Avoiding Shared Clocks in Networks of Timed Automata

Sandie Balaguer and Thomas Chatain[*]

INRIA & LSV (CNRS & ENS Cachan), Cachan, France
{balaguer,chatain}@lsv.ens-cachan.fr

Abstract. Networks of timed automata (NTA) are widely used to model distributed real-time systems. Quite often in the literature, the automata are allowed to share clocks. This is a problem when one considers implementing such model in a distributed architecture, since reading clocks a priori requires communications which are not explicitly described in the model. We focus on the following question: given a NTA $A_1 \parallel A_2$ where A_2 reads some clocks reset by A_1, does there exist a NTA $A'_1 \parallel A'_2$ without shared clocks with the same behavior as the initial NTA? For this, we allow the automata to exchange information during synchronizations only. We discuss a formalization of the problem and give a criterion using the notion of contextual timed transition system, which represents the behavior of A_2 when in parallel with A_1. Finally, we effectively build $A'_1 \parallel A'_2$ when it exists.

Keywords: networks of timed automata, shared clocks, implementation on distributed architecture, contextual timed transition system, behavioral equivalence for distributed systems

1 Introduction

Timed automata [3] are one of the most famous formal models for real-time systems. They have been deeply studied and very mature tools are available, like Uppaal [22], Epsilon [16] and Kronos [13].

Networks of Timed Automata (NTA) are a natural generalization to model real-time distributed systems. In this formalism each automaton has a set of clocks that constrain its real-time behavior. But quite often in the literature, the automata are allowed to share clocks, which provides a special way of making the behavior of one automaton depend on what the others do. Actually shared clocks are relatively well accepted and can be a convenient feature for modeling systems. Moreover, since NTA are almost always given a sequential semantics, shared clocks can be handled very easily even by tools: once the NTA is transformed into a single timed automaton by the classical product construction, the notion of distribution is lost and the notion of shared clock itself becomes meaningless. Nevertheless, implementing a model with shared clocks in a distributed architecture is not straightforward since reading clocks a priori requires communications which are not explicitly described in the model.

[*] This work is partially supported by the French ANR project ImpRo.

Our purpose is to identify NTA where sharing clocks could be avoided, i.e. NTA which syntactically use shared clocks, but whose semantics could be achieved by another NTA without shared clocks. We are not aware of any previous study about this aspect. To simplify, we look at NTA made of two automata A_1 and A_2 where only A_2 reads clocks reset by A_1. The first step is to formalize what aspect of the semantics we want to preserve in this setting. Then the idea is essentially to detect cases where A_2 can avoid reading a clock because its value does not depend on the actions that are local to A_1 and thus unobservable to A_2. To generalize this idea we have to compute the knowledge of A_2 about the state of A_1. We show that this knowledge is maximized if we allow A_1 to communicate its state to A_2 each time they synchronize on a common action.

In order to formalize our problem we need an appropriate notion of behavioral equivalence between two NTA. We explain why classical comparisons based on the sequential semantics, like timed bisimulation, are not sufficient here. We need a notion that takes the distributed nature of the system into account. That is, a component cannot observe the moves and the state of the other and must choose its local actions according to its partial knowledge of the state of the system. We formalize this idea by the notion of contextual timed transition systems (contextual TTS).

Then we express the problem of avoiding shared clocks in terms of contextual TTS and we give a characterization of the NTA for which shared clocks can be avoided. Finally we effectively construct a NTA without shared clocks with the same behavior as the initial one, when this is possible. A possible interest is to allow a designer to use shared clocks as a high-level feature in a model of a protocol, and rely on our transformation to make it implementable.

Related work. The semantics of time in distributed systems has already been debated. The idea of localizing clocks has already been proposed and some authors [1,6,19] have even suggested to use local-time semantics with independently evolving clocks. Here we stay in the classical setting of perfect clocks evolving at the same speed. This is a key assumption that provides an implicit synchronization and lets us know some clock values without reading them.

Many formalisms exist for real-time distributed systems, among which NTA [3] and time Petri nets [24]. So far, their expressiveness was compared [7, 12, 15, 26] essentially in terms of sequential semantics that forget concurrency. In [5], we defined a concurrency-preserving translation from time Petri nets to networks of timed automata.

While partial-order semantics and unfoldings are well known for untimed systems, they have been very little studied for distributed real-time systems [11,14]. Partial order reductions for (N)TA were proposed in [6,23,25]. Behavioral equivalence relations for distributed systems, like history-preserving bisimulations were defined for untimed systems only [8, 20].

Finally, our notion of contextual TTS deals with knowledge of agents in distributed systems. This is the aim of epistemic logics [21], which have been extended to real-time in [18,27]. Our notion of contextual TTS also resembles the technique of partitioning states used in timed games with partial observability [9, 17].

Organization of the paper. The paper is organized as follows. Section 2 recalls basic notions about TTS and NTA. Section 3 presents the problem of avoiding shared clocks on examples and rises the problem of comparing NTA component by component. For this, the notion of contextual TTS is developed in Section 4. The problem of avoiding shared clocks is formalized and characterized in terms of contextual TTS. Then Section 5 presents our construction.

The proofs are given in a research report [4].

2 Preliminaries

2.1 Timed Transition Systems

The behavior of timed systems is often described as timed transition systems.

Definition 1. *A* timed transition system *(TTS) is a tuple* $(S, s_0, \Sigma, \rightarrow)$ *where* S *is a set of states,* $s_0 \in Q$ *is the initial state,* Σ *is a finite set of actions disjoint from* $\mathbb{R}_{\geq 0}$*, and* $\rightarrow \subseteq S \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times S$ *is a set of edges.*

For any $a \in \Sigma \cup \mathbb{R}_{\geq 0}$, we write $s \xrightarrow{a} s'$ if $(s, a, s') \in \rightarrow$, and $s \xrightarrow{a}$ if for some s', $(s, a, s') \in \rightarrow$. A *path* of a TTS is a possibly infinite sequence of transitions $\rho = s \xrightarrow{d_0} s'_0 \xrightarrow{a_0} \cdots s_n \xrightarrow{d_n} s'_n \xrightarrow{a_n} \cdots$, where, for all i, $d_i \in \mathbb{R}_{\geq 0}$ and $a_i \in \Sigma$. A path is *initial* if it starts in s_0. A path $\rho = s \xrightarrow{d_0} s'_0 \xrightarrow{a_0} \cdots s_n \xrightarrow{d_n} s'_n \xrightarrow{a_n} s'_n \cdots$ generates a *timed word* $w = (a_0, t_0)(a_1, t_1) \ldots (a_n, t_n) \ldots$ where, for all i, $t_i = \sum_{k=0}^{i} d_k$. The duration of w is $\delta(w) = \sup_i t_i$ and the untimed word of w is $\lambda(w) = a_0 a_1 \ldots a_n \ldots$, and we denote the set of timed words over Σ and of duration d as $\mathrm{TW}(\Sigma, d) = \{w \mid \delta(w) = d \wedge \lambda(w) \in \Sigma^*\}$. Lastly, we write $s \xrightarrow{w} s'$ if there is a path from s to s' that generates the timed word w.

