Analysis of Timed Systems Based on Time-Abstracting Bisimulations

S. Tripakis* and S. Yovine*

VERIMAG, France

Abstract. We adapt a generic minimal model generation algorithm to compute the coarsest finite model of the underlying infinite transition system of a timed automaton. This model is minimal modulo a time-abstracting bisimulation. Our algorithm uses a refinement method that avoids set complementation, and is considerably more efficient than previous ones. We use the constructed minimal model for verification purposes by defining abstraction criteria that allow to further reduce the model and to compare it to a specification.

1 Introduction

Behavioral equivalences based on bisimulation relations have proven useful for verifying the correctness of concurrent systems. They allow comparing an implementation to a usually more abstract specification both represented as labeled transition systems. This approach also allows reducing the size of the system by identifying equivalent states which is crucial to avoid the explosion of the state-space. Since the introduction of strong bisimulation in [Mil80], many equivalences have been defined. Moreover, practice followed theory and several algorithms and tools have been developed.

Despite this fact, behavioral equivalences have not been thoroughly studied in the framework of timed systems. In particular, there is a lack of tools based on this approach. The transition system modeling the behavior of a timed system comprises two kinds of transitions, namely *timeless actions* representing the discrete evolutions of the system, and *time lapses* corresponding to the passage of time. Due to density of time, there are infinitely many time transitions. A finite model can be obtained by defining an appropriate equivalence relation inducing a finite number of equivalence classes. Examples of such relations are the *region-graph* equivalence [AD94] and the *ta-bisimulation* [LY93]. The main idea behind these relations is that they abstract away from the exact amount of time elapsed and they are therefore refer to as *time-abstracting* equivalences.

An important problem consists in constructing the quotient of a labeled transition system w.r.t. an equivalence relation. Many generic algorithms exist to solve this problem, e.g. [BFH+92, LY92]. For timed systems represented by timed automata [AD94], these algorithms have been adapted for computing the

* E-mail: {Stavros.Tripakis,Sergio.Yovine}@imag.fr. Tel: +33 76 90 96 30. Fax: +33 76 41 36 20. Miniparc-Zirst, Rue Lavoisier, 38330 Montbonnot St. Martin.

minimal region graph in [ACD+92b, ACD+92a]. Based on the results reported in [ACD+92a] it comes out that straightforward implementations of those algorithms result in poor performances. In fact, one main obstacle towards efficiency is the cost of computing set complementation.

In this paper, we adapt the generic minimal model generation algorithm of [BFH+92] in order to avoid set complementation, in the spirit of [YL93]. Experimental results carried out on several benchmarks show that this algorithm is more efficient than the ones implemented in [ACD+92a]. Furthermore, we use the constructed minimal model for verification purposes by defining an appropriate abstraction criterion that allows using the tool ALDEBARAN [FGM+92] for further reducing the transition system or comparing it to a specification.

2 Background

2.1 Bisimulations, models, and minimal models

A *model* (or LTS) is a triple $\langle Q, Q^0, \rightarrow \rangle$. Q is a set of states, $Q^0 \subseteq Q$ is the set of initial states, and $\rightarrow \subseteq Q \times L \times Q$ is a set of labeled transitions, for some label set L. We write $q \xrightarrow{l} q'$ instead of $(q, l, q') \in \rightarrow$. A relation $r \subseteq Q \times Q$ is a *bisimulation* iff : $\forall (q_1, q_2) \in r$, $\forall l \in L$,

(1) $\forall q_1' \in Q$ s.t. $q_1 \xrightarrow{l} q_1'$, $\exists q_2'$ s.t. $q_2 \xrightarrow{l} q_2'$ and $(q_1', q_2') \in r$, and

(2) $\forall q_2' \in Q$ s.t. $q_2 \xrightarrow{l} q_2'$, $\exists q_1'$ s.t. $q_1 \xrightarrow{l} q_1'$ and $(q_1', q_2) \in r$.

From now on, $\approx$ denotes the greatest bisimulation. Two models G_1, G_2, $G_i = \langle Q_i, Q_i^0, \rightarrow_i \rangle$, $i = 1, 2$, are bisimilar, denoted $G_1 \approx G_2$, if $\forall q_1 \in Q_1^0, q_2 \in Q_2^0$, $q_1 \approx q_2$.

