ELSEVIER

Computer Communications 23 (2000) 1104–1123

computer communications

www.elsevier.com/locate/comcom

Experience with RT-LOTOS, a temporal extension of the LOTOS formal description technique

J.-P. Courtiat[a,*], C.A.S. Santos[a], C. Lohr[a], B. Outtaj[b]

[a]LAAS—CNRS, 7 avenue du Colonel Roche, 31077 Toulouse Cedex, France
[b]Ecole Mohamedia d'Ingénieurs, Rabat, Morocco

Abstract

This paper is devoted to the presentation of the RT-LOTOS formal description technique, which is a formalism suited for applications where concurrency, complex synchronization patterns, asynchronous interactions and timing constraints have to be dealt with together. The paper does not emphasize the details of the RT-LOTOS formal semantics, but intends to explain and illustrate its main features, as well as the main capabilities of its associated software tool `rtl`. Finally, this paper reports on different applications and case studies that have recently been developed at LAAS—CNRS, some of them in cooperation with the industry. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Formal specification and verification; LOTOS; RT-LOTOS; Timed automata; Temporal consistency

1. Introduction

Language of Temporal Ordering Specifications (LOTOS) is a formal description technique standardized at ISO [1] based on CCS [2] (extended by a multiway synchronization mechanism inherited from CSP [3]). Real-Time LOTOS (RT-LOTOS) [4] is one of the temporal extensions of the standard LOTOS formal description technique, which, by means of few additional operators (delay, time restriction and latency), permits to express general time-constrained behaviors.

For historical reasons, RT-LOTOS is similar to Enhanced Timed LOTOS (ET-LOTOS) [5], the successor of Timed LOTOS [6], both of which have been developed at the University of Liège, Belgium. The main difference between RT-LOTOS and ET-LOTOS will be explained with simple examples, and the interest of the latency operator of RT-LOTOS will be demonstrated.

However, the main issue considered in this paper is not a debate between RT-LOTOS and ET-LOTOS, but our experience with the application of RT-LOTOS and its associated software tool `rtl` in different domains (protocol engineering, fault-tolerant control systems and hypermedia authoring). We will briefly present the purpose of these applications, the specification and validation methodologies we developed, the results we achieved, as well as the return of our experience [7].

The paper is organized as follows: Section 2 deals with a non-formal introduction to RT-LOTOS. Section 3 gives some details about the `rtl` tool environment, in particular its simulation and verification capabilities. Section 4 reports on some case studies developed with RT-LOTOS and `rtl`. Section 5 details some lessons learned with these case studies. Finally, Section 6 draws some final conclusions.

2. A non-formal introduction to RT-LOTOS

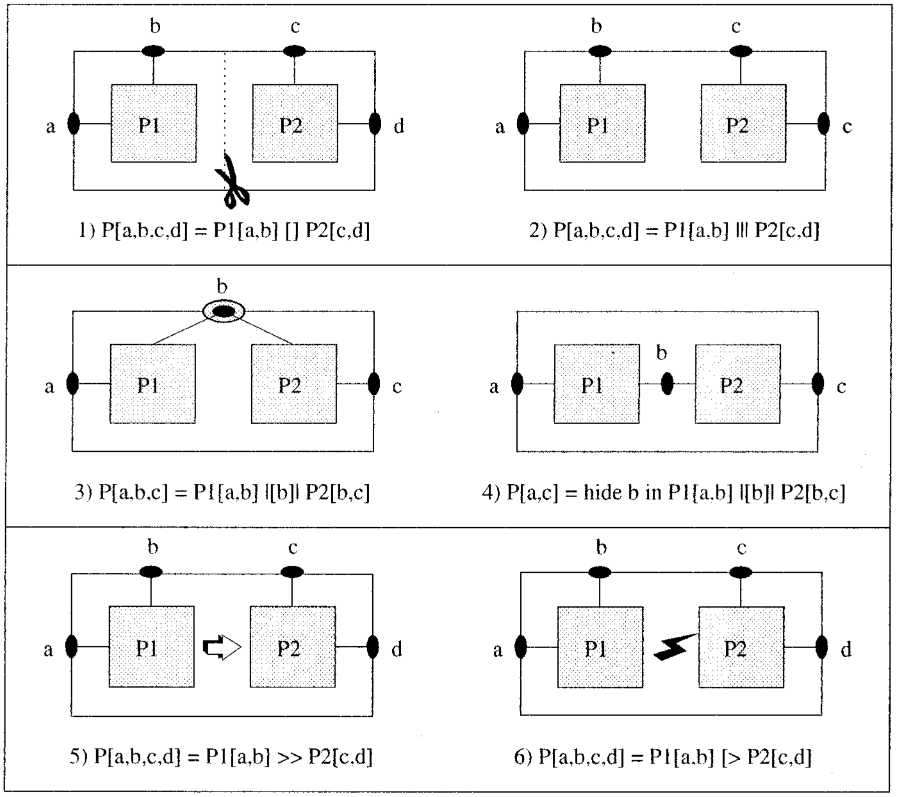
This section provides background information on LOTOS and RT-LOTOS. The first paragraph introduces the standard LOTOS formal description technique. The second sub-section presents intuitively the temporal operators defined for RT-LOTOS and the third introduces briefly how data types are expressed in RT-LOTOS.

2.1. The LOTOS formal description technique

The basic idea supporting LOTOS is that systems can be specified by expressing the relations among the interactions that constitute their externally observable behavior. In LOTOS, a system is seen as a process, possibly consisting of several sub-processes. Likewise a sub-process is a process in itself, and a LOTOS specification describes a system via a hierarchy of process definitions. A process is an entity capable of performing internal, unobservable

* Corresponding author.
E-mail addresses: courtiat@laas.fr (J.-P. Courtiat); saibel@laas.fr (C.A.S. Santos); lohr@laas.fr (C. Lohr); benaceur@laas.fr (B. Outtaj).

PII: S0140-3664(99)00240-6



1) P[a,b,c,d] = P1[a,b] [] P2[c,d]

2) P[a,b,c,d] = P1[a,b] ||| P2[c,d]

3) P[a,b,c] = P1[a,b] |[b]| P2[b,c]

4) P[a,c] = hide b in P1[a,b] |[b]| P2[b,c]

5) P[a,b,c,d] = P1[a,b] >> P2[c,d]

6) P[a,b,c,d] = P1[a,b] [> P2[c,d]

Fig. 1. Illustration of the LOTOS operators.

actions, and of interacting with other processes which form its environment.

In that sense, LOTOS implements a *black box* paradigm used to develop high level, concise and abstract specifications of complex systems. At some abstraction level, it is possible to express the interactions of a process with its environment without having to describe the internal structure (or implementation) of that process. Process definitions are expressed by the specification of behavior expressions that are constructed by means of a restricted set of powerful operators making it possible to express behaviors as complex as desired. Processes may in general be defined recursively, and the multi-way rendezvous mechanism represents the basic communication facility between processes. Among the operators, action prefixing, choice, parallel composition and hiding play a fundamental role.

Let us call, as usual, Basic LOTOS the subset of LOTOS where the processes interact with each other by pure synchronization without exchanging any value. Fig. 1 provides an intuitive illustration of the main Basic LOTOS operators:

1. *Choice*: either P1 [a,b] or P2 [c,d] depending on the environment;
2. *Parallel composition without synchronization*: P1 [a,b] is independent from P2 [c,d]
3. *Parallel composition with synchronization on action* b: P1 [a,b] is independent from P2 [b,c] for all actions except for action b on which both processes should synchronize;
4. *Parallel composition with synchronization on action* b *and hiding of action* b: P1 [a,b] is independent from P2 [b,c] for all actions except for action b on which both processes should synchronize, action b being furthermore not available for any potential synchronization with process belonging to the environment of process P;
5. *Sequential composition*: P1 [a,b] is followed, when P1 terminated, by P2 [c,d];
6. *Disrupt*: P1 [a,b] may be interrupted at any time before its termination by P2 [c,d].

The reader should consult Refs. [8,9] for detailed tutorials of LOTOS, as well as Refs. [1,8] for the formal definition of LOTOS.

2.2. The RT-LOTOS formal description technique

Using LOTOS, only the *qualitative* ordering of events (i.e. occurrence of actions) can be expressed, without any mention of the quantitative time at which these events may occur. This limitation together with the need to apply formal methods to complex real-time distributed systems has motivated several researches to extend LOTOS. The initial inspiration of RT-LOTOS [4] came from an in-depth analysis of two other time extensions of LOTOS, namely Timed LOTOS [6] and LOTOS-T [10]. Several assumptions and decisions made during the design of RT-LOTOS have then been taken to preserve as much convergence as possible with the successor of Timed LOTOS, namely ET-LOTOS [5]. As a result, RT-LOTOS, and ET-LOTOS feature several points in common:

1. the non-urgency assumption of observable actions (one considers that the occurrence of an observable action cannot be forced, since its depends on the willingness of the environment);
2. the capacity offered by the language to restrict the time during which an observable action is offered to its environment.

The latter point has been generalized in RT-LOTOS to arrive at the temporal violation concept and its associated recovery mechanism [4]. The main difference between RT-LOTOS and ET-LOTOS is, however, related to the way time latency (i.e. non-deterministic delay) may be expressed. The solution developed for RT-LOTOS, based on the introduction of a new latency operator, is more general and powerful than the approach proposed for ET-LOTOS. A detailed comparison between RT-LOTOS and ET-LOTOS is provided in Section 3.4.

Let us analyze the following time-constrained behaviors for understanding the intuition behind the basic RT-LOTOS temporal operators. These behaviors are characterized by means of time lines; let $\theta[\mathtt{a}]$ be the actual time, on the time line, at which action `a` is enabled from a pure LOTOS point of view (i.e. without considering timing constraints). Furthermore, let us call $window[\mathtt{a}]$ the temporal window during which action `a` may occur from a RT-LOTOS point of view (i.e. taking into account all the time constraints). A complete formal definition of the operators can be found in Refs. [4,11,12] (Fig. 2).

