12
A timed automaton model for ET-LOTOS verification

Christian Hernalsteen
University of Brussels, Computer Science Department
Boulevard du Triomphe CP 212, 1050 Brussels, Belgium
chernals@ulb.ac.be

Abstract

We present in this paper a method to transform ET-LOTOS expressions in a subclass of timed automaton with timers, where a timer is not restarted before its reset. We show that this subclass is equivalent to timed automata and we show that this model can be used for ET-LOTOS verification. We have implemented this transformation method and interfaced our tool with the KRONOS model-checker. It is then possible to verify temporal properties, expressed in TCTL, on ET-LOTOS specifications. We illustrate our tool with a small robot controller example.

Keywords

ET-LOTOS, Timed automaton, Model-checking, Tools

1 INTRODUCTION

In the last few years, many Formal Description Techniques have been extended to support the design of real-time systems. These systems are quite complex to design due to their timing constraints. Moreover they are often safety critical; failures can have disastrous consequences which can put, for instance, human life in peril. The development of such systems can therefore be eased with the support of formal techniques.

Many process algebras have been extended to allow the description of timed dependent systems, see [10] for an overview of FDT's supporting time. ET-LOTOS [9] is a timed extension of LOTOS which allows the modeling of real-time behaviors. This extension is currently used to define the timed semantics of the future ISO standard E-LOTOS (for Extended LOTOS). Such a language can only be useful however, if tools can support it and especially its real-time characteristics. Tools which deal with time already exists, like KRONOS [4] which allows to verify that a formula of the real-time logic TCTL [1] is verified by a system described by a timed automaton. Other tools based on Hybrid automata [2] allow to describe and analyze real-time systems like HyTech [6] and SHIFT [11]. Timed automata is a subclass of hybrid automata where the

Formal Description Techniques and Protocol Specification, Testing and Verification
T. Mizuno, N. Shiratori, T. Higashino & A. Togashi (Eds.)

automaton variables are used to represent the time passing. This model is less complex to handle than hybrid automata and is more adapted to real-time systems description.

In [5], a transformation method of an ET-LOTOS subset into timed automata is proposed but one of the timed operators of ET-LOTOS, the time capture, is not supported. We propose in this paper to extend this method in order to allow the transformation of this operator. To reach this goal, we use timed automaton with timers to represent ET-LOTOS expressions. We show that the resulting automaton model is a subclass of timed automaton with timers where a timer is never restarted before its reset and we prove that this subclass is equivalent to timed automata. Timed automata are decidable for reachability analysis [2]. We have then captured an ET-LOTOS subset including all the operators, with some restrictions on their use, which is decidable. A tool implementing the transformation has been developed and interfaced with the KRONOS model-checker. ET-LOTOS specifications can then be transformed in timed automaton and verified with KRONOS. Since timed semantics of E-LOTOS will be based on ET-LOTOS, it will be possible to extend this work to this future ISO standard.

The approach of transforming a real time process algebra in a automaton model for model-checking purposes have already bee used for RT-LOTOS [3]. In this work a specific automaton model (*Dynamic Timed Automaton*) and a transformation in two phases have been defined. With this transformation, the reachability analysis may be performed on the fly which is not possible with our compositional transformation.

2 ET-LOTOS

ET-LOTOS extends LOTOS with a global time which can be discrete or dense. We consider, in this paper, a dense time domain: $\mathbb{Q}^+$. Three timing constructs are added to the LOTOS ones. These can be used to restrict the time interval where an action can occur. With the *life reducer* (`a{d}; B`), the action `a` is limited to occur within `d` time units. It is not an mandatory for an observable action `a` to occur within `d` time units, but if it does not, the process behaves like `stop`. If `a` is the internal action (`i`), the action must occur within `d` time units (or be preempted by another alternative). The internal action is urgent. The *time capture* (`a @x; B`) allows to capture in an ET-LOTOS variable (`x`) the time at which the action `a` has occurred (the time elapsed between the moment the behavior has been offered and the one at which the action has occurred). The third operator is the *delay* (Δ^d `B`) where
the behavior B is offered to the environment after a waiting time of `d` time units.

The formal semantics of ET-LOTOS is given in terms of labeled transition systems composed of two kinds of transitions, namely discrete and timed. This semantics can be found in found in [9].

