The Coarsest Congruence for Timed Automata with Deadlines Contained in Bisimulation*

Pedro R. D'Argenio[1]** and Biniam Gebremichael[2]

[1] CONICET – FaMAF, Universidad Nacional de Córdoba
Ciudad Universitaria, 5000 Córdoba, Argentina.
[2] Institute for Computing and Information Sciences. Radboud University Nijmegen
P.O. Box 9010, 6500 GL Nijmegen, The Netherlands
dargenio AT famaf.unc.edu.ar and B.Gebremichael AT cs.ru.nl

Abstract. Delaying the synchronization of actions may reveal some hidden behavior that would not happen if the synchronization met the specified deadlines. This precise phenomenon makes bisimulation fail to be a congruence for the parallel composition of timed automata with deadlines, a variant of timed automata where time progress is controlled by deadlines imposed on each transition. This problem has been known and unsolved for several years. In this paper we give a characterization of the coarsest congruence that is included in the bisimulation relation. In addition, a symbolic characterization of such relation is provided and shown to be decidable. We also discuss the pitfalls of existing parallel compositions in this setting and argue that our definition is both reasonable and sufficiently expressive as to consider the modeling of both soft and hard real-time constraints.

1 Introduction

Design and specification languages allow to model systems in a modular manner by linking small modules or components using the language operations —such as the sequential composition or the parallel composition— in order to build larger modules. Hence a desirable requirement is that the language is *compositional* with respect to its semantics. By compositional we mean that components can be replaced by behaviorally equivalent components without changing the properties of the larger model in which they are embedded. The preservation of such properties can be guaranteed by means of semantic equivalences or preorders. For example branching bisimulation preserves CTL* [11], language inclusion preserves LTL [22] and, in particular, timed bisimulation preserves (timed) properties expressed in logics such as TCTL [27]. Hence, compositionality amounts to requiring that relations like these are *congruences* (or precongruences) for the different operations of the language.

Timed automata [1, 18] are used to model real-time systems and have become popular as modeling language for several model checkers because of its simplicity and tractability [2, 9, 10]. Timed automata are automata with the additional ingredients of *clocks*. Clocks are variables that increase at the same rate in order to register time

* Supported by the EC project IST-2001-35304 AMETIST, URL: ametist.cs.utwente.nl.
** Also at Formal Methods and Tools, Dep. of Comp. Sci. University of Twente. Supported by the NWO Vernieuwingsimpuls and the ANPCyT project PICT 11-11738.

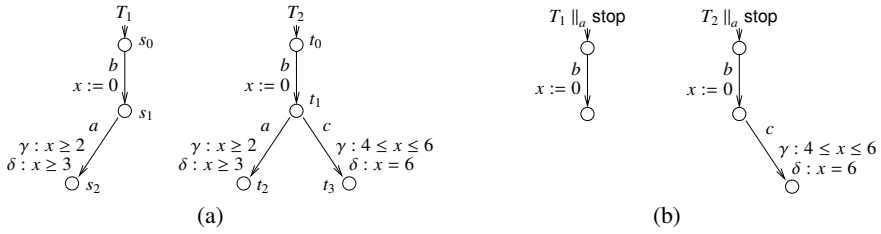

Fig. 1. TAD and compositionality

progress. Transitions are labeled with constraints on clocks, called *guards*, that indicate when such transition *may* take place. Usually timed automata are used to model real-time systems with *hard* constraints. In this cases, timed automata are equipped with an *invariant*, which is a constraint on clocks that limits time progress in each control state [18]: the system is obliged to leave such state before invalidating the invariant.

Because of the nature of invariants, time is allowed to progress in the composed timed automaton only if time is allowed to progress in all component automata. Therefore, if one of the automaton fails to meet the deadline imposed by a partner in a synchronisation, then the entire system crashes (which is represented by the so called *time deadlock*: the composed system has reached a state where time is blocked forever). This is the nature of hard deadlines. But it is debatable whether hard deadlines are always appropriate to model synchronisation in real-time systems. An alternative composition is, a composition with *soft* deadlines that allows the fast partner to wait the slow partner if nothing else is possible. In this case the deadlines can be violated, but the synchronisation is performed urgently whenever possible.

Timed automata with deadlines (TAD for short) [26, 7, 5, 6] were introduced for these reasons. Parallel compositions with hard and soft deadlines as well as urgency can be naturally defined in TAD. At the same time, the TAD model ensures, under reasonable assumption, what is called time reactivity in [6] and time lock freedom in [8], that is, whenever time progress stops there exists at least one transition enabled. (Note that time reactivity and hard constraints are not fully compatible.) This model is nowadays embedded in modeling languages such as IF [10] and MoDeST [16, 4], and urgent transitions in Uppaal [3] can be seen as a particular instance of TAD transitions.

TADs do not have invariants. Instead, a TAD transition has associated a second clock constraint, called *deadline*, that indicates in which moment such transition *must* be taken. As a consequence, a deadline is required to hold *only if* the corresponding guard holds ensuring the transition can be taken after the deadline is reached. In this sense, the deadline impose an *urgency constraint*.

Contrary to the traditional timed automata setting, bisimulation in the TAD model *is not* preserved by parallel composition [6]. This is illustrated in the following example. T_1 in Fig. 1.(a) depicts a TAD in which circles represent control state and arrows are control transitions. In particular the small incoming arrow identifies the initial state. T_1 performs first an action b at any moment and sets clock x to 0. As time progresses, the value of x increases and when it takes value 2 action a becomes enabled. This is controlled by guard $\gamma : x \geq 2$. At any point after x takes value 2, this transition may take place, but as time continues to progress and x takes value 3, the deadline $\delta : x \geq 3$ obliges the execution of the transition. Notice that T_2 shows a similar behavior since

action c cannot be executed: the deadline of a obliges its execution before the guard of c becomes enabled. In fact, T_1 and T_2 are timed bisimilar in the sense of [6].

Suppose now that T_1 is composed in parallel with the automaton stop requiring synchronization on action a. (stop is the automaton with a single location and no transition; hence, it does not do anything but idling.) This blocks the execution of action a in T_1. The resulting automaton $T_1 \parallel_a$ stop is depicted in Fig. 1.(b). Similarly, the composition of T_2 with stop in $T_2 \parallel_a$ stop also blocks the execution of a, but in this case time progresses beyond 3 time units allowing the execution of c after 4 time units (see Fig. 1.(b)). As a consequence $T_1 \parallel_a$ stop and $T_2 \parallel_a$ stop are not bisimilar.

To the best of our knowledge there is no characterization of a congruence for parallel composition on TADs. The only exception is what is called strong congruence in [6], which is the usual bisimulation applied directly on TADs. This relation is, however, far too strong as it requires the syntactic equality of guards, deadlines, and clock resets.

In this paper we present a congruence relation for parallel composition and prove that it is the coarsest congruence included in the bisimulation relation. This new relation, which we call ∇-bisimulation (read "drop-bisimulation"), is in fact the usual bisimulation on an extended semantics of TAD. Such semantics allows for time progressing beyond deadlines but carefully accounting the actions whose deadline have been overruled. We also give a symbolic characterization of ∇-bisimulation, that is, a relation defined directly on TADs. As a corollary of this characterization we obtain that ∇-bisimulation is decidable. Another particular contribution of this paper is that the proof of congruence is entirely carried out at symbolic level (i.e., without resorting to the underlying transition system in which ∇-bisimulation is defined). We finally discuss different kind of parallel compositions on TADs (mostly defined already in the literature) reporting which of them preserves ∇-bisimulation and which do not and why.

