From the Specification to the Scheduling of Time-Dependent Systems

Christophe Lohr and Jean-Pierre Courtiat

LAAS – CNRS, 7 avenue du Colonel Roche 31077 Toulouse Cedex, France
{lohr,courtiat}@laas.fr

Abstract. This paper introduces and formalizes a new variant of Timed Automata, called Time Labeled Scheduling Automata. A Time Labeled Scheduling Automaton is a single clock implementation model expressing globally the time constraints that a system has to meet. An algorithm is presented to show how a Time Labeled Scheduling Automaton can be synthesized from a minimal reachability graph derived from a high-level specification expressing the composition of different time constraints. It is shown how the reachability graph may be corrected before synthesizing the Time Labeled Scheduling Automaton to remove all its potentially inconsistent behaviors. Current applications of the model are the scheduling of interactive multimedia documents and a simple illustration is given in this area.

Keywords: Process Algebra, RT-Lotos, Timed Automata, Minimal Reachability Graph, Temporal Consistency, Time Labeled Scheduling Automata

1 Introduction

Starting from a high-level specification of a time-dependent system that explicitly expresses a composition of time constraints, our purpose is to derive an operational specification of that system based on a new temporal model, called a Time Labeled Scheduling Automaton (TLSA) for scheduling.

The TLSA model has been designed within the framework of the RT-Lotos project at LAAS-CNRS, and the associated tools have been integrated within the rtl (RT-Lotos Laboratory) tool environment. It is assumed that the system high-level specification is written in RT-Lotos [8], but results are also applicable to other formalisms that can express composition of time constraints, like timed automata [3], temporal extensions of Petri nets [17] and process algebra [11].

The main purpose of the paper is to introduce the TLSA model and to show how a TLSA can be synthesized from a (minimal) reachability graph expressing the global behavior of a specification. A TLSA is a variant of classical timed automata intended to express *globally* the time constraints that a system has to meet. It is *not* a specification model, but rather an operational implementation model using a single clock.

Performing the synthesis of a TLSA from a reachability graph presents a major advantage: it allows to take into account the results of the reachability graph

W. Damm and E.-R. Olderog (Eds.): FTRTFT 2002, LNCS 2469, pp. 129–145, 2002.

analysis at the level of the TLSA synthesis. Thus, when the verification process has detected that a reachability graph features inconsistent behaviors (a behavior is said to be *inconsistent* if it leads to a *deadlock state*, i.e. a state where no transition is enabled), it is easy to eliminate these inconsistencies from the reachability graph just by removing the paths leading to the deadlock states, and then to generate the TLSA from the corrected reachability graph. Thus, a consistent operational model of the system may be obtained without having to modify the high-level specification. This approach has been completely automated, and it has proven to be very powerful for the design and scheduling of interactive multimedia presentations [13,14,15]. It will be illustrated in the next paragraph on a simple example.

The paper is organized as follows: Section 2 provides the intuition of the TLSA model and its use, based on an example derived from the multimedia area. Section 3 formalizes the Timed Labeled Scheduling Automaton model and Section 4 proposes an algorithm to derive a TLSA from a (minimal) reachability graph. Finally, Section 5 reviews some current work in this area, and Section 6 draws some conclusions.

2 Intuition through an Example

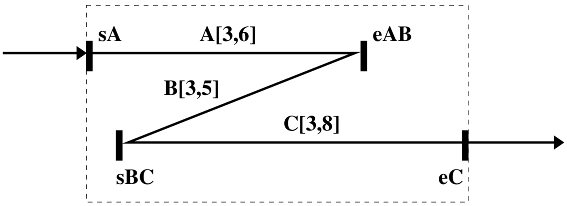
This example deals with the specification and scheduling of interactive multimedia documents. Within this framework, a formal method is used to describe the temporal constraints characterizing both the presentation of the media composing a document and the interactions with the user, as well as the global synchronization constraints among media defined by the author of the document. The formal specification, derived from the high-level authoring model, consists essentially in a composition of simple (RT-Lotos) processes describing elementary time constraints. Reachability analysis is then performed on the formal specification to check the temporal consistency of the specification. Different variations of this temporal consistency property have been defined, depending on whether the actions leading to potential deadlock states are controllable or not (see [15,16] for details).

Consider the example depicted in Fig. 1a. The author of this multimedia scenario wishes to present three media called A, B and C respectively. The presentation durations of these media are respectively defined by the following time intervals: [3,6], [3,5] and [3,8]. This means that, for example, the presentation of media A should last at least 3 seconds and 6 seconds at the most. Thus, from the author's perspective, any value belonging to the time interval is acceptable.

Moreover, the author expresses the following synchronization constraints:

1. the presentation of media A and B should end simultaneously;
2. the presentation of media B and C should start simultaneously;
3. the start of this multimedia scenario is determined by the start of A, and its end is determined either by the end of A and B, or by the end of C.

The RT-Lotos specification is presented in Fig. 1b, and the associated (minimal) reachability graph obtained with the rtl tool is presented in Fig. 2a.



