Université de Liège

Faculté des Sciences Appliquées

Systèmes et Automatique

Institut d'Electricité Montefiore, B28
Université de Liège au Sart Tilman
B-4000 Liège 1 (Belgique)

A timed LOTOS supporting a dense time domain and including new timed operators

Guy Leduc,[1]
Luc Léonard[2]

[1] Research Associate of the F.N.R.S.
(National Fund for Scientific Research, Belgium)
[2] Research Assistant of the F.N.R.S.

Published in:
Proceedings of FORTE'92, Perros-Guirec, France, 13-16 octobre 1992
M. Diaz & R. Groz (Ed.),
Elsevier Science Publishers BV (North Holland), 1993
pages 87-102

Date : October 1992
Status : Public
RS 93-01

A timed LOTOS supporting a dense time domain and including new timed operators

Guy Leduc and Luc Léonard

Research Associate and Research Assistant of the
National Fund for Scientific Research (Belgium)
Université de Liège, Institut d'Electricité Montefiore, B 28, B-4000 Liège 1, Belgium
Tel: + 32 41 562691 Fax: + 32 41 562989 E-mail: leduc@montefiore.ulg.ac.be

Abstract

A time extended version of LOTOS, denoted Timed LOTOS, is proposed for the modelling of quantitative timed behaviours. In this language neither the syntax nor the semantics are restricted to a specific time domain, i.e. a dense time domain is supported as well. Timed LOTOS incorporates a notion of urgency which is restricted to the internal actions. This is usually referred to as the maximal progress or minimum delay property. Timed LOTOS processes have also some pleasing properties such as the deadlock freeness property (i.e. processes can never stop the progression of time), and the persistency property (i.e. by idling, a process will not lose any capability of performing an action).
In Timed LOTOS the delay operator is powerful because it allows the specification of a time interval in which the delay is nonderministically chosen. Two other powerful timed operators are defined which allow the expression of timed constraints on **interactions**, i.e. on actions involving several processes. The first one introduces a delay before the execution of any instance of a given action in a process. A second operator allows to start a time-out on any instance of a given action in a process, and to activate another process when such a time-out expires. The originality of these two operators is that they constrain interactions between processes, and support adequately a structured specification style.

Keyword Codes: F.4.3; D.3.1; D.3.3
Keywords: Mathematical Logic and Formal Languages, Formal Languages; Programming Languages, Formal Definitions and Theory; Language Constructs and Features

1. Introduction

In recent years, many quantitative timed extensions of well-known asynchronous process algebras have been proposed, as well as new timed process algebras. CSP [Hoa 85], CCS [Mil 89], ACP [BeK 85], LOTOS [ISO 8807, BoB 87] have been extended and new process algebras have been proposed which are intended to model time in a quantitative way. A non exhaustive list of them are referred to by the following acronyms in the sequel: ACP_ρ [BaB 90], $ACP_{\tau\varepsilon}^{t}$ [Gro 90a], ATP (Algebra of Timed Processes) [NRS 90, NiS 91b, NSY 91], LOTOS-T [MFV 92], PADS (Process Algebra for Distributed Systems) [Azc 90], TCCS (Temporal CCS) [MoT 90], TIC [QAF 89], Timed CCS [Wan 90, Wan 91], Timed CSP [ReR 88, DaS 89, Ree 90], Timed-interaction LOTOS [BLT 90], TLOTOS [Led 92], T-LOTOS and U-LOTOS [BoL 92], TPCCS (Timed Probabilistic CCS) [HaJ 90, Han 91], TPL (Temporal Process Language) [HeR 90]. An overview and synthesis on Timed Process Algebras may be found in [NiS 91a].

The time extended version of LOTOS that we propose in this paper for the modelling of quantitative timed behaviours has been inspired by some of the above-mentioned languages, and will be compared to them as often as possible in the sequel.

A first important characteristic of Timed LOTOS is that neither its syntax nor its semantics are restricted to a specific time domain. In particular, Timed LOTOS supports a discrete time do-

main such as the natural numbers, as well as a dense time domain such as e.g. the rational numbers. The generic time domain is presented in section 3.
A second important characteristic is the notion of urgency that we have included in the language. We have decided not to allow action urgency, that means that it is impossible to enforce the occurrence of an observable action at a specific instant or before a certain time limit. Instead, Timed LOTOS incorporates a notion of urgency on internal actions which do not involve the environment. This is usually referred to as the maximal progress or minimum delay property, also existing in e.g. TPL, Timed CCS and Timed CSP. These points are discussed in depth and justified in section 2.

The operational semantics of the classical LOTOS operators in the new framework is presented in section 4. Very few timed operators are included in the language. The first operator, presented in section 5, is a delay operator, denoted $\Delta^{[d_1,d_2]}\ P$, which is used to delay the execution of *P*. The power and originality of this operator is that it allows the specification of an interval $[d_1,d_2]$ in which the delay is nondeterministically chosen.

In section 6, two other powerful operators are defined, which allow the expression of timed constraints on **interactions**[1], but without any urgency constraint. The first one, denoted $\Delta_g^I\ P$, introduces a nondeterministic delay from the time interval *I* to every action *g* in process *P* (i.e. including interactions between subprocesses in *P*). The second operator, denoted $\lfloor P \rfloor_g^d(Q)$, allows the activation of a time-out of value *d* on every action *g* in process *P*, and the activation of process *Q* when such a time-out expires. The originality of these two operators is that they can constrain **interactions** between processes, and thereby support adequately a structured specification style, such as the constraint-oriented style in LOTOS [Bri 89].

In order to define these operators a special "elapsed time" action, denoted θ, has been introduced in the semantics, i.e. θ, like δ, can never appear in the Timed LOTOS syntax (e.g. in an action-prefix construct, in the list of gates of the parallel composition and hiding operators). Its features and the reasons for its introduction are explained in section 5.

2. Modelling urgency

In an untimed process algebra, the absence of any time concept makes it impossible to express an urgency concept such as the requirement that actions have to occur at some instant or as soon as they are enabled. Indeed, when no quantitative notion of time exist, what do we mean by as soon as possible ?

In timed process algebras, an opportunity is given to express urgency in the model, simply by requiring that a given action occur immediately or within some precise time limits, or after a certain delay.

Whereas enforcing a process to wait a certain amount of time is quite simple and intuitive, enforcing that an action occur before a certain time limit, or that an interaction between several processes occur as soon as enabled is more complex and may create some counter-intuitive results when combined with the process algebraic paradigm.

Before justifying our choice in this matter, let us briefly recall some basic notations and concepts. Let *L* be the classical alphabet of actions, and *i* the internal action. Let $L^i = L \cup \{i\}$.[2]

$P \xrightarrow{a} P'$ where $a \in L^i$, means that process *P* may engage in action *a* and, after doing so, behave like process *P'*.

$P \xrightarrow{a}$ means $\exists$ P' such that $P \xrightarrow{a} P'$

$P \not\xrightarrow{a}$ where $a \in L^i$, means that $\nexists$ P', such that $P \xrightarrow{a} P'$, i.e. *P* cannot accept (or must refuse) the action *a*.

---

[1] An interaction is an action which involves several processes. This syntactic distinction will be further explained and justified later in the paper.

[2] We do not consider actions δ and θ in this section.

$P \xrightarrow{t} P'$ where $t \notin L^i$, means that $\exists$ P' such that process *P* may idle (i.e. not execute any action in L^i) during a period of *t* units of time and, after doing so, behave like process *P'*. The domain of *t* will be presented later.

$P \xrightarrow{t} \!\!\!\!/$ where $t \notin L^i$, means that $\nexists$ P', such that $P \xrightarrow{t} P'$, i.e. *P* cannot idle during a period of *t* units of time.