Related Work. The failure of bisimulation to be a congruence becomes apparent when soft deadlines are considered, that is actions that may be urgent in isolation are required to wait if they are intended for synchronization i.e. synchronizing actions need to be *patient*. This problem has appeared in the context of stochastic process algebra where synchronization is required to be patient (e.g. [20, 19, 14]). It becomes evident (in a similar manner as above) if bisimulation is considered for the underlying probabilistic transition system rather than for the finer symbolic model [14]. The problem of compositionality also showed up in other process algebras for performance behavior [13].

In [21], compositionality is studied on timed automata with urgent actions w.r.t. simulation. (An urgent action corresponds to an action in TADs for which guard and deadline are the same.) In this case, it suffices to add a condition of readiness on the urgent actions to achieve precongruence. Recently, [17] defined a variant of TADs where actions are distinguished between input and output following the model of [25] and for which bisimulation *is* a congruence for the parallel composition. This is possible due to input enabling and to the fact that only output actions are allowed to be urgent (i.e. to have deadline.) Therefore there is no need to wait for synchronization as it is always possible. Though the restrictions imposed by [17] makes the new model much simpler and tractable, using it to describe soft real-time systems may result in complex models.

In addition to the solution for the compositionality problem, we also give a symbolic characterization of the congruence. Our work is based on the result of Lin & Yi [23]

who gave a symbolic characterization of the bisimulation for timed automata. In turn, their result is based on Čerāns' who determined that bisimulation for timed automata is decidable [12]. We use also this result to show the decidability of the ∇-bisimulation.

Paper Outline. The paper is organized as follows. Section 2 gives the preliminaries recalling timed automata with deadlines, its semantics in terms of transition systems, the definition of bisimulation, and particularly, the definition of parallel composition. In Section 3 we discuss the pitfalls of the composition and progressively construct the semantics that leads to the definition of ∇-bisimulation. The symbolic characterization is provided in Section 4 and shown to be the coarsest congruence in Section 5. We conclude in Section 6 discussing decidability of ∇-bisimulation and the different kind of synchronization in parallel composition. A full version of this paper appeared as [15].

2 Preliminaries

Timed Automata with Deadlines. A *clock* is a non-negative real-valued variable, which can be reset to zero at the occurrence of an event, and between two resets, its derivative with respect to time is equal to 1. We denote $C = \{x_1, \ldots, x_N\}$ to be a finite set of clocks. A *clock constraint* $\mathcal{F}(C)$ is a conjunction of formula(s) of atomic constraints in the form of $x_i \bowtie n$ or $x_i - x_j \bowtie m$, where x_i and x_j are clocks in C, $\bowtie \in \{<, >, \leq, \geq, =\}$ and n, m are natural numbers. The constraints **tt** and **ff** are used to denote, respectively, the atomic constraints which are constantly true and false. We will assume that there is a global finite set of actions $\mathcal{A}$ for all timed automata with deadlines.

Definition 1. *A* timed automaton with deadlines *[6] (TAD for short) is a structure* $T = (\mathcal{L}, l^0, C, \rightarrow)$ *where (i)* $\mathcal{L}$ *is a finite set of locations, (ii)* $l^0 \subseteq \mathcal{L}$ *is the set of initial locations, (iii)* C *is a finite set of clocks, (iv)* $\rightarrow \subseteq \mathcal{L} \times (\mathcal{A} \times \mathcal{F}(C) \times \mathcal{F}(C) \times 2^C) \times \mathcal{L}$*, is a finite set of edges. If* $(s, a, \gamma, \delta, \mathbf{x}, s') \in \rightarrow$ *we write* $s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ *and require that* $\delta \Rightarrow \gamma$ *holds, moreover we assume* δ *is left-closed (left-closure is formally defined in Def. 2).*

The notion $s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ represents an edge from location s to s' that executes action a whenever *guard* γ becomes true. In addition, *deadline* predicate δ impose an urgency condition: the transition cannot be delayed whenever δ is satisfied. When executing the transition, clocks in $\mathbf{x}$ are set to 0.

Parallel composition of TADs. Parallel composition allows the independent execution of the activity of the component automata except if they are intended to synchronize. We assume CSP synchronization in which actions with equal name synchronize if and only if they belong to a set of *synchronizing actions* $B \subseteq \mathcal{A}$. Since enabling of actions is determined by guards, we define the guard on the synchronized transition to be the conjunction of the guards on the synchronizing transitions. Therefore synchronization takes place only if both partners are able to execute the same synchronizing action. (Other compositions of guards are discussed in Sec. 6). Similarly, the deadlines of the synchronizing transitions should affect the deadline of the synchronization. In this case, we do not fix any particular operation. Instead, we assume a given operator $\otimes$ that, when applied to guards and deadlines of the synchronizing transitions, returns the deadline of the synchronization. We require that $\otimes$ satisfies the following:

1. $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) \Rightarrow (\gamma_1 \wedge \gamma_2)$ whenever $\delta_1 \Rightarrow \gamma_1$ and $\delta_2 \Rightarrow \gamma_2$
2. $\otimes$ preserves *left-closure*, that is, if δ_1 and δ_2 are left closed, so is $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2)$
3. $\otimes$ distributes with respect to $\vee$ in all its arguments, that is
   $\left(\bigvee_i \left(\delta_1^i, \gamma_1^i\right) \otimes \left(\delta_2^i, \gamma_2^i\right) \right) \Leftrightarrow \left(\bigvee_i \delta_1^i, \bigvee_i \gamma_1^i\right) \otimes \left(\bigvee_i \delta_2^i, \bigvee_i \gamma_2^i\right)$
4. There exists a constraint $\mathbf{0}_\delta$ such that $(\mathbf{0}_\delta, \mathbf{tt})$ acts as a neutral element for $\otimes$ in the following sense: $((\delta_1, \gamma_1) \otimes (\mathbf{0}_\delta, \mathbf{tt})) \Leftrightarrow \delta_1$

$(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2)$ has to imply the guard $\gamma_1 \wedge \gamma_2$ of the resulting transition in order to preserve this property on the composed TAD. This is required in 1. Similarly, condition 2 ensures that deadlines of the composed TAD are left-closed. The distributivity of 3 is needed to prove congruence (see proof of Theorem 2). As we will see in the next section, time passage in a location is limited by the complement of the disjunction of the outgoing deadlines. Therefore condition 3 states compositionality for $\otimes$, allowing to represent the deadline of a synchronized action in terms of the deadlines and guards of the component automata. Constraint 4 is only necessary to show that our definition is the coarsest congruence included in the bisimulation (see Lemma 6). For operators not meeting this condition there may exist coarser congruences than ours that are also bisimulation. Constraint 4 guarantees a way to test the validity of the original deadline in a component's transition by means of a synchronization. In Sec. 6 we discuss different implementations of $\otimes$.

Let $T_i = (\mathcal{L}_i, \mathsf{l}^0{}_i, C_i, \rightarrow_i)$, such that $C_1 \cap C_2 = \varnothing$ for $i \in \{1, 2\}$, and let $B \subseteq \mathcal{A}$ be a set of *synchronizing actions*, and $\otimes$ be an operation for synchronizing deadlines. The *parallel composition* $T_1 \parallel_B^{\otimes} T_2$ is defined by the TAD $(\mathcal{L}_1 \times \mathcal{L}_2, \mathsf{l}^0{}_1 \times \mathsf{l}^0{}_2, C_1 \cup C_2, \rightarrow)$ where $\rightarrow$ is defined as the smallest relation satisfying:

$$\frac{s_i \xrightarrow{a,\gamma,\delta,\mathbf{x}}_i s_i' \quad s_j = s_j' \quad \{i,j\}=\{1,2\} \quad a \notin B}{(s_1, s_2) \xrightarrow{a,\gamma,\delta,\mathbf{x}} (s_1', s_2')} \qquad \frac{s_1 \xrightarrow{a,\gamma_1,\delta_2,\mathbf{x}_1}_1 s_1' \quad s_2 \xrightarrow{a,\gamma_2,\delta_2,\mathbf{x}_2}_2 s_2' \quad a \in B}{(s_1, s_2) \xrightarrow{a,\gamma_1\wedge\gamma_2,(\delta_1,\gamma_1)\otimes(\delta_2,\gamma_2),\mathbf{x}_1\cup\mathbf{x}_2} (s_1', s_2')}$$

The rules are fairly standard. Notice, in particular, that the last rule only allows to synchronize guards when both of them are valid. This is a significant restriction w.r.t. [6]. We later argue that this is nevertheless reasonable and discuss the feasibility of compositions not consider here. From now on, subscripts on edges will be omited.