1. *Delay*: in process `P1`, the possible occurrence of action `a` is delayed by time d. We have, therefore, $window[a] = [\theta[a] + d, \infty[$, since `a` is an observable action which then is not urgent (the occurrence of `a` depends on its environment).
2. *Delay and hiding*: in process `P2`, action `a` is hidden and then behaves as an internal action. The action no longer depends on its environment and it is then urgent. As a result, $window[\mathtt{i}(a)] = [\theta[a] + d, \theta[a] + d]$ where `i(a)` denotes the internal action resulting from the hiding of action `a`.
3. *Delay and time restriction*: in process `P3`, an additional time constraint is associated with the occurrence of `a`, since `a`, once enabled, is only offered by `P3` during a limited period of time t. We have, therefore, $window[a] = [\theta[a] + d, \theta[a] + d + t]$. If, for any reason, `a` cannot occur during this time interval, a temporal violation situation will result leading to transforming `P3` into stop, unless a specific temporal violation recovery mechanism has been specified (not further discussed in the paper).
4. *Delay, time restriction and hiding*: in process `P4`, action `a` is hidden leading to $window[\mathtt{i}(a)] = [\theta[a]+, d\theta[a] + d]$ as in the situation described for process `P2`.
5. *Delay, time restriction and synchronization*: process `P5` represents the synchronization of two processes. If action `a` occurs, that will be at some time within the intersection of the time intervals characterizing the constraints associated with each process composing process `P5`. Then $window[a] = [\theta[x] + d1, \theta[x] + d1 + t1] \cap [\theta[y] + d2, \theta[y] + d2 + t2]$. In case of disjoint time intervals, `a` cannot occur and a temporal violation situation arises for which the environment is not responsible. As previously mentioned, even if the time intervals are not disjoint a temporal violation situation may still arise, if the environment cannot accept the occurrence of an action.
6. *Delay, time restriction, synchronization and hiding*: in process `P6`, action `a` has been hidden and consequently $window[\mathtt{i}(a)] = \max(\theta[x] + d1, \theta[y] + d2)$, assuming that the intersection of the time intervals is not empty.
7. *Latency*: in process `P7`, action `a` is first delayed by time d, then possibly offered to the environment during time l and then, if action `a` has not occurred, it is continuously offered to the environment. As for process `P1`, we have $window[a] = [\theta[a] + d, \infty[$, since it is not possible to distinguish the reason why an observable action did not occur during the latency interval $[d, d + l[$ (it may be due either to the process itself which did not offer the action, or to the environment which did not accept it). As a consequence, without hiding, process `P7` cannot be distinguished from process `P1`.
8. *Latency and hiding*: hiding action `a` in process `P7` leads to process `P8`. As there is time non-determinism in the offering of action `a` within the latency interval $[d, d + l[$, and as this time non-determinism does not depend exclusively on the environment, $window[\mathtt{i}(a)] = \theta[a] + d, \theta[a] + d + l]$.
9. *Delay, time restriction and latency*: process `P9` features the combination of the delay and latency operators together with an offering of action `a` restricted to a period t. Assuming $l \leq t$, we have $window[a] = [\theta[a] + d, \theta[a] + d + t]$ which is identical to the situation described for process `P3`.
10. *Delay, time restriction, latency and hiding*: hiding action `a` in process `P9` leads to process `P10` for which we have $window[\mathtt{i}(a)] = [\theta[a] + d, \theta[a] + d + l]$, exactly as for process `P8`.
11. *Delay, time restriction and latency*: with assumption $l > t$, process `P11` is still similar to process `P9` leading to $window[a] = [\theta[a] + d, \theta[a] + d + t]$.
12. *Delay, time restriction, latency and hiding*: with assumption $l > t$, hiding action `a` in process `P11` leads to process `P12` where $window[\mathtt{i}(a)] = [\theta[a] + d, \theta[a] + d + t]$. As the latency is greater than the time restriction,

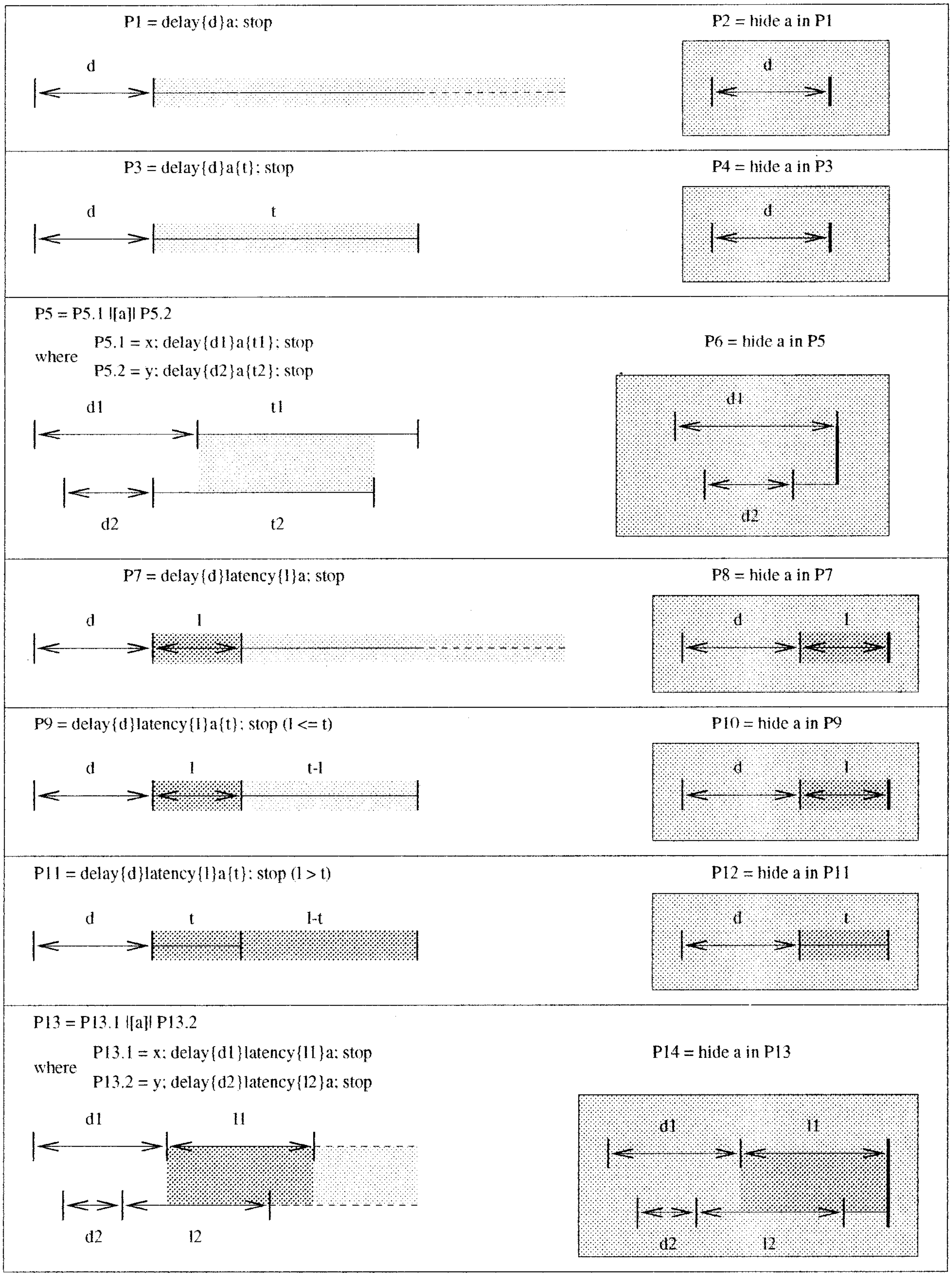


Fig. 2. Illustration of some RT-LOTOS operators.

action `i(a)` may possibly not occur, i.e. it is not urgent at the right of the time interval, as it was the case in situation 10.

13. *Delay*, *latency and synchronization*: as for process `P5`, if action `a` may occur in process `P13`, it will occur some time within the intersection of the time intervals characterizing the time constraints associated with each process composing process `P13`. Then $window[a] = [\theta[x] + d1, \infty] \cap [\theta[y] + d2, \infty]$. As previously, in case of disjoint time intervals, a temporal violation situation will occur.
14. *Delay*, *latency and synchronization*: in process `P14`, action `a` has been hidden and the reader will check that $window[i(a)] = [\max(\theta[x] + d1, \theta[y] + d2, \max(\theta[x] + d1 + l1, \theta[y] + d2 + l2)]$.

In summary, RT-LOTOS provides three main temporal operators:

1. The *delay operator* aiming to delay in a deterministic way the occurrence of observable and internal actions—see construct `delay(d)`.
2. The *latency operator* aiming to express non-deterministic time latency—see construct `latency(l)` (note that delay and latency may be expressed together by a single syntactic construct `delay(dmin, dmax)` meaning `delay(dmin) latency(dmax–dmin)`).
3. The *time restriction operator* aiming to limit the time during which an observable action a is offered to its environment—see construct `a{t}`.

RT-LOTOS provides other operators that are not detailed here since they are not fully supported by the `rtl` software tool (see Ref. [4] for details).

2.3. Expressing data types in RT-LOTOS

In standard LOTOS, the description of data type signatures is completed by the definition of equations, expressed in the Act-One formalism, for providing the type of semantics. For various reasons, related mainly to the non-obvious industrial applicability of Act-One, only the data type signature is expressed in RT-LOTOS, the meaning of data type operations being provided by C++ or Java classes defined in a user library accessible from the `rtl` software tool (itself written in C++).

3. The `rtl` tool environment

Selecting some system properties and proving them aim to convince a designer that the model of the system under construction does correspond to the intended behavior. Wherever one can obtain a finite model representing the global behavior of a specification (in general, some kind of automaton corresponding to a labeled transition system), a wide range of analysis techniques are available:

1. If the automaton is small enough, it can be directly observed and the designer can be assured of the correct behavior of his model.
2. If the automaton is not so small (but also not too large), it can be reduced using classical techniques (for instance, by hiding actions considered as less important than the others, and by deriving a reduced automaton—a quotient automaton with respect to some equivalence relation). The resulting automaton will usually be smaller than the original one, and hopefully small enough for being directly observed.
3. Model checking is another technique consisting in proving temporal logic formulas on a (finite) automaton. The main advantage of model checking is that it is a very powerful method that has been continuously improved over the last years. The drawback, however, is that correctly expressing the desired properties is often much more difficult than initially expected. Quantitative temporal logics like TCTL [13] have been introduced to express properties depending on explicit time constraints.
4. Another (more pragmatic) technique is related to the so-called observer (or tester) approach [14]. Basically, observers are modules synchronizing themselves with the specification on some internal actions, and checking on-line whether some particular condition characterizing the violation of the property arises; in such a case, the observer offers a specific `error` action. Proving that the property checked by the observer is valid consists, therefore, in showing that the `error` action is not reachable. The advantage of the technique lies in that it can easily be implemented, since it results in reachability analysis. The main drawback is that the method is less powerful than model checking (some properties cannot be expressed using observers), but experience has shown however that it is more a theoretical than a practical limitation. Finally, one may point out that the property is expressed with the same formalism as the one used for specifying the system; this may be seen as an advantage (only one formalism needs to be known) or a drawback (it is better to separate the concerns).

Within this general framework, this section addresses the validation techniques that have been implemented with the `rtl` software tool. A simple example, the communication medium example, will be used throughout the section to illustrate some of the capabilities of the `rtl` tool, and the kind of information it provides. Real-life applications and case studies will be discussed in the next section.

`rtl` Validation techniques may roughly be divided into two main categories:

1. Simulation techniques aiming to observe traces of a specification's global behavior; their purpose is to understand better the global behavior of a specification and possibly to gain a certain level of confidence on the validity of some properties (i.e. properties that have not been violated during a simulation run).
2. Verification techniques aiming to formally prove some

RT-LOTOS specification

```
specification MEDIUM : noexit :=
 (...)
behaviour
 hide iu_s, iu_d in
 let period : nat = 30000 in

 stream_sender[iu_s](0,period)
 |[iu_s]|
 medium[iu_s,iu_d](14000,20000)

where
  process stream_sender [iu_s] (n : nat, period : nat) : noexit :=
   iu_s0!n; delay(period) stream_sender[iu_s] (n+1, period)
  endproc
 process medium [n_in,m_out] (dmin, dmax : nat) : noexit :=
  m_in?x:nat; delay(dmin,dmax)m_out!x;
  medium[m_in,m_out](dmin,dmax)
 endproc
endspec
```

Simulation trace

```
Time Action State
0 i(iu_s<0>) 1
0 14000 2
14000 5811 3
19811 w-i(iu_d<0>) 4
119811 10189 5
30000 i(iu_s<1>) 6
30000 14000 7
44000 2894 8
46894 w-i(iu_d<1>) 9
46894 13106 10
60000 i(iu_s<2>) 11
60000 14000 12
...
```

Fig. 3. rtl Simulation capabilities: simulation trace.

properties; here the purpose is to analyze a complete (finite) model of the (RT-)LOTOS specification; these techniques are no longer feasible if this model cannot be computed (either because it is infinite or just because it is too large with respect to the size of the available RAM memory).

The trade-off between verification and simulation is very easy to understand. Verification is the ultimate goal, but in practice one is often faced with the unfortunate and classical problem of state space explosion. Depending on the available RAM memory and on the average size of a state representation in memory, one can easily estimate, for some particular specification, the maximal number of states that can potentially be produced before entering the swap zone. As discussed later, both approaches have proven useful and complementary for the different case studies developed with RT-LOTOS.