In the following definitions, we use two TTS $T_1 = (S_1, s_1^0, \Sigma_1, \rightarrow_1)$ and $T_2 = (S_2, s_2^0, \Sigma_2, \rightarrow_2)$, and $\Sigma_i^{\not\varepsilon}$ denotes $\Sigma_i \setminus \{\varepsilon\}$, where ε is the silent action.

Product of TTS. The product of T_1 and T_2, denoted by $T_1 \otimes T_2$, is the TTS $\big(S_1 \times S_2, (s_1^0, s_2^0), \Sigma_1 \cup \Sigma_2, \rightarrow\big)$, where $\rightarrow$ is defined as:

- $(s_1, s_2) \xrightarrow{a} (s'_1, s_2)$ iff $s_1 \xrightarrow{a}_1 s'_1$, for any $a \in \Sigma_1 \setminus \Sigma_2^{\not\varepsilon}$,
- $(s_1, s_2) \xrightarrow{a} (s_1, s'_2)$ iff $s_2 \xrightarrow{a}_2 s'_2$, for any $a \in \Sigma_2 \setminus \Sigma_1^{\not\varepsilon}$,
- $(s_1, s_2) \xrightarrow{a} (s'_1, s'_2)$ iff $s_1 \xrightarrow{a}_1 s'_1$ and $s_2 \xrightarrow{a}_2 s'_2$, for any $a \in (\Sigma_1^{\not\varepsilon} \cap \Sigma_2^{\not\varepsilon}) \cup \mathbb{R}_{\geq 0}$.

Timed Bisimulations. Let $\approx$ be a binary relation over $S_1 \times S_2$. We write $s_1 \approx s_2$ for $(s_1, s_2) \in \approx$. $\approx$ is a *strong timed bisimulation* relation between T_1 and T_2 if $s_1^0 \approx s_2^0$ and $s_1 \approx s_2$ implies that, for any $a \in \Sigma \cup \mathbb{R}_{\geq 0}$, if $s_1 \xrightarrow{a}_1 s'_1$, then, for some s'_2, $s_2 \xrightarrow{a}_2 s'_2$ and $s'_1 \approx s'_2$; and conversely, if $s_2 \xrightarrow{a}_2 s'_2$, then, for some s'_1, $s_1 \xrightarrow{a}_1 s'_1$ and $s'_1 \approx s'_2$.

Let $\Rightarrow_i$ (for $i \in \{1, 2\}$) be the transition relation defined as:

- $s \overset{\varepsilon}{\Rightarrow}_i s'$ if $s (\xrightarrow{\varepsilon}_i)^* s'$,
- $\forall a \in \Sigma$, $s \overset{a}{\Rightarrow}_i s'$ if $s (\xrightarrow{\varepsilon}_i)^* \xrightarrow{a}_i (\xrightarrow{\varepsilon}_i)^* s'$,
- $\forall d \in \mathbb{R}_{\geq 0}$, $s \overset{d}{\Rightarrow}_i s'$ if $s (\xrightarrow{\varepsilon}_i)^* \overset{d_0}{\Longrightarrow}_i (\xrightarrow{\varepsilon}_i)^* \cdots \overset{d_n}{\Longrightarrow}_i (\xrightarrow{\varepsilon}_i)^* s'$, where $\sum_{k=0}^{n} d_k = d$.

Then, $\approx$ is a *weak timed bisimulation* relation between T_1 and T_2 if $s_1^0 \approx s_2^0$ and $s_1 \approx s_2$ implies that, for any $a \in \Sigma \cup \mathbb{R}_{\geq 0}$, if $s_1 \xrightarrow{a}_1 s_1'$, then, for some s_2', $s_2 \overset{a}{\Rightarrow}_2 s_2'$ and $s_1' \approx s_2'$; and conversely. We write $T_1 \approx T_2$ (resp. $T_1 \sim T_2$) when there is a strong (resp. weak) timed bisimulation between T_1 and T_2.

2.2 Networks of Timed Automata

The set $\mathcal{B}(X)$ of clock constraints over the set of clocks X is defined by the grammar $g ::= x \bowtie k \mid g \wedge g$, where $x \in X$, $k \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. Invariants are clock constraints of the form $i ::= x \leq k \mid x < k \mid i \wedge i$.

Definition 2. *A* network of timed automata (NTA) *[3] is a parallel composition $A_1 \parallel \cdots \parallel A_n$ of timed automata (TA), with $A_i = (L_i, \ell_i^0, X_i, \Sigma_i, E_i, Inv_i)$ where L_i is a finite set of* locations, *$\ell_i^0 \in L_i$ is the* initial *location, X_i is a finite set of* clocks, *Σ_i is a finite set of* actions, *$E_i \subseteq L_i \times \mathcal{B}(X_i) \times \Sigma_i \times 2^{X_i} \times L_i$ is a set of* edges, *and $Inv_i : L_i \to \mathcal{B}(X_i)$ assigns* invariants *to locations.*

If $(\ell, g, a, r, \ell') \in E_i$, we also write $\ell \xrightarrow{g,a,r} \ell'$. For such an edge, g is the *guard*, a the *action* and r the set of clocks to *reset*. $C_i \subseteq X_i$ is the set of clocks reset by A_i and for $i \neq j$, $C_i \cap C_j$ may not be empty.

Semantics. To simplify, we give the semantics of a network of two TA $A_1 \parallel A_2$. We denote by $((\ell_1, \ell_2), v)$ a *state* of the NTA, where ℓ_1 and ℓ_2 are the current locations, and $v : X \to \mathbb{R}_{\geq 0}$, with $X = X_1 \cup X_2$, is a *clock valuation* that maps each clock to its current value. A state is legal only if its valuation v satisfies the invariants of the current locations, denoted by $v \models Inv_1(\ell_1) \wedge Inv_2(\ell_2)$. For each set of clocks $r \subseteq X$, the valuation $v[r]$ is defined by $v[r](x) = 0$ if $x \in r$ and $v[r](x) = v(x)$ otherwise. For each $d \in \mathbb{R}_{\geq 0}$, the valuation $v + d$ is defined by $(v + d)(x) = v(x) + d$ for each $x \in X$. Then, the *TTS generated by* $A_1 \parallel A_2$ is $\mathrm{TTS}(A_1 \parallel A_2) = (S, s_0, \Sigma_1 \cup \Sigma_2, \to)$, where S is the set of legal states, $s_0 = ((\ell_1^0, \ell_2^0), v_0)$, where v_0 maps each clock to 0, and $\to$ is defined by

- Local action: $((\ell_1, \ell_2), v) \xrightarrow{a} ((\ell_1', \ell_2), v')$ iff $a \in \Sigma_1 \setminus \Sigma_2^{\not\varepsilon}$, $\ell_1 \xrightarrow{g,a,r} \ell_1'$, $v \models g$, $v' = v[r]$ and $v' \models Inv_1(\ell_1')$, and similarly for a local action in $\Sigma_2 \setminus \Sigma_1^{\not\varepsilon}$,
- Synchronization: $((\ell_1, \ell_2), v) \xrightarrow{a} ((\ell_1', \ell_2'), v')$ iff $a \neq \varepsilon$, $\ell_1 \xrightarrow{g_1,a,r_1} \ell_1'$, $\ell_2 \xrightarrow{g_2,a,r_2} \ell_2'$, $v \models g_1 \wedge g_2$, $v' = v[r_1 \cup r_2]$ and $v' \models Inv_1(\ell_1') \wedge Inv_2(\ell_2')$,
- Time delay: $\forall d \in \mathbb{R}_{\geq 0}, ((\ell_1, \ell_2), v) \xrightarrow{d} ((\ell_1, \ell_2), v + d)$ iff $\forall d' \in [0, d], v + d' \models Inv_1(\ell_1) \wedge Inv_2(\ell_2)$.