(*a*)

```
specification example : exit
 type natural is ... endtype
behavior
 hide sA, sBC, eAB, eC  in
 ( (
   (sA; delay(3,6) eAB{3}; stop)
   |||
   (sBC; delay(3,8) eC{5}; stop)
  )
  |[sBC, eAB]|
  (sBC; delay(3,5) eAB{2}; stop)
 )
 |[sA, sBC]|
 (sA; latency(inf) sBC; stop)
endspec

Note that:
- delay(d1,d2) means delay(d1)latency(d2-d1)
- inf denotes infinity
```

(*b*)

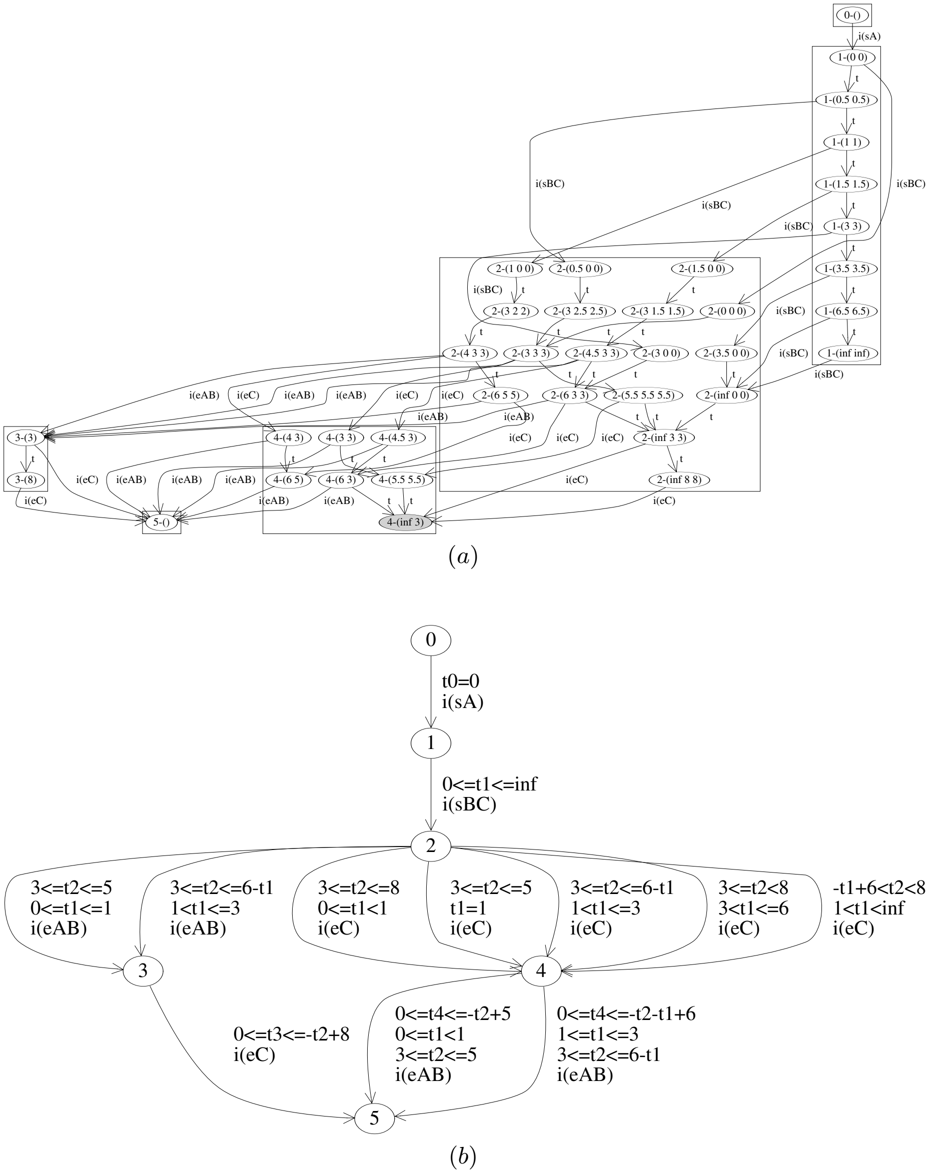
Fig. 1. (*a*) Multimedia scenario and (*b*) RT-Lotos specification

The minimal reachability graph allows the temporal consistency of the multimedia scenario to be analyzed. The scenario is said to be *potentially consistent*, if there exists at least one path starting with `i(sA)` (action `i(sA)` characterizes the start of medium A presentation, hence the start of the scenario presentation) and leading to the end of the scenario presentation (either the occurrence of actions `i(eC)` or `i(eAB)`). Looking at Fig. 2a, the scenario is indeed potentially consistent.

Analyzing the high-level requirements, one may note that the start of the presentation of B and C has been left completely unspecified regarding the start of A. On such a simple example, it is easy to realize that if the start of media B and C is triggered too late with respect to the start of A, some time constraints will never be met (for instance, the constraint that A and B should finish simultaneously). This characterizes potentially inconsistent behaviors in the presentation of the multimedia scenario, which are represented in the reachability graph by the paths starting from the initial state and leading to deadlock states (see, for instance, state denoted *4-(inf 3)* at the bottom of Fig. 2a).

Deadlock states are not desirable and therefore, one wants to eliminate them by removing from the reachability graph all the paths starting from the initial state and leading to these deadlock states. Deadlock states are states without outgoing edges, which are not final states. Edges leading to those deadlock states and nowhere elsewhere are called inconsistent paths and removed. By removing all the inconsistent paths from the reachability graph (see Fig. 3a) we ensure the scheduling of a consistent scenario applying the concept of controllability, that is, proposing a valid temporal interval inside which the scenario is consistent [13,14].

Fig. 2b represents the `TLSA` derived from the reachability graph of Fig. 2a, following the algorithm presented in Section 4. This `TLSA` cannot be used as a basis for scheduling the multimedia scenario, since it features the same inconsistent behaviors as its associated reachability graph. For example, let us assume the behavior where actions `i(sA)`, `i(sBC)`, `i(eC)` occur respectively at times $t_0 = 0$, $t_1 = 5$ and $t_2 = 6$, leading the scenario in state 4 of Fig. 2b. It can be note that, in this state, there is no transition enabled, since the two enabling



