Formal Aspects of Computing (1993) 5: 224–252

Formal Aspects of Computing

TIC: A TImed Calculus*

Juan Quemada[1], David de Frutos[2] and Arturo Azcorra[1]

[1]Dpto. Ingeniería Telemática, ETSI Telecomunicación, Universidad Politécnica de Madrid, Madrid, Spain
[2]Dpto. Informática y Automática, Fac. Ciencias Matemáticas, Universidad Complutense, Madrid, Spain

Keywords: Time; Timed algebraic calculus; TIC; Weak bisimulation equivalence

Abstract. TIC is a timed algebraic calculus which combines ideas from asynchronous and synchronous calculi. Time is introduced by assigning explicit time restrictions to the events of an asynchronous calculus. The semantics is defined in an operational way. Interleaving of behaviours is defined in such a way that a proper merge of events in time is achieved. Weak timed bisimulation is also defined. Examples are presented to show the applicability of the calculus to the study of timed behaviours.

1. Introduction

TIC is an algebraic calculus of processes which allows a precise definition of timed systems. This approach assigns a precise timing to the events of an asynchronous calculus, to make it suitable for quantitative time specification. The asynchronous calculus taken as the starting point for TIC is the basic behaviour calculus of LOTOS [ISO88]. Nevertheless, the calculus is general, and the same approach could be used to introduce time in other asynchronous calculi. The present work can be considered as a timed interpretation of the basic calculus of LOTOS, by adding timing restrictions to the events of a specification. The notation and terminology are similar to those used in LOTOS.

Specifications written using an asynchronous calculus (in the sense of [Mil83]) define the relative ordering in time of a given set of events. A time ambiguity exists in such specifications because there are many different real systems which have

* This work was partially supported by CICYT under the TIC program (MEDAS project)
Correspondence and offprint requests to: Juan Quemada, Departmento Ingeniería Telemática, ETSI Telecommunicatión, Universidad Politécnica de Madrid, 28040 Madrid, Spain, email: Quemada@dit.upm.es

the same relative ordering of events, but different timing. There are applications in which the specification of the relative ordering of events gives a sufficiently complete description of the system, but in others, quantitative timing may be an essential part of the behaviour of the system. When considering performance aspects, timing is especially relevant.

In TIC every event must have an explicit time restriction. This may be seen from some points of view as an overspecification. In other approaches, related to the so called *maximum parallelism hypothesis*, some events are supposed to be of negligible duration and to happen as soon as possible. This is a very implementation oriented hypothesis which tries to make good use of the computing resource so that when something is ready to be done and the processor is free, the processor shall not stay idle and will do it. Our approach is specification oriented and tries to be implementation independent, as any FDT should be. Thus, no underlying assumptions are made. To represent events with negligible time separation, a zero time attribute which is a very convenient approximation from the application point of view, can be used.

The effect of timing constraints on interleaved behaviours establishes the major difference with respect to asynchronous calculi, where the merging of events is defined in a way that allows all the possible combinations which maintain the relative ordering of each part. With our definition of interleaving, the events will be merged according to their occurrence in time.

The semantics of TIC is defined operationally by deriving a labelled transition system over which the definition of equivalences is possible. A notion of Weak Timed Bisimulation has been defined and is described in the paper. Other equivalences or relations presented in the literature also seem more or less easy to define, although we have not studied them yet.

The present work is a continuation of [QuF87] and [QAF89] in which different treatments of the passing of time were studied. In the first work, the calculation of the passing of time was solved with a simple mechanism, but a proper merging in time of the events of the interleaved behaviours was not achieved. In the second work this problem was solved by means of a new definition of interleaving, but without achieving all the desired properties. The present approach achieves, by means of the introduction of empty transitions and a new definition of weak bisimulation, all the desired properties similar to those existing in untimed LOTOS. TIC is, in fact, a timed extension of the behaviour calculus of LOTOS. The untimed basic LOTOS calculus is a subcalculus of TIC. The main limitation of TIC is its inability to represent *as soon as possible (asap)* time requirements. There is also a related work [Mig91] which extends [QuF87] with an *asap* timing requirement for internal actions and which has applications in performance analysis.

This work can be situated in the framework of Algebraic Calculi of processes, which started with Milner's CCS [Mil80], CIRCAL [Mil85], CSP[Hoa85], SCCS [Mil83], MEIJE [AuB87], ACP [BeK85] and others followed. Some of these calculi were of asynchronous type, such as CCS, CSP or ACP. Others were of synchronous type, such as SCCS. Some calculi have been compared with SCCS by giving an interpretation in it, but in our case this seems impossible. We found several difficulties when trying to interpret TIC as a subcalculus of SCCS, being one of them the need to distinguish between passing of time and internal actions, because SCCS combines both in a single event. Timed choice and parallelism also present problems.

One important aim of this work has been the definition of a synchronous

calculus where equivalence relations, such as weak bisimulation [Par81] or testing equivalence [dNH84] which have been developed only for asynchronous calculi, can be defined. The same applies to the conformance relation of Brinksma and Scollo [BSS86].

Some of the first works about formal models of time are [K+85] [ReR86] [GeB87]. The second paper [ReR86] (see also [ReR87]) presents a timed model for CSP which includes a delay statement as the basic timing element. The representation of simultaneity in this work has some similarities to our approach. In [Azc89] [NRS90] [HeR90] a different approach based on the separation of time and actions is studied; a special action representing the passing of time is considered, whereas in [OdF90] it is the execution of the actions that takes some positive quantity of time. Different possibilities concerning the instants at which each action can be executed are discussed there. In [BeR85] timing restrictions of tightly coupled automata with time interval constraints are studied in a way which has a slight relation to the present approach. In [Tof88, MoT90] a timed interpretation of CCS is given where an order relation, which expresses the notion of *faster than*, is introduced. Bolognesi's [BLT90, BoL91] presents a timed extension of LOTOS which includes the notion of timers which have an *as soon as possible* timing. Finally, we have had recent notice of a timed extension of ACP presented in [BaB91], which has some points in common with the calculus here presented. The work by Bergstra and Klop focuses on the algebraic characterization of strong bisimulation in ACP but it does not deal with weak bisimulation as we do. These commonalities stress our confidence in the interest of this approach.

Our approach is also related to the way time was introduced in Petri Nets by Merlin and Farber [MeF76], and other related Petri Net models. In addition to using different models, the main difference is that in timed Petri Nets, time constraints are associated with complete transitions, whereas here the time constraints are associated with the individual parts of a specification.

2. The Calculus

2.1. Syntax of the Language

We will work over a class of timed concurrent processes whose observable behaviour is defined in terms of the sequence of observed gates during execution. As in basic-LOTOS and in many other simplified models of concurrency, we will not allow the exchange of values through these gates. Consequently, only the emission of undistinguishable signals through the gates is observable. Gates will be denoted by a name.

We will also have internal actions which are not observable from outside, but whose execution can change the state of the process and therefore its future observable behaviour.

In addition, we can observe the time at which each action occurs, with respect to a discrete scale of time. In fact we will take the set of natural numbers as our domain for time, at both the syntactic and the semantic levels.

Definition 1.

(i) We will consider a universe of gates $\mathscr{G}$ that includes the names of the gates that can appear in any process.

(ii) We have an internal event denoted by i, such that $i \notin \mathscr{G}$.
(iii) The set of events $\mathscr{E}$ is the set $\mathscr{E} = \mathscr{G} \cup \{i\}$. □

Notation: $B, B', B_1, \ldots$ are variables denoting behaviour expressions; $t, t_1, \ldots$ are variables ranging over time, that is to say over N; $T, T_1, \ldots$ range over intervals of time instants $[t^-, t^+]$, where $[t^- \in N, t^+ \in N \cup \{\infty\}]$; $g, g', \ldots$ are variables ranging over $\mathscr{G}$; $a, a', a_1, \ldots$ range over $\mathscr{E}$; and finally $G, G_1, \ldots$ are variables ranging over finite gate name sets.

Definition 2. The algebra of finite basic behaviour expressions is defined by the operators in Table 1. Each behaviour expression B has an associated gate set $L(B)$, also defined in the table. □

Table 1. Syntax of TIC

Name	syntax	Gate set
Timed Deadlock	*stop(t)*	$\emptyset$
Idle	*idle*	$\emptyset$
Action Prefix	$aT;B$	$L(B) \cup \{a \mid a \neq i\}$
Timed Choice	$aT;B$	$L(B) \cup \{a \mid a \neq i\}$
Choice	$B_1 [] B_2$	$L(B_1) \cup L(B_2)$
Parallelism	$B_1 \lvert[G]\rvert B_2$	$L(B_1) \cup L(B_2)$
Hiding	*hide G in B*	$L(B) - G$
Relabelling	$B[g_1/g'_1, \ldots g_n/g'_n]$	$(L(B)\text{-}\{g'_1, \ldots g'_n\}) \cup \{g_1, \ldots g_n\}$

The basic component of the syntax is the pair formed by an event in $\mathscr{E}$ and a natural number which represents an instant of time.

Definition 3. Timed events will be pairs in $\mathscr{E} * N$, represented by the name of the gate followed by its occurrence time. We will denote the set of timed events by $\mathscr{T}\mathscr{E}$. □

For example, $a3$ indicates that event a will happen at instant 3. The time attribute of a time event is relative to the instant at which the previous event occurred and on which the action causally depends.

For instance, in $a1; c2; idle \lvert[\emptyset]\rvert b2; idle$, a will occur at instant 1, b at 2 and c at 3; so c occurs 2 instants after the occurrence of a to which it is causally related.

Time attributes of events are non-negative numbers. A null separation between the occurrence of two events is allowed. This has been considered admissible, not only to allow the parallel execution of several actions at the same time, generated for instance by plain interleaving, but also to capture a negligible separation between events. It is clear that in every case, we must be careful to prevent the possible appearance of unbounded sequences of such events. Without this restriction we would have to admit the execution of an infinite number of actions in finite time, against any physical law. But such sequences would only be introduced by specifications including recursive calls without any positive lapse of time, and this can be prevented at the syntax level by an adequate generalization of the concept of guarded definition.

Although we have not given any semantics yet, the reader will probably guess that starting from *stop(t)* or *idle* as the constants in our signature, the (finite) application of the rest of the operators will generate the set of finite processes of TIC. In order to obtain infinite behaviours, some kind of recursive definition of

Table 2. Syntax of the auxiliary operators

Name	Syntax	Gate set $L(B)$
Time Passing	$Age(t, B)$	$L(B)$
Inaction	$Stop$	$\emptyset$

processes is mandatory. This generalization of behaviour expressions must include variables whose meaning will be defined by means of the least fixed points of the defining equations.