A *run* of a NTA is an initial path in its TTS. The semantics of a TA A alone can also be given as a TTS denoted by $\mathrm{TTS}(A)$ with only local actions and delay. A TA is *non-Zeno* iff for every infinite timed word w generated by a run, time diverges (i.e. $\delta(w) = \infty$). This is a common assumption for TA. In the sequel, we always assume that the TA we deal with are non-Zeno.

Remark 1. Let $A_1 \parallel A_2$ be such that $X_1 \cap X_2 = \emptyset$. Then $\mathrm{TTS}(A_1) \otimes \mathrm{TTS}(A_2)$ is isomorphic to $\mathrm{TTS}(A_1 \parallel A_2)$. This is not true in general when $X_1 \cap X_2 \neq \emptyset$. For example, in Fig. 2, performing $(b, 0.5)(e, 1)$ is possible in $\mathrm{TTS}(A_1) \otimes \mathrm{TTS}(A_2)$ but not in $\mathrm{TTS}(A_1 \parallel A_2)$, since b resets x which is tested by e.

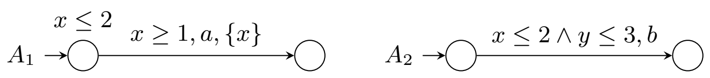

Fig. 1. A_2 could avoid reading clock x which belongs to A_1.

3 Need for Shared Clocks

3.1 Problem Setting

We are interested in detecting the cases where it is possible to avoid sharing clocks, so that the model can be implemented using no other synchronization than those explicitly described by common actions.

To start with, let us focus on a network of two TA, $A_1 \parallel A_2$, such that A_1 does not read the clocks reset by A_2, and A_2 may read the clocks reset by A_1. We want to know whether A_2 really needs to read these clocks, or if another NTA $A'_1 \parallel A'_2$ could achieve the same behavior as $A_1 \parallel A_2$ without using shared clocks.

First remark that our problem makes sense only if we insist on the distributed nature of the system, made of two separate components. On the other hand, if the composition operator is simply used as a convenient syntax for describing a system that is actually implemented on a single sequential component, then a product automaton perfectly describes the system and all clocks become local.

So, let us consider the example of Fig. 1, made of two TA, supposed to describe two separate components. Remark that A_2 reads clock x which is reset by A_1. But a simple analysis shows that this reading could be avoided: because of the condition on its clock y, A_2 can only take transition b before time 3; but x cannot reach value 2 before time 3, since it is reset between time 1 and 2. Thus, forgetting the condition on x in A_2 would not change the behavior of the system.

3.2 Transmitting Information during Synchronizations

Consider now the example of Fig. 2. Here also A_2 reads clock x which is reset by A_1, and here also this reading could be avoided. The idea is that A_1 could transmit the value of x when synchronizing, and A_2 could copy this value locally to a new clock x'. Afterwards, any reading of x in A_2 could be replaced by the reading of x'. Therefore A_2 can be replaced by A'_2 pictured in Fig. 2, while preserving the behavior of the NTA, but also the behavior of A_2 w.r.t. A_1.

We claim that we cannot avoid reading x without this copy of clock. Indeed, after the synchronization, the maximal delay depends on the exact value of x, and even if we find a mechanism to allow A'_2 to move to different locations according to the value of x at synchronization time, infinitely many locations would be required (e.g., if s occurs at time 2, x may have any value in $(1, 2]$).

Coding Transmission of Information. In order to model the transmission of information during synchronizations, we allow A'_1 and A'_2 to use a larger synchronization alphabet than A_1 and A_2. This allows A'_1 to transmit discrete information like its current location, to A'_2.

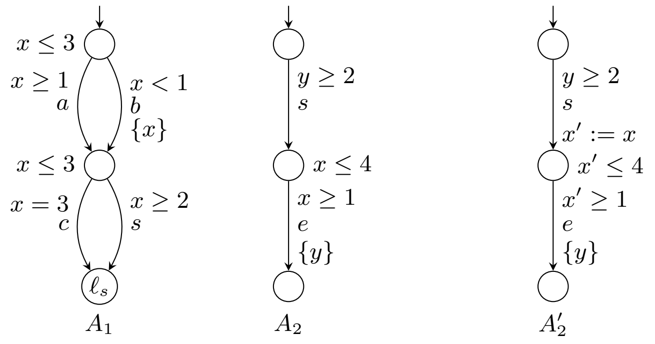

Fig. 2. A_2 reads x which belongs to A_1 and A'_2 does not.

But we saw that A'_1 also needs to transmit the exact value of its clocks. For this we allow an automaton to copy its neighbor's clocks into local clocks during synchronizations. This is denoted as updates of the form $x' := x$ in A'_2 (see Fig. 2). This is a special case of updatable timed automata as defined in [10]. Moreover, as shown in [10], the class we consider, with diagonal-free constraints and updates with equality (they allow other operators) is not more expressive than classical TA for the sequential semantics (any updatable TA of the class is bisimilar to a classical TA), and the emptiness problem is PSPACE-complete.

Semantics. TTS$(A_1 \parallel A_2)$ can be defined as previously, with the difference that the synchronizations are now defined by: $((\ell_1, \ell_2), v) \xrightarrow{a} ((\ell'_1, \ell'_2), v')$ iff $\ell_1 \xrightarrow{g_1,a,r_1}_1 \ell'_1$, $\ell_2 \xrightarrow{g_2,a,r_2,u}_2 \ell'_2$ where u is a partial function from X_2 to X_1, $v \models g_1 \wedge g_2$, $v' = (v[r_1 \cup r_2])[u]$, and $v' \models \mathit{Inv}(\ell'_1) \wedge \mathit{Inv}(\ell'_2)$. The valuation $v[u]$ is defined by $v[u](x) = v(u(x))$ if $u(x)$ is defined, and $v[u](x) = v(x)$ otherwise.

Here, we choose to apply the reset $r_1 \cup r_2$ before the update u, because we are interested in sharing the state reached in A_1 after the synchronization, and r_1 may reset some clocks in $C_1 \subseteq X_1$.