Transition Systems and Bisimulation. A *transition system (TS for short)* is a structure $TS = (\mathcal{S}, \mathsf{s}^0, \Sigma, \longrightarrow)$ where $\mathcal{S}$ is an infinite set of states, s^0 is the set of initial states, Σ is a set of labels, and $\longrightarrow \subseteq (\mathcal{S} \times \Sigma \times \mathcal{S})$ is a set of transitions. Since we use TSs to model timed systems, we consider two kind of labels: those representing the execution of discrete actions and those representing the passage of time. Then $\Sigma = \mathcal{A} \cup \mathbb{R}_{\geq 0}$.

A *bisimulation* [24] is a symmetric relation $R \in \mathcal{S} \times \mathcal{S}$ such that for all $a \in \Sigma$, whenever $(p, q) \in R$ and $p \xrightarrow{a} p'$ then $q \xrightarrow{a} q'$ and $(p', q') \in R$ for some q'. We write $p \sim q$ if $(p, q) \in R$ for some bisimulation relation R on TS. Given two TSs TS_1 and TS_2 with set of initial states $\mathsf{s}^0{}_1$ and $\mathsf{s}^0{}_2$, respectively, we say that they are bisimilar (notation $TS_1 \sim TS_2$) if there is a bisimulation R in the disjoint union of $TS_1 \uplus TS_2$ such that $\mathsf{s}^0{}_j \subseteq R(\mathsf{s}^0{}_i)$ for $\{i, j\} = \{1, 2\}$, i.e. every initial state of TS_1 is related to some initial state of TS_2 and vice-versa.

Semantics of TADs. In the following we recall the semantics of TADs in terms of TSs. A state of the timed system is divided in two parts, one indicating the current control

location in the TAD, and the other the current time values. This last part is represented by means of a *clock valuation* which is a function $\rho : C \to \mathbb{R}_{\geq 0}$ mapping to each clock the time elapsed since the last time it was reset to 0. Given a clock valuation ρ and $d \in \mathbb{R}_{\geq 0}$ the function $\rho + d$ denotes the valuation such that for each clock $x \in C$, $(\rho + d)(x) = \rho(x) + d$. The function $\rho\{\mathbf{x}{:=}0\}$ denotes the valuation such that for each clock $x \in \mathbf{x} \cap C$, $\rho\{\mathbf{x}{:=}0\}(x) = 0$, otherwise $\rho\{\mathbf{x}{:=}0\}(x) = \rho(x)$. We first define what it means for a constraint to be left-closed, followed by the semantics of TADs.

Definition 2. *A constraint ϕ is called* left closed *if and only if for all valuations ρ, $\rho \models \neg\phi \;\Rightarrow\; \exists \varepsilon > 0 : \forall \varepsilon' \leq \varepsilon : \rho + \varepsilon' \models \neg\phi$.*

Definition 3. *Let $T = (\mathcal{L}, l^0, C, \rightarrow)$ be a TAD. Its semantics is given by $TS(T) = (\mathcal{L} \times (C \mapsto \mathbb{R}_{\geq 0}), l^0 \times (C \mapsto 0), \mathcal{A} \cup \mathbb{R}_{\geq 0}, \longrightarrow)$, where $\longrightarrow$ is the smallest relation satisfying:*

A1: discrete transition $s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ *and* $\rho \models \gamma$ *implies* $s\rho \xrightarrow{a} s'\rho\{\mathbf{x}{:=}0\}$*; and*

A2: delay transition $\forall d' < d : \; \rho + d' \models tpc(s)$ *implies* $s\rho \xrightarrow{d} s(\rho + d)$

where $tpc(s) = \neg \bigvee\{\delta \mid \exists a, \gamma, \mathbf{x}, s' : s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'\}$ *is the* time progress condition *in s.*

Rule **A1** states that an edge $s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ defines a discrete transition in current location s whenever the guard holds in current valuation ρ. After the transition is taken clocks in $\mathbf{x}$ are set to 0 in the new valuation. According to **A2**, time can progress in s only when $tpc(s)$ is true, that is as long as no deadline of an edge leaving s becomes true. Notice that $tpc(s)$ is required to hold for all $d' < d$ but not for d itself. Therefore it is indistinguishable whether $tpc(s)$ holds in the limit or not. For instance, if $\rho(x) = 0$ both $x < 3$ and $x \leq 3$ hold in all $\rho + d'$ with $d' < 3$. Thus our assumption that deadline has to be specified as left-closed predicate is not a limitation but a preference to avoid technical complications which do not contribute to the work.

As a consequence of Def. 3 the notion of bisimulation extends to TADs straightforwardly: two TADs T_1 and T_2 are bisimilar (notation $T_1 \sim T_2$) if $TS(T_1) \sim TS(T_2)$.

Example. Consider automata T_1 and T_2 of Fig. 1. Using Def. 3 it is routine to check that relation $\{(s_0\{x{:=}d\}, t_0\{x{:=}d\}) \mid 0 \leq d\} \cup \{(s_1\{x{:=}d\}, t_1\{x{:=}d\}) \mid 0 \leq d \leq 3\} \cup \{(s_2\{x{:=}d\}, t_2\{x{:=}d\}) \mid 2 \leq d\}$ is a bisimulation witnessing $T_1 \sim T_2$. Besides, if $\mathsf{stop} = (\{r\}, \{r\}, \varnothing, \varnothing)$, then $T_2 \parallel_a^{\otimes} \mathsf{stop}$ can execute the trace $b\,5\,c$, which is not possible in $(s_0, r)\{x{:=}0\}$. Consequently, $T_1 \parallel_a^{\otimes} \mathsf{stop} \not\sim T_2 \parallel_a^{\otimes} \mathsf{stop}$.

3 Towards a Congruence Relation

In the following we discuss different proposals for congruence until finding a satisfactory definition. All proposals are bisimulation relations on different modifications of the transition system underlying the TAD.

The example in Fig. 1 suggests that action c could be distinguished if time would be allowed to elapse beyond the deadline. Therefore, a first naive proposal would be to let time progress beyond the time progress condition but this would not be compatible with the bisimulation since TADs with different deadlines but equal guards may become equated. So, a modification of this semantics could consider separately a potential time progress by adding a new kind of transition: $s\rho \xrightarrow{[d]} s(\rho + d)$ for all $d \geq 0$.