Notation: $PVar$ will denote the set of process variables. $P, P', P_1, \ldots$ will range over $PVar$.

Definition 4. Generalized basic behaviour expressions are defined by adding process variables in $PVar$ as new basic expressions so that metavariables appearing in the definition of finite basic behaviour expressions range over the set of generalized basic behaviour expressions. We denote by $PVar(B)$ the set of process variables appearing in B. Finally $L(B)$ will be undefined for processes including variables. □

In the following definitions Bs will represent generalized basic behaviour expressions.

Definition 5.

(i) A system of recursive definitions of processes (*srdp*) is a system $|\overline{P} : \overline{B}| = |P_i : B_i \,/\, i \in I \,|$ where $\forall i \in I \; PVar(B_i) \subseteq \{P_i \,/\, i \in I\}$.

(ii) If $|\overline{P} : \overline{B}|$ is a *srdp*, we define the set of gates of the processes so defined as the least solution of the system of equations $L(P_i) = L(B_i)$, where $L(B_i)$ is defined as if B_i were a finite basic behaviour expression, by taking as the set of gates of each appearance of each process variable P_j along B_i, just the unknown $L(P_j)$. □

Definition 6.

(i) An infinite basic behaviour expression is a pair $(|\overline{P} : \overline{B}|, B)$ with $\mathrm{PVar}(B) \subseteq \overline{P}$. We denote by $\mathscr{P}$ the class of infinite behaviour expressions. Usually an infinite behaviour expression will be denoted by giving only the generalized behaviour expression B if the corresponding system of equations can be deduced from the context.

(ii) We define the label set of an infinite basic behaviour expression $(|\overline{P} : \overline{B}|, B)$ as the set $L(B)$ calculated by the rules in def. 2, taking as $L(P_i)$ for each P_i appearing in B, the corresponding set of gates defined in def. 5. □

An auxiliary operator *Age*, whose syntax is defined in Table 2, is needed to represent the passing of time over a process that does not evolve in any other way. We will also consider a derived (from the combination of the original ones and the added *Age* operator) operator *stop* that represents any deadlocked behaviour. All the classes of basic behaviour expressions of definitions 5 and 6 can be extended by allowing also the appearance of these new operations in their definitions. The extended classes will be known as the original ones, but suppressing from their names the adjective "basic".

If not otherwise specified, Bs will represent infinite behaviour expressions (*IBEs*) in the rest of the paper.

2.2. Operational Semantics

The operational semantics of the calculus is defined by means of a labelled transition system. Let us first define the empty event which is used to represent the passing of time.

Notation:

(i) We will consider a new event ϵ, called empty event, such that $\epsilon \notin \mathscr{E}$.
(ii) The set of extended actions will be the set $\mathscr{X} = \mathscr{E} \cup \{\epsilon\}$.
(iii) We will denote by $e, e', e_1, \ldots$ an arbitrary element in $\mathscr{X}$.
(iv) The set of extended timed events is the set $\mathscr{X}\mathscr{T}\mathscr{E} = \mathscr{T}\mathscr{E} \cup (\{\epsilon\} * N)$.

Definition 7. We define the operational semantics of an *IBE B*, as the labelled transition system $\langle \mathscr{P}, \mathscr{X}\mathscr{T}\mathscr{E}, TR, B \rangle$ where TR is the set of labelled transitions derived from applying the following rules: □

Timed Deadlock:

$$\frac{}{Stop(t) - \epsilon t' \rightarrow Stop(t - t')} < \quad t' \leq t$$

The timed deadlock –*Stop(t)*– represents a deadlock after time t. It only generates empty transitions until time t has elapsed.
The *stop(t)* behaviour could have been omitted, but it has been included for the purpose of having an explicit representation of deadlock situations.

Idle:

$$\frac{}{Idle - \epsilon t \rightarrow Idle}$$

The *idle* behaviour allows the time to pass, by deriving empty transitions at any moment. It is used for representing the graceful termination of a behaviour.

Action Prefix:

$$\frac{}{at; B - at \rightarrow B}$$

The next rule defines the spontaneous passing of time in action prefix statement. Intuitively, time is allowed to pass (empty transition) whenever it does not cause the behaviour to deadlock.

$$\frac{}{at; B - \epsilon t' \rightarrow a(t - t'); B} < \quad t' \leq t$$

Timed Choice:

$$\frac{}{aT; B - at \rightarrow B} < \quad t \in T$$

Timed choice is an extension of the simple action prefix operator. It constrains an action to occur at any instant of the given set T. It is a combination of action prefix and choice over time from the semantic point of view. If the set of time instants T is finite, timed choice can be defined in terms of action prefix and choice. In fact, action prefix is a particular case of timed choice, when T has only one element; but due to the frequent use of action prefix, it has been considered convenient to maintain this simpler syntactic form.
The spontaneous passing of time in a timed choice statement is again allowed only when the executed empty transition does not cause the behaviour to deadlock.

For $T = [t^-, t^+]$ we have

$$\frac{}{aT; B - \epsilon t \rightarrow aT'; B} < \quad t \leq t^+$$

where $T' = [max(t^- - t, 0), t^+ - t]$.

Choice:

$$\frac{B_1 - at \rightarrow B_1'}{B_1 [] B_2 - at \rightarrow B_1'} \qquad \frac{B_2 - at \rightarrow B_2'}{B_1 [] B_2 - at \rightarrow B_2'}$$

The spontaneous passing of time in a choice statement may age the whole composition if one of the components can be aged. It must be noted that the Aging operator will disallow the transitions whose occurrence time has passed.

$$\frac{B_1 - \epsilon t \rightarrow B_1'}{B_1 [] B_2 - \epsilon t \rightarrow Age(t, B_1 [] B_2)} \qquad \frac{B_2 - \epsilon t \rightarrow B_2'}{B_1 [] B_2 - \epsilon t \rightarrow Age(t, B_1 [] B_2)}$$

Parallel: The definition of the parallel operator includes two conceptually different types of operations: synchronization and interleaving. Synchronization may occur when both counterparts offer the same event at the same instant,

$$\frac{B_1 - et \rightarrow B_1',\ B_2 - et \rightarrow B_2'}{B_1 |[G]| B_2 - et \rightarrow B_1' |[G]| B_2'} < \quad e \in G \cup \{\epsilon\}$$

Remark: Note that the spontaneous passing of time is always synchronized in parallel behaviours.

Example: $(b2; idle|[b]|b3; idle)$ would not synchronize because both b's occur at different instants; whereas $(b2; idle|[b]|b2; idle)$ could synchronize, generating the event $b2$.

$$\frac{B_1 - at \rightarrow B_1', B_2 - e't' \rightarrow B_2', t \leq t'}{B_1 |[G]| B_2 - at \rightarrow B_1' |[G]| Age(t, B_2)} < \quad a \notin G$$

$$\frac{B_2 - at \rightarrow B_2', B_1 - e't' \rightarrow B_1', t \leq t'}{B_1 |[G]| B_2 - at \rightarrow Age(t, B_1) |[G]| B_2'} < \quad a \notin G$$

Interleaving may occur when one process can evolve while the other remains idle. But as time passes everywhere, the passing of time associated with the executed transition must also pass in the idle component. This is the reason for introducing the Aging operator *Age*. As the Aging function disables all the transitions whose time has passed, in the future only the transitions occurring after the executed one (also including zero time separation,) will be allowed.

On the other hand, interleaving must be consistent with timing: if a component must execute some action at some given instant t, it is not possible that the first action executed by the parallel composition will be executed, even by the other component, at some later instant $t' > t$. This is what is expressed by the second premise of the rule. Nevertheless we do not impose maximal parallelism: so far as a component may execute a transition at the instant t, the other one may execute an action at any instant $t' < t$ even if the first component has lost in this way some other possible transitions corresponding to instants $t'' < t'$. Only if all the possible transitions of the first component correspond to instants $t'' < t'$, will it not be possible to execute as first action the one at t'.

Any parallel composition containing a blocked behaviour as a component will also be blocked. This means that the defined semantics is very sensitive to deadlock situations.

Example:
$(b2;idle|[c]|b4;idle)$ would interleave. It has only one possible evolution:

$$(b2;idle|[c]|b4;idle) - b2 \rightarrow (idle|[c]|Age(2,(b4;idle))$$

$$(idle|[c]|Age(2,(b4;idle)) - b2 \rightarrow (Age(2,idle)|[c]|idle)$$

Only non empty transitions are shown. There are additional empty transitions.

Aging:

$$\frac{B - et' \rightarrow B'}{Age(t,B) - e(t'-t) \rightarrow B'} < \quad t \leq t'$$

The passing of time disallows transitions which should have occurred before the adjusted time. Besides, *Age* adjusts the relative time count of the behaviour to which it is applied. Note that the effect of the *Age* operator is *local* to the first actions executed by the aged process.
Example:

$$(Age(3,(a5;idle)) - a2 \rightarrow idle$$

Stop: This behaviour generates no transition at all. Thus, the parallel operator is strict with respect to deadlock.

Hiding:

$$\frac{B - et \rightarrow B'}{hide\ G\ in\ B - et \rightarrow hide\ G\ in\ B'} < \quad e \notin G$$

$$\frac{B - gt \rightarrow B'}{hide\ G\ in\ B - it \rightarrow hide\ G\ in\ B'} < \quad g \in G$$

Example:

$$(hide\ b\ in\ b1;idle) - i1 \rightarrow \ hide\ b\ in\ idle$$

Relabelling:

$$\frac{B - et \rightarrow B'}{B[g_1/g'_1,..,g_n/g'_n] - e[g_1/g'_1,..,g_n/g'_n]t \rightarrow B'[g_1/g'_1,..,g_n/g'_n]}$$

The meaning of the "meta-action" $e[g_1/g'_1,..,g_n/g'_n]t$ is quite obvious: it is $g'_i t$ if $e = g_i$ for some $i \in \{1,..,n\}$ and et otherwise.
Example:

$$(b1;idle)[c/b,b/c] - c1 \rightarrow idle[c/b,b/c]$$

Process:

$$\frac{B_i - et \rightarrow B'_i}{(\,|\,\overline{P} : \overline{B}\,|\,,P_i) - et \rightarrow B'_i} < \quad i \in I$$

This is the only rule in which the system of equations defining the named processes is made explicit. We systematically try to avoid this cumbersome notation, using it only in those cases in which an explicit reference to the subsumed system of equations is needed. On the other hand, the fact that this system of equations is never modified by the application of the rules justifies the avoidance of explicit references.

