

# Two examples of verification of multirate timed automata with Kronos <sup>\*</sup>

C. Daws and S. Yovine <sup>†</sup>

VERIMAG <sup>‡</sup>

Miniparc-Zirst, Rue Lavoisier  
38330 Montbonnot St. Martin, France

## Abstract

*Multirate timed automata [2] are an extension of timed automata [3] where each clock has its own speed varying between a lower and an upper bound that may change from one control location to another. This formalism is well-suited for specifying hybrid systems where the dynamics of the continuous variables are defined or can be approximated by giving the minimal and maximal rate of change.*

*To avoid the difficulties inherent in the verification of multirate timed automata, we follow the approach suggested in [8]. This approach consists of first transforming the multirate timed automata into timed automata and then applying the symbolic techniques implemented in KRONOS. We show the practical interest of this approach analyzing two examples recently proposed in the literature and considered to be realistic case studies: the manufacturing plant of [10] and the Philips audio control protocol [4, 7].*

## 1 Introduction

Multirate timed automata [2] are an extension of timed automata [3] where clocks do not advance at an uniform rate, but each clock has its own speed varying between a lower and an upper bound that may change from one control location to another. This formalism is well-suited for specifying hybrid systems where the dynamics of the continuous variables are defined or can be approximated by giving the minimal and maximal rate of change. Examples of such systems are (1) a robot where the speed of the angular displacement of its mobile arm varies between 15 to 18 degrees per second, and (2) a real-time protocol running on hardware with clocks that measure time with a bounded error.

The verification algorithm proposed in [9] consists in discretizing the state-space with an appropriate granu-

larity of time. The major drawback of discretization is the exponential growth of the size of the discrete state-space for arbitrary large time-bounds. The symbolic method presented in [2] avoids state-space blow-up by representing sets of states as convex polyhedra, though leading in the general case to complex decision procedures. However, efficient symbolic algorithms have been implemented in the tool KRONOS [5] for analyzing timed automata.

The difficulty of the verification of multirate timed automata is due to the timing uncertainty introduced by the drift of the clocks. In order to avoid handling explicitly multiple slopes, it is possible to verify properties on multirate timed automata by transforming them into timed automata. This approach, suggested in [8], is based on a linear transformation of the state-space that preserves many interesting real-time requirements such as invariants and bounded-response properties. The goal of this work is to show that this approach can be successfully applied in practice for analyzing non-trivial systems with KRONOS.

The paper is organized as follows. In section 2 we define the syntax and semantics of multirate timed automata. In section 3 we review the logic TCTL [1, 6] (Timed Computation Tree Logic) that we use to specify the real-time properties. We briefly discuss the symbolic algorithms implemented in KRONOS, as well as the main results of [8]. In the last two sections we show how these results are applied for analyzing two systems, the manufacturing plant of [10], and the physical layer of the Philips audio control protocol reported in [4, 7].

## 2 Multirate timed automata

### 2.1 Definition

A multirate timed automaton  $\mathbf{A}$  is a tuple  $\langle \mathcal{S}, \mathcal{X}, \mathcal{L}, \mathcal{E}, \theta, \rho \rangle$  where:

1.  $\mathcal{S}$  is a finite set of *locations*.
2.  $\mathcal{X}$  is a finite set of *clocks*. A *valuation*  $v \in \mathcal{V}$  is a function that assigns a non-negative real-value  $v(x) \in \mathbb{R}^+$

<sup>\*</sup>In *Proc. IEEE RTSS'95. Pisa, Italy, dec 5-7, 1995.*

<sup>†</sup>E-mail: Conrado.Daws@imag.fr, Sergio.Yovine@imag.fr. Tel: +33 76 90 96 30. Fax: +33 76 41 36 20.

<sup>‡</sup>VERIMAG is a joint laboratory of CNRS, INPG, Université Joseph Fourier, and Verilog SA, associated with IMAG.

to each clock  $x \in \mathcal{X}$ . Let  $\Psi$  be the set of predicates  $\psi$  over  $\mathcal{X}$  of the form  $\bigwedge_{x \in \mathcal{X}} x \in I_x$  where  $I_x$  is an interval with end-points in  $\mathbb{Q}^+ \cup \{+\infty\}$ . We say that  $v$  satisfies  $\psi$  if  $v(x) \in I_x$  for all  $x \in \mathcal{X}$ .

3.  $\mathcal{L}$  is a finite set of *labels*.
4.  $\mathcal{E}$  is a finite set of *edges*. Each edge  $e$  is a tuple  $(s, L, \psi, X, s')$  where  $s \in \mathcal{S}$  is the *source*,  $s' \in \mathcal{S}$  is the *target*,  $L \subseteq \mathcal{L}$  are the *labels*<sup>1</sup>,  $\psi \in \Psi$  is the *guard*, and  $X \subseteq \mathcal{X}$  is the set of clocks to be reset to 0. There is a subset  $\mathcal{E}_u \subseteq \mathcal{E}$  of *urgent* edges that must be executed as soon as possible. We assume that the guard of any urgent edge is always true.
5.  $\theta$  associates each location  $s \in \mathcal{S}$  with an *invariant* condition  $\theta(s) \in \Psi$ .
6.  $\rho$  gives for each location the interval of rate of all clocks. For all locations  $s \in \mathcal{S}$  and clocks  $x \in \mathcal{X}$ ,  $\rho(s, x)$  is a closed interval whose end-points belong to  $\mathbb{Q}^+ - \{0\}$ .

More general extensions of timed automata with intervals of rate over  $\mathbb{Q}$ , that is, allowing clocks with negative slopes and chronometers that can be stopped at any time and resumed later, have also been studied [2].

## 2.2 Semantics

A *state* of  $\mathbf{A}$  is a location and a valuation of clocks satisfying the invariant associated with the location. Let  $\mathcal{Q} \subseteq \mathcal{S} \times \mathcal{V}$  be the set of states of  $\mathbf{A}$ , that is, all pairs  $(s, v)$  such that  $v$  satisfies  $\theta(s)$ . When  $\mathbf{A}$  is in a state  $(s, v) \in \mathcal{Q}$ , it can evolve either by moving through an edge that changes the location and resets some of the clocks (*discrete transition*), or by letting time pass without changing the location (*time transition*).

**Discrete transitions.** Let  $e = (s, L, \psi, X, s')$ . We define  $enable(e)$  to be the set of states  $(s, v)$  such that  $v$  satisfies the guard  $\psi$ . The state  $(s, v)$  has a discrete transition to  $(s', v')$ , denoted  $(s, v) \xrightarrow{L} (s', v')$ , if

1.  $(s, v) \in enable(e)$ ,
2.  $v'(x) = 0$  if  $x \in X$ , otherwise  $v'(x) = v(x)$ .