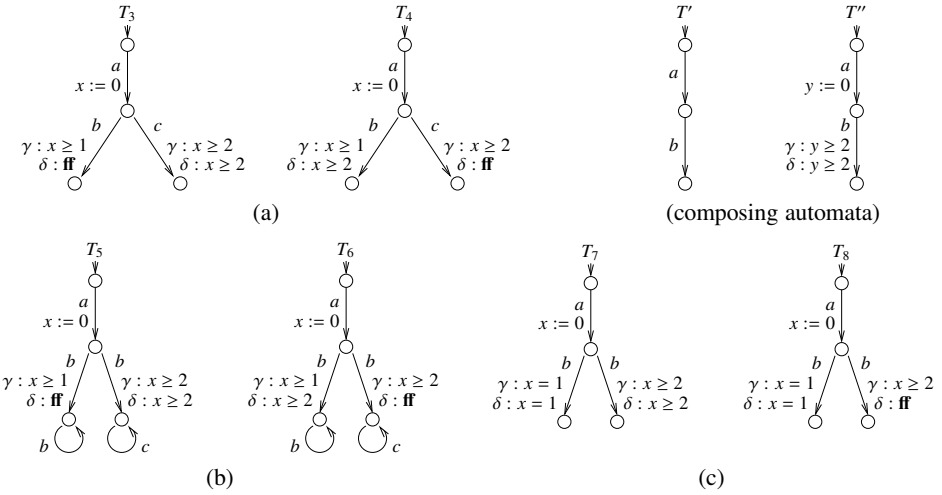

Fig. 2. (Counter)examples for congruence

Though clearly stronger than bisimulation —notice that it would distinguish T_1 and T_2 in Fig. 1— it fails to be a congruence. This is shown in Fig. 2(a). The relation would equate T_3 and T_4, but not their compositions $T_3 \|^{\otimes}_{B} T'$ and $T_4 \|^{\otimes}_{B} T'$ with $B = \{a, b, c\}$. Notice that after realization of action a, $T_3 \|^{\otimes}_{B} T'$ lets (non-potential) time progress beyond 2 time units while this is not possible in $T_4 \|^{\otimes}_{B} T'$ due to the deadline in b.

As a consequence, we may think to consider different potential time progress transition for each edge in the TAD, but this turns to be too strong (apart from cumbersome). See automata T_5 and T_6 in Fig. 2(b) which share some similitude with the previous example, only that c has been renamed to b. They are expected to be congruent.

The new example suggests that time can potentially progress differently for every action name since they can be delayed or preempted independently. A possible solution seems to consider a different kind of potential time progress for each action. Since time progress is associated to deadlines, we follow a different approach: instead of considering potential time progress, we consider a new type of discrete action ∇_D, $D \subseteq \mathcal{A}$, that indicates that from the moment action ∇_D is issued, deadlines of actions in D would be disregarded. We call this type of action "drop" (since it drops the deadline). Notice that a drop action can be performed at any moment.

Let $\mathcal{A}_\nabla = \{\nabla_D \mid D \subseteq \mathcal{A}\}$. To keep track of which deadlines have to be disregarded, states also need to book keep the current set of actions whose deadlines were dropped. The extended semantics of $T = (\mathcal{L}, l^0, C, \rightarrow)$ is then given by the TS $(\mathcal{L} \times 2^{\mathcal{A}} \times (C \mapsto \mathbb{R}_{\geq 0}), l^0 \times \{\varnothing\} \times (C \mapsto 0), \mathcal{A} \cup \mathcal{A}_\nabla \cup \mathbb{R}_{\geq 0}, \longrightarrow)$, where $\longrightarrow$ is the smallest relation satisfying:

A1$_\nabla$: discrete transition $s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ and $\rho \models \gamma$ implies $(s, D)\rho \xrightarrow{a} (s', \varnothing)\rho\{\mathbf{x}{:=}0\}$

A2$_\nabla$: delay transition $\forall d' {<} d : \rho{+}d' \models \neg dl(s, \mathcal{A} - D)$ implies $(s, D)\rho \xrightarrow{d} (s, D)(\rho{+}d)$

A3: drop transition $(s, D)\rho \xrightarrow{\nabla_E} (s, D \cup E)\rho$

where $dl(s, A)$ is the deadline collected by actions in $A \subseteq \mathcal{A}$ in location s and is defined by $dl(s, A) = \bigvee\{\delta \mid s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s'$ and $a \in A$ for some $a, \gamma, \mathbf{x}, s'\}$. Bisimulation in this new

semantics distinguishes automata in Figs. 1(a) and 2(a), and equates those in Fig. 2(b). Regarding to the new predicate $dl(s, A)$ notice that for any location s, $tpc(s) = \neg dl(s, \mathcal{A})$.

Notice that once a deadline is dropped, it cannot be observed anymore. Example in Fig. 2(c) shows that this semantics does not yet yields a congruence. According to this semantics T_7 and T_8 are equated. However, under the assumption that deadlines of synchronizing transitions are arranged in a conjunction (i.e. $\otimes$ is $\wedge$), the compositions $T_7 \parallel_B^{\otimes} T''$ and $T_8 \parallel_B^{\otimes} T''$, with $B = \{a, b\}$, are distinguished by the usual bisimulation: after executing action a, $T_8 \parallel_B^{\otimes} T''$ let time progress beyond 2 time units while this is not the case in $T_7 \parallel_B^{\otimes} T''$ due to the composed deadline $(x \geq 2) \wedge (y \geq 2)$ in b.

This phenomenon is due to the fact that after action a is performed, automaton T'' *temporarily* disregard the deadline of b during the first 2 units of time, but later it allows to observe it again. As a consequence, we introduce a new action Δ (read "undrop") which indicates that in the future all deadlines will be consider again.

Definition 4. *The extended semantics of* $T = (\mathcal{L}, l^0, C, \rightarrow)$ *is given by* $TS_\nabla(T) = (\mathcal{L} \times 2^{\mathcal{A}} \times (C \mapsto \mathbb{R}_{\geq 0}), l^0 \times \{\varnothing\} \times (C \mapsto 0), \mathcal{A} \cup \mathcal{A}_\nabla \cup \{\Delta\} \cup \mathbb{R}_{\geq 0}, \longrightarrow)$*, where* $\longrightarrow$ *is the smallest relation satisfying* $\mathbf{A1}_\nabla$*,* $\mathbf{A2}_\nabla$*, and* $\mathbf{A3}$ *above plus*

A4: undrop transition $\quad (s, D)\rho \xrightarrow{\Delta} (s', \varnothing)\rho$

Notice that the undrop action can be performed at any moment. Notice also that the execution sequence $a\, \nabla_{\{b\}}\, 2\, \Delta\, 1$ is possible in T_8 but not in T_7. Hence, a bisimulation in this setting distinguishes T_7 from T_8. We define such a relation as follows.

Definition 5 (∇-bisimulation). *We say that automata* T_1 *and* T_2 *are* ∇-bisimilar*, notation* $T_1 \sim^\nabla T_2$*, if* $TS_\nabla(T_1) \sim TS_\nabla(T_2)$*. We also say that locations* s *and* t *are* ∇-bisimilar *in some valuation* ρ*, notation* $s\rho \sim^\nabla t\rho$*, if* $(s, \varnothing)\rho \sim (t, \varnothing)\rho$*.*

Notice that two ∇-bisimilar automata are also bisimilar. We conclude this section by stating two basic properties (lemmas) of ∇-bisimulation. They are needed to prove Theorem 1 which relates $\sim^\nabla$ to a symbolic bisimulation.

Notice that the ability of dropping all the deadlines, letting time pass, and then undropping the deadlines, ensures that if two locations are ∇-bisimilar at a certain moment, no matter how long the activity is blocked, this two locations will still be ∇-bisimilar. This is stated in Lemma 1. Moreover, if two locations are ∇-bisimilar at some given valuation ρ then both satisfy the deadline associated to some action in valuation ρ, or none of them does. This is easy to check by dropping all the deadlines except those associated to the action of interest. This is formally stated in Lemma 2.

