{E}_1)||_A(\mathcal{S}_1, \mathcal{E}_1)$ of the components. The timing constraints associated with the edges are highlighted with boxes.

Example 1. Consider the behavior of a medium that receives messages from a sender (in) and transmits them to a receiver (out). The system satisfies the following three timing requirements: two consecutive inputs are separated by a time equal to 7, two consecutive outputs are separated by a time greater than or equal to 4, and the transmission delay between an input and an output is in the time interval [3,8]. Figure 3 shows the description of the medium obtained by the parallel composition of the three timed automata modeling the requirements.

Notice that the use of invariants can lead to deadlocks. For instance, if the interval [3,8] for the transmission delay is replaced by (7,8], when the system is at location ABD with clocks x and z equal to 7, neither time can progress nor the action out is enabled, and therefore a time deadlock occurs. Such a situation is interpreted as an inconsistency of the specification.

3 Timed Petri Nets

For the sake of simplicity, we use the term "timed Petri nets" to denote a subclass of a timed extension of 1-safe Petri nets known as *Time Stream* Petri nets [10]. Other timed extensions of 1-safe Petri nets, with timed transitions [6] or with timed places [11], correspond to special cases of the model considered here.

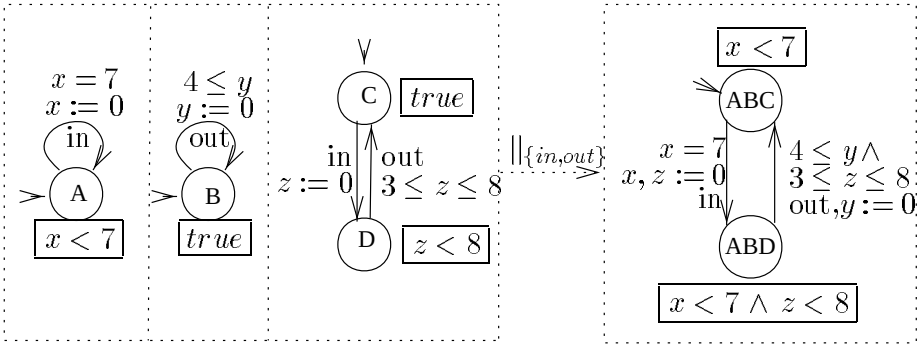


Fig. 3. Timed specification of the medium.

Definition

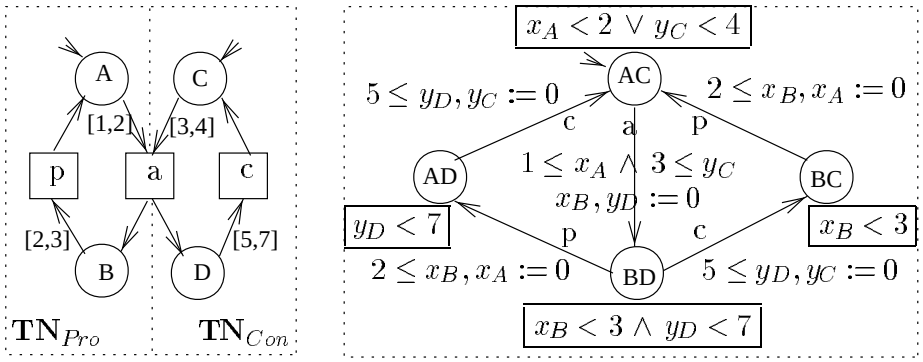
A Petri net **N** is a tuple $(\mathcal{S},\mathcal{T})$ where $\mathcal{S}$ is a finite set of *places* and $\mathcal{T} \subseteq \mathbf{2}^{\mathcal{S}} \times \mathcal{A} \times \mathbf{2}^{\mathcal{S}}$ is a finite set of *transitions*. For $t = (S, a, S') \in \mathcal{T}$, we write $\bullet t$ for the set S of *input* places of t, $t\bullet$ for the set S' of *output* places of t, and $\ell(t)$ for the action a labeling t. A Petri net $(\mathcal{S},\mathcal{T})$ represents the automaton $(\mathbf{2}^{\mathcal{S}}, \mathcal{E}_{\mathcal{T}})$ where for $S \in \mathcal{S}$ and $t \in \mathcal{T}$, whenever $\bullet t \subseteq S$ and $(S - \bullet t) \cap t\bullet = \emptyset$, the edge $e = (S, \ell(t), ((S - \bullet t) \cup t\bullet)) \in \mathcal{E}_{\mathcal{T}}$.

