A Complete Proof System for Timed Observations

Yolanda Ortega-Mallén
David de Frutos-Escrig
Sección Departamental de Informática y Automática
Facultad de Matemáticas
Universidad Complutense de Madrid, Spain*

August 1990

Abstract

Timed Observations is a failure and divergence semantic model for concurrent processes, suitable for real-time systems. Actions are not instantaneous but need some time to complete their execution, and "true" concurrency is expressed by action multisets (bags). Time is global and discrete.

The model is applied to TCSP, obtaining a denotational semantics, for which a complete proof system is developed.

1 Introduction

In [OdF90] a new