Lemma 1. *If* $t\rho \sim^\nabla u\rho$ *then* $t(\rho + d) \sim^\nabla u(\rho + d)$*, for all* $d \geq 0$*.*

Lemma 2. *If* $t\rho \sim^\nabla u\rho$ *then* $\rho \models dl(t, D) \Leftrightarrow dl(u, D)$*, for any* $D \subseteq \mathcal{A}$*.*

4 Symbolic Characterization of ∇-bisimulation

We postpone the proof that ∇-bisimulation is a congruence until Sec. 5 and give first a symbolic characterization of $\sim^\nabla$. That is, we give a relation directly in TADs which does not resort to the underlying transition system and equates exactly the same automata

as $\sim^{\nabla}$ does. The symbolic bisimulation we propose works in a similar fashion to that of [23]. The construction of such relation is based on zone and region manipulation. A clock region or *region* for short, is a consistent conjunction of atomic constraints of the form, $\psi \equiv \bigwedge_{x\in C} \psi_x \wedge \bigwedge_{\{x,y\}\subseteq C, x\neq y} \psi_{\{x,y\}}$ where

- each ψ_x is either $x = n$, $m < x < m+1$ or $x > N$, and
- each $\psi_{\{x,y\}}$ is either $x{-}y = n$, $m < x{-}y < m+1$ or $x{-}y > N$.

with n, m, N non-negative integers such that $0 \leq n \leq N$, and $0 \leq m < N$. Regions can be expressed by constraints as we defined above, and any constraint can be expressed as a disjunction of regions. Similar to the clock resetting ($\rho\{\mathbf{x} := 0\}$) and time successor ($\rho{+}d$) of the clock valuation defined earlier, we define below their symbolic counterpart.

Reset: For a constraint ϕ and a set of clocks $\mathbf{x}$, the reset $\phi{\downarrow_{\mathbf{x}}}$ is a predicate such that for all ρ, $\rho \models \phi{\downarrow_{\mathbf{x}}}$ iff $\rho = \rho'\{\mathbf{x} := 0\}$ and $\rho' \models \phi$ for some ρ'

Time successor: For a constraint ϕ, the time successor $\phi{\Uparrow}$ is a predicate such that for all ρ, $\rho \models \phi{\Uparrow}$ iff $\rho = \rho' + d$ and $\rho' \models \phi$ for some ρ' and $d \geq 0$

A constraint ϕ *is* $\Uparrow$*-closed* if and only if $\phi{\Uparrow} \Leftrightarrow \phi$ is valid (i.e. a tautology). The operations above distribute on disjunction and are expressible in terms of constraints (see e.g. [28, 23].) The following facts can be derived from the definitions or have already appear elsewhere [28, 23].

Fact 1. *(1)* Let ψ and ϕ be regions. Let ρ and ρ' be valuations s.t. $\rho \models \psi$ and $\rho' \models \psi$. If $\rho + d \models \phi$ for some $d \geq 0$, there exists $d' \geq 0$ s.t. $\rho' + d' \models \phi$. *(2)* If ϕ is a region then, for any constraint ψ, either $\phi \Rightarrow \psi$ is valid or $\phi \wedge \psi$ is a contradiction. *(3)* If ϕ is a region, so does $\phi{\downarrow_{\mathbf{x}}}$. *(4)* $\rho \models \phi$ implies $\rho \models \phi{\Uparrow}$. *(5)* $\phi{\Uparrow}$ is $\Uparrow$-closed. *(6)* If ϕ is $\Uparrow$-closed then $\rho \models \phi$ implies $\rho + d \models \phi$ for all $d \in \mathbb{R}_{\geq 0}$. *(7)* If ϕ_1 and ϕ_2 are $\Uparrow$-closed (resp. left-closed), so are $\phi_1 \wedge \phi_2$ and $\phi_1 \vee \phi_2$.

Given a constraint ϕ, a ϕ-partition [23] is a finite set of constraints Φ if $\bigvee \Phi \Leftrightarrow \phi$ and for any two distinct $\psi, \psi' \in \Phi$, ψ and ψ' are disjoint (i.e. $\psi \wedge \psi'$ is a contradiction). A ϕ-partition Φ is called *finer* than another ϕ-partition Ψ if Φ can be obtained from Ψ by decomposing some of its elements. $\mathcal{RC}(\phi)$ denotes the set of all regions that constitute ϕ. Notice that $\phi \Leftrightarrow \bigvee \mathcal{RC}(\phi)$ and that $\mathcal{RC}(\phi)$ is the finest of all ϕ-partitions.

Lemma 3. *Let ψ be a region and ρ be such that $\rho \models \psi$. For all $\phi \in \mathcal{RC}(\psi{\Uparrow})$ exists $d \geq 0$ such that $\rho + d \models \phi$.*

The definition of symbolic bisimulation we propose is based on Lin & Yi's definition [23], which in turns is based on Čerāns' result [12]. A symbolic bisimulation is a relation containing tuples (s, t, ϕ) meaning that locations s and t are related in any valuation that satisfies constraint ϕ. Here ϕ is a constraint over the disjoint union of the set of clocks of the two automata. In this way, the relation ensures that clocks in both automata progress at the same rate. In turn, this guarantees that the related locations can idle the same time until some given deadline becomes true.

Definition 6 (Symbolic Bisimulation). *Let T_1 and T_2 be two TADs with disjoint set of clocks C_1 and C_2 and disjoint set of locations $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. A relation $S \subseteq (\mathcal{L}_1 \times \mathcal{L}_2 \cup \mathcal{L}_2 \times \mathcal{L}_1) \times \mathcal{F}(C_1 \cup C_2)$ (where $\mathcal{F}(C)$ denotes the set of all constraints with clocks in C) is a* symbolic bisimulation *if for all $(t, u, \phi) \in S$,*

(1) $(u, t, \phi) \in S$, (2) ϕ is $\Uparrow$-closed,

(3) whenever $t \xrightarrow{a,\gamma,\delta,\mathbf{x}} t'$*, there is a* $(\phi \wedge \gamma)$*-partition* Φ *such that for each* $\phi' \in \Phi$*,* $u \xrightarrow{a,\gamma',\delta',\mathbf{y}} u'$*,* $\phi' \Rightarrow \gamma'$ *and* $(t', u', \phi'\downarrow_{\mathbf{xy}}\Uparrow) \in S$*, for some* γ'*,* δ'*,* $\mathbf{y}$ *and* u'*; and*
(4) $\phi \Rightarrow (dl(t, A) \Leftrightarrow dl(u, A))$ *is valid for all* $A \subseteq \mathcal{A}$*.*

We write $t \sim^{\phi} u$ *if* $(t, u, \phi) \in S$ *for some symbolic bisimulation* S*. We also write* $T_1 \sim^{\phi} T_2$ *if for every initial location* t *of* T_1 *there is an initial location* u *in* T_2 *such that* $t \sim^{\phi} u$*, and the same with the roles of* T_1 *and* T_2 *exchanged.*

Property 1 states the symmetric characteristics of a bisimulation. The requirement that ϕ is $\Uparrow$-closed (property 2) ensures that location t and u show an equivalent behavior any time in the future which is necessary if deadlines are dropped. Property 3 ensures the transfer properties of discrete transitions. This is similar to [23] except that there is no invariant to consider. Finally, property 4 states that any possible combination of deadlines should match under the assumption that ϕ holds. This ensures that the time elapsed until a deadline associated to a given action is the same in both locations. Notice that property 4 is equivalent to requiring that $\phi \Rightarrow (dl(t, \{a\}) \Leftrightarrow dl(u, \{a\}))$ for all $a \in \mathcal{A}$. This makes evident that deadlines may be "changed" from one edge to another as long as both edges are labeled with the same action (see Fig. 2(b)). Moreover property 4 is comparable to the property of invariants in [23]. Like in [23], the use of partitioning allows that one edge is matched by several edges as is the case in Fig. 3 where both $T_9 \sim^{\nabla} T_{10}$ and $T_9 \sim^{x=y} T_{10}$.