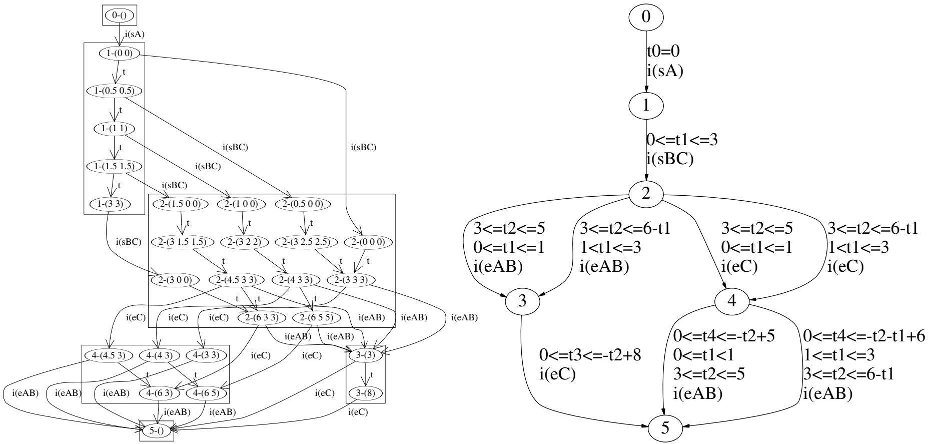
(*a*)

(*b*)

Fig. 2. (*a*) Minimal Reachability Graph and (*b*) associated TLSA

conditions (see the formal semantics of the TLSA in the next paragraph) feature either $0 \leqslant t_1 < 1$ or $1 \leqslant t_1 \leqslant 3$ whereas $t_1 = 5$.

Fig. 3b represents the TLSA derived from the reachability graph of Fig. 3a. This TLSA allows implementation of a consistent scheduling of the multimedia scenario, where the presentation of media B and C should begin no later than 3



(*a*) (*b*)

Fig. 3. (*a*) Consistent Minimal Reachability Graph and (*b*) associated TLSA

seconds after the beginning of A (see firing condition $0 \leqslant t_1 \leqslant 3$ defined for the transition between states 1 and 2). This information appears explicitly at the level of a transition of the TLSA, whereas it is hidden in the state characterization of the reachability graph.

Therefore, the proposed approach consists in the three following steps:

1. using RT-Lotos for expressing a composition of time constraints,
2. deriving the reachability graph of the RT-Lotos specification and removing all its inconsistent behaviors to obtain a *consistent* reachability graph,
3. performing the synthesis of the TLSA from the consistent reachability graph.

The TLSA may then be used for scheduling purposes, in the sense that it is an implementation model that defines when actions should occur to meet the initial composition of time constraints.

Semantics of the TLSA model will be presented in Section 3 and the algorithm for deriving a TLSA from a reachability graph will be described in Section 4.

3 Time Labeled Scheduling Automata

3.1 Introduction

Timed automata [3] have been proposed to model finite-state real-time systems. A timed automaton has a finite set of control states and a finite number of clocks. All clocks proceed at the same rate and measure the amount of time that has elapsed since they were started or reset. Each transition of the system might reset some of the clocks, and features an associated enabling condition expressed as a constraint on the values of the clocks.

A TLSA enjoys some specific features that differentiate it from classical timed automata. A TLSA has as many clocks as there are control states in the automaton these clocks being called `timers`. Each timer measures the time during which the automaton remains in a control state. No explicit function is defined to determine when timers should be started or reset. Indeed, the timer associated with a control state is reset when the automaton enters the control state, and its current value is frozen when the automaton leaves the control state[1]. Two timed conditions are associated with each transition of the automaton:

1. the *firing window* (denoted W) that defines the temporal slot during which the transition may be fired. It takes the form of an inequality to be satisfied by the timer associated with the transition input control state.
2. the *enabling condition* (denoted K) that defines the temporal constraints to be satisfied to fire the transition. It is expressed as a conjunction of inequalities to be satisfied by a subset of timers excluding the timer associated with the input control state of the transition.

Note: W, the temporal firing window, relates only to the current timer. Intuitively, W expresses the time during which the system is allowed to stay in the current control state. Note also that K, the enabling condition, relates only to past timers and expresses which transitions are enabled with respect to the past behavior of the system.

3.2 Formal Definition of a Time Labeled Scheduling Automaton

Let L be a set of action labels. Let $D = \{t \in \mathbb{Q} \mid t > 0\}$ the time domain, $D_0 = D \cup \{0\}$ and $D_0^\infty = D_0 \cup \{\infty\}$.

Let $\perp$ denote the "undefined" value, and $D_{0\perp} = D_0 \cup \{\perp\}$. By definition, we state that $\perp + \delta = \perp$, $\perp - \delta = \perp$ for any $\delta \in D_0^\infty$.

Let $T_{set} = \{t_i | i \in [0, n-1]\}$ be a set of *timers*, with $t_i \in D_{0\perp}$. Within this context, a timed condition is a conjunction of inequalities of the form $m \prec t_i \prec M$ where m, M are linear expressions of constants (in D_0^∞) *and timers* t_j ($t_j \in T_{set}$, $j \neq i$) and $\prec \in \{<, \leqslant\}$.

Let $\mu = (\mu^0, \ldots, \mu^{n-1}) \in D_{0\perp}^n$ be the value of timers t_i with $i \in [0, n-1]$, and $\mathcal{K}$ a timed condition defined on a subset of timers t_i. Notation $\mu \models \mathcal{K}$ indicates that all inequalities of $\mathcal{K}$ are *true* when replacing timers t_i by their values μ^i.

Definition 1 (Time Labeled Scheduling Automaton).
A Time Labeled Scheduling Automaton is a 3-tuple (S, E, s_0) *where:*

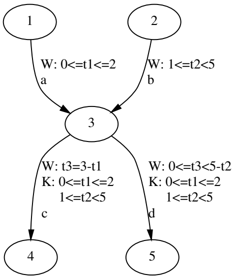
- $S = \{s_0, \ldots, s_{n-1}\}$ *is a finite set of control states;*
- E *is a finite set of transitions of the form* (s_i, s_j, W, K, a), *where* $s_i, s_j \in S$ *are the source and destination control states of the transition,* W *and* K *are timed conditions,* $a \in L$ *is a labeling action*
- s_0 *is the initial control state.*

[1] For this reason, we said previously that a TLSA is a single clock model, since, in any control state, there is one and only one running timer.

