FORMAL DESCRIPTION TECHNIQUES, IV
K.R. Parker and G.A. Rose
Elsevier Science Publishers B.V. (North-Holland)

An upward compatible timed extension to LOTOS

Guy Leduc

Research Associate 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 662698 Fax: + 32 41 662989
E-mail: leduc@montefiore.ulg.ac.be

Abstract

We propose a timed extension of LOTOS, denoted TLOTOS, which is upward compatible with the standard LOTOS. It means first that a timed behaviour expression which does not use any language extension will have the same semantics as in standard LOTOS. In addition, we have pushed this upward compatibility one step beyond by requiring and obtaining that all the familiar equivalence laws of standard LOTOS be preserved, e.g. *B [] B ~ B*, *B [] stop ~ B*. This is needed in order to keep the intuition of a standard LOTOS user unchanged. TLOTOS has been mainly inspired by other approaches such as Moller & Tofts's TCCS (Temporal CCS), Hennessy & Regan's TPL (Temporal Process Language) and Nicollin & Sifakis's ATP (Algebra of Timed Processes). Our model is not strictly asynchronous (like standard LOTOS, CCS, CSP, ACP) nor strictly synchronous (like SCCS, CIRCAL, Meije). For compatibility and simplicity, we decided to keep the model asynchronous for a large part and, for dealing with time, to introduce a "synchronous part" by way of a new basic "synchronous" or timed action in the asynchronous semantic model. The design of our TLOTOS is presented together with other possible design alternatives which are all rejected according to our compatibility or expressive power requirements. The solution that we obtain is simple and satisfactory, and its operational semantics is presented in depth. Two examples are provided to illustrate the use of TLOTOS.

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] have been extended to Timed CSP [ReR 88, Ree 90], TCCS (Temporal CCS) [MoT 89], TIC [QAF 89], ACP_ρ [BaB 90], CELOTOS [HTZ 90], $ACP_{\tau\varepsilon}^{t}$ [Gro 90], Timed-Interaction LOTOS [BLT 90, BoL 91], Timed CCS [Wan 90, Wan 91] and TPCCS [HaJ 90, Han 91]. New process algebras have been proposed which are intended to model time in a quantitative way: first, synchronous process algebras such as SCCS [Mil 83], Meije [AuB 85] or CIRCAL [Mil 85] and, in a second step, timed process algebras such as TPL (Temporal Process Language) [HeR 90], ATP (Algebra of Timed Processes) [NRS 90, NS 90, NSY 91], PADS (Process Algebra for Distributed Systems) [Azc 90]. An overview and synthesis on Timed Process Algebras may be found in [NS 91].

In this paper, we present a timed extension of LOTOS, denoted TLOTOS, with the objective of upward compatibility with standard LOTOS. This means that LOTOS will be a proper subset of TLOTOS (i.e. LOTOS specifications will be correct TLOTOS specifications, and their semantics will be unchanged). In addition, we have been able to preserve all the LOTOS strong and weak bisimulation laws in TLOTOS. This is fundamental in order to facilitate as much as possible its use by those who were trained on standard LOTOS. These are our requirements for claiming (true) upward compatibility. They will be further developed in section 2.

TLOTOS has been inspired by other approaches such as TPL [HeR 90], ATP [NS 90], $ACP_{\tau\varepsilon}^{t}$ [Gro 90] and PADS [Azc 90]. Our model is between a strictly asynchronous algebra (like LOTOS, CCS, CSP, ACP) and a strictly synchronous one (like SCCS, CIRCAL, Meije). The idea has been suggested in the works previously mentioned here-above and may be informally presented as the addition of a "synchronous action" in an asynchronous algebra. The motivation is twofold. First, this way of doing is clearly in the line of an upward extension of LOTOS since the basic asynchronous model is our starting point. Second, this corresponds to the fact that distributed systems are mainly systems which behave most of the time asynchronously, but resynchronize themselves periodically. It is therefore unacceptable to burden a whole specification with timing considerations when only a small subset of it really depends on time. This is the main shortcoming of purely synchronous models.

The timed action that we will add in LOTOS may be defined with different properties in mind, which would lead to several possible extensions. The design of TLOTOS will thus be explained by presenting other possible design alternatives which will be all rejected according to our requirements of compatibility and expressing power. In this process, we do not claim of course to have explored all the possible timed extensions of LOTOS. However, we have tried to be as exhaustive as possible with respect to the possible extensions based on the fundamental concept of a distinct timed action.

The timed action is the basic element of the extended semantics. At the syntactic level, several additional constructs are needed to benefit totally from this new expressive facility. These constructs will be introduced and illustrated. Let us note that, at this level, other operators might have been used or could advantageously replace the chosen ones. We have simply selected an existing set of such timed operators (viz. those proposed in ATP) in order to avoid unneeded additional notations. However, this choice is not arbitrary: we think that the ATP timed operators are simple, easy to use and understand, and perfectly adequate to model many timed behaviours such as time-outs, watchdogs, … TLOTOS is proposed with its operational semantics for the main operators.

In order to illustrate the use of TLOTOS, we will also present two small examples. The first one is the transmitter entity of the well-known alternating bit protocol. The second one is a unidirectional medium which introduces delays on the exchanged messages.

2. Upward compatibility with LOTOS

The main objective of this timed extension that we are looking for is its upward compatibility with current LOTOS [ISO 8807]. This means that, in TLOTOS, a behaviour expression which does not use any language extension should have the same semantics as in standard LOTOS. This allows the description in TLOTOS of untimed behaviours, exactly in the same way as in LOTOS. This requirement is necessary to achieve upward compatibility, but we would like to require more. We would like to use TLOTOS with the same intuition as LOTOS. This means that behaviour expressions which were (strongly, weakly, …) bisimulation equivalent in LOTOS should remain equivalent in TLOTOS. In other words, we want to preserve all the familiar laws of LOTOS: e.g. *B [] B ~ B*, *B [] stop ~ B*, the commutativity and associativity of various operators such as choice or parallel composition with the same gate list. Whereas the first requirement only applies to the LOTOS subset of TLOTOS, this last requirement applies to the full TLOTOS.

The first requirement can be achieved as follows: first, we should preserve, in the operational semantics, all the axioms and inference rules of the LOTOS operators; second, if new axioms and inference rules are added for **standard** LOTOS **operators**, they should not allow new transitions when the **operands** are untimed processes. By contrast, the axioms and inference rules of the new timed operators remain unconstrained by this first requirement. Indeed, in this case, any behaviour expression in TLOTOS which does not use any new operator, will be an untimed behaviour expression, and will thus have exactly the same Labelled Transition System (LTS) as in LOTOS. A precise definition of a LTS will be given later on.

The second requirement, however, will add further constraints to the new axioms and rules of the standard operators.