Let $G = \langle Q, Q^0, \rightarrow \rangle$. A *partition* Π of Q is a set of disjoint *classes* $B \subseteq Q$, the union of which yields Q. The quotient of G w.r.t. Π is $\langle \Pi, \pi, \rightarrow \rangle$, where $\pi = \{B \in \Pi \mid B \cap Q^0 \neq \emptyset\}$, and $B \xrightarrow{l} C$ iff $pre_l(B, C) \neq \emptyset$, where $pre_l(B, C) = \{q \in B \mid \exists q' \in C.\ q \xrightarrow{l} q'\}$. We write $B \rightarrow C$ if $\exists l \in L.\ B \xrightarrow{l} C$. We define $Succs_\pi(B) = \bigcup_{l \in L} Succs_\Pi^l(B)$ where $Succs_\Pi^l(B) = \{C \in \Pi \mid B \xrightarrow{l} C\}$ is the set of successors of B by l, and $Preds_\pi(B) = \bigcup_{l \in L} Preds_\Pi^l(B)$ where $Preds_\Pi^l(B) = \{C \in \Pi \mid C \xrightarrow{l} B\}$ is the set of predecessors of B by l.

B is *stable* w.r.t. C if $\forall l \in L.\ pre_l(B, C) \in \{B, \emptyset\}$. B is stable w.r.t. Π if it is stable w.r.t. all classes $C \in \Pi$. Π is stable if all its classes are stable w.r.t. Π. Let $\Pi_\approx$ be the partition induced by $\approx$. Clearly, $\Pi_\approx$ is stable. The *minimal model* of G modulo bisimulation, is the quotient of G w.r.t. $\Pi_\approx$, denoted $G_\approx$. Notice that $\forall l \in L, B, C \in \Pi_\approx$, $B \xrightarrow{l} C$ iff $pre_l(B, C) = B$.

2.2 A general minimal model generation algorithm

We recall here the generic algorithm developed in [BFH+92] (referred to as MMGA) for computing the reachable part of the minimal model $G_\approx$.

$\Pi := \Pi_0$; $\alpha := \{B \in \Pi_0 \mid B \cap Q^0 \neq \emptyset\}$; $\sigma := \emptyset$;
while $(\exists B \in \alpha \setminus \sigma)$ **do** { (0)
$C_B := Split(B, \Pi)$; (1)
if $(C_B = \{B\})$ **then** { (2)
$\sigma := \sigma \cup \{B\}$; $\alpha := \alpha \cup Succs^l_\Pi(B)$; (3)
} **else** { (4)
$\alpha := \alpha \setminus \{B\}$; $\Pi := (\Pi \setminus \{B\}) \cup C_B$; $\sigma := \sigma \setminus Preds_\pi(B)$; (5)
if $B \cap Q^0 \neq \emptyset$ **then** $\alpha := \alpha \cup \{C \in C_B, \mid C \cap Q^0 \neq \emptyset\}$;
} }

Π denotes the current partition, α the set of accessible classes (i.e., containing at least one accessible state), and $\sigma \subseteq \alpha$ the set of stable accessible classes. $Split(B, \Pi)$ refines the class B by choosing a class C w.r.t. which B is potentially unstable, then computing $B_1 = pre_l(B, C)$, $B_2 = B \cap \overline{pre_l(B, C)}$. If indeed $B_i \neq \emptyset, i = 1, 2$, B is *effectively* split (4), and its predecessors become unstable (5). Otherwise (2), B is both accessible (i.e., it contains a reachable state, say q) and stable, meaning that each one of its successors C has a state q' such that $q \rightarrow q'$. Thus, C contains at least one reachable state, and can be inserted to α (3). Termination depends on whether $\approx$ induces a finite partition of the initial model.

2.3 Avoiding complementation

In the context of timed systems set complementation is very costly and should be avoided. This can be done following the idea presented in [YL93]. Let us first illustrate it with an example. Assume that $B \in \alpha$ is found stable, so that one of its successors, C, becomes accessible, and is split into C_1, C_2, C_3, thus B is no longer stable. Now, instead of splitting B w.r.t. only one of the C_i's, which would yield $\{pre_l(B, C_i), B \cap \overline{pre_l(B, C_i)}\}$, B can be split directly into B_1, B_2, B_3, where $B_i = pre_l(B, C_i)$ (possibly, some B_i's are empty). Now, let

$$Ref^l_\Pi(B) \stackrel{\text{def}}{=} \{B' \mid \exists C \in Succs^l_\Pi(B).\ B' = pre_l(B, C) \wedge B' \neq \emptyset\}.$$

Assuming that whenever $Ref^l_\Pi(B) \notin \{\emptyset, \{B\}\}$, the classes in $Ref^l_\Pi(B)$ satisfy :
(1) *coverness*: $\bigcup Ref^l_\Pi(B) = B$, and
(2) *disjointness*: $\forall B', B'' \in Ref^l_\Pi(B)$, if $B' \neq B''$ then $B' \cap B'' = \emptyset$,
the function *Split* can be redefined as follows:

$$Split(B, \Pi) = \begin{cases} Ref^l_\Pi(B) & if\ \ \exists l \in L.\ Ref^l_\Pi(B) \notin \{\emptyset, \{B\}\} \\ \{B\} & otherwise \end{cases}$$

which does not require using set complementation.