3 TIMED AUTOMATON WITH TIMERS

We define in this section a model of automaton extended with clocks and timers (also called *integrators* or *stop watch* in the literature). Clocks value grows uniformly in all vertices proportionally with the time passing. A timer is a clock which can be started and stopped on a transition.

Let $\mathcal{V} = \mathcal{H} \cup \mathcal{I}$ be an infinite set of variables having a value in the set of positive rationals $\mathbb{Q}^+$; $\mathcal{H}$ and $\mathcal{I}$ are respectively the infinite sets of clocks and timers with $\mathcal{H} \cap \mathcal{I} = \emptyset$. Let $\mathcal{S}$ denotes an infinite set of vertices and Lab a label set. Let $\Psi(\mathcal{V})$ be the set of constraints on variables defined as the smallest set satisfying: $\psi ::= true | u \prec c | u_1 - u_2 \prec c | x \prec c | \neg\psi | \psi \wedge \psi$ where $u, u_1, u_2 \in \mathcal{H}, x \in \mathcal{I}, c \in \mathbb{Z}$ *and* $\prec \in \{<, \leq\}$.

Definition 1 *A timed automaton with timers is defined as a tuple* $< S, V, R, E, s^0, Inv >$ *where*

$S \subset \mathcal{S}$ *is a finite set of vertices,*
$V \subset \mathcal{V}$ *is a finite set of variables,*
$R : S \times V \longrightarrow \{0, 1\}$ *is the rate labeling function,*
$E \subseteq S \times Lab \times \Psi(V) \times 2^V \times S$ *is the finite set of transitions,*
$s^0 \in S$ *is the initial vertex,*
$Inv : S \longrightarrow \Psi(V)$ *is the vertex invariant*

where $\forall x \in V \cap \mathcal{H}, \forall s \in S, R(s)(x) = 1$.

A transition $e = (s, a, \psi, V', s')$ from the vertex s to a vertex s' is labeled by an action (a). The transition can be fired if its guard ψ is evaluated to true and the variables of V' are reset when the transition occurs. We say that a transition starts (resp. stops) a timer x if $R(s)(x) = 0$ and $R(s')(x) = 1$ (resp. $R(s)(x) = 1$ and $R(s')(x) = 0$). The vertex invariant determines for each vertex whenever the time can progress, and hence whenever the automaton can stay in the vertex.

We define two subclasses of timed automaton with timers: *timed automaton* where all the variables are clocks ($V \subset \mathcal{H}$) and *timed automaton with semi-timers* where a timer is always reseted before being restarted ($\forall e = (s, a, \psi, V', s') \in E,\ \forall x \in V \cap \mathcal{I} : (R(s)(x) = 0$ *and* $R(s')(x) = 1) \Longrightarrow x \in V'$).

We denote v a valuation, i.e., a function which associates to each variable $x \in \mathcal{V}$ a value in $\mathbb{Q}^+$. The set of valuations on variables is denoted $\mathbb{V}$ and the one restricted to clocks is denoted $\mathbb{H}$. Valuation can be naturally extended to constraints on variables: for each $\psi \in \Psi(\mathcal{V})$ and $v \in \mathbb{V}$, $\psi(v)$ represents the value of the constraints ψ evaluated in v. The valuation $v[V := 0]$ represents the valuation v where all the variables of V have been reset to 0. For a valuation $v \in \mathbb{V}$, a rate labeling function R, and $t \in \mathbb{Q}^+$, $v + R.t$ denotes a new valuation v' such that for every variable $x \in \mathcal{V}$, $v'(x) = v(x) + R(x).t$. For a

valuation $v \in \mathbb{H}$, and $t \in \mathbb{Q}^+$, $v + t$ denotes a new valuation v' such that for every clock $u \in \mathcal{H}$, $v'(u) = v(u) + t$.

The semantics of a timed automaton with timers is given by the following definition.