Basic design choices of LOTOS also have to be preserved in order to achieve this upward compatibility. For instance, actions will remain atomic and instantaneous (i.e. with duration 0), and the semantics of parallelism will still be based on interleaving. We will see that this combination may lead to a satisfactory approximation of real parallelism. Intuitively, sequences of instantaneous actions may be considered to occur during the same unit time interval. If the set of possible interleaved sequences are all specified, this may be an adequate model of true parallelism. These sequences will be separated from subsequent actions (i.e. which should occur at a later time) by means of a special timed action, as presented later on in this paper.

3. Basic choices related to the timed extension

We decided to follow the approaches of TCCS, TPL and ATP to define TLOTOS, i.e. the definition of a model which is not strictly asynchronous (like LOTOS, CCS, CSP, ACP, …) nor strictly synchronous (like SCCS, CIRCAL, Meije).We think that an appropriate model should be asynchronous for a large part, but should allow the expression of precise timing constraints. An asynchronous model as a starting point is thus justified for two reasons: first, an asynchronous model is in general simpler than a synchronous one; second, since we are looking for an upward compatible extension of the asynchronous model of LOTOS, we think that this approach is more natural and also simpler.

One problem remains however: the introduction of a "synchronous part" in the LOTOS asynchronous model. The basic idea has been presented in TPL, ATP, $ACP_{\tau\varepsilon}^{t}$ and PADS, and consists in introducing a new basic action in the semantic model. This action, which will be denoted χ (like in ATP) in the sequel, is called a timed action. Its occurrence models the passing of one unit of time. A process may thus either execute an observable action (i.e. a synchronization with its environment), or execute an asynchronous internal action (the well-known *i*), or execute a timed action χ modelling the passing of one unit of time.

This χ action is however a very low level construct which would, if introduced as such in the language, unnecessarily complicate TLOTOS, as well as its use for the description of time-dependent systems. Therefore, following ATP, we decided to make χ invisible at the syntactic level of the language. This is quite similar to the introduction, in the semantic model only, of the δ action, modelling a successful termination: no δ action ever appears in a LOTOS behaviour expression, it is hidden in a syntactic construct which is, in this case, the *exit* process.

We will follow the same approach, and define several new syntactic constructs whose semantics refer to the χ action. Some additional axioms and inference rules of standard LOTOS will also be defined and make use of this χ action.

We will adopt the same three basic timing constructs as in ATP. We only present two of them in section 3 for illustration, the third one will be justified later on in section 4.

Notations

L is the alphabet of observable actions.

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

$P \text{ –}a\text{→ } P'$ where $a \in L^{i,\delta,\chi}$, means that process P may engage in action a and, after doing so, behave like process P'.

$P \text{ –}a\text{↛}$ where $a \in L^{i,\delta,\chi}$, means that $\neg$ *(*$\exists$ *P', such that* $P \text{ –}a\text{→ } P'$*)*, i.e. P cannot accept (or must refuse) the action a.

The *start delay* operator (or time-out operator), $\lfloor\ \rfloor^d$

$\lfloor P \rfloor^d(Q)$ is a process which behaves like *P* provided *P* starts before *d* units of time, otherwise it behaves like *Q*. Note that *d* is required to be strictly greater than 0, and that, when d = 1, it may be omitted. Its operational semantics is defined by the following four inference rules [NS 90]:

(SD1)	$\dfrac{P \xrightarrow{a} P'}{\lfloor P \rfloor^d(Q) \xrightarrow{a} P'}$	$(a \in L^{i,\delta}, d \geq 1)$	(SD3)	$\dfrac{P \xrightarrow{\chi} P'}{\lfloor P \rfloor^{d+1}(Q) \xrightarrow{\chi} \lfloor P' \rfloor^d(Q)}$	$(d \geq 1)$
(SD2)	$\lfloor P \rfloor^1(Q) \xrightarrow{\chi} Q$		(SD4)	$\dfrac{P \xrightarrow{\chi} \not\rightarrow \ \$}{\lfloor P \rfloor^{d+1}(Q) \xrightarrow{\chi} \lfloor P \rfloor^d(Q)}$	$(d \geq 1)$

The process $\lfloor P \rfloor^d (Q)$ would advantageously replace the following imprecise construction often used in standard LOTOS for modelling a time-out:

P [] (Time-out (d) >> Q) where Time-out (d:nat) := exit

As an example, the figure 1 gives the LTS[§] associated with the process $\lfloor a; stop \rfloor^2(b; stop)$.

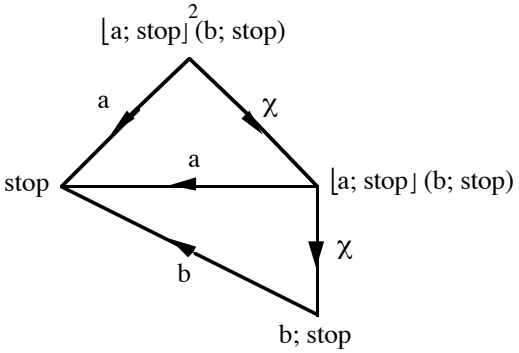

Figure 1: the LTS of $\lfloor a; stop \rfloor^2(b; stop)$

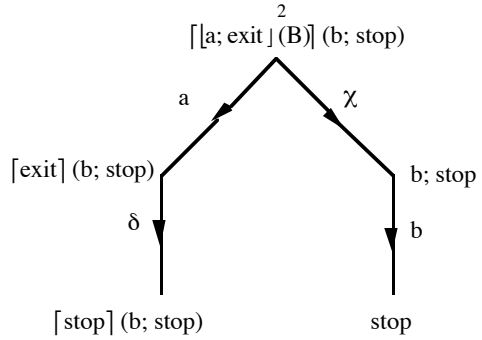

Figure 2: the LTS of $\lceil \lfloor a; exit \rfloor^2(B) \rceil (b; stop)$

When a well-defined semantics is given to *stop* in the next section, we will see that a simple delay of *d* units of time before the start of a process *Q* will be introduced as follows:

$$\lfloor stop \rfloor^d(Q)$$

The idea is that this process may only execute the trace χ^d and then behave like *Q*.

As mentioned in [NS 90], one might think (by looking at the union of their premises, and at their common inferred part) that the rules SD3 and SD4 could be replaced by a single simpler axiom such as $\lfloor P \rfloor^{d+1}(Q) \xrightarrow{\chi} \lfloor P \rfloor^d(Q)$. However, this is not intuitively satisfactory in the case of nested delays as explained on the following example from [NS 90]: in such a semantics, the process $\lfloor \lfloor a; P \rfloor^1(b; Q) \rfloor^2(R)$ would not have the ability to execute *b*; in fact, it would be equivalent to $\lfloor a; P \rfloor^2(R)$.