The *firing window* W of a transition (s_i, s_j, W, K, a) is a timed condition defined as an inequality of the form $m \prec t_i \prec M$ where m, M are linear expressions of constants and timers t_k ($t_k \in T_{set}$, $k \neq i$). If either m or M is equal to $\perp$ when replacing timers t_k by their value μ^k, then we assume $\mu \nvDash W$.

The *enabling condition* K of a transition (s_i, s_j, W, K, a) is a timed condition defined as a conjunction of inequalities of the form $\mathcal{I}_k = (m \prec t_k \prec M)$, where $k \neq i$, and m, M are linear expressions of constants and timers t_l ($t_l \in T_{set}$, $l \neq k$ and $l \neq i$). If t_k is equal to $\perp$, then we assume $\mu \vDash \mathcal{I}_k$; furthermore, if either m or M is equal to $\perp$ when replacing timers t_l by their value μ^l, then we assume $\mu \vDash \mathcal{I}_k$.

These rules are illustrated in Fig. 4.



Assuming that we are in control state 1, transition $1 \xrightarrow{a} 3$ is fired inside temporal window $0 <= t_1 <= 2$. Coming from control state 1, transition $3 \xrightarrow{d} 5$ is not enabled, because $t_2 = \perp$ and then `W` is false (see the definition of the firing window). Looking at enabling condition `K` of transition $3 \xrightarrow{c} 4$, the first inequality is satisfied ($0 <= t_1 <= 2$), as well as the second one, since $t_2 = \perp$ (see the definition of the enabling condition). Transition $3 \xrightarrow{c} 4$ will then be fired at time $t_3 = 3 - t_1$.

Fig. 4. Illustrations of the firing rules

A labeled transition system $LTS(TLSA)$ is associated with each time labeled automaton $TLSA$. A state (s, μ) of $LTS(TLSA)$, also called a configuration, is fully described by specifying the control state s of $TLSA$ and the value $\mu \in D^n_{0\perp}$ of all timers of the automaton. The transitions of $LTS(TLSA)$ correspond either to explicit transitions of $TLSA$, representing the occurrence of an action, or to implicit transitions representing the passage of time. The former are described by the transition successor rule and the latter by the time successor rule.

Definition 2 (Initial state of *LTS(TLSA)*).
The initial state of $LTS(TLSA)$ *is the configuration* (s_0, μ_0)*, where* $\mu_0 \in D^n_{0\perp}$ *with* $\mu_0 = (0, \perp, \dots, \perp)$.

Definition 3 (Explicit transition of *LTS(TLSA)*).
Let $(s, \mu) \in LTS(TLSA)$ *and* $(s, s', W, K, a) \in E$ *a transition of* $TLSA$*. If* $\mu \vDash W$ *and* $\mu \vDash K$*, then* $(s, \mu) \xrightarrow{a} (s', F(s, s', \mu))$.

where $F : S \times S \times D^n_{0\perp} \to D^n_{0\perp}$ *is the function*[2] *defined as:*

$$F(s_i, s_j, \mu) = \mu' \quad \textit{where} \quad \begin{cases} \mu'^j = 0 & \\ \mu'^k = \perp & \forall k \neq i \mid (s_k, s_j, W, K, a) \in E \\ \mu'^l = \mu^l & \forall l \in [0, n-1], l \neq j, l \neq k \end{cases}$$

[2] In the example of Fig. 4, while firing transition $1 \xrightarrow{a} 3$, timer t_2 is re-initialized with $\perp$ by function F.

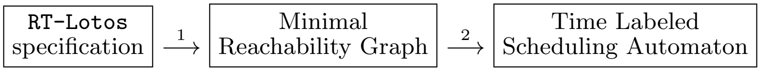


Fig. 5. Derivation of the TLSA

The role of function F is to avoid possible firing history conflicts in cyclic TLSA. In the presence of cycles and in the absence of function F, an action fired in a previous iteration around the cycle but not belonging to the current flow of execution, could influence the future conditions W and K.

Definition 4 (Implicit transition of *LTS(TLSA)*).
Let $(s, \mu) \in LTS(TLSA)$ and $\delta \in D$. If $act(s, G(s, \mu, \delta))$, then $(s, \mu) \xrightarrow{\delta} (s, G(s, \mu, \delta))$.

where $G : S \times D^n_{0\bot} \times D \to D^n_{0\bot}$ is the function defined as:

$$G(s_i, \mu, \delta) = \mu' \quad where \quad \begin{cases} {\mu'}^i = \mu^i + \delta \\ {\mu'}^k = \mu^k & \forall k \neq i \in [0, n-1] \end{cases}$$

and where $act : S \times D^n_{0\bot} \to \{true, false\}$ is the predicate that indicates whether there is, for configuration (s_i, μ), at least one enabled transition (s_i, s_j, W, K, a) (i.e. a transition satisfying both its firing window and its enabling condition), thus authorizing a temporal progression. This predicate is defined as:

$$act(s_i, \mu) = \bigvee_{\forall (s_i, s_j, W, K, a) \in E} (\mu \vDash W) \wedge (\mu \vDash K)$$

4 An Algorithm to Synthesize a TLSA

In this section, we develop an algorithm to synthesize a TLSA from a minimal reachability graph, itself derived from a RT-Lotos specification.

As illustrated in Fig. 5, the first sub-section summarizes the main steps of the RT-Lotos reachability analysis. The second sub-section details the algorithm which has been implemented in the rg2tlsa module integrated within the rtl tool.

4.1 RT-Lotos Reachability Analysis

RT-Lotos is one of the temporal extensions of the standard Lotos formal description technique. A detailed description of RT-Lotos is provided in [8].

