Design of timed systems using a real-time process algebra

Thierry Massart, Laurent Van Begin, Eric Van Nuffel

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

April 12, 1999

Abstract

We describe in this paper a method to specify and analyze real-time specifications. We present a set of tools dedicated to the analysis of ET-LOTOS specifications. ET-LOTOS is a timed extension of the process algebra LOTOS which allows the description of time dependent systems. The tools are based on a transformation of ET-LOTOS expressions into an extended model of timed automata. Using this intermediate model, the specification can be either simulated or verified using a model-checker which allows the verification of formulas on timed automata. For our purpose, we have defined the temporal logic ATCTL: an action based real-time extension of the temporal logic CTL. It is then possible to verify temporal properties expressed in ATCTL on an ET-LOTOS specification. A toy example is used to present the functionalities of the tools.

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. Alur and Dill have defined *timed automata* [Alu91] able to determine, by use of clock variables representing the time passing, how long a system can stay in some state and when an event can (or must) occur. Moreover, methods and associated tools have been defined for the analysis of timed automata; model-checking allows to prove that a formula Φ, expressed in temporal logic, is satisfied in the model given by the specification S of the system; i.e. that $S \models \Phi$. For instance UPPAAL [LPY97] and KRONOS [Yov97] allows to verify that a timed automaton satisfies some real-time temporal logic formula.

Another important approach to formally specify systems, is the process algebras. E.g., CCS [Mil89], CSP [Hoa85] or LOTOS [ISO88] have been designed to cope with the lack of structuration in the specification languages based on automata. The key idea of process algebra, is to give, by the use of various algebraic operators composing behaviors, the possible ordering of actions a specification can have. Operator as sequence (action prefix), choice, parallel interleaving or synchronization, disabling and recursion (or process instantiation) are defined. Depending on the style used, one can specify a system as a collection of, dynamically created, processes; each of them specifying some resource, constraint the system must satisfy, ... LOTOS, like other untimed formal description languages, is supported by tools for the design, verification and code generation of specifications [T3.90]. Most of these tools are based on an intermediate representation like Petri nets or automata [FGK+96].

Some process algebras have been extended to allow the design and specification of timed systems. ET-LOTOS [LL94] is a timed extension of LOTOS which allows the formal specification of real-time systems.

It appears that timed automata are adapted for the analysis of systems using model-checking while real-time process algebra as ET-LOTOS are better equiped for the design of specifications. Therefore, timed automaton seems an adequate intermediate model to analyze and verify specifications defined in ET-LOTOS.

In a previous paper [HM95], we have given a transformation, based on an idea given in [DOY94] of ET-LOTOS into timed automata (transformation preserving some temporal branching bisimularity). Moreover, we have defined the corresponding compilation tool [Her97]. Moreover, we have defined a simulator [dJHM98] which helps the user in the specification of his system. Interfacing our compiler with KRONOS, we are able to verify temporal properties expressed in TCTL [Yov93] on a subset of ET-LOTOS.

In this paper we present the logic ATCTL we have defined for our purpose. ATCTL is an action based real-time extension of CTL [CES86] which is more adapted to express properties on specifications defined in process algebras where the notion of action is the key concept on which all specifications are built.

This paper is organized as follows: we start by giving a brief introduction to ET-LOTOS. We present and specify in section 3 a simple example which will be used to illustrate the use of our tools. Sections 4 and 5 present the extended timed automaton model and the transformation of ET-LOTOS into this model. ATCTL and its model checker are described in sections 6 and 7. Finally, conclusions and further works are presented.

2 ET-LOTOS

ET-LOTOS is a timed extension of the ISO standard LOTOS[1]. It introduces three new constructs which can be used to restrict the time intervals where an action can occur:

- 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.

- Time capture: `a @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 variable `t` may be used in guards which allow to disable some choices in the behavior.

- Delay: Δ^d `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 (see [LL94]) is given in terms of labeled transition system composed by two kinds of transitions, namely discrete and timed.

3 A simple railway control system

To illustrate our approach, we model here the behavior of a simple railway control system. The system is composed of a simple railroad on which a train is running. A sensor controls a barrier: when the train reaches the *Sensor1* the barrier starts to go down and is *Closed* after between 8 and 16 seconds. When the train reaches the *Sensor2*, the barrier goes up and is *Open* in between 10 and 20 seconds. After the train has passed the *Sensor1*, it takes between 15 and 20 seconds to *Enter* the level crossing and then between 10 and 15 seconds to reach the *Sensor2*. After that the train takes between 100 and 150 seconds to reach *Sensor1* again and loops.