**Time transitions.** Let  $(s, v) \in \mathcal{Q}$ ,  $t \in \mathbb{R}^+$ , and  $\lambda$  be a function that maps each  $x \in \mathcal{X}$  to a value  $\lambda(x)$  belonging to the interval of rate  $\rho(s, x)$  of  $x$  at  $s$ . We define  $v + \lambda t$  to be the valuation  $v'$  such that  $v'(x)$  is equal to  $v(x) + \lambda(x)t$  for all  $x \in \mathcal{X}$ . The state  $(s, v)$  has a time transition to  $(s, v + \lambda t)$ , denoted  $(s, v) \xrightarrow{\emptyset} (s, v + \lambda t)$ , if

<sup>1</sup>Notice that edges are labeled with sets of actions instead of single actions. This allows modeling simultaneous events as only one edge of the automaton instead of a sequence of edges to be executed in zero time.

1. for all  $t' \leq t$ ,  $v + \lambda t'$  satisfies the invariant  $\theta(s)$ ,
2. for all  $t' < t$  and for all  $e \in \mathcal{E}$ , if  $(s, v + \lambda t') \in enable(e)$  then  $e \notin \mathcal{E}_u$ .

that is, neither the invariant is violated nor an urgent edge becomes enabled during this delay (an urgent edge cannot be postponed).

We associate with  $\mathbf{A}$  a labeled transition system  $\mathbf{T}$  which is a tuple  $\langle \mathcal{Q}, \rightarrow \rangle$  where  $\rightarrow$  is the transition relation defined by the rules above. We will distinguish an initial state  $q_{init} = (s_{init}, v_{init}) \in \mathcal{Q}$  of  $\mathbf{T}$ .

A *run*  $r \in \mathcal{R}$  of  $\mathbf{A}$  is an infinite sequence  $q_0 \xrightarrow{L_0} q_1 \xrightarrow{L_1} \dots$  such that  $\sum_{i \geq 0} t_i$  diverges. A *position*  $\pi$  of  $r$  is a pair  $(i, t)$  with  $0 \leq t \leq t_i$ , and we write  $\sigma_r(\pi)$  for the the state  $q_i + \lambda_i t$ , and  $\delta_r(\pi)$  for the time  $\sum_{j < i} t_j + t$  elapsed since the beginning of the run. An *initialized* run is a run starting at the initial state  $q_{init}$ .

## 2.3 Example

We model here the behavior of a robot, called D-Robot, that executes a task in an automatic manufacturing plant. Figure 1 shows a graphical representation of the multirate timed automaton of the D-Robot. Dashed arrows correspond to urgent edges, and initial locations are indicated with small incoming arrows. To simplify the figure we have omitted all true conditions on edges and invariants as well as rate intervals of the form  $[1, 1]$ .

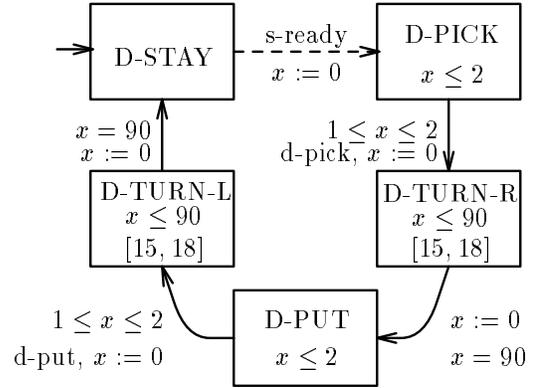


Figure 1: D-Robot.

The D-Robot behaves as follows. Initially, it is waiting for a box near the service station (D-STAY). As soon as a box is ready (s-ready), the robot picks it up (D-PICK), turns right 90 degrees (D-TURN-R), and puts the box on the moving assembly line (D-PUT). Then it turns left (D-TURN-L) to go back to its initial position. The angular horizontal displacement of the arm of the

robot is measured by  $x$  at the locations D-TURN-R and D-TURN-L. Its speed varies between 15 and 18 degrees per second, i.e., its interval of rate is [15, 18]. In locations D-STAY, D-PUT and D-PICK  $x$  measures the time, i.e., its interval of rate is [1, 1]. The vertical movement of the arm to pick up and to put down boxes takes between 1 to 2 seconds ( $1 \leq x \leq 2$ ).

## 2.4 Composition

Usually, the behavior of a system is described by the product of several automata that synchronize through their labels.

Let  $\mathbf{A}_i$  be  $\langle \mathcal{S}_i, \mathcal{X}_i, \mathcal{L}_i, \mathcal{E}_i, \theta_i, \rho_i \rangle$  for  $i = 1, 2$ , with disjoint sets of locations and clocks (i.e.,  $\mathcal{S}_1 \cap \mathcal{S}_2 = \mathcal{X}_1 \cap \mathcal{X}_2 = \emptyset$ ). We call *synchronization* labels all the labels in the set  $\mathcal{L}_1 \cap \mathcal{L}_2$ , that is, all those that are common to both automata. The labels that are not synchronization labels are called *internal*. Two edges  $e_1 \in \mathcal{E}_1$  and  $e_2 \in \mathcal{E}_2$  synchronize if they have the same synchronization labels. Edges whose labels are all internal correspond to asynchronous moves of the components (but can also happen simultaneously). Let  $\text{sync}(e)$  be the set of synchronization labels of an edge  $e$ .

The parallel composition  $\mathbf{A}_1 \parallel \mathbf{A}_2$  is the automaton  $\mathbf{A}$  with  $\langle \mathcal{S}, \mathcal{X}, \mathcal{L}, \mathcal{E}, \theta, \rho \rangle$  such that

1. The set  $\mathcal{S}$  of locations is  $\mathcal{S}_1 \times \mathcal{S}_2$ .
2. The set  $\mathcal{X}$  of clocks is  $X_1 \cup X_2$ .
3. The set  $\mathcal{L}$  of labels is  $\mathcal{L}_1 \cup \mathcal{L}_2$ .
4. The set  $\mathcal{E}$  of edges is obtained as follows. For all  $e_i \in \mathcal{E}_i$  of the form  $(s_i, L_i, \psi_i, X_i, s'_i)$ ,
  - (a)  $e = ((s_1, s_2), L_1 \cup L_2, \psi_1 \wedge \psi_2, X_1 \cup X_2, (s'_1, s'_2))$ ,  
 $e \in \mathcal{E}$  iff  $\text{sync}(e_1) = \text{sync}(e_2)$ , and  
 $e \in \mathcal{E}_u \subseteq \mathcal{E}$  if  $e_1 \in \mathcal{E}_{u_1}$  or  $e_2 \in \mathcal{E}_{u_2}$ ,
  - (b)  $e = ((s_1, s_2), L_1, \psi_1, X_1, (s'_1, s_2))$ ,  
 $e \in \mathcal{E}$  iff  $\text{sync}(e_1) = \emptyset$ , and  $e \in \mathcal{E}_u \subseteq \mathcal{E}$  if  $e_1 \in \mathcal{E}_{u_1}$ ,
  - (c)  $e = ((s_1, s_2), L_2, \psi_2, X_2, (s_1, s'_2))$ ,  
 $e \in \mathcal{E}$  iff  $\text{sync}(e_2) = \emptyset$ , and  $e \in \mathcal{E}_u \subseteq \mathcal{E}$  if  $e_2 \in \mathcal{E}_{u_2}$ .
5. The invariant  $\theta(s_1, s_2)$  is  $\theta_1(s_1) \wedge \theta_2(s_2)$ .
6. The drift intervals of the clocks are such that  $\rho(x)$  is  $\rho_1(x)$  if  $x \in \mathcal{X}_1$  and  $\rho_2(x)$  if  $x \in \mathcal{X}_2$ .