In order to review the existing approaches to model urgency, we propose to classify LOTOS actions according to two orthogonal criteria: observability and number of parallel processes involved.

Observability

Any action is observable or external except *i*.

Number of parallel processes involved

Here, among the actions, we distinguish between simple actions and interactions. The distinction is only relevant at a syntactical level: an interaction involves more than one process, i.e. it requires at least the participation of two parallel processes of the system (the environment not being considered).

Example 1: consider A := b; stop and B := a; b; stop |[b]| c; b; stop.
In A, b is an observable simple action, whereas in B, b is an observable interaction.
Example 2: consider A := i; stop and B := hide b in (a; b; stop |[b]| c; b; stop)
In A, i is an internal simple action, whereas in B, i is an internal interaction.

In other words, we will look at the modelling of urgency in two stages: first the sequential processes (urgency on simple actions), and then the parallel processes (urgency on interactions).

Even if there is clearly no difference between a simple action and an interaction in the underlying semantic model, such a distinction will turn out to be very useful in the sequel to explain the necessity of certain new timed operators. Note that we keep the term "action" to refer collectively to simple actions and interactions in the sequel.

Let us already stress that, depending on the model, an interaction is necessarily internal or not.

For instance, in CCS-like languages, interactions are always τ actions, and consequently only two processes may interact. In LOTOS-like or ACP-like languages, an interaction remains observable, which allows more than two processes to interact. This distinction will be important as modelling urgency on interactions will be strongly related to this characteristic.

A model may express *action urgency* iff $\exists\ a \in L^i, P$ such that $(P \xrightarrow{a} \wedge \forall t \neq 0,\ P \xrightarrow{t} \!\!\!\!/)$

Informally, this means that it is possible to write down a process which may only execute an action immediately, that is without letting time pass.

When this definition is focused uniformly on internal actions, this property is called *minimum delay* or *maximal progress*.

A model has the *maximal progress property* iff $\forall\ P,\ (P \xrightarrow{i} \Rightarrow \forall t \neq 0,\ P \xrightarrow{t} \!\!\!\!/)$

This property requires that all the operators be carefully defined; in particular the action-prefix but also, depending on the model, choice, parallel composition or hiding.

Simple action urgency in sequential processes

The ability to model action urgency on simple actions is usually introduced in a language via a kind of action-prefix with an urgency semantics, that is: $\forall\ a \in L^i, \forall\ t \neq 0,\ a;P \xrightarrow{t} \!\!\!\!/$

An additional construct is also often provided to disable urgency (e.g. a δ action in TCCS, or a $\lfloor\rfloor^{\omega}$ operator in ATP).

The dual approach may also be followed where action-prefix has no urgency semantics (i.e. a; P $\xrightarrow{t}$ a; P) but an additional operator (e.g. *asap* in U-LOTOS, *timer* in T-LOTOS) is proposed to enforce urgency on some actions.

In models where only internal actions are urgent, the first approach is usually adopted.

Existing timed process algebras may be classified according to the way they have incorporated action urgency on simple actions as follows:

- Urgent action-prefix for all actions in e.g. ATP, TCCS, $ACP_{\tau\varepsilon}^{t}$, TLOTOS.
- Urgent action-prefix for i but not for other actions in e.g. TPL, Timed CCS, TPCCS.
- Non urgent action-prefix for all actions, but operator to introduce urgency in e.g. U-LOTOS, T-LOTOS.

Interaction urgency in parallel processes

Now that urgency has been introduced on sequential processes, let us investigate the new problems which generally appear when dealing with several parallel processes which have to synchronise urgently, that is as soon as all processes are ready to do so. Not all the languages may express this form of urgency even if they incorporate the ability to express urgency on simple actions as above. This fact has been first explained by T. Bolognesi in [BLT 90, BoL 92].

As pointed out earlier, the way urgency on interactions is handled strongly depends on the nature of the model. Let us consider successively CCS-like models and LOTOS-like models on the following simple LOTOS example from [BLT 90].

Let *A := d_1; x; stop* and *B := d_2; x; stop*.

A |[x]| B may be expanded as *d_1; d_2; x; stop [] d_2; d_1; x; stop*, that is both actions d_1 and d_2 must occur (in any order) before *x* can occur.

In a timed LOTOS, it would be very powerful to be able to express in a similar modular way this kind of constraint, but with an additional requirement which is "*x* must occur *as soon as* both d_1 and d_2 have occurred". It is obvious that this urgency cannot be expressed in process *A* alone, nor in *B* alone, nor even in both *A* and *B*. This urgency must necessarily be introduced at a non local level. There are two existing methods to achieve this goal.

In CCS-like languages which have the maximal progress property, the τ action resulting from the synchronisation on *x* is urgent, and thereby implies the urgency on the interaction. Therefore, we can conclude that in such models, the maximal progress property suffices to obtain the required urgency on interactions. This is the case e.g. in TPL, Timed CCS and TPCCS. Of course, urgency on observable interactions has no meaning in these models.

In LOTOS-like models, it is possible to distinguish between urgency on hidden interactions and on observable interactions.

a) Hidden interactions

In such languages it is necessary to hide the interaction explicitly, e.g.

hide x in (A |[x]| B) which may be expanded as *d_1; d_2; i; stop [] d_2; d_1; i; stop*.

Again, if the language has the maximal progress property, urgency on the interaction is obtained as in CCS-like languages. This is the case in e.g. LOTOS-T.

b) Observable interactions

To impose urgency on x but not on d_1 and d_2 in *A |[x]| B*, the only way is to introduce a specific operator, such as the asap operator of U-LOTOS.

The process is then written *asap x in (A |[x]| B)*.

This operator defines the scope of the urgency requirement on *x*. This is needed because more than two processes might be involved: *(asap x in (A |[x]| B)) |[x]| C* is different from *asap x in (A |[x]| B |[x]| C)*. In particular, *asap x in (A |[x]| B)* is different from *(asap x in A) |[x]| (asap x in B)*. This is precisely the power of the asap construct which allows a local as well as a non-local expression of urgency.

Existing timed process algebras may be classified according to the way they handle interaction urgency as follows:

- On internal interactions via maximal progress in TPL, Timed CCS, TPCCS, LOTOS-T, Timed CSP (where the internal action is implicit).
- On observable interactions via an asap operator in U-LOTOS

Some models do not provide such facility: TCCS, TIC, ATP, $ACP_{\tau\varepsilon}^{t}$, TLOTOS.

In this paper and w.r.t. this classification on urgency, we will present an extension of LOTOS which is in the line of TPL, Timed CSP, Timed CCS and TPCCS, i.e. with the maximal progress property, but where no time constraint can be imposed on observable actions.

We think that urgency on observable actions is not needed, but that, on the other hand, some constructs should be added to the language to specify delays or timers on interactions. This will be explained in section 6. We justify our choices as follows. Internal or hidden actions are autonomous (i.e. under the control of the system only) and therefore can be time-constrained at will. On the other hand, a system cannot enforce its environment to interact with it at a given instant. Therefore, observable actions, which rely on the willingness of an environment cannot be forced to occur at (or before) a given instant, or within a bounded time interval.

When extending this power to external actions such as in ATP, $ACP_{\tau\varepsilon}^{t}$, TCCS, PADS, TIC, LOTOS-T, TLOTOS or U-LOTOS, the model allows the blocking of time when such an action cannot occur. This is mathematically sound and needed to preserve the urgency semantics, but certainly very counter-intuitive. In TLOTOS, we have ourselves followed this idea, but we now believe that another possibility turns out to be more intuitive and sufficiently flexible in practice. In this paper, the Timed LOTOS will have the property that no process will ever be able to block time.