The simplified ET-LOTOS specification is given by the figure 1. One can notice that only the active elements, i.e. the train and the barrier, are specified. Moreover, the `hide` operator allows to "close" the system from the environment, i.e. to prevent interactions with other processes.

With this specification, the goal may be to verify various properties. For instance, one can verify if the barrier is always *Closed* before the train *Enter* the level-crossing.

4 Transformation of ET-LOTOS

In this section, we briefly describe our ET-LOTOS versus (extended) timed automata translator. This construction is compositional. First, the basic ET-LOTOS behaviors are translated into their equivalent timed-automata. After, the construction of the timed-automata corresponding to behaviors composed through some ET-LOTOS operator, are defined from the timed-automata corresponding to the operands. Definition of our extended timed-automata, details of this translation and proof that

[1] We suppose the reader as some knowledges on LOTOS

specification Railway: **noexit**
behavior
  hide `Sensor1`, `Sensor2`, `Enter`, `Closed`, `Open` **in**
    TRAIN |[`Sensor1`, `Sensor2`]| BARRIER
where
  process TRAIN: **noexit** :=
    `Sensor1`{0}; Δ^{15} `i`{5}; `Enter`; Δ^{10} `i`{5};
    `Sensor2`{0}; Δ^{100} `i`{50}; TRAIN
  endproc
  process BARRIER: **noexit** :=
    `Sensor1`; Δ^{8} `i`{6}; `Closed`;
    `Sensor2`; Δ^{10} `i`{10}; Open; BARRIER
  endproc
endspec

Figure 1: ET-LOTOS specification of the manufacturing plant

the provided timed-automata are *equivalent* from the corresponding ET-LOTOS behaviors, may be found in [Her98].

Let us take one small example to illustrate our transformation method. Let us suppose we have built the timed automaton for the ET-LOTOS behavior expression `B`. Then the extended timed automaton resulting from the transformation of the expression Δ^5`a {3}; B` is given by the automaton where the initial state (state 1) has its invariant set to `true` and has a transition to the initial state of the automaton corresponding to the behavior `B` (state 2). One vertex and one transition are added to the automaton. This transition is labeled by the action `a` and guarded by the expression $5 \leq c \leq 8$ representing the composition of the delay and the life reducer restriction. Indeed, the clock `c` being initialized to zero in vertex `1`, the transition cannot occur before 5 time units and can only occur if no more than `8` time units are spent in this vertex. It is not an obligation for the transition to occur within this time limits since the invariant of `1` is true, but if it does not, it will never occur. If the gate `a` was the internal action `i`, the vertex invariant would be replaced by the expression $c \leq 8$ representing the urgency of the internal action.

In practice, particular care have been taken to reduce as much as possible the size of the extended timed automaton produced. Special techniques have been implemented to reuse, when possible, the same automaton for two different instantiations of a same process.

5 Model-checking of ATCTL formulas

We address here the problem of verifying whether a given ET-LOTOS specification satisfies some real-time requirements. The animator can help the user in writing and detecting errors in his specification but cannot formally verify that some needed properties are satisfied. To undertake such verification we have chosen the model-checking approach. In previous work, we have reported the use of KRONOS to check the validity of TCTL formulas on specifications expressed in a subset of ET-LOTOS. This verification is possible through the use of the tool we have developed to transform this ET-LOTOS subset into the KRONOS timed automaton model.

The real-time logic TCTL [ACD90] is an extension of CTL [EC82] where two operators have been extended with timing constraints to allow quantitative temporal reasoning. The TCTL formulas accepted by KRONOS are defined by the following grammar:

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

where r is a positive integer, $\sim \in \{<, \leq, =, \geq, <\}, c$ is a clock of the timed automaton under verification, *init* represents the initial state with all the clocks set to zero and $enable(a)$ defines the states where transitions labeled with a are enabled. Intuitively, $\rho_1\exists\mathcal{U}_{\sim r}\rho_2$ means that it exists a run which continuously verifies ρ_1 until a state which verifies ρ_2 is reached at a time $t \sim r$. In the same way, $\rho_1\forall\mathcal{U}_{\sim r}\rho_2$ means that all runs satisfy the above property. Some typical abbreviations are used such as $\forall\Diamond_{\sim r}\rho$ for $true\forall\mathcal{U}_{\sim r}\rho$, $\exists\Diamond_{\sim r}\rho$ for $true\exists\mathcal{U}_{\sim r}\rho$, $\exists\Box_{\sim r}\rho$ for $\neg\forall\Diamond_{\sim r}\neg\rho$ and $\forall\Box_{\sim r}\rho$ for $\neg\exists\Diamond_{\sim r}\neg\rho$. The unrestricted operators correspond to the operator subscripted by ≥ 0. This logic allows to specify complex safety and liveness properties.