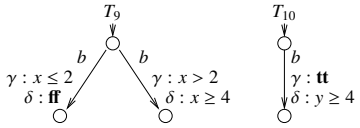

Fig. 3. $T_9 \sim^{x=y} T_{10}$

The following theorem states that symbolic bisimulation completely captures the notion of ∇-bisimulation.

Theorem 1. *For* $\Uparrow$*-closed* ϕ*,* $t \sim^{\phi} u$ *iff* $t\rho \sim^{\nabla} u\rho$ *for any* $\rho \models \phi$

Proof (Sketch). From the results exposed above, it follows that if S be a symbolic bisimulation, then $\{((t, D)\rho, (u, D)\rho) \mid \exists \phi : \rho \models \phi : (t, u, \phi) \in S \text{ and } D \subseteq \mathcal{A}\}$ is a bisimulation up to $\sim$ [24], which proves the "only if". Moreover, it also follows that $\{(t, u, \phi) | \phi \text{ is } \Uparrow \text{ -closed and } \forall \psi \in \mathcal{RC}(\phi) : \exists \rho : \rho \models \psi : t\rho \sim^{\nabla} u\rho\}$ is a symbolic bisimulation, which proves the other implication. □

Corollary 1. *Let* $\phi_0 \equiv \bigwedge_{x,y \in C_1 \cup C_2} (0 \leq x = y)$*.* $T_1 \sim^{\phi_0} T_2$ *iff* $T_1 \sim^{\nabla} T_2$*.*

5 The Coarsest Congruence Included in ∼

In this section, we show that $\sim^{\phi_0}$ (and hence $\sim^{\nabla}$, too) is the coarsest congruence for the parallel composition included in bisimulation. The first part of the section is devoted to prove that $\sim^{\phi_0}$ is a congruence. It is interesting to notice that the proof of congruence is carried out fully at symbolic level (in contrast to the usual proof using the underlying transition system). To the best of our knowledge, this is a novel approach. In the second part we show that $\sim^{\nabla}$ is the coarsest congruence included in $\sim$.

The next two lemmas are required for the proof of congruence. Lemma 4 implies that a deadline of a set of actions can be decomposed as a disjunction of the deadlines

of each of the actions. Lemma 5 states that if two locations t and u are symbolically bisimilar under a constraint ϕ, then a given action a is enabled in t if and only if it is enabled in u for all valuations that satisfy constraint ϕ.

Lemma 4. *$dl(s, D \cup E) \Leftrightarrow (dl(s, D) \vee dl(s, E))$*

Lemma 5. *Define $gd(s, a) = \bigvee\{\gamma \mid s \xrightarrow{a,\gamma,\delta,\mathbf{x}} s' \text{ for some } \delta, \mathbf{x}, s'\}$. If S is a symbolic bisimulation s.t. $(t, u, \phi) \in S$, then $\phi \Rightarrow (gd(t, a) \Leftrightarrow gd(u, a))$ is valid for all $a \in \mathcal{A}$.*

In particular, these lemmas are needed to check that property 4 of the symbolic bisimulation is preserved in the congruence.

Now, we are in conditions to prove that $\sim^\phi$ is a congruence for any parallel composition defined as in Sec. 2. In particular, we notice that the proof does not use constraints 1 and 4 imposed on $\otimes$.

Theorem 2. *Let $T_i^j = (\mathcal{L}_i^j, \mathsf{I}^0{}_i^j, C_i^j, \longrightarrow)$, for $i, j \in \{1, 2\}$ such that $C_i^j \cap C_k^l = \varnothing$ if $i \neq k$ or $j \neq l$. Then $T_1^1 \sim^\phi T_2^1$ and $T_1^2 \sim^\phi T_2^2$ imply $T_1^1 \|_B^\otimes T_1^2 \sim^\phi T_2^1 \|_B^\otimes T_2^2$ for all $B \in \mathcal{A}$, operation $\otimes$ and constraint ϕ.*

Proof (Sketch). Let S_1 and S_2 be symbolic bisimulations witnessing $T_1^1 \sim^{\phi_1} T_2^1$ and $T_1^2 \sim^{\phi_2} T_2^2$, resp. The proof checks that $S = \{((t_1, t_2), (u_1, u_2), \phi_1 \wedge \phi_2) \mid (t_1, u_1, \phi_1) \in S_1$ and $(t_2, u_2, \phi_2) \in S_2\}$ is also a symbolic bisimulation. Properties 1 and 2 in Def. 6 follow easily since S_1 and S_2 also satisfy them. Property 3 follows from the definitions of parallel composition and symbolic bisimulation making careful manipulations of constraints, regions, and partitions using Fact 1. Because of Lemma 4, property 4 is a consequence of implication $(\phi_1 \wedge \phi_2) \Rightarrow (dl((t_1, t_2), \{a\}) \Leftrightarrow dl((u_1, u_2), \{a\}))$, which, for $a \notin B$, follows from the definitions. For $a \in B$, conditions 2 and 3 on $\otimes$ allow to show that $dl((t_1, t_2), \{a\})$ is equivalent to $(dl(t_1, \{a\}), gd(t_1, a)) \otimes (dl(t_2, \{a\}), gd(t_2, a))$, and similarly for (u_1, u_2). Then, by Lemma 5, and since S_1 and S_2 are symbolic bisimulations, $dl((t_1, t_2), \{a\})$ and $dl((u_1, u_2), \{a\})$ can be proved equivalent. □

Because of Corollary 1 and Theorem 2, $\sim^\nabla$ is also a congruence.

The next lemma is core for the proof that $\sim^\nabla$ is the coarsest congruence included in $\sim$. We notice that it does not use constraints 1, 2, and 3 imposed on $\otimes$. The lemma exhibits a test automata T_t that distinguish, modulo bisimulation, two automata that are not ∇-bisimilar. Automata T_t is built by adding extra actions in such a way that, when composed with an automata T, the composition can mimic in the original semantics the behavior of T in the extended semantics. In fact, the extra actions are the same drop (∇_D) and undrop (Δ) actions of the extended semantics.

Lemma 6. *Define the test automata T_t with set of locations $\mathcal{L}_t = \{s_D \mid D \subseteq \mathcal{A}\}$, $\mathsf{I}^0{}_t = \{s_\varnothing\}$, set of clocks $C_t = \varnothing$, set of actions $\mathcal{A} \cup \mathcal{A}_\nabla \cup \{\Delta\}$ and, for all $D, D' \subseteq \mathcal{A}$, $a \notin D$, define $s_D \xrightarrow{a,tt,\mathbf{0}_\delta,\varnothing} s_\varnothing$, $s_D \xrightarrow{\nabla_{D'},tt,ff,\varnothing} s_{D \cup D'}$, and $s_D \xrightarrow{\Delta,tt,ff,\varnothing} s_\varnothing$. Let T_1 and T_2 be TADs with set of locations $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. Suppose that $T_1 \|_\mathcal{A}^\otimes T_t \sim T_2 \|_\mathcal{A}^\otimes T_t$. Then, $R = \{((t_1, D)\rho_1, (t_2, D)\rho_2) \mid t_1 \in \mathcal{L}_1,\ t_2 \in \mathcal{L}_2,\ s_D \in \mathcal{L}_t,$ and $(t_1, s_D)\rho_1 \sim (t_2, s_D)\rho_2\}$ is a bisimulation relation that witnesses $T_1 \sim^\nabla T_2$.*

The proof of the lemma is fairly straightforward except in the case of the delay transition. Notice that a delay transition from (t, D) is governed by satisfaction of $\neg dl(t, \mathcal{A}{-}D)$ (by $\mathsf{A2}_\nabla$) while in (t, s_D), it is governed by $tpc(t, s_D)$. To show that both predicates are equivalent it is necessary that $(\mathbf{0}_\delta, \mathbf{tt})$ is neutral for $\otimes$.

