A SIMULATOR OF REAL-TIME SYSTEMS
SPECIFIED IN ET-LOTOS

Antony de Jacquier* Joël Goossens Christian Hernalsteen Thierry Massart

University of Brussels, Computer Science Department
Boulevard du Triomphe CP 212, 1050 Brussels, Belgium
e-mail: {a.jacquier, joel.goossens, christian.hernalsteen, thierry.massart}@ulb.ac.be
Tel:32 2 650 59 14
Fax:32 2 650 56 09

**Keywords:** Computer-aided analysis, Verification, Real-time simulation, System engineering.

**ABSTRACT**

We present in this paper a tool for the simulation of ET-LOTOS specifications. ET-LOTOS is a time extension of the formal description language LOTOS and is dedicated to the specification of distributed and reactive systems where timing constraints are given. The simulation tool allows to analyze the behavior of the specified system and to make a first verification about its correctness. The specification to simulate is first mapped into a timed automaton model extended with variables. From this intermediate representation, the simulator produces an interactive step by step execution of the specified system.

1 Introduction

The growing use of real-time protocols and distributed systems has pointed out the need for formal methods able to deal with quantitative time. Many process algebras have been extended to allow the design and specification of timed systems, see [NS81] for an overview of FDTs supporting time. ET-LOTOS [LL94] is a timed extension of LOTOS [BB87, ISO88] which allows the description of real-time systems. This extension has been used to define the timed semantics of the new proposed ISO standard E-LOTOS (for Extended LOTOS) [JTC97]. Such a language can only be useful if tools can support it and especially its real-time features.

A simulator is of prime importance in the design process. It allows to analyze the behavior of the specified system and to make a first verification about its correctness, showing that the specified behavior corresponds to the expected one.

A good solution to develop analysis and verification tools for ET-LOTOS is to use an extended automaton model as intermediate model. Timed automaton [ACD90] seems to fit well for this purpose. This model extends the model of automata with clocks and predicates on clocks which allow to specify the time constraints.

Therefore, in previous works [HM95] we have extended the work of Dawes et al. [DOWY94] to define a translation of ET-LOTOS into timed automaton. To reach this goal, we have extended the timed automaton model with variables and we have defined an ET-LOTOS compiler into this model.

This paper is organized as follows: we start by giving a brief introduction to ET-LOTOS. In section 3, we describe the main functionalities and the interface of our simulator. The methods used to implement the tool is then presented in section 4 and finally, some possible extensions are discussed in the concluding remarks.

*This work was supported by the Belgian National Fund for Scientific Research (FNRS).*


2  ET-LOTOS

ET-LOTOS is a timed extension of the ISO standard LOTOS. As in LOTOS, an ET-LOTOS specification contains a data part, not further described here, and a behavior part using a process algebra and specifying the temporal ordering of actions observable by the environment. A LOTOS action occurs at a gate and may have data offers. For instance, in ?y: rational ?z: rational stands for an action at gate in which inputs in variable y and z two rational values from its environment. Moreover, LOTOS uses a special action, the internal action i which is not observable by the environment. LOTOS includes various operators as the sequence of actions (prefix), choice between two behaviors, parallel composition of behaviors, .... Moreover, ET-LOTOS includes:

- the life reducer: a[d]; B
  The action a is limited to occur within d time units. It is not an obligation for an observable action a to occur within d, but if it doesn't, the process behaves like stop. If a is the internal action i, the action must occur within d time units or must be preempted by the firing of another action during this period. Actions must not always be time-restricted with the {d} construction. Observable actions without explicit time restriction have an infinite default value, i.e. these actions can occur at any time. The internal action is considered differently with a default time restriction of zero: it has to occur immediately or must immediately be preempted by another action.

- the time capture: a @t [SP(t)]; B
  If the action a occurs, the variable t contains the time elapsed between the moment this behavior has been offered and the one at which the action has occurred. The selection predicate SP(t) stands for a guard which must be verified when the action occurs. It is used to put timed constraints on the occurrence of the action.

- the delay: A^t B
  The behavior B is offered to the environment after a waiting of d time units.

These three new constructs can be used to define more complex real-time behaviors like time-outs, watch-dogs or timers. The formal semantics of ET-LOTOS can be found in [LL94].