3.3 Towards a Formalization of the Problem

We want to know whether A_2 really needs to read the clocks reset by A_1, or if another NTA $A'_1 \parallel A'_2$ could achieve the same behavior as $A_1 \parallel A_2$ without using shared clocks. It remains to formalize what we mean by "having the same behavior" in this context.

First, we impose that the locality of actions is preserved, i.e. A'_1 uses the same set of local actions as A_1, and similarly for A'_2 and A_2. For the synchronizations, we have explained earlier why we allow A'_1 and A'_2 to use a larger synchronization alphabet than A_1 and A_2. The correspondence between the two alphabets will be done by a mapping ψ (this point will be refined later).

Now we have to impose that the behavior is preserved. The first idea that comes in mind is to impose bisimulation between $\psi(\text{TTS}(A'_1 \parallel A'_2))$ (i.e.

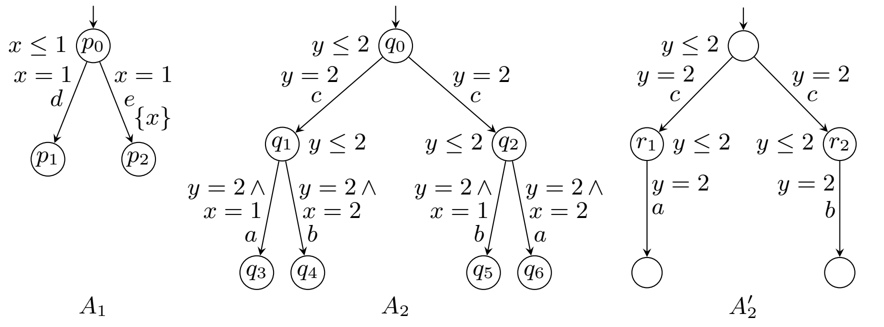

Fig. 3. A_2 needs to read the clocks of A_1 and $\mathrm{TTS}(A_1 \parallel A_2) \sim \mathrm{TTS}(A_1 \parallel A'_2)$.

$\mathrm{TTS}(A'_1 \parallel A'_2)$ with synchronization actions relabeled by ψ) and $\mathrm{TTS}(A_1 \parallel A_2)$. But this is not sufficient, as illustrated by the example of Fig. 3 (where ψ is the identity). Intuitively A_2 needs to read x when in q_1 (and similarly in q_2) at time 2, because this reading determines whether it will perform a or b, and the value of x cannot be inferred from its local state given by q_1 and the value of y. Anyway $\mathrm{TTS}(A_1 \parallel A'_2)$ is bisimilar to $\mathrm{TTS}(A_1 \parallel A_2)$, and A'_2 does not read x. For the bisimulation relation $\mathcal{R}$, it suffices to impose $(p_1, q_1)\ \mathcal{R}\ (p_1, r_1)$ and $(p_2, q_1)\ \mathcal{R}\ (p_2, r_2)$.

What we see here is that, if we focus on the point of view of A_2 and A'_2, these two automata do not behave the same. As a matter of fact, when A_2 fires one edge labeled by c, it has not read x yet, and there is still a possibility to fire a or b, whereas when A'_2 fires one edge labeled by c, there is no more choice afterwards. Therefore we need a relation between A'_2 and A_2, and in the general case, a relation between A'_1 and A_1 also.

4 Contextual Timed Transition Systems

As we are interested in representing a partial view of one of the components, we need to introduce another notion, that we call *contextual timed transition system*. This resembles the powerset construction used in game theory to capture the knowledge of an agent about another agent.

Notations. $\mathbb{S} = \Sigma_1^{\not\varepsilon} \cap \Sigma_2^{\not\varepsilon}$ denotes the set of common actions. Q_1 denotes the set of states of $\mathrm{TTS}(A_1)$. When $s = ((\ell_1, \ell_2), v)$ is a state of $\mathrm{TTS}(A_1 \parallel A_2)$, we also write $s = (s_1, s_2)$, where $s_1 = (\ell_1, v_{|X_1})$ is in Q_1, and $s_2 = (\ell_2, v_{|X_2 \setminus X_1})$, where $v_{|X}$ is v restricted to X.

Definition 3 **($\mathrm{UR}(s)$).** *Let* $\mathrm{TTS}(A_1) = (Q_1, s_0, \Sigma_1, \rightarrow_1)$ *and* $s \in Q_1$. *The set of states of* A_1 *reachable from* s *by local actions in 0 delay (and therefore not observable by* A_2*) is denoted by* $\mathrm{UR}(s) = \{s' \in Q_1 \mid \exists w \in \mathrm{TW}(\Sigma_1 \setminus \Sigma_2^{\not\varepsilon}, 0) : s \xrightarrow{w}_1 s'\}$.

Contextual States. The states of this contextual TTS are called *contextual states*. They can be regarded as possibly infinite sets of states of $\mathrm{TTS}(A_1 \parallel A_2)$ for which A_2 is in the same location and has the same valuation over $X_2 \setminus X_1$. A_2 may not be able to distinguish between some states (s_1, s_2) and (s'_1, s_2). In $\mathrm{TTS}_{A_1}(A_2)$, these states are grouped into the same contextual state. However, when $X_2 \cap X_1 \neq \emptyset$, it may happen that A_2 is able to perform a local action or delay from (s_1, s_2) and not from (s'_1, s_2), even if these states are grouped in a same contextual state.

Definition 4 (Contextual TTS). *Let* $\mathrm{TTS}(A_1 \parallel A_2) = (Q, q_0, \Sigma_1 \cup \Sigma_2, \Rightarrow)$. *Then, the* TTS of A_2 *in the context of* A_1, *denoted by* $\mathrm{TTS}_{A_1}(A_2)$, *is the TTS* $(S, s_0, (\Sigma_2 \setminus \mathbb{S}) \cup (\mathbb{S} \times Q_1), \rightarrow)$, *where*

- $S = \{(S_1, s_2) \mid \forall s_1 \in S_1, (s_1, s_2) \in Q\}$,
- $s_0 = (S_1^0, s_2^0)$, *s.t.* $(s_1^0, s_2^0) = q_0$ *and* $S_1^0 = \mathrm{UR}(s_1^0)$,
- $\rightarrow$ *is defined by*
  - *Local action: for any* $a \in \Sigma_2 \setminus \mathbb{S}$, $(S_1, s_2) \xrightarrow{a} (S'_1, s'_2)$ *iff* $\exists s_1 \in S_1 : (s_1, s_2) \overset{a}{\Rightarrow} (s_1, s'_2)$, *and* $S'_1 = \{s_1 \in S_1 \mid (s_1, s_2) \overset{a}{\Rightarrow} (s_1, s'_2)\}$
  - *Synchronization: for any* $(a, s'_1) \in \mathbb{S} \times Q_1$, $(S_1, s_2) \xrightarrow{a, s'_1} (\mathrm{UR}(s'_1), s'_2)$ *iff* $\exists s_1 \in S_1 : (s_1, s_2) \overset{a}{\Rightarrow} (s'_1, s'_2)$
  - *Local delay: for any* $d \in \mathbb{R}_{\geq 0}$, $(S_1, s_2) \xrightarrow{d} (S'_1, s'_2)$ *iff* $\exists s_1 \in S_1$, $w \in \mathrm{TW}(\Sigma_1 \setminus \Sigma_2^{\not\in}, d) : (s_1, s_2) \overset{w}{\Rightarrow} (s'_1, s'_2)$, *and* $S'_1 = \{s'_1 \mid \exists s_1 \in S_1, w \in \mathrm{TW}(\Sigma_1 \setminus \Sigma_2^{\not\in}, d) : (s_1, s_2) \overset{w}{\Rightarrow} (s'_1, s'_2)\}$