A *timed* Petri net **TN** is a tuple $((\mathcal{S},\mathcal{T}),L,U)$ where $(\mathcal{S},\mathcal{T})$ is a Petri net, and $L : \mathcal{S} \times \mathcal{T} \to \mathbb{N}$ and $U : \mathcal{S} \times \mathcal{T} \to \mathbb{N} \cup \{+\infty\}$, are partial functions defined for all (s,t) such that $s \in \bullet t$. We require $L(s,t) \leq U(s,t)$.

TN can be represented as a bipartite labeled digraph with set of nodes $\mathcal{S} \cup \mathcal{T}$ and set of arcs $\{(s,t) \mid s \in \bullet t\} \cup \{(t,s) \mid s \in t\bullet\}$. The arc (s,t) is labeled with the interval $[L(s,t),U(s,t)]$.

Timed Petri nets are used to model concurrent timed systems. Transitions express synchronization of streams of tokens representing information processed at places. Intervals associated with arcs specify times when tokens become available and can therefore be used for firing the corresponding transition.

Example 2. Consider a system composed of a producer and a consumer. The producer takes a time between 2 and 3 to produce (p) an item, and then it is ready to make it available (a) to the consumer after a delay between 1 and 2. The consumer takes between 5 and 7 to consume (c) an item, and then it is ready to get an available (a) item from the producer after a time between 3 and 4. Figure 4(a) shows the timed Petri net for this system. The synchronization between the producer and the consumer on transition "a" requires that the tokens have been at the input places A and C for a time greater than or equal to the lower bounds, but only one of the upper bounds is required to hold in order to fire the transition.



(a) Timed Petri Net (b) Timed Automaton

Fig. 4. Producer-Consumer.

Translating timed Petri nets into timed automata

We propose here an operational semantics for timed Petri nets. With every place we associate a clock that measures the sojourn time of a token. Each time the place receives a token, the clock is reset to zero. Time intervals on arcs are represented by timing constraints on these clocks.

Let $\mathbf{TN} = ((\mathcal{S},\mathcal{T}),L,U)$ and $\mathcal{X} = \{x_s \mid s \in \mathcal{S}\}$ be a set of clocks. We associate with each $t \in \mathcal{T}$ a guard $G(t)$ and a set of clocks $R(t)$ to be reset to zero, and with each $S \subseteq \mathcal{S}$ an invariant $I(S)$, such that:

$$G(t) = \bigwedge_{s \in \bullet t} L(s,t) \leq x_s,\ R(t) = \{x_s \mid s \in t\bullet\},\ I(S) = \bigwedge_{\bullet t \subseteq S} \bigvee_{s \in \bullet t} x_s < U(s,t).$$

A state of $\mathbf{TN}$ is a pair $(S,v) \in \mathbf{2}^{\mathcal{S}} \times \mathcal{V}$, and for $a \in \mathcal{A}$ and $\delta \in \mathbb{R}^+$ the relations $\xrightarrow{a} \subseteq (\mathbf{2}^{\mathcal{S}} \times \mathcal{V})^2$ and $\xrightarrow{\delta} \subseteq (\mathbf{2}^{\mathcal{S}} \times \mathcal{V})^2$ are defined as follows:

$$\frac{t \in \mathcal{T},\ \bullet t \subseteq S,\ G(t)(v)}{(S,v) \xrightarrow{\ell(t)} ((S - \bullet t) \cup t\bullet, v[R(t) := 0])} \qquad \frac{\forall \delta' \in \mathbb{R}^+, \delta' < \delta.\ I(S)(v+\delta')}{(S,v) \xrightarrow{\delta} (S, v+\delta)}$$

Notice that for every state, time can progress if for every enabled transition there exists *at least one* input place whose deadline has not expired.