When initial locations  $s_{init}^1$  and  $s_{init}^2$  are given we will consider only the set of locations of the product reachable from the location  $(s_{init}^1, s_{init}^2)$ .

## 3 Verification of real-time properties

We specify real-time properties in the logic TCTL [1, 6]. Model-checking of TCTL on multirate timed automata can be done with the tool KRONOS [5] by applying the transformation provided in [8]. We review here TCTL and summarize the main results of [8].

### 3.1 TCTL

For clarity we consider here a simple version of TCTL with subscripted unary temporal operators. The syntax and semantics of full TCTL is given in [1, 6].

**Syntax.** Let  $\mathcal{P}$  be a set of atomic propositions and  $\mathcal{X}$  be a set of clocks. The formulas of this fragment of TCTL are defined by the following grammar:

$$\varphi ::= p \mid x \in I \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists\Diamond_I\varphi \mid \forall\Diamond_I\varphi$$

where  $p \in \mathcal{P}$ ,  $x \in \mathcal{X}$  and  $I$  is an interval with end-points in  $\mathbb{Q}^+ \cup \{+\infty\}$ .

Intuitively,  $\exists\Diamond_I\varphi$  means that there exists a run such that  $\varphi$  holds at some state at a time  $t \in I$ , and  $\forall\Diamond_I\varphi$  means that every run has a state satisfying  $\varphi$  at a time  $t \in I$ . We write  $\exists\Diamond\varphi$  for  $\exists\Diamond_{[0,+\infty)}\varphi$  meaning that a state satisfying  $\varphi$  is reachable, and  $\forall\Box\varphi$  for  $\neg\exists\Diamond\neg\varphi$  meaning that  $\varphi$  is an invariant, that is, it holds for all the states.

**Semantics.** The formulas of TCTL are interpreted over the set of states of a multirate timed automaton  $\mathbf{A}$ . Let  $\mu$  be a function that associates with each location  $s \in \mathcal{S}$  a set  $\mu(s) \subseteq \mathcal{P}$  of propositions. Let  $q$  be a state  $(s, v) \in \mathcal{Q}$ . We write  $q(x)$  for  $v(x)$ ,  $\mu(q)$  for  $\mu(s)$ , and  $\mathcal{R}(q)$  for the set of runs starting at  $q$ .

Formally, a state  $q \in \mathcal{Q}$  satisfies the formula  $\varphi$ , denoted  $q \models \varphi$ , if

- $q \models p$  iff  $p \in \mu(q)$ ;
- $q \models x \in I$  iff  $q(x) \in I$ ;
- $q \models \neg\varphi$  iff  $q \not\models \varphi$ ;
- $q \models \varphi_1 \vee \varphi_2$  iff  $q \models \varphi_1$  or  $q \models \varphi_2$ ;
- $q \models \exists\Diamond_I\varphi$  iff for some run  $r \in \mathcal{R}(q)$  there exists  $\pi$  such that  $\sigma_r(\pi) \models \varphi$  and  $\delta_r(\pi) \in I$ ;
- $q \models \forall\Diamond_I\varphi$  iff for all runs  $r \in \mathcal{R}(q)$  there exists  $\pi$  such that  $\sigma_r(\pi) \models \varphi$  and  $\delta_r(\pi) \in I$ .

The set of states that satisfy the formula  $\varphi$  is called the characteristic set of  $\varphi$ . We extend the satisfaction relation to set of states. Let  $Q \subseteq \mathcal{Q}$  be a set of states. We say that  $Q$  satisfies  $\varphi$ , and we write  $Q \models \varphi$ , if for all states  $q \in Q$ ,  $q \models \varphi$ . We say that  $\mathbf{A}$  satisfies  $\varphi$ , and write  $\mathbf{A} \models \varphi$ , if  $Q \models \varphi$ .

### 3.2 The tool KRONOS

Given an automaton  $\mathbf{A}$  and a formula  $\varphi$  of TCTL, the model-checking problem consists in computing the characteristic set of  $\varphi$ . The tool KRONOS [5] solves this problem for timed automata. KRONOS implements a symbolic verification method based on predicate transformers for computing the set of successors (forward analysis) and predecessors (backward analysis) of sets of states. The characteristic set of a formula is symbolically represented as a disjunction of convex polyhedra and computed as a fixed-point defined in terms of the predicate transformers. We briefly review here the basis of the algorithms implemented in KRONOS. A more detailed description of the symbolic verification method can be found in [6, 2].

**Backward analysis.** Given a set of states  $Q \subseteq \mathcal{Q}$ , the predecessors of  $Q$  are those states that can reach a state in  $Q$  by letting time elapse for some amount and then executing a discrete transition. We define:

$$\text{pred}(Q) = \{q' \mid \exists q \in Q, t \in \mathbb{R}^+, L \subseteq \mathcal{L}. q' \xrightarrow{t}_i \xrightarrow{L}_0 q\}.$$

Now, if  $Q$  is the characteristic set of  $\varphi$ , the set of states  $Q'$  that satisfy  $\exists \diamond \varphi$  is the least fixed-point of the equation  $Z = Q \cup \text{pred}(Z)$ . We have that  $Q$  is reachable by an initialized run  $r \in \mathcal{R}(q_{\text{init}})$  iff  $q_{\text{init}}$  belongs to  $Q'$ . All the formulas of TCTL can be expressed by fixed-point equations on the predicate transformer  $\text{pred}$  [6].

**Forward analysis.** Given a set of states  $Q \subseteq \mathcal{Q}$ , the successors of  $Q$  are those states that can be reached from a state in  $Q$  by a time delay followed by a transition through an edge of the automaton. We define:

$$\text{post}(Q) = \{q' \mid \exists q \in Q, t \in \mathbb{R}^+, L \subseteq \mathcal{L}. q \xrightarrow{t}_i \xrightarrow{L}_0 q'\}.$$

Now, the set  $\text{reach}(\mathbf{A})$  of states reachable by an initialized run  $r \in \mathcal{R}(q_{\text{init}})$  is the least fixed-point of the equation  $Z = q_{\text{init}} \cup \text{post}(Z)$ . If  $Q$  is the characteristic set of  $\varphi$ , then  $q_{\text{init}}$  satisfies  $\exists \diamond \varphi$  iff  $\text{reach}(\mathbf{A}) \cap Q \neq \emptyset$ . This method can also be used to prove invariants:  $\forall \square \varphi$  holds for all reachable states iff  $\text{reach}(\mathbf{A}) \subseteq Q$ .