3 Timed systems

3.1 Timed automata

Let $\Omega = \{x_1, ..., x_n\}$ be a finite set of *clocks*. All clocks advance at the same rate. A *valuation* is an n-tuple $v \in \mathbb{R}^n_+$. $v(x_i)$ is the value of clock x_i in v, and

$v+t$, $t \in \mathbb{R}_+$ stands for the valuation v', such that $\forall x \in \Omega.\ v'(x) = v(x) + t$, and $v[\mathcal{X} := 0]$, $\mathcal{X} \subseteq \Omega$ is the valuation v'', such that $v''(x) = 0$ if $x \in \mathcal{X}$, $v''(x) = v(x)$ otherwise. A *clock constraint* ψ is a conjunction of atoms of the form $x \# c$, where $x \in \Omega$, $c \in \mathbb{Z}$, $\# \in \{<, \leq, =, \geq, >\}$.

A *timed automaton* is a quadruple $\langle S, s_0, E, I, \Omega \rangle$. S is a finite set of *control states*, $s_0 \in S$ being the *initial* one. E is a finite set of *arcs*, where an arc $(s, a, s', \psi, \mathcal{X})$ from s to s', is annotated with a label $a \in L$, a clock constraint ψ, and a set of clocks $\mathcal{X} \subseteq \Omega$ to reset. I is a function associating with each control state s an *invariant*. The semantics of a TA is a LTS $G = \langle Q, Q^0, \rightarrow \rangle$, where: $Q = \{\langle s, v \rangle \mid s \in S, v \in I_s\}$; $Q^0 = \{\langle s_0, v \rangle \mid v \in I_{s_0}\}$; and $\rightarrow \subseteq Q \times (E \cup \mathbb{R}_+) \times Q$ is defined by the following rules :

1. (time passage) $$\frac{v, (v+t) \in I_s\ ,\quad t \in \mathbb{R}_+}{\langle s, v \rangle \xrightarrow{t} \langle s, v+t \rangle}$$
2. (action) $$\frac{e = (s, a, s', \psi, \mathcal{X}) \in E,\quad v \in \psi,\quad v' = v[\mathcal{X} := 0]}{\langle s, v \rangle \xrightarrow{e} \langle s', v' \rangle}$$

For $q = \langle s, v \rangle$, $q[\mathcal{X} := 0]$ denotes $\langle s, v[\mathcal{X} := 0] \rangle$, and $q + t$ stands for $\langle s, v + t \rangle$.

3.2 Tai-bisimulation

Given $G = \langle Q, Q^0, \rightarrow \rangle$ we define $G_{tai} = \langle Q, Q^0, \Rightarrow_{tai} \rangle$ by abstracting away the exact amount of time elapsed in a time transition. This is done by replacing all labels $t \in \mathbb{R}^+$ by the label $\varepsilon \notin (E \cup \mathbb{R}^+)$ as follows:

$$\frac{q \xrightarrow{e} q'}{q \overset{e}{\Rightarrow}_{tai} q'} \qquad \frac{q \xrightarrow{t} q'}{q \overset{\varepsilon}{\Rightarrow}_{tai} q'}$$

The tai-bisimulation,[2] denoted $\approx_{tai}$, is the greatest bisimulation defined on G_{tai}, that is, $G \approx_{tai} G'$ iff $G_{tai} \approx G'_{tai}$.

It can be easily shown that $\approx_{ta}$ is coarser than the *region graph* equivalence [AD94] which induces a finite partition. Thus, we can state the following.

Proposition 1. *The partition induced by the tai-bisimulation is finite.*

4 Minimization with respect to the tai-bisimulation

The set of valuations Z satisfying a clock constraint is a simple convex polyhedron, called a *convex zone*. A *(non-convex) zone* is a union of convex zones. The class of zones is closed under complementation and set difference, whereas the class of convex zones is not. We write $\langle s, Z \rangle$, for the class $\{\langle s, v \rangle \mid v \in Z\}$, and say that $\langle s, Z \rangle$ is convex if Z is a convex zone. A partition Π is convex iff all its classes are convex. Finally, we say that Π satisfies the *enabledness* condition iff for each class $\langle s, Z \rangle \in \Pi$ and each arc $e = (s, a, s', \psi, \mathcal{X}) \in E$, it holds : $Z \cap \psi \in \{Z, \emptyset\}$. From now on, we only consider initial partitions respecting convexity and enabledness.