3.1. rtl Simulation capabilities

Simulation is very useful for debugging complex specifications. This is particularly true for a specification technique like LOTOS where errors in the specification of process synchronization frequently lead to undesirable deadlock situations, due to the multiway rendezvous.

Let us consider the simple RT-LOTOS specification of Fig. 3 for illustrating the simulation capabilities of rtl. Specification MEDIUM describes a simple situation where some periodic stream is sent (process stream_sender) through a one-slot medium (process medium) with a transmission delay belonging to interval [14 ms, 20 ms]. With each information unit is associated an integer sequence number. The stream information units, assumed to be submitted by the environment on action iu_s, are delivered by the medium on action iu_d. Due to the one-slot assumption, the non-deterministic transmission delay is chosen to be less than the period (30 ms).

The rtl simulation facility consists essentially in performing simulation runs, that may then be processed in order to provide the user with different granularity levels for observing the behavior of the specification.

1. The first granularity level (Fig. 3) consists in looking at a particular event trace; each element of the trace corresponds to an event (action occurrence) characterized by:
   - the time reference at which the event occurs;
   - the identification of the event which may be either an action occurrence or a time progression (note that the action name may be prefixed by w- to indicate the occurrence of a weak or lazy action—i.e. a non-urgent internal action—see the RT-LOTOS operational semantics defined in Refs. [4,11] for details);
   - the reference of the global state reached after the occurrence of the event, from which the user can get an in-depth knowledge of the specification global state.
2. The second granularity level consists in observing action occurrences on some specific actions selected interactively by the user; action occurrences are featured (possibly with the data values exchanged on their associated gates) on time lines (Fig. 4).
3. The third granularity level consists in displaying some variables (or functions of these variables) versus the time. This granularity level corresponds to simulation graphs (Fig. 5).

An interesting feature of rtl consists in also implementing

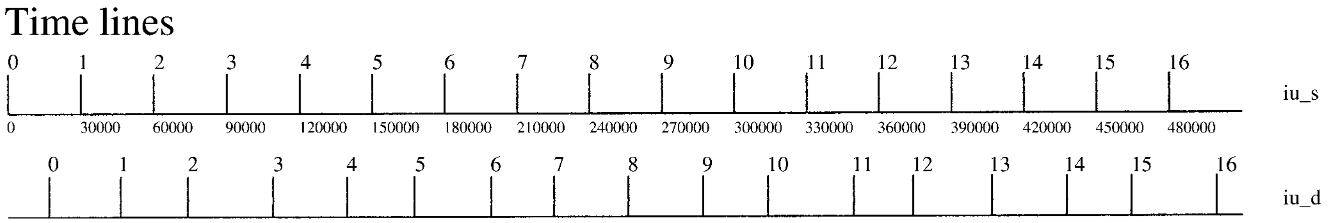


Fig. 4. rtl Simulation capabilities: simulation time lines.

the observer technique within a simulation framework. In this case, the error action has to be specified as a goal for the simulation kernel; if, in some configuration, action error is enabled, it will necessarily occur, showing then that the property implemented by the observer is not valid (a simulation run that violates the property has then been identified).

3.2. rtl Verification capabilities

Depending on the formalisms, the labeled transition system may either be directly derived from the rules of the operational semantics, or from an intermediate model into which the (high-level) formalism has been translated.

With the assumption of a dense time domain, the time dimension leads to an additional difficulty, since the underlying model becomes an infinitely branching transition system [15]. Different finite representations of such transition systems have been proposed in the area of time Petri nets [16], timed automata [17], and timed communication automata [18].

Starting from this background, a reachability analysis of RT-LOTOS specifications has been developed [12]. The method consists in translating a RT-LOTOS specification into a timed automaton model, on which reachability analysis is performed. The specific method implemented in rtl presents two advantages:

- it permits to minimize the number of clocks in each control state of the timed automaton, thanks to the definition of the DTA (Dynamic Timed Automata) model;
- reachability analysis may be performed on-the-fly when generating the DTA model from the RT-LOTOS specification.

Both advantages are important from a practical point of view, since the complexity of verification algorithms developed for timed automata depends directly on the number of clocks [19].

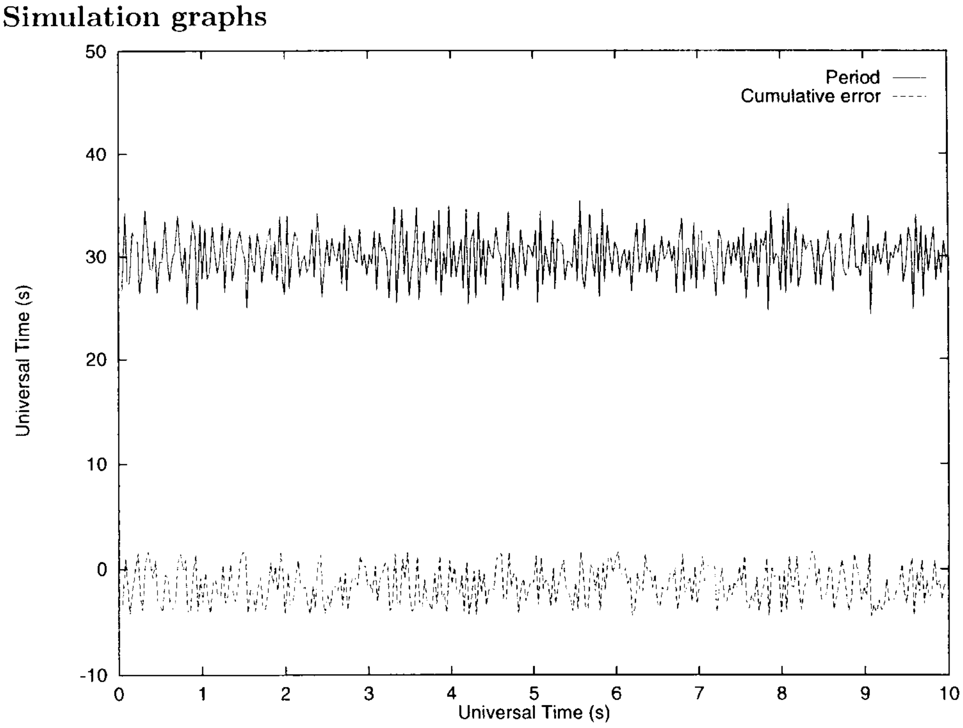
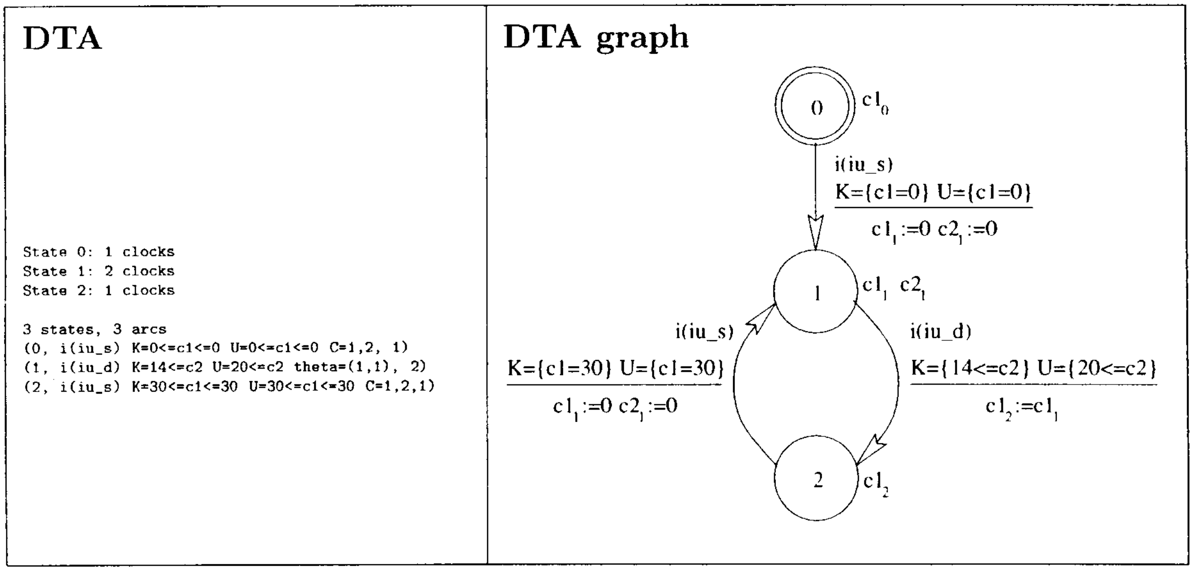


Fig. 5. rtl Simulation capabilities: simulation graphs.



DTA

```
State 0: 1 clocks
State 1: 2 clocks
State 2: 1 clocks

3 states, 3 arcs
(0, i(iu_s) K=0<=c1<=0 U=0<=c1<=0 C=1,2, 1)
(1, i(iu_d) K=14<=c2 U=20<=c2 theta=(1,1), 2)
(2, i(iu_s) K=30<=c1<=30 U=30<=c1<=30 C=1,2,1)
```

DTA graph

Fig. 6. DTA graph.

3.2.1. Translation of a RT-LOTOS specification into a DTA model

Timed automata [20] have been proposed to model finite-state real-time systems. Each 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 has an associated enabling condition expressed as a constraint on the values of the clocks.

The DTA model [12] features three main extensions: it is a labeled timed automaton model making a distinction between *urgent* and *non-urgent* actions and allowing a variable number of clocks to be associated with the different control states of the model.

The DTA model has been slightly modified in the latest release of `rtl` enhancing the way non-urgent internal actions are treated, with the consequent avoidance of the use of explicit weak internal actions in a DTA, as it was the case in the original formal definition of the DTA [12]. The new definition of the DTA model is presented in Appendix A.

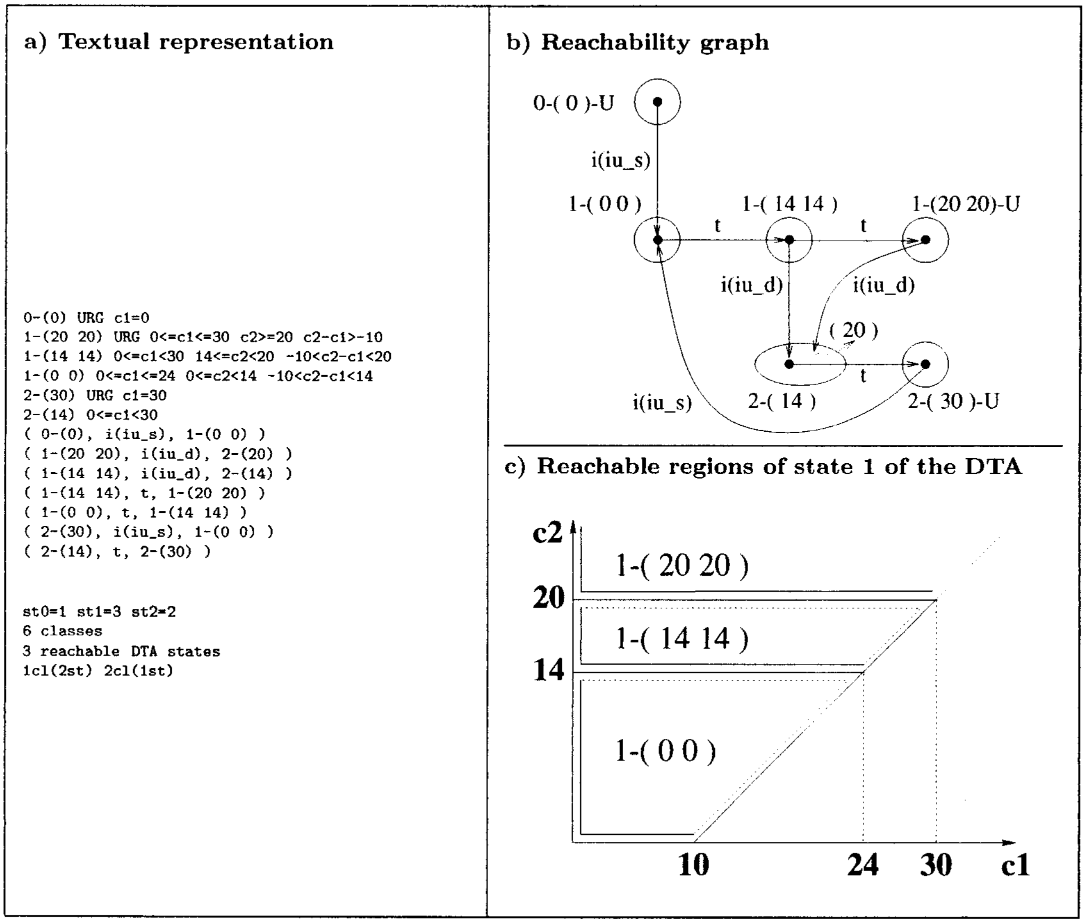
Fig. 6 illustrates the construction of the DTA for our toy RT-LOTOS specification. To ensure that it remains finite, the RT-LOTOS specification has been slightly modified by removing the sequence numbers associated with the information units. Note that the enabling time domain K and the urgent time domain U play two distinct roles (see the formal definitions in Appendix A). For instance, action `i(iu_d)`, resulting from the hiding of action `iu_d`, is an internal action which is urgent within its U domain i.e., for $c2 \geq 20$; it is however enabled but not urgent within time domain $K-U$, i.e. for $c2 \in [14,20[$.