### 3.3 Transforming multirate timed automata

Model-checking on a multirate timed automaton  $\mathbf{A}$  can be done with KRONOS by transforming it into a timed automaton  $\mathbf{A}'$  by applying the method proposed in [8].  $\mathbf{A}$  and  $\mathbf{A}'$  have the same graph structure (locations and edges), and every clock  $x \in \mathcal{X}$  is replaced by a perfect clock  $x' \in \mathcal{X}'$  with a constant slope of 1. The invariants and the conditions of the edges over  $\mathcal{X}$  are transformed on predicates over  $\mathcal{X}'$  by a linear transformation defined for each location.

**Transformation.** Let us first explain the transformation with an example. Consider the multirate timed automaton of figure 1. The interval of rate of  $x$  at location D-TURN-R is  $[15, 18]$ . Let  $x'$  be a clock that advances with rate equal to one, that records the time elapsed since the arm started its horizontal displacement. We have that the values of  $x$  and  $x'$  are such that  $\frac{1}{18}x \leq x' \leq \frac{1}{15}x$ . Now, the arm of the robot will take a time between 5 ( $90/18$ ) and 6 ( $90/15$ ) seconds to turn 90 degrees to the right. Then, the condition  $x = 90$  on the angular displacement becomes the condition  $5 \leq x' \leq 6$  on the time elapsed when turning. Applying a similar transformation to all the locations we get the timed automaton of figure 2.

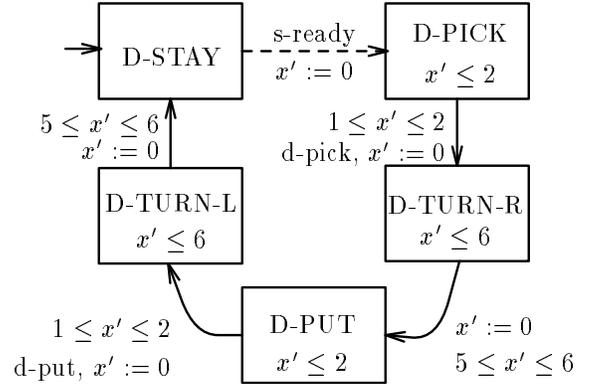


Figure 2: Transformed automata of D-Robot.

Let  $\mathbf{A}$  be a multirate timed automaton. For all locations  $s \in \mathcal{S}$  and clocks  $x \in \mathcal{X}$ , we define  $\kappa_s(x, x')$  to be the relation  $\frac{1}{\beta}x \leq x' \leq \frac{1}{\alpha}x$  where  $[\alpha, \beta]$  is the interval of rate of  $x$  in  $s$ . Then, we transform  $\mathbf{A}$  into  $\kappa(\mathbf{A})$  which is the timed automaton  $\langle \mathcal{S}, \mathcal{X}', \mathcal{L}, \mathcal{E}', \theta' \rangle$ , where every edge  $\kappa(e) \in \mathcal{E}'$  is obtained from the corresponding  $e \in \mathcal{E}$  by applying the transformation  $\kappa_s$  to the guard, where  $s$  is the source location of  $e$ . That is, for every condition  $x \in I$  where  $I$  is an interval with end-points  $a \leq b$ ,  $\kappa_s(x \in I)$  is the condition  $x' \in I'$  where  $I'$  is the interval whose end-points are  $\frac{1}{\beta}a$  and  $\frac{1}{\alpha}b$ . The invariant  $\theta'(s)$  is obtained accordingly.

**Properties.** It is shown in [8] that  $\kappa(\mathbf{A})$  is *more abstract than*  $\mathbf{A}$ , in the sense that the transition system of  $\kappa(\mathbf{A})$  has at least all the transitions of the one of  $\mathbf{A}$ . More precisely,  $\kappa$  induces a *simulation* relation between the corresponding transition systems.

**Proposition 3.1** [8] *Let  $\langle \mathcal{Q}, \rightarrow \rangle$  and  $\langle \mathcal{Q}', \mapsto \rangle$  be the transition systems of  $\mathbf{A}$  and  $\kappa(\mathbf{A})$ , respectively. Then, for all pairs of states  $(q_1, q'_1)$  such that  $\kappa(q_1, q'_1)$ , if*

$q_1 \xrightarrow{I} q_2$  then there exists  $q'_2$  such that  $q'_1 \xrightarrow{I} q'_2$  and  $\kappa(q_2, q'_2)$ .

We have that for all  $q \in \text{reach}(\mathbf{A})$ , there exists  $q'$  such that  $\kappa(q, q')$  and  $q' \in \text{reach}(\kappa(\mathbf{A}))$ . Proposition 3.1 implies that  $\kappa$  preserves the following fragment of TCTL:

$$\varphi' ::= p \mid \neg p \mid x' \in I \mid \varphi' \vee \varphi' \mid \varphi' \wedge \varphi' \mid \forall \square_I \varphi' \mid \forall \diamond_I \varphi'$$

Preservation means that if a state  $q' \in Q'$  satisfies a formula  $\varphi'$  in this fragment, then all the states  $q \in Q$  such that  $\kappa(q, q')$  satisfy the formula  $\kappa(\varphi')$  obtained from  $\varphi'$  by transforming the conditions on  $x'$  into conditions on  $x$ .

**Proposition 3.2** [8] *If  $\kappa(\mathbf{A}) \models \varphi'$  then  $\mathbf{A} \models \kappa(\varphi')$ . If  $\text{reach}(\kappa(\mathbf{A})) \models \varphi'$  then  $\text{reach}(\mathbf{A}) \models \kappa(\varphi')$ .*

So, in order to prove that  $\mathbf{A}$  satisfies a formula  $\varphi$  we need to find a formula  $\varphi'$  such that  $\kappa(\varphi')$  implies  $\varphi$  and check that  $\kappa(\mathbf{A})$  satisfies  $\varphi'$ . When no constraints on the clocks of  $\mathbf{A}$  appear in  $\varphi$ , we take  $\varphi'$  to be  $\varphi$ . When  $\kappa$  is a bijection we take  $\varphi'$  to be  $\varphi$  where the conditions on  $x$  are transformed into conditions on  $x'$ , and clearly we have that then  $\kappa(\varphi')$  is equal to  $\varphi$ .