3. Time domain

Many proposed languages (e.g. TPL, PADS, ATP, TPCCS, TLOTOS) are based on a discrete time domain. Informally this means that time instants are expressed by natural numbers, and time only progresses at such time instants. Time progresses consistently in all parts of the system. More formally[1], a discrete time domain is a time domain[2] which has the following property: $\forall\, d\ \exists\, d'$ such that $d < d' \wedge \forall\, d'': d < d'' \Rightarrow d' \leq d''$.

When using a discrete time domain, it is necessary to define carefully the smallest grain of time of the specified system, in order to be able to express any timed constraint w.r.t. this basic unit or reference. In practice, such a grain of time is very likely to exist, but expressing all time durations or time instants in this unit will also very likely turn out to be inconvenient. This phenomenon becomes really critical in a practical design where this grain of time is tightly bound to the abstraction level of the specification. Therefore, when a process is refined, this may require the selection of a smaller grain of time, and the whole specification needs to be rewritten for consistency. Consequently, we think that in practice a discrete time domain may be inadequate. In this paper, neither the syntax nor the semantics are restricted to a discrete time domain. Dense time domains are supported as well.

Formally, a dense time domain is a time domain which has the following property:

$$\forall\, d, d': d < d' \Rightarrow \exists\, d'' \text{ such that } d < d'' < d'.$$

Informally, in a dense time domain, time instants are expressed by rational or real numbers. With such a time domain the two shortcomings of a discrete time domain are overcome.

---

[1] The formal definitions of this section 3 are reproduced from [NiS 91a].

[2] Formally, a time domain is a commutative monoid (D, +, 0) satisfying the two requirements:

(i) $d + d' = d \Leftrightarrow d' = 0$

(ii) the preorder $\leq$ defined by $d \leq d' \Leftrightarrow \exists\, d''$ *such that* $d + d'' = d'$ is a total preorder satisfying the property that any descending sequence $d_0 > d_1 > \ldots d_i > d_{i+1} \ldots$, *such that* $\exists d\ \forall i: d_i \geq d_{i+1} + d$, is finite.

In the sequel we will restrict the time domain to a countable one (e.g. the rational numbers). This will allow us to preserve the LTS model as an underlying semantic model. It will be denoted D. D^{ω} denotes $D \cup \{\omega\}$ where ω is an element not in D such that $\forall\, d \in D, d < \omega$, and $\omega + d = \omega$. To denote the possibility of a system to make progress in time, i.e. to idle, we use the notation:

$P \xrightarrow{t} P'$ where $t \in D^{\omega}$, means that process P may idle during t units of time and, after doing so, behave like process P'.

4. Basic untimed operators of Timed LOTOS

In this section the axioms and inference rules for the classical LOTOS operators will be given which will define the semantics of Timed LOTOS.

Notations

L is the alphabet of observable actions. i, δ and θ are special actions, respectively the internal action, the termination action and the elapsed time action. θ should not be confused with a tick action often found in discrete timed process algebras.

$L^{\delta} = L \cup \{\delta\}$, $L^{\theta} = L \cup \{\theta\}$, $L^{i,\delta} = L \cup \{i,\delta\}$, $L^{i,\delta,\theta} = L \cup \{i,\delta,\theta\}$, …

θ is an invisible action which is, like δ, only present in the semantic model, i.e. θ can never appear in an action-prefix construct, nor in the list of gates of the parallel composition and hiding operators. The introduction of a second internal action may be questioned. However, its existence is justified because θ will not have the same properties as i. By contrast to the i action, θ is not always urgent. This allows the modelling of a non-deterministic occurrence of θ (an elapsed time event) within a time interval, because θ will not have any "priority" over time, as i has. In addition, θ will not resolve a choice. In this last sense, θ is closer to the implicit "internal action" of CSP.

The main justification for the existence of θ and its main interest is that it will allow us to give Timed LOTOS the persistency property, i.e. by idling a process will not lose any capability of performing an action. Formally:

$$\forall\ P,P',\forall\ t \in\ D^{\omega}, \forall\ a \in\ L^{i,\delta,\theta}, (P \xrightarrow{t} P' \wedge P \xrightarrow{a} \Rightarrow P' \xrightarrow{a})$$

θ will also induce a dual property, that we will call reverse persistency, which states that by idling a process will not get any additional capability of performing an action (except possibly θ). Formally:

$$\forall\ P,P',\forall\ t \in\ D^{\omega}, \forall\ a \in\ L^{i,\delta}, (P \xrightarrow{t} P' \wedge P \not\xrightarrow{a} \Rightarrow P' \not\xrightarrow{a})$$

These last two properties will be of paramount importance for the definition of more complex timed operators such as those introduced in section 6.

The axioms and inference rules will be presented in two columns, as first proposed for TCCS. In the first one are presented the usual LOTOS axioms and inference rules extended to cope with the θ action. In the second column, axioms and inference rules of another type are added to give the semantics of the operators w.r.t. the possible evolution in time. An important advantage of this approach is that no rule is based on a premiss containing an impossibility to evolve in time (i.e. a negative premiss on a t transition).

Table 2 presents the axioms and inference rules for the basic LOTOS operators. We give hereafter some explanations of the new axioms and rules, i.e. those generating θ and t transitions.

The **S axiom** expresses that the *stop* process allows time to pass: *stop* is an idling process. This approach is consistent with the view that no process may ever block time. The **AP2 axiom** expresses the non urgency semantics of action-prefix: action a may occur now or at any later time. Let us remark that, in AP2, a is restricted to L, thus excluding i. Therefore, $i;P$ can-

not let time pass and *i* must occur immediately, which is needed to get the maximal progress property. The **Ex2 axiom** expresses the non urgency of termination. This is consistent with the fact that *exit* is nothing else that a special form of action-prefix with δ. The **Ch2 rule** expresses that θ is different from any other action in the alphabet, because it does not resolve a choice. The **Ch3 rule** expresses that time does not resolve a choice: choice is deterministic w.r.t. time. The **PC3 rule** expresses that time passes at the same pace in both concurrent processes. The **H3 rule** expresses that when observable actions are hidden, the resulting internal actions *i* are necessarily urgent, i.e. no timed arc may leave a state where an internal action appears. The **En3 rule** expresses that the termination of a process is urgent when there is a subsequent process *Q*, i.e. the internal action generated when control passes from *P* to *Q* is urgent. This is consistent with the fact that enabling is a special form of combination of parallel composition and hiding. The **Di4 rule** expresses that θ is different from any other action in the alphabet, because a θ in *Q* does not disable *P*. This is consistent w.r.t. the semantics of the choice operator. The **Di5 rule** expresses that time does not resolve a disabling: disabling is deterministic w.r.t. time. This is again consistent w.r.t. the semantics of the choice operator. The **In2 rule** is the classical one extended to timed transitions.