RT-Lotos provides three main temporal operators, namely: the *delay* - see construct $delay(d)$, the *latency* operator - see construct $latency(l)$ and the *time restrictor* operator - see construct $a\{t\}$.

As detailed in [7], the reachability analysis of RT-Lotos specifications is carried out as follows: An RT-Lotos specification is first translated into a DTA - Dynamic Timed Automaton; reachability analysis is then applied to the DTA

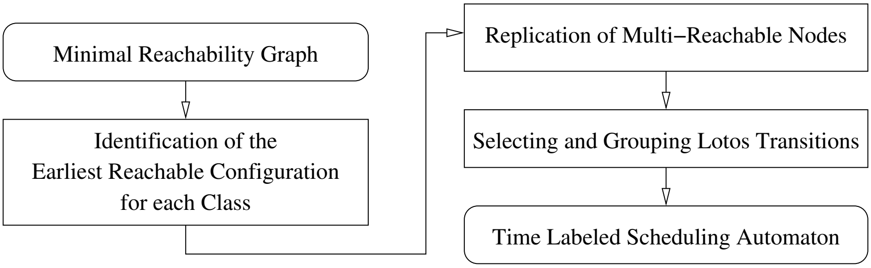

Minimal Reachability Graph
Identification of the Earliest Reachable Configuration for each Class
Replication of Multi–Reachable Nodes
Selecting and Grouping Lotos Transitions
Time Labeled Scheduling Automaton

Fig. 6. Overview of the algorithm

relying on a minimization algorithm [18] that generates a minimal reachability graph, which describes the global behavior of the RT-Lotos specification.

A global state or *configuration* of a timed system includes the control state of the timed automaton (the DTA), and the clocks values. Therefore, the number of configurations is not finite. A finite analysis of such a system requires a partition of the configuration space into a finite number of regions. Algorithms to perform reachability analysis, and to minimize timed transition systems were proposed in [2,18]. The second algorithm has been adapted to DTA and implemented in the rtl tool. The resulting graph is a *minimal reachability* graph where:

- a *node* (also called a *class*) defines both a control state, and a region represented as a convex polyhedron whose dimension is equal to the number of clocks of the control state; configurations belonging to the same region have the same reachability properties since they cannot be differentiated in terms of future actions that may occur; hence, a class corresponds to a finite representation of an infinite number of configurations (s, ν).
- an *arc* corresponds either to a Lotos action occurrence or to a time progression (arc labeled by t).

Due to the minimization algorithm, all configurations of a class are not necessarily reachable from the initial configuration; it can be proven that at least one configuration per class is actually reachable [18]. The minimization algorithm allows regions larger than those required from a strict reachability point of view to be considered, minimizing thereby, the number of regions within a graph, compared to other algorithms such as [6].

The formal definition of a minimal reachability graph and the way it may be derived from a DTA are presented in [7].

4.2 TLSA Synthesis Algorithm

The input of the TLSA synthesis algorithm is a minimal reachability graph. Three main steps are defined in the algorithm in order to produce a TLSA (see Fig. 6).

The algorithm will first process each transition of the minimal reachability graph (labeled either by t or a Lotos action). It starts from the initial reachable configuration (s_0, ν_0) in order to determine the earliest reachable configuration of each class of the minimal reachability graph. Then, it determines for each transition between two classes, the time progression between their earliest reachable configurations. Finally, it selects the transitions labeled by a `Lotos` action, defines their associated firing window and enabling condition, and groups them together in order to minimize the number of transitions, leading to the `TLSA`. These steps are recapitulated in the pseudo-code algorithm of Fig. 7.

```
Set the clocks of the earliest reachable configuration of initial class to 0.
For each transition of the minimal reachability graph do:
    Evaluate the earliest reachable configuration of the reached class.
    Set the temporal valuation of the transition with the time elapsed from the earliest reachable configuration of the starting class until the earliest reachable configuration of the reached class.
    Add this temporal valuation to the history of the transitions leaving the reached class.
End loop.
For each Lotos transition do:
    Set firing window $W$ with the temporal valuation of the transition plus the valuations of the temporal transitions fired into the current control state.
    Set enabling condition $K$ according to the history of the transition.
End loop
Extract Lotos transitions and rename nodes with the number of their control state.
Group transitions according to $W$ and $K$.
```

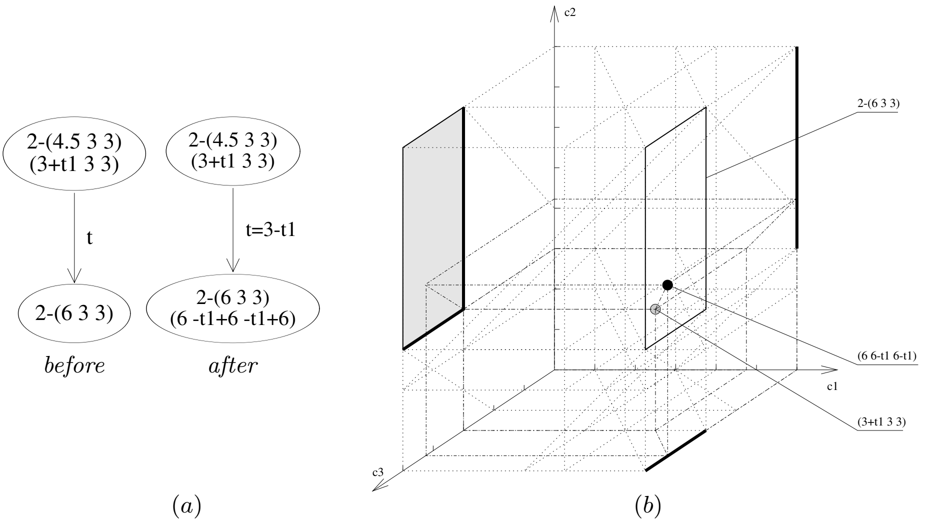
Fig. 7. Steps of the algorithm

4.2.1 Identification of the Earliest Reachable Configuration for Each Class

Let $DTA = (S, Nclock, E, s_0)$ be a dynamic timed automaton and $RG = LTS(DTA/\pi')$ its associated minimal reachability graph derived from an `RT-Lotos` specification (see [7] for details).