$ This is called a negative premiss. This is the first time that we encounter one of them. This kind of transition system specification may create inconsistencies in general. In TLOTOS, these potential problems have been analysed in [Led 91].

§ A precise definition of a LTS is given later on. Moreover, at this stage, without knowing the semantics of action-prefix and *stop* in TLOTOS, it may seem too early to give the LTS associated with this process. Therefore, this LTS should be considered only as a didactic aid to understand the operator. In fact, it will turn out to be the correct LTS according to semantics that we will define. This note is also valid for the subsequent examples.

The *execution delay* operator (or watchdog operator), $\lceil\rceil^d$

$\lceil P\rceil^d$ (Q) is a process which behaves like P before the d^{th} unit of time, and like *Q* afterwards provided *P* has not terminated successfully before this d^{th} unit of time (i.e. executed a δ action). When d = 1, it may be omitted. Its operational semantics is defined by the following four inference rules:

(ED1)	$\dfrac{P \xrightarrow{a} P'}{\lceil P\rceil^d(Q) \xrightarrow{a} \lceil P'\rceil^d(Q)}$	$(a \in L^i, d \geq 1)$	(ED3)	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P\rceil^1(Q) \xrightarrow{\chi} Q}$	
(ED2)	$\dfrac{P \xrightarrow{\delta} P'}{\lceil P\rceil^d(Q) \xrightarrow{\delta} P'}$	$(d \geq 1)$	(ED4)	$\dfrac{P \xrightarrow{\chi} P'}{\lceil P\rceil^{d+1}(Q) \xrightarrow{\chi} \lceil P'\rceil^d(Q)}$	$(d \geq 1)$

The process $\lceil P\rceil^d$ (Q) would advantageously replace the following imprecise construction often used in standard LOTOS for modelling a watchdog:

P [> (Time-out (d) >> Q) where Time-out (d:nat):exit := exit

An example is presented in figure 2. In this case, it appears that *B* cannot be executed. The reason is that *B* cannot start before time 2 and, on the other hand, the watchdog is programmed to start at time 1. The LTS would have been the same for $\lceil\lfloor$a; exit$\rfloor^1$(B)$\rceil$ (b; stop) since the watchdog has the priority at time 1 according to the semantics.

ED1, ED3 and ED4 are given in [NS 90], whereas ED2 is different from its analogous rule where the consequent is instead $\lceil P\rceil^d(Q) \xrightarrow{i} P'$. This difference has the following interpretation: in ATP a δ action[1] is intended to cancel only one level of execution delay, whereas our semantics cancels all the levels of execution delay. Our choice is justified because it is very close to the semantics of the disabling operator where δ cancels all the levels of disabling. It is therefore interesting to keep this interpretation for cancellations of "pending watchdogs".

4. Modelling urgency on action occurrences

We think that a timed model cannot be useful if it does not allow the expression that a given synchronization must occur urgently, i.e. as soon as possible; which means, in our context, before the next χ action. This requirement is closely related to the ability to model timers: without urgent actions, there is no way to specify the necessity for an action to occur at (or before) a specific time instant, or to occur as soon as possible.

An interesting and easy way to provide this capability in our framework is achieved by keeping unchanged the axioms of the action-prefix operator and *exit* process, i.e.

a; B $\xrightarrow{a}$ *B* remains of course an axiom (remember that $a \neq \chi$),

but *a; B* does not allow a χ action, i.e. *a* cannot be delayed: *a; B* $\not\xrightarrow{\chi}$

In particular, in *i; B* the internal action *i* must occur before the next χ action.

Similarly, *exit* $\xrightarrow{\delta}$ *stop* remains valid,

but the successful termination cannot be delayed: *exit* $\not\xrightarrow{\chi}$

This is the approach chosen in TCCS [MoT 89], ATP [NS 90] and $ACP_{\tau\varepsilon}^{t}$ [Gro 90] which contrasts with the one followed in Timed CSP [ReR 88, Ree 90], TPL [HeR 90] and TPCCS [Han 91] where only internal actions occur urgently (the so-called *maximal progress* property).

---

[1] δ is denoted ξ in ATP and has a different semantics with respect to the parallel composition, i.e. all parallel processes are not required to terminate simultaneously.

By specifying carefully when a χ action is enabled by way of the two timed operators presented above, this allows the specification of processes which behave asynchronously most of the time but resynchronize periodically on χ actions.

In order to allow a process to wait an arbitrary delay before starting its execution, we follow the same approach as in [NS 90] and provide a third timed operator.

The *unbounded start delay* operator, $\lfloor\ \rfloor^{\omega}$

$\lfloor P \rfloor^{\omega}$ is a process which behaves like process *P* except that it may wait an arbitrary long delay before starting. This operator disables the "as soon as possible" rule on the first possible actions of *P*. Its operational semantics is defined by the following three inference rules [NS 90]:

(USD1)	$\dfrac{P \xrightarrow{a} P'}{\lfloor P \rfloor^{\omega} \xrightarrow{a} P'}$ $(a \in L^{i,\delta})$	(USD3)	$\dfrac{P \xrightarrow{\chi} \not\rightarrow}{\lfloor P \rfloor^{\omega} \xrightarrow{\chi} \lfloor P \rfloor^{\omega}}$
(USD2)	$\dfrac{P \xrightarrow{\chi} P'}{\lfloor P \rfloor^{\omega} \xrightarrow{\chi} \lfloor P' \rfloor^{\omega}}$		

Examples are presented in figures 3 and 4.

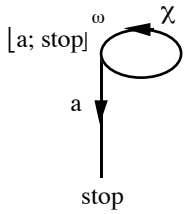

Figure 3: the LTS of $\lfloor a; stop \rfloor^{\omega}$

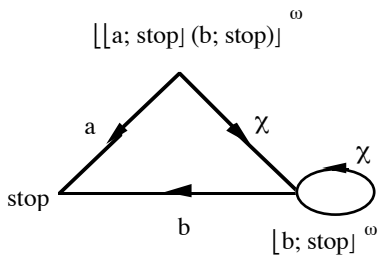

Figure 4: the LTS of $\lfloor \lfloor a; stop \rfloor (b; stop) \rfloor^{\omega}$

In conclusion, unless specifically disabled by an unbounded start delay operator, all actions (observable or not) are required to occur as soon as possible, i.e. before the next χ action. As previously explained, another fundamental exception is when these actions are in the scope of one of the other two timed operators (see rules SD2, SD3, SD4, ED3, ED4).

5. Possible semantics for the timed extension

In this section, we will try to survey all the possible ways to define the semantics of the TLOTOS basic operators, taking account of the requirements and basic choices of the previous sections.

A first basic decision is the semantics of process *stop*, i.e. does it allow a χ action or not ?