3.2.2. Reachability analysis

A global state or configuration of a timed system consists of the control state of the timed automata (the DTA here) and the values of the clocks. There is, therefore, an infinite number of configurations. A finite analysis of such a system requires to partition configuration space into a finite number of regions. Algorithms for performing reachability analysis and minimization of timed transition systems simultaneously were proposed in Refs. [16,17,19]. The second algorithm has been adapted for Dynamic Timed Automata and implemented in the `rtl` tool. The resulting graph is a *minimal reachable graph* where:

- a *node* (also called class) defines both a control state and a region, represented as a convex polyhedron whose dimension equals the number of clocks of the control state; configurations belonging to a same region have the same reachability properties, since they are indistinguishable in terms of the future actions that may occur; hence a class corresponds to a finite representation of a infinite number of configurations (s, ν).
- an *arc* corresponds either to an action occurrence or a time progression.

Applying this approach to our RT-LOTOS specification leads to the reachability graph depicted in Fig. 7b. As expected, it is very simple, and contains only 6 classes associated with the 3 reachable (control) states of the DTA. Note that different classes may be constructed from the same DTA state (3 classes are associated with control state 1). In general, unreachable DTA states may exist, meaning that there is no class associated with them in the final reachability graph. If time progresses within a class, this class contains an infinite number of elements (s, ν), because the time domain is dense. If the class is urgent (i.e. time cannot progress within the class), then the class



a) Textual representation

```
0-(0) URG c1=0
1-(20 20) URG 0<=c1<=30 c2>=20 c2-c1>-10
1-(14 14) 0<=c1<30 14<=c2<20 -10<c2-c1<20
1-(0 0) 0<=c1<=24 0<=c2<14 -10<c2-c1<14
2-(30) URG c1=30
2-(14) 0<=c1<30
( 0-(0), i(iu_s), 1-(0 0) )
( 1-(20 20), i(iu_d), 2-(20) )
( 1-(14 14), i(iu_d), 2-(14) )
( 1-(14 14), t, 1-(20 20) )
( 1-(0 0), t, 1-(14 14) )
( 2-(30), i(iu_s), 1-(0 0) )
( 2-(14), t, 2-(30) )


st0=1 st1=3 st2=2
6 classes
3 reachable DTA states
1cl(2st) 2cl(1st)
```

b) Reachability graph

c) Reachable regions of state 1 of the DTA

Fig. 7. Reachability graph.

has only a finite number of elements related to the number of arcs entering this class.

The textual part of Fig. 7a shows the file generated by `rtl`. In the other parts of the figure, the same information is presented as a reachability graph and the details of the three clock regions associated with DTA state 1 are displayed. On the reachability graph, the following notation conventions have been considered:

- `x-(n m)` identifies a class associated with state `x` of the DTA, with `(n m)` characterizing the values of the clocks of one reachable configuration belonging to the class; this reachable configuration is called the *representative configuration* of the class.
- `x-(n m)-U` represents a urgent class.
- a transition labeled by `t` represents some time progression; the actual value of time depends on which configuration the transition applies, and it cannot consequently be explicit globally.

As a consequence of the minimization algorithm implemented in `rtl` (adapted from Ref. [19]), configurations of a class are not necessarily all reachable from the initial configuration; it can be proven, that at least one configuration per class is actually reachable. Looking at class `1-(0 0)`, only the configurations belonging to the first bisector of the region are actually reachable. The minimization algorithm proposed in Ref. [19] permits to consider regions larger than the ones required from a strict reachability point of view, thereby minimizing the number of regions within a graph when compared to others algorithms like Refs. [16,17] (Fig. 7c).

3.2.3. Reachability analysis and observer approach

Relying on reachability analysis, the observer approach is used in `rtl` to analyze specific properties. Once the observer module has been specified and composed with the specification—the observer should obviously not interfere with the nominal behavior of the specification—, the property is *true* if the associated `error` action is not reachable from the initial state of the composed specification. The same technique may also be used within a simulation framework, once the `error` action is specified as a goal for the simulation kernel; if, in some state, `error` is enabled, then it will necessarily be fired, characterizing a violation of the property. In case `error` does not appear in the simulation trace, no decision about the property validity can be made.

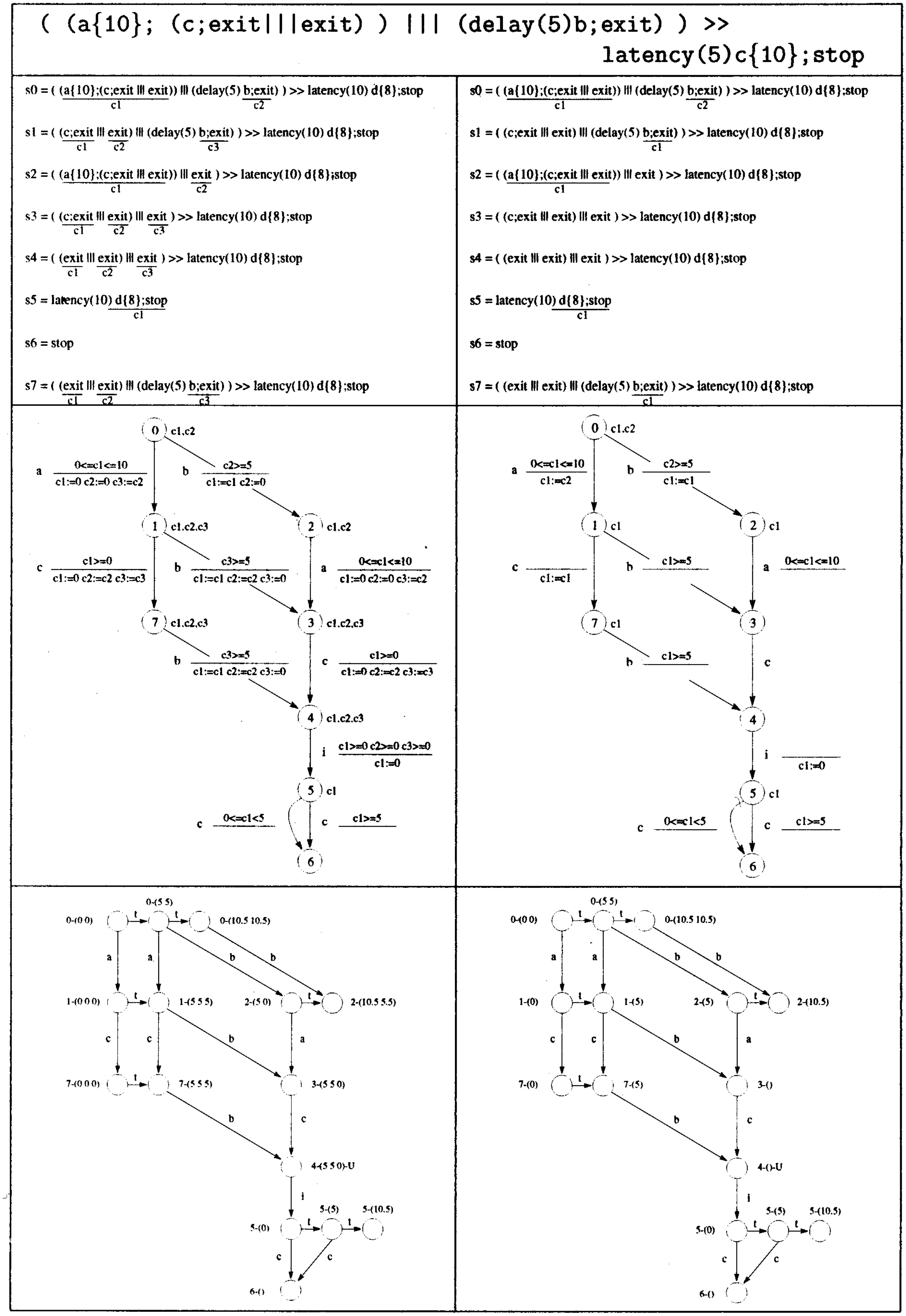

((a{10}; (c;exit|||exit)) ||| (delay(5)b;exit)) >>
latency(5)c{10};stop
s0 = ((a{10};(c;exit ||| exit)) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1 c2
s1 = ((c;exit ||| exit) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1 c2 c3
s2 = ((a{10};(c;exit ||| exit)) ||| exit) >> latency(10) d{8};stop c1 c2
s3 = ((c;exit ||| exit) ||| exit) >> latency(10) d{8};stop c1 c2 c3
s4 = ((exit ||| exit) ||| exit) >> latency(10) d{8};stop c1 c2 c3
s5 = latency(10) d{8};stop c1
s6 = stop
s7 = ((exit ||| exit) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1 c2 c3
s0 = ((a{10};(c;exit ||| exit)) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1 c2
s1 = ((c;exit ||| exit) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1
s2 = ((a{10};(c;exit ||| exit)) ||| exit) >> latency(10) d{8};stop c1
s3 = ((c;exit ||| exit) ||| exit) >> latency(10) d{8};stop
s4 = ((exit ||| exit) ||| exit) >> latency(10) d{8};stop
s5 = latency(10) d{8};stop c1
s6 = stop
s7 = ((exit ||| exit) ||| (delay(5) b;exit)) >> latency(10) d{8};stop c1
0 c1.c2
a 0<=c1<=10 / c1:=0 c2:=0 c3:=c2
b c2>=5 / c1:=c1 c2:=0
1 c1.c2.c3
2 c1.c2
c c1>=0 / c1:=0 c2:=c2 c3:=c3
b c3>=5 / c1:=c1 c2:=c2 c3:=0
a 0<=c1<=10 / c1:=0 c2:=0 c3:=c2
7 c1.c2.c3
3 c1.c2.c3
b c3>=5 / c1:=c1 c2:=c2 c3:=0
c c1>=0 / c1:=0 c2:=c2 c3:=c3
4 c1.c2.c3
i c1>=0 c2>=0 c3>=0 / c1:=0
5 c1
c 0<=c1<5
c c1>=5
6
0 c1.c2
a 0<=c1<=10 / c1:=c2
b c2>=5 / c1:=c1
1 c1
2 c1
c c1:=c1
b c1>=5
a 0<=c1<=10
7 c1
3
b c1>=5
c
4
i c1:=0
5 c1
c 0<=c1<5
c c1>=5
6
0-(0 0)
0-(5 5)
0-(10.5 10.5)
t
a
b
1-(0 0 0)
1-(5 5 5)
2-(5 0)
2-(10.5 5.5)
c
7-(0 0 0)
7-(5 5 5)
3-(5 5 0)
4-(5 5 0)-U
i
5-(0)
5-(5)
5-(10.5)
6-()
0-(0 0)
0-(5 5)
0-(10.5 10.5)
1-(0)
1-(5)
2-(5)
2-(10.5)
7-(0)
7-(5)
3-()
4-()-U
5-(0)
5-(5)
5-(10.5)
6-()

Fig. 8. Illustration of the DTA generation.