Definition 5 (Earliest reachable configuration within a class). *This is a recursive definition. Let A, B be classes of RG, S_0 the initial class of RG ($(s_0, \nu_0) \in S_0$), and transition $A \to B \in RG$.*

- *(s_0, ν_0) is the earliest reachable configuration of class S_0.*
- *Let $(s, \nu) \in A$ be the earliest reachable configuration of class A, where values of ν are expressed as linear expressions of constants (defined in D_0^∞) and timers (whose values are defined in $D_{0\perp}$):*
    - *if $A \xrightarrow{t} B$, then:*
      Let δ_m be the minimal time value such that $(s, \nu + \delta_m) \in B$; δ_m is expressed as linear expressions of constants and timers. Then, $(s, \nu + \delta_m)$ is reachable and it is, furthermore, the earliest reachable configuration of class B. δ_m is called the temporal valuation of the `t` transition. Note that, in this case, classes A and B are associated with the same DTA state, namely control state s.



(a) (b)

Fig. 8. Processing a `t` transition

- *if* $A \xrightarrow{a} B$, *then:*
  Let $(s, s', K, U, a, C, \theta)$ *a transition of the* DTA*, where* s' *is the control state of* B*. Let* δ_M *be the maximal time value such that* $(s, \nu + \delta_M) \in A$*;* δ_M *is expressed as linear expressions of constants and timers; note that if* A *is urgent then* $\delta_M = 0$*. Let* δ *be the time spent within all the classes associated with control state* s *before reaching the earliest configuration of class* A*. Let timer* t_s *be defined as:* $\delta \leqslant t_s \leqslant \delta + \delta_M$*. Then* $(s', \nu') \in B$ *is the earliest reachable configuration of class* B*, where* ${\nu'}^i = 0 \quad \forall i \in C$*, and* ${\nu'}^i = \nu^{\theta^{-1}(i)} + t_s - \delta \quad \forall i \in [1, Nclock(s')]$*,* $i \notin C$*.*

Intuitively the earliest reachable configuration of a class of RG may be defined as the first configuration of the class that the system may reach (starting from the initial configuration of the system) before any time progression within the class.

The following paragraphs detail the `TLSA` synthesis algorithm based on examples coming from the Minimal Reachability Graph depicted in figure 3a.

- *Processing* `t` *Transitions*

Let $a = (s, ({\nu_a}^1, \ldots, {\nu_a}^N))$ be the earliest reachable configuration of class A, with $N = Nclock(s)$; for example, as depicted in Fig. 8a, $a = (2, (3 + t1, 3, 3))$ and $A = 2 - (4.5\ 3\ 3)$.

A `t` transition from A to B means that there is a time progression to reach class B. Let $b = (s, ({\nu_b}^1, \ldots, {\nu_b}^N))$ be the earliest reachable configuration of class B. Configuration b can be defined by determining the *minimal value* δ_m, such that $\nu_b = ({\nu_a}^1 + \delta_m, \ldots, {\nu_a}^N + \delta_m)$ belongs to the region of class B (i.e.

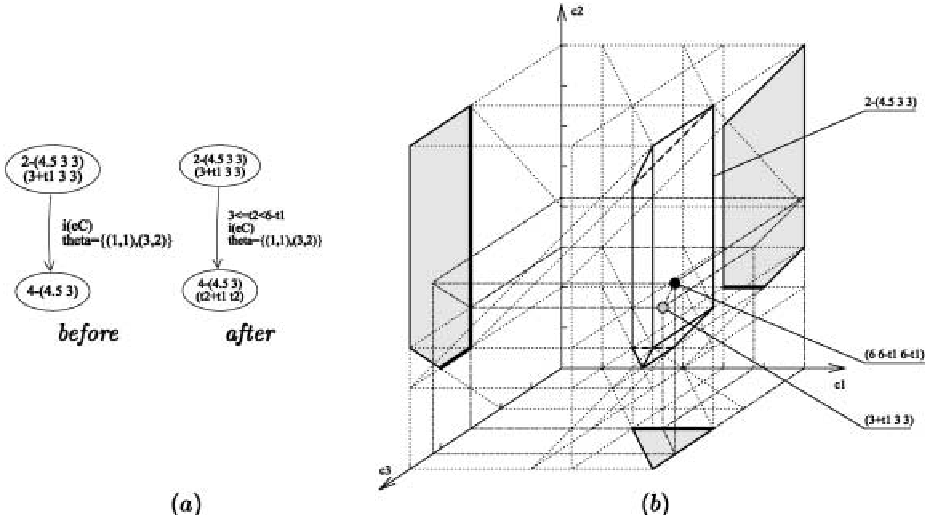


Fig. 9. Processing a Lotos transition

it satisfies the system of inequations bounding this region). For instance, the region of class $B = 2 - (6\ 3\ 3)$ is bounded by

$$\begin{cases} c_1 = 6 \\ 3 \leqslant c_2 < 8 \\ 3 \leqslant c_3 < 5 \\ -3 \leqslant c_2 - c_1 < 2 \\ -3 \leqslant c_3 - c_1 < -1 \\ -5 < c_3 - c_2 < 2 \end{cases}$$

$$\text{With} \quad \begin{cases} c_1 = {\nu_b}^1 = 3 + t1 + \delta_m \\ c_2 = {\nu_b}^2 = 3 + \delta_m \\ c_3 = {\nu_b}^3 = 3 + \delta_m \end{cases} \qquad \text{the solution is} \quad \delta_m = 3 - t1$$

Thus, the earliest reachable configuration of class B is $b = (2, (6, 6 - t1, 6 - t1)$, and the temporal valuation of transition $A \xrightarrow{t} B$ is equal to $3 - t1$.