Unfortunately, if TCTL is adapted to express real-time properties on timed automata, where formula are state formula built using the propositions associated to these states, it is not adapted to express properties on process algebra where actions are the key elements. Indeed, `enable(a)`, the only kind of formula expressing something on actions, is a state proposition true if `a` is possible, but it does not express the fact that the action `a` *will* actually be fired.

Therefore, we have defined ATCTL, an action based extension of TCTL. ATCTL extends TCTL is a similar way than ACTL extends CTL [GL91]. To define the set of action formulas, we first define the set of action formulas by the following:

$$\chi ::= \ a \mid \neg\chi \mid \chi \wedge \chi \ (\texttt{for any action } a)$$

The ATCTL formulas are then defined by the following grammar

$$\rho ::= true \mid x \sim c \mid x - y \sim c_2 \mid \neg\rho \mid \rho^1 \ \wedge \ \rho^2 \mid x.\rho \mid$$

$$\rho^1 \ {}_{\chi}\exists\mathcal{U}_{\chi'} \ \rho^2 \mid \rho^1 \ {}_{\chi}\exists\mathcal{U} \ \rho^2 \mid \rho^1 \ {}_{\chi}\forall\mathcal{U}_{\chi'} \ \rho^2 \mid \rho^1 \ {}_{\chi}\forall\mathcal{U} \ \rho^2$$

where x, y are clocks, c is a positive integer, c_2 is an integer and $\sim \in \{<, \leq\}$.

Intuitively, $x.\rho$ resets the clock x in the formula ρ; this allows to measure the time spent from the current state. $\rho^1 \ {}_{\chi}\exists\mathcal{U}_{\chi'} \ \rho^2$ means that it exists a run where ρ^1 is continuously true and every transition satisfies χ or is an internal action, until a transition which satisfies χ' leading to some state where ρ^2 is verified. $\rho^1 \ {}_{\chi}\forall\mathcal{U}_{\chi'} \ \rho^2 \mid$ is similar but for every runs instead of one. $\rho^1 \ {}_{\chi}\exists\mathcal{U} \ \rho^2 \mid$ and $\rho^1 \ {}_{\chi}\forall\mathcal{U} \ \rho^2$ are defined to allow formulas where a condition may become `true` not necessarily after a visible action.

The idea to verify that an ATCTL formula ϕ is satisfied in the timed automaton M ($M \models \phi$),is similar to the procedure for ACTL. A translation of the ATCTL formula ϕ into a TCTL one $f(\phi)$ is done. Moreover, M is translated into M' as follows: each transition in M from state s to state s' labeled by an action `a` is replaced by a sequence of two transitions linked by a new vertex n. The first transition goes from s to n and resets a special clock t. The vertex n has a proposition `a` and its invariant contains $t = 0$. The second transition goes from n to s'. Moreover, a proposition $\perp$ is attached to all the "normal" vertices to distinguish original vertices from the added ones. $f(\phi)$ is defined as follows:

$f(true) = \perp \wedge true$
$f(x \sim c) = \perp \wedge x \sim c$
$f(x - y \sim c) = \perp \wedge x - y \sim c$
$f(\neg\phi) = \perp \wedge \neg f(\phi)$
$f(x.\phi) = \perp \wedge x.f(\phi)$
$f(\phi^1 \wedge \phi^2) = f(\phi^1) \wedge f(\phi^2)$
$f(\phi^1 \ {}_{\chi}\exists\mathcal{U}_{\chi'} \ \phi^2) = \perp \wedge((\neg(\neg\chi \wedge \neg(f(\phi^1) \wedge \perp)))\exists\mathcal{U}(\chi' \wedge (\chi'\exists\mathcal{U}f(\phi^2))))$
$f(\phi^1 \ {}_{\chi}\exists\mathcal{U} \ \phi^2) = \perp \wedge((\neg(\neg\chi \wedge \neg(f(\phi^1) \wedge \perp)))\exists\mathcal{U}(\perp \wedge f(\phi^2)))$
$f(\phi^1 \ {}_{\chi}\forall\mathcal{U}_{\chi'} \ \phi^2) = \perp \wedge((\neg(\neg\chi \wedge \neg(f(\phi^1) \wedge \perp)))\forall\mathcal{U}(\chi' \wedge (\chi'\exists\mathcal{U}f(\phi^2))))$
$f(\phi^1 \ {}_{\chi}\forall\mathcal{U} \ \phi^2) = \perp \wedge((\neg(\neg\chi \wedge \neg(f(\phi^1) \wedge \perp)))\exists\mathcal{U}(\perp \wedge f(\phi^2)))$

$M \models \phi$ is then equivalent to $M' \models f(\phi)$.