Remarks:

(i) We have not identified internal and empty events, even if both of them generate no observable actions. The reason can be found in the choice rule in which the execution of some internal event by any of the components would make the choice, whereas this must not be the case with empty events. The passing of time can disallow actions whose occurrence time is in the past but it can never disallow actions whose occurrence time is in the present or future.

(ii) As one can observe, time is dealt with as an extension of the gate names in nearly all cases. The only exception is interleaving in which an aging function is used for preemptive time passing.

(iii) The time values appearing in the labels of the defined labelled transition system are relative counts with respect to the previous action. However, from these local clocks we can infer an implicit global clock in a very natural and simple way by considering the tree of transitions generated by the transition system. We can associate a global time stamp with each node of this tree (representing a state of the process) that is computed by adding all the time attributes of the arcs of the path reaching the node from the root of the tree. An example is seen in Fig. 1, where the value of the global time clock *ck* is shown for each node.

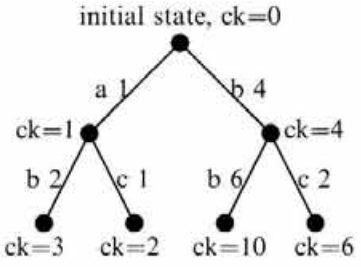

Fig. 1. Global time count.

3. Strong Timed Bisimulation: Some Properties

Strong Timed Bisimulation (STB) is defined by introducing time in the classical definition.

Definition 8.

(i) A relation $STB \in \mathscr{P} \times \mathscr{P}$ is a *Strong Timed Bisimulation* relation iff $\forall \langle B_1, B_2 \rangle \in STB, \forall et \in \mathscr{IT}\mathscr{E}$, the following conditions hold:

- $\forall B_1' \;\; B_1 - et \rightarrow B_1' \implies \exists B_2' \;\; B_2 - et \rightarrow B_2' \;\; \langle B_1', B_2' \rangle \in STB$
- $\forall B_2' \;\; B_2 - et \rightarrow B_2' \implies \exists B_1' \;\; B_1 - et \rightarrow B_1' \;\; \langle B_1', B_2' \rangle \in STB$

(ii) We say that B_1 and B_2 are strongly bisimilar iff there exists some Strong Timed Bisimulation STB such that $\langle B_1, B_2 \rangle \in STB$. □

Some equational properties of the *Age* operator are presented in the next proposition, which will illustrate the definition of Strong Timed Bisimilarity.

Notation: The equality symbol is used to represent Strong Timed Bisimilarity.

Proposition 1. These are some of the properties of Strong Timed Bisimilarity concerning the *Age* operator.

$$Age(t, idle) = idle \qquad Age(t, stop) = stop \tag{1}$$

$$t \leq t' \implies Age(t', stop(t)) = stop(t - t') \tag{2}$$

$$t' < t \implies Age(t', stop(t)) = stop \tag{3}$$

$$t \leq t' \implies Age(t, (at'; B)) = a(t' - t); B \tag{4}$$

$$t' < t \implies Age(t, (at'; B)) = stop \tag{5}$$

$$t \leq t^+ \implies Age(t, (a[t^-, t^+]; B)) = a[max(t^- - t, 0), t^+ - t]; B \tag{6}$$

$$t > t^+ \implies Age(t, (a[t^-, t^+]; B)) = stop \tag{7}$$

$$Age(t, hide\ G\ in\ B) = hide\ G\ in\ Age(t, B) \tag{8}$$

$$Age(t, B_1 [] B_2) = Age(t, B_1)\ []\ Age(t, B_2) \tag{9}$$

$$Age(t, B_1 |[G]| B_2) = Age(t, B_1)\ |[G]|\ Age(t, B_2) \tag{10}$$

$$Age(t, B[g_1/g'_1, .., g_n/g'_n]) = Age(t, B)[g_1/g'_1, .., g_n/g'_n] \tag{11}$$

$$Age(t, ((\overline{P} : \overline{B}), P_i) = Age(t, B_i) \tag{12}$$

$$Age(t_1, Age(t_2, B)) = Age(t_2 + t_1, B) \tag{13}$$

Proof. Let us first relate the transitions generated by *Age*(t,B) with those generated by B. We have that

- If $B - at' \rightarrow B'$ with $t' \geq t$ then $Age(t, B) - a(t' - t) \rightarrow B'$.
- Age(t,B) has no other transitions. In particular, if $B - at' \rightarrow B'$ with $t' < t$ then Age(t,B) has no corresponding transition.

The proof of the equational properties based on this characterization is an easy but cumbersome exercise. Let us present as examples the proofs of 10 and 13.

Equation 10: The right hand side can derive a transition only by applying either the synchronization, or any of the interleaving rules.

- **Synchronization:** We would have $Age(t, B_1) - at' \rightarrow B'_1$ and $Age(t, B_2) - at' \rightarrow B'_2$. Then $B_1 - a(t+t') \rightarrow B'_1$ and $B_2 - a(t+t') \rightarrow B'_2$, so that $(B_1 |[G]| B_2) - a(t+t') \rightarrow (B'_1 |[G]| B'_2)$, and finally $Age(t, B_1 |[G]| B_2) - at' \rightarrow (B'_1 |[G]| B'_2)$.
- **Interleaving:** If $Age(t, B_1) - at' \rightarrow B'_1$ and $Age(t, B_2) - et'' \rightarrow B'_2$ for some $t'' \geq t'$, then $B_1 - a(t+t') \rightarrow B'_1$ and $B_2 - e(t+t'') \rightarrow B'_2$, and as $t+t'' \geq t+t'$ we can apply the interleaving rule to obtain $(B_1 |[G]| B_2) - a(t + t') \rightarrow B'_1 |[G]| Age(t + t', B_2)$, and then $Age(t, B_1 |[G]| B_2) - at'' \rightarrow B'_1 |[G]| Age(t + t', B_2)$, and applying (11) we conclude taking $B' \equiv B'_1 |[G]| Age(t + t', B_2)$, that $Age(t, B_1 |[G]| B_2) - at'' \rightarrow B'$ for some $B' = B'_1 |[G]| Age(t', Age(t, B_2))$.
  The case corresponding to the application of the other interleaving rule is symmetrical.

On the other hand, using the characterization given at the beginning of this proof, we have that each transition of the left hand side of (8) corresponds to a

transition of $B_1|[G]|B_2$, and the rest of the proof is just the converse of the one just developed.

Equation 13: $Age(t_1, Age(t_2, B)) - at \rightarrow B' \iff Age(t_2, B)) - a(t + t_1) \rightarrow B' \iff B - a(t + t_1 + t_2) \rightarrow B' \iff Age(t_1 + t_2, B)) - at \rightarrow B'$. □

Remarks: Equations (1)–(11) give us a denotational definition of the Aging operator, equivalent to our former operational definition. This means that *Age* is indeed a derived operator that does not generate any new semantic process –except for stop– when applied to behaviour expressions (that by definition do not include the Aging operator). This is true even for infinite processes: if, for instance, we have $(\overline{P} : \overline{B})$ and we consider the process $Age(t, P)$, we can take the equation $Age(t, P) = Age(t, B)$, and if we unfold its right hand side we obtain an expression B' in which the only appearances of the Aging operator would have the form $Age(t', P)$ with $t' \leq t$. Then we can consider the corresponding equations $Age(t', P) = Age(t', B)$, obtaining a system whose unknowns are the processes $Age(t', P)$ with $t' \leq t$. But as this system is an ordinary one, once we forget about the names of the unknowns, its solution is just a collection of ordinary infinite behaviours.

However we can obtain new semantic processes if we allow the appearance of the Aging operator on the right hand side of the system defining an infinite behaviour. This will not happen in practice, since *Age* is an auxiliary operator. Therefore we can omit the consideration of the Aging operator when developing proofs by structural induction.

Let us give more equational properties of the basic operators.

Proposition 2. The following pairs of *IBEs* are Strongly Timed Bisimilar.

$$B_1 [] B_2 = B_2 [] B_1, \quad (B_1 [] B_2) [] B_3 = B_1 [] (B_2 [] B_3) \tag{14}$$

$$B [] B = B, \quad B [] stop(0) = B, \quad B [] stop = B, \quad idle [] stop(t) = idle \tag{15}$$

$$hide\ G\ in\ stop(t) = stop(t) \tag{16}$$

$$\begin{aligned} a \in G \ &=> \ hide\ G\ in\ at; B = it; hide\ G\ in\ B \\ a \notin G \ &=> \ hide\ G\ in\ at; B = at; hide\ G\ in\ B \end{aligned} \tag{17}$$

$$hide\ G\ in\ B_1 [] B_2 = (hide\ G\ in\ B_1) [] (hide\ G\ in\ B_2) \tag{18}$$

$$stop(t)[g_1/g_1', .., g_n/g_n'] = stop(t) \tag{19}$$

$$idle[g_1/g_1', .., g_n/g_n'] = idle \tag{20}$$

$$(at; B)[g_1/g_1', .., g_n/g_n'] = a[g_1/g_1', .., g_n/g_n']t; (B[g_1/g_1', .., g_n/g_n']) \tag{21}$$

$$(B_1 [] B_2)[g_1/g_1', .., g_n/g_n'] = B_1[g_1/g_1', .., g_n/g_n'] \ [] \ B_2[g_1/g_1', .., g_n/g_n'] \tag{22}$$

$$B_1|[G]|B_2 = B_2|[G]|B_1 \tag{23}$$

$$(B_1|[G]|B_2) \ |[G]| \ B_3 = B_1 \ |[G]| \ (B_2|[G]|B_3) \tag{24}$$

Proof. Except for the last one, all are more or less immediate checks. The difficulty of the proof of associativity of the parallel operator is due to the application of the interleaving rule. Let us suppose, for instance, that $(B_1|[G]|B_2) \ |[G]| \ B_3$ executes a transition $-at \rightarrow$ corresponding to B_1. Then we have $B_2 - e't' \rightarrow$

with $t \leq t'$ and $B_3 - e''t'' \rightarrow$ with $t \leq t''$. But if $e' \in G$ or $e'' \in G$ it is not trivial that we have some $e'''t'''$ with $B_2|[G]|B_3 - e'''t''' \rightarrow$ and $t \leq t'''$. In order to prove it, we need first to check, by structural induction, that whenever we have $B - at \rightarrow B'$ we also have for any $t' \leq t$ $B - \epsilon t' \rightarrow$. Then we can take $e' = e'' = \epsilon$ and $t' = t''$, obtaining $B_2|[G]|B_3 - \epsilon' t' \rightarrow$. □

Finally we present without full proof, the fact that strong timed bisimulation is a congruency with respect to all the operators of our algebra.