The two approaches exist. In TPL [HeR 90], ATP [NS 90], PADS [Azc 90] and TPCCS [Han 91], the process which plays the role of *stop* allows a χ action, and *stop* satisfies the following axiom: $stop \xrightarrow{\chi} stop$. In TCCS [MoT 89] and $ACP_{\tau\varepsilon}^{t}$ [Gro 90] the opposite decision has been taken; in this case, *stop* not only refuses any action, but also stops the progression of time. Even if it may seem to be an undesirable effect at first glance, we may argue in favour of this second approach. First, when *stop* is used explicitly in a specification, it is intended to model that the process has reached a final state where nothing more will happen; therefore, the fact that time is also stopped does not matter any more, since time is no more needed. Second, if *stop* is not modelled explicitly, but results from a synchronization deadlock between processes, this is an error situation which is certainly not wanted; therefore, in this case too, the fact that time is stopped does not matter. An additional argument in favour of the second semantics is the upward compatibility with LOTOS: the LTS associated with *stop* remains the

same, i.e. a unique node without transition. In the first semantics, the LTS associated with *stop* becomes a LTS with a unique node, but with a loop labelled by χ. Therefore, in order to preserve an upward compatibility with LOTOS in this case, we have to be sure that the semantics of any process equivalent to *stop* in standard LOTOS (e.g. *a; stop || b; stop*) is also this LTS with a χ loop. In other words, in order to remain consistent in this case, we must be sure that any process may never refuse the whole alphabet $L^{i,\delta,\chi}$ (since even *stop* cannot). In ATP, when a process may reach a state where it refuses $L^{i,\delta,\chi}$, this is a particular case of a so-called non well-timed process.

Let us summarize this by giving the two possible semantics for *stop*:

(A1) no axiom for *stop*, or

(A2) $stop \xrightarrow{\chi} stop$

The second basic LOTOS construct is the *choice* operator. We have to preserve the usual inference rules of choice, but some alternatives exist to define the semantics of this operator w.r.t. the χ action. For instance, if χ would be considered as an ordinary action, like in $ACP_{\tau\varepsilon}^{t}$ [Gro 90], we would have the following inference rule (nondeterminism of choice w.r.t. time):

(B) $$\frac{P \xrightarrow{\chi} P'}{P \;[]\; Q \xrightarrow{\chi} P'}$$ and its symmetric

However, this rule contradicts our ASAP criterion: suppose that $Q \xrightarrow{a} Q'$ (with $a \in L^{i,\delta}$), then *P [] Q* would allow χ, which is in contraction with the ASAP criterion stating that, in this case, *a* **must** occur before χ. The ASAP criterion appears as a way to restrict the occurrence of χ actions in a very simple way: give χ a lower priority than other actions in $L^{i,\delta}$ **in the scope of a choice operator**.

Note that the absence of such a rule allows anyway the modelling of a process which may (or may not) execute the action *a* of *Q* before the next χ action; it suffices to use the unit delay operator which has been specifically designed for that purpose: $\lfloor Q \rfloor(P')$ in this case.

Being aware of this difficulty, we come naturally to the question of replacing the classical choice rule by (an)other one(s). Several possibilities will be described and their consequences analysed, and finally our proposal will be derived.

Two basic questions that we have to consider when dealing with choice and time are the following:

- may time resolve a choice ?
- may time pass differently for both sides of a choice ?

The first question may also be expressed as: is the choice operator nondeterministic with respect to time ?

In addition to the classical choice rules, let us provide possible answers for the treatment of χ actions in terms of inference rules (the symmetric rules are implicit in the sequel).

(B1) $$\frac{P \xrightarrow{\chi} P',\ Q \xrightarrow{a} \not\quad \forall\ a \in L^{i,\delta}}{P \;[]\; Q \xrightarrow{\chi} P'}$$

(B2) $$\frac{P \xrightarrow{\chi} P',\ Q \xrightarrow{a} \not\quad \forall\ a \in L^{i,\delta}}{P \;[]\; Q \xrightarrow{\chi} P' \;[]\; Q}$$

(B3) $$\frac{P \xrightarrow{\chi} P',\ Q \xrightarrow{\chi} Q'}{P \;[]\; Q \xrightarrow{\chi} P' \;[]\; Q'}$$

(B4) $$\frac{P \xrightarrow{\chi} P',\ Q \xrightarrow{a} \not\quad \forall\ a \in L^{i,\delta,\chi}}{P \;[]\; Q \xrightarrow{\chi} P'}$$

B1 expresses that time may resolve a choice, i.e. when no observable action can be executed in *Q* (however χ may be offered by *Q*), a χ action from *P* may be executed and thereby resolve the choice in favour of *P*.

B2 expresses that time may pass differently on both sides of a choice, i.e. when no observable action can be executed in Q (however χ may be offered by Q), a χ action from P may be executed without resolving the choice (i.e. Q remains enabled afterwards but time has not passed for Q).

B3 expresses that time may pass in both P and Q without resolving a choice. This rule is present in TPL and ATP.

B4 is only relevant when combined with *A1*, it expresses the same idea as *B1*. Moreover, *B4* is useless in the presence of *B1*, i.e. *B1 = B1 + B4*.

With respect to these axioms and rules, *stop* and *choice* in TPL and ATP are defined by A2 + B3 + the classical rules for choice.

Let us consider how these rules *B1,… B4* may be combined together with *A1* and *A2* while preserving well-known and desirable properties of LOTOS operators, such as the strong bisimulation equivalence between *B [] stop* and *B*, or between *B [] B* and *B*, or between *(A |[γ]| B) |[γ]| C* and *A |[γ]| (B |[γ]| C)*, …

Before going further in this discussion, we give the classical definitions of a LTS and the strong bisimulation equivalence. Note that in this last definition, the χ action is considered like any other action.

Definition (strong bisimulation)

Consider a $LTS = <S, L^{i,\delta,\chi}, T, s_0>$. A relation $\underline{R} \subseteq S \times S$ is a strong bisimulation iff :

$\forall <B_1, B_2> \in \underline{R}$,$\forall a \in L^{i,\delta,\chi}$, we have

(i) if $B_1 \text{–}a\rightarrow B_1'$, then $\exists B_2'$ *such that* $B_2 \text{–}a\rightarrow B_2'$ *and* $<B_1', B_2'> \in \underline{R}$

(ii) if $B_2 \text{–}a\rightarrow B_2'$, then $\exists B_1'$ *such that* $B_1 \text{–}a\rightarrow B_1'$ *and* $<B_1', B_2'> \in \underline{R}$

This is the classical definition of a strong bisimulation, where χ is considered as any other action.

Now we define the strong bisimulation equivalence between two LTS's.