Definition 2 *The model of a timed automaton with timers* $< S, V, R, E, s^0, Inv >$ *is the labeled transition system* $< \mathcal{Q}, \longrightarrow, q^0 >$ *where :*

1. $\mathcal{Q} = \{(s, v) | Inv_s(v), s \in S, v \in \mathbb{V}\}$
2. $q^0 = (s^0, v^0)$, *where* $\forall x \in V \;\; v^0(x) = 0$
3. *The transition relation* $\longrightarrow \subseteq \mathcal{Q} \times (Lab \cup \mathbb{Q}^+) \times \mathcal{Q}$ *is defined on* $\mathcal{Q}$ *by the smallest relation defined by the following rules:*

$$(R1)\ \frac{(s, a, \psi, V', s') \in E \wedge \psi(v)}{(s, v) \xrightarrow{a} (s', v[V' := 0])} \qquad (R2)\ \frac{\forall d' \leq d \;\; Inv_s(v + R(s).d')}{(s, v) \xrightarrow{d} (s, v + R(s).d)}$$

where $a \in Lab, v \in \mathbb{V}$, $d \in \mathbb{Q}^+$.

4 TRANSFORMATION OF ET-LOTOS

We provide in this section, a method to transform an ET-LOTOS expression in a timed automaton with timers where $Lab = L \cup \{i, \Delta, \varepsilon\}$ (Δ represents the expiration of a delay and ε an empty transition). This transformation is inspired from the work presented in [5]. We restrict ourself to some of the operators but the method can be extended to all the basic operators of ET-LOTOS (see [8]).

We first define two notations to handle rate labeling functions. We define $R = R_1 \cup R_2$ the union of two rate labeling functions $R_1 : S_1 \times V_1$ and $R_2 : S_1 \times V_2$ with $S_1 \cap S_2 = V_1 \cap V_2 = \emptyset$ as $R : (S_1 \cup S_2 \cup (S_1 \times S_2)) \times (V_1 \cup V_2)$ where $\forall s_1 \in S_1, s_2 \in S_2, x_1 \in V_1, x_2 \in V_2, x \in V_1 \cup V_2$:

$$R(s_i)(x_i) = R_i(s_i)(x_i)\ i \in \{1, 2\}$$
$$R([s_1, s_2])(x) = \begin{cases} R_1(s_1)(x) & if\ x \in V_1 \\ R_2(s_2)(x) & if\ x \in V_2 \end{cases}$$

We define the extension a rate labeling function with a new variable and vertex. If R is a rate labeling function and $(s, x) \notin Dom(R)$ a vertex and a variable, we define $R' = R \uplus (s, x)$ the new rate labeling function resulting from the extension of R with the couple (s, x) where :

$$\forall x' \in Dom_v(R), \forall s' \in Dom_s(R) \;\; R'(s')(x') = R(s')(x')$$
$$\forall x' \in Dom_v(R) \;\; R'(s)(x') = \begin{cases} 1 & if\ x' \in \mathcal{H} \\ 0 & if\ x' \in \mathcal{I} \end{cases}$$

$$\forall s' \in Dom_s(R) \;\; R'(s')(x) = \begin{cases} 1 & if\ x \in \mathcal{H} \\ 0 & if\ x \in \mathcal{I} \end{cases}$$

$$R'(s)(x) = 1$$

where $Dom_v(R)$ and $Dom_s(R)$ represent, respectively, the variable and vertex domain of the rate labeling function R.

The transformation method is defined in a compositional way, where the automaton corresponding to an ET-LOTOS expression depends on the automata of the operand's expressions. It results that produced automata may contain guard transitions with variables which does not belong to V since a variable is introduced when it is defined via a time capture and not when it is used via a guard. In all cases the final automaton is complete if the original ET-LOTOS specification is semantically correct.

1. The process **stop**:
   This process is transformed in $< \{s\}, \{u\}, R, \emptyset, s, Inv >$ where $u \in \mathcal{H}, Inv(s) = true$ and $R(s)(u) = 1$.
   This timed automaton with timers (automaton for short in the sequel) has no transition and has no variable, but the clock u is needed to represent the time passing as specified in the semantic rule (R2).
2. Action prefix:
   If $< S, V, R, E, s^0, Inv >$ is the automaton corresponding to the behavior B and $B' \equiv a@x\{d\}; B$ then the timed automaton corresponding to B' is given by:

   $$< S \cup \{s\}, V \cup \{x\}, R \uplus (s, x), E \cup \{(s, a, x \leq d, V, s^0)\}, s, Inv' >$$

   where $s \notin S, x \in \mathcal{I} \setminus V$ and for all $s' \in S$, $Inv'(s') = Inv(s')$.
   The activation function of the new vertex is defined by $Inv'(s) = true$ if the action a is observable, since the action is not obliged to occur before time d but only restricted to occur within this limit; if action $a = i$, the action must occur within the time limit, then the function Inv is defined as $Inv'(s) = x \leq d$. The new rate labeling function states that the timer x is frozen on all the vertices but s. If the time capture is not present, we introduce a new clock $u \in \mathcal{H} \setminus V$ and use it instead of x.
3. Delay:
   If $< S, V, R, E, s^0, Inv >$ is the automaton corresponding to B and $B' \equiv \Delta^d B$ then the timed automaton corresponding to B is given by :

   $$< S \cup \{s\}, V \cup \{u\}, R \uplus (s, u), E \cup \{(s, \Delta, u = d, V, s^0)\}, s, Inv' >,$$

   where $s \notin S, u \in \mathcal{H} \setminus V$.
   The label Δ represents the expiration of the delay. For all $s' \in S$, $Inv'(s') =$

$Inv(s')$ and $Inv(s) = u \leq d$; in this way, and due to the transition guard, the automaton is restricted and obliged to leave the vertex s after d time units.

4. Choice:
If $< S_i, V_i, R_i, E_i, s_i^0, Inv_i >$ is the automaton of $B_i, i \in \{1,2\}$ where $S_1 \cap S_2 = \emptyset$ and $V_1 \cap V_2 = \emptyset$ and if $B \equiv B_1 [] B_2$ then the automaton corresponding to B is given by :
$$< S_1 \cup S_2 \cup (S_1 \times S_2), V_1 \cup V_2, R_1 \cup R_2, E \cup E_1 \cup E_2, [s_1^0, s_2^0], Inv >$$
where E is given by:

$$\begin{aligned} E &= \{([s_1,s_2], a, \psi, V', s_1') \mid (s_1, a, \psi, V', s_1') \in E_1, a \in L \cup \{i\}\} \\ &\cup \{([s_1,s_2], a, \psi, V', s_2') \mid (s_2, a, \psi, V', s_2') \in E_2, a \in L \cup \{i\}\} \\ &\cup \{([s_1,s_2], \omega, \psi, V', [s_1', s_2]) \mid (s_1, \omega, \psi, V', s_1') \in E_1,\ \omega \in \{\Delta, \varepsilon\}\} \quad (1) \\ &\cup \{([s_1,s_2], \omega, \psi, V', [s_1, s_2']) \mid (s_2, \omega, \psi, V', s_2') \in E_2,\ \omega \in \{\Delta, \varepsilon\}\} \quad (2) \end{aligned}$$

For all $s_i \in S_i$, $Inv(s_i) = Inv_i(s_i)$ and $Inv([s_1, s_2]) = Inv_1(s_1) \wedge Inv_2(s_2)$ to represent that time passes at the same rate in both processes. The last two transition sets (equations 1 and 2) represent the fact that delay and empty transitions do not resolve the choice.

5. Parallel composition:
If $< S_i, V_i, R_i, E_i, s_i^0, Inv_i >$ is the automaton of $B_i, i \in \{1,2\}$ where $S_1 \cap S_2 = \emptyset$ and $V_1 \cap V_2 = \emptyset$ and if $B \equiv B_1 |[\Gamma]| B_2$ where $\Gamma \subseteq L$, then the automaton corresponding to B is given by:
$$< S_1 \times S_2, V_1 \cup V_2, R_1 \cup R_2, E, [s_1^0, s_2^0], Inv >$$
where the transition set E is given by:

$$\begin{aligned} E &= \{([s_1,s_2], a, \psi, V', [s_1', s_2]) \mid (s_1, a, \psi, V', s_1') \in E_1, a \in L \cup \{\Delta, i, \varepsilon\} \setminus \Gamma\} \\ &\cup \{([s_1,s_2], a, \psi, V', [s_1, s_2']) \mid (s_2, a, \psi, V', s_2') \in E_2, a \in L \cup \{\Delta, i, \varepsilon\} \setminus \Gamma\} \\ &\cup \{([s_1,s_2], a, \psi_1 \wedge \psi_2, V_1 \cup V_2, [s_1', s_2']) \mid (s_i, a, \psi_i, V_i, s_i') \in E_i, a \in \Gamma \cup \{\delta\}\} \end{aligned}$$