In the next two sections we show how these results can be applied in practice for verifying the correctness of two systems described by multirate timed automata.

## 4 A manufacturing plant

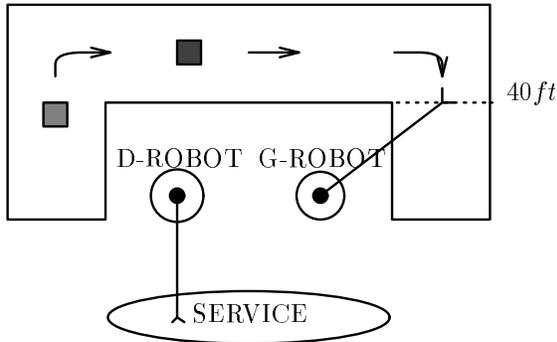


Figure 3: Configuration of the manufacturing system.

We model here the behavior of the manufacturing plant [10] sketched in figure 3. This system is composed of a 50-foot-long belt moving from left to right, two robots located at both extremes of the belt, a service station and two boxes. One of these robots, the D-Robot, has already been modeled in section 2.3 (figures 1 and 2).

## 4.1 Model

The G-Robot (figure 4) picks up a box on the right side of the belt and deposits it on the station where boxes get serviced.

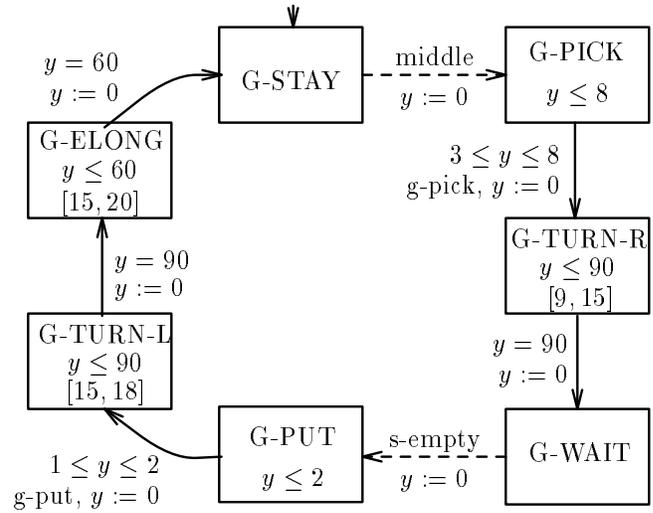


Figure 4: G-Robot.

The robot waits at the 40 feet mark of the assembly line for a box to come. As soon as a box passes across the mark (modeled by the label *middle*), the robot follows it and picks it up by retracting its arm inwards and displacing it to the right until it forms an angle of 90 degrees with the station. This movement takes between 3 to 8 seconds. Then it turns right at an angular speed between 9 to 15 degrees per second and stops near the station. It deposits the box as soon as the station is empty. Then it goes back to its initial position by turning left the first 90 degrees at a speed in the interval [15, 18] and the next 60 degrees (to reach the 40 feet mark) at a speed between 15 to 20 degrees per second (the arm is also elongated in this part).

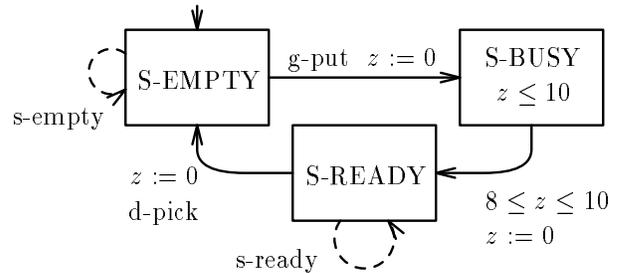


Figure 5: Service station.

The service station (figure 5) is initially empty. A

box gets serviced on the station between 8 to 10 seconds. Then the box is ready to be picked up by the D-Robot.

The movement of the box is tracked by breaking the assembly line in two segments: before and after the 40 feet mark. Initially, each box is on the conveyor belt somewhere in the first segment. A box takes between 133 and 134 seconds to reach the mark. When it enters into the last segment of the belt it must be picked up by the G-Robot, otherwise it falls down. This behavior is modeled by the automaton of figure 6.

Notice that it is not necessary to transform neither the service station nor the boxes since they are already modeled by timed automata. In other words, the transformation for these systems is just the identity.

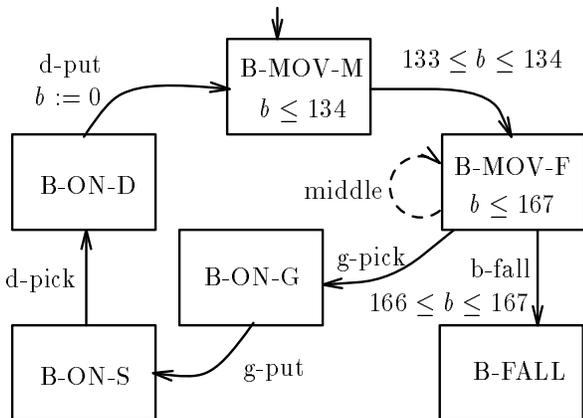


Figure 6: Box.

## 4.2 Verification

The goal is to find all the safe initial positions of the boxes on the assembly line, that is, all those from where a fall is not reachable. Let B-FALL stand for B-FALL-1  $\vee$  B-FALL-2. The formula  $\varphi$  characterizing the set of safe initial states is

$$\text{S-EMPTY} \wedge \text{D-STAY} \wedge \text{G-STAY} \wedge \\ \text{B-MOV-M-1} \wedge \text{B-MOV-M-2} \wedge \forall \square \neg \text{B-FALL}$$

Since no constraints on clocks appear in  $\varphi$ , we use  $\varphi$  as the formula to be evaluated on the transformed system. KRONOS evaluates the formula in 26 seconds (on a Sun Sparc Station 10 with 64MB of memory) applying the backward-computation algorithm, and returns as result the predicate

$$b_1 > b_2 + 9 \vee b_2 > b_1 + 9$$

stating that the boxes can be placed anywhere on the belt but their relative distance must be greater than 9

seconds. This constraint is precisely the characteristic set of  $\varphi$  on the original system because the transformation relation associated with all the initial locations is the identity.

## 5 Philips audio control protocol

We analyze here the behavior of the physical layer of the Philips audio control protocol. Our model is a direct translation of the formal specification given in [4] into multirate timed automata. We automatically verify with KRONOS the properties stated and proved by hand in [4].

### 5.1 Informal description

The purpose of the physical layer is to transport a raw bit stream through a wire. The Philips protocol uses Manchester encoding of bit streams where a bit slot is of length  $4Q$ , where  $Q$  is a given parameter. A 1 bit is sent by raising the voltage in the middle of the bit slot, that is, the voltage is low during the first half of the slot and high in the second one. Sending a 0 bit is just the reverse. When the same bit is sent in two consecutive bit slots, an additional edge is placed in between. Figure 7 shows an example of this encoding.