For example, consider A_1 and A_2 of Fig. 3. The initial state is $\big(\{(p_0, 0)\}, (q_0, 0)\big)$. From this contextual state, it is possible to delay 2 time units and reach the contextual state $\big(\{(p_1, 2), (p_2, 1)\}, (q_0, 2)\big)$. Indeed, during this delay, A_1 has to perform either e and reset x, or d. Now, from this contextual state, we can take an edge labeled by c, and reach $\big(\{(p_1, 2), (p_2, 1)\}, (q_1, 2)\big)$. Lastly, from this new state, a can be fired, because it is enabled by $((p_2, 1), (q_1, 2))$ in the TTS of the NTA, and the reached contextual state is $\big(\{(p_2, 1)\}, (q_3, 2)\big)$.

We say that there is no restriction in $\mathrm{TTS}_{A_1}(A_2)$ if whenever a local step is possible from a reachable contextual state, then it is possible from all the states (s_1, s_2) that are grouped into this contextual state. In the example above, there is a restriction in $\mathrm{TTS}_{A_1}(A_2)$ because we have seen that a is enabled only by $((p_2, 1), (q_1, 2))$, and not by all states merged in $\big(\{(p_1, 2), (p_2, 1)\}, (q_1, 2)\big)$. Formally, we use the predicate $\mathit{noRestriction}_{A_1}(A_2)$ defined as follows.

Definition 5 ($\mathit{noRestriction}_{A_1}(A_2)$**).** *The predicate* $\mathit{noRestriction}_{A_1}(A_2)$ *holds iff for any reachable state* (S_1, s_2) *of* $\mathrm{TTS}_{A_1}(A_2)$, *both*

- $\forall a \in \Sigma_2 \setminus \mathbb{S}, (S_1, s_2) \xrightarrow{a} (S'_1, s'_2) \iff \forall s_1 \in S_1, (s_1, s_2) \overset{a}{\Rightarrow} (s_1, s'_2)$, *and*
- $\forall d \in \mathbb{R}_{\geq 0}, (S_1, s_2) \xrightarrow{d} \iff \forall s_1 \in S_1, \exists w \in \mathrm{TW}(\Sigma_1 \setminus \Sigma_2^{\not\in}, d) : (s_1, s_2) \overset{w}{\Rightarrow}$

Remark 2. If A_2 does not read X_1, then $\mathit{noRestriction}_{A_1}(A_2)$.

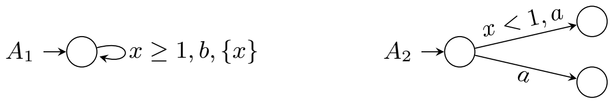

Fig. 4. $\mathrm{TTS}_{Q_1}(A_1) \otimes \mathrm{TTS}_{A_1}(A_2) \approx \mathrm{TTS}_{Q_1}(A_1 \parallel A_2)$, although there is a restriction in $\mathrm{TTS}_{A_1}(A_2)$.

Sharing of Information on the Synchronizations. Later we assume that during a synchronization, A_1 is allowed to transmit all its state to A_2, that is why, in $\mathrm{TTS}_{A_1}(A_2)$, we distinguish the states reached after a synchronization according to the state reached in A_1. We also label the synchronization edges by a pair $(a, s_1) \in \mathbb{S} \times Q_1$ where a is the action and s_1 the state reached in A_1.

For the sequel, let $\mathrm{TTS}_{Q_1}(A_1)$ (resp. $\mathrm{TTS}_{Q_1}(A_1 \parallel A_2)$) denote $\mathrm{TTS}(A_1)$ (resp. $\mathrm{TTS}(A_1 \parallel A_2)$) where the synchronization edges are labeled by (a, s_1), where $a \in \mathbb{S}$ is the action, and s_1 is the state reached in A_1.

We can now state a nice property of unrestricted contextual TTS that is similar to the distributivity of TTS over the composition when considering TA with disjoint sets of clocks (see Remark 1). We say that a TA is *deterministic* if it has no ε-transition and for any location ℓ and action a, there is at most one edge labeled by a from ℓ.

Lemma 1. *If there is no restriction in* $\mathrm{TTS}_{A_1}(A_2)$*, then* $\mathrm{TTS}_{Q_1}(A_1) \otimes \mathrm{TTS}_{A_1}(A_2) \approx \mathrm{TTS}_{Q_1}(A_1 \parallel A_2)$*. Moreover, when A_2 is deterministic, this condition becomes necessary.*

The example of Fig. 4 shows that the reciprocal does not hold when A_2 is not deterministic.

4.1 Need for Shared Clocks Revisited

We have argued in Section 3.3 that the existence of a NTA $A'_1 \parallel A'_2$ without shared clocks and such that $\psi(\mathrm{TTS}_{Q'_1}(A'_1 \parallel A'_2)) \sim \mathrm{TTS}_{Q_1}(A_1 \parallel A_2)$ is not sufficient to capture the idea that A_2 does not need to read the clocks of A_1. We are now equipped to define the relations we want to impose on the separate components, namely $\psi(\mathrm{TTS}_{Q'_1}(A'_1)) \sim \mathrm{TTS}_{Q_1}(A_1)$ and $\psi(\mathrm{TTS}_{A'_1}(A'_2)) \sim \mathrm{TTS}_{A_1}(A_2)$. And since we have seen the importance of using labeling the synchronization actions in contextual TTS by labels in $\mathbb{S} \times Q_1$ rather than in $\mathbb{S}$, the correspondence between the synchronization labels of $A'_1 \parallel A'_2$ with those of $A_1 \parallel A_2$ is now done by a mapping $\psi : \mathbb{S}' \times Q'_1 \to \mathbb{S} \times Q_1$.