Two LTS's $Sys_1 = <S_1, L^{i,\delta,\chi}, T_1, s_{0_1}>$ and $Sys_2 = <S_2, L^{i,\delta,\chi}, T_2, s_{0_2}>$ are strong bisimulation equivalent, denoted $Sys_1 \sim Sys_2$, iff

$\exists$ a strong bisimulation relation $\underline{R} \subseteq S_1 \times S_2$, such that $<s_{0_1}, s_{0_2}> \in \underline{R}$

The weak bisimulation would be defined similarly by replacing, in the definition of the strong bisimulation, the action $a \in L^{i,\delta,\chi}$ by the sequence $\sigma \in (L^{\delta,\chi})^*$. Note that the χ action is considered, like δ and unlike i, in the sequences.

Possible associations of axioms and rules

Since *A1* and *A2* are inconsistent, let us first examine the case where *A2* is selected.

A2 + B1 or A2 + B2

It is shown in [Led 91] that *B [] stop ~ B* is not preserved. Take B := ⌊stop⌋(b; Q).

In addition, for A2 + B2, it is also shown in [Led 91] that *B [] B ~ B* is not preserved either. Take B := $\lfloor stop \rfloor^2$(b; Q).

A2 + B3

This combination is more subtle. It has been investigated in [Led 91] where it was shown that it makes it impossible to preserve the properties of the parallel composition, e.g. *a; stop |[a]| stop ~ stop* together with the associativity of *|[γ]|*. This problem is not discussed in [NS 90] where the combination A2 + B3 was chosen for ATP, but let us note that this problem might have a different nature because their parallel composition is defined differently, i.e. like in ACP [BeK 84].

A2 + B4

This combination has no interesting expressive power since *B4* is only applicable when one of the alternatives cannot execute any action (even χ), whereas even *stop* may execute a χ action according to *A2*.

A1 + B2

It is shown in [Led 91] that *B [] B ~ B* is not preserved. Take B := ⌊stop⌋(b; Q).

A1 + B3 + ¬ B1 + ¬ B2 + ¬ B4

It is shown in [Led 91] that *B [] stop ~ B* is not preserved. Take B := ⌊stop⌋(b; Q).

Remaining possible combinations of inference rules

A1 + B1 or A1 + B4 or A1 + B1 + B3 or A1 + B3 + B4

We can see that *A2* and *B2* are naturally rejected.

Among the four remaining possibilities, we have selected *A1 + B3 + B4* for the following reasons.

The second case *A1 + B4* is not expressive enough since it only allows the passing of time when one process in the alternative behaves like stop.

In addition, rule *B1*, when combined with *A1*, means that time may resolve a choice; which is not intuitively appealing, e.g. if $P \xrightarrow{\chi} P'$ and $Q \xrightarrow{\chi} Q'$, then *P [] Q* may lead by χ either to *P'* or to *Q'*. The pair *A1 + B4* has not this drawback even if *B4* is very similar to *B1*. These considerations reject cases 1 and 3.

Therefore, it remains *A1 + B3 + B4*.

As a conclusion to this section, we may say that, on the basis of an ASAP principle (viz. χ has a lower priority than other actions **with respect to choice**) and the objective of an upward compatibility with standard LOTOS, we have been able to reject many inference systems for the choice operator. Theoretically, four of them remained. Finally, we have selected one of them on two bases: expressive power and intuitive understanding of time.

6. The proposed operational semantics of TLOTOS

Based on the preliminary study of section 5, our proposed TLOTOS is thus the following.

Stop

(S) No axiom for stop, i.e. axiom *A1* above, or stated otherwise $stop \xrightarrow{a} \forall\, a \in L^{i,\delta,\chi}$

Action prefix

(AP) $a; P \xrightarrow{a} P \quad (a \in L^{i})$

This semantics expresses the ASAP criterion, i.e. action *a* should occur before the next χ action since no χ action is enabled. As mentioned earlier, at the syntactic level, no χ action (as well as no δ action) may appear in an action-prefix construction. A χ action is transparently introduced however when one uses the timed operators presented in sections 3 and 4. This is similar to the introduction of a δ action by way of the *exit* construct.

Termination

(Ex) $exit \xrightarrow{\delta} stop$

This semantics also expresses the ASAP criterion, i.e. δ should occur before the next χ action since no χ action is enabled.

Idling

(Id) $idle \xrightarrow{\chi} idle$

This process, already introduced in PADS, plays the role of the *stop* process of ATP (where it is denoted δ). It may be considered as a shorthand notation for $\lfloor stop \rfloor^{\omega}$.

Choice

The choice operator has been discussed in length in section 5. The rules *B3* and *B4*, as well as the classical choice rules are reproduced hereafter for completeness.

(Ch1) $\dfrac{P \xrightarrow{a} P'}{P \,[]\, Q \xrightarrow{a} P'} \quad (a \in L^{i,\delta})$	(Ch3) $\dfrac{P \xrightarrow{\chi} P',\ Q \not\xrightarrow{a}\ \forall\ a \in L^{i,\delta,\chi}}{P \,[]\, Q \xrightarrow{\chi} P'}$
(Ch2) $\dfrac{P \xrightarrow{\chi} P',\ Q \xrightarrow{\chi} Q'}{P \,[]\, Q \xrightarrow{\chi} P' \,[]\, Q'}$	Plus the symmetric rules Ch1' and Ch3'

Parallel composition

The rules for parallel composition are the well-known rules, except that they have been extended to express transitions in the presence of χ actions. We have decided not to allow the introduction of a χ action in the list of gates of the parallel composition operator in order to keep χ actions hidden at the syntactic level (again this is also the case for δ). Formally, if γ is the list of gates of the parallel operator, we have the requirement that $\gamma \cap (\delta, \chi\} = \varnothing$. However, we have to decide whether the parallel composition enforces or not an implicit synchronization on χ actions. This is again related to the ASAP criterion: if one of the processes running in parallel may execute alone a χ action, this makes it impossible to impose that some actions of the other process occur ASAP, i.e. before this χ action. A way to achieve this is to enforce an implicit synchronization on χ, as for δ actions. Furthermore, this synchronization means that time must pass at the same pace in both processes; which is intuitively appealing.

(PC1) $\dfrac{P \xrightarrow{a} P'}{P\|[\gamma]\|Q \xrightarrow{a} P'\|[\gamma]\|Q} \quad (a \in L^{i} - \gamma)$	(PC2) $\dfrac{P \xrightarrow{a} P',\ Q \xrightarrow{a} Q'}{P\|[\gamma]\|Q \xrightarrow{a} P'\|[\gamma]\|Q'}\ (a \in \gamma \cup \{\delta, \chi\})$
Plus the symmetric rule PC1'	

Note that the parallel operator has the so-called "must-timing" semantics (in the sense of [QAF 89]). It can be argued that this semantics is counter-intuitive for the interleaving operator, and that a "may-timing" semantics (in the sense of [QuF 87]) is more intuitive. Let us first recall the usual argument against "must-timing" on the following example: consider the interleaving B ||| C, and suppose that B := b; B'. With a "must-timing" semantics, such as the TLOTOS semantics, if the environment does not offer b, then C will only be allowed to proceed until it comes to a state where a timed action is required to occur. Therefore, C is not independent from B because a deadlock of B may induce a deadlock of C at the next time instant. This is a priori counter-intuive because it violates the independence between B and C.