For all $[s_1, s_2] \in S_1 \times S_2$, $Inv([s_1, s_2]) = Inv_1(s_1) \wedge Inv_2(s_2)$ to state that the time passes at the same rate in the two processes. The first two sets represent transitions resulting from independent actions while the last one represents synchronized actions.

6. The guard:
If $< S, V, R, E, s^0, Inv >$ is the automaton of B and $B' \equiv [G] \longrightarrow B$ where $G \in \Psi(V)$ then the timed automaton of B' is given by :
$$< S \cup \{s\}, V \cup \{u\}, R \uplus (s,u), E \cup (s, \varepsilon, G, V, s^0), s, Inv' >$$
where $s \notin S$ and $u \notin V$.
For all $s' \in S$, $Inv'(s') = Inv(s')$ and $Inv'(s) = (u = 0) \vee \neg G$.

An empty transition is added to verify the value of the guard. The activation function of the initial vertex forces the transition to occur immediately whenever the guard is evaluated to true. In the other case, the function is equivalent to true since, following the ET-LOTOS semantics, the expression can be aged indefinitely when the guard is false.

It can be easily proved that the automaton model produced with this method, when the guard expressions are restricted to $\Psi(\mathcal{V})$, is a timed automaton with semi-timers. Indeed, timers are only introduced to represent time capture. These timers are never restarted once they are stopped.

5 DECIDABILITY OF TIMED AUTOMATON WITH SEMI-TIMERS

We define in this section a transformation of timed automata with semi-timers to timed automata and we show that this transformation provides equivalent automata modulo the strong bisimulation. Since reachability is decidable on timed automaton [2], we thus prove that this is also the case for timed automaton with semi timers, and for the ET-LOTOS subset considered in this paper. Moreover this result can be extended to all the basic ET-LOTOS operators [8].

The principle of the transformation is to substitute any timer x with the difference between two clocks x_u and x_l. The clock x_u represents the last time the timer has been started and x_l the last time it has been stopped. The transformation function resets both clocks when the corresponding timer is reset and x_l is reset when the transition stops the timer. The formal definition of this transformation function is given below.

Definition 3 *Let $T_r : \mathcal{T}_{st} \longrightarrow \mathcal{T}$ be a transformation function from timed automaton with semi-timers into timed automaton. This function is defined by $T_r(< S_1, V_1, R_1, E_1, s_1^0, Inv_1 >) = < S_1, H_2, E_2, s_1^0, Inv_2 >$ such that*

- $H_2 = (V_1 \cap \mathcal{H}) \cup \{x_l, x_u \mid x \in V_1 \cap \mathcal{I}\}$
- $E_2 = \{(s, a, \psi[I], H', s') \mid (s, a, \psi, V', s') \in E_1,$
  $H' = (V' \cap \mathcal{H}) \cup \{x_l, x_u \mid x \in V' \cap \mathcal{I}\} \cup$
  $\{x_l \mid R_1(s)(x) = 1 \text{ and } R_1(s')(x) = 0,\ x \in V_1 \cap \mathcal{I}\},$
  $I = \{x \mid x \in V_1 \cap \mathcal{I}, R_1(s)(x) = 0\}\}$
- $\forall s \in S_2, Inv_2(s) = Inv_1(s)[I], I = \{x \mid x \in V_1 \cap \mathcal{I}, R_1(s)(x) = 0\}.$

where $\forall x \in V_1 \cap \mathcal{I}, \{x_u, x_l\} \cap V_1 = \emptyset$ and $\forall \psi \in \Psi(\mathcal{V}), I \subseteq \mathcal{I}$, $\psi[I] \in \Psi(\mathcal{V})$ represents the constraint ψ where all the instances of $x \in I$ have been replaced by $x_u - x_l$ and where all instances of $x \in \mathcal{I} \setminus I$ have been replaced by x_u where $x_u, x_l \in \mathcal{H}$ such that $\forall x \neq y, x_l \neq y_l$ and $x_u \neq y_u$.