[2] The name comes from *time-abstracting, action-immediate*.

4.1 Refinement

There are two types of preconditions, corresponding to time and action transitions of the timed model. For $e \in E$ we define:

$$pre_e(\langle s, Z\rangle, \langle s', Z'\rangle) \stackrel{\text{def}}{=} \begin{cases} \langle s, Z \cap \psi \cap (Z'[\mathcal{X} := 0])\rangle & \text{if } e = (s, a, s', \psi, \mathcal{X}), \\ \emptyset & \text{otherwise} \end{cases}$$

Proposition 2. *1. $q \in pre_e(B, C)$ iff $q \in B \wedge \exists q' \in C. q \stackrel{s}{\Rightarrow}_{tai} q'$.*
2. If B, C are convex, then $pre_e(B, C)$ is also convex.

The time precondition is nonempty only for pairs of classes having the same control-state component, since the latter does not change with time transitions:

$$pre_\varepsilon(\langle s, Z\rangle, \langle s, Z'\rangle) \stackrel{\text{def}}{=}$$
$$\langle s, \{v \in Z \mid \exists t \in \mathbb{R}_+. (v + t) \in Z' \ \wedge \forall 0 < t' < t. (v + t') \in Z \cup Z'\}\rangle$$

Proposition 3. *1. If $q \in pre_\varepsilon(B, C)$, then $q \in B \wedge (\exists q' \in C,\ q \stackrel{s}{\Rightarrow}_{tai} q')$.*
2. If B, C are convex, then $pre_\varepsilon(B, C)$ is also convex.

Note that the inverse of case 1 above does not hold, contrary to proposition 2. For example, if $B = \langle s, \{x < 1\}\rangle$, $C = \langle s, \{x > 2\}\rangle$, then $\langle s, x = 0\rangle \stackrel{s}{\Rightarrow}_{tai} \langle s, x = 3\rangle$, but $pre_\varepsilon(B, C) = \emptyset$. Indeed, $pre_\varepsilon(B, C)$ is nonempty only if B can lead to C by letting time pass while the system continuously stays in $B \cup C$ during the passage from B to C. Nevertheless, no information is lost regarding time stability in the sense of tai-bisimulation, as the following lemma shows. (See also section 4.2 for more.)

Lemma 4. *Let B, C be two classes of a partition Π such that $q \stackrel{s}{\Rightarrow}_{tai} q'$ for some $q \in B$, $q' \in C$. Then, there exist classes $B = D_0, D_1, ..., D_m = C$ in Π such that $q \in pre_\varepsilon(D_0, pre_\varepsilon(D_1, ...pre_\varepsilon(D_{m-1}, D_m)...))$.*

In the previous example, we have $D_0 = B, D_2 = C$, and $D_1 = \langle s, \{1 \leq x \leq 2\}\rangle$.

The definition of $Succs^l_\Pi(B)$ for $l \in E$ is identical to the one given in section 2.1. Care must be taken in the case $l = \varepsilon$, where we remove the (trivial) time successor of every class, that is, the class itself. The definition of $Ref^l_\Pi(B)$, for $l \in E \cup \{\varepsilon\}$, is identical to the one given in section 2.3.

It remains to prove that coverness and disjointness are preserved during the refinement of B. This is true if the partition is *complete*, i.e., $\forall s \in S$, $I_s = true$. In section 4.3 we discuss the alternatives in the case this condition does not hold.

Proposition 5. *Let Π be a complete partition, and $B \in \Pi$. Also let $Ref^l_\Pi(B)$ be the set $\{B_1, ..., B_m\}$. Then, $\forall i \neq j$. $B_i \cap B_j = \emptyset$, and $\bigcup B_i = B$.*

4.2 The minimal model

In this section we make explicit the relation between $G_{\approx_{tai}}$, the quotient graph w.r.t. $\approx_{tai}$, and G_{min}, the actual model computed by the MMGA adapted as above. Although not identical to G_{min}, $G_{\approx_{tai}}$ can be easily computed from the former by a simple saturation of its ε-transitions.