Let us try to explain why we are anyway in favour of a "must-timing" semantics. Let us first note that, even in standard LOTOS, processes B and C are not independent since they are required to synchronize on termination. In TLOTOS, this synchronization is stronger because it is extended to the point that B and C have to synchronize "periodically" on every timed action. Therefore, a special care is required in the design of B and C in order to avoid such unwanted propagation of local deadlocks from B (or C) to the whole combined process B ||| C. This is up to the designer to specify this or not. For instance, such deadlock of C by B is avoided if the designer accepts to specify the system as $\lfloor b; B' \rfloor(idle) \,|||\, C$: if action b does not occur before the next timed action then B becomes idle, and does not block C. This way of doing offers the possibility to simulate a "may-timing" behaviour on top of the basic "must-timing" semantics. We think that the opposite is not feasible.

In conclusion, a "must-timing" semantics is necessary in TLOTOS to express urgency of actions in composed processes. Of course, requiring such strong requirements may have dangerous consequences such as time deadlocks when the environment is not ready to participate in these urgent actions. But this is the normal price to pay. The designer may always specify a "may-timing" behaviour if it is the intended behaviour. The only drawback is an additional complexity in the resulting behaviour expression.

Hiding

The rules for hiding are the classical LOTOS rules. Again we require that no χ action be allowed in the list of hidden gates (as for i and δ) for two reasons. First, χ does not appear at the syntactic level, and second it would be counter-intuitive to hide the passing of time. Formally, if γ is the list of hidden gates, we have the requirement that: $\gamma \cap \{i, \delta, \chi\} = \varnothing$.

(H1) $\dfrac{P \xrightarrow{a} P'}{\text{hide } \gamma \text{ in } P \xrightarrow{a} \text{hide } \gamma \text{ in } P'}$ $(a \in L^{i,\delta,\chi} - \gamma)$	(H2) $\dfrac{P \xrightarrow{a} P'}{\text{hide } \gamma \text{ in } P \xrightarrow{i} \text{hide } \gamma \text{ in } P'}$ $(a \in \gamma)$

Enabling

For the enabling operator, a χ action is considered as another non-δ action, and the classical rules remain unchanged.

(En1) $\dfrac{P \xrightarrow{a} P'}{P >> Q \xrightarrow{a} P' >> Q}$ $(a \in L^{i,\chi})$	(En2) $\dfrac{P \xrightarrow{\delta} P'}{P >> Q \xrightarrow{i} Q}$

Disabling

The three usual rules *Di1*, *Di2* and *Di3* hereafter are the classical inference rules for disabling. *Di4* and *Di5* have been added to provide χ transitions and follow the same spirit as the choice operator in order to preserve the expansion theorem of the disabling operator.

(Di1) $\dfrac{P \xrightarrow{a} P'}{P [> Q \xrightarrow{a} P' [> Q}$ $(a \in L^{i})$	(Di4) $\dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{\chi} Q'}{P [> Q \xrightarrow{\chi} P' [> Q'}$
(Di2) $\dfrac{P \xrightarrow{\delta} P'}{P [> Q \xrightarrow{\delta} P'}$	(Di5) $\dfrac{P \xrightarrow{\chi} P', Q \xrightarrow{a} \not\quad \forall\ a \in L^{i,\delta,\chi}}{P [> Q \xrightarrow{\chi} P'}$
(Di3) $\dfrac{Q \xrightarrow{a} Q'}{P [> Q \xrightarrow{a} Q'}$ $(a \in L^{i,\delta})$	Plus the symmetric rule Di5'

Instantiation

(In) $\dfrac{P\,[g_1/h_1, \ldots g_n/h_n] \xrightarrow{a} P',\ Q\,[h_1, \ldots h_n] := P}{Q\,[g_1, \ldots g_n] \xrightarrow{a} P'}$ $(a \in L^{i,\delta,\chi})$

Timed operators

The timed operators have been defined in sections 3 and 4: *SD1, SD2, SD3, SD4, ED1, ED2, ED3, ED4, USD1, USD2* and *USD3*. It can be seen that:

- all the axioms and rules of LOTOS are preserved;
- when new axioms and rules are added, the operators are nevertheless defined such that no new transitions are created in the LTS **when the operands are untimed processes**.

These two criteria imply the upward compatibility: untimed TLOTOS processes have the same semantics as their corresponding (i.e. equal) standard LOTOS version.

In section 5, the semantics of *stop* and *choice* have been selected on an additional requirement: usual equivalence laws should be preserved in order to facilitate as much as possible the use of TLOTOS by those who were trained on standard LOTOS.

With these axioms and rules, it has been shown in [Led 91] that the usual strong bisimulation laws are preserved, e.g. P [] stop ~ P, P [] P ~ P, P |[γ]| (Q |[γ]| R) ~ (P |[γ]| Q) |[γ]| R, …

Some interesting laws related to the three timed operators are also provided in [Led 91].

7. Examples

We just provide two small examples of the use of the start delay and the unbounded start delay operators on the transmitter of the alternating bit protocol, as well as on a possible medium.

We first present the usual standard LOTOS specification (figure 5) and then the specification in TLOTOS.

In this example, *nat0* is a positive natural number, *sdu* is the service data unit that the protocol must transfer, *pdu* is an operation which builds a protocol data unit from a *sdu* and a *bit*, *bit* is a renaming of *bool* with *0* for *false*, *1* for *true* and *compl* for *not*.

```
Process Transmitter [in,s,r] (time:nat0) :noexit := T [in,s,r] (0 ofsort bit,time)
where process T [in,s,r] (b:bit,t:nat0) :noexit :=
            in?x:sdu; Send [s,r] (x,b,t) >> T [in,s,r] (compl(b),t)
      where process Send [s,r] (x:sdu,b:bit,t:nat0) :exit :=
                  s!pdu(x,b);
                  (r?ack:bit [ack=b]; exit
                  [] r?ack:bit [ack ne b]; Send [s,r] (x,b,t)
                  [] (Time-out (t) >> Send [s,r] (x,b,t)))
                  where process Time-out (t:nat0) :exit := exit endproc
                endproc (* Send *)
        endproc (* T *)
endproc (* Transmitter *)
```