	(S) $stop \xrightarrow{t} stop \quad (t \in D^{\omega})$
(AP1) $a; P \xrightarrow{a} P \quad (a \in L^{i})$	(AP2) $a; P \xrightarrow{t} a; P \quad (a \in L, t \in D^{\omega})$
(Ex1) $exit \xrightarrow{\delta} stop$	(Ex2) $exit \xrightarrow{t} exit \quad (t \in D^{\omega})$
(Ch1) $\dfrac{P \xrightarrow{a} P'}{P \,[]\, Q \xrightarrow{a} P'} (a \in L^{i,\delta})$ (+ Ch1') (Ch2) $\dfrac{P \xrightarrow{\theta} P'}{P \,[]\, Q \xrightarrow{\theta} P' \,[]\, Q}$ (+ Ch2')	(Ch3) $\dfrac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P \,[]\, Q \xrightarrow{t} P' \,[]\, Q'} (t \in D^{\omega})$
(PC1) $\dfrac{P \xrightarrow{a} P'}{P \,\lvert[\Gamma]\rvert\, Q \xrightarrow{a} P' \,\lvert[\Gamma]\rvert\, Q} (a \in L^{i,\theta} - \Gamma)$ (+ PC1') (PC2) $\dfrac{P \xrightarrow{a} P', Q \xrightarrow{a} Q'}{P \,\lvert[\Gamma]\rvert\, Q \xrightarrow{a} P' \,\lvert[\Gamma]\rvert\, Q'} (a \in \Gamma \cup \{\delta\})$	(PC3) $\dfrac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P \,\lvert[\Gamma]\rvert\, Q \xrightarrow{t} P' \,\lvert[\Gamma]\rvert\, Q'} (t \in D^{\omega})$
(H1) $\dfrac{P \xrightarrow{a} P'}{hide\ \Gamma\ in\ P \xrightarrow{a} hide\ \Gamma\ in\ P'} (a \in L^{i,\delta,\theta} - \Gamma)$ (H2) $\dfrac{P \xrightarrow{a} P'}{hide\ \Gamma\ in\ P \xrightarrow{i} hide\ \Gamma\ in\ P'} (a \in \Gamma)$	(H3) $\dfrac{P \xrightarrow{t} P', P \not\xrightarrow{a} \forall a \in \Gamma}{hide\ \Gamma\ in\ P \xrightarrow{t} hide\ \Gamma\ in\ P'} (t \in D^{\omega})$
(En1) $\dfrac{P \xrightarrow{a} P'}{P >> Q \xrightarrow{a} P' >> Q} (a \in L^{i,\theta})$ (En2) $\dfrac{P \xrightarrow{\delta} P'}{P >> Q \xrightarrow{i} Q}$	(En3) $\dfrac{P \xrightarrow{t} P', P \not\xrightarrow{\delta}}{P >> Q \xrightarrow{t} P' >> Q} (t \in D^{\omega})$
(Di1) $\dfrac{P \xrightarrow{a} P'}{P \,[>\, Q \xrightarrow{a} P' \,[>\, Q} (a \in L^{i,\theta})$	(Di5) $\dfrac{P \xrightarrow{t} P', Q \xrightarrow{t} Q'}{P \,[>\, Q \xrightarrow{t} P' \,[>\, Q'} (t \in D^{\omega})$

(Di2) $\dfrac{Q \xrightarrow{a} Q'}{P\ [>\ Q \xrightarrow{a} Q'}\ (a \in L^{i,\delta})$	
(Di3) $\dfrac{P \xrightarrow{\delta} P'}{P\ [>\ Q \xrightarrow{\delta} P'}$	
(Di4) $\dfrac{Q \xrightarrow{\theta} Q'}{P\ [>\ Q \xrightarrow{\theta} P\ [>\ Q'}$	
(In1) $\dfrac{P[g_1/h_1,\ldots g_n/h_n] \xrightarrow{a} P', Q[h_1,\ldots h_n]:=P}{Q\ [g_1,\ \ldots g_n] \xrightarrow{a} P'}\ (a \in L^{i,\delta,\theta})$	(In2) $\dfrac{P[g_1/h_1,\ldots g_n/h_n] \xrightarrow{t} P', Q[h_1,\ldots h_n]:=P}{Q\ [g_1,\ldots g_n] \xrightarrow{t} P'}\ (t \in D^{\omega})$

Table 2: Operational semantics of the basic LOTOS operators

Note that the LTS generated by the rules is necessarily infinite states and infinitely branching when *D* is a countable dense time domain, due to the second type of inference rules. We will comment on that in the conclusion of the paper.

At this stage, the main differences w.r.t. other close approaches are as follows.

Our options	Similar to ours	Different from ours
Syntax and semantics independent from the time domain, no tick action	TCCS, Timed CCS, Timed CSP, LOTOS-T	σ in TPL, χ in ATP, TPCCS and TLOTOS, *t* in $ACP_{\tau\varepsilon}^{t}$ and PADS, *(t)* in T-LOTOS and U-LOTOS
Deadlock freeness	TPL, Timed CCS, TPCCS, Timed CSP	TCCS, $ACP_{\tau\varepsilon}^{t}$, PADS, ATP, TIC, LOTOS-T, T-LOTOS, U-LOTOS, TLOTOS
Urgency on i and only i	TPL, Timed CCS, Timed CSP, TPCCS	General action urgency in: ATP, TCCS, TIC, $ACP_{\tau\varepsilon}^{t}$, T-LOTOS, U-LOTOS, LOTOS-T, TLOTOS
Persistency	Timed CCS (but rule PC3 had to be restricted)	ATP, TPCCS, TLOTOS
Urgency on i technically introduced via hiding	Timed CSP	Via the parallel operator in TPL, Timed CCS, TPCCS. Via hiding but also choice, parallel composition and disabling in LOTOS-T
θ action to model elapsed time		All others

5. The delay operator

In this section, we extend the language presented in section 4 with one basic timed operator, denoted $\Delta^{[d_1,d_2]}\ P$, which allows the introduction of a delay before the execution of process *P*. The particularity of this operator is that this delay has not a fixed value, but instead may have any value from the time interval $[d_1,d_2]$.

A very important characteristic of this operator is that it generates a θ action when it expires. Therefore any change in the action offers due to the expiration of a delay is preceded by a θ occurrence. This occurrence is always possible because θ is, like *i*, an internal action. This characteristic induces naturally the persistency property, which will be necessary in the next section.

It may be observed in the D1 rule in table 3 where the θ action cannot be avoided before activating P.

The **D1 rule** expresses that a nondeterministic delay may expire when the lower bound of the interval is 0. The **D2 rule** expresses that, when this lower bound is not 0, time may pass. The **D3 rule** expresses that, when this lower bound is 0, time may also pass provided that the upper bound is not 0.

(D1) $\Delta^{[0,d]} P \xrightarrow{\theta} P \quad (d \in D^{\omega})$	(D2) $\Delta^{[d_1+t,d_2+t]} P \xrightarrow{t} \Delta^{[d_1,d_2]} P \quad (d_1,d_2,t \in D^{\omega})$ (D3) $\Delta^{[0,d+t]} P \xrightarrow{t} \Delta^{[0,d]} P \quad (d,t \in D^{\omega})$

Table 3: Operational semantics of the basic delay operator

The main differences with other works are the new delay operator and the role of the θ action. The nondeterminism associated with the delay is due to the non urgency of θ in general. The urgency on θ is only enforced at the end of the interval, which is possible because θ is internal.

Based on this delay operator, it is possible to define various timed behaviours, such as a time-out or a watchdog, as explained hereafter.

A time-out of value d

P [] $\Delta^{[d,d]}$ i; Q is a process which behaves like *P* provided *P* starts before time *d*, otherwise, after the delay *d* and occurrence of θ, it behaves like *P [] i;Q*. Due to the maximal progress property, this process cannot idle, it must either execute immediately an action of *P* or the internal action followed by *Q*.

P [] $\Delta^{[d,d]}$ i; Q is written $\lfloor P \rfloor^{d}(Q)$ in ATP.

A watchdog of value d

P [> $\Delta^{[d,d]}$ i; Q is a process which behaves like *P* before time *d*, and like *P [> i;Q* afterwards (and after occurrence of θ) provided *P* has not terminated successfully before this time *d* (i.e. executed a δ action). Due to the maximal progress property, process *P [> i;Q* cannot idle, it must either execute immediately an action of *P* or the internal action followed by *Q*.

P [> $\Delta^{[d,d]}$ i; Q is written $\lceil P \rceil^{d}$ (Q) in ATP.