This settles the problem of the example of Fig. 3 where $\mathrm{TTS}_{A_1}(A'_2) \not\sim \mathrm{TTS}_{A_1}(A_2)$ (here $A'_1 = A_1$), but as shown in Fig. 5, a problem remains. In this example, we can see that A_2 needs to read clock x of A_1 to know whether it has to perform a or b at time 2, and yet $\mathrm{TTS}_{A_1}(A_2) \sim \mathrm{TTS}_{A_1}(A'_2)$ (here again $A'_1 = A_1$). The intuition to understand this is that the contextual TTS merge too many states for the two systems to remain differentiable. However we remark that

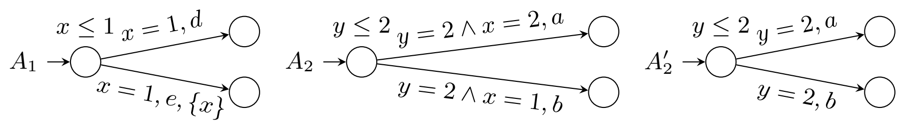

Fig. 5. A_2 needs to read the clocks of A_1 and $\mathrm{TTS}_{A_1}(A_2) \sim \mathrm{TTS}_{A_1}(A'_2)$.

here, the first condition that we have required in Section 3, namely the global bisimulation between $\psi(\mathrm{TTS}(A'_1 \parallel A'_2))$ and $\mathrm{TTS}(A_1 \parallel A_2)$, does not hold.

Now we show that the conjunction of global and local bisimulations actually gives the good definition.

Definition 6 (Need for shared clocks). *Given $A_1 \parallel A_2$ such that A_1 does not read the clocks of A_2,* A_2 does not need to read the clocks of A_1 *iff there exists a NTA $A'_1 \parallel A'_2$ without shared clocks (but with clock copies during synchronizations), using the same sets of local actions and a synchronization alphabet $\mathbb{S}'$ related to the original one by a mapping $\psi : \mathbb{S}' \times Q'_1 \to \mathbb{S} \times Q_1$, and such that*

1. $\psi(\mathrm{TTS}_{Q'_1}(A'_1 \parallel A'_2)) \sim \mathrm{TTS}_{Q_1}(A_1 \parallel A_2)$ *and*
2. $\psi(\mathrm{TTS}_{Q'_1}(A'_1)) \sim \mathrm{TTS}_{Q_1}(A_1)$ *and*
3. $\psi(\mathrm{TTS}_{A'_1}(A'_2)) \sim \mathrm{TTS}_{A_1}(A_2)$.

Notice that this does not mean that the clock constraints that read X_1 can simply be removed from A_2 (see Fig. 2).

Lemma 2. *When noRestriction$_{A_1}(A_2)$ holds, any NTA $A'_1 \parallel A'_2$ without shared clocks and that satisfies items 2 and 3 of Definition 6 also satisfies item 1.*

We are now ready to give a criterion to decide the need for shared clocks.

Theorem 1. *When noRestriction$_{A_1}(A_2)$ holds, A_2 does not need to read the clocks of A_1. When A_2 is deterministic, this condition becomes necessary.*

We remark from the proof that when there is a restriction in $\mathrm{TTS}_{A_1}(A_2)$, even infinite A'_1 and A'_2 would not help. Next section will be devoted to the constructive proof of the direct part of this theorem. The indirect part follows from Lemma 1. The counterexample in Fig. 4 also works here to argue that the conditions of Lemma 2 and Theorem 1 are not necessary when A_2 is not deterministic. Indeed A'_2 with only one unguarded edge labeled by a and $A'_1 = A_1$ satisfy the three items of Definition 6 but there is a restriction in $\mathrm{TTS}_{A_1}(A_2)$.

5 Constructing a NTA without Shared Clocks

This section is dedicated to proving Theorem 1 by constructing suitable A'_1 and A'_2. To simplify, we assume that in A_2, the guards on the synchronizations do not read X_1.

5.1 Construction

First, our A'_1 is obtained from A_1 by replacing all the labels $a \in \mathbb{S}$ on the synchronization edges of A_1 by $(a, \ell_1) \in \mathbb{S} \times L_1$, where ℓ_1 is the output location of the edge. Therefore the synchronization alphabet between A'_1 and A'_2 will be $\mathbb{S}' = \mathbb{S} \times L_1$, which allows A'_1 to transmit its location after each synchronization.

Then, the idea is to build A'_2 as a product $A_{1,2} \otimes A_{2,mod}$ ($\otimes$ denotes the product of TA as it is usually defined [3]), where $A_{2,mod}$ plays the role of A_2 and $A_{1,2}$ acts as a local copy of A'_1, from which $A_{2,mod}$ reads clocks instead of reading those of A'_1. For this, as long as the automata do not synchronize, $A_{1,2}$ will evolve, simulating a run of A'_1 that is compatible with what A'_2 knows about A'_1. And, as soon as A'_1 synchronizes with A'_2, A'_2 updates $A_{1,2}$ to the actual state of A'_1. If the clocks of $A_{1,2}$ always give the same truth value to the guards and invariants of $A_{2,mod}$ than the actual value of the clocks of A'_1, then our construction behaves like $A_1 \parallel A_2$. To check that this is the case, we equip A'_2 with an error location, ☹, and edges that lead to it if there is a contradiction between the values of the clocks of A'_1 and the values of the clocks of $A_{1,2}$. The guards of these edges are the only cases where A'_2 reads clocks of A'_1. Therefore, if ☹ is not reachable, they can be removed so that A'_2 does not read the clocks of A'_1. More precisely, a contradiction happens when $A_{2,mod}$ is in a given location and the guard of an outgoing edge is true according to $A_{1,2}$ and false according to A'_1, or vice versa, or when the invariant of the current location is false according to A'_1 (whereas it is true according to $A_{1,2}$, since $A_{2,mod}$ reads the clocks of $A_{1,2}$).

Namely, $\mathcal{S}_{mod} = A'_1 \parallel (A_{1,2} \otimes A_{2,mod})$ where $A_{1,2}$ and $A_{2,mod}$ are defined as follows. $A_{1,2} = (L_1, \ell_1^0, X'_1, \mathbb{S}' \cup \{\varepsilon\}, E'_1, Inv'_1)$, where

- each clock $x' \in X'_1$ is associated with a clock $c(x') = x \in X_1$ (c is a bijection from X'_1 to X_1). γ' denotes the clock constraint where any clock x of X_1 is substituted by x' of X'_1.
- $\forall \ell \in L_1, Inv'_1(\ell) = Inv_1(\ell)'$
- $E'_1 = \{\ell_1 \xrightarrow{g', \varepsilon_a, r'} \ell_2 \mid \exists a \in \Sigma_1 \setminus \Sigma_2^{\not\varepsilon} : \ell_1 \xrightarrow{g, a, c(r')} \ell_2 \in E_1\}$
  $\cup \ \{\ell \xrightarrow{\top, (a, \ell_2), c} \ell_2 \mid \ell \in L_1 \wedge a \in \mathbb{S} \wedge \exists \ell_1 \xrightarrow{g, a, r} \ell_2 \in E_1\}$
  where $\top$ means true, and c denotes the assignment of any clock $x' \in X'_1$ with the value of its associated clock $c(x') = x \in X_1$ (written $x' := x$ in Fig. 6).