Let's remark that $\psi[I]$ belongs to $\Psi(\mathcal{V})$ since $I \subset \mathcal{I}$ and the difference between two timers $x_1, x_2 \in \mathcal{I}$ is not allowed in $\Psi(\mathcal{V})$

In [7] we proof formally that the transformation function T_r preserves the semantics of timed automaton with semi-timers modulo the strong bisimulation i.e. $\forall M \in \mathcal{T}_{st}, T_r(M)$ is strongly bisimilar to M. It results that state reachability is decidable on our timed automaton model with semi timers.

6 IMPLEMENTATION

This transformation method has been implemented in a tool which takes an ET-LOTOS specification and transform it in a timed automaton for KRONOS. The tool supports all the basic ET-LOTOS operators, including the process instantiation. Usual constraints are put on recursive behaviors: it is not allowed through a parallel and on the left hand side of the enabling and disabling operators. The implementation of the method has been optimized to reduce the number of transitions, vertices and clocks produced. For instance, a new clock is not introduced for each prefix operator and Δ transitions are, in some cases, removed and replaced by an extended guard on the subsequent transitions. Optimizations are also done on the treatment of process instantiation in order to avoid, as much as possible, the duplication for each instantiation, of an automaton representing the process instantiated.

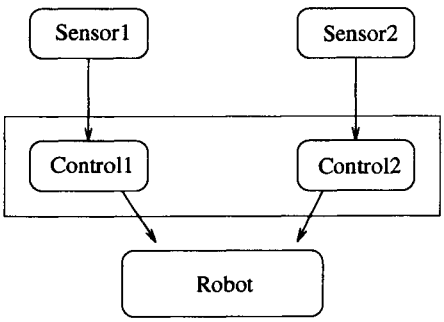

Figure 1 The robot controller structure

The KRONOS tool allows to verify TCTL formulas on timed automata. The real-time logic TCTL [1] is an extension of CTL where the two main operators have been extended with timing constraints to allow quantitative temporal reasoning. The TCTL formulas accepted by KRONOS are defined by the following grammar:

$\rho ::= init \mid enable(a) \mid after(a) \mid c \sim r \mid \neg\rho \mid \rho_1 \wedge \rho_2 \mid \rho_1 \exists\mathcal{U}_I \rho_2 \mid \rho_1 \forall\mathcal{U}_I \rho_2$

where r is a positive integer, $\sim \in \{<, \leq, =, \geq, >\}$, c is a clock of the timed automaton under verification, I is a time interval, *init* represents the initial

state with all the clocks set to zero, $enable(a)$ defines the states where transitions labeled with a are enabled and $after(a)$ defines the set of states reached by transitions labeled by a.

Intuitively, $\rho_1 \exists \mathcal{U}_I \rho_2$ means that it exists a run which continuously verifies ρ_1 until a state which verifies ρ_2 is reached at a time $t \in I$. In the same way, $\rho_1 \forall \mathcal{U}_I \rho_2$ means that all runs satisfy the above property. Some typical abbreviations are used such as $\forall \Diamond_I \rho$ for $true \forall \mathcal{U}_I \rho$, $\exists \Diamond_I \rho$ for $true \exists \mathcal{U}_I \rho$, $\exists \Box_I \rho$ for $\neg \forall \Diamond_I \neg \rho$ and $\forall \Box_I \rho$ for $\neg \exists \Diamond_I \neg \rho$. The unrestricted operators correspond to the operator subscripted by $[0, \infty[$. This logic allows to specify complex safety and liveness properties.