More sophisticated time-out or watchdogs

If the interval [d,d] is replaced by a non punctual interval $[d_1,d_2]$, the two behaviours above are those of a time-out and a watchdog activated nondeterministically in this interval.

Possibly unbounded delay

We allow ω to be specified as the upper bound of a timed interval, which makes it possible to specify a possibly unbounded delay, not depending on the environment. For instance, the processes $\Delta^{[0,\omega]}$ *a;P* and *a;P* have different behaviours even if they have the same sequences of (temporal and non temporal) actions (except θ): when synchronised with another process *Q* through *a* and hiding of *a*, the action *a* in *a;P* will be executed as soon as *Q* is ready to do so, whereas when hiding *a* in $\Delta^{[0,\omega]}$ *a;P* the arbitrary delay remains.

The capability to model a time-out and a watchdog behaviour so simply in Timed LOTOS is due to two main reasons. First, the maximal progress property which is used to model both the expiration of the time-out and the activation of the watchdog. Second, the existence of the LOTOS disabling operator which is necessary in the watchdog case.

Other languages are also based on a single timed operator: e.g. a delay construct in Estelle, *Wait* in Timed CSP, σ in TPL, *(t)* in U-LOTOS. The new feature introduced in this paper is the ca-

pability of introducing a delay that is nondeterministically chosen in a time interval, as done e.g. in Timed PNs. In particular, this allows the description of a temporal nondeterminism on the occurrence of internal actions. This is not possible in other models based on maximal progress, except in Timed CSP. For instance, in other maximal progress models, *i{d1,d2}; B* is equivalent to *i{d1,d1}; B,* whereas in Timed LOTOS, there is a difference between $\Delta^{[d1,d2]}$ *i; B* and $\Delta^{[d1,d1]}$ *i; B*. The fact that Timed CSP has this capability is due to the existence of the nondeterministic choice operator. Such an approach was not followed in Timed LOTOS because it would have required the introduction of a nondeterministic choice operator in addition to the existing choice operator.

6. More powerful operators: delays and timers on interactions

As defined up to this point, Timed LOTOS is equipped to express sequential timed behaviours. In this section, we extend the language with two powerful timed operators, which allow the expression of time constraints on interactions. These operators do not extend the expressive power of Timed LOTOS, but extend its expressive flexibility: without these operators, some timed behaviours could not be expressed in a modular way (i.e. as a parallel composition of subprocesses), but only on the equivalent expanded sequential behaviour. Our two operators could be changed in favour of others, and new operators could be added. As always, there is probably a good balance to be found between the temptation to add as many new operators as needed to increase the flexibility, and the need to keep the language clean and manageable. We think that our operators are interesting because they allow us to express timed behaviours in a structured way, thereby preserving the structured specification styles.

The first one, denoted $\Delta_g^I P$, introduces a nondeterministic delay from the time interval *I* to every action *g* in process *P*.

The delay on interactions

The purpose of this operator is to extend to the interactions the capability given by Δ^I to delay non deterministically the moment when a simple action becomes enabled. Consider the process *P := a; g; P'* . If we want to enforce a delay *d* before the occurrence of the simple action *g*, we simply write *a;* $\Delta^{[d,d]}$*(g; P').* Consider now that *P* is composed of two subprocesses: *P := a; g; P' |[g]| b; g; Q'*. How is it possible to start a delay exactly when the interaction *g* would normally be enabled in both processes? This can only be specified if there is a means to introduce a delay constraint at a global level, i.e. at the level where the parallel composition of *a; g; P'* and *b; g; Q'* is seen as a single process. A similar reasoning has been first presented in [BLT 90] to point out the necessity to introduce an asap operator in Timed-interaction LOTOS for modelling urgency on interactions. This has been discussed already in section 2. In this paper, we follow a similar approach to justify the two operators described in this section. The common characteristic of our $\Delta_g^I P$ and the *asap g in P* from [BLT 90, BoL 92] is precisely that they allow an extension of the modelling facilities, which exist for monolithic processes, to the structured ones.

In Timed LOTOS, the solution to the above problem will be: $\Delta_g^{[d,d]}$ *(a; g; P' |[g]| b; g; Q').*

The Δ_g^I operator will be defined via an intermediate operator, denoted $\Delta_g^{I,d}$, according to the following definition: $\Delta_g^I P := \Delta_g^{I,0} P$. The second argument *d* is used in the semantics to count the time elapsed since the enabling of *g*: it is referred to as the counter hereafter. The special value ω is assigned to it when the delay *d* has elapsed. The operational semantics is given in table 4.

The **Dg1 rule** expresses that, when *g* is not offered at the same time as action *a*, the execution of action *a* resets the counter to 0. The **Dg2 rule** expresses that, when *g* is offered at the same time as action *a*, the execution of action *a* has no effect on the counter. The **Dg3 rule** expresses that the execution of *g* is possible when the delay has elapsed (role of ω), and that in this case the counter is reset to 0. The **Dg4 rule** expresses that the delay may elapse when the counter

reaches a value *d* in the time interval *I* and action *g* is still offered. In this rule, "*d in I*" means $d_1 \leq d \leq d_2$ where $I := [d_1,d_2]$. The premiss is needed to avoid the expiration of the delay when the *g* offer has just disappeared whereas the counter has not been reset yet. The **Dg5 rule** expresses that, when g is not offered, the process may idle but the counter is not affected. The **Dg6 rule** expresses that, when g is offered, the process may idle until the upper bound of the time interval is reached, and the counter is updated accordingly. In this rule, "d+t ≤ Up(I)" means $d+t \leq d_2$ where $I := [d_1,d_2]$. The **Dg7 rule** expresses that, when *g* is offered and the delay has elapsed, the process may continue idling.

The timer on interactions

In figure 1, the PN represents two processes *P* and *Q* interacting through the *g* gate. None of the processes *P* and *Q* may know when *g* is enabled, and therefore may want to start a local timer to abandon the *g* interaction after some time. Figure 1 illustrates the case where *P* has started a timer of value d_1 as soon as it reaches the place where it offers *g*. In a similar way, *Q* has started a timer of value d_2.

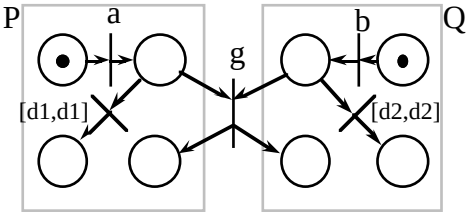

Figure 1: Separate time-outs on *g*

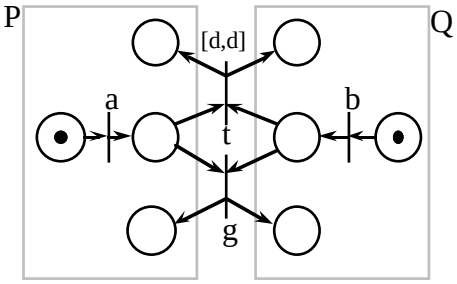

Figure 2: Time-out on the interaction *g*

In Timed LOTOS this system may be simply expressed as follows:

P |[g]| Q where *P := a; (g; P' [] $\Delta^{[d_1,d_1]}$ i; P")* and *Q := b; (g; Q' [] $\Delta^{[d_2,d_2]}$ i; Q").*

Now, let us consider the behaviour of figure 2. In this system there is a timer, represented by the t transition, which is started exactly when *g* is enabled (i.e. by both *P* and *Q*). Such a timer protects the interaction *g*, by providing an escape when *g* has not fired after an enabling time *d*. This modelling has not much sense if the PN is autonomous, because in this case it suffices to require that *g* occur in the interval [0,d]. On the other hand, if the PN is not autonomous and a third process (say an environment) has also to participate in *g*, then this t transition turns out to be very useful. This is precisely the case in LOTOS if *g* is an observable interaction.