We use the following notations: $a\exists\mathcal{U}b$ stands for $true_a\exists\mathcal{U}_b true$ and ${}_{.}\exists\mathcal{U}_b$ denotes ${}_{\neg void}\exists\mathcal{U}_b$ where *void* is an action not used by the timed automaton.

Using our tool, it is now possible to check ATCTL formulas on timed automata. E.g. we have checked that the train cannot `enter` the level-crossing before the barrier is `closed`, formally:

$$not(true_{.}\exists\mathcal{U}_{sensor1}((not\ closed)\exists\mathcal{U}enter))$$

Timed formula may also be verified, e.g. the train cannot `enter` the level-crossing after less than1 second after it has been `closed`

6 Conclusions

We have defined a new action based real-time logic. ATCTL is based on an extension of ACTL and TCTL. This logic is well adapted to verify properties expressed in the real-time process algebra ET-LOTOS. The verification uses a transformation of ET-LOTOS expressions into a timed automaton model.

The model checking of ATCTL formulas on our timed automata are based on transformation of these time automata which allows to express the formula in an TCTL one.

Moreover, our TCTL model checker suffers of the state explosion problem which requests a big amount of memory space to save the data structure. Various possible tracks could be followed. Abstractions may be done on the model depending on the properties which must be verified. Other models, as timed Petri nets, and verification techniques, as the one based on partial orders, could also be investigated. Nevertheless, our verification technique is a first step in the development of analysis tool for timed process algebras.

References

[ACD90] R. Alur, C. Courcoubetis, and D.L. Dill. Model-checking for real-time systems. In *Proceedings of the 5th Symposium on Logic in Computer Science*, pages 414–425, Philadelphia, June 1990.

[Alu91] R. Alur. *Techniques for automatic verification of real-time systems*. PhD thesis, Departement of Computer Science, Stanford University, 1991.

[CES86] E.M. Clarke, E. A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transaction on Programming Languages and Systems*, 8(2):1512–1542, 1986.

[dJHM98] A. de Jacquier, C. Hernalsteen, and T. Massart. A simulator of real-time systems specified in ET-LOTOS. In *Advanced Simulation Technologies Conference*, April 1998.

[DOY94] C. Daws, A. Olivero, and S. Yovine. Verifying ET-LOTOS programs with KRONOS. In *Seventh International Conference on Formal Description Techniques*, pages 227–242. Chapman & Hall, 1994.

[EC82] E.A. Emerson and E. Clarke. Using branching time temporal logic to synthesize synchronization skeletons. *Science of Computer Programming*, 2:241–266, 1982.

[FGK+96] Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. Cadp (cæsar/aldebaran development package): A protocol validation and verification toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Sciences*, pages 437–440. Springer-Verlag, August 1996.

[GL91] O. Grumberg and D. Long. Model-checking and modular verification. In J. Baeten and J.F. Groote, editors, *CONCUR'91*, volume 527 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[Her97] C. Hernalsteen. A timed automaton model for ET-LOTOS verification. In T. Higashino adn A. Togashi T. Mizuno, N. Shiratori, editor, *Formal Description Techniques and Protocol Specification, Testing and Verification*. Chapman & Hall, 1997.

[Her98] C. Hernalsteen. *Specification, validation and verification of real-time systems in ET-LOTOS*. PhD thesis, Departement of Computer Science, University of Brussels (ULB), 1998.

[HM95] C. Hernalsteen and T. Massart. An extended timed automaton to model ET-LOTOS Specifications. In *Proceedings of the first workshop on Design and Analysis of Real-Time Systems*, pages 95–112, 1995.

[Hoa85] CAR. Hoare. *Communicating Sequential Processes*. PHI Series in Computer Science. Prentice Hall, 1985.

[ISO88] ISO/IEC. Lotos — a formal description technique based on the temporal ordering of observational behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, September 1988.

[LL94] Luc Léonard and Guy Leduc. An enhanced version of timed LOTOS and its application to a case study. In *FORTE-VI*, pages 483–500. North-Holland, 1994.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1+2):134–152, 1997.

[Mil89] R. Milner. *Communication and Concurrency*. PHI Series in Computer Science. Prentice Hall, 1989.

[T3.90] T3.2. Deliverable D3.2S/1P part III: LOTOS specifications. Technical report, LOTOSPHERE Project, ESPRIT 2304, European Commission, Brussels, Belgium, April 1990.

[Yov93] S. Yovine. *Méthodes et outils pour la vérification symbolique de systèmes temporisés*. PhD thesis, Institut National Polytechnique de Grenoble, France, 1993.

[Yov97] S. Yovine. Kronos: a verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1+2):123–133, 1997.