A robot controller example

Let us take a small example to illustrate the use of the tools. We consider a robot controller which provides commands, based on recent measurements of the environment, to a robot. The system consists of five components: two sensors, two controller processes and the robot itself (figure 1). The sensors probe the environment periodically; all 8 time units for the first one and 12 time units for the second. A controller process is associated to each sensor. The sensor's readings are sent to the controllers which take some amount of time to process the information and to send the new command to the robot; 1 time unit for the first one and 2 time units for the second. These processes share the same processor and are controlled by a simple scheduler. This non-preemptif scheduler gives the processor to each controller for a given amount of time. When a controller receive the processor its waits for the reading of its sensor. If this reading does not arrive in the time interval defined by the scheduler, the processor is given to the other controller. On the other hand, the scheduler waits for the end of the reading processing before giving the processor to the other controller. In all cases, the scheduler never interrupts a reading processing. The problem is to determine the time period given to each controller to insure some time limits on the processing of the sensor's readings. The processing of the first controller must be started within 6 time units after the corresponding sensor's reading. This time limit is set to 8 time units for the second controller.

This robot controller system has been specified in ET-LOTOS (figure 2). The sensors are described by two processes **Sensor** which generate periodically the action **Reading** representing a sensor reading. These actions are synchronized with the **TimeOut** processes which specify the timing requirements on the sensor's reading processing. Once a sensor's reading is captured, the process offers to its associated controller to start its processing (action **StartControl**). If the processing is not started in the time limit an exception error is raised (action **error**). The controllers are represented by two simple processes (**Controller**); they wait for the sensor reading and then execute their processing (represented by the delay) before exiting. The **Scheduler**

```
hide StartControl1, StartControl2, Reading1, Reading2 in (
    (Scheduler |[StartControl1, StartControl2]| (TimeOut1 ||| TimeOut2))
    |[Reading1, Reading2]| (Sensor1 ||| Sensor2)
where
    Scheduler ::=
        (Controller1 [] Δ^period1 exit) >>
        (Controller2 [] Δ^period2 exit) >> Scheduler
    Controller1 ::= StartControl1; Δ^1 exit
    Controller2 ::= StartControl2; Δ^2 exit
    TimeOut1 ::= Reading1; (StartControl1; TimeOut1 [] Δ^6 error; stop)
    TimeOut2 ::= Reading2; (StartControl2; TimeOut1 [] Δ^8 error; stop)
    Sensor1 ::= Reading1; Δ^8 Sensor1
    Sensor2 ::= Reading2; Δ^12 Sensor2
```

Figure 2 The ET-LOTOS robot system

process gives the processor to the two controllers in a cyclic way. The choice expressions specify the period of processor allocation. If a controller has not begun its processing before the **period** deadline, the processor is given to the other controller. The different synchronized actions are hidden to insure their urgency; they occur as soon as all the synchronized components offered the considered action. This ET-LOTOS specification does not use the time capture operator but is, nevertheless, well adapted to illustrate our verification process.

We have used our transformation tool to produce the KRONOS timed automaton corresponding to the robot controller specification. The transformation tool has produced, in less than 2 seconds, a timed automaton of 207 states, 659 transitions and 9 clocks on a mono-processor SUN Ultra1 with 64MB. We have then used KRONOS to verify some safety requirements on the system. We first verify that two sensor's readings are not processed at the same time, which can be described in TCTL by:

$$init \Rightarrow \forall\Box(after(\texttt{StartControl1}) \Rightarrow \forall\Box_{[0,1[}\ \neg(enable(\texttt{StartControl1})\ or\ enable(\texttt{StartControl2})))$$

$$init \Rightarrow \forall\Box(after(\texttt{StartControl2}) \Rightarrow \forall\Box_{[0,2[}\ \neg(enable(\texttt{StartControl1})\ or\ enable(\texttt{StartControl2})))$$

These two formulas state that during the processing of one of the sensor, no other processing can start. The first formula has been verified in $7.3s$ and the second in $39.9s$.

We have then used KRONOS to define the values of **period1** and **period2** which insures that all the sensor's readings are processed in time. The system is then considered safe if the action **error** is not reachable which can be represented by the following TCTL formula:

$$init \Rightarrow \forall\Box\neg enable(\texttt{error})$$

This formula states that it is not possible to reach a state where the action **error** is enabled from the initial state of the automaton.

period1	5	6	7	5	4	3	3	4	4	4	3
period2	3	3	3	4	3	3	4	4	5	6	5
time	23s	27s	30s	88s	21s	21s	28s	27s	95s	453s	39s
eval	true	true	true	false	true	true	true	true	false	false	true

Table 1 Verification results with KRONOS

The table 1 shows the results obtained for various values of the **period** parameters; the line labeled by **time** gives the running times given in seconds and the **eval** one gives the results of the TCTL formula evaluation. We have tried various configurations and have found 8 configurations which insure the safety of the system. Other kind of systems have also been verified where processing time of the controllers are non deterministic and where a premption time is considered. The ET-LOTOS specification must only be slightly changed to consider these systems.