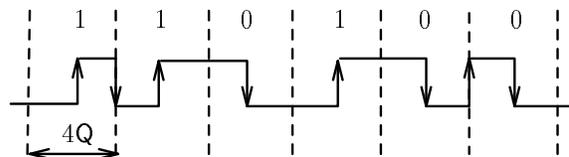


Figure 7: Manchester encoding of 110100.

This encoding ensures that every bit slot has a voltage change in the middle, so it should be easy for the receiver to synchronize with the sender. However, since the particular equipment used is such that voltage transitions on the wire from high to low cannot be reliably detected, the receiver must decode the binary signal only from the upgoing voltage-edges, by counting the time elapsed between them. To do so, the protocol requires all bit streams to have odd length or end with two 0-bits.

Besides, the hardware clocks drift with some known tolerance  $T$ , so the protocol has to take care of the timing uncertainty of the events that makes more difficult the task of the receiver to decode the bit stream. The tolerance allowed by Philips is of at most  $\pm 5\%$ .

### 5.2 Model

The protocol is defined as the composition of the corresponding automata modeling the sender, the receiver

and all the allowed messages.

**The sender.** Figure 8 shows the automaton SENDER of the sender. The input action  $IN$  corresponds to a request to transmit a message. The action  $UP$  corresponds to an upgoing edge on the bus. Downgoing edges are modeled by the action  $DOWN$ . The sender also uses the following actions:  $Is$  says whether the string is empty ( $Is_\epsilon$ ) or its first bit is either “0” ( $Is_0$ ) or “1” ( $Is_1$ ),  $Head_0$  and  $Head_1$  indicate that the corresponding bit at the head of the message has been sent and removed from the list, and  $Last_0$  indicates that the remaining list is equal to  $\langle 0 \rangle$ . The atomic proposition **wire\_high** (**wire\_low**) holds whenever the voltage on the wire is high (low), **S\_idle** (**S\_busy**) holds whenever the sender is idle (busy), **first** (**second**) corresponds to the first (second) half of the bit slot. The sender uses a drifting clock  $x$  with tolerance  $T$  that is reset in the middle of each bit slot. If we take  $T$  to be  $\frac{1}{20}$  (corresponding to a tolerance of  $\pm 5\%$ ) we have that the interval of rate of  $x$  is  $[\frac{19}{20}, \frac{21}{20}]$  for all locations (omitted in the figure).

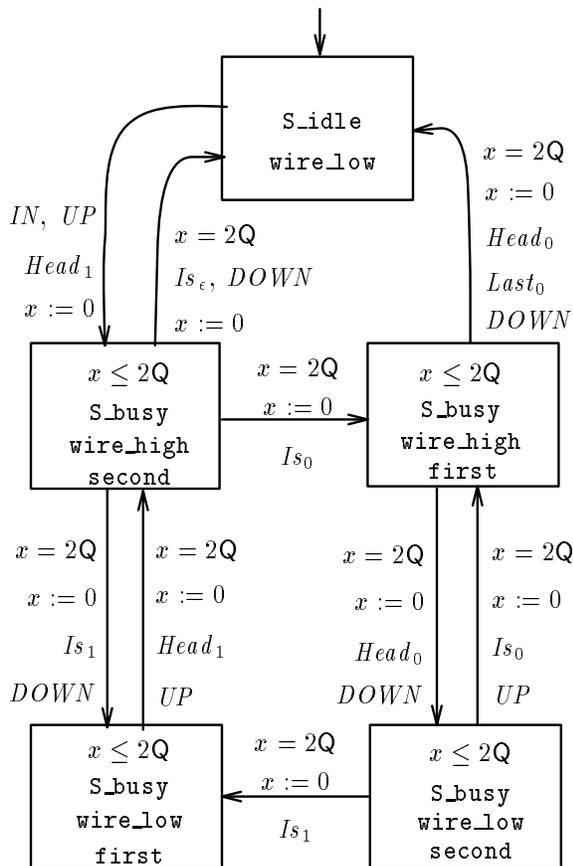


Figure 8: SENDER ( $\rho(x) = [\frac{19}{20}, \frac{21}{20}]$ ).

The sequence of  $UP$ 's and  $DOWN$ 's is generated as required by the Manchester encoding. The first  $UP$  occurs concurrently with the action  $IN$ . Subsequent  $UP$ 's occur only if the voltage on the wire is low and the remaining message is not empty.  $UP$ 's occur when  $x = 2Q$ , and in the first half of the slot if the head is “1”, otherwise in the second half. In the first case, the “1” is removed from the string and  $x$  is set to zero.  $DOWN$ 's occur when the wire is high and  $x = 2Q$ , and in the first half of the slot if the head is “0”, otherwise in the second half. The head is removed if it is “0”. The sender returns to the initial state if the list is empty or  $\langle 0 \rangle$ .

**The receiver.** Figure 9 shows the automaton RECEIVER of the receiver. It synchronizes with the sender through the action  $UP$ . The proposition **R\_idle** holds at the initial location where the receiver is idle, waiting for the first  $UP$  from the sender. It is busy (**R\_busy**) elsewhere. The receiver keeps track of the last bit received (**R\_last\_0**, **R\_last\_1**) and of the parity (**Even**, **Odd**) of the received sequence of bits. The receiver uses a drifting clock  $y$  with tolerance  $T$  that is reset whenever an upgoing edge is detected, that is,  $y$  records the time elapsed since the last action  $UP$  with an error of  $\pm T$ . For  $T$  equal to  $\frac{1}{20}$  we have that the interval of rate of  $y$  is  $[\frac{19}{20}, \frac{21}{20}]$  for all locations (omitted in the figure).

Whenever an  $UP$  comes, it adds the sequence  $\langle 0 \rangle$  ( $Add_0$ ),  $\langle 1 \rangle$  ( $Add_1$ ) or  $\langle 01 \rangle$  ( $Add_{01}$ ) to the list of already received bits, according to the last bit previously added, and to the time elapsed since the last  $UP$ . When the sequence is interpreted as being complete, it outputs an  $OUT$ . This occurs when  $y = 7Q$  if the last bit received is “0”, otherwise when  $y = 9Q$ . If the last bit received is “1” and the length of the sequence is **Odd**(i.e., propositions **R\_last\_1** and **Odd** hold), the receiver adds a “0” at the end of the message.

**Allowed messages.** The protocol requires that all bit streams have odd length or end with two 0-bits. Clearly, this requirement can be modeled by an automaton MESSAGE omitted here. We assume that the locations of MESSAGE have labels **S\_empty**, **S\_head\_0** and **S\_head\_1**, indicating that the message is either empty or the head is “0” or “1”.