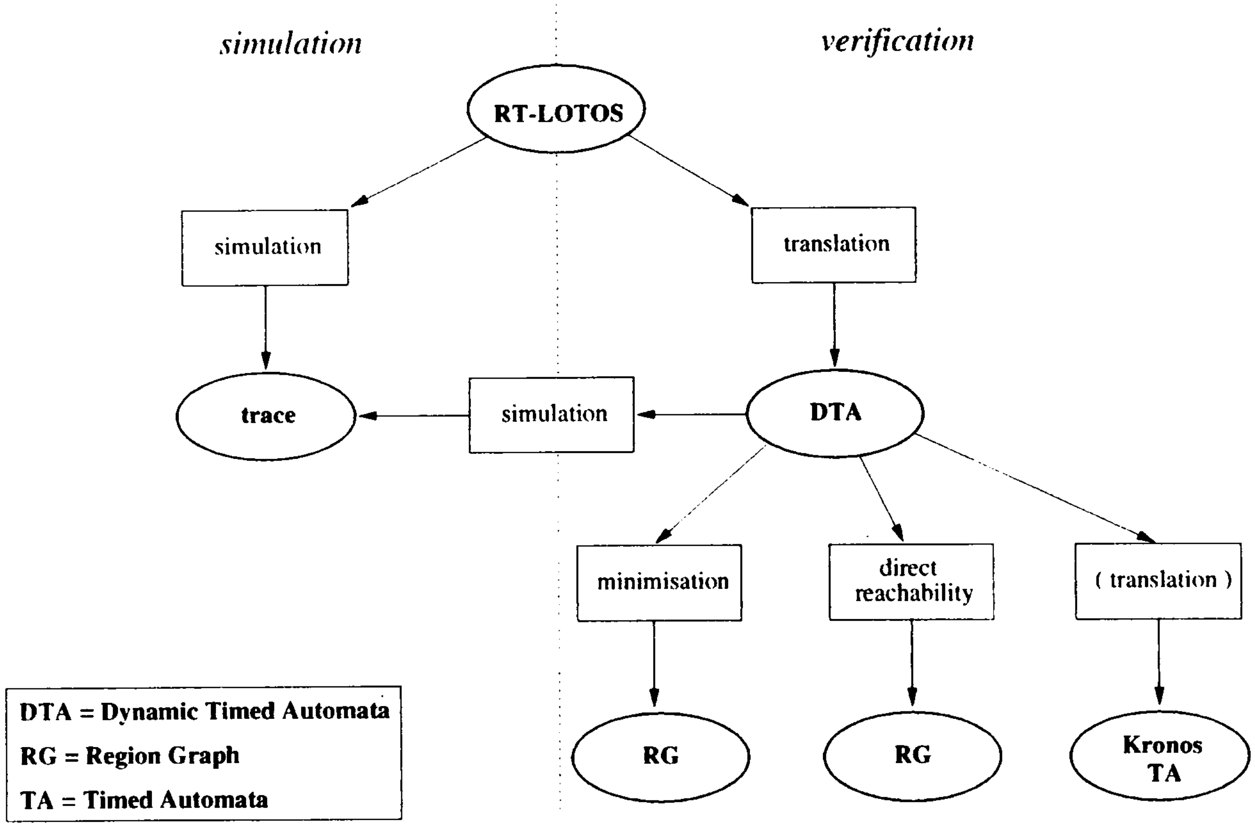


Fig. 9. rtl Tool building blocks.

3.2.4. Model checking

In an attempt to study in greater detail the practical gap between the observer approach and classical model-checking, rtl is being interfaced with a model checker developed for classical timed automata (TA) [21]. For this purpose, the translation of an RT-LOTOS specification has been adapted to produce a TA model rather than a DTA, as introduced previously. The full integration of rtl with Kronos is being implemented.

3.2.5. Why is the DTA model attractive?

The DTA model, initially developed to take into account non-regular RT-LOTOS processes, has proven to be very efficient since it drastically reduced the number of clocks to be associated with each control state of the model. The price to be paid corresponds to the definition of function θ, which expresses how some clocks are copied from one state to another (see the formal definition of the DTA in Appendix A). However, this price is not too high, since we consider the DTA model as an intermediate model automatically derived from the RT-LOTOS specification, and not as a specification model.

Fig. 8 illustrates two ways of generating a DTA from an RT-LOTOS specification: on the left, the classical approach as formalized in Ref. [12] (the idea is basically to associate a clock with each parallel component of the specification); on the right, the optimized approach permits to reduce the number of clocks in each state. The interesting point is that both reachability graphs are strictly identical (the definition of the clock regions are obviously different since they directly depend on the clocks defined for each DTA state, but this does not impact the resulting reachability graphs).

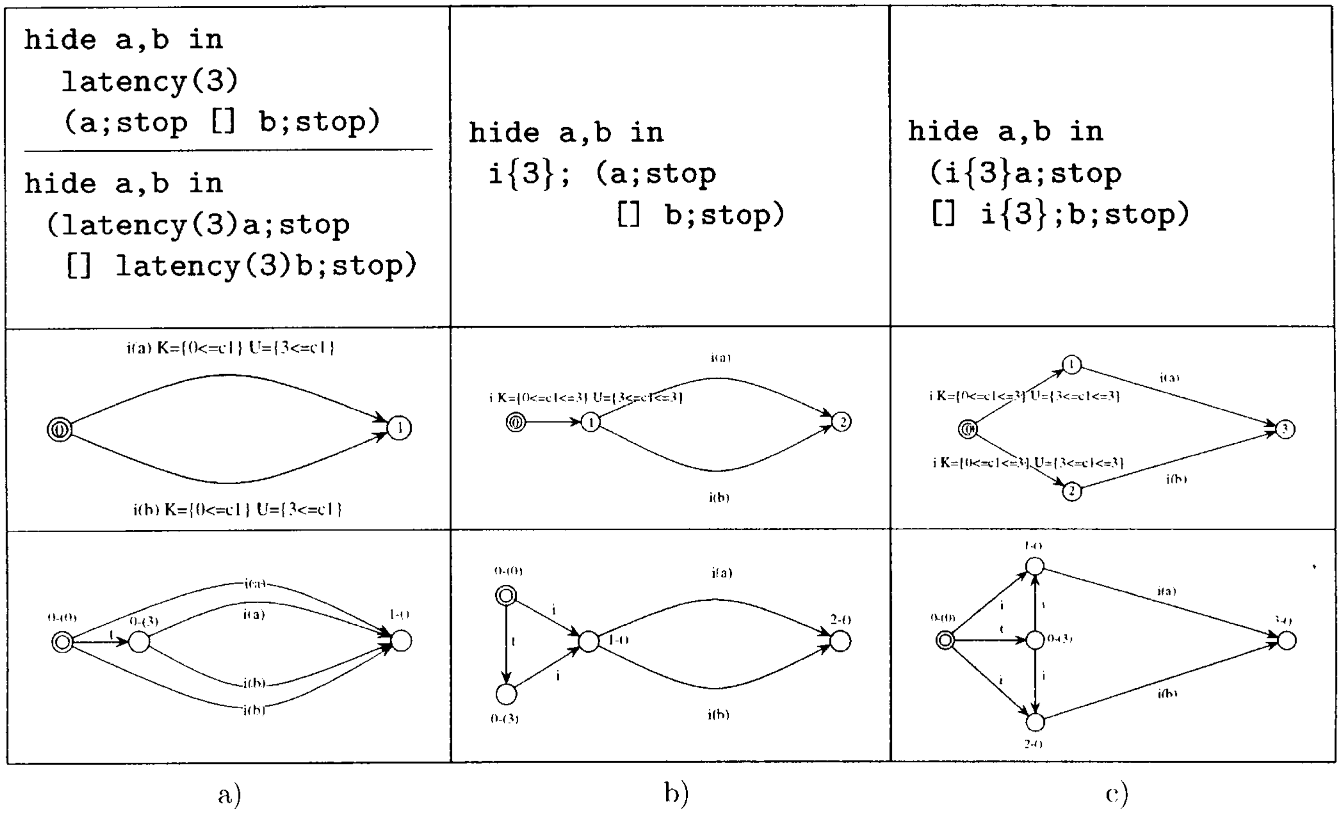
3.3. Availability of the rtl tool

A global picture of the rtl tool is presented in Fig. 9, allowing identification of the basic modules discussed so far.

As far as simulation is concerned, several options for driving a simulation have been defined within the simulation kernel and various graphical output formats have also been implemented for displaying the behavior of a specification. Several parameters related to the size of the state space can also be analyzed when performing the reachability analysis.

rtl has been interfaced to several other tools like the Auto/Graph [22] tool for displaying reachability graphs and the Aldebaran [23] tool for performing reachability graphs minimization (derivation of quotient automata with respect to some (un-timed) behavioral equivalences).

rtl is freely available for universities on both Solaris and Linux (information for using rtl is available in Ref. [24]).



```
hide a,b in
  latency(3)
  (a;stop [] b;stop)
```

```
hide a,b in
 (latency(3)a;stop
  [] latency(3)b;stop)
```

```
hide a,b in
 i{3}; (a;stop
        [] b;stop)
```

```
hide a,b in
 (i{3}a;stop
 [] i{3};b;stop)
```

Fig. 10. Comparison between `latency(t)` and `i{t}`.

3.4. Comparison between RT-LOTOS and ET-LOTOS

RT-LOTOS and ET-LOTOS [5] present many features in common from the syntactic and semantic points of view. In fact, whenever possible, RT-LOTOS has been based on the predecessor of ET-LOTOS, namely Timed LOTOS [6]. As a consequence, RT-LOTOS shares important properties with ET-LOTOS, like the non-urgency of observable actions, as well as the urgency of internal actions. Timed LOTOS also introduced a non-deterministic delay operator, whose formal definition led to several delicate semantic problems. Based on this observation, ET-LOTOS and RT-LOTOS proposed independently alternative solutions for expressing non-deterministic delays.

The solution proposed in ET-LOTOS consists in removing the non-deterministic delay operator of Timed LOTOS, keeping only a classical deterministic delay operator. Non-deterministic temporal behaviors are then expressed by a variation of the `i` action, namely the `i{t}` action, which permits to relax temporarily the urgency of action `i` within time interval [0,*t*].

The solution proposed for RT-LOTOS consists in introducing the `latency` operator to express, in a general way, time non-determinism without suffering the side effects introduced by the internal action (remember that the choice is resolved by an `i` action).

To better understand the subtle semantic implications of both solutions, let us look at the simple examples of Fig. 10, where, for each specification, both the DTA and the reachability graph are provided.

Fig. 10a illustrates the use of the latency operator, with two specifications leading to the same behavior (as a result, note that it is possible to distribute the latency with respect to the choice without changing the behavior).

The behavior expressed in Fig. 10b is quite different, since the actual time at which i(a) or i(b) may occur is not determined independently for both actions, but at the level of the occurrence of action `i{t}`. To ensure this independence, one could distribute the `i{t}` operator, leading to a new behavior where the occurrences of actions i(a) and i(b) are no longer synchronized (Fig. 10c). However, in this situation, the choice is not resolved by the occurrence of either i(a) or i(b) as expected, but instead by the `i{t}` action, leading consequently to other potential problems when composing this process with others in a larger specification.