7 CONCLUSION AND FURTHER WORKS

We have presented, in this paper, a method which allows the transformation of all the ET-LOTOS operators in a subclass of timed automaton with timers. We have shown that this subclass is equivalent to timed automaton by providing a conservative transformation between the two models. This work has allowed to capture a subset of ET-LOTOS where state reachability is decidable and to develop a tool which allows the verification of real-time logic formulas, expressed in TCTL, on an ET-LOTOS specification. This approach gives nice results, as shown with our robot controller example, but suffers of the state explosion problem. Even for quite small ET-LOTOS specifications the corresponding timed automata are large. Moreover, the KRONOS tool can only analyze, in a reasonable computation time and memory space, automata with no more than approximatively 50.000 states. The execution time depends also of the TCTL formula under verification. Further works will optimize the transformation method to obtain smaller timed automata. Nevertheless, this optimization does not really resolve the state explosion problem.

Our tool is limited to a subset of ET-LOTOS. We are extending it to full ET-LOTOS. The idea is to used the hybrid automaton model of HyTech [6] which can be used to support the data part of ET-LOTOS. This tool implements a semi-decision procedure for the reachability analysis of hybrid automata. This method does not resolve the state explosion problem and moreover, the new intermediate model will be more complex than timed automata. We are expecting from this last point, less effective results than with KRONOS.

Our approach is a first step in the analyze and tool development of timed process algebras. It can be used for other process algebras and especially for E-LOTOS whose timed semantics is based on ET-LOTOS. Another approach will be to use an intermediate representation of ET-LOTOS which avoid the explosion problem, like Timed Petri Nets, and to develop a verification technique adapted to this model. It will then be possible to develop more adapted methods to the verification of timed process algebras than the one used here.

REFERENCES

[1] R. Alur and C. Courcoubetis and D.L. Dill (1990). Model-checking for real-time systems, in Proceedings of the 5th Symposium on Logic Computer Science, pages 414-425.

[2] R. Alur, C. Courcoubetis, N. Hallbwachs, T.A. Henzinger, P.-H HO, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine (1995). The algorithmic analysis of hybrid systems, in TCS 138, pages 3-34.

[3] J.-P. Courtiat and R.C. de Oliveira (1995). A Reachability Analysis of RT-LOTOS Specifications, in Formal Description techniques VIII, chapman & Hall, pages 117-124.

[4] C. Daws and A. Olivero and S. Tripakis and S. Yovine (1996). The tool KRONOS, in Hybrid Systems III, Lecture Notes in Computer Science 1066, Springer-Verlag

[5] C. Daws and A. Olivero and S. Yovine (1994). Verifying ET-LOTOS programs with KRONOS, in Seventh International Conference on Formal Description Techniques, Chapman & Hall, pages 227-242.

[6] T. A. Henzinger and P. H. Ho (1994). HyTech: the cornell HYbrid TECHnology tool, in Hybrid Systems II, Lecture Notes in Computer Science 999, Springer-Verlag, pages 265-294.

[7] C. Hernalsteen (1997). Timed automaton with semi timers for ET-LOTOS verification, Technical report TR-363, Computer science department, Free University of Brussels.

[8] C. Hernalsteen and T. Massart (1995). An extended timed automaton to model ET-LOTOS Specification, in participant's proceedings of DARTS'95, pages 95-112

[9] Luc Léonard and Guy Leduc (1994). An Enhanced Version of Timed LOTOS and Its Application to a Case Study, in FORTE-VI, pages 483-500, North-Holland.

[10] X. Nicollin and J. Sifakis (1991). An overview and synthesis on timed process algebra, in Proc. 3rd Workshop on Computer-Aided Verification.

[11] P. Varaiya (1997). SHIFT: A language for simulating Interconnected Hybrid Systems, in Hybrid and Real-Time Systems, Lecture Notes in Computer Science 1201, Springer-Verlag.