The system depicted in figure 2 can be expressed in Timed LOTOS as follows (note that *t* is hidden but not *g*):

hide t in $\Delta_t^{[d,d]}$ *(P |[g]| Q)* where *P := a; (g; P' [] t; P")* and *Q := b; (g; Q' [] t; Q").*

This behaviour makes use of the previous delay operator applied on the *t* interaction. However, we think that this Timed LOTOS behaviour remains somehow tricky, because it necessitates the introduction of an additional gate *t*. The purpose of the $\lfloor P \rfloor_g^d(Q)$ operator is precisely to avoid this trick and thus provide a cleaner way to model such a basic mechanism.

With the timer operator on *g*, the Timed LOTOS specification becomes:

$\lfloor P \,|[g]|\, Q \rfloor_g^d$ *(P" |[g]| Q")* where *P := a; g; P'* and *Q := b; g; Q'*.

One will appreciate that the specification of P and Q are kept very simple.

The $\lfloor P \rfloor_g^d(Q)$ operator will also be defined via an intermediate operator, denoted $\lfloor P \rfloor_g^{d,d'}(Q)$, according to the following definition: $\lfloor P \rfloor_g^d(Q) := \lfloor P \rfloor_g^{d,d}(Q)$. The second argument *d'*, referred to as the counter, is used to count down the remaining time until the expiration of the time-out on *g*. The operational semantics is given in table 4.

The **Tg1 rule** expresses that, when *g* is not offered at the same time as action *a*, the execution of action *a* resets the counter to *d*. The **Tg2 rule** expresses that, when *g* is offered at the same time as action *a*, the execution of action *a* has no effect on the counter. The **Tg3 rule** expresses that the execution of *g* resets the counter to *d*. The **Tg4 rule** expresses that, when *g* is enabled, the timer expires if the counter has reached the value 0. The premiss is needed to avoid the expiration of the timer when the *g* offer has just disappeared whereas the counter has not been reset yet. The **Tg5 rule** expresses that, when *g* is not offered, the process may idle but the counter is not affected. The **Tg6 rule** expresses that, when *g* is offered, the process may idle provided that the counter, which is decremented accordingly, remains ≥ 0.

(Dg1) $\dfrac{P \xrightarrow{a} P',\ P \not\xrightarrow{g}}{\Delta_g^{I,d}(P) \xrightarrow{a} \Delta_g^{I,0}(P')} \left(\begin{array}{c} a \in L^{i,\delta,\theta} - \{g\}, \\ d \in D^{\omega} \end{array}\right)$	(Dg5) $\dfrac{P \xrightarrow{t} P',\ P \not\xrightarrow{g}}{\Delta_g^{I,d}(P) \xrightarrow{t} \Delta_g^{I,d}(P')} (d,t \in D^{\omega})$
(Dg2) $\dfrac{P \xrightarrow{a} P',\ P \xrightarrow{g}}{\Delta_g^{I,d}(P) \xrightarrow{a} \Delta_g^{I,d}(P')} \left(\begin{array}{c} a \in L^{i,\delta,\theta} - \{g\}, \\ d \in D^{\omega} \end{array}\right)$	(Dg6) $\dfrac{P \xrightarrow{t} P',\ P \xrightarrow{g}}{\Delta_g^{I,d}(P) \xrightarrow{t} \Delta_g^{I,d+t}(P')} \left(\begin{array}{c} d,t \in D^{\omega} \\ d+t \le Up(I) \end{array}\right)$
(Dg3) $\dfrac{P \xrightarrow{g} P'}{\Delta_g^{I,\omega}(P) \xrightarrow{g} \Delta_g^{I,0}(P')}$	(Dg7) $\dfrac{P \xrightarrow{t} P',\ P \xrightarrow{g}}{\Delta_g^{I,\omega}(P) \xrightarrow{t} \Delta_g^{I,\omega}(P')} (t \in D^{\omega})$
(Dg4) $\dfrac{P \xrightarrow{g}}{\Delta_g^{I,d}(P) \xrightarrow{\theta} \Delta_g^{I,\omega}(P)}$ (d in I)	
(Tg1) $\dfrac{P \xrightarrow{a} P',\ P \not\xrightarrow{g}}{\lfloor P \rfloor_g^{d,d'}(Q) \xrightarrow{a} \lfloor P' \rfloor_g^{d,d}(Q)} \left(\begin{array}{c} a \in L^{i,\delta,\theta} - \{g\}, \\ d,d' \in D \end{array}\right)$	(Tg5) $\dfrac{P \xrightarrow{t} P',\ P \not\xrightarrow{g}}{\lfloor P \rfloor_g^{d,d'}(Q) \xrightarrow{t} \lfloor P' \rfloor_g^{d,d'}(Q)} \left(\begin{array}{c} d,d' \in D, \\ t \in D^{\omega} \end{array}\right)$
(Tg2) $\dfrac{P \xrightarrow{a} P',\ P \xrightarrow{g}}{\lfloor P \rfloor_g^{d,d'}(Q) \xrightarrow{a} \lfloor P' \rfloor_g^{d,d'}(Q)} \left(\begin{array}{c} a \in L^{i,\delta,\theta} - \{g\}, \\ d,d' \in D \end{array}\right)$	(Tg6) $\dfrac{P \xrightarrow{t} P',\ P \xrightarrow{g}}{\lfloor P \rfloor_g^{d,d'+t}(Q) \xrightarrow{t} \lfloor P' \rfloor_g^{d,d'}(Q)}$ (d,d',t ∈ D)
(Tg3) $\dfrac{P \xrightarrow{g} P'}{\lfloor P \rfloor_g^{d,d'}(Q) \xrightarrow{g} \lfloor P' \rfloor_g^{d,d}(Q)}$ (d,d' ∈ D)	
(Tg4) $\dfrac{P \xrightarrow{g}}{\lfloor P \rfloor_g^{d,0}(Q) \xrightarrow{\theta} Q}$ (d ∈ D)	

Table 4: Operational semantics of additional timed operators

The correctness of the operational rules of the second column for these two operators relies on the persistency and reverse persistency properties. For instance, suppose that the persistency property is not verified for action *g*, and consider the Dg6 rule. The fact that the second premiss is true for *P* does not imply that it remains true along any t transition when *P* becomes *P'*. If this is not the case, it is incorrect to increase the value of the counter by *t*. The same reasoning can be done for Tg6.

Now suppose that the reverse persistency property is not preserved for action *g*, and consider Dg5. The fact that the second premiss is true for *P* does not imply that it remains true along any t transition when *P* becomes *P'*. If this is not the case, it is incorrect to keep the value of the counter unchanged. The same reasoning can be done for Tg5.

Thanks to these two operators, the design of complex timed behaviours is possible in a modular way: the components of a complex system may be specified separately, and then combined with the parallel operator and these timed operators to obtain a structured specification, e.g. in a constraint-oriented or resource-oriented style.

It should be stressed however that these $\Delta_g^I(P)$ and $\lfloor P \rfloor_g^d(Q)$ operators impose the **same** time constraints to all actions g in P. This is not considered as a major shortcoming as explained hereafter. Let us consider two cases.

1) If these *g* actions are simple actions of subprocesses in P (or of P itself), the time constraints can be expressed without using these operators which were not designed for this purpose.

2) If these *g* actions are interactions between subprocesses in P, we do not know any model that tackles the problem better than we do. The timer and asap operators in T-LOTOS and U-LOTOS have also the same shortcoming, and other models have simply no facility to express similar constraints on interactions.