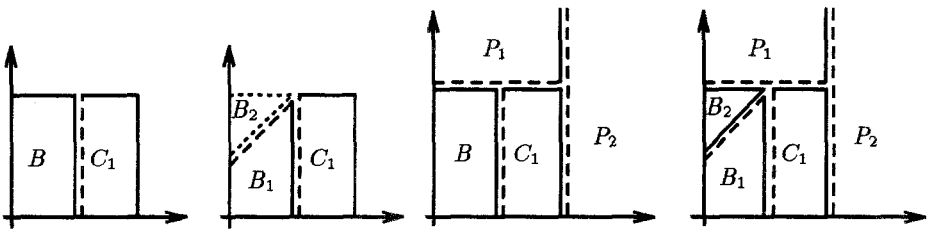
Formally, let $G_{\approx_{tai}} = \langle \Pi_{\approx_{tai}}, \pi_{\approx_{tai}}, \Rightarrow_{tai} \rangle$, and $G_{min} = \langle \Pi, \pi, \Rightarrow \rangle$. Let $\stackrel{\varepsilon}{\Rightarrow}^*$ be the reflexive, transitive closure of $\stackrel{\varepsilon}{\Rightarrow}$.

Proposition 6. *$\Pi = \Pi_{\approx_{tai}}$, $\pi = \pi_{\approx_{tai}}$, and for all $B, C \in \Pi$, (1) $B \stackrel{\varepsilon}{\Rightarrow}_{tai} C$ iff $B \stackrel{\varepsilon}{\Rightarrow} C$, and (2) $B \stackrel{\varepsilon}{\Rightarrow}_{tai} C$ iff $B \stackrel{\varepsilon}{\Rightarrow}^* C$.*

In other words, the partitions of the two graphs are identical, as well as their action transitions, while $\stackrel{\varepsilon}{\Rightarrow}_{tai}$ is the reflexive, transitive closure of $\stackrel{\varepsilon}{\Rightarrow}$.

4.3 Correctness in the presence of strict control-state invariants

If $I_s \subset \mathbb{R}_+$ then the timed model does not contain states $\langle s, v \rangle$ such that $v \in \mathbb{R}_+ \setminus I_s$. In this case coverness is not ensured, as shows the example of figure 1(a), where $B \cup C_1$ is the invariant, $\Pi = \{B, C_1\}$, and $Succs^{\varepsilon}_{\Pi}(B)$ is $\{C_1\}$. Then, $Ref^{\varepsilon}_{\Pi}(B) = \{B_1\}$, which does not cover B. There are several ways to solve this problem:



$Split(B, \{B, C_1\}) = \{B_1\}$
$B_1 = pre_\varepsilon(B, C_1)$
$B_2 = B \setminus B_1$

(a)

$Split(B, \{B, C_1, P_1, P_2\}) = \{B_1, B_2\}$
$B_1 = pre_\varepsilon(B, C_1)$
$B_2 = pre_\varepsilon(B, P_1)$

(b)

Fig. 1. Incomplete refinement (a) ; Adding pseudo-classes to a partial partition (b)

1. A class $\langle s, Z \rangle$ is called a *border* one, if $\exists v \in Z, t \in \mathbb{R}_+$. $(v + t) \in \overline{I_s}$ and $\forall t' \leq t$, $(v + t') \in (Z \cup \overline{I_s})$. In figure 1(a), B is a border class, while B_1 is not. Assume that a border class B is refined w.r.t. $\{C_1, ..., C_m\}$, which yields $\{B_1, ..., B_l\}$ ($l \leq m$, since some B_i may be empty). Let $B' = B \setminus \bigcup B_i$,

which is not convex in general. If $B' \neq \emptyset$, we take an arbitrary (but minimal in number) partition of B' into convex classes $\{B'_1, ..., B'_k\}$, and define $Split(B, \Pi) = \{B_1, ..., B_l, B'_1, ..., B'_k\}$. This solution makes complementation inevitable. What is more, the number of times where complementation will be employed cannot be determined a priori. Indeed, it is always the case that after splitting a border class, at least one of its subclasses is border. The latter may in turn become accessible, be split, and so on.

2. A second solution is to start with a complete initial partition respecting the invariants : $\forall \langle s, Z \rangle \in \Pi_0.\ Z \cap I_s \in \{Z, \emptyset\}$. A class $\langle s, Z \rangle$ is called a *pseudo*-class if $Z \cap I_s = \emptyset$, otherwise it is normal. Pseudo-classes are never split (it suffices to make sure that they are never inserted into the set α of accessible classes). Normal classes can be split w.r.t. pseudo-classes. If all successors of a normal class B are pseudo-classes, then B need not be split. Figure 1(b) shows how the situation of figure 1(a) changes after applying this solution. P_1, P_2 are pseudo-classes, and we now have $Succs^{\varepsilon}_{\Pi}(B) = \{C_1, P_1\}$, and $Ref^{\varepsilon}_{\Pi}(B) = \{B_1, B_2\}$, which covers B. On the other hand, C does not have to be split, since $Succs^{\varepsilon}_{\Pi}(C) = \{P_1, P_2\}$, that is, all its successors are pseudo-classes.