- *Processing* Lotos *Transitions*

Let $a = (s, ({\nu_a}^1, \ldots, {\nu_a}^N)$ be the earliest reachable configuration of class A, with $N = Nclock(s)$; for example, in Fig. 9a, $a = (2, (3 + t1, 3, 3))$ and $A = 2 - (4.5\ 3\ 3)$. A Lotos transition from A to B means that time may progress within class A (unless A is an urgent class) before reaching class B by the occurrence of a Lotos action. Therefore, we are looking for the *maximal value* of δ_M, such that $({\nu_a}^1 + \delta_M, \ldots,\ {\nu_a}^N + \delta_M)$ belongs to the region of class A. In the example, this region is bounded by

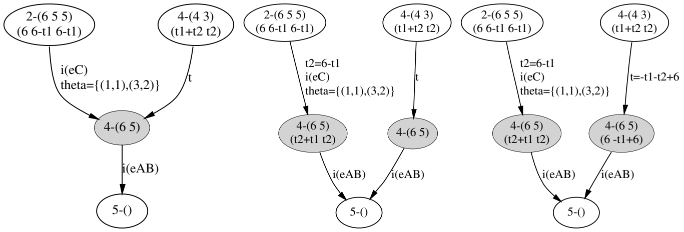


Fig. 10. Replicating a node

$$\begin{cases} 4 < c1 < 6 \\ 3 \leqslant c_2 < 8 \\ 3 \leqslant c_3 < 5 \\ -3 < c_2 - c_1 < 2 \\ -3 < c_3 - c_1 < -1 \\ -5 < c_3 - c_2 < 2 \end{cases}$$

With $\begin{cases} c_1 = 3 + t1 + \delta_M \\ c_2 = 3 + \delta_M \\ c_3 = 3 + \delta_M \end{cases}$ the solution is $\delta_M \leqq 3 - t1$

which means that δ_M is the maximal value less than $3 - t1$.

In our example, the system has to fire two t transitions between classes associated with control state 2 before reaching class A; then $\delta = (3 - t1) + (t1) = 3$, which characterizes the time spent in all the classes associated with control state 2 before reaching class A.

Therefore, the `Lotos` action can be fired in class A after a progression of time t_s comprised between δ and $\delta + \delta_M$.

In the example, the system can stay $3 \leqslant t2 < 3 - t1$ units of time in the classes associated with control state 2 before firing action `i(eC)`. This defines the *firing window* of this transition.

When the system leaves class A, the values of clocks are:

$$\begin{cases} c1 = {\nu_a}^1 + t_s - \delta \\ \vdots \\ cN = {\nu_a}^N + t_s - \delta \end{cases}$$

By applying functions C (clocks reset) and θ (copy of clocks values) (see the formal definition of the `DTA` in [7]) of this transition, we obtain the earliest reachable configuration of class B. In our example, the earliest reachable configuration of class $4 - (4.5\ 3)$ is: $(4, (t2 + t1, t2))$.

4.2.2 Replication of Multi-reachable Nodes

Some classes (such as $4 - (6\ 5)$ in Fig. 10) are reachable by two or more transitions. In these cases, the algorithm may determine distinct earliest reachable

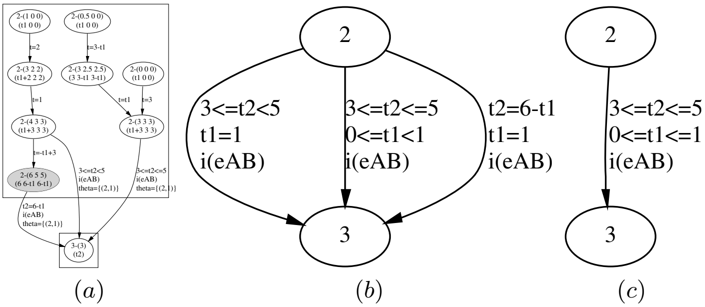


Fig. 11. Selecting and grouping Lotos transitions

configurations. As a consequence, the minimal reachability graph nodes associated with these classes must be replicated as many times as there are distinct earliest reachable configurations; outgoing transitions are replicated as well.

This case appears in particular in the presence of cycles: for each class concerned with a cycle, the algorithm finds a first earliest reachable configuration at the first iteration, then it finds a second one, corresponding to the following iterations, which may include timers affected during the cycle.

4.2.3 Selecting and Grouping Lotos Transitions

Finally, in order to produce the TLSA, the algorithm selects the Lotos transitions. It adds to these transitions their respective *firing window* and *enabling condition*. The firing window has been defined previously, and the enabling condition corresponds to the firing window's history of the Lotos transitions that lead to the class where the current transition is enabled.

The nodes of the TLSA are named according to the corresponding control state of the DTA, as it is illustrated in the transformation of Fig. 11a to Fig. 11b.

At this point, the algorithm has already produced a TLSA, although some transitions may still be redundant regarding firing windows and enabling conditions. The algorithm will then try to group together transitions having the same input and output control state and the same label. Space defined by the new *firing window* and *enabling condition* is the convex hull of polyhedra defined by the set of the *firing window* and *enabling condition* of each transition to be grouped[3]. For instance, the three transitions of Fig. 11b can be grouped together in the single transition on Fig. 11c.

5 Related Works

A certain number of work propose new types of temporal automata to express clearly the temporal relations between events.

[3] The vector of the current timer have only two coefficients equal to 1 and -1 in order to get a pair of inequalities - current timer superior to a constraint and current timer inferior to a constraint - which define W.

In [10], I. Kang and I.Lee employ the notion of time distance between transitions to propose a reachability algorithm. This seems to be close to how our timed conditions are expressed. However, these relations do not appear in their resulting graph, whereas the transition firing windows are explicit in the TLSA and they may themselves depend on the instants at which previous transitions were fired.