7. Some properties of Timed LOTOS

The first obvious requirement to be fulfilled is the consistency of the proposed operational semantics. Such consistency is indeed easily falsified in the presence of inference rules with negative premises.

Propositions 7.1

(i) The operational semantics of Timed LOTOS without the Δ_g^I and $\lfloor \rfloor_g^d$ operators is consistent.
(ii) If we forbid unguarded recursion in the bodies of process definitions, the operational semantics of full Timed LOTOS is consistent.

Proof

(i) The transition system specification given in tables 2 and 3, which is intended to define an operational semantics of our timed LOTOS, is stratifiable according to definition 2.3.1 of [Gro 90b]. The following function *S* can be proved to be a stratification: $S(P \xrightarrow{-a} P') := rk(a)$ *where* $rk(a) = 0$ *if* $a \in L^{i,\delta,\theta}$ *and* $rk(a) = 1$ *if* $a \in D^{\omega}$. Therefore, applying theorem 2.4.2 of [Gro 90b], the semantics is consistent.
(ii) If we forbid unguarded recursion in the bodies of process definitions, the transition system specification given in tables 2, 3 and 4 is stratifiable, because the following function *S* can be proved to be a stratification: $S(P \xrightarrow{-a} P') := rk(a)$ + *number of* Δ_g^I *and* $\lfloor \rfloor_g^d$ *in P + number of* Δ_g^I *and* $\lfloor \rfloor_g^d$ *in the bodies Q' of all process names Q (so Q := Q') that occur unguarded in P.* rk(a) is defined as above. The proof of consistency follows as in (i). ❐

The second important requirement is that strong bisimulation be a congruence. This is very important in order to be able to replace a part of a Timed LOTOS description by another strongly bisimilar process without changing the semantics of the description, i.e. the overall description remains strongly bisimilar to the original one.

We have first to define the meaning of strong bisimulation in our context. This is very simple because our underlying model is the usual LTS. Of course this LTS will generally be infinite states and infinitely branching with D as a dense time domain, but this does not matter here. Note that since D is a countable domain, the LTS model can be used here.

Consider a $LTS = \langle S, L^{i,\delta,\theta} \cup D^{\omega}, T, s_0 \rangle$. A relation $\underline{R} \subseteq S \times S$ is a *strong bisimulation* iff :
$\forall \langle B_1, B_2 \rangle \in \underline{R}, \forall a \in L^{i,\delta,\theta} \cup D^{\omega}$, we have

(i) if $B_1 \xrightarrow{-a} B_1'$, then $\exists B_2'$ *such that* $B_2 \xrightarrow{-a} B_2'$ *and* $\langle B_1', B_2' \rangle \in \underline{R}$

(ii) if $B_2 \xrightarrow{-a} B_2'$, then $\exists B_1'$ *such that* $B_1 \xrightarrow{-a} B_1'$ *and* $\langle B_1', B_2' \rangle \in \underline{R}$

This is the classical definition of a strong bisimulation, where θ and timed arcs from D^{ω} are considered as any other action. The strong bisimulation equivalence between two LTSs is then defined as usual.

Proposition 7.2

(i) In Timed LOTOS without the Δ_g^I and $\lfloor \rfloor_g^d$ operators, strong bisimulation is a congruence.

(ii) If we forbid unguarded recursion in the bodies of process definitions in full Timed LOTOS, strong bisimulation is a congruence.

Proofs

It can be checked very easily that all the rules of tables 2, 3 and 4 are in the *ntyft/ntyxt* formats and are well founded (refer to definitions 4.2 and 4.3 of [Gro 90b]). Moreover, this transition system specification is stratifiable (refer to the proofs of propositions 7.1); and therefore, applying theorem 4.4 from [Gro 90b], strong bisimulation is a congruence. ❒

The persistency and reverse persistency properties must also be verified, otherwise some inference rules would not be adequate.

Propositions 7.3

Timed LOTOS has the persistency and the reverse persistency properties:

(i) $\forall\ P,P',\forall\ t \in D^{\omega}, \forall\ a \in L^{i,\delta,\theta}, (P \xrightarrow{t} P' \wedge P \xrightarrow{a} \Rightarrow P' \xrightarrow{a})$

(ii) $\forall\ P,P',\forall\ t \in D^{\omega}, \forall\ a \in L^{i,\delta}, (P \xrightarrow{t} P' \wedge P \not\xrightarrow{a} \Rightarrow P' \not\xrightarrow{a})$ Note that $a \neq \theta$ in (ii).

Proofs

(i) We proceed by structural induction on the transition system specification given in tables 2, 3 and 4. We have to check all the t-axioms and t-rules (right columns of the tables). It is obvious that all t-axioms preserve the persistency property. For every t-rule, the induction is carried out as follows: we suppose that all the t-transitions in the premises satisfy the persistency property, and we prove that the consequent of the t-rule then also satisfies the property, provided that the premises are satisfied.

(ii) Similar proof for the reverse persistency. ❒

8. Conclusion and perspective

We have presented a timed extension to LOTOS in which neither the syntax nor the semantics are restricted to a specific time domain, i.e. a dense time domain is supported as well. In this paper, we kept the LTS as the semantic model of Timed LOTOS. However, as pointed out earlier in this paper, when the time domain is dense (but countable), the LTS associated with a Timed LOTOS process is necessarily infinite states and infinitely branching. Therefore, even if this is not a problem in the theoretical work on Timed LOTOS, it would be useful to find a better model for any practical work on it such as the design of tools. One such model, called a timed graph model, is presented in [ACD 90]. A timed graph is a state-transition graph extended with a mechanism that allows the expression of constraints on the delays between the state transitions. Constraints are expressed as predicates on state variables representing timers. In addition to the limited state explosion, there exist model checking methods for temporal logics with quantitative temporal operators which are directly applicable to them. In [NSY91] a method for the translation of ATP into timed graph is presented. For Timed PNs, a similar finite representation exists which combines the usual marking with inequalities on time values. Therefore, it is likely that timed graphs or another similar model be more adequate than a usual LTS for Timed LOTOS. This is an interesting topic which is beyond the scope of this paper.

The semantics of Timed LOTOS (without unguarded recursion) is proved consistent and strong bisimulation is a congruence. Timed LOTOS does not allow the general action urgency but is based on the well-known maximal progress property (i.e. urgency on internal actions). The operational semantics has some interesting characteristics: there are no negative premises on *t* transitions, the urgency is introduced via hiding (and of course enabling which incorporates an implicit hiding), there is a special action θ which models the "elapsed time event" and gives the model the persistency and reverse persistency properties.

We have proposed three original timed operators. The simple delay operator Δ^{I} allows the specification of a time interval *I* in which the delay is nondeterministically chosen. With respect to other models based on maximal progress, this operator has an advantage: it allows to specify a temporal nondeterminism on the occurrence of internal actions. Two other timed operators are defined which allow the expression of timed constraints on interactions, i.e. on actions involv-

ing several processes. The only other timed algebras that have a similar flexibility are T-LOTOS and U-LOTOS (timer and asap operators) but they are restricted to the interaction urgency. In this paper, as we have rejected the general action urgency, we do not consider such operators, but we focus our interest on the ability to express delays and timeouts on such interactions.

Even if no example is treated here due to lack of place, the basic choices of Timed LOTOS, as well as its timed operators, have been greatly influenced by a companion study in project ESPRIT II/OSI95 in which a case study for the assessment of timed FDTs has been designed [LLD 92]. This case study incorporates many timed features of real systems.

This work should be extended to define an adequate weak bisimulation equivalence which abstracts away from both internal actions i and θ. It will be also useful to verify whether the classical weak bisimulation laws of LOTOS still hold in Timed LOTOS. This is left for further study. Finally, as pointed out in [Wan 91], when dealing with a dense time domain, the expansion theorem cannot be preserved directly, and we will probably have to face the same difficulty. In [Wan 91] a solution to this problem is proposed which consists in including a timed action-prefix in the language.