5 Applications

We have implemented the algorithm and applied it to generate the minimal models for a number of case studies. Further, we have used the tool ALDEBARAN to compare the constructed minimal models against labeled transition systems modeling untimed requirements. The main idea consists in considering ε-transitions to be τ-transitions, that is, non-observable or silent ones. Other labels can also be hidden (i.e. replaced by τ) according to the property to be verified. The resulting transition system is then reduced or compared to another model. In particular, we have used the τ^*a-bisimulation equivalence, denoted $\approx_{\tau^*a}$, as well as the τ^*a-simulation preorder [FM91][3].

Due to space limitations, here we illustrate this methodology in detail for only one application, namely the Philips audio control protocol [BPV94]. Experimental results obtained for other well-known examples (e.g. CSMA-CD and FDDI [DOTY95] and Tick-Tock [DOY94] communication protocols) are shown in table 1. The TA column presents the size of the input TA. The M column displays the size of the minimal model, while C_{tot} is the total number of classes created (including classes which were finally found non-accessible). The "splittings" column presents the total number of *Split* operations, the effective time ones (ε subcolumn) and the effective action ones (e subcolumn). N is the number of processes, stations, etc, depending on the protocol. We have used a Sparc 10 with 128 Mbytes of main memory.

[3] Recall that a simulation preorder is a relation satisfying only (1), in the definition of bisimulation given in section 2.

Example	N	TA		M		C_{tot}	splittings			time
		states	arcs	states	trans		total	ε	e	(secs)
CSMA-CD	2	9	21	26	52	62	112	18	15	0.4
	3	26	90	340	1,055	559	1,264	150	173	3.8
	4	72	312	3,828	16,066	4,855	13,592	1,070	1,797	90.9
FDDI	3	19	25	525	933	1,873	3,202	377	637	8.5
	4	25	33	1,606	2,859	7,760	10,980	1,341	2,264	57.4
	5	31	41	4,621	8,801	26,900	32,385	3,878	6,755	315
Tick-Tock	1	24	64	78	121	202	223	31	15	1
	2	72	240	585	976	1,663	1,658	243	163	8.7

Table 1. Minimization results of various examples

5.1 Philips audio-control protocol

The protocol deals with the transmission of a bit stream through a wire, using a Manchester encoding. The receiver can only detect low-to-high voltage changes, which imposes that a bit stream either has an odd length or ends with two 0-bits (all streams start by "1"). Also, the protocol permits a small drift in the clock rates of the sender and the receiver. This is modeled in [DY95] using *multirate* TA, a subclass of hybrid automata which can be transformed into TA [OSY94]. Here, we follow directly the TA model obtained after the transformation, using the automata Sender, Receiver, and Stream(the last one models correct bit streams), which are omitted here (see [DY95] for a full description).

model	TA		M		C_{tot}	splittings			time
	states	arcs	states	trans.		total	ε	e	(secs)
TA_1	146	351	50	61	815	289	114	36	0.9
TA_1 †			283	402	1,557	1,326	445	151	2.3
TA_2	77	207	51	62	674	300	126	39	1
TA_2 †			62	86	672	312	126	39	1.1

Table 2. Philips protocol : minimization results

The main correctness property we want to prove is that the stream received is identical to the one sent. In fact, this can be done only if we make sure that the sender does not start transmitting (action *IN*) a new stream before the last one has been completely received (action *OUT*), that is, no two consecutive *IN* actions take place without on intermediate *OUT*. In order to ensure this property, we have two options :

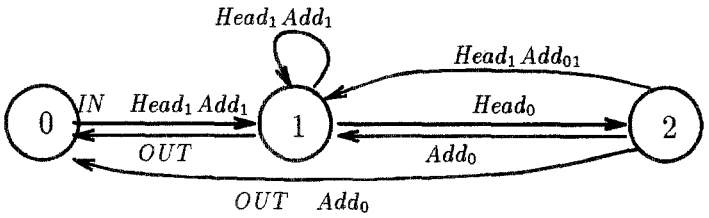
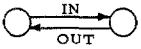


Fig. 2. Good

1. Either to compose the system with the following automaton (called InOut) which prevents the above bad behaviors : 

. Let TA_1 be Sender||Receiver||Stream||InOut.
2. Or to modify Sender by adding a clock which controls the delay between the end of a transmission and the beginning of the next one. This delay should be greater than the time elapsed between the last bit sent by the sender and the action OUT of the receiver. Let TA_2 be Sender'||Receiver||Stream.