Theorem 1. Strong Timed Bisimulation is preserved by all the operators of our algebra, including the auxiliary ones.

Proof. We do not include the full proof for strong equivalence because we have already proved several rules in detail and we include also the proof of the congruency of all the operators for weak bisimulation, except for choice.

Nevertheless, we will present the proof of congruency of the choice operator, for we will see that $\{(P[]R, Q[]R) | P, Q, R \in \mathscr{P}, \mathscr{P} = \mathscr{Q}\}$ is a strong bisimulation. Let $P[]R - et \rightarrow S$ be a transition from $P[]R$. We distinguish two cases:

- $e = a \in \mathscr{E}$. Then we have two possibilities:
  - $P - at \rightarrow P'$ and then $S = P'$. In such a case we have $Q - at \rightarrow Q'$ for some $P' = Q'$, since $P = Q$, and thus $Q[]R - at \rightarrow Q'$.
  - $R - at \rightarrow R'$ and the $S = R'$. This is an immediate case.
- $e = \epsilon$. There are again two possibilities:
  - $P - \epsilon t \rightarrow P'$ and then $S = Age(t, P[]R)$. But as $P = Q$ we have $Q - \epsilon t \rightarrow Q'$ for some $P' = Q'$, and thus $Q[]R - \epsilon t \rightarrow Age(t, Q[]R) = Age(t, P[]R)$, since Age operator also preserves strong equivalence.
  - $R - \epsilon t \rightarrow R'$ is similar to the previous case. □

4. Expansion Theorems

Two expansion theorems are presented. The first one considers only processes with finite choices of timed actions, whereas the second one is the generalization in which the Timed Choice operator can also appear, and therefore includes infinite choices.

Theorem 2. (Expansion theorem without timed choice)
Let $B_1 = \sum_{i \in I} a_i t_i; B_i'$, $B_2 = \sum_{j \in J} a_j t_j; B_j'$ two behaviours, where I, J are two finite disjoint sets, and $\sum$ represents the generalization of choice as a multi-ary operator (remember that choice is a commutative and associative operator). Let $End_1 = Max(\{t_i \mid i \in I\})$ and $End_2 = Max(\{t_j \mid j \in J\})$. Then, we have

$$
\begin{aligned}
B_1|[G]|B_2 \quad = \quad & \sum_{i \in I'} a_i t_i; (B_i'|[G]|Age(t_i, B_2)) \;\; [] \\
& \sum_{j \in J'} a_j t_j; (Age(t_j, B_1)|[G]|B_j') \;\; [] \\
& \sum_{a_i = a_j t_i = t_j} a_i t_i; (B_i'|[G]|B_j') \;\; [] \;\; stop(m)
\end{aligned}
$$

where $I' = \{i \in I \mid a_i \notin G,\ t_i \leq End_2\}$, $J' = \{j \in J \mid a_j \notin G,\ t_j \leq End_1\}$ and $m = min\{End_1, End_2\}$.

Proof. It is again a routine check. We will merely explain the role of the stop summand in the *rhs*. It covers the case in which the *lhs* cannot generate any nonempty transition; in that case we have a deadlocked behaviour, but this is detected only after time m has elapsed, and one of the components becomes deadlocked itself. If the *lhs* is not deadlocked, then it can also execute an ϵt transition for any $t \leq m$, and thus the *stop* summand is also correct. □

Remark: The former theorem can be generalized to cover the case in which idle behaviours appear as summands of B_1 or B_2 by considering the following three cases.

- If $B_1' = B_1 [] idle$ the expansion of $B_1'|[G]|B_2$ is defined exactly as that of $B_1|[G]|B_2$, but taking $End_1 = \infty$.
- If $B_2' = B_2 [] idle$ the expansion of $B_1|[G]|B_2'$ is defined exactly as that of $B_1|[G]|B_2$, but taking $End_2 = \infty$.
- If $B_1' = B_1 [] idle$ and $B_2' = B_2 [] idle$ the expansion of $B_1'|[G]|B_2'$ is defined exactly as that of $B_1|[G]|B_2$, but taking $End_1 = \infty$, $End_2 = \infty$; then the $stop(m)$ summand would be substituted by idle.

A similar extension can be developed to cover the case in which we have timed deadlocked behaviours as summands of B_1 or B_2.

Theorem 3. (Expansion theorem including timed choice)
Let $B_1 = \sum_{i \in I} a_i T_i; B_i'$, $B_2 = \sum_{j \in J} a_j T_j; B_j'$ two behaviours, where I, J are two finite disjoint sets, $T_i = [t_i^-, t_i^+]$, $T_j = [t_j^-, t_j^+]$, and, as in the previous theorem, $\sum$ represents multi-ary choice. Let us define

$$
\begin{aligned}
Begin_1 &= Max(\{t_i^- \mid i \in I\} \cup \{t_i^+ + 1 \mid i \in I,\ t_i^+ \neq \infty\}) \\
Begin_2 &= Max(\{t_j^- \mid j \in J\} \cup \{t_j^+ + 1 \mid j \in J,\ t_j^+ \neq \infty\}) \\
End_1 &= Max(\{t_i^+ \mid i \in I\}) \\
End_2 &= Max(\{t_j^+ \mid j \in J\})
\end{aligned}
$$

Then we have

$$
\begin{aligned}
B_1|[G]|B_2 = \\
&\sum_{i \in I'} ((\sum_{t \in T_i \cap [t_i^-, Begin_2]} a_i t; (B_i'|[G]|Age(t, B_2))) \\
&\quad [] a_i T_i \cap [Begin_2, min\{t_i^+, End_2\}]; (B_i'|[G]|Age(Begin_2, B_2))) \ [] \\
&\sum_{j \in J'} ((\sum_{t \in T_j \cap [t_j^-, Begin_1]} a_j t; (Age(t, B_1)|[G]|B_j')) \\
&\quad [] a_j T_j \cap [Begin_1, min\{t_j^+, End_1\}]; (Age(Begin_1, B_1)|[G]|B_j')) \ [] \\
&\sum_{a_i = a_j} a_i [max\{t_i^-, t_j^-\}, min\{t_i^+, t_j^+\}]; (B_i'|[G]|B_j')) \ [] \ stop(m)
\end{aligned}
$$

where again $I' = \{i \in I \mid a_i \notin G,\ t_i \leq End_2\}$, $J' = \{j \in J \mid a_j \notin G,\ t_j \leq End_1\}$ and $m = min\{End_1, End_2\}$.

Remarks:

(i) By convention, any summand in the expression above, including an empty interval of time t^-, t^+ with $t^- > t^+$, can be removed from the expression.

(ii) The choice of $Begin_k$ is motivated by the need to find a finite bound so that, for instance, whenever we would execute any action from B_1 at any instant later than $Begin_2$ we could put as a continuation of the second component a uniform behaviour $Age(Begin_2, B_2)$, whatever the instant might be. This is because if $K = \{j \in J \mid t_j^+ = \infty\}$, for all $t \geq Begin_2$ we have $Age(t, B_2) = \sum_{k \in K} a_k[0, \infty]\,; B'_k = Age(Begin_2, B_2)$.

(iii) The meaning of the remaining parameters is the same as in the case not including timed choice.

(iv) The theorem can also be generalized to cover idle and timed deadlock summands exactly as in the case not including timed choice.

Proof. The key idea underlying the proof is included in the previous remarks: when considering interleaving, we have to distinguish the different continuations of the idle component, which initially depend on the exact instant at which the corresponding action is executed. Fortunately this is true only until an instant is reached, after which the continuation remains the same forever. Thanks to this fact we avoid infinite sums that of course would not be admissible at the practical level at which we desire to apply the expansion. □

Remark: The use of time intervals to define the set of options of a Timed Choice allows the existence of a generalized expansion theorem. A more general language allowing more or less arbitrary sets instead of these intervals would have led to an infinite number of different continuations after the execution of a first action, depending on the instant at which it is executed. This would lead us to an inadmissible infinite expression. However, it is true that with some additional work it would be possible to generalize somewhat the class of intervals, without loosing the expansion theorem. For instance we could accept finite unions of intervals, sets such as $\{t \in N \mid t \equiv k \bmod n\}$, or finite intersections of both classes of sets. And probably some other *regular sets* would be also admissible.

We think that these expansion theorems together with the previous rules constitute a reasonable set of rules for strong equivalence. Nevertheless we have not included any completeness result for two main reasons:

First, we are mainly interested in weak equivalence and in the properties of strong and weak equivalence which ensure the practical applicability of the calculus, such as the congruency of the parallel operator. Secondly, our experience shows that completeness results for timed calculi are in general much more complicated than the corresponding results for untimed cases and would justify a complete work by itself. The only detailed proof of completeness of an axiomatization of an equivalence between timed behaviours known to the authors is [OdF91], in which a timed version of CSP is studied and in which the normal forms are indeed much more complicated than those for ordinary CSP.

5. Weak Timed Bisimulation

As in the non timed case (see Milner [Mil80]), strong bisimulation is too strong because it is based on the observation of internal and visible actions. Internal

actions should not be observable. In order to formalize this idea, we have to derive an operational semantics showing only the execution of external actions. There is an important difference between the timed and the untimed case: in the untimed case internal actions disappear (we should say *nearly disappear* if we want to be strict), while now the time dedicated to the execution of internal actions remains as a track of the execution.

The new operational semantics will show only the execution of external actions, the time between them, and the time elapsed since the execution of the last one. The time associated with the internal actions is accumulated in the next observable action, or constitutes a part of the time elapsed since the execution of the last of them. All this is formalized by the new transition system described below.

Definition 9.

(i) $OT = \{gt \mid g \in G, t \in N\}$ is the set of observable timed events.

(ii) $IT = \{it \mid t \in N\} \cup \{\epsilon t \mid t \in N\}$ is the set of internal timed events. We will represent by xt the elements of IT.

(iii) Let B, B' infinite behaviour expressions, $a_1t_1, \ldots, a_nt_n \in \mathscr{XTE}$ with $n > 0$; we say that $B - a_1t_1, \ldots, a_nt_n \rightarrow B'$ is an *extended transition* iff $\exists B_0, \ldots, B_n$ infinite behaviour expressions with $B = B_0$ and $B' = B_n$, such that $\forall i \in \{0, \ldots, n-1\}$ $B_i - a_{i+1}t_{i+1} \rightarrow B_{i+1}$.