From Lemma 6, it follows that $\sim^\nabla$ and $\sim^{\phi_0}$ are the coarsest congruence in $\sim$:

Theorem 3. *Fix $\otimes$ satisfying conditions 1 and 2 in Sec. 2. Then $\sim^\nabla$ (and hence $\sim^{\phi_0}$) is the coarsest congruence included in $\sim$ for the family of operators $\|^\otimes_B$, with $B \subseteq \mathcal{A}$.*

6 Concluding Remarks

On Deciding ∇-bisimulation. Our symbolic characterisation is based on [23] and [12]. In particular, [12] states that bisimulation is decidable for timed automata. The same applies to our relation. Since the number of regions is finite so is the number of (relevant) constraints (modulo logic equivalence) and as a consequence also the number of relevant $\Uparrow$-closed constraints. Therefore, any possible symbolic bisimulation relating two TADs will also be finite. Besides, operations $\downarrow_{\mathbf{x}}$ and $\Uparrow$ are expressible in terms of constraints, and it is possible to decide validity of the constraints on clocks. Following [12], checking that two TADs T_1 and T_2 are ∇-bisimilarity is then possible by taking relation $S = \{(t, u, \phi\Uparrow) \mid \phi \in \mathcal{RC}(\mathbf{tt})\}$ (which is the finest partition possible since $\mathcal{RC}(\mathbf{tt})$ is the set of all regions) and checking that the transfer rules in Def. 6 hold for all tuples reachable from some set $I \subseteq (S \cap (\mathsf{ini}_1 \times \mathsf{ini}_2 \times \mathcal{RC}(\phi_0)))$ such that it relates all initial states of T_1 (resp. T_2) with some initial state of T_2, (resp. T_1).

A Remark on Symbolic Bisimulation. The third constraint in the definition of symbolic bisimulation (Def. 6) can be relaxed as follows:

> whenever $t \xrightarrow{a,\gamma,\delta,\mathbf{x}} t'$, there is a $(\phi\wedge\gamma)$-partition Φ s.t. for each $\phi' \in \Phi$, $u \xrightarrow{a,\gamma',\delta',\mathbf{y}} u'$, $\phi' \Rightarrow \gamma'$, $\phi'\downarrow_{\mathbf{xy}}\Uparrow \Rightarrow \psi$, and $(t', u', \psi) \in S$, for some ψ, γ', δ', $\mathbf{y}$ and u'.

the difference being on the existence of ψ such that $\phi'\downarrow_{\mathbf{xy}}\Uparrow \Rightarrow \psi$. It is not difficult to check that the new characterisation is equivalent to the original definition. This modification is important since it allows to obtain smaller relations due to the fact that a tuple $(t, u, \phi) \in S$ is redundant if there is a different tuple $(t, u, \phi') \in S$ such that $\phi \Rightarrow \phi'$.

On Synchronising Constraints in Parallel Compositions. In [6] the synchronisation of guards and deadlines of synchronising actions are defined by two operations which we call here $\oplus$ and $\otimes$ respectively. Some conditions are imposed in $\oplus$ and the only condition imposed in $\otimes$ is that $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) \Rightarrow (\gamma_1 \oplus \gamma_2)$ whenever $\delta_1 \Rightarrow \gamma_1$ and $\delta_2 \Rightarrow \gamma_2$ ([6] also suggest that $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) \Rightarrow (\delta_1 \vee \delta_2)$ should hold). We will only discuss here some particular examples that have recurred on the works of Sifakis et al. (see, e.g. [7, 5, 6]). We first focus on the guard:

- $\oplus = \wedge$. This is the one we use and amounts to check that both guards are enables in order to enable the synchronised transition.
- $\oplus = \vee$. The synchronised transition can execute if any of the partners can do so.
- $\oplus = \mathsf{max}$, where $\gamma_1 \mathsf{max} \gamma_2 = (\gamma_1 \wedge \gamma_2\Uparrow) \vee (\gamma_2 \wedge \gamma_1\Uparrow)$. In this case, a component is willing to synchronise if the synchronising transition was enabled in the past and the other component is ready to synchronise now.

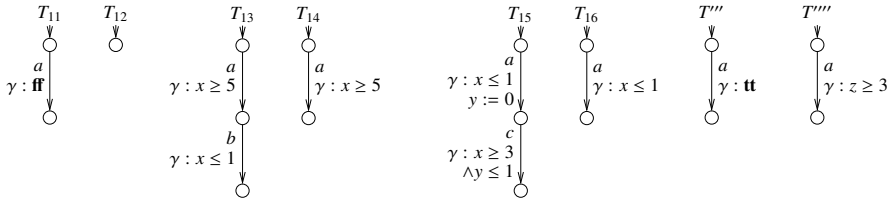

Fig. 4. $T_{11} \sim^{\nabla} T_{12}$, $T_{13} \sim^{\nabla} T_{14}$, and $T_{15} \sim^{\nabla} T_{16}$

$\oplus$ = min, where $\gamma_1 \mathsf{min} \gamma_2 = (\gamma_1 \wedge \gamma_2 \Downarrow) \vee (\gamma_2 \wedge \gamma_1 \Downarrow)$ with $\Downarrow$ being the *time predecessor* operator (the dual of $\Uparrow$). In this case, the synchronised guard anticipates the execution of the synchronising transitions.

Our congruence relation only works for $\wedge$. It is debatable how reasonable are the other operations. Synchronisation through $\vee$ is highly questionable. It is expected that automata T_{11} and T_{12} in Fig. 4 are equivalent under any reasonable criterion. Nevertheless, the composition $T_{11} \|_a^{\otimes} T'''$ can perform action a at any moment while $T_{12} \|_a^{\otimes} T'''$ cannot.

Under min, a component may anticipate the future behaviour of the synchronising partner. [7] and [6] suggest that the intention of this synchronisation is that the earliest synchronising transition makes irrelevant the second one (e.g. a tramway leaves a crossing and after a while it signals to allow the change of the traffic light though it may be ignored if the light has already changed [6]). This intuition does not completely match the behaviour of min which will speed up the slower component allowing it to do activity otherwise impossible. This is observed when automata T''' is composed with T_{13} and with T_{14} synchronising on a (see Fig. 4). Notice that T_{13} and T_{14} exhibit an apparent equal behaviour since action a in T_{13} is always too late to execute b. However, the composition $T_{13} \|_a^{\otimes} T'''$ may hasten the synchronisation on a making b apparent.

Dually, under max, an automata may allow the execution of the synchronising action if it was enabled in the past. Notice that T_{15} and T_{16} in Fig. 4 exhibit equivalent behaviour: c cannot be executed in T_{15} since clock y is always set too early. Instead, the composition with T'''' synchronising on a will delay the execution long enough as to set y sufficiently late to enable the c transition. The intention behind this form of synchronisation is that the fastest component can always wait for the slowest. This design choice seems an adequate choice to use with soft deadlines. Notice also that the appearance of new activity is reasonable since it may be important to cope with the occasional delay. What is debatable is the need of max since this type of synchronisation can easily be represented using $\wedge$: Notice that the max synchronisation does not allow any test automata to distinguish between γ and $\gamma \Uparrow$. Hence, it is more reasonable to model this kind of synchronisation using $\wedge$ instead of max and let all guards be $\Uparrow$-closed.