For each $TA_i, i = 1, 2$, we obtain two minimal models, one for a correct case (where the maximum drift is $\frac{1}{20}$) and one for an erroneous case (max. drift: $\frac{1}{17}$). Table 2 shows performance results. (The erroneous cases are marked with †.)

Then, we model the correctness requirement by the LTS Good, shown in figure 2 [4]. Let M_i be the minimal model of TA_i for $i = 1, 2$. As expected, in the correct case, we find that $M_i \sqsubseteq_{\tau^* a}$ Good. This does not hold in the erroneous case, and as a diagnostic, we find sequences where the receiver terminates before the sender does.

However, Good $\not\sqsubseteq_{\tau^* a} M_i$, since Good also models bit streams that not satisfy the requirement imposed by Stream. In order to explain this further, consider the LTS depicted in figure 3 obtained by reducing the correct M_1 w.r.t. the $\tau^* a$-bisimulation [5]. This is almost the automaton modeling correct bit streams, except that it contains an additional state 5, grouping all states of the timed model where the sender has sent a "0", but still has bits to transmit. Therefore, the receiver does not have time to perform OUT, since it will first see the next bit transmission taking place. Although trivial, this example shows that often the actual behavior of the system is not exactly the one expected.

[4] $Head_i$ (Add_i) means that bit i is sent (resp. received).
[5] The reduction of the correct M_2 gives exactly the same LTS.

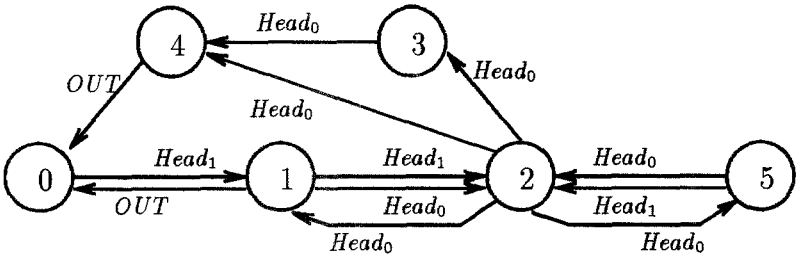


Fig. 3. Minimization with respect to $\approx_{\tau^* a}$

6 Related work

6.1 The ta-bisimulation

In [LY93] another time-abstracting bisimulation has been studied. Given $G = \langle Q, Q^0, \rightarrow \rangle$, we define $G_{ta} = \langle Q, Q^0, \Rightarrow_{ta} \rangle$, as follows:

$$\frac{q \xrightarrow{t} q'' \xrightarrow{\epsilon} q'}{q \overset{\epsilon}{\Rightarrow}_{ta} q'} \qquad \frac{q \xrightarrow{t} q'}{q \overset{\epsilon}{\Rightarrow}_{ta} q'}$$

The ta-bisimulation, denoted $\approx_{ta}$, is the greatest bisimulation defined on G_{ta}, that is, $G \approx_{ta} G'$ iff $G_{ta} \approx G'_{ta}$.

G_{ta} is more abstract than G_{tai}, in the sense that $\Rightarrow_{tai} \subseteq \Rightarrow_{ta}$. Since greater abstractions yield weaker bisimulations [FM91], $\approx_{tai}$ is stronger than $\approx_{ta}$. In fact, we shall prove a stronger property. Let $G_d = \langle \Pi_{\approx_{tai}}, \pi_{\approx_{tai}}, \Rightarrow_d \rangle$, where $B \overset{l}{\Rightarrow}_d C$ iff $\exists D. B \overset{\epsilon^*}{\Rightarrow}_{tai} D \overset{l}{\Rightarrow}_{tai} C\}$, and let $\approx_d$ denote the greatest bisimulation on G_d [6].

Proposition 7. *$G \approx_{ta} G'$ iff $G_{tai} \approx_d G'_{tai}$.*

This result, combined with the one of proposition 6, shows how the ta-minimal model can be computed in two steps: first, one computes the model G_{min} using our adapted algorithm, next, G_{min} is further minimized w.r.t. $\approx_d$.

6.2 Other algorithms

In [ACD+92a] the generic minimization algorithms of [BFH+92] and [LY92], referred to as MAI and MAII respectively, have been adapted for timed systems. Table 3 shows the results obtained with our algorithm and compares its running times ($\star$) to the ones reported in [ACD+92a] for two well known examples, namely the Train-Gate Controller (TGC) and the Fischer's Mutual Exclusion protocol (FMX). The authors of [ACD+92a] used a DEC-5100 with 40 Mbytes

[6] This is essentially the *delay-bisimulation*[FM91].

of main memory. It should be mentioned that our algorithm also required much less memory that the others. $\perp$ denotes nontermination due to memory shortage, and "—" is used for cases that do not appear in [ACD+92a].