(iv) Let $B - I_0, g_0t_0, I_1, g_1t_1, \ldots, I_{n-1}, g_{n-1}t_{n-1}, I_n \rightarrow B'$ be an extended transition, where for each $i \in \{0, \ldots, n-1\}$ $g_it_i \in OT$, and each I_i represents a possibly empty sequence $xt_{i,1}, \ldots, xt_{i,n_i}$. Then, for each $j \in \{0, \ldots, n\}$ we define t'_j by

$$t'_j = t_j + \sum_{k=1}^{n_j} t_{j,k} \quad \forall j \in \{0, \ldots, n-1\}$$

$$t'_n = \sum_{k=1}^{n_n} t_{n,k}$$

Then, we say that $B = g_0t'_0, \ldots, g_{n-1}t'_{n-1}, t'_n \Rightarrow B$ is the *observable timed transition* associated with the given extended transition. □

Remark: Only external actions can appear as labels of observable timed actions. When a transition does not contain external actions we have an *empty observable timed action* which corresponds to the case where $n = 0$. The time consumption of such a transition is determined by the time label I_0. Nevertheless, contrary to what happens in the Untimed Calculus, extended transitions must contain at least one transition. This is always possible because any derivative B' of any behaviour (in which only the basic operators can appear) can generate the empty transition $\epsilon 0$, becoming $Age(0, B')$ which is strongly timed bisimilar to B'. Thus we have $B' = \langle\rangle, 0 \Rightarrow Age(0, B')$ which corresponds to the idle transition $B' \Rightarrow B'$ of the Untimed Calculus. The following lemma formalizes our previous assertion.

Lemma 1. For any infinite basic behaviour expression B, and for any extended transition $B - a_1t_1, \ldots, a_nt_n \rightarrow B'$ we have $B' - \epsilon 0 \rightarrow Age(0, B')$.

Proof. It is easy to check that only the appearance of the auxiliary operators *stop* and *Age* can make the transition $\epsilon 0$ non executable. But stop never appears in a derivative of a process that does not contain it already. A behaviour B

cannot contain it, since *stop* is an auxiliary operator. On the other hand, the *Age* operator only appears when the interleaving and some of the choice rules are applied. But the executability of $\epsilon 0$ is always maintained. Let us take for instance an application of the first interleaving rule:

$$\frac{B_1 - at \rightarrow B_1', B_2 - e't' \rightarrow B_2', t \leq t'}{B_1 | [G] | B_2 - at \rightarrow B_1' | [G] | Age(t, B_2)} < \quad a \notin G$$

Then, as $t \leq t'$ we have $Age(t, B_2) - e'(t - t') \rightarrow B_2'$ and by applying the result given on the proof of prop. 2, we have that $Age(t, B_2) - \epsilon 0 \rightarrow Age(0, Age(t, B_2))$. □

Definition 10.

(i) A relation $WTB \subseteq \mathscr{P} \times \mathscr{P}$ is a *Weak Timed Bisimulation* iff $\forall \langle B_1, B_2 \rangle \in WTB$, $\forall q \in OT^*$ we have

- $B_1 = q, t_n \Rightarrow B_1' \implies \exists B_2' \ B_2 = q, t_n \Rightarrow B_2' \ \wedge \langle B_1', B_2' \rangle \in WTB$
- $B_2 = q, t_n \Rightarrow B_2' \implies \exists B_1' \ B_1 = q, t_n \Rightarrow B_1' \ \wedge \langle B_1', B_2' \rangle \in WTB$

(ii) We say that B_1 and B_2 are weakly timed bisimilar, and we will write $B_1 \sim B_2$, iff there exists some weak timed bisimulation WTB with $\langle B_1, B_2 \rangle \in WTB$. □

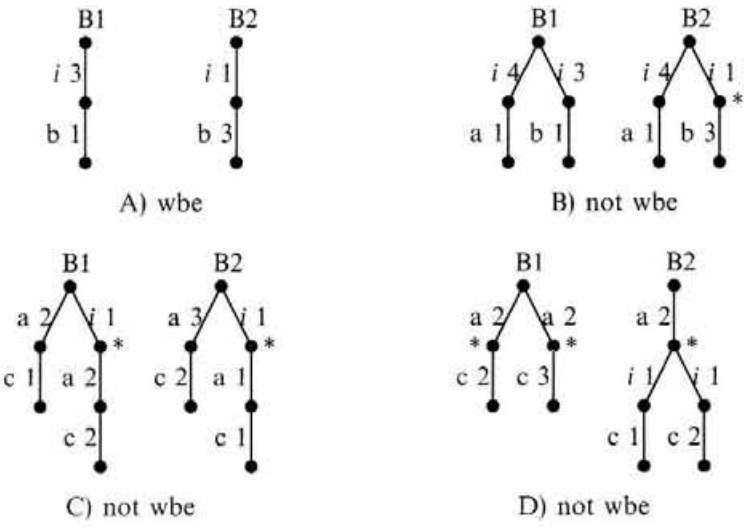

Fig. 2. Examples of weak timed bisimulation.

Some examples illustrating this definition are presented in Fig. 2, where *wbe* stands for weak timed bisimilar, empty transitions are omitted and the states which cannot be bisimulated by the peer behaviour, if any, are indicated by an asterisk. From cases B) and D), you can see that our definition of equivalence is perhaps too strong, as possibly you would like to identify the pairs of processes involved. It is in fact probable that a reasonable definition of testing equivalence would declare them to be equivalent. But in any case, it is well known that already in the ordinary untimed case, bisimulation is stronger than testing equivalences.

Let us see in detail why the pairs of processes compared in B), C) and D), are non-equivalent. In B) if the second process decides to execute its internal action corresponding to the instant 1, it forces itself merely to execute *b* in the future; but the first process cannot impose such an obligation on time 1 because it cannot make a choice until the instant 3. In C), it is reasonable to declare the processes non-equivalent because in fact they are not even failure-equivalent: *a3 c2* is always accepted by the first process, while the second one accepts it only sometimes. Of course this is due to their respective timing because in the non timed case they would be bisimilar. Finally D) is the timed version of one of the classical CCS examples showing a pair of processes that are testing equivalent but are not weak bisimilar. The two different actions *c* and *d* of the untimed case, have been substituted by the timed actions *c2* and *c3*.

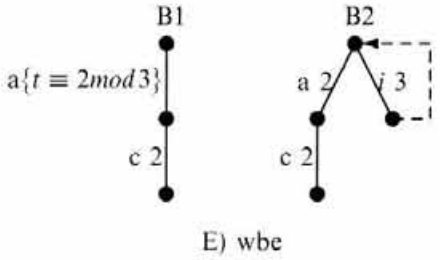

Fig. 3. Example of i-loop removal.

Another interesting example is shown in Fig. 3. The example illustrates an internal action loop elimination. This is also possible in the untimed case, but now we have to keep the time consumption due to the executed internal actions. We introduce a time choice to represent this time consumption, where the set of time instants at which the action *a* can be executed is the set $\{t \mid t \equiv 2mod3\}$. This is a case in which the use of one of the already proposed extensions of our original definition of time choice is necessary. Thus, the effect of i-loops does not vanish as in an untimed calculus, but is kept as a potentially unbounded time delay.

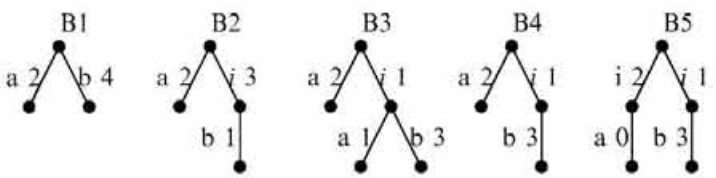

Fig. 4. Choice, internal actions and time.

Finally, Fig. 4 depicts a set of examples selected to show the interrelation between choice, internal actions and time. In it *B1* and *B2* are *wbe*, and *B4* and *B5* are also equivalent, but there are no more equivalences between the shown processes. Equivalence of *B1* and *B2* indicates that the time at which the internal decisions occur is relevant. The internal event, *i3*, in *B2* can occur only after the visible event *a2*, offered in choice with it. Therefore, it cannot influence the occurrence of *a2*. In *B3* the internal event, *i1*, can occur before *a2*, which thus

can be rejected. In *B4* the internal event, *i1*, may not occur at instant 1; in such a case only event *a* would be offered at instant 2. Thus, we have an internal choice between *a2* and *b4*, which is made explicit in *B5*.

It is interesting to note that the classic notion of weak bisimulation, applied to the untimed versions of the depicted processes, would relate the four behaviours in a completely different way: we would have $B1 \sim B3$, and $B2 \sim B4$, with *B5* not equivalent to any one of them. An untimed calculus, like basic LOTOS, can be obtained in TIC by having actions with no time restrictions as shown in definition 11, but not just by removing the timing.

We have not yet gone deeply enough into the subject of an equational characterization of weak timed bisimulation. Nevertheless, we have already studied the basic axioms from the classical theory covering internal actions, finding that it is possible to extend them to the timed case, although to do it, we need a trivial generalization of the *Aging* operator to cover also negative periods of time. Using it we have

Proposition 3. The following pairs of processes are weakly timed bisimilar.

$$at; it'; B \sim at; Age(-t', B)$$

$$Age(-t, B)[]it; B \sim it; B$$

$$at; (B_1[]it'; B_2)[]at; Age(-t', B_2) \sim at; (B_1[]it'; B_2)$$

Proof. Again they are easy but tedious checks. □

Of course, all the axioms and rules presented in the previous sections for strong equivalence remain valid for weak equivalence.

Concerning the congruence properties of the equivalence, the situation is very similar to that in LOTOS or in CCS: between our basic operators, only the choice operator does not preserve the equivalence. Nevertheless, it is also true that our *Aging* operator does not preserve it. Take for instance $B_1 = i1; a2; idle \sim B_2 = a3; idle$. Aging them two instants, we obtain $Age(2, B_1) \sim stop \not\sim a1; idle \sim Age(2, B_2)$. But as we explained before, this kind of negative result concerning an auxiliary operator does not matter, since it cannot appear in user's specifications.

Theorem 4. Weak Timed Bisimulation is preserved by Action Prefix, Time Choice, Parallel Composition, Hiding and Relabelling. □

To develop the proof we first present a characterization of Weak Timed Bisimulations.