The following proposition is a direct consequence of the above definition. The term equivalent means that the underlying models are strongly bisimilar.

Proposition 1. *The timed Petri net* $((\mathcal{S},\mathcal{T}),L,U)$ *is equivalent to the timed automaton* $((\mathbf{2}^{\mathcal{S}},\mathcal{E}_{\mathcal{T}}),\mathcal{X},G,R,I)$, *where for* $e = (S,\ell(t),S') \in \mathcal{E}_{\mathcal{T}}$, $G(e) = G(t)$ *and* $R(e) = R(t)$.

Figure 4(b) shows the corresponding timed automaton for the timed Petri net of the producer-consumer system. Notice that though we have used four clocks x_A,

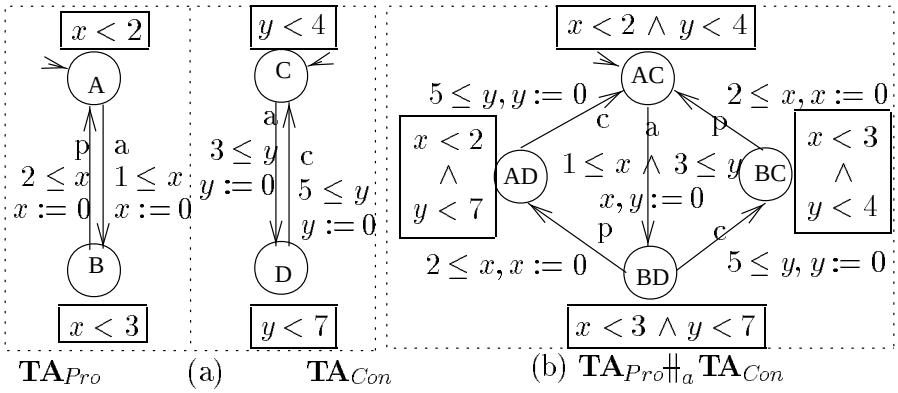


Fig. 5. Timed automata for Producer-Consumer.

x_B, y_C and y_D, only two clocks, say x for the producer and y for the consumer, are really needed.

Parallel composition

Given two Petri nets $(\mathcal{S}_i, \mathcal{T}_i)$, $i = 1, 2$, such that $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, and $A \subseteq \mathcal{A}$, we define a Petri net $(\mathcal{S}, \mathcal{T})$ corresponding to the parallel composition $(\mathbf{2}^{\mathcal{S}_1}, \mathcal{E}_{\mathcal{T}_1})||_A(\mathbf{2}^{\mathcal{S}_2}, \mathcal{E}_{\mathcal{T}_2})$, denoted by $(\mathcal{S}_1, \mathcal{T}_1)||_A(\mathcal{S}_2, \mathcal{T}_2)$. The Petri net $(\mathcal{S}, \mathcal{T})$ can be directly defined as follows: $\mathcal{S}$ is $\mathcal{S}_1 \cup \mathcal{S}_2$, and

$$\frac{t_i \in \mathcal{T}_i,\ \ell_i(t_i) \notin A}{t_i \in \mathcal{T}} i = 1, 2 \qquad \frac{t_1 \in \mathcal{T}_1,\ t_2 \in \mathcal{T}_2,\ \ell_1(t_1) = \ell_2(t_2) = a \in A}{(\bullet t_1 \cup \bullet t_2, a, t_1\bullet \cup t_2\bullet) \in \mathcal{T}}$$

This parallel composition operator can be trivially extended to timed Petri nets by labeling the arcs of $(\mathcal{S},\mathcal{T})$ with the corresponding time intervals. Given $\mathbf{TN}_1$ and $\mathbf{TN}_2$, we write $\mathbf{TN}_1 \between_A \mathbf{TN}_2$, to denote the resulting timed Petri net.