Figure 5: The transmitter in standard LOTOS

The difference between this specification and the next one (figure 6) is twofold.

```
Process Transmitter [in,s,r] (time:nat0) :noexit := T [in,s,r] (0 ofsort bit,time)
where process T [in,s,r] (b:bit,t:nat0) :noexit :=
            ⌊in?x:sdu; Send [s,r] (x,b,t)⌋^ω >> T [in,s,r] (compl(b),t)
      where process Send [s,r] (x:sdu,b:bit,t:nat0) :exit :=
                  s!pdu(x,b);
                  ⌊r?ack:bit [ack=b]; exit [] r?ack:bit [ack ne b]; Send [s,r] (x,b,t)⌋^t
                        (Send [s,r] (x,b,t))
                endproc (* Send *)
        endproc (* T *)
endproc (* Transmitter *)
```

Figure 6 : The transmitter in TLOTOS

First, in TLOTOS we have to disable at some places the ASAP criterion on the occurrence of actions, e.g. when the transmitter is waiting at *in* for a sdu to transfer. Without the introduction of an unbounded time delay, the action *in* would be required to occur before the next χ action. This is not reasonable because we do not know anything about the behaviour of the external user who can deliver its sdu's at any time. The unbounded delay operator allows the transmitter to idle until the matching at *in* between this user and the transmitter, which is the expected behaviour in this case.

Second, we have to quantify correctly the time-out mechanism by replacing the imprecise process *time-out* by an adequate start delay operator. The overall structure is preserved: the branch *time-out (t) >> send* is just (simplified and) placed as a second argument of the start de-

lay operator, while the other alternatives are embedded in the first argument of this operator. The parameter *t* is then simply added to quantify the time-out delay.

It can be seen that the action at *s* is not embedded in an unbounded start delay operator. This is because we want to enforce that this sending occur immediately (i.e. before the next χ action) after the reception at *in* of a sdu. This should be used with care, because it may create unexpected time deadlock when the transmitter is synchronized with the medium. However, when designing a protocol, one knows exactly the underlying medium (or service), and we suppose here that this medium is always in a state where it may receive the pdu.

The ASAP principle is very useful to model that no time passes for instance between an *r* action and the subsequent *exit* or call to process *send*, as well as during the relay between *send* and *T*.

As another small example, we provide the specification of a unidirectional underlying medium (figure 7), which is always ready to receive, may loose pdu's and introduces a constant delay *d*.

```
Process Medium [in,out] (d:nat0) :noexit :=
⌊in?x:pdu; exit⌋^ω >>
        (⌊i; idle⌋^d (out!x; idle)
          |||
         ⌊stop⌋ (Medium [in,out] (d)))
endproc (* Medium *)
```

Figure 7: A medium in TLOTOS

In this specification, the medium cannot accept two successive *in* actions within the same unit of time (see last line of the specification). This choice is arbitrary, but illustrates a means to preserve the FIFO ordering of pdu's. Of course, as a consequence, this simple medium does not allow more than *d* pdu's in transit.

Again, the action *out* is required to occur immediately after the delay *d*, which may be problematic if the receiver is not ready to receive at any time. In this case, *out* should be embedded in an unbounded delay operator as usual.

Finally, let us note the use of process *idle*. It cannot be replaced by *stop* because this would create a time deadlock. This is due to the semantics of the parallel composition which requires that all processes in parallel execute their χ actions simultaneously in order to proceed.

8. Comparison with other works

In this section, we will briefly compare our proposal with previous works on timed extensions in several process algebras, including LOTOS. We will start with the approaches which are the closest to ours (ATP [NS 90], TPL [HeR 90b], $ACP_{\tau\varepsilon}^{t}$ [Gro 90], TCCS [MoT 89], PADS [Azc 90], Timed CSP [ReR 88, Ree 90], Timed CCS [Wan 90, Wan 91] and TPCCS [Han 91]), and compare the differences between the basic choices in the semantics of the operators (see table below).

The LOTOS parallel composition operator is specific, and our work cannot be strictly compared with others. For instance, in TPL and TCCS, the parallel operators are binary. In ATP and $ACP_{\tau\varepsilon}^{t}$ they are multiway but with a different semantics, viz. the ACP one.

Some other works are also worth mentioning here even if they are based on other principles: ACP_{ρ} [BaB 89], TIC [QuF 87, QAF 89] and Timed-Interaction LOTOS [BLT 90]. All these models have chosen to work with timestamps associated to the actions, either directly in the language syntax or at the semantic level.

Our options	Choices similar to ours	Choices different from ours
Presence of a distinct timed action χ	σ in TPL, χ in ATP and TPCCS, t in $ACP_{\tau\varepsilon}^{t}$ and PADS	No equivalent in TCCS, Timed CCS and Timed CSP which are based on delays
Timed action not allowed in action-prefix	ATP, TPCCS	Allowed in TPL, $ACP_{\tau\varepsilon}^{t}$, PADS
Stop is a time deadlock	TCCS, $ACP_{\tau\varepsilon}^{t}$, PADS	in ATP, TPL, Timed CCS and TPCCS, *stop* (or *NIL*) is like our *idle*.
ASAP Action-prefix	ATP, TCCS, $ACP_{\tau\varepsilon}^{t}$, synchronous actions of PADS	Only internal actions in TPL, Timed CCS, Timed CSP and TPCCS occur ASAP
Choice is deterministic w.r.t. time	ATP, TPL, "+" in TCCS[1], Timed CSP, Timed CCS, PADS and TPCCS	Nondeterministic in $ACP_{\tau\varepsilon}^{t}$
Non-urgent actions via $\lfloor\rfloor^{\omega}$	$\lfloor\rfloor^{\omega}$ in ATP	δ in TCCS, "nolimit value" in PADS, implicit in TPL, Timed CCS, Timed CSP and TPCCS, no equivalent in $ACP_{\tau\varepsilon}^{t}$

9. Conclusion

We have presented TLOTOS which is an upward compatible extension of LOTOS. Its operational semantics has been given. The use of TLOTOS has been illustrated on two small examples which prove that TLOTOS is powerful and easy to use. LOTOS specifications may be easily adapted to TLOTOS when one wants to formalize some time-dependent parts which were specified in an imprecise way in LOTOS.

Finally, we would like to recall all the basic choices of TLOTOS:

- The existence of a distinct action χ to model the passing of one unit of time. Therefore, time is discrete and abstract.
- The ASAP principle on non-χ actions in an action-prefix construction, which means that these actions should occur before the next χ action.
- The determinism of the choice operator with respect to time, which means that χ actions cannot resolve a choice.
- The absence of χ actions at the syntactic level, i.e. as a first argument of an action-prefix or in the gate lists of the parallel composition or hiding operators.
- The introduction of timed behaviours by way of three timed operators instead of timestamps in actions.
- The synchronization on χ actions imposed by the parallel composition, in order to model that time progresses at the same pace in all the components of a system.
- The priority of the outermost χ action in nested delay operators.
- The disabling of all levels of execution delay operators by way of δ.
- The timelock behaviour of *stop*, and the introduction of a new *idle* process which allows time to pass.