### 5.3 Verification

Let PROTOCOL be the composition of SENDER, RECEIVER and MESSAGE. The protocol starts at location **S\_idle**  $\wedge$  **R\_idle**  $\wedge$  **S\_empty** with both clocks set to 0. The main correctness property to be proved is that the received string of bits is equal to the message sent by the sender. To do this we must take care of new incoming messages that arrive before the last one has been output

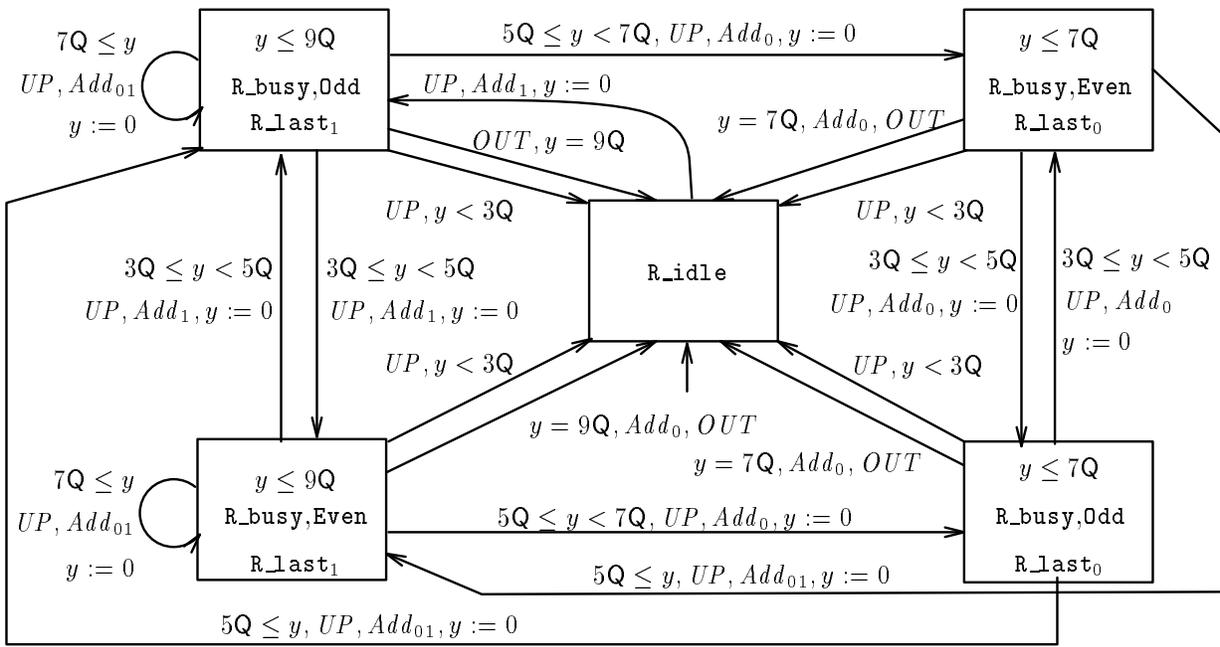


Figure 9: RECEPTOR ( $\rho(y) = [\frac{19}{20}, \frac{21}{20}]$ ).

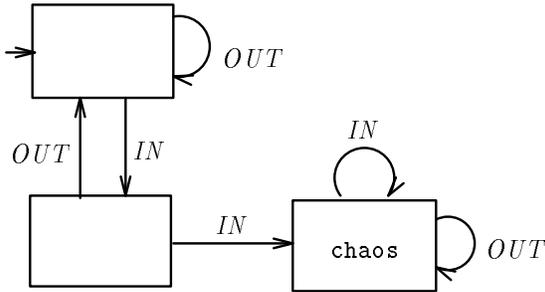


Figure 10: CHAOS.

by the receiver. This is considered as a chaotic situation in [4], that is, whenever an action *IN* happens before the action *OUT* corresponding to the previous message, the protocol moves to a state of chaos where everything is allowed. This behavior is modeled by the automaton CHAOS in figure 10. Let SYSTEM be the composition of PROTOCOL and CHAOS.

**Correctness.** We want first to check that the invariants stated in [4] hold for all reachable states of SYSTEM. Recall that  $\varphi$  is an invariant of the reachable states iff  $q_{init} \models \forall \square \varphi$ , or equivalently,  $reach(\text{SYSTEM}) \models \varphi$ . The most interesting invariants are those that relate the states of the sender and the receiver, as well as the distance between their clocks. Table 1 shows some of those invariants. The intended meaning of  $\approx$  is “almost equal”

by abstracting away from the amount of drift. For a tolerance  $T$  of  $\frac{1}{20}$  we have

$$y \approx x + c \quad \text{iff} \quad \frac{19}{21}y \leq x + c \leq \frac{21}{19}y.$$

All these invariants give insight into the behavior of the protocol, but we still have to prove that the receiver outputs correctly the stream of bits produced by the sender.

To prove this correctness property we need to relate the string of bits already sent with the one already received. For the Philips protocol this comes down to proving that at any moment there is at most a pending “0” in transit that is about to be received. This situation is modeled by the automaton INCORRECT in figure 11 where the state *incorrect* is entered whenever an erroneous reception is detected. Let REQUIREMENT be the composition of SYSTEM and INCORRECT. The properties of table 2 hold for all reachable states of REQUIREMENT.

To establish the correctness of the protocol we must check that state *incorrect* can only be reached in the chaotic situation. The following property holds for all reachable states of REQUIREMENT:

$$\text{incorrect} \Rightarrow \text{chaos} \quad (9)$$

For any formula  $\varphi$  of tables 1 and 2, we have to find a formula  $\varphi'$  such that  $\varphi$  holds in SYSTEM if  $\varphi'$  holds in  $\kappa(\text{SYSTEM})$ . We define the relation  $\approx'$  on the transformed system as follows:

$$y' \approx' x' + c \quad \text{iff} \quad x' + \frac{c}{21} \leq y' \leq x' + \frac{c}{19}$$

---

$S\_busy \wedge wire\_low \wedge \neg chaos$	$\Rightarrow$	$y \approx x + 4Q \vee y \approx x + 2Q \vee y \approx x \wedge S\_head_1$	(1)
$S\_busy \wedge wire\_high \wedge \neg chaos$	$\Rightarrow$	$y \approx x \vee y \approx x - 2Q \wedge S\_busy \wedge S\_head_0$	(2)
$R\_idle \wedge \neg chaos$	$\Rightarrow$	$S\_idle \wedge wire\_low$	(3)
$S\_busy \wedge enable(OUT)$	$\Rightarrow$	$chaos$	(4)

---

Table 1: Invariants of SYSTEM.