The question arises whether following the compositionality principle depicted in Figure 1, it is possible to define the parallel composition operator $\between_A$ on timed automata such that for any timed Petri nets $\mathbf{TN}_1$ and $\mathbf{TN}_2$, with corresponding timed automata $\mathbf{TA}_1$ and $\mathbf{TA}_2$, $\mathbf{TN}_1 \between_A \mathbf{TN}_2$ is equivalent to $\mathbf{TA}_1 \between_A \mathbf{TA}_2$.

The example of the producer-consumer system shows that this question has no simple answer: the timed automaton corresponding to $\mathbf{TN}_{Pro} \between_a \mathbf{TN}_{Con}$ shown in Figure 4(b) is different from the timed automaton $\mathbf{TA}_{Pro} \between_a \mathbf{TA}_{Con}$ illustrated in Figure 5. Actually, both have the same untimed structure, as well as guards and resets, but they do not have the same invariants. It is not difficult to see that the right invariants cannot be obtained compositionally using the $\between_A$ parallel composition operator. In the following section we propose a variant of timed automata and a parallel composition operator that corresponds to $\between_A$.

4 Timed Automata with Deadlines

Definition

A timed automaton with deadlines **TAD** is a tuple $((\mathcal{S},\mathcal{E}),\mathcal{X},G,R,D)$ where $\mathcal{S}$, $\mathcal{E}$, $\mathcal{X}$, G and R are defined as for timed automata, and $D:\mathcal{E}\to\Psi$ associates with each edge $e\in\mathcal{E}$ a *deadline* condition specifying when the edge e becomes urgent. For $s\in\mathcal{S}$, we define

$$D(s)=\bigvee_{e=(s,a,s')\in\mathcal{E}} D(e) \quad \text{and} \quad I(s)=\neg D(s)$$

By definition **TAD** behaves like the timed automaton $\mathbf{TA}=((\mathcal{S},\mathcal{E}),\mathcal{X},G,R,I)$. That is, timed automata with deadlines are timed automata where time can progress at a location as long as all the deadline conditions associated with the outgoing edges are not satisfied.

Associating deadline conditions with edges enhances flexibility in the expression of timing constraints. For a given edge e, the guard $G(e)$ determines when e *may* be executed, while $D(e)$ determines when it *must* be executed. Clearly, for all the states satisfying $\neg G(e)\wedge D(e)$, time can be blocked. Therefore, it is reasonable to require that $D(e)$ implies $G(e)$ to avoid time deadlocks. If this condition holds for all the edges, then no time deadlock can occur. When $D(e)$ is equal to $G(e)$, e is immediate and must be executed as soon as it becomes enabled. If $D(e)$ is false, e is delayable at any state. Between these two extreme cases, several intermediate situations of practical interest can be defined by an appropriate choice of $D(e)$.

Proposition 2. *The timed Petri net* $((\mathcal{S},\mathcal{T}),L,U)$ *is equivalent to the timed automaton with deadlines* $((\mathbf{2}^{\mathcal{S}},\mathcal{E}_{\mathcal{T}}),\mathcal{X},G,R,D)$, *where for* $e=(S,\ell(t),S')\in\mathcal{E}_{\mathcal{T}}$, $D(e)=\bigwedge_{s\in\bullet t} x_s\geq U(s,t)$.

Figure 6(b) shows the timed automaton with deadlines corresponding to the timed Petri net of Figure 4(a). The deadline conditions are highlighted with boxes.