Proposition 4. *WTB* is a Weak Timed Bisimulation iff $\forall \langle B_1, B_2 \rangle \in WTB, \forall gt \in OT, \forall xt \in IT$ we have

- $B_1 - gt \rightarrow B_1' \implies \exists B_2'\ B_2 = \langle gt \rangle, 0 \Rightarrow B_2'\ \langle B_1', B_2' \rangle \in WTB$
- $B_2 - gt \rightarrow B_2' \implies \exists\ B_1'\ B_1 = \langle gt \rangle, 0 \Rightarrow B_1'\ \langle B_1', B_2' \rangle \in WTB$
- $B_1 - xt \rightarrow B_1' \implies \exists\ B_2'\ B_2 = \langle \rangle, t \Rightarrow B_2'\ \langle B_1', B_2' \rangle \in WTB$
- $B_2 - xt \rightarrow B_2' \implies \exists\ B_1'\ B_1 = \langle \rangle, t \Rightarrow B_1'\ \langle B_1', B_2' \rangle \in WTB$

Proof. $\implies$ In the first case we have $B_1 = \langle gt \rangle, 0 \Rightarrow B_1'$ and then we just apply the definition of *wtb*. The other case is similar: we would have $B_1 = \langle \rangle, t \Rightarrow B_1'$, and again we would apply the definition.

$\impliedby$ Let $B_1 = q, t \Rightarrow B_1'$. This observable timed transition will be the one

associated with some extended transition $B_1 - a_1t_1,\ldots,a_kt_k \rightarrow B_1'$. Then we distinguish two cases:

- $a_1 \in G$. Then we have $B_1 - a_1t_1 \rightarrow B_1'' - a_2t_2,\ldots,a_kt_k \rightarrow B_1'$, so that we have some B_2'' verifying $B_2 = \langle a_1t_1\rangle,0 \Rightarrow B_2''$ and $\langle B_1'', B_2''\rangle \in WTB$.
- $a_1 = x \notin G$. In this case $B_1 - xt_1 \rightarrow B_1'' - a_2t_2,\ldots,a_kt_k \rightarrow B_1'$, and we would have some B_2'' verifying $B_2 = \langle\rangle,t_1 \Rightarrow B_2''$ with $\langle B_1'', B_2''\rangle \in WTB$.

And iterating we would obtain an extended transition $B_2 - a_1't_1',\ldots,a_l't_l' \rightarrow B_2'$ whose associated observable timed transition is $B_2 = q,t \Rightarrow B_2'$, and such that $\langle B_1', B_2'\rangle \in WTB$. □

Let us go now to the proof of the congruency theorem.

Proof. The only difficult cases correspond to Parallel Composition and Hiding.

- **Parallel Composition:** For we will prove that

$$WTB = \{(P|[G]|R,\ Q|[G]|R) \mid P,Q,R \in \mathscr{P},\ P \sim Q,\ G \subseteq \mathscr{G}\}$$

is a weak timed bisimulation. We will use the former characterization of weak bisimulation. Let $P|[G]|R - gt \rightarrow P'|[G]|R'$. We have three cases:

(i) $g \in G$. Then we have $P - gt \rightarrow P'$ and $R - gt \rightarrow R'$, and then as $Q \sim P, Q = \langle gt\rangle,0 \Rightarrow Q'$ with $Q' \sim P'$, and it is easy to check that then we have $Q|[G]|R = \langle gt\rangle,0 \Rightarrow Q'|[G]|R'$, since applying the interleaving rule $Q|[G]|R$ can execute the internal actions of Q before its execution of g and then synchronize to execute this g.

Remark: One could think that a separate consideration of the case must be made where the extended transition leading from Q to Q' contains empty events, since these are not considered by our interleaving rule. But this is not necessary because observing how these empty transitions are generated, it can be concluded that whenever they appear in an extended transition followed by an observable or internal action, they can be removed to get a new extended transition which generates the same observable transition. This means that for any observable transition $B = q,0 \Rightarrow B'$ we can always find an equivalent extended transition free of empty events.

(ii) $g \notin G$, $P - gt \rightarrow P'$, $R' = Age(t,R)$, and $R - et' \rightarrow R''$ with $t \leq t'$. Then, as in the previous case, we have some $Q = \langle gt\rangle,0 \Rightarrow Q'$ with $Q' \sim P'$, and then, repeating the application of the interleaving rule, and applying property (9) of the *Aging* operator, we obtain the observable transition $Q|[G]|R = \langle gt\rangle,0 \Rightarrow Q'|[G]|R'$.

(iii) $g \notin G$, $R - gt \rightarrow R'$, $P' = Age(t,P)$, and $P - et' \rightarrow P''$ with $t \leq t'$. Then, by definition of empty transitions, we have $P - \epsilon t \rightarrow Age(t,P)$, and as $P \sim Q$ we will have some observable transition $Q = \langle\rangle,t \Rightarrow Q'$ with $Age(t,P) \sim Q'$. Some extended transition generating this observable one could be decomposed in the form $Q - q_i \rightarrow Q'' - \epsilon(t - t'') \rightarrow Q'$, where q_i is a –possibly empty– sequence of internal actions that takes t'' instants to execute. We also have $Q' = Age(t - t'', Q'')$. And finally we can interleave the computations $R - gt \rightarrow R'$ and $Q - q_i \rightarrow Q''$, obtaining $Q|[G]|R = \langle gt\rangle,0 \Rightarrow Age(t - t'', Q'')|[G]|R'$, as desired.

Remark: The reason why we do not allow observable transitions generated

by extended transitions of length zero is related to the current case. If the (now disallowed) idle extended transition $Q - \langle\rangle \rightarrow Q$ could generate the observable transition $Q = \langle\rangle, 0 \Rightarrow Q$, then we would have no transition $Q - \epsilon 0 \rightarrow Q'$, and then if $t = 0$ we could not apply the interleaving rule to generate the desired computation $Q|[G]|R = \langle g 0\rangle, 0 \Rightarrow Age(0, Q'')|[G]|R'$.

A consequence of the present definition of observable transition (at first glance a bit tricky) is that:

$$i0\ ;\ stop \quad \not\sim \quad stop$$

These two behaviours could be made equivalent if we allow $B = \langle\rangle, 0 \Rightarrow Age(0, B)$ for any B, and in particular $stop = \langle\rangle, 0 \Rightarrow Age(0, stop)$. We have not allowed this transition because it is clear that with either of the definitions we would have:

$$a0; stop \ \ |[\emptyset]|\ \ i0; stop \quad \not\sim \quad a0; stop$$

With the choice made, we obtain the congruency of all the basic operators except choice, not only over basic behaviours but also over arbitrary ones. As a matter of fact, our proof already covers the general case.

Anyway, if the reader dislikes our choice and prefers the definition which leads to $i0; stop \sim stop$, the proof of congruency of the basic operators except choice is still possible if it is made on behaviours that are basic or derivatives of basic ones. But this proof is much more difficult because a careful application of lemma 1 is necessary in order to prove that *stop* or equivalent behaviours can never appear. This would lead us to a difficult situation since we have seen that in this case congruency is lost when these behaviours appear.

Concerning non-observable transitions $P|[G]|R - xt \rightarrow P'|[G]|R'$ with $xt \in IT$, we will have that either $x = i$ or $x = \epsilon$. In the first case, we will be in one of the following subcases:

(i) $P - it \rightarrow P'$, $R' = Age(t, R)$, and $R - et' \rightarrow R''$ with $t \leq t'$. Then we have some $Q = \langle\rangle, t \Rightarrow Q'$ with $Q' \sim P'$. As before, some extended transition generating this observable one, can be decomposed as $Q - q_i \rightarrow Q'' - \epsilon(t - t'') \rightarrow Q'$, where q_i is a possibly empty sequence of internal actions which needs t'' instants to get executed. Then we have $Q|[G]|R - q_i \rightarrow Q''|[G]|Age(t'', R)$, and we can apply the parallel rule, obtaining $Q''|[G]|Age(t'', R) - \epsilon(t - t'') \rightarrow Q'|[G]|Age(t - t'', Age(t'', R))$, and finally $Q|[G]|R = \langle\rangle, t \Rightarrow Q'|[G]|Age(t, R)$.

(ii) $R - it \rightarrow R'$, $P' = Age(t, P)$, and $P - et' \rightarrow P''$ with $t \leq t'$. This case is identical to the corresponding one for the previously considered case of transitions executing an observable action.

Finally, if $x = \epsilon$, we must apply the parallel rule. Then we would also have $P - \epsilon t \rightarrow P'$ and $R - \epsilon t \rightarrow R'$, and the rest is as in the previous case.

- **Hiding:** We will prove that $WTB = \{(hide\ G\ in\ P,\ hide\ G\ in\ Q) \mid P, Q \in \mathscr{P},\ P \sim Q\}$ is a weak timed bisimulation.

(i) Let $hide\ G\ in\ P - it \rightarrow hide\ G\ in\ P'$ be an internal transition generated by hiding a transition of P corresponding to some action $g \in G$. Then we have $P - gt \rightarrow P'$, and since $P \sim Q$, we have some Q' with $Q = \langle at\rangle, 0 \Rightarrow Q'$ with $P' \sim Q'$. Then we also have $hide\ G\ in\ Q = \langle\rangle, t \Rightarrow hide\ G\ in\ Q'$.

(ii) Any other transition of $hide\ G\ in\ P$ is also a transition of P, and then the corresponding transitions of Q are also transitions of $hide\ G\ in\ Q$. □

As a final result, before turning to the applications of TIC, we will assert without providing the proof, that our calculus is indeed a timed extension of the untimed one. The proof is again a lengthy but easy check. Let us define the mapping between the timed and the untimed case.

Definition 11.

(i) We define (the different classes of) untimed behaviours as basic timed ones, but removing time labels from the action prefix operator, and eliminating the timed choice operator. The operational and bisimulation semantics of untimed behaviours are defined in the usual way (see [ISO88]).

(ii) Let B be an untimed behaviour. We define its associated timed behaviour $Timed(B)$ by substituting any appearance of an action prefix operator a; in it, by the Timed Choice operator $a[0,\infty]$; and any appearance of a LOTOS *stop* by an *idle*. □

Theorem 5. Let B, B_1, B_2 untimed behaviours, then we have

(i) $B - a \rightarrow B' \Rightarrow \forall t \in I\!N \;\; \exists B_t \;\; B_t = Timed(B') \;\; Timed(B) - at \rightarrow B_t$

(ii) $Timed(B) - at \rightarrow B_t \Rightarrow \exists B' \;\; B_t = Timed(B') \;\; B - a \rightarrow B'$

(iii) $B_1 = B_2 \Longleftrightarrow Timed(B_1) = Timed(B_2)$

(iv) $B_1 \sim B_2 \Longleftrightarrow Timed(B_1) \sim Timed(B_2)$

where $=$ represents in each case the corresponding notion of strong bisimulation equivalence. □

Remark: Observe that we have defined untimed behaviours taking *idle* and not *stop* as the sole constant. If you prefer to name by *stop* the constant in the untimed algebra, you must take care of substituting *stop*s by *idle*s when computing the associated timed behaviours; otherwise the theorem would not work at all. Remember that

$$i \; ; \; stop \quad \sim \quad stop$$

but

$$i[0,\infty] \; ; \; stop \quad \not\sim \quad stop$$

On the other hand we have:

$$i[0,\infty] \; ; \; idle \quad \sim \quad idle$$

as desired.