With respect to deadlines, [6] is more liberal. The two type of synchronising deadlines that stand out are:

Patient synchronisation: $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) = \delta_1 \wedge \delta_2$ with $\mathbf{0}_\delta = \mathbf{tt}$, and
Impatient synchronisation: $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) = (\delta_1 \vee \delta_2) \wedge (\gamma_1 \wedge \gamma_2)$ with $\mathbf{0}_\delta = \mathbf{ff}$.

The nomenclature corresponds to [16] but these definitions are already introduced in [26] with the names of *flexible* and *stiff* respectively. Patient synchronisation allows to model soft deadlines, in the sense that one of the components is always willing to wait for the other (as long as its guards remain valid). On the other hand, impatient synchronisation

impose urgency and obliges the execution as soon as both partners are ready to execute the synchronising transition. Both [26] and [16, 4] give a weaker definition of impatient synchronisation: $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) = \delta_1 \vee \delta_2$. Taking $\mathbf{0}_\delta = \mathbf{ff}$, our result is also valid for this definition. The only problem with it is that it does not preserve time reactivity, i.e. condition 1 on $\otimes$ (see Sec. 2) does not hold[3].

We finally mention that ∇-bisimulation is still a congruence for $\|_B^\otimes$ if condition 4 on $\otimes$ is dropped. However, it is *not* the coarsest congruence in $\sim$ any longer. (This can easily be seen by taking $(\delta_1, \gamma_1) \otimes (\delta_2, \gamma_2) = \mathbf{ff}$).

Conclusions. We have characterised the coarsest congruence for parallel compositions of TADs with soft and hard deadline synchronisation that is included in bisimulation. We also gave a symbolic characterisation of it and show that it is decidable. An aside novelty in our result is that the proof of congruence was entirely carried out in the symbolic semantics rather than resorting to the underlying transition system. The choice on this strategy is not fortuitous. It is mainly due to the complexity on defining an equivalent parallel composition on transition systems. To begin with, any possible definition needs to be tailored for a particular choice of deadline. Besides, it would need complex bookkeeping to know which possible deadline is blocking the passage of time. Many other different complications appear depending on the choice of $\otimes$.

We finally discussed different types of synchronisation in parallel composition and conclude that our choice is both reasonable and sufficiently expressive as to consider the modelling of both soft and hard real-time constraints.

Acknowledgments. We thank Frits Vaandrager for his remarks on early drafts that helped to improve the quality of the paper. Referees are also acknowledged for their useful remarks.

References

1. R. Alur and D. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.
2. G. Behrmann, A. David, K.G. Larsen, O. Möller, P. Pettersson, and Wang Yi. Uppaal – present and future. In *Proc. of 40*th *IEEE Conf. on Decision and Control*. IEEE Press, 2001.
3. J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Uppaal - A tool suite for automatic verification of real-time systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, eds., *Hybrid Systems III: Verification and Control*, *LNCS* 1066, pp. 232–243. Springer, 1996.
4. H. Bohnenkamp, P.R. D'Argenio, H. Hermanns, and J.-P. Katoen. MoDeST: A compositional modeling formalism for real-time and stochastic systems. CTIT Tech. Rep. 04-46, University of Twente, 2004. Submitted for publication.
5. S. Bornot and J. Sifakis. On the composition of hybrid systems. In Thomas A. Henzinger and Shankar Sastry, eds., *Hybrid Systems: Computation and Control, First International Workshop, HSCC'98*, *LNCS* 1386, pp. 49–63. Springer, 1998.
6. S. Bornot and J. Sifakis. An algebraic framework for urgency. *Inf. & Comp.*, 163:172–202, 2000.

[3] To strictly model hard deadlines, this composition requires some modification on the rules in order to ensure the time-blockage produced when a component is ready to synchronise but the other cannot do it at all. A possible solution appears in [4].

7. S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In Roever W.-P. de, H. Langmaack, and A. Pnueli, eds., *Compositionality: The Significant Difference*, *LNCS* 1536, pp. 103–129. Springer, 1998.
8. H. Bowman. Modelling timeouts without timelocks. In J.-P. Katoen, ed., *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop, ARTS'99*, *LNCS* 1601, pp. 334–353. Springer, 1999.
9. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: A model-checking tool for real-time systems. In A.J. Hu and M. Vardi, eds., *Procs. of 10th CAV*, *LNCS* 1427, pp. 546–550. Springer, 1998.
10. M. Bozga and L. Mounier S. Graf. IF-2.0: A validation environment for component-based real-time systems. In E. Brinksma and K.G. Larsen, eds., *Procs. of 14th CAV*, *LNCS* 2404, pp. 343–348. Springer, 2002.
11. M.C. Browne, E.M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *TCS*, 59(1,2):115–131, 1988.
12. K. Čerāns. Decidability of bisimulation equivalences for parallel timer processes. In G. von Bochmann and D.K. Probst, eds., *Procs. of 4th CAV*, *LNCS* 663, pp. 302–315. Springer, 1992.
13. F. Corradini. On performance congruences for process algebras. *Inf. & Comp.*, 145(2):191–230, 1998.
14. P.R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, Department of Computer Science, University of Twente, 1999.
15. P.R. D'Argenio and B. Gebremichael. The coarsest congruence for timed automata with deadlines contained in bisimulation. Tech. Rep. ICIS-R05015. Radboud University Nijmegen, 2005.
16. P.R. D'Argenio, H. Hermanns, J.-P. Katoen, and R. Klaren. MoDeST - a modelling and description language for stochastic timed systems. In L. de Alfaro and S. Gilmore, eds., *Procs. of PAPM-PROBMIV 2001*, *LNCS* 2165, pp. 87–104. Springer, 2001.
17. B. Gebremichael and F.W. Vaandrager. Specifying urgency in timed I/O automata. To appear *In Procs. of 3*rd *IEEE Conference on Software Engineering and Formal Methods*, 2005.
18. T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Inf. & Comp.*, 111:193–244, 1994.
19. H. Hermanns. *Interactive Markov Chains : The Quest for Quantified Quality*, *LNCS* 2428. Springer, 2002.
20. J. Hillston. *A Compositional Approach to Performance Modelling*. Distinguished Dissertation in Computer Science. Cambridge University Press, 1996.
21. H.E. Jensen, K.G. Larsen, and A. Skou. Scaling up UPPAAL automatic verification of real-time systems using compositionality and abstraction. In M. Joseph, ed., *Procs. of FTRTFT 2000*, *LNCS* 1926, pp. 19–30. Springer, 2000.
22. L. Lamport. What good is temporal logic? In R.E. Mason, ed., *Information Processing 83*, pp. 657–668. North-Holland, 1983.
23. H. Lin and W. Yi. Axiomatizing timed automata. *Acta Informatica*, 38(4):277–305, 2002.
24. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
25. R. Segala, R. Gawlick, J.F. Søgaard-Andersen, and N.A. Lynch. Liveness in timed and untimed systems. *Inf. & Comp.*, 141(2):119–171, 1998.
26. J. Sifakis and S. Yovine. Compositional specification of timed systems. In *Procs. of the STACS'96*, *LNCS* 1046, pp. 347–359, Grenoble, France, 1996. Springer.
27. S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Formal Methods in System Design*, 18(1):25–68, 2001.
28. S. Yovine. Model checking timed automata. In G. Rozenberg and F.W. Vaandrager, eds., *Lectures on Embedded Systems*, *LNCS* 1494, pp. 114–152. Springer, 1998.