Parallel composition

Let $\mathbf{TAD}_i$ be $((\mathcal{S}_i,\mathcal{E}_i),\mathcal{X}_i,G_i,R_i,D_i)$ for $i=1,2$, with disjoint sets of locations and clocks, and $A\subseteq\mathcal{A}$. We define two operators: $\|_A$ and $\Downarrow_A$, respectively called *stiff* and *flexible* parallel composition. The resulting timed automata with deadlines $\mathbf{TAD}_{stiff}$ and $\mathbf{TAD}_{flex}$ have the same structure $((\mathcal{S},\mathcal{E}),\mathcal{X},G,R)$ obtained by the composition rules given for timed automata, and only differ on the way deadlines are associated with edges. For the stiff parallel composition, D_{stiff} is given by the following rules:

$$\frac{e_1=(s_1,a,s_1')\in\mathcal{E}_1,\ e_2=(s_2,a,s_2')\in\mathcal{E}_2,\ a\in A}{e=((s_1,s_2),a,(s_1',s_2'))\in\mathcal{E},\ \boxed{D_{stiff}(e)=D_1(e_1)\vee D_2(e_2)}}$$

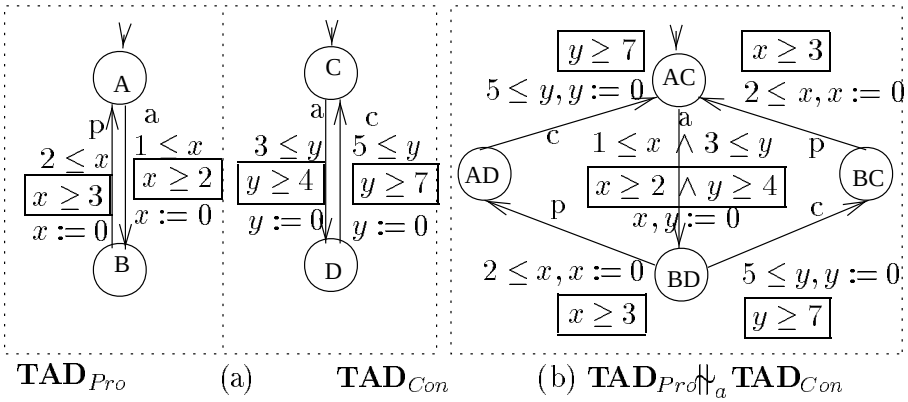


Fig. 6. Timed automata with deadlines for Producer-Consumer.

$$\frac{e_1 = (s_1, a_1, s_1') \in \mathcal{E}_1,\ a_1 \notin A}{e = ((s_1, s_2), a_1, (s_1', s_2)) \in \mathcal{E},\ \boxed{D_{stiff}(e) = D_1(e_1) \vee D_2(s_2)}}$$

For the flexible parallel composition, D_{flex} is given by the following rules:

$$\frac{e_1 = (s_1, a, s_1') \in \mathcal{E}_1,\ e_2 = (s_2, a, s_2') \in \mathcal{E}_2,\ a \in A}{e = ((s_1, s_2), a, (s_1', s_2')) \in \mathcal{E},\ \boxed{D_{flex}(e) = D_1(e_1) \wedge D_2(e_2)}}$$

$$\frac{e_1 = (s_1, a_1, s_1') \in \mathcal{E}_1,\ a_1 \notin A}{e = ((s_1, s_2), a_1, (s_1', s_2)) \in \mathcal{E},\ \boxed{D_{flex}(e) = D_1(e_1)}}$$

Proposition 3. *For $i = 1, 2$, if $\mathbf{TA}_i$ is the timed automaton corresponding to the timed automaton with deadlines $\mathbf{TAD}_i$, then $\mathbf{TA}_1 \Vdash_A \mathbf{TA}_2$ is equivalent to $\mathbf{TAD}_1 \Vdash_A \mathbf{TAD}_2$.*

Proposition 4. *For $i = 1, 2$, if $\mathbf{TAD}_i$ is the timed automaton with deadlines corresponding to the timed Petri net $\mathbf{TN}_i$, then $\mathbf{TAD}_1 \Vdash_A \mathbf{TAD}_2$ is equivalent to $\mathbf{TN}_1 \Vdash_A \mathbf{TN}_2$.*

Figure 6(b) shows the timed automata with deadlines obtained as the flexible parallel composition of the timed automata with deadlines $\mathbf{TAD}_{Pro}$ and $\mathbf{TAD}_{Con}$ corresponding to the producer and the consumer respectively. Notice that $\mathbf{TAD}_{Pro} \Vdash_a \mathbf{TAD}_{Con}$ is equivalent to the timed automaton of Figure 4(b).