One may argue that the modelling of time by a distinct action which is not always enabled is counter-intuitive, since this means that time may be stopped at some places when actions cannot proceed, e.g. *a; P* may stop time if action *a* cannot be matched by the environment. This prob-

[1] In TCCS, $\oplus$ is closer to our $\lfloor\rfloor$ operator

lem may yet be presented in another way. If a process proposes a long (possibly infinite) sequence of actions before offering a χ action by way of one of the timed operators, this can be considered as a non implementable system, because the execution of so many actions is impossible in one unit of time. In our model, where actions take no time, this is of course a non problem, but this approximation may be questionable. In conclusion, time is considered here as an abstract time (by contrast to physical time). It is a convenient assumption at a conceptual level, and timed actions should be considered here as an abstract time reference which is **not** strongly bound to a physical time by a kind of "periodic" sequence of clock ticks.

TLOTOS expresses naturally urgency on observable **actions**, but cannot express adequately urgency on observable **interactions**, i.e. that an action should occur as soon as all the partners involved in an interaction are ready to do so. This problem is presented in depth in [BLT 90] and considered as a basic expressive power limitation. Note that this is **not** a lack of (theoretical) expressive power, but a lack of flexibility. Theoretically, there is no difference between an observable action and an observable interaction in LOTOS. However expressing urgency on interactions is closely related to the ability of expressing urgency in a modular way, which is essential in practice. The more general approach we know of has been proposed in Timed-Interaction LOTOS [BLT 90] where an "asap" operator (and the more general "timer" operator) is defined which solves this problem and allows the expression of urgency on interactions. This lack of flexibility in TLOTOS to express urgency on interactions is probably its main shortcoming. A simple way to add this flexibility in TLOTOS would be to include this "asap" operator. However we are currently working on another kind of extension.

Section 7 illustrates how a standard LOTOS specification (containing imprecise and unquantified timed behaviours) may be rewritten in TLOTOS. It appears that in many places, the LOTOS "action-prefix" needs to be changed into a careful combination of TLOTOS (asap) "action-prefix" and the unbounded start delay operator. This is somehow unsatisfactory because we would like to keep the major part of a standard LOTOS specification unchanged. This problem is also under study in the above-mentioned extension of TLOTOS.

Finally, let us recall that, in [Led 91], we have proved that the operational semantics of TLOTOS is consistent, and that strong bisimulation is a congruence. Examples of equivalence laws associated with the three timed operators, and the details of the rejection of the combination A2 + B3 in section 5 are also included in this report.

Acknowledgements

I am grateful to the anonymous referees for their judicious comments.

References

[AuB 84] D. Austry, G. Boudol, *Algèbre de Processus et Synchronisation,* Theoretical Computer Science 30 (1984) 91 - 131 (North-Holland, Amsterdam).

[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.

[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).

[BoL 91] T. Bolognesi, F. Lucidi, *LOTOS-like process algebras with urgent or timed interactions,* in: K. Parker, G. Rose, eds., FORTE'91 (North-Holland, Amsterdam, 1992).

[Car 82] L. Cardelli, *Real Time Agents,* in: M. Nielsen, E.M. Schmidt, eds., Automata, Languages and Programming (LNCS 140, Springer-Verlag, Berlin Heidelberg New York, 1982) 94-106.

[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 90] 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).

[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., FORTE '90, Madrid, Spain, Nov. 90 (to be published by North-Holland, Amsterdam, 1991).

[HTZ 90] W. van Hulzen, P. Tilanus, H. Zuidweg, *LOTOS Extended with Clocks,* in: S. T. Vuong, ed., FORTE '89 (North-Holland, Amsterdam, 1990).

[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 91] G. Leduc, *On the design and properties of TLOTOS,* Rept. No. S.A.R.T. 91/04/13, Université de Liège, Dept. Systèmes et Automatique, B28, 4000 Liège, Belgium, May 1991.

[Mil 83] A.J.R.G. Milner, *Calculi for Synchrony and Asynchrony,* Theoretical Computer Science, Vol. 25, No. 3, July 1983, 267-310 (North-Holland, Amsterdam).

[Mil 85] G. Milne, *CIRCAL and the Representation of Communication, Concurrency and Time,* ACM Transactions on Programming Languages and Systems, Vol. 7, No. 2, April 1985, 270-298.

[MoT 89] F.Moller, C. Tofts, *A temporal calculus of communicating systems,* Rept. No. ECS-LFCS-89-104, University of Edinburgh, Department of Computer Science, Edinburgh, Dec. 89.

[MRF 91] J.M. Martin Espinosa, J. M. Robles Roman, L. Fuertes Prieto, *Concurrent Modelling in LOTOS as a solution to real time problems,* in: J. Quemada, J. Mañas, E. Vazquez, eds., FORTE '90, Madrid, Spain, Nov. 90 (to be published by North-Holland, Amsterdam, 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).

[NS 90] X. Nicollin, J. Sifakis, *The Algebra of Timed Processes ATP: Theory and Application,* Rept. No. RT-C26, Projet Spectre, LGI-IMAG, Dec. 1990.

[NS 91] X. Nicollin, J. Sifakis, *An Overview and Synthesis on Timed Process Algebras,* in: K.G. Larsen, ed., Computer-Aided Verification, III, Aalborg, Denmark, July 1991.

[NSY 91] X. Nicollin, J. Sifakis, S. Yovine, *From ATP to Timed Graphs and Hybrid Systems,* in: REX Workshop "Real-Time: Theory and Practice", Mook, The Netherlands, June 1991.

[QAF 89] J. Quemada, A. Azcorra, D. Frutos, *A timed calculus for LOTOS,* in: S. T. Vuong, ed., FORTE '89, Vancouver, Canada, Dec. 89 (North-Holland, Amsterdam, 1990).

[QuF 87] J. Quemada, A. Fernandez, *Introduction of Quantitative Relative Time into LOTOS,* in: H. Rudin, C.H. West, eds., Protocol Specification, Testing and Verification, VII, (North-Holland, Amsterdam, 1987, ISBN 0-444-70293-8) 105-121.

[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).

[Tof 88] C. Tofts, *Temporal Orderings for Concurrency,* Rept. No. ECS-LFCS-88-49, University of Edinburgh, Department of Computer Science, Edinburgh, April 88.

[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, ISBN 3-540-53048-7) 502-520.

[Wan 91] Y. Wang, *CCS + Time = an Interleaving Model for Real Time Systems,* in: ICALP'91, Madrid, Spain, July 1991.