Thus, the latency operator makes it possible to express non-deterministic delays without internal events. It is consequently a much more general solution than the `i{t}` construct for expressing time non-determinism. This claim is supported by the three following rationales:

- Behaviors expressed by means of the `i{t}` construct can always be modeled using the latency operator (behavior i{t}; P is expressed by (latency(t) exit) ≫ P), but the reverse is not true (see the previous example).

- The use of the latency operator avoids, in general, introducing useless internal actions, as well as their associated states, in both the DTA and the reachability graphs. It is further recognized that the occurrence of these internal actions may be uncomfortable in choice contexts.
- There exists a weak bisimulation equivalence for RT-LOTOS having the same congruence properties as Basic LOTOS, which is not the case for ET-LOTOS (see Ref. [4] for details).

A criticism made against the latency operator of RT-LOTOS has been that its introduction complicates the operational semantics; this is true for the classical SOS operational semantics which has been proposed for RT-LOTOS (additional subscripts are introduced to distinguish internal actions that are urgent from those that are not) [4,11]. However, this complexity vanishes with the DTA generation, since for the new DTA definition detailed in Appendix A (see Ref. [12] for comparison), urgency conditions are expressed globally at the level of the definition of the *U* domain, without having to discriminate between different types of internal actions as previously.

4. RT-LOTOS case studies: a return from experience

The benefits of using RT-LOTOS are demonstrated by three case studies that have been carried out at LAAS–CNRS over the past three years. Different application domains have been addressed, different validation techniques, as well as different rtl features have been used to complete them. The first deals with the use of RT-LOTOS for specifying and validating new multimedia synchronization protocols. The second has been carried out over two years in cooperation with the industry, namely *Electricité de France* (EDF in short). Its purpose was the assessment of the application of a formal method to the reverse engineering of (a part of) a fault-tolerant system monitoring system. The third, currently supported by the *Télécommunications* research Program of the CNRS, deals with the formal design of hypermedia documents and addresses the issue of checking the temporal consistency of these documents.

4.1. The formal specification and validation of multimedia synchronization protocols

Multimedia synchronization allows the co-ordination, scheduling and presentation of multimedia objects in time and space. In this context, temporal synchronization raises two main issues:

- How simple temporal dependencies can be guaranteed when delivering a particular media, like an audio or a video stream; this is usually called *intra-stream synchronization*;
- How structural temporal dependencies can be guaranteed when presenting different media together, like for instance a video stream to be lip-synchronized with its associated audio stream; this is usually called *inter-stream synchronization*.

RT-LOTOS has been used to formally specify and validate intra-stream and inter-stream synchronization mechanisms. A main motivation of this work has been to produce unambiguous specifications of complex synchronization mechanisms and to show their performance and correctness by simulation.

A first experiment has dealt with classical jitter-compensation mechanisms for intra-stream synchronization. It includes the formal specification and validation of three different algorithms (details reported in Ref. [25]), aiming, respectively:

- to perform jitter compensation by a prefetch technique,
- to estimate the drift between the local clock rate (at the receiver) and the sender clock rate,
- to adapt the delivery period of the stream with respect to the local clock rate in order to match as closely as possible the nominal period of the stream (itself determined with respect to the sender clock rate).

Starting from raw simulation traces generated by the rtl simulation kernel, different interesting results were obtained for the considered protocol, like Ref. [25]:

- The evolution of the upper and lower bounds of the reception buffer, used for compensating the jitter and estimating the drift between the sender and the receiver clocks;
- The tuning of the delivered stream period, as a function of time, for different assumptions about the drift between the sender and receiver clocks.

A second similar experiment has dealt with classical lip-synchronization mechanisms [26]. A lip-synchronization characterizes the synchronization, up to a certain level of QoS, of a video stream with its associated audio stream. The following QoS requirements are commonly recognized as expressing good quality lip-synchronization:

- *Intra-stream synchronization requirement for audio*: audio packets are to be presented every 30 ms, without jitter;
- *Intra-stream synchronization requirement for video*: video packets are to be presented every 40 ms, with a tolerable jitter of 5 ms;
- *Inter-stream synchronization between audio and video*: video must not lag the associated audio packet by more than 150 ms; video must also not precede the associated audio packet by more than 15 ms.

The simulation strategy carried out for this experiment has been based on the observer approach. Process QoS_Watcher, depicted in Fig. 11, is an observer module

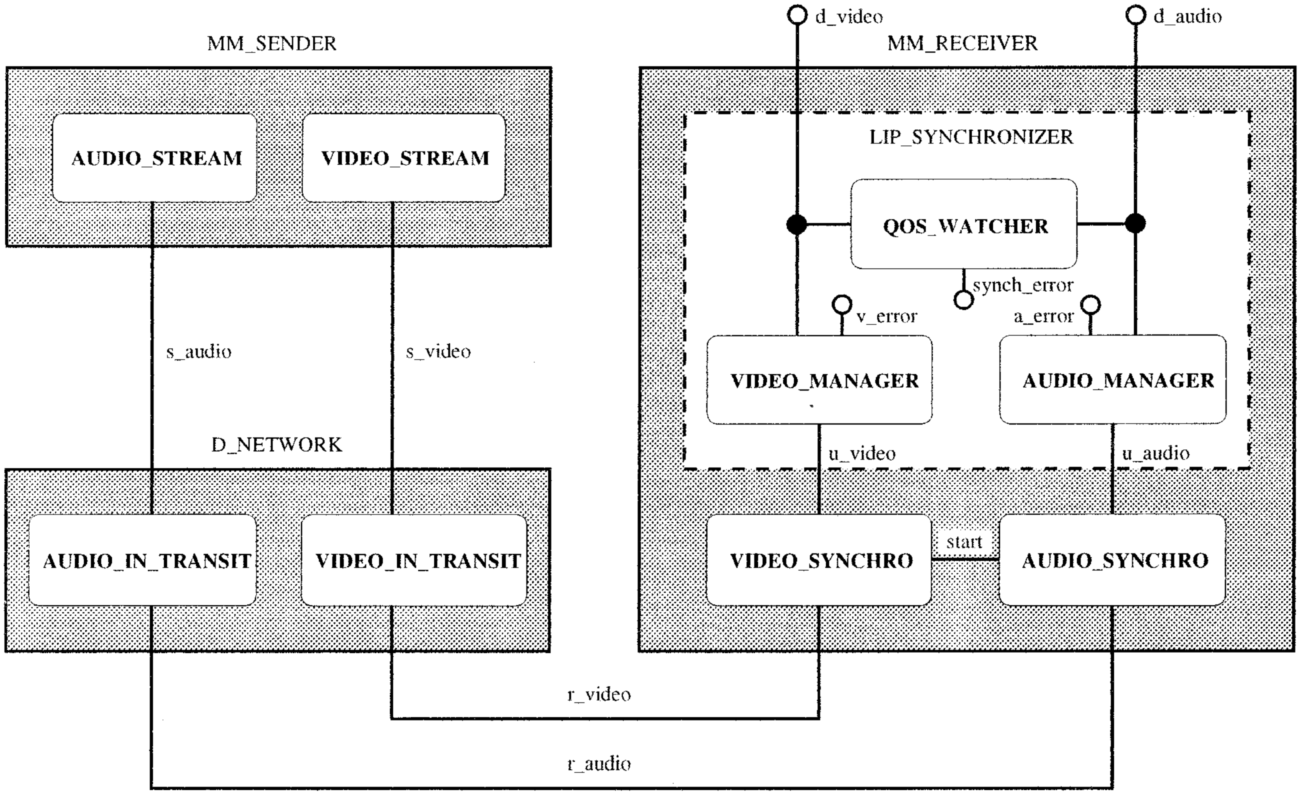


Fig. 11. High-level architecture of the lip-synchronization RT-LOTOS specification.

checking at runtime whether the previous QoS requirements are met. In the case of a QoS violation, action `synch_error` is offered (action specified as a goal for the simulation kernel). Different results of the proposed lip-synchronization mechanism have been provided under different assumptions related to the delay and jitter of the underlying network service (expressed at the level of the specification of processes `Audio_in_Transit` and `Video_in_Transit`).

A third experiment has dealt with the specification of a new multimedia synchronization mechanism based on both causal and temporal relations. The motivation behind the use of RT-LOTOS has been to specify formally this new synchronization mechanism, called the conditional delivery mechanism, and to assess via simple simulation scenarios the correct behavior of the specification. Details can be found in Ref. [27].

4.2. Reverse engineering of a fault-tolerant monitoring system

4.2.1. General framework

This experiment deals with the formal specification and validation of a fault-tolerant monitoring system for the control room of French N4 nuclear power plants. The monitoring system, which has been designed to be transparent to a single failure, is composed of two computers operating in hot redundancy. Both machines, master and slave, process the same application inputs and monitor their internal errors. The master is in charge of application process scheduling and transmission of application messages. In case of a single error, the faulty computer is isolated and the other one becomes master. A distributed algorithm has been devised for implementing this hot redundancy scheme (see the monitoring system functional decomposition in Fig. 12).

Although non-critical for the safety of the nuclear power plant, the monitoring system has been deemed sufficiently representative by the French EDF to launch and support an experiment aiming to assess the use of formal methods in the reverse engineering of a part of this system, namely the scheduling algorithm. The main expectations on the project achievements were two-fold:

- to assess the feasibility of reverse engineering starting from an analysis of the monitoring system source code, written in Ada, which has been implemented by a third party following the (informal) requirements of EDF;
- to better understand and assess the fault-tolerant capabilities of the scheduling algorithm under several faulty conditions.

4.2.2. Why RT-LOTOS?

Three main requirements have been expressed by EDF for the selection of a particular formal description technique for this study:

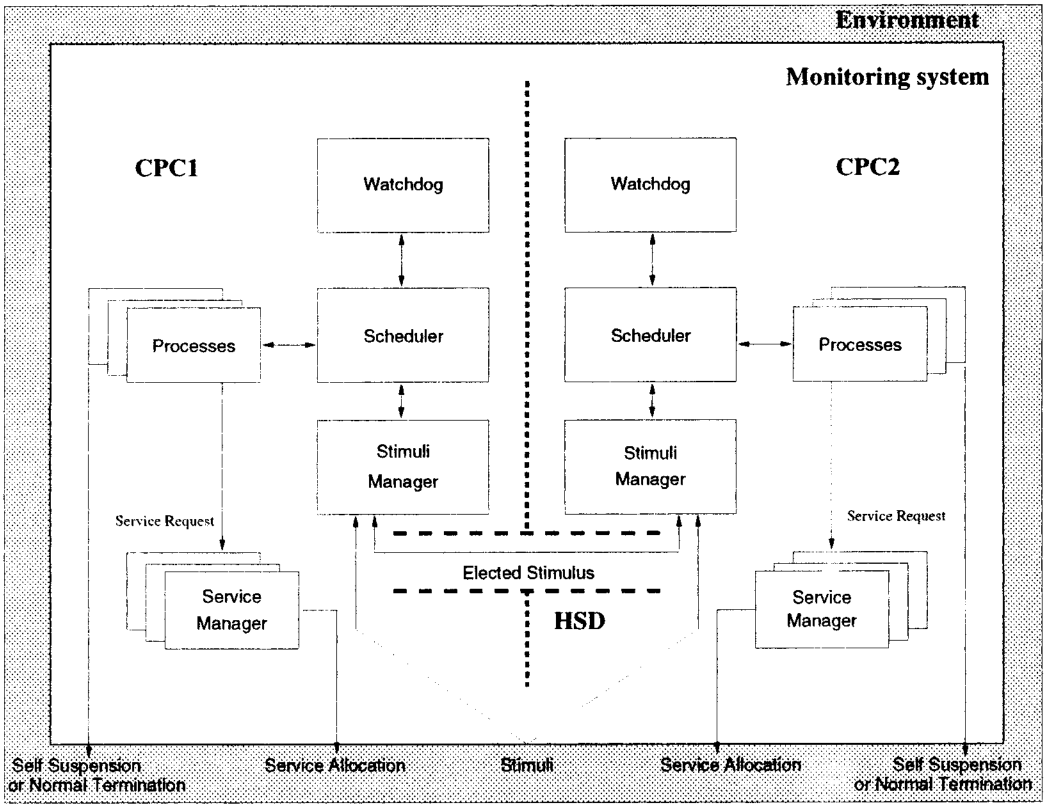


Fig. 12. Functional decomposition of the monitoring system.

