# Model-checking for real-time systems specified in Lotos *

N. Rico †, G.v. Bochmann † and O. Cherkaoui ‡

† Département d'informatique et de recherche opérationnelle
Université de Montréal
‡ Département d'informatique et de mathématiques
Université du Québec à Montréal

**Abstract.** This paper aims at describing and analyzing concurrent systems whose behavior is dependent on explicit time delays. The formal description technique Lotos [Loto 89] is extended with time intervals in the following way: actions in Lotos must occur at a time $t$ within a given interval $[t_{min}, t_{max}]$ relative to the previous action executed by the process. The syntax and semantics of Time Interval Lotos is given. The model is defined as a labelled transition systems with clocks associated with states and timing conditions associated with transitions. The labelled transition system derived corresponds to a timed graph model [Alur 90]. The logic TCTL (Computation Tree Logic with time) which allows quantitative operators in the formulas can be used to specify assertions. Model-checking is used to determine the truth of a TCTL-formula with respect to a labelled transition system derived from the Time Interval Lotos specification. We illustrate the approach by a simple example. We also present an alternative approach for verifying timing properties. A labelled transition system with time intervals is derived. This graph does not represent the precise evolution of the system in time. Each transition is labelled with an action and a time interval showing the range of possible time occurrences for the action.

## 1 Introduction

With the proliferation of computer-communication networks and the increasing importance of distributed processing, researchers have worked intensively on the modelling of distributed systems. Formal description techniques such as Lotos [Loto 89, Bolo 87], Estelle [Budk 87] and SDL [SDL 87] have been developed to describe OSI (Open Systems Interconnection) communication protocols and services as well as other kind of distributed systems. Those specification formalisms and the verification methods, however, make abstraction from the quantitative aspect. But for certain type of systems such as real-time systems, the specification methods and verification techniques should incorporate time

* Work supported by Bell Northern Research, the Ministry of Education of Quebec and the IDACOM-NSERC-CWARC Industrial Research Chair on Communication Protocols at the University of Montreal

values since the correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced.

This paper addresses the issue of correctness of a system which is specified with a formal description technique and where actual time values are included in the specification. The idea is to check whether the specified system satisfies a number of desirable properties that include time. Properties can be formulated in a temporal logic formalism augmented with quantitative time. If the system has a finite number of states, then we can use a model-checking approach which is an algorithmic method for verifying automatically those properties. It consists of checking that a given state graph derived from the formal description of the system satisfies a given temporal formula.

The formal description technique we consider in this paper is a variant of Lotos defined in Section 2, called Time Interval Lotos, which adds time intervals of the form $[t_{min}, t_{max}]$ to Lotos actions. An action $a$, once enabled, cannot occur before time $t_{min}$ and must occur before $t_{max}$ time has elapsed since its enabling, unless it is disabled by the occurrence of another action. The semantics is defined in terms of a labelled transition system which includes clocks associated to states and timing enabling conditions associated to transitions. Temporal formulas are expressed in the logic TCTL [Alur 90], which is an extension of CTL with continuous time. The time assertions written in TCTL are checked against the labelled transition system with clocks and timing conditions generated from the Timed Interval Lotos specification. We also present in Section 5 an alternative approach for verifying timing properties. A labelled transition system with time intervals is derived. This graph shows all the possible paths of timed actions but does not represent the precise evolution of the system in time. Each transition is labelled with an action and a time interval showing the range of possible time occurrences for the action. The contribution of the paper is hence, on the first hand, the definition of an extension of Lotos with Time Intervals and the derivation of a labelled transition system on which model-checking can be performed, and, on the second hand, the presentation of an alternative approach to verify timing properties.

**Related work:** Process algebras such as CCS [Miln 80], CSP [Hoar 85] and ACP [Berg 84] have been extended with timing characteristics. Nicollin and Sifakis [Nico 91] presented an overview of existing results about timed process algebras. Quemada et al. [Quem 89] proposed a time extension of Lotos where an occurrence time is associated with each action: this time indicates the global time when the action must occur. Bolognesi et al. [Bolo 90b] define a time extension of Lotos which offers operators for specifying the urgency of a specified action. Their model is similar to the Time Petri Net of Merlin and Farber [Merl 76]. In [Rico 91], the model considered is an extension of Lotos with actions that have an associated specific time of occurrence and weights associated in the case of a probabilistic choice. The goal was to predict the performance of distributed systems. The model proposed in this paper considers an extension of Lotos with time intervals associated with actions: time intervals are better suited for the verification of timing assertions. Emerson et al. [Emer 89] defined an extension of

CTL (RTCTL) with discrete time which can be used to specify and verify hard deadlines. Hansson [Hans 91] also extended CTL with time and probability. His temporal logic formulas are interpreted over a labelled transition system derived from a modification of CCS which includes discrete time and probabilities. Alur, Courcoubetis, and Dill [Alur 90] also proposed an extension to CTL, but in their logic (TCTL) formulas are interpreted over models with continuous time. They introduce the concept of a timed graph to model a finite-state real-time system: the system is equipped with a finite set of clocks which record the time elapsed since they were reset. They developed a model checking algorithm for determining the truth of a TCTL formula with respect to a timed graph. Lewis [Lewi 90] presented a variation of CTL with continuous time that is interpreted over a finite-state model in which the time delays between events are constrainted to fall between upper and lower integer time bounds.

The paper is organized as follows. In the next section, we describe a semantics for Time Interval Lotos. In Section 3, the branching time logic with time TCTL is presented as well as the model-checking method for checking TCTL-formulas in respect to a given Time Interval Lotos specification. In Section 4, a small example is given. In Section 5, a method for enumerating all possible timed paths is described. Finally, we discuss the modelling approach in the conclusion.

## 2 Lotos with Time Intervals

Time Interval Lotos is a variant of Lotos (Language Of Temporal Ordering Specification) [Loto 89, Bolo 87], which adds time intervals to Lotos actions. In this section, we will present Lotos and define our timing interval extension.

### 2.1 Lotos

Lotos [Loto 89, Bolo 87] is an algebraic specification language based on CCS [Miln 80]. Lotos is defined in terms of processes and uses rendez-vous interactions. The rendez-vous may involve two or more processes and occurs at an interaction point called a gate: it happens when all Lotos processes coupled to the gate are ready for that interaction.Interaction and process parameters are described by ACT ONE [Ehri 85] abstract datatype definitions. We are interested in the basic Lotos language which does not include the interaction and process parameters. The operations of basic Lotos are the following, where $B, B_1, B_2$ are behavior expressions, $a$ is an action and $g_1, \ldots, g_n$ are gate identifiers:

| | | | |
|---|---|---|---|
| Inaction | stop | Process instantiation | $p[g_1, \ldots, g_n]$ |
| Action prefix | $a; B$ | Parallel composition | $B_1 \|[g_1, \ldots, g_n]\| B_2$ |
| Choice | $B_1 [] B_2$ | Pure Interleaving | $B_1 \|\|\| B_2$ |
| Termination | exit | Relabeling | $p[g_1/g'_1, \ldots, g_n/g'_n]$ |
| Enabling | $B_1 >> B_2$ | Hiding | hide $g_1, \ldots, g_n$ in B |
| Disabling | $B_1 [> B_2$ | | |

## 2.2 Time Interval Lotos: Presentation of the Model

This section describes LOTOS which has been enhanced with time intervals of the form $[t_{min}, t_{max}]$ where $t_{min}$ and $t_{max}$ are natural numbers. Those intervals are associated with each action $a$ and are relative to the moment at which the previous action within the same process was executed. When the previous action is executed, we say that action $a$ is "locally" enabled or that action $a$ is enabled due to the execution of a previous "local" action of the process. An implicit global clock exists in the transition system. Assuming that the previous local action has been executed at a global time $g_t$, action $a$ cannot fire before time $g_t + t_{min}$ and must fire before or at time $g_t + t_{max}$ unless it is disabled by the occurrence of another action. The time domain is represented as real numbers.

Time Interval Lotos assumes that the time intervals associated with the actions have a local meaning. This is the same in Quemada's model [Quem 89]: every action is assigned a single time stamp which indicates the exact time at which the action shall happen relative to the previous local action. A time choice construct also exists for representing the occurrence of an event at an unspecified instant of time out of a given set. The difference between Time Interval Lotos and Quemada's model resides in the definition of the labelled transition system (LTS): the LTS derived in Quemada's model contains transitions with their time of occurrence whereas the LTS derived for Time Interval Lotos has clocks and timing conditions and can therefore deal with time intervals. The other main difference between the two models is that in Quemada's model, some transitions may not be derived due to the timing relation between the different components whereas the LST derived for our model is the same LTS derived by the usual Lotos inference rules with additional timing constraints.

In the model of Bolognesi et al. [Bolo 90b], the time intervals associated with actions have a global meaning, which makes the model semantically very different from ours. An action is enabled when all the processes participating in the action are ready to interact. The model of Bolognesi expresses the urgency of actions, i.e. the fact that something happens as soon as all the processes are ready for it. Bolognesi's model has more expressive power since it can simulate a Turing machine [Bolo 90b]. It can easily model the situation where two processes must synchronize after each one independently executes an action with unbounded delay (i.e. with interval $[0, \infty]$). This situation is not modeled adequately by models in which actions have a local meaning. However, in the case where two processes must synchronize after only one of them executes an action with unbounded delay, then the two types of models are as expressive. This second case occurs more frequently in real examples hence the limitation mentionned in the case of symmetric unbounded delay is not often encountered. One disavantage of the approach of Bolognesi is that there are more properties which are undecidable whereas our model can be more easily analyzed.

## 2.3 Time Interval Lotos Semantics

The syntax of Time Interval Lotos is the same as standard Lotos except for the action prefix: instead of writing $a; B$ for an action followed by behavior $B$, we

write $a[t_{min}, t_{max}]; B$ which means that action $a$ must occur at a time $t$ in the interval $[t_{min}, t_{max}]$.

The model of Time Interval Lotos is a labelled transition system as defined by the Lotos operational semantics. Each system has a finite set of clocks $Cks$. We associate with every behavior expression a subset $C$ of $Cks$. $C$ is a set of clock identifiers corresponding to the clocks which are reset to 0. We associate with each interaction offer a clock which is a fictitious component that keeps track of the possible time at which the event can occur. For example, in the expression $a; b; stop|[b]|b; c; stop$, we associate a clock with $a$, with $c$, with offer $b$ in the left side of the parallel expression and with offer $b$ in the right side of the parallel expression. There are as many clock identifiers as there are interaction offers in the system. We use the following notation for the clock identifier: $c_{ai}$ is the clock identifier of offer $a$ where $i$ is a natural number. In the case of an action that is not involved in a rendez-vous, the clock identifier is simply noted $c_a$. Transitions are labelled with actions and with an enabling condition which is built using the boolean connectives over the formulas of the form $x[t_{min}, t_{max}]$, where $x \in Cks$, and $t_{min}, t_{max} \in N$. An action $a$ with enabling condition $\tau(a) = x[t_{min}, t_{max}]$ is possible if the value of clock identifier $x$ is in the interval $[t_{min}, t_{max}]$.

**Definition (Model of Time Interval Lotos behavior: Timed graph)** For a given behavior $B_0$, the model of Time Interval Lotos behavior is a labelled transition system $< Cks, S, A, TR, s_0 >$, where :

- $Cks$ is a finite set of clocks.
- $S = \{B_C\}$ is the set of all possible states, represented by behaviors, to which we associate a set of clocks that are reset to 0 when the state is entered.
- $A = \{a\ \tau(a) | a \in L(B) \cup \{i\}\}$ is the set of all possible actions. $L(B)$ is the alphabet of actions and $i$ is the internal action. $\tau(a)$ is a function that associates with each edge an enabling condition built using the boolean connectives over the formulas of the form $x[t_{min}, t_{max}]$, where $x \in Cks$, and $t_{min}, t_{max} \in N$.
- $TR \subset S \times S$ is a set of relations $-a\,\tau(a) \rightarrow$ defining the pairs of states associated with action $a$. We write $B_1 - a\ \tau(a) \rightarrow B_2$ iff $< B1, B2 > \in TR$.
- $s_0 = B_0\, C_0$ is the initial state where $C_0$ is the set of clocks corresponding to actions initially enabled.

The timed graph obtained resembles the timed graph of Alur et al. [Alur 90] except that it does not include the proposition truth value assignments to states. The semantic rules of Lotos operators [Loto 89] have to be redefined in order to associate timing conditions with transitions and a set of clocks with behavior expressions. The rules for some of the operators are the following:

**Action with Time Interval:** The timing condition associated with interaction offer $ai$ for gate $a$ states that the value of clock $c_{ai}$ must be in $[t_{min}, t_{max}]$.

$$(a[t_{min}, t_{max}]; B)_C - a\ c_{ai}[t_{min}, t_{max}] \rightarrow B_{C'}$$
$$where\ C' = \{c_{bj} \mid \exists B'\ such\ that\ B_{C'} - b\ \tau(b) \rightarrow B'_{C''}\}$$

**Choice** $B_1 [] B_2$:

$$\text{C1.}\ \frac{B_{1C_1} - a_1\ \tau(a_1) \rightarrow B'_{1C'_1}}{(B_1 [] B_2)_C - a_1\ \tau(a_1) \rightarrow B'_{1C'_1}}$$

C2. *symmetric of rule C1*

**Parallelism** $B_1 |[G]| B_2$ :

$$P1.\ \frac{B_{1C_1} - a\ \tau_1(a) \rightarrow B'_{1C'_1} \quad B_{2C_2} - a\ \tau_2(a) \rightarrow B'_{2C'_2}}{(B_1 |[G]| B_2)_C - a\ (\tau_1(a)\ and\ \tau_2(a)) \rightarrow (B'_1 |[G]| B'_2)_{C'_1 \cup C'_2}}\ a \in G$$

$$P2.\ \frac{B_{1C_1} - a\ \tau(a) \rightarrow B'_{1C'_1}}{(B_1 |[G]| B_2)_C - a\ \tau(a) \rightarrow (B'_1 |[G]| B_2)_{C'_1}}\ a \notin G$$

P3. *symmetric of rule P2*

**Process instantiation** $B = p[g_1, g_2, \ldots, g_n]$ :

$$\frac{Bp[g_1/h_1, \ldots, g_n/h_n]_C - a\ \tau(a) \rightarrow B'_{C'}}{B_C - a\tau(a) \rightarrow B'_{C'}}$$

where $B_P$ represents the body of the definition of process $p$, $(g_1, \ldots, g_n)$ is a list of formal gates, $(h_1, \ldots, h_n)$ is a list of actual gates, $[g_1/h_1, \ldots, g_n/h_n]$ is the relabeling postfix operator, (gate $g_i$ becomes gate $h_i$ *for* $i = 1, \ldots, n$). The semantic rules of the other operators are defined in [Rico 92].

## 2.4 Example

The following example illustrates a timeout situation. This example is semantically the same as the one described in [Bolo 90a].
$P[a, b, c] = Q[a, b] |[b]| R[b, c]$
$Q[a, b] = a[0, \infty]; b[0, 0]; Q[a, b]$
$R[b, c] = b[0, \infty]; R[b, c] [] c[10, 10]; stop$

Let the clock identifier for actions $a, c, b$ (in process $Q$) and $b$ (in process $R$) be respectively $c_a, c_c, c_{b1}, c_{b2}$. $C$ is the subset of the clocks reset to 0. The labelled transition system for this example is the following:

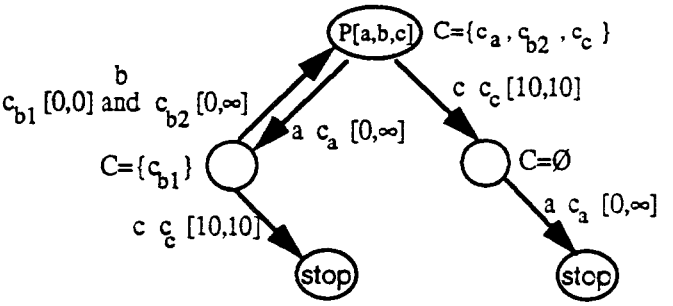


Figure 1. Labelled transition system of P[a.b,c]

# 3 Model Checking using Time Computation Tree Logic

In this section, we will describe the temporal logic TCTL [Alur 90] and explain how to perform model checking for Time Interval Lotos specifications.

## 3.1 Time Computation Tree Logic (TCTL)

Computation Tree Logic (CTL) is a branching time logic that was introduced by Emerson and Clarke [Clar 83] as a specification language for finite-state systems. Alur, Courcoubetis and Dill [Alur 90] proposed an extension of CTL with continuous time.The resulting logic is called TCTL and is interpreted over continuous computation trees, i.e. trees in which paths are maps from the set of nonnegative reals to system states. The syntax of TCTL is the following:

**Definition [TCTL Syntax]**

Let $P$ be the set of atomic propositions. The TCTL formulas are inductively defined as follows: $\phi := p \mid \text{false} \mid \phi_1 \rightarrow \phi_2 \mid \exists\, \phi_1 U_{\sim c}\, \phi_2 \mid \forall\, \phi_1 U_{\sim c}\, \phi_2$

where $p \in P, c \in N, \phi, \phi_1$ and $\phi_2$ are TCTL-formulas, and $\sim$ stands for one of the binary relation $<, \leq, =, \geq$ *or* $>$.

Intuitively $\exists \phi_1 U_{<c} \phi_2$ ($\forall \phi_1 U_{<c} \phi_2$) means that for some (all) path(s), there exists an initial prefix of time length less than $c$ such that $\phi_2$ holds at the last state of the prefix and $\phi_1$ holds at all the intermediate states.

The semantics is defined with respect to a structure $M = (S, \mu, f)$, where $S$ is a set of states, $\mu: S \rightarrow 2^P$ gives an assignment of truth values to propositions in each state, and $f$ is a map giving for each $s \in S$ a set of dense paths starting at that state. $f$ satisfies: $\forall s \in S, \forall \rho \in f(s), \forall t \in R, \rho_t f[\rho(t)] \subseteq f(s)$ where $\rho_t$ is the prefix of path $\rho$ upto time $t$ and $\rho(t)$ is a state corresponding to time $t$.

**Definition [TCTL Semantics].**

Let $p \in P$, $c \in N$ and $\sim$ stands for one of the binary relation $<, \leq, =, \geq$ or $>$. For a structure $M = (S, \mu, f)$, a state $s \in S$ and a formula $\phi$, the satisfaction relation $(M, s) \models \phi$ is defined inductively as follows:

$(M, s) \not\models \text{false}$

$(M, s) \models p$ iff $p \in \mu(s)$

$(M, s) \models (\phi_1 \rightarrow \phi_2)$ iff $s \not\models \phi_1$ or $s \models \phi_2$

$(M, s) \models \exists \phi_1 U_{\sim c} \phi_2$ iff for some $\rho \in f(s)$, for some $t \sim c$, $\rho(t) \models \phi_2$, and for all $0 \leq t' < t, \rho(t') \models \phi_1$.

$(M, s) \models \forall \phi_1 U_{\sim c} \phi_2$ iff for all $\rho \in f(s)$, for some $t \sim c$, $\rho(t) \models \phi_2$, and for all $0 \leq t' < t, \rho(t') \models \phi_1$.

A TCTL-formula $f$ is called satisfiable iff there is a structure $M$ and a state $s$ such that $(M, s) \models \phi$.

## 3.2 Model-Checking

Model-checking is a method for verifying concurrent systems in which a given state graph of the system behavior is compared with a given temporal logic

formula. It is one of the most successful techniques for automatically checking that a given temporal formula, written in propositional temporal logic, is satisfied by a state-transition graph that represents the actual behavior of the system. One of the advantages of the method is its efficiency. Model-checking is linear in the product of the size of the structure and the size of the formula when the logic is the branching-time temporal logic CTL. With time values, the complexity of the model checking algorithm using TCTL is exponential in the number of clocks and the length of the timing constraints, but linear in the size of the state-transition graph and the length of the formula [Alur 90].

In the model-checking approach for Time Interval Lotos, the idea is to construct the timed graph from the Lotos specification and to add the proposition truth value assignments to the states of this timed graph. The generation of this timed graph is explained in Section 2. We add to this timed graph a labeling function $\mu : S \rightarrow 2^P$ which assigns to each state the set of atomic propositions true in that state. The resulting structure is a timed graph in the sense of [Alur 90] and is the structure used for model-checking. Properties to be verified are written in TCTL. User-defined TCTL-formulas are checked against this structure using the algorithm of Alur et al. [Alur 90].

# 4 Example: Stop and Wait Protocol

## 4.1 Specification of the Protocol

In this section, we demonstrate the modeling approach described above by presenting the stop-and-wait protocol, which is a simplified version of the alternating bit protocol. This protocol uses two types of messages : information (info) frames and acknowledgement (ack) frames. The transmitter sends an info frame and waits for an ack frame from the receiver. The medium is unreliable in both directions. The specification of the protocol is the following:

```
specification stop-and-wait [get, give] : noexit
behaviour
hide sendinfo, recinfo, sendack, recack in
     ((transmitter[get,sendinfo,recack] |[]| receiver[give,sendack,recinfo])
     |[sendinfo,recinfo,sendack,recack]|
     medium[sendinfo,recinfo,sendack recack] )
where process transmitter[get,sendinfo recack]: noexit :=
     get [0, 1]; sendinfo [0, 1]; sending[get,sendinfo,recack]
where process sending[get,sendinfo,recack] : noexit :=
     recack [0, 2];transmitter[get,sendinfo,recack]
     []i[10, 10];sendinfo[0, 1];sending[get,sendinfo,recack] (*timeout*)
     endproc (*sending*)
endproc (*transmitter*)
process receiver[give,sendack,recinfo]: noexit :=
     recinfo[0, ∞];(give[0, 1];(sendack[0, 1][]i[0, 1]); receiver[give,sendack,recinfo]
                    []i[0, 1] receiver[give,sendack,recinfo])
```

```
endproc (*receiver*)
process medium[sendinfo,recinfo,sendack,recack] : noexit :=
      sendinfo[0, ∞]; (recinfo [0, 2];medium[sendinfo,recinfo,sendack,recack])
                      []( i[0, 2];medium[sendinfo,recinfo,sendack,recack]) )
[]    sendack[0, ∞]; (recack [0, 2];medium[sendinfo,recinfo,sendack,recack])
                      []( i[0, 2];medium[sendinfo,recinfo,sendack,recack]) )
endproc (*medium*)
endspec (*stop-and-wait*)
```

The global behavior is determined by applying the inference rules described in Section 2. The following graph shows the labelled transition system with clocks and timing conditions, where $i(s), i(m)$ and $i(r)$ denote the internal action associated with process sending, medium and receiver, respectively. For readability, we did not indicate the timing condition associated with the internal actions: they are $ci(s)[10, 10], ci(r)[0, 1], ci(m)[0, 2]$.

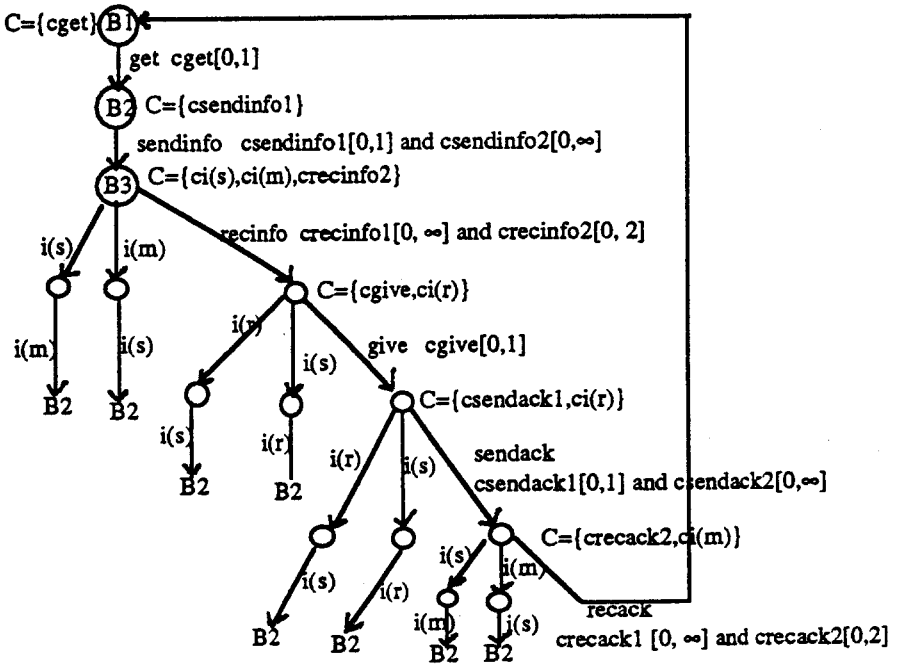


Figure 2. Timed graph of the stop-and-wait protocol

## 4.2 Model-Checking

One property that may be of interest is that for each *sendinfo* action there is always a subsequent *recack*. For an open system where we do not make any

assumptions about the environment, this property does not hold since there is a possibility that the environment will never attempt to communicate a *get* or a *give*. On the other hand, in a closed system, we can verify the following hard deadline that specifies that a *recack* will appear within 10 time units. The closed system is obtained by the following specification, where the user is always prepared to engage in the actions *get* or *give*:

hide |[get,give]| in stop-and-wait [get, give] |[get,give]| user[get, give]
where process user[get,give]: noexit :=
get[0,1]; user[get,give] [] give[0,1]; user[get give]
endproc (*user*)

The property to be verified is the following (the silent action $i$ is parametrized by the action to which the hide operator was applied):

for all paths, $[i(sendinfo) \rightarrow \text{true}\ U_{\leq 10}\ i(recack)]\ U_{<\infty}\ false$

The labelling starts from the subformulas $i(sendinfo)$, $i(recack)$,true and false. It then proceeds to the modal formulas $U_{\leq 10}$. The formula $\phi$ is not verified for the labelled transition system of the stop and wait protocol since the process *sending* may execute the internal action several times and the *recack* may appear after 10 time units.

## 5 Path Enumeration for Time Interval Lotos

We assume that we have a Time Interval Lotos specification where a time interval is associated with each action. An alternative approach to the verification of timing properties is to derive a graph that captures all the allowed sequences of actions as well as certain constraints on the time interval in which the allowed actions are to take place. This approach enumerates the possible paths composed of timed actions. Hence we can determine from the path enumeration graph if the system contains livelocks or deadlocks. This approach is similar to the one proposed in [Bert 91].

The timed graph described in Section 2 shows paths which are not possible due to the timing constraints associated with the transitions. For example, in the timed graph of the stop-and-wait protocol shown in figure 3, the paths where action $i(s)$ is followed by action $i(m)$ are not valid since the timing conditions associated with those actions cannot be satisfied. On the opposite, the paths shown in the path enumeration graph are all possible.

We will present informally how the path enumeration graph can be constructed. This graph is composed of nodes and transitions. Each node corresponds to a state augmented with time constraints $< B, T >$ where $T$ is the system of inequalities showing the time constraints of the offers enabled. The time constraints $T$ are the following (we will explain below why the constraints have this specific form):

$tmin_{ai} \leq t_{ai} \leq tmax_{ai}$ for all interaction offer $ai$ enabled
$t_{ai} - t_{bj} \leq d_{ai,bj}$ for all $ai$ and $bj$, $a \neq b$

Each transition is labelled with an action and a time interval: the time interval indicates the range of possible time when the action may occur.

It is assumed that the times are relative to the moment at which a previous local action has been executed. When first enabled, the times of the offers must satisfy the following inequalities: $imin_{ai} \leq t_{ai} \leq imax_{ai}$ where $imin_{ai}$ and $imax_{ai}$ correspond to the upper and lower bounds of the interval associated with offer $ai$ and $t_{ai}$ is the time of occurrence of $a$.

Let us consider a transition corresponding to the execution of action $f$. When $f$ is executed at time $t_f$, the inequalities of the remaining interaction offers enabled must be updated to eliminate the variable $t_f$ from the system of inequalities. The modified system of inequalities is associated to the next state. Those inequalities become: $tmin'_{ai} \leq t'_{ai} \leq tmax'_{ai}$ for all $a \neq f$ where the new bounds for the action $a$ are $tmin'_{ai}$ and $tmax'_{ai}$ and where $t_{ai} = t_f + t'_{ai}$.

When the variable $t_f$ is eliminated from the system T, this may introduce a relationship between two other variables $t_{ai}$ and $t_{bj}$ that remain enabled. This relationship can be expressed by the following constraint: $t'_{ai} - t'_{bj} \leq d'_{ai,bj}$ where $d'_{ai,bj}$ is the maximal difference between the two variables $t'_{ai}$ and $t'_{bj}$. This additionnal constraint may narrow the possible occurrence times of future actions, resulting in a time interval for future actions with tigher bounds.

In the initial state or for any new offer $a$ enabled, we can take the following default values for $tmin_{ai}, tmax_{ai}, d_{ai,bj}$:
$tmin_{ai} = imin_{ai}$ and $tmax_{ai} = imax_{ai}$ for all interaction offer $ai$
$d_{ai,bj} = tmax_{bj} - tmin_{ai}$ for all pair of offers $ai$, $bj$ with $a \neq b$.

The path enumeration graph is derived from a Time Interval Lotos behavior expression. Given a state $< B, T >$, we determine all the possible transitions that can be derived. We label the transitions of the graph with one of the possible offer $f$ enabled. Action $f$ can occur in the following interval: $[\max_i(tmin_{f_i}), \min_{a,j}(tmax_{aj})]$. The new state reached is $< B', T' >$ where $B'$ is a new behavior expression derived and $T'$ is the new set of constraints having the same form as T.

The new time region $T'$ is computed from $T$ in the following way:

**Step 1.** Eliminate from the system the time variables that corresponds to the offers disabled by the occurrence of action $f$. Each elimination of a variable $t_ej$ transforms the system of inequalities in the following way:
$tmin'_{ai} = max(tmin_{ai}, tmin_{ej} - d_{ej,ai})$
$tmax'_{ai} = min(tmax_{ai}, tmax_{ej} + d_{ai,ej})$
$d'_{ai,bk} = min(d_{ai,bk}, d_{ai,ej} + d_{ej,bk})$

**Step 2.** Express all remaining times $t_{ai}$ of all actions $a$ where $f \neq a$ as the sum of $t_f$ and a new variable $t'_{ai}$, and eliminate from T all old variables, including $t_f$.
$tmin'_{ai} = max(0, -d_{fj,ai}, tmin_{ai} - tmax_{fj})$
$tmax'_{ai} = min(d_{ai,fj}, tmax_{ai} - tmin_{fj})$
$d'_{ai,bj} = min(d_{ai,bj}, tmax_{ai} - tmin_{bj})$

**Step 3.** Add the time interval constraints corresponding to the new actions enabled. The time intervals are the one associated with the interaction offer in the specification.

$tmin'_{ai} = imin_{ai}$ and $tmax'_{ai} = imax_{ai}$ for all new offers $ai$ enabled
$d'_{ai,bj} = imax_{bj} - imin_{ai}$ for all offers $bj$ enabled $(b \neq a)$

Consider the following example. The corresponding graph is shown in figure 3.
$B = B_1 |[a_3, a_4]| B_2 |[a_3, a_4]| B_3$
$B_1 = a_1[1,6]; (a_2[1,6] \,|||\, a_3[2,3] \,|||\, a_4[1,4]); B_1$
$B_2 = a_4[1,4]; B_2$
$B_3 = a_3[2,3]; B_3$

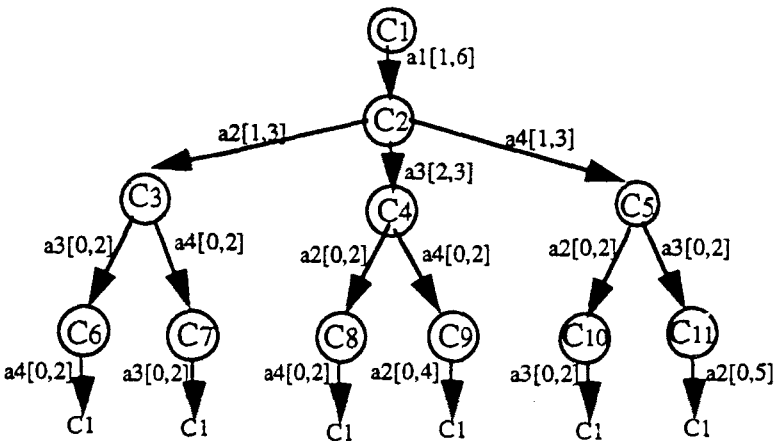


Figure 3. Graph enumarating all possible paths of B

The set of time constraints associated with each state is the following:

| State | T | State | T | State | T |
|---|---|---|---|---|---|
| $C_1$ | $1 \leq t_{a1} \leq 6$ | $C_2$ | $1 \leq t_{a2} \leq 6$<br>$2 \leq t_{a3} \leq 3$<br>$1 \leq t_{a4} \leq 4$ | $C_3$ | $0 \leq t_{a3} \leq 2$<br>$0 \leq t_{a4} \leq 3$<br>$t_{a4} - t_{a3} \leq 2$<br>$t_{a3} - t_{a4} \leq 2$ |
| $C_4$ | $0 \leq t_{a2} \leq 4$<br>$0 \leq t_{a4} \leq 2$<br>$t_{a4} - t_{a2} \leq 3$<br>$t_{a2} - t_{a4} \leq 5$ | $C_5$ | $0 \leq t_{a2} \leq 5$<br>$0 \leq t_{a3} \leq 2$<br>$t_{a2} - t_{a3} \leq 4$<br>$t_{a3} - t_{a2} \leq 2$ | $C_6$ | $0 \leq t_{a4} \leq 2$ |
| $C_7$ | $0 \leq t_{a3} \leq 2$ | $C_8$ | $0 \leq t_{a4} \leq 2$ | $C_9$ | $0 \leq t_{a2} \leq 4$ |
| $C_{10}$ | $0 \leq t_{a3} \leq 2$ | $C_{11}$ | $0 \leq t_{a2} \leq 5$ | | |

In the graph enumerating all paths, each state has only a finite number of successor states, at most one for each action enabled. In deriving the states of this graph, a state $< B', T' >$ is equal to a previously defined state $< B, T >$, if the behavior reached is the same $(B = B')$ and if the time constraints are the

same ($T = T'$). At each new state, if the behavior reached equals a behavior already generated ($B = B'$), the system of inequalities associated with $B'$ is solved and compared with the solution of a system of inequalities associated with $B$. This can be done in polynomial time because the system of inequalities have only at most two variables per inequality [Aspv 79]. When the two systems yield the same solution, the new state is the same as the one already derived. The derivation of the graph stops when processes come into a deadlock state or if the states have all been generated. Hence the same behavior can be associated with two different states if the time regions are different for the two states.

## 6 Summary and Conclusions

Lotos and other formalisms abstract away from time, retaining only the sequencing of events in a system. But for a large class of systems including real-time systems, we need to be able to specify time values and to verify that the system meets certain hard real-time constraints. This paper presents an extension of Lotos with Time Intervals. Its semantics is defined in terms of a labelled transition system augmented with clocks and timing conditions associated with transitions. In the model of Time Interval Lotos presented, time is continuous and not discrete and represented in the form of an interval associated with actions.

Model-checking can be applied to the timed graph obtained from the Time Interval Lotos specification. The formalism we use to express timing assertions about the system is an extension of the branching time logic CTL called TCTL. TCTL is a temporal logic formalism defined by Alur et al. [Alur 90] which handles continuous time. A model-checking approach developed by Alur et al. [Alur 90] is used for verifying that properties expressed in TCTL are verified by the model of Time Interval Lotos.

Another approach to the verification of timing properties is the derivation of all the possible sequences of actions and associated time. The graph generated shows sequences of actions and indicates in what interval the actions can occur. It does not give the evolution for particuliar time values but shows all the possible occurrence times for each action in the path. With this graph, one can determine if the system contains livelock and deadlocks since all the possible execution times are generated.

## 7 References

[Alur 90] Alur, R., Courcoubetis, C. and Dill, D., "Model-checking for real-time systems", Proc. of 5th IEEE Symp. on Logic in Computer Science, June 90.
[Aspv 79] Aspvall, B and Shiloach, "A polynomial time algorithm for solving systems of inequalities with two variables per inequality", in Proc. 20th Annu. Symp. Foundations of Computer Sciences, Oct. 1979, pp.205-217.
[Berg 84] Bergstra, J.A. and Klop, J.W., "Process Algebra for Synchronous Communication", Information and Control, 60 (1-3), 1984.

[Bert 91] Berthomieu, B. and Diaz, M., "Modeling and Verification of Time Dependent Systems Using Time Petri Nets", IEEE Trans. on SE,17,3, March 1991, pp.259-273.
[Bolo 87] Bolognesi, T. and Brinskma, E., "Introduction to the ISO specification language LOTOS", Computer Networks and ISDN Systems,14,1, 1987.
[Bolo 90a] Bolognesi, T., Lucidi, F. and Trigila, S., "From Timed Petri Nets to Timed LOTOS", Proceedings of 10th IFIP WG6.1 PSTV, June 1990.
[Bolo 90b] Bolognesi, T. and Lucidi F., "LOTOS-like process algebras with urgent or timed interactions", FORTE 90, 1990.
[Budk 87] Budkowski, S. and Dembinski, P., "An introduction to Estelle: a specification language for distributed systems", Computer Networks and ISDN Systems,14,1, 1987, pp.3-23.
[Clar 83] Clarke, E., Emerson, E. and Sistla, A., "Automatic verification of finite-state concurrent systems using temporal logic specifications: A practical approach", in Proc.10th ACM Symp. on Principles of Programming Languages, pp.117-126, 1983.
[Ehri 85] Ehrig, H. and Mahr, B., "Fundamentals of Algebraic Specification 1", Springer Verlag, 1985.
[Emer89] Emerson, E.A., Mok, A.K., Sistla, A.P. and Srinivasan, J., "Quantitative temporal reasoning",in Proceedings of workshop on Automatic Verif. Methods for Finite State Systems, June 1989.
[Hans 91] Hansson, H., "Time and Probability in Formal Design of Distributed Systems", Ph.D. Thesis, Uppsala University, September 1991.
[Hoar 85] Hoare, C., "Communicating sequential processes", Prentice Hall, 1985.
[Hulz 90] van Hulzen, W., Tilanus, P., Zuidweg, H., "LOTOS extended with clocks", Proceedings of FORTE'89, Noth-Holland 1990.
[Lewi 90] Lewis, H.R., "A logic of concrete time intervals", 5th IEEE Symnposium on Logic in Computer Science, June 1990.
[Loto 89] ISO/TC97/SC21, "LOTOS: A Formal Description Technique based on the temporal ordering of observational behavior", Tech. Report IS8807, 1989.
[Merl 76] Merlin, P. and Farber, D., "Recoverability of communication protocols-Implication of a theoretical study", IEEE Trans. on Comm.,24,6, Sept.1976.
[Miln 80] Milner, R., "A Calculus of Communicating Systems", LNCS 92, Springer Verlag, 1980, 171p.
[Nico 91] Nicollin, X. and Sifakis, J., "An overview and synthesis on timed process algebras", CAV'91, Aalborg, Denmark, July 1991.
[Quem 89] Quemada, J., Azcorra, A. and Frutos, D., "A Timed Calculus for LOTOS", Proceedings of FORTE'89, Vancouver, June 1989.
[Rico 91] Rico, N., and Bochmann, G.v., "Performance description and analysis for distributed systems using a variant of Lotos", Proceedings 11th PSTV,1991.
[Rico 92] Rico, N., and Bochmann, G.v., "Time Interval Lotos", Technical Report, University of Montreal, 1992.
[SDL 87] CCITT SG XI, Recommendation Z.100 (1987).