$A_{2,mod} = (L_2 \cup \{☹\}, \ell_2^0, X_2 \cup X'_1, (\Sigma_2 \setminus \Sigma_1) \cup \mathbb{S}', E'_2, Inv'_2)$, where

- $\forall \ell \in L_2, Inv'_2(\ell) = Inv_2(\ell)'$ and $Inv'_2(☹) = \top$,
- $E'_2 = \{\ell_1 \xrightarrow{g', a, r} \ell_2 \mid \ell_1 \xrightarrow{g, a, r} \ell_2 \in E_2 \wedge a \notin \mathbb{S}\}$
  $\cup \ \{\ell_1 \xrightarrow{g, (a, \ell), r} \ell_2 \mid \ell_1 \xrightarrow{g, a, r} \ell_2 \in E_2 \wedge a \in \mathbb{S} \wedge \ell \in L_1\}$
  $\cup \ \{\ell \xrightarrow{\neg Inv_2(\ell), \varepsilon, \emptyset} ☹ \mid \ell \in L_2\}$
  $\cup \ \{\ell \xrightarrow{g' \wedge \neg g, \varepsilon, \emptyset} ☹ \mid \ell \xrightarrow{g, a, r} \ell' \in E_2 \wedge a \notin \mathbb{S}\}$
  $\cup \ \{\ell \xrightarrow{\neg g' \wedge g, \varepsilon, \emptyset} ☹ \mid \ell \xrightarrow{g, a, r} \ell' \in E_2 \wedge a \notin \mathbb{S}\}$.

For the example of Fig. 2, $A_{1,2}$ and $A_{2,mod}$ are pictured in Fig. 6.

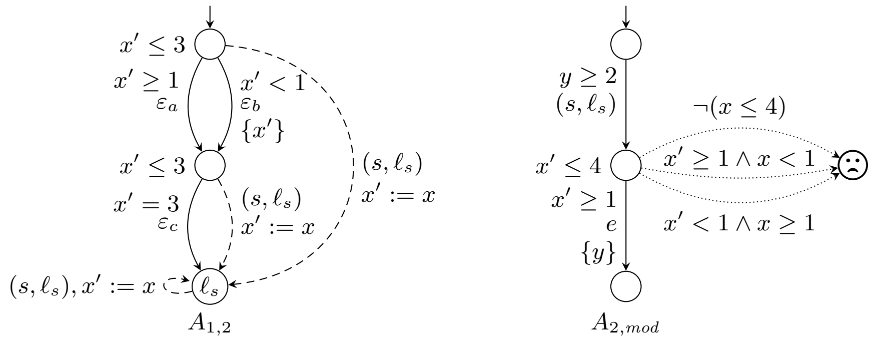

Fig. 6. $A_{1,2}$ and $A_{2,mod}$ for the example of Fig. 2

Lemma 3. *☹ is reachable in $\mathcal{S}_{mod}$ iff there is a restriction in* $\mathrm{TTS}_{A_1}(A_2)$.

We first give a case for which Theorem 1 can be proved easily. We say that A_1 has no urgent synchronization if for any location, when the invariant expires, a local action is enabled. Under this assumption, we show that $A'_2 = A_{1,2} \otimes A'_{2,mod}$, where $A'_{2,mod}$ is $A_{2,mod}$ without location ☹ (that is unreachable by Lemma 3) and its ingoing edges, is suitable. Indeed, A'_2 does not read X_1 and $\psi(\mathrm{TTS}_{A'_1}(A'_2)) \sim \mathrm{TTS}_{A_1}(A_2)$, where for any $((a, \ell_1), s_1) \in \mathbb{S}' \times Q'_1$, $\psi(((a, \ell_1), s_1)) = (a, s_1)$. Item 2 of Definition 6 is immediate, and item 1 holds by Lemma 2.

When A_1 has urgent synchronizations, this construction allows one to check the absence of restriction in $\mathrm{TTS}_{A_1}(A_2)$, but it does not give directly a suitable A'_2. We will give the idea of the construction of A'_2 for the general case later.

In the example of Fig. 2, ☹ is not reachable in $\mathcal{S}_{mod}$ (see Fig. 6), therefore A_2 does not need to read X_1. For an example where ☹ is reachable, consider the same example with an additional edge $\xrightarrow{\top, f, \{x\}}$ from the end location of A_1 to a new location. Location ☹ can now be reached in $\mathcal{S}_{mod}$, for example consider a run where s is performed at time 2 leading to a state where $v(x) = 2$ and $v(x') = 2$, and then A_1 immediately performs f and resets x, leading to a state where the valuation v' is such that $v'(x) = 0$ and $v'(x') = 2$, and satisfies guard $x' \geq 1 \wedge x < 1$ in $\mathcal{S}_{mod}$. Therefore, with this additional edge in A_1, A_2 needs to read X_1. Indeed, without this edge, A_2 knows that A_1 cannot modify x after the synchronization, but with this edge, A_2 does not know whether A_1 has performed f and reset x, while this may change the truth value of its guard $x \geq 1$.

5.2 Complexity

The reachability problem for timed automata is known to be PSPACE-complete [2]. We will reduce this problem to our problem of deciding whether A_2 needs to read the clocks of A_1. Consider a TA A over alphabet Σ, with some location ℓ. Build the TA A_2 as A augmented with two new locations ℓ' and ℓ'' and two edges, $\ell \xrightarrow{\top, \varepsilon, \emptyset} \ell'$ and $\ell' \xrightarrow{x=1, a, \emptyset} \ell''$, where x is a fresh clock, and a is some action in Σ. Let A_1 be the one of Fig. 4 with an action $b \notin \Sigma$. Then, ℓ is reachable in A iff A_2 needs to read x which belongs to A_1. Therefore the problem of deciding whether A_2 needs to read the clocks of A_1 is also PSPACE-hard.

Moreover, we can show that when A_2 is deterministic, our problem is in PSPACE. Indeed, by Theorem 1 and Lemma 3, ☹ is not reachable iff $\mathit{noRestriction}_{A_1}(A_2)$ iff A_2 does not need to read the clocks of A_1. Since the size of the modified system on which we check the reachability of ☹ is polynomial in the size of the original system, our problem is in PSPACE.

5.3 Dealing with Urgent Synchronizations

If we use exactly the same construction as before and allow urgent synchronizations, the following problem may occur. Remind that $A_{1,2}$ simulates a possible run of A'_1 while A'_1 plays its actual run. There is no reason why the two runs should coincide. Thus it may happen that the run simulated by $A_{1,2}$ reaches a state where the invariant expires and only a synchronization is possible. Then A'_2 is expecting a synchronization with A'_1, but it is possible that the actual A'_1 has not reached a state that enables this synchronization. Intuitively, A'_2 should then realize that the simulated run cannot be the actual one and try another run compatible with the absence of synchronization.

But it is simpler to avoid this situation, by forcing $A_{1,2}$ to simulate one of the runs of A'_1 (from the state reached after the last synchronization) that has maximal duration[1] before it synchronizes again with $A_{2,mod}$ (or never synchronizes again if possible). This choice of a run of A'_1 is as valid as the others, and subtle situation described above.