- To have executable specifications to facilitate the reverse engineering process;
- To represent the physical distribution of the monitoring system components, these components running asynchronously relative to each other;
- To specify a large and complex system made of several components; the method should therefore provide facilities for composing large specifications from simpler and possibly reusable components.

The previous requirements led to the choice of a process algebra. Assuming also that explicit time constraints had to be expressed, the availability of the `rtl` software tool for validating formal specifications was the main reason which led to the choice of RT-LOTOS.

4.2.3. Results achieved

4.2.3.1. Formal specification The design method for producing the system formal specification has followed the LOTOS design methodology developed within the European LotoSphere project [32]. The key concept of the approach is the design trajectory. A *design trajectory* is made up of different design steps. Starting from an initial high-level specification expressed in LOTOS, the execution of each design step leads to refining the specification by using *transformations*. Two of these transformations, known as the *functionality decomposition* and the *functionality rearrangement* are particularly useful for building complex specifications step-by-step. The same design method has been adapted to RT-LOTOS, since the difference between RT-LOTOS and LOTOS lies mainly in the elemenmtary action offering, and not in the composition operators.

In the informal specification, it is easy to make a distinction between the behavior and the data parts of the system. The behavior part results in a composition of RT-LOTOS processes using the parallel composition, the choice,…operators. The data part describes the value (messages) exchanged between processes through the synchronization actions. Every message structure (stimulus, event notification, mode,…) defined for the monitoring system has been translated into a particular data type.

Modeling (internal) failures of the monitoring system consists basically in introducing new behaviors in the formal specification of the system that lead, after some random delay, to the occurrence of an event characterizing a failure detection and leading to the activation of the associated recovery mechanism (this is consistent with the fact that the failure detection mechanism is considered as fully reliable).

The resulting specification comprises around 50 processors for a total of approximately one thousand RT-LOTOS lines (without the data part). Each leaf process (defined at the bottom of the process hierarchy) is rather simple and merely corresponds to a state machine of few symbolic states.

4.2.3.2. Validation. Simulation has been used extensively for debugging the specification. It has been particularly useful for identifying undesirable deadlock situations caused by the incorrect specification of RT-LOTOS process synchronizations. Debugging has essentially been achieved by the display of simulation event traces, from which scheduling diagrams have been constructed. The monitoring system specification is composed of many processes and gates where data values are exchanged. Pattern scanning languages and processing languages, like Awk and Perl, have been used for displaying relevant parameters of the system as a function of time, starting from the raw data extracted from the simulation traces. It is thus possible, with a minimal effort, to observe the evolution of several parameters of the system like: the time required for executing a process, the load of the stimuli queues, the number of elected stimuli per period of time, among others.

The nominal behavior of the scheduling system is characterized by the following property: the stimuli sequence elected on the slave computer is identical to that of the master, with the possible exception of the last stimulus elected on the master (which may not have been scheduled by the slave yet).

Using the observer approach, intensive simulations of the monitoring system specification have been performed with different sets of parameter values (essentially time parameters). Assuming the time parameter configuration provided by EDF, no action `error` has been detected in these simulation runs (some lasting many hours).

Then, temporal values characterizing some slave computer delay when running the application processes, were selected. Since the slave computer elected stimuli queue has by definition a limited capacity, an excessive delay with respect to the master computer causes stimuli to be lost, leading to the occurrence of action `error`. In this way, it has been possible to identify a set of parameters leading to an incorrect behavior of the scheduling algorithm. These parameter values have been analyzed by EDF and the third party software company in charge of the implementation of the scheduling algorithm. Several changes have been made in the monitoring system in order to overcome the (potential) error situations identified by these simulations. Of course, validation by simulation cannot be considered as a formal proof, since it does not cover the complete specification state space. However, simulation results already provide some level of confidence on the specification quality and on the validity of the desired properties.

Using observers, the verification approach is simple and merely corresponds to a standard reachability analysis. The same observers used within a simulation framework can be selected for formal verification.

Verification by reachability analysis is faced with the classical problem of state space explosion. Several simplifications have been made to the initial specification:

- The number of parallel components has been reduced by decreasing the number of application processes in each computer and the number of services that may be requested by these application processes.
- The specification has been simplified by withdrawing any behavior that does not directly relate to the property under verification.
- The internal architecture of the specification has been simplified, by replacing whenever possible a process composition by an equivalent unique process.
- The value domain of some parameters has been reduced, and parameters that do not directly interfere with the scheduling algorithm have been removed.

Various verification-oriented specifications have been derived from the initial formal specification that has extensively been validated by simulation. Each verification-oriented specification includes an observer process to verify the relevant property. Details on the results achieved are reported in Ref. [28].

4.3. Checking the temporal consistency of hypermedia documents

Hypermedia documents are expected to satisfy temporal consistency properties stating that temporal synchronization constraints to be met during a document presentation are not in conflict with one another. Depending on how these synchronization constraints are defined, there does exist a risk of creating inconsistent situations, where different contradictory synchronization requirements cannot be satisfied together, leading to undesirable deadlocks (global or partial) during a document presentation.

4.3.1. Formal specification of hypermedia documents

A first experiment for formalizing hypermedia documents has been reported in Ref. [29], where authoring is directly made in RT-LOTOS. Such an approach is not really convenient in practice, since the author uses RT-LOTOS objects rather than hypermedia objects, thereby rendering the authoring process too complex.

Another experiment has been reported in Ref. [30] within the framework of a new hierarchical temporal synchronization model. This model is based on a library of predefined objects representing basic presentations and constraints inherited from various temporal models and defines rules for composing these objects. The objects have been modeled as RT-LOTOS processes, and the model composition rules are expressed as RT-LOTOS process parallel compositions with synchronization. From a high-level authoring based on the previous objects, it is

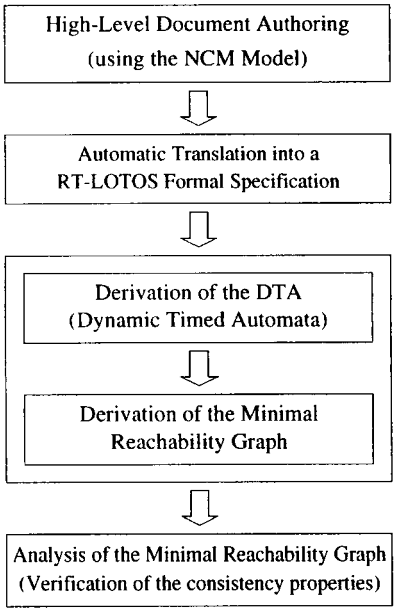

High-Level Document Authoring
(using the NCM Model)
Automatic Translation into a
RT-LOTOS Formal Specification
Derivation of the DTA
(Dynamic Timed Automata)
Derivation of the Minimal
Reachability Graph
Analysis of the Minimal Reachability Graph
(Verification of the consistency properties)

Fig. 13. Main steps of the formal design methodology.

then possible to derive automatically the RT-LOTOS formal specification of the document.

This approach, which, in practice, is much more convenient than the previous one, still presented two weaknesses: the synchronization model failed to handle completely the link concept as used within the hypermedia community, and above all the model failed to make a clear distinction between a node type and a node instance; the latter point has been considered as a major limitation, since it is not easy to re-use part of a document (a node) in the design of another document.

An extended approach has been proposed in Ref. [31] to overcome the previous limitations. Instead of defining a new formalism for the description of the temporal structure of a hypermedia document, we use a general-purpose authoring model developed with the hypermedia community and perform the mapping of this model into RT-LOTOS. The authoring model relies on objects that are usually found in hypermedia document specifications, namely nodes (including composite nodes), anchors and links. The authors use these objects for editing their documents, whereas the RT-LOTOS formal specification is automatically derived from the high-level authoring, and then completely hidden to them. This approach, illustrated in Fig. 13, has been fully implemented for the NCM (Nested Context Model) authoring model implemented within the Hyperprop environment (see Refs. [31,32] for details).

4.3.2. Consistency checking of hypermedia documents

By definition [30], we consider that a document presentation is consistent if the action characterizing the start of the document presentation is necessarily followed, some (finite) time later, by an action characterizing the end of the presentation. As a result, action `end` should be reachable from the initial state of the RT-LOTOS reachability graph, assuming, by construction, that action *start* is the unique action enabled in the initial state. This definition of consistency has recently been refined to distinguish inconsistencies generated by internal or external non-deterministics events (see Ref. [33] for details).

Temporal consistency definition is unique, it may be checked on different specifications which lead to the characterization of two basic types of temporal consistency properties, called *intrinsic* and *extrinsic* temporal consistency, respectively [31].

Intrinsic temporal consistency properties are checked on the hypermedia document RT-LOTOS specification. These properties are said to be intrinsic since they depend only on the inter-media synchronization constraints defined by the user when authoring the hypermedia document. Many reasons may lead to temporal intrinsic inconsistencies:

- Inconsistencies between the expected duration of the nodes and the logic of the synchronization links enforcing their termination.
- Conflicting synchronization links.
- Bad timing of the synchronization links.
- Omission of some default cases (for instance, what happens to a node featuring several anchors, when none of them is triggered by the user: should the node end or not? what is the impact on the other nodes?).

Extrinsic temporal consistency properties are checked on a new RT-LOTOS specification corresponding to the composition of the hypermedia document specification with an RT-LOTOS process characterizing additional presentation constraints related to the physical or even virtual devices on which one wants to play the different media of the document. These properties are said to be *extrinsic* since they do not only depend on the hypermedia document temporal structure, but also on the additional constraints. The purpose of these additional presentation constraints is essentially two-fold:

- To model constraints occurring at the level of the *physical presentation* devices that belong to the underlying multimedia platform; these constraints permit to check whether extra delays enforced by the multimedia platform do impact the document presentation; they also permit to check whether, during the document presentation, concurrent accesses to shared physical presentation devices may lead to deadlock situations.
- To model constraints occurring at the level of the *virtual presentation devices* that characterize the type of media content one wants to present.

5. Some lessons learned

Over the last three years, RT-LOTOS has been experimented in three different application areas (protocol engineering, fault-tolerant monitoring systems, hypermedia authoring), with different purposes in mind.

In the first case study (protocol engineering), the specifications were built from scratch using informal textual specifications and the main purpose was to provide unambiguous specifications of complex synchronization mechanisms, and to perform various simulation runs so as to get a good level of confidence on the synchronization mechanism behavior. Some performance results have also been extracted from the raw simulation traces (use of simulation graphs to display some variables—or functions of these variables—versus time).

The second is the most complex application developed so far with RT-LOTOS. Several features of RT-LOTOS have proven particularly well-suited to this application where concurrency, synchronization and explicit timing constraints have to be dealt with together in an asynchronous environment. Some of the lessons gained through this experiment are the following:

- *The specification phase has been much simpler than initially expected*; the re-engineering process has been greatly facilitated by the existence of Ada flow charts, state diagrams, …, and by the industrial partners' in-depth knowledge of their system (to validate specification alternatives).
- *The simulation phase has brought much more results than initially expected*; many simulations runs have been conducted for debugging the initial specification, and then for validating the scheduling algorithm behavior with various parameter configurations. Error situations have been reported for some parameter values, which led to modifications in the code of the scheduling algorithm.
- *The verification phase has been as difficult as initially expected*; several improvements to the `rtl` tool have been made during the project, mostly for reducing the number of bytes encoding an RT-LOTOS state in memory; although verification results have been obtained only for simplified configurations of the monitoring system, the validity of the proposed approach has been shown on a real-size industrial application.