---

$R\_busy \wedge \neg chaos \wedge R\_last_1 \wedge (y \approx x \vee y \approx x + 2Q)$	$\Rightarrow$	RS	(5)
$R\_busy \wedge \neg chaos \wedge R\_last_0 \wedge y \approx x$	$\Rightarrow$	RS	(6)
$R\_busy \wedge \neg chaos \wedge R\_last_1 \wedge (y \approx x + 4Q \vee y \approx x + 6Q)$	$\Rightarrow$	RoS	(7)
$R\_busy \wedge \neg chaos \wedge R\_last_0 \wedge (y \approx x + 2Q \vee y \approx x + 4Q)$	$\Rightarrow$	RoS	(8)

---

Table 2: Invariants of REQUIREMENT.

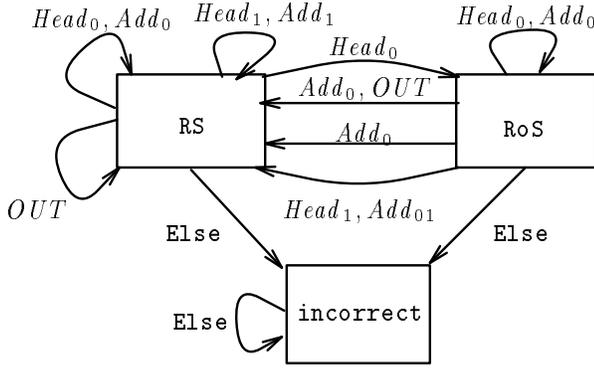


Figure 11: INCORRECT.

We can show that  $\approx'$  and  $\approx$  are related as follows:

*For all  $x, y, x', y'$  such that  $\kappa(x, y, x', y')$ ,  $y' \approx' x' + c$  implies  $y \approx x + c$ .*

Now, let  $\varphi'$  be any of the invariants above where constraints of the form  $y \approx x + c$  are replaced by  $y' \approx' x' + c$ . As a consequence of proposition 3.2 we have that

$reach(\kappa(\text{SYSTEM})) \models \varphi' \text{ implies } reach(\text{SYSTEM}) \models \varphi$ .

**Bounded transmission time.** We want to prove that the action  $OUT$  by the receiver follows the corresponding action  $IN$  from the sender within some bounded time. Clearly this bound depends on the length  $m$  of the bit stream. An upper bound of  $\frac{20}{19}(4m + 5)Q$  is found in [4].

We can check this bound by verifying that the time elapsed from any bit sent to the following ( $m - 1$  times) never exceeds  $\frac{20}{19}4Q$ , and that the delay between the last bit and the output by the receiver is less than  $\frac{20}{19}9Q$ . To do this we add a new clock  $z$  with rate equal to one that is reset by the transitions of the automaton INCORRECT that correspond to a bit ( $Head_0, Head_1$ ) produced by the sender or to an output  $OUT$  by the receiver. Table 3 shows the corresponding properties that are proved to hold for all reachable states of REQUIREMENT.

Table 4 shows the running times obtained with KRONOS to check all the invariants on the transformed system using backward and forward symbolic analysis. The second row corresponds to the results obtained when the invariant is checked “on-the-fly” while the reachable state-space is constructed. The third row corresponds to an alternative strategy consisting of first constructing the set of reachable states once and forever and then verifying that it satisfies all the formulas. KRONOS computes the set of reachable states in 1.5 seconds.

**Allowed tolerance.** We have also checked that all the invariants hold for tolerances of  $\frac{1}{18}$  and  $\frac{1}{19}$ . As it is shown in [4] we have found that the protocol does not work correctly when the tolerance is equal to  $\frac{1}{17}$ . This is done by providing a counter-example for invariant (4). Using the forward-analysis algorithm, KRONOS finds a run that invalidates this invariant for the transformed system. This run corresponds to a run of the original system such that the sender’s clock grows maximally slow, the receiver’s one grows maximally fast, and the message sent is “101”.

$$s\_busy \wedge (enable(Head_0) \vee enable(Head_1)) \wedge \neg chaos \Rightarrow \frac{20}{21}4Q \leq z \leq \frac{20}{19}4Q \quad (10)$$

$$enable(OUT) \wedge \neg chaos \Rightarrow z \leq \frac{20}{19}9Q \quad (11)$$

Table 3: Bounded transmission time properties of REQUIREMENT.

formula	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
backward	7.33	2.90	2.20	2.77	4.38	3.05	3.08	2.95	1.47	12.27	2.65
forward	1.43	1.33	0.27	0.35	0.83	0.43	0.53	0.55	0.17	1.82	1.27
forward(*)	0.38	0.38	0.12	0.20	0.23	0.17	0.18	0.20	0.10	0.53	0.30

Table 4: Running times in seconds obtained on a Sun Sparc Station 10.

## 6 Conclusion

This paper focused on the analysis of multirate timed automata by first transforming them into timed automata and then applying the symbolic techniques implemented in KRONOS. In order to show the practical application of this approach we have considered two examples recently proposed in the literature and considered to be realistic case studies.

For the manufacturing plant we have been able to find the set of all safe initial positions of the boxes, while this problem could not be solved using discretization techniques [10].

For the Philips audio control protocol we have automatically checked all the invariants stated and hand-proved in [4] for a tolerance  $T$  less than  $\frac{1}{17}$ . We have also found a trace of the system that violates one of the invariants when  $T$  is equal to  $\frac{1}{17}$ . Another automated analysis of this protocol is reported in [7] using the tool HyTech that deals directly with the multirate timed automaton modeling the system. However, in practice, the better performances obtained with KRONOS, reducing the running times from hours to seconds, show the advantage of our approach.

Besides, the symbolic forward-analysis algorithm recently implemented in KRONOS appears to be well-adapted for checking invariants as shown by the results given in table 4. Also, this new feature allows diagnosis of errors by providing a trace when the system does not satisfy a required invariant.

## References

[1] R. Alur, C. Courcoubetis, and D.L. Dill. Model

checking in dense real time. *Information and Computation*, 104(1):2–34, 1993.

- [2] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *TCS*, 138:3–34, 1995.
- [3] R. Alur and D.L. Dill. A theory of timed automata. *TCS*, 126:183–235, 1994.
- [4] D. Bosscher, I. Polak, and F. Vaandrager. Verification of an audio control protocol. *FTRFT'94*, pages 170–192. LNCS 863, 1994.
- [5] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. *Proc. 7th. FORTE'94*, pages 227–242. Formal Description Techniques VII, Chapman & Hall, 1995.
- [6] T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
- [7] P-H. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *Proc. 7th CAV*, pages 381–394. LNCS 939, 1995.
- [8] A. Olivero, J. Sifakis, and S. Yovine. Using abstractions for the verification of linear hybrid systems. In *Proc. 6th CAV*, pages 81–94. LNCS 818, 1994.
- [9] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *Proc. 6th CAV*. LNCS 818, 1994.
- [10] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In *Hybrid Systems Workshop*, MSI, Cornell University, October 1994.