A TLSA may also be compared to event-recording-clock automata [4]. However, a TLSA does not record time of events, but instead, the time elapsed in control states. Furthermore, timed constraints compare values to linear expressions including values of other timers, and not just constants.

The problem of synthesizing scheduling automata is also the subject of several studies.

By constraining the specification of certain types of temporal constraints known as *implementable*, an algorithm to synthesize schedulers is proposed in [9]. Our approach does not make any assumption on the nature of the specified temporal constraints, the inconsistent behaviors being explicitly removed before the synthesis of the TLSA.

Another approach, proposed initially in [5] then extended in [12], aims to restrict the transition relation of temporal automata, based on a real time game, so that the resulting behaviors satisfy certain properties. Our approach is different, in the sense that the non-desirable inconsistent behaviors are removed at the level of the reachability graph. Thus, our approach is completely automatic, although it is necessary that the system knows in advance the set of all configuration classes.

Finally, in [1] the authors propose a method to build a scheduler satisfying a set of temporal constraints as well as a policy for scheduling. In the opposite, the only goal of our approach is to synthesize a temporal automata (TLSA) characterizing the set of the consistent behaviors of our high level specification expressed in RT-Lotos. However, the policy for scheduling to be applied is not described on the level of the TLSA, but on the level of the application that implement it (for example a browser for a multimedia document [15]).

6 Conclusion

We have introduced the TLSA model as an operational model intended to express globally time constraints that a system has to meet. Also an algorithm designed to synthesize a TLSA from a minimal reachability graph (derived from an RT-Lotos specification) has been described.

A TLSA expresses desirable consistent behaviors when it is synthesized from a minimal reachability graph from which all inconsistent paths (those leading to a deadlock configuration) have been removed. It describes in a simple way how to schedule events because there is one and only one running timer at a moment. Firing conditions of a transition are easy to compute: when the system enters a control state, the bounds of conditions W and K of all outgoing edges can immediately be evaluated by substitution of variables and simple arithmetic operations. So, as time progresses, the current timer has just to be compared to some constant values. Thus, the `TLSA` ensure the scheduling of consistent scenario applying the concept of controllability.

Works on the verification analysis and scheduling of multimedia documents based on the `TLSA` have been developed in [15]. In this sense, all the inconsistent behaviors of a document can be detected and removed and, further on, the scheduling of a consistent document can be accomplished by means of the concept of controllability, that is proposing a valid temporal interval inside which the scenario is always consistent. Fur this purpose, a `TLSA Player` was developed.

References

1. K. Altisen, G. Gossler, and J. Sifakis. A methodology for the construction of scheduled systems. In *FTRTFT 2000*, volume 1926 of *Lecture Notes in Computer Science*, pages 106–120, Pune, India, September 2000.
2. R. Alur, C. Courcoubetis, and N. Halbwachs. Minimization of timed transition systems. In *CONCUR'92*, volume 630 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.
3. R. Alur and D. Dill. The theory of timed automata. In *REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.
4. R. Alur, L. Fix, and T.A. Heizinger. Event-clock automata : A determinizable class of timed automata. In 6^{th} *Annual Conference on Computeraided Verification*, Lecture Notes in Computer Science 818, pages 1–13. Springer Verlag, 1994.
5. E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems*, pages 1–20, 1994.
6. B. Berthomieu and M. Diaz. Modeling and verification of time-dependent systems using time Petri nets. In *IEEE Transactions on Software Engineering*, volume 17. $n°3$, 1991.
7. J.P. Courtiat and R.C. R. de Oliveira. A reachability analysis of RT-Lotos specifications. In 8^{th} *International Conference on Formal Description Techniques*, Montreal, Canada, October 1995. Chapman&Hall.
8. J.P Courtiat, C.A.S. Santos, C. Lohr, and B. Outtaj. Experience with RT-Lotos, a temporal extension of the Lotos formal description technique. *Computer Communications*, 23:1104–1123, 2000.
9. H. Dierks. Synthesising controllers from real-time specifications. In *Tenth International Symposium on System Synthesis*, pages 126–133. IEEE Computer Society, September 1997.
10. I. Kang and I. Lee. An efficient state space generation for analysis of real-time systems. In *International Symposium on Software Testing and Analysis*, pages 4–13, 1996.
11. X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In *REX Workshop "Real-Time: Theory in Practice"*, volume 600 of *Lecture Notes in Computer Science*. Springer Verlag, 1991.
12. A. Pnueli, E. Asarin, O. Maler, and J. Sifakis. Controller synthesis for timed automata. In *System Structure and Control*. Elsevier Science, 1998.

13. P.N.M. Sampaio and J.P. Courtiat. A formal approach for the presentation of interactive multimedia documents. In *ACM Multimedia'2000*, pages 435–438, Los Angeles, USA, October 2000.
14. P.N.M. Sampaio and J.P. Courtiat. Scheduling and presenting interactive multimedia documents. In *International Conference on Multimedia and Exposition'2001*, pages 1227–1227, Tokyo, Japan, August 2001.
15. P.N.M Sampaio, C. Lohr, and J.P. Courtiat. An integrated environment for the presentation of consistent SMIL 2.0 documents. In *ACM Symposium on Document Engineering*, Atlanta, Georgia, USA, November 2001.
16. C.A.S. Santos, P.N.M. Sampaio, and J.P. Courtiat. Revisiting the concept of hypermedia document consistency. In *ACM Multimedia'99*, Orlando, USA, November 1999.
17. P. Sénac, M. Diaz, A. Leger, and P. de Saqui-Sannes. Modelling logical and temporal synchronization in hypermedia systems. In *IEEE Journal on Selected Areas in Communications*, volume 14(1), pages 84–103, January 1996.
18. M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems. In *CAV'93*, volume 697 of *Lecture Notes in Computer Science*. Springer Verlag, 1993.