One important return of this case study has been also the successful trade-off achieved between simulation and verification. Both have to be carried out consistently and in cooperation:

- verification-oriented specifications have been derived using a strict methodology, from the initial complete formal specification; thus, the designer can assess the validity of the simplification assumptions made;
- the observer approach has been used for both simulation and reachability analysis; errors detected by simulation have been better understood by analyzing reachability graphs, and vice versa reasons preventing the convergence of reachability graph minimization have been better understood thanks to the simulation.

The third case study is related to a different application area, multimedia authoring. Emphasis has been laid on the formal verification of temporal consistency properties performed on a RT-LOTOS specification automatically generated from a high-level authoring model. This has been possible by the existence in LOTOS of a general parallel composition operator with multi-way synchronization, that makes it easy to compose synchronization constraints. It is then possible to express large complex behaviors by merely composing very simple basic processes. This study clearly demonstrates as well that formal methods may be very useful in practice, once they are completely *hidden* from the users!

6. Conclusion

In this paper, RT-LOTOS and its associated software tool `rtl` have been described. We have shown, with various case studies pertaining to different application domains, that RT-LOTOS is a *general-purpose formal description technique* well-suited to situations where both time constraints and asynchronous events, as well as non-determinism and concurrency have to be dealt with.

We have also discussed the relationship between RT-LOTOS and ET-LOTOS, detailing the unique important difference between both approaches, namely the constructs used for expressing time non-determinism (`i{t}` construct of ET-LOTOS versus the `latency(t)` operator of RT-LOTOS), and their implication on the resulting reachability graphs. In spite of this difference, several conclusions that have been reached and the methods we applied, could also be directly applicable to ET-LOTOS formal specifications.

Finally, we have emphasized the importance of design methods based on formal approaches, such as the one we are developing for hypermedia document authoring.

Acknowledgements

We are grateful to Mr Roberto Cruz de Olivera for his implementation of the first version of the `rtl` tool. We wish also to thank all the persons who have been involved in this project, and in particular: Laurent Andriantsiferana, Luiz Fernando Rust da Costa Carmo, Rachid Benabbou, Emanuel Castel and Paulo Sampaio. This project has been partially supported by EDF and CNRS. The second author has been partially supported by a grant from CAPES/Brazil.

Appendix A. Formal definition of the DTA model

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

Let $C_{\mathrm{set}} = \{c_i | i \in \mathbb{N}^+\}$ be a set of clocks. A timed condition is a conjunction of inequalities of the form $m \prec c_i \prec M$ where $m, M \in D_0^\infty, \prec \in \{<, \leq\}$, and $C_i \in C_{\mathrm{set}}$. These inequalities are also represented by the notation (c_i, I_i) where I_i is a time interval like $[m, M]$, $]m, M]$, $[m, M[$ or $]m, M[$. Let $\nu \in D_0^N$ be an N-tuple and K_N a timed condition defined only on the clocks $\{c_i | i \leq N\}$. We will use the notation $\nu \models K_N$ to indicate that $\nu(i) \in I_i$ for each $(c_i, I_i) \in K_N$.

Definition 1 (Dynamic Time Automaton). *A Dynamic Timed Automation is a 4-tuple* $(S, Nclock, E, s_0)$, where

- S is a finite set of control states,
- $Nclock : S \rightarrow N$ is a function associating the number of clocks with each control state,
- E is a finite set of transitions of the form $(s, s', K, U, a, C, \theta)$, where $s, s' \in S$ are the source and destination control states of the transition, K and U are timed conditions, $a \in L$ is a labeling action, $C \subseteq \{1, \ldots Nclock(s')\}$ defines the indexes of the clocks to be reset when the transition is fired, $\theta : N^+ \rightarrow N^+$ is a partial injective clock setting function (see the constraints to be satisfied by a clock setting function in Ref. [12]),
- s_0 is the initial control state.

A labeled transition system $LTS(T)$ is associated with each dynamic timed automation T. A state $\sigma = (s, \nu)$ of $LTS(T)$, also called a configuration, is fully described by specifying the control state s of T and the values $\nu \in D_0^{Nclocks(s)}$ of all clocks defined for that control state. The transitions of $LTS(T)$ correspond either to explicit transitions of T 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 $LTS(T)$). The initial state of *LTS(T)* is the state (s_0, ν_0) where $\nu_0 \in D_0^{Nclocks(s_0)}$ with $\nu_0(i) = 0$ for $i \in [1, N\,\mathrm{clock}(s_0)]$.

Definition 3 (*Explicit transitions of* $LTS(T)$). Let $(s, \nu) \in LTS(T)$ and $e = (s, s', K, U, a, C, \theta) \in E$ a transition of T. If $\nu \models K$ then $(s, \nu) \xrightarrow{a} (s', \nu')$ where $\nu'(i) := 0$ for $i \in C$ and $\nu'(i) := \nu(\theta^{-1}(i))$ for $i \notin C$ and $i \in [1, Nclock(s')]$.

Definition 4 (*Implicit transitions of* $LST(T)$). Let $(s, \nu) \in LTS(T)$ and $t \in D$. If $\nu + t' \not\models K \cap U$ for each $0 \leq t' \leq t$ and each $(s, s', K, U, a, C, \theta) \in E$ with isUrgent (a), then $(s, \nu) \xrightarrow{t} (s, \nu + t)$, assuming that isUrgent is a predicate defined on L characterizing whether an action is urgent within a domain U.

References

[1] ISO Standard 8807, LOTOS, a formal description technique based on temporal ordering of observational behavior, 1988.

[2] R. Milner, Communication and Concurrency, Prentice Hall, Englewood Cliffs, NJ, 1989.

[3] C.A.R. Hoare, Communicating Sequential Processes, Prentice Hall, Englewood Cliffs, NJ, 1989.

[4] J.-P. Courtiat, R. Cruz de Oliveira, About time nondeterminism and exception handling in a temporal extension of LOTOS, Fourteenth International Conference on Protocol Specification, Testing and Verification (PSTV'94), Vancouver, Canada, Chapman and Hall, London, 1994.

[5] L. Léonard, G. Leduc, An introduction to ET-LOTOS for the description of time-sensitive systems, Computer Networks and ISDN Systems 29 (1997) 271–292.

[6] G. Leduc, L. Léonard, A Timed LOTOS supporting a dense time domain and including new timed operators, Fifth International Conference on Formal Description Technique Protocol (FORTE'92), Lannion, France, North-Holland, Amsterdam, 1992.

[7] J.-P. Courtiat, P. Dembinski, G. Holzmann, L. Logrippo, H. Rudin, P. Zave, Formal methods after 15 years: status and trends, Computer Networks and ISDN Systems 28 (1996) 1845–1855.

[8] T. Bolognesi, E. Brinksma, Introduction to the iso specification language LOTOS, Computer Networks and ISDN Systems 14 (1987) 1.

[9] L. Logrippo, M. Faci, M. Haj-Hussein, An introduction to LOTOS learning by examples, Computer Networks and ISDN Systems 23 (1992) 325–342.

[10] T. Bolognesi, F. Lucidi, LOTOS-like process algebras with urgent or timed interactions, Proceedings of the Fourth Conference on Formal Description Techniques (FORTE'91), New York, USA, Elsevier, Amsterdam, 1992.

[11] J.-P. Courtiat, R. Cruz de Oliveira, On RT-LOTOS and its application to the formal design of multimedia protocols, Annals of Telecommunications 50 (1995) 11–12.

[12] J.-P. Courtiat, R. Cruz de Oliveira, A reachability analysis of RT-LOTOS specifications, Eighth International Conference on Formal Description Techniques Protocol (FORTE'95), Montreal, Canada, Chapman and Hall, London, 1995.

[13] R. Alur, C. Courcoubetis, D. Dill, Model-checking for real-time systems, Proceedings of the Fifth IEEE Symposium on Logic in Computer Science, 1990.

[14] T. Bolognesi, J. van de Lagemaat, C. Vissers, LOTOSphere: Software Development with LOTOS, Kluwer Academic, Dordrecht, 1995.

[15] X. Nicollin, J. Sifakis, An overview and synthesis on timed process algebras, REX Workshop Real-Time: Theory in Practice, Lecture Notes in Computer Science, vol. 600, Springer, Berlin, 1991.

[16] B. Berthomieu, M. Diaz, Modeling and verification of time-dependent systems using time Petri nets, IEEE Transactions on Software Engineering 17 (1991) 3.

[17] R. Alur, C. Courcoubetis, N. Halbwachs, Minimization of timed transition systems, CONCUR'92, vol. 630, Springer, Berlin, 1992.

[18] L. Cacciari, O. Rafiq, A temporal reachability analysis, Proceedings of the 15th International Symposium on Protocol Specification, Testing and Verification (PSTV'95), Warsaw, Poland, 1995.

[19] M. Yannakakis, D. Lee, An efficient algorithm for minimizing real-time transition systems, CAV'93, Lecture Notes in Computer Science, vol. 697, Springer, Berlin, 1993.

[20] R. Alur, D. Dill, The theory of timed automata, REX Workshop Real-Time: Theory in Practice, Lecture Notes in Computer Science, 600, Springer, Berlin, 1991.

[21] Kronos, http://www-verimag.imag.fr/temporise/kronos .

[22] ATG-FCTOOLS, http://www-sop.inria.fr/meije/meijetools.html.

[23] H. Garavel, Open/Caesar: an open software architecture for verification, simulation and testing, First International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'98, March 1998.

[24] RT-LOTOS, http://www.laas.fr/~courtiat.

[25] J.-P. Courtiat, R. Cruz de Oliveira, RT-LOTOS and its application to multimedia protocol specification and validation. In invited paper at

the International Conference on Multimedia and Networking, Aizu, Japan, 1995.

[26] J.-P. Courtiat, R. Cruz de Oliveira, L. Andriantsiferana, Specification and validation of multimedia protocols using RT-LOTOS, Invited paper at the Fifth IEEE International Conference on Future Trends of Distributed Computing Systems, Cheju Island, Korea, 1995.

[27] J.-P. Courtiat, L.F. Rust da Costa Carmo, R. Cruz de Oliveira, A general-purpose multimedia synchronization mechanism based on casual relations, IEEE Journal on Selected Areas in Communications 14 (1996) 1.

[28] L. Andriantsiferana, J.-P. Courtiat, R.C. de Oliveira, L. Picci, An experiment in using RT-LOTOS for the formal specification and verification of a distributed scheduling algorithm in a nuclear power plant monitoring system, Tenth International Conference on Formal Description Techniques (FORTE'97), Osaka, Japan, Chapman and Hall, London, 1997.

[29] J.-P. Courtiat, M. Diaz, R.C. de Oliveira, P. Sénac, Formal models for the description of timed behaviors of multimedia and hypermedia distributed systems (invited paper), Computer Communications 19, Elsevier, Amsterdam, 1996.

[30] J.-P. Courtiat, R. Cruz de Oliveira, Proving temporal consistency in a new multimedia synchronization model, ACM Multimedia'96, Boston, USA, 1996.

[31] C.A.S. Santos, L.F.G. Soares, G.L. de Souza Filho, J.-P. Courtiat, Design methodology and formal validation of hypermedia documents, ACM multimedia'98, Bristol, UK, 1998.

[32] C.A.S. Santos, J.-P. Courtiat, P. de Saqui-Sannes, A formal methodology for the specification and verification of hypermedia documents, design methodology and formal validation of hypermedia documents, Eleventh International Conference on Formal Description Techniques (FORTE'98), Paris, France, Chapman and Hall, London, 1998.

[33] C.A.S. Santos, P.N.P. Sampaio, J.P. Courtiat, Revisiting the concept of hypermedia document consistency, ACM Multimedia'99, Orlando, USA, 1999.