5 General framework

In this section we propose a framework that extends the results of the previous sections. The basic idea is that for timed systems, exactly as for untimed

systems, different parallel composition operators can be defined by considering the (synchronous) parallel composition of labeled structures and defining appropriately a composition operation between labels. This idea for untimed systems goes back to [7] and has been widely used in process algebras.

We suppose that the timed behavior resulting by combining two edges $e_i = (s_i, a_i, s_i')$, for $i = 1, 2$, is an edge $e = ((s_1, s_2), a, (s_1', s_2'))$, and $G(e)$ and $D(e)$ are functions of the $G_i(e_i)$'s and $D_i(e_i)$'s respectively. This is already true in the case of stiff and flexible composition where $G(e)$ is the conjunction of guards and $D(e)$ is the disjunction or the conjunction of the deadline conditions. It is not difficult to realize that the definition of synchronization in timed Petri nets in terms of flexible parallel composition is not robust enough. For instance, in the producer-consumer example of Figure 6, replacing in the edge "a" of $\mathbf{TAD}_{Pro}$ the guard $1 \leq x$ by $1 \leq x \leq 2$ and/or the deadline condition $x \geq 2$ by $x = 2$ leads to equivalent descriptions. However, the flexible composition of any of them with $\mathbf{TAD}_{Con}$ gives different (wrong) results.

To obtain a robust flexible composition it is necessary to take into account in the composition of the guards and deadlines not only the present state but also its past and future in order to express waiting. Our thesis is that the description of synchronization mechanisms in a compositional framework, requires the use of temporal properties. We consider here a simple modal language that suffices for characterizing the parallel composition operator $\Vdash$. The formulas are obtained from timing constraints $\psi \in \Psi$ by using two modalities $before(\psi)$ and $after(\psi)$ defined as follows:

$$before(\psi)(v) \text{ iff } \exists \delta \in \mathbb{R}^+.\ \psi(v + \delta)$$
$$after(\psi)(v) \text{ iff } \exists \delta \in \mathbb{R}^+.\ \psi(v - \delta)$$

Consider the generalization of the synchronization mechanism of timed Petri nets characterized as follows. For edges e_i, $i = 1, 2$, with guards $G_i(e_i)$ and deadline conditions $D_i(e_i)$, the guard $G(e)$ and the deadline $D(e)$ of the composed edge e are such that:

- $G(e)$ is true if both $G_1(e_1)$ and $G_2(e_2)$ are true or if one of them is true and the other has been true. This requirement expresses the fact that a process may wait even though its guard becomes disabled due to the progress of time. That is,

$$G(e) = after(G_1(e_1)) \wedge G_2(e_2) \vee G_1(e_1) \wedge after(G_2(e_2))$$

- $D(e)$ is true if both $D_1(e_1)$ and $D_2(e_2)$ are true but also if one of the deadlines is satisfied and a deadline of the other will never be encountered in the future. In other words, $D(e)$ does not contain deadline states of one component that precedes deadline states of the other. This rule avoids time deadlocks due to unilateral requirement from one component to meet a deadline for a communication action that cannot occur. That is,

$$\begin{aligned} D(e) = {} & D_1(e_1) \wedge D_2(e_2) \vee \\ & \neg before(D_1(e_1)) \wedge D_2(e_2) \vee D_1(e_1) \wedge \neg before(D_2(e_2)) \end{aligned}$$

One can check that in the producer-consumer example, for the different timed automata with deadlines equivalent to $\mathbf{TAD}_{Pro}$, the guards and deadline conditions for the transition "a" obtained by applying the above rules describe the flexible parallel composition.