Let us note that the idea of avoiding set complementation has been suggested in [YL93]. However, the algorithm presented there is an adaptation of [LY92], whereas our algorithm is based on the [BFH+92] one.

Example	N	TA		M		C_{tot}	splittings			time (secs)		
		states	arcs	states	trans		total	ε	e	$\star$	MAI	MAII
TGC		24	69	25	50	125	113	30	17	0.2	6	12
	†			62	138	159	201	36	27	0.5	57	155
FMX	2	24	34	22	26	34	34	2	0	0	1	2
	2†			47	85	63	118	7	13	0	3	6
	3	119	213	77	108	182	133	15	0	0	8	146
	3†			402	1,117	708	1,379	157	172	1.5	893	$\perp$
	4	548	1,164	252	420	872	493	76	0	2.1	496	$\perp$
	4†			4,437	17,902	7,850	16,144	1,931	2,022	40.4	$\perp$	$\perp$
	5	2,402	5,850	807	1,590	3,887	1,785	325	0	16.3	—	—
	5†									$\perp$	—	—

Table 3. TGC and FMX: minimization results and comparison.

7 Conclusions

We have implemented the algorithm on top of the tool KRONOS [DY95] and have performed experiments with different options. As a result, we have found that among the strategies described in section 4.3 concerning the invariant conditions, the pseudo-classes solution gave in general the worst performances. One the other hand, it turned out that giving priority to splitting w.r.t. timed instead of untimed transitions does not make an important difference. Our implementation includes these options, as well as other ones, that allow, for instance, to specify a set of initial states and/or an initial partition. Experimental results obtained on several case studies are presented in table 1. Based on these results, we claim that using a refinement technique which avoids costly complementations leads to considerable gains in efficiency (both in running times and memory usage) that make minimization possible for larger systems.

We have used the tool ALDEBARAN to further reduce the model generated by our algorithm and compare it to a requirement modeled as an untimed transition system. The requirement does not specify quantitative timing constraints, however its verification strongly depends on the timing conditions embedded in the timed automaton which are indeed preserved by the tai-bisimulation. As we

have found out by the examples, the real behavior of a system is often more complex than expected. Discovering unexpected behaviors helps to gain insight of a system, often revealing intrinsic design problems, and at the same time offering diagnostic traces which are valuable for debugging.

It is worth noting that model checking of TCTL formulas on the minimal model is possible, in the manner of [ACD+92b]. We intend to exploit this possibility as part of our future work. We are also currently studying in more depth the combinations of time-abstracting bisimulations with untimed bisimulation and simulation equivalences and preorders.

References

[ACD+92a] R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *Proc. IEEE RTSS'92*, 1992.

[ACD+92b] R. Alur, C. Courcoubetis, D. Dill, N. Halbwachs, and H. Wong-Toi. Minimization of timed transition systems. In *Proc. CONCUR 1992*. LNCS 630, 1992.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[BFH+92] A. Bouajjani, J.C. Fernandez, N. Halbwachs, P. Raymond, and C. Ratel. Minimal state graph generation. *Science of Computer Programming*, 18:247–269, 1992.

[BPV94] D. Bosscher, I. Polak and F. Vaandrager. Verification of an audio control protocol. In *Proc. FTRTFT'94*, LNCS 863, 1994.

[DOTY95] C. Daws, A. Olivero, S. Tripakis and S. Yovine. The tool KRONOS. Workshop on Hybrid Systems and Autonomous Control, DIMACS, 1995. To appear in LNCS.

[DOY94] C. Daws, A. Olivero and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Proc. FORTE'94*, 1994.

[DY95] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proc. IEEE RTSS'95*, 1995.

[FGM+92] J.Cl. Fernandez, H. Garavel, L. Mounier, A. Rasse, C. Rodriguez, and J. Sifakis. A tool box for the verification of LOTOS programs. In *14th Int. Conf. on Software Engineering*, 1992.

[FM91] J.C. Fernandez and L. Mounier. On the fly verification of behavioural equivalences and preorders. In *Proc. CAV'91*, LNCS 757, 1991.

[LY92] D. Lee and M. Yannakakis. On-line minimization of transition systems. In *Proc. ACM Symposium on Theory of Computing*, 1992.

[LY93] K. G. Larsen and W. Yi. Timed abstracted bisimulation: implicit specification and decidability. In *Proc. MFPS'93*, 1993.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, LNCS 92, 1980.

[OSY94] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In *CAV'94*, LNCS 818, 1994.

[YL93] M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems. In *CAV'93*, LNCS 697, 1993.