For example, consider automaton A_1 in Fig. 2 without the edge labeled by c and with guard $x \leq 1$ instead of $x < 1$. We can see that $A_{1,2}$ has to fire b at time 1 and is able to wait 3 time units before synchronizing, although it is still able to synchronize at any time (we add the same dashed edges as in Fig. 6). This can be generalized for any A_1. The idea is essentially to force $A_{1,2}$ to follow the appropriate finite or ultimately periodic path in the region automaton [3] of A_1.

6 Conclusion

We have shown that in a distributed framework, when locality of actions and synchronizations matter, NTA with shared clocks cannot be easily transformed into NTA without shared clocks. The fact that the transformation is possible can be characterized using the notion of contextual TTS which represents the knowledge of one TA about the other. Checking the resulting criterion is PSPACE-complete.

One conclusion is that, contrary to what happens when one considers the sequential semantics, NTA with shared clocks are strictly more expressive if we take distribution into account. This somehow justifies why shared clocks were introduced: they are actually more than syntactic sugar.

[1] There may not be any maximum if some time constraints are strict inequalities, but the idea can be adapted even to this case.

Another interesting point is the use of transmitting information during synchronizations. It is noticeable that infinitely precise information is required in general. This advocates the interest of updatable (N)TA used in an appropriate way, and more generally gives a flavor of a class of NTA closer to implementation.

Perspectives. Our first perspective is to generalize our result to the symmetrical case where A_1 also reads clocks from A_2. Then of course we can tackle general NTA with more than two automata.

Another line of research is to focus on transmission of information. The goal would be to minimize the information transmitted during synchronizations, and see for example where are the limits of finite information. Even when infinitely precise information is required to achieve the exact semantics of the NTA, it would be interesting to study how this semantics can be approximated using finitely precise information.

Finally, when shared clocks are necessary, one can discuss how to minimize them, or how to implement the model on a distributed architecture and how to handle shared clocks with as few communications as possible.

References

1. Akshay, S., Bollig, B., Gastin, P., Mukund, M., Narayan Kumar, K.: Distributed timed automata with independently evolving clocks. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 82–97. Springer, Heidelberg (2008)
2. Alur, R., Dill, D.: Automata for modeling real-time systems. In: Paterson, M. (ed.) ICALP 1990, LNCS, vol. 443, pp. 322–335. Springer, Heidelberg (1990)
3. Alur, R., Dill, D.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)
4. Balaguer, S., Chatain, Th.: Avoiding shared clocks in networks of timed automata. Rapport de recherche 7990, INRIA (2012)
5. Balaguer, S., Chatain, Th., Haar, S.: A concurrency-preserving translation from time Petri nets to networks of timed automata. FMSD (2012)
6. Bengtsson, J., Jonsson, B., Lilius, J., Yi, W.: Partial order reductions for timed systems. In: CONCUR 1998. LNCS, vol. 1466, pp. 485–500. Springer, Heidelberg (1998)
7. Bérard, B., Cassez, F., Haddad, S., Lime, D., Roux, O.H.: Comparison of the expressiveness of timed automata and time Petri nets. In: Pettersson, P., Yi, W. (eds.) FORMATS 2005. LNCS, vol. 3829, pp. 211–225. Springer, Heidelberg (2005)
8. Best, E., Devillers, R.R., Kiehn, A., Pomello, L.: Concurrent bisimulations in petri nets. Acta Inf. 28(3), 231–264 (1991)
9. Bouyer, P., D'Souza, D., Madhusudan, P., Petit, A.: Timed control with partial observability. In: Hunt, Jr, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 180–192. Springer, Heidelberg (2003)
10. Bouyer, P., Dufourd, C., Fleury, E., Petit, A.: Updatable timed automata. Theor. Comput. Sci. 321(2-3), 291–345 (2004)
11. Bouyer, P., Haddad, S., Reynier, P.A.: Timed unfoldings for networks of timed automata. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 292–306. Springer, Heidelberg (2006)

12. Boyer, M., Roux, O.H.: On the compared expressiveness of arc, place and transition time Petri nets. Fundam. Inform. 88(3), 225–249 (2008)
13. Bozga, M., Daws, C., Maler, O., Olivero, A., Tripakis, S., Yovine, S.: Kronos: a model-checking tool for real-time systems. In: Hu, A.J., Vardi, M.Y. (eds.) CAV 1998. LNCS, vol. 1427, pp. 546–550. Springer, Heidelberg (1998)
14. Cassez, F., Chatain, Th., Jard, C.: Symbolic unfoldings for networks of timed automata. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 307–321. Springer, Heidelberg (2006)
15. Cassez, F., Roux, O.H.: Structural translation from time Petri nets to timed automata. Jour. of Systems and Software (2006)
16. Cerans, K., Godskesen, J.C., Larsen, K.G.: Timed modal specification - theory and tools. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 253–267. Springer, Heidelberg (1993)
17. David, A., Larsen, K.G., Li, S., Nielsen, B.: Timed testing under partial observability. In: ICST. pp. 61–70. IEEE Computer Society (2009)
18. Dima, C.: Positive and negative results on the decidability of the model-checking problem for an epistemic extension of timed ctl. In: TIME. pp. 29–36. IEEE Computer Society (2009)
19. Dima, C., Lanotte, R.: Distributed time-asynchronous automata. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) ICTAC 2007. pp. 185–200. LNCS, Springer, Heidelberg (2007)
20. van Glabbeek, R.J., Goltz, U.: Refinement of actions and equivalence notions for concurrent systems. Acta Inf. 37(4/5), 229–327 (2001)
21. Halpern, J.Y., Fagin, R., Moses, Y., Vardi, M.Y.: Reasoning About Knowledge. MIT Press (1995)
22. Larsen, K.G., Pettersson, P., Yi, W.: Uppaal in a nutshell. Jour. on Software Tools for Technology Transfer 1(1-2), 134–152 (1997)
23. Lugiez, D., Niebert, P., Zennou, S.: A partial order semantics approach to the clock explosion problem of timed automata. Theor. Comput. Sci. 345(1), 27–59 (2005)
24. Merlin, P.M., Farber, D.J.: Recoverability of communication protocols – implications of a theorical study. IEEE Transactions on Communications 24 (1976)
25. Minea, M.: Partial order reduction for model checking of timed automata. In: Baeten, J.C.M., Mauw, S. (eds.) CONCUR 1999. LNCS, Heidelberg, vol. 1664, pp. 431–446. Springer (1999)
26. Srba, J.: Comparing the expressiveness of timed automata and timed extensions of Petri nets. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 15–32. Springer, Heidelberg (2008)
27. Wozna, B., Lomuscio, A.: A logic for knowledge, correctness, and real time. In: Leite, J.A., Torroni, P. (eds.) CLIMA 2004. LNCS, vol. 3487, pp. 1–15. Springer, Heidelberg (2004)