6. Applications

Examples of application of TIC to model behaviours, where precise timing constraints are necessary, are presented in this section. This timed calculus has been developed with the purpose of modelling systems which have a time dependent behaviour. Such behaviours are understood as situations where the time at which the events occur may influence not only the performance of a system, but also the correctness behaviour.

The first example is a stop and wait protocol which illustrates the way time outs are modelled in TIC. Time outs are one of the main reasons why quantitative time is needed in asynchronous protocols.

The second example is a railroad crossing. This example is very adequate to illustrate a timing dependent behaviour, because the correct behaviour of the system depends entirely on the timing of the individual actions.

6.1. A Stop and Wait Protocol

The time-out mechanism is one of the most frequent time dependent behaviours in the protocol world. A *stop and wait protocol* is used to show how TIC models such a behaviour. As shown in Fig. 5, a whole unidirectional communication link is specified including a transmitter process (transmitting entity), a receiver process (receiving entity) and a line process, which is a simplified model of a semiduplex line.

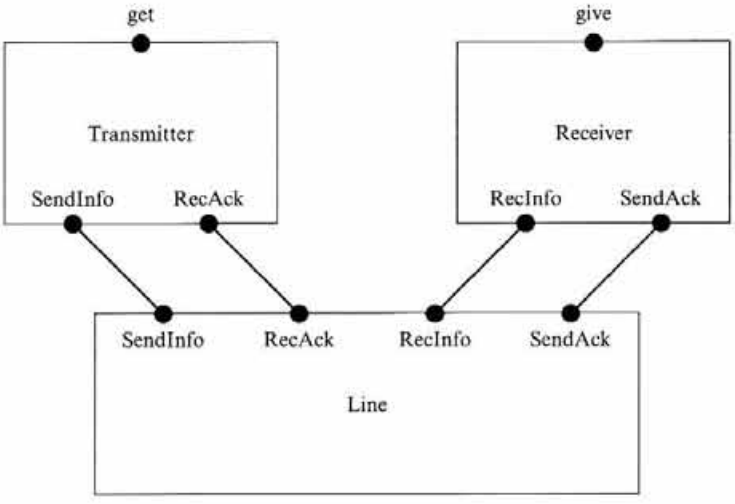

Fig. 5. The stop and wait protocol.

The stop and wait protocol is a very simple and unreliable protocol, but it is small enough to allow its inclusion here. The protocol uses two types of frames: information and acknowledgment frames. The transmitter sends an information frame `SendInfo` and waits for an acknowledgment frame `RecAck`. It also has a time-out mechanism that causes a retransmission of the previous information frame if the acknowledgment frame does not arrive before a given time. Once the acknowledgment frame has been received, new data can be accepted from the user. Once it has been obtained, they can be sent in a new information frame.

The receiver waits for information frames. Each time an information frame is received, the data part is given to the user and an acknowledgment frame is sent back. This protocol is unreliable and may duplicate information, but shows quite clearly how a timed specification evolves in time.

The time-out mechanism and the loss of data in the line because of errors are both modelled with internal events.

```
LINK := ((Transmitter |[]| Receiver)
|[SendInfo,SendAck,RecInfo,RecAck]| Line)

Transmitter :=
        get {0..no_limit}       (* waiting for a new information unit *)
      ; SENDING
```

```
SENDING :=
        SendInfo 8                          (* transmit frame *)
      ; (    RecAck {0..39}                 (* ack arrives *)
           ; Transmitter
        []   i 40                           (* time-out *)
           ; SENDING                        (*retransmission*)
        )

Receiver :=
        RecInfo {0..no_limit}
      ; give 0
      ; SendAck 2
      ; Receiver

Line :=
    (   SendInfo {0..no_limit}
      ; (    RecInfo 10                     (* transmission time*)
           ; Line
        []   i 10                           (* transm. errors, transm. time *)
           ; Line
        )
    []  SendAck {0..no_limit}
      ; (   RecAck 10                       (* transmission time*)
          ; Line
        []  i 10                            (* transm. errors, transm. time *)
          ; Line
        )
    )
```

The timing assigned tries to model a typical point to point line, where a fixed transmission delay exists and a variable transmission time depending on the length of the frame and the modem setup time. Fig. 6 explains the interpretation of each one.

`get {0..no_limit}`	Waiting to get user data
`SendInfo 8`	Frame Sending time
`RecAck {0..39}`	Waiting for RecAck before t-out
`i 40`	Time out
`RecInfo {0..no_limit}`	Waiting for info frame
`give 0`	Give data to user immediately
`SendAck 2`	Ack Sending time
`RecInfo 10`	Line delay
`RecAck 10`	Line delay
`i 10`	Line delay

Fig. 6. Meaning of the events.

Event occurrence is assumed instantaneous. TIC events represent the end of the real events. For example `SendInfo 8` represents the instant where the last bit goes into the line interface or `RecInfo 10` the instant of time where the last bit is received at the other end.

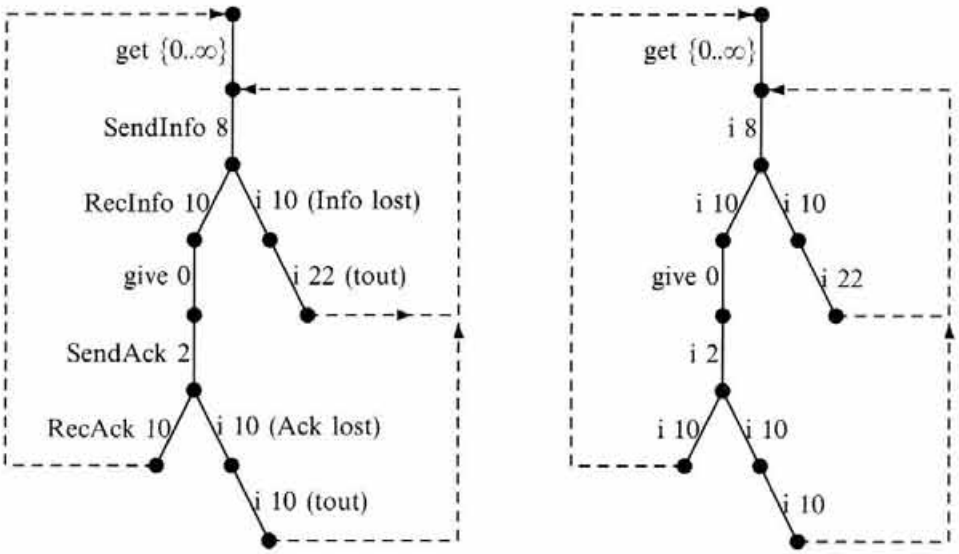

Fig. 7. Evolution of the stop and wait protocol.

The evolution of the system is represented graphically by the tree shown in Fig. 7. The dashed lines represent a recursive definition in the transitions generated. In the left tree all the events remain visible. In the rightmost tree all the communication between the transmitter and the line and between the receiver and the line is hidden. This evolution shows clearly how the information can be duplicated by such a protocol.

The important difference between the evolution of the system obtained here and other asynchronous calculi lies in the way the merging of events is done. In a non timed interpretation (for example, LOTOS without time) other evolutions of the system which do not make sense in a timed context would have occurred.

6.2. The Railroad Crossing Example

A second example of the use of TIC which has been taken from Leveson's paper [LeS87] where it is used as a paradigm of time is presented in this section. The example models a simple railroad crossing. The system is composed of three parts (Fig. 8): the train model, the computer or controlling device model and the gate model.

The exact TIC description of the railroad crossing is given in the specification of `RailRoadCrossing`. Each process of this specification models one part of the system: `Sensors` models the train approach as detected by the sensors, `Controller` models the computer o controlling device and `Gate` models the gate behaviour. The time attributes of the specification reflect a given set of hypotheses about train speed, distance between trains and gate moving times.

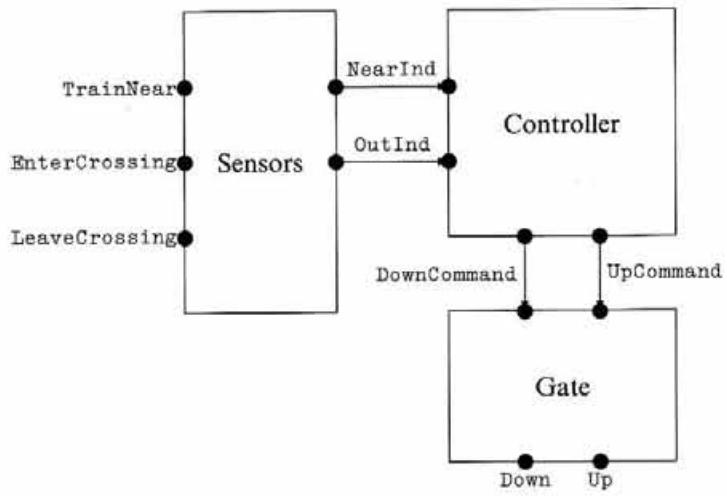

Fig. 8. The railroad crossing: Block structure.

```
Process RailRoadCrossing :=

(   (                    Sensors
    |[NearInd, OutInd]| Controller
    )
|[DownCommand, UpCommand]| Gate
)
Endproc

Process Sensors :=

   TrainNear {0..no_limit}   (* waiting for the train approaching *)
 ; NearInd 0                 (* sensor sends signal to controller *)
 ; EnterCrossing 3000        (* train enters the crossing *)
 ; LeaveCrossing 20          (* train leaves the crossing *)
 ; OutInd 0                  (* sensor sends signal to controller *)
 ; i 101                     (* stabilization delay *)
 ; Sensors
Endproc

Process Controller :=

  (   NearInd {0..no_limit}  (* controller waiting for NearInd signal *)
    ; DownCommand 0          (* controller sends command to gate *)
    ; Controller

  []  OutInd {0..no_limit}   (* controller waiting for OutInd signal *)
    ; UpCommand 0            (* controller sends command to gate *)
    ; Controller
  )
Endproc

Process Gate :=

  (   DownCommand {0..no_limit} (* gate waiting for command *)
```

```
    ; Down 100                    (* gate reaches down position *)
    ; Gate

  []  UpCommand {0..no_limit}     (* gate waiting for command *)
    ; Up 100                      (* gate reaches up position *)
    ; Gate
  )
Endproc
```

The evolution of the timed system is shown in Fig. 9. The left hand side evolution includes all the events of the model. The right hand side shows an abstraction of the evolution where the internal communication between the parts has been removed. This simplified evolution is weak bisimulation equivalent to the `RailRoadCrossing` behaviour when the internal communication between parts is hidden, i.e. gates `NearInd`, `OutInd`, `DownCommand` and `UpCommand`.