Figure 1 gives an extract of an ET-LOTOS specification and illustrates some of these constructs. Two rational values are waited on the gate in and their sum are then provided, after at least three time units, on gate out. If no interaction occur on the gate out after a time equal to the time waited to receive the two values, an internal action may occur and disable the offers on the gate out.

\[
\begin{align*}
\text{in } & \text{?y: rational } \text{?z: rational } @t; \\
A^3 \text{ (out } & !y+z; \text{ stop} \\
\text{ / } & \text{stop})
\end{align*}
\]

Figure 1: A small example for simulation purposes

3  THE SIMULATOR TOOL

A simulation tool, as the ET-LOTOS specification simulator we have developed, is particularly useful at the design stage of a real-time system. It allows the step by step execution of the formal specification which brings to the specifier a better understanding of the system and allows to verify that the specification actually corresponds to what the designer expected. The simulation of a formal specification allows to detect specification and design errors, early in the development life cycle.

We first present the functionalities of the simulator to highlight the interactions between the tool and the user. A small example is used to illustrate these various functionalities. A description of the tool graphical interface is then given.

3.1 Simulator Functionalities

The goal of our simulator is to help the user to understand its specification and to allow the detection of specification and design errors. To reach this goal the interaction with the simulator must be simple but expressive in term of the information provided. The tool must ease the interaction with the user by providing valuable information but the user must have the complete control on its system. The main concept is that the simulator does not let the system evolve without beforehand informs the user or asks its interaction. The difference with untimed simulation is that the possible evolution of a system is not limited to the occurrence of an action. New events, linked to the timing aspects, must now be
considered as the expiration of a delay, the urgency of an action or the enabling or disabling of an action due to the time passing. The user must control this time passing but the tool must provide information, as for instance the next time an action becomes enabled or disabled, to help the user in the simulation process.

At each simulation step the user may have the choice between the firing of an action and the ageing of the system of a given amount of time. To ease the choice of the user, the simulator provides him several information like the set of enabled actions and the time interval in which this set is left unchanged. The offered actions are the only one enabled in the specified interval, no new action can become enabled by a simple ageing of the system.

The upper bound of the interval, indicates the first time at which the set of enabled actions changes if the system is aged. The user may decide to age the system until this moment where the set of enabled actions is changed. The tool informs also the user when an action is urgent and that the system can no longer be aged. Moreover the tool has also the possibility to age automatically the system when no action is enabled. The automatic ageing is used to reach the first moment when an interaction with the system is possible (when an action becomes enabled).

Let us take the example of figure 1 to illustrate the functionalities of our tool.

Figure 2 illustrates a possible simulation trace of this example. Each box represents the information provided by the tool at each simulation step: the set of enabled actions and the time interval in which this set stays unchanged. The transitions between these boxes show the interactions with the user.

![Simulation Trace Diagram]

Figure 2: A simulation trace

Only one action, in ?rat ?rat is enabled in the initial state. The associated time interval means that this action cannot be disabled by the time passing. Indeed, no life reducer has been specified on this gate. The user can fire the action or age the system of a time quantity in the interval [0, \(\infty\)]. Let us age the system by 10 time units. The simulator ensures to the user, through the specified time interval, that no important events, like the enabling of an action, are missed during this ageing. The new state offers the same action but the time interval has changed. The lower bound of the interval always indicates the current absolute time since the beginning of the simulation. Once again the system can be aged by any time quantity without changing the set of enabled actions. Let us now fire the action in with the rational values 3 and 4. An ageing is now undertaken automatically by the simulator since no action is enabled after the occurrence of the action in 13 14. The system is aged until at least one action becomes enabled; after the expiration of the delay. In the new state the action out 17 is offered in the time interval [13, 23]. This action is the only one enabled during 10 time units which corresponds to the moment where the internal action becomes enabled. The user can now fire the action out 17 or age the system by a maximum of 10 time units. The user is not allowed to age the system by more than 10 time units to ensure that it will not miss an important event of the system. If we age the system by 10 time units, we reach a state where the internal action becomes enabled. The internal action is urgent which is denoted by the time interval; the system cannot be aged before the firing of one of the enabled actions. An action must occur at time 23 to let the system evolve. Whatever the chosen action by the user, the simulator reaches a state where no more actions are enabled and where the system ageing can be infinite. This final state corresponds to the stop behavior.

3.2 Simulator’s Graphical Interface

The figure 3 shows the graphical interface of the simulator executing the small example of figure 1. The main window on the upper left of the figure contains on the left side information about the compilation and the translation into timed automaton of the ET-LOTOS specification to simulate. Three other windows can be launched by the activation of buttons located underneath this information area; one showing the ET-LOTOS specification in course of simulation (the lower left window), another one giving the current trace of simulation (the upper right window) and the last one giving the declared variables with their values (the lower right window).

The right side of the main window is used to con-