6 Discussion

The paper defines a general framework for extending compositionality from untimed to timed descriptions. This framework is obviously applicable to hybrid systems too. Based on the assumption that sequential timed components are timed extensions of untimed ones, it shows that it is possible to compositionally describe a global timed behavior if the underlying untimed one is obtained compositionally and the timing constraints associated with its transitions can be expressed as functions of local timing constraints.

When composing timed systems three independent parameters operate:

1. The parallel composition for the associated untimed system. There are different choices ranging from completely synchronous to completely asynchronous parallel composition.

2. The composition rules for guards. The composed guard may depend on the past if waiting is possible.

3. The composition rules for deadline conditions.The composed deadline condition maydepend on the future, if waiting is possible. To avoid time deadlocks, it is sufficient to restrict deadline conditions to guards.

Replacing invariants in timed automata with deadline conditions simplifies the compositionality problem and allows a better intuition about how a system behaves. The two cases of parallel composition correspond to two different ways of composing timing constraints. Stiff parallel composition considers all deadlines to be equally important; deadlines that cannot be met lead to time deadlocks (inconsistency). The flexible one can be used to characterize coordination of loosely coupled components submitted to local constraints that are satisfied following a "best effort" principle: components wait as long as a synchronization condition is not satisfied and ignore their own deadlines provided there exists a component with a forthcoming deadline. The waiting components synchronize as soon as the synchronization condition holds. It is important to notice that waiting simply means that deadline conditions are ignored; in any case, time progresses in the same manner for all the parallel components.

An important methodological question concerns the use of the appropriate parallel composition operator in a description. Consider the gate crossing problem often taken to illustrate the use of timed automata for modeling timed systems [4]. The solutions are usually expressed as the (stiff) parallel composition of two timed automata $TRAIN$ and $GATE$. $TRAIN$ sends signals *approach* and *exit* to notify its approach and exit from the crossing section. $GATE$ reacts to these signals by changing its state. The use of stiff parallel composition

in this case reflects the fact that violating the deadlines is not acceptable as it may be catastrophic. On the contrary, to model the interaction between the $TRAIN \| GATE$ system and a CAR component, representing a car crossing the tracks, it is natural to use flexible parallel composition as CAR does not signal its arrival to $GATE$, but simply might wait for $GATE$ to become open.

This work is, to our knowledge, a first attempt to connect two different approaches for composing timed behaviors. It proposes a general framework which is still incomplete and is based on ideas wich remain to be validated.

Acknowledgement: We thank Amir Pnueli for constructive critiques of the ideas developed in the paper.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In *Proc. REX Workshop "Real-Time: Theory in Practice"*. Lecture Notes in Computer Science 600, Springer-Verlag, 1991.
3. S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31(3):560–599, 1984.
4. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
5. O. Maler and A. Pnueli. Timing analysis of asynchronous circuits using timed automata. In *CHARME'95*, pages 189–205. Lecture Notes in Computer Science 987, Springer-Verlag, 1995.
6. P. Merlin and D. J. Farber. Recoverability of communication protocols. *IEEE Transactions on Communications*, 24(9), September 1976.
7. R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.
8. X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *Proc. 3rd Workshop on Computer-Aided Verification*, pages 376–398. Lecture Notes in Computer Science 575, Springer-Verlag.
9. X. Nicollin, J. Sifakis, and S. Yovine. Compiling real-time specifications into extended automata. *IEEE TSE Special Issue on Real-Time Systems*, 18(9):794–804, September 1992.
10. P. Sénac, M. Diaz, and P. de Saqui-Sannes. Toward a formal specification of multimedia scenarios. *Annals of telecommunications*, 49(5-6):297–314, 1994.
11. J. Sifakis. Use of petri nets for performance evaluation. In *Measuring, modeling and evaluating computer systems*, pages 75–93. North-Holland, 1977.