References

[ACD 90] R. Alur, C. Courcoubetis, D. Dill, *Model-checking for real-time systems*, in: Fifth annual IEEE symposium on logic in computer science, Philadelphia, P.A., June 1990, 414-425.

[Azc 90] A. Azcorra-Saloña, *Formal Modeling of Synchronous Systems*, Ph. D. Thesis, ETSI Telecomunicación, Universidad Politécnica de Madrid, Spain, Nov. 1990.

[BaB 90] J.C.M. Baeten, J.A. Bergstra, *Real time process algebra*, Rept. No. P8916b, University of Amsterdam, Amsterdam, March 1990.

[BeK 85] J.A. Bergstra, J. W. Klop, *Algebra of Communicating Processes with Abstraction*, Theoretical Computer Science 37 (1985) 77-121 (North-Holland, Amsterdam).

[BLT 90] T. Bolognesi, F. Lucidi, S. Trigila, *From Timed Petri Nets to Timed LOTOS*, in: L. Logrippo, R. Probert, H. Ural, eds., Protocol Specification, Testing and Verification, X (North-Holland, Amsterdam, 1990) 395-408.

[BoB 87] T. Bolognesi, E. Brinksma, *Introduction to the ISO Specification Language LOTOS*, Computer Networks and ISDN Systems 14 (1) 25-59 (1987).

[BoL 92] T. Bolognesi, F. Lucidi, *LOTOS-like process algebras with urgent or timed interactions*, in: K. Parker, G. Rose, eds., Formal Description Techniques, IV (North-Holland, Amsterdam, 1992) 249-264.

[Bri 89] E. Brinksma, *Constraint-oriented specification in a constructive formal description technique*, in: J.W. de Bakker, W.-P. de Roever, G. Rozenberg, eds., Stepwise Refinement of Distributed Systems, Models Formalisms, Correctness, LNCS 430 (Springer-Verlag, Berlin, 1989) 130-152.

[DaS 89] J. Davies, S. Schneider, *An introduction to timed CSP*, Rept. No. PRG-75, Oxford University Computing Laboratory, Programming Research Group, Aug. 1989.

[Gro 90a] J. F. Groote, *Specification and Verification of Real Time Systems in ACP*, in: L. Logrippo, R. Probert, H. Ural, eds., Protocol Specification, Testing and Verification, X (North-Holland, Amsterdam, 1990) 261-274.

[Gro 90b] J. F. Groote, *Transition system specifications with negative premisses*, in: J.C.M. Baeten, J.W. Klop, eds., CONCUR '90, Theories of Concurrency: Unification and Extension, LNCS 458 (Springer - Verlag, Berlin Heidelberg New York, 1990) 332-341.

[HaJ 90] H. Hansson, B. Jonsson, *A calculus for communicating systems with time and probabilities*, in: 11th IEEE Real-Time Systems Symposium, Orlando, Florida, 1990, IEEE Computer Society Press

[Han 91] H. Hansson, *Time and Probability in Formal Design of Distributed Systems*, Ph. D Thesis, DoCS 91/27, Uppsala University, Dept. of Computer Science, P.O. Box 520, S-75120 Uppsala, Sweden.

[HeR 90] M. Hennessy, T. Regan, *A temporal process algebra*, in: J. Quemada, J. Mañas, E. Vazquez, eds., Formal Description Techniques, III (North-Holland, Amsterdam, 1991) 33-48.

[Hoa 85] C.A.R. Hoare, *Communicating Sequential Processes*, (Prentice-Hall International, London, 1985).

[ISO 8807] ISO/IEC-JTC1/SC21/WG1/FDT/C, *IPS - OSI - LOTOS, a Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, IS 8807, February 1989.

[Led 92] G. Leduc, *An upward compatible timed extension to LOTOS*, in: K. Parker, G. Rose, eds., Formal Description Techniques, IV (North-Holland, Amsterdam, 1992) 217-232.

[LLD 92] L. Léonard, G. Leduc, A. Danthine, *The Tick-Tock case study for the assessment of Timed FDTs*, Rept. No. S.A.R.T. 92/02/05, Université de Liège, B28, B-4000 Liège, Belgium, January 1992.

[MeF 76] P.M. Merlin, D.J. Farber, *Recoverability of Communication Protocols - Implications of a theoretical Study*, IEEE Transaction on Communication, 24, Sept. 76, 1036-1043.

[MFV 92] C. Miguel, A. Fernandez, L. Vidaller, *Extending LOTOS towards performance evaluation*, this vol.

[Mil 89] R. Milner, *Communication and Concurrency*, (Prentice-Hall International, London, 1989).

[MoT 90] F.Moller, C. Tofts, *A temporal calculus of communicating systems*, in: J.C.M. Baeten, J.W. Klop, eds., CONCUR '90, Theories of Concurrency: Unification and Extension, LNCS 458 (Springer - Verlag, Berlin Heidelberg New York, 1990) 401-415.

[NiS 91a] X. Nicollin, J. Sifakis, *An Overview and Synthesis on Timed Process Algebras*, in: K.G. Larsen, A. Skou, eds., Computer-Aided Verification, III (LNCS 575, Springer-Verlag, Berlin Heidelberg New York, 1992) 376-398. Also in: LNCS 600.

[NiS 91b] X. Nicollin, J. Sifakis, *The Algebra of Timed Processes ATP: Theory and Application*, Rept. No. RT-C26, Projet Spectre, LGI-IMAG, Nov. 1991.

[NRS 90] X. Nicollin, J.-L. Richier, J. Sifakis, J. Voiron, *ATP : An algebra for timed processes*, in: M. Broy, C.B. Jones, eds., IFIP Working Conference on Programming Concepts and Methods, Sea of Gallilee, Israel (North-Holland, Amsterdam, 1990).

[NSY 91] X. Nicollin, J. Sifakis, S. Yovine, *From ATP to Timed Graphs and Hybrid Systems*, in: J.W. de Bakker, C. Huizing, W.P. de Roever, G. Rozenberg, eds., Real-Time: Theory and Practice (LNCS 600, Springer-Verlag, Berlin Heidelberg New York, 1992) 549-572.

[QAF 89] J. Quemada, A. Azcorra, D. Frutos, *TIC: A timed calculus for LOTOS*, in: S. T. Vuong, ed., Formal Description Techniques, II (North-Holland, Amsterdam, 1990) 195-209.

[Ree 90] G.M.Reed, *A Hierarchy of Domains for Real Time Distributed Computing*, in: M. Main, A. Melton, M. Mislove, D. Schmidt, eds., Mathematical Foundations of Programming Semantics (LNCS 442, Springer-Verlag, Berlin Heidelberg New York, 1990) 80-128.

[ReR 88] G.M.Reed, A.W. Roscoe, *A Timed Model for Communicating Sequential Processes*, Theoretical Computer Science 58 (1988) 249 - 261 (North-Holland, Amsterdam).

[Wan 90] Y. Wang, *Real-Time Behaviour of Asynchronous Agents*, in: J.C.M. Baeten, J.W. Klop, eds., CONCUR '90, Theories of Concurrency: Unification and Extension, LNCS 458 (Springer - Verlag, Berlin Heidelberg New York, 1990) 502-520.

[Wan 91] Y. Wang, *CCS + Time = an Interleaving Model for Real Time Systems*, in: J. Leach Albert, B. Mounier, M. Rodríguez Artalego, eds., Automata, Languages and Programming, 18 (LNCS 510, Springer-Verlag, Berlin Heidelberg New York, 1991) 217-228.