The correctness of the behaviour should be determined by assuring that the `Down` event occurs before the `EnterCrossing` event. Some safety time interval should be also assured between both events.

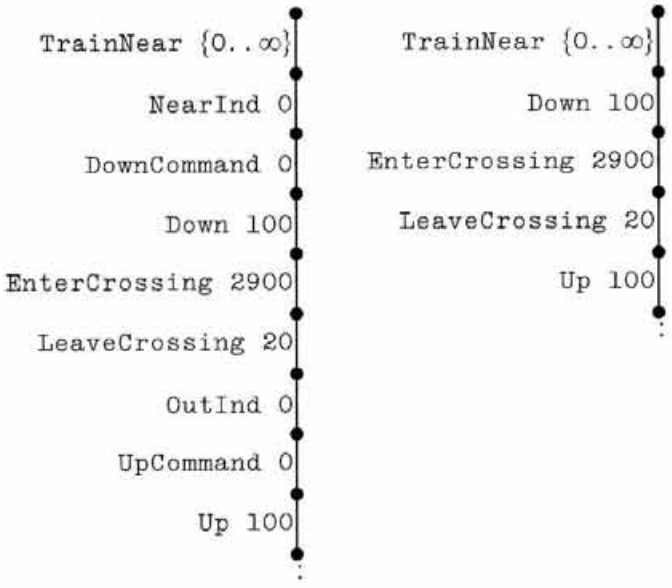

Fig. 9. The railroad crossing: Evolution in time.

7. Conclusions and Further Work

The paper presents a calculus which is an upward compatible extension of basic LOTOS. The calculus differs strongly from the untimed case in the definition of interleaving of events in the presence of timing constraints. The paper also includes the definition of timed bisimulation equivalences. The three main goals of this development have been: 1) To give a definition of interleaving for which

equivalences may be given, which has similar properties to the untimed case. 2) To define the interleaving in such a way that the merge of events in time of interleaved behaviours follows the natural evolution of events in time. 3) To have an operational definition of the calculus, so that it is possible to extend the definition of tools like LOLA [QPF89], to the timed case. The three goals have been achieved, although the definition of the new interleaving operator may be a bit complex.

From the expressiveness point of view, the compositionality in time of interleaved behaviours is adequate. This means that you can compose timed systems, and the evolution of the composition models the natural evolution of such a system. On the other hand, the new interleaving could look rather counterintuitive, because interleaved behaviours are not independent. Nevertheless, this definition of the evolution of a system has given no problems in any of the studied examples. The main limitation of TIC is its inability to represent *as soon as possible (asap)* timing requirements. The introduction of asap requirements is one of the future directions of work.

The use of null separation time between events is also a controversial point. From the expressiveness point of view, it is a good approximation of negligible time separation, and from the semantic point of view it is necessary for the representation of interleaving. To be exact, it is necessary, in that the calculus does not allow simultaneous execution of two events. The introduction of real parallelism to represent simultaneous events could have avoided the allowance of the zero separation time.

From the applicability point of view, the calculus has been developed to make possible the study of the evolution of systems in time. This evolution will be studied mainly by state exploration and testing, using tools like LOLA [QPF89], which can be easily adapted to the timed semantics, and hopefully will have a more or less similar performance as in the non timed case. Testing TIC processes is a practical necessity, so a definition and study of testing equivalence is needed.

Although it seems that the defined equivalences have the properties required from the application point of view, the development of a complete equational characterization of the weak timed bisimulation would be of interest because it would provide a more complete knowledge of the properties of the equivalence, and it would probably lead to a denotational model of the induced semantics.

Acknowledgements

The authors would like to thank those who have contributed to this work with ideas, suggestions or reviews. Among them, Carlos Miguel Nieto deserves specially grateful thanks for his reviews of the many versions of the paper, which have been very valuable. Angel Fernandez del Campo, Jose Manas Argemi, Yolanda Ortega Mallen and Tommasso Bolognesi must also be thanked for their useful comments on the different versions of this calculus.

References

[AuB87] Austry, D. and Boudol, G.: Algebre de Processus et Synchronisation. *J TCS*, 53:225–241, 1987.

[Azc89] Azcorra, A.: *Modelado Formal de Sistemas Sincronos.* PhD thesis, Escuela Tecnica Superior de Ingenieros de Telecomunicacion, Universidad Politecnica de Madrid, 1989.

[BaB91] Baeten, J. C. M. and Bergstra, J. A.: Real Time Process Algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.

[BeK85] Bergstra, J. A. and Klop, J. W.: Algebra of Communicating Processes with Abstraction. *Theoretical Computer Science*, 37(1):77–121, 1985.

[BoL91] Bolognesi, T. and Lucidi, F.: LOTOS-like Process Algebras with Urgent or Timed Interactions. In *FORTE'91: Formal Techniques IV*, Sidney, November 1991.

[BLT90] Bolognesi, T., Lucidi, F. and Trigila, S.: From Timed Petri Nets to Timed LOTOS. In L. Logrippo, R. Probert, and H. Ural, editors, *Tenth International IFIP Symposium on Protocol Specification, Testing and Verification*, pages 377–406. North-Holland, June 1990.

[BeR85] Bolognesi, T. and Rudin, H.: On the Analisys of Time-Dependent Protocols by Network Flow Algorithms. In *Fifth International Workshop on Protocol Specification, Testing and Verification*, New York, June 1985.

[BSS86] Brinksma, E., Scollo, G. and Steenbergen, C.: LOTOS Specifications, their Implementation and their Tests. In *Sixth International Workshop on Protocol Specification, Testing and Verification*, Montreal, June 1986.

[dNH84] Nicola, R. de. and Hennessy, M.: Testing Equivalences for Processes. *Theoretical Computer Science*, 34(1,2):83–133, Nov 1984.

[GeB87] Gerth, R. and Boucher, A.: *A timed Failures Model for Extended Communicating Processes*, volume LNCS. ICALP 87, 1987.

[Hoa85] Hoare, C.A.R.: *Communicating Sequential Processes*. Prentice-Hall Int., 1985.

[HeR90] Hennessy, M. and Regan, T.: A Temporal Process Algebra. In *FORTE'90: Formal Techniques III*, Madrid, November 1990.

[ISO88] ISO. LOTOS a Formal Description Technique based on the Temporal Ordering of Observational Behaviour. IS 8807, TC97/SC21, 1988.

[K+85] Koymans, R., et al.: Compositional Semantics for Real Time Distributed Computing. In *Conference on Logics of Programs*. Springer Verlag, 1985.

[LeS87] Leveson, N.G. and Stolzy, J.L.: Safety analysis using Petri nets. *IEEE Transactions on Software Engineering*, 13(3), 1987.

[MeF76] Merlin, P. M. and Farber, D. J.: Recoverability of Communication Protocols - Implication of a theoretical Study. *IEEE Trans. on Com.*, 24:1036–1043, Sep 1976.

[Mig91] Nieto, C. M.: *Tecnicas de descripcion Formal aplicadas la Evaluacion de Prestaciones de Sistemas de Comunicacion*. PhD thesis, Escuela Tecnica Superior de Ingenieros de Telecomunicacion, Universidad Politecnica de Madrid, 1991.

[Mil80] Milner, R.: *Calculus of Communicating Systems*. Number 92 in Lecture Notes in Computer Science. Springer Verlag, Berlin, 1980.

[Mil83] Milner, R.: Calculi for Synchrony and Asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil85] Milne, G.: CIRCAL and the Representation of Communication, Concurrency and Time. *ACM, TOPLAS*, 7(2):270–298, April 1985.

[MoT90] Moller, F. and Tofts, C.: *A Temporal Calculus of Communicating Systems*, pages 401–415. Number LNCS-458, ISBN 3-540-53048-7 in Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, New York, 1990.

[NRS90] Nicollin, X., Ritchier, J. L., Sifakis, J. and Voiron, J.: ATP: An Algebra for Timed Processes. In *TC2 Working Conference on Programming Concepts and Methods*. North Holland, 1990.

[OdF90] Ortega, Y. and de Frutos, D.: Timed Observations: A semantic Model for Real-Time Concurrency. In *TC2 Working Conference on Programming Concepts and Methods*. North Holland, 1990.

[OdF91] Ortega, Y. and de Frutos, D.: A Complete Proof System for Timed Observations. In *TAPSOFT'91 (CAAP'91)*. LNCS 493, Springer Verlag, 1991.

[Par81] Park, D.: *Concurrency and Automata on Infinite Sequences*, volume 104 of *LNCS*. Springer-Verlag, 1981.

[QAF89] Quemada, J., Azcorra, A. and de Frutos, D.: A Timed Calculus for LOTOS. In *FORTE'89: Formal Techniques II*, Vancouver, December 1989.

[QuF87] Quemada, J. and Fernandez, A.: Introduction of Quantitative Relative Time into LOTOS. In *IFIP workshop on Protocol Specification, Testing and Verification : VII*, Zurich, May 5, 1987.

[QPF89] Quemada, J., Pavon, S. and Fernandez, A.: State Exploration by Transformation with LOLA. In *Workshop on Automatic Verification Methods for Finite State Systems*, Grenoble, June 1989.

[ReR86] Reed, G. M. and Roscoe, A. W.: A timed model for communicating sequential processes. In *ICALP 86*, volume LNCS 226. Springer Verlag, 1986.

[ReR87] Reed, G. M. and Roscoe, A. W.: *Metric Spaces as Models for Real-Time Concurrency*. Springer Verlag, 1987.

[Tof88] Tofts, C.: Temporal Ordering for Concurrency. Technical report, University of Edinburgh, April 1988.

Received August 1989
Accepted in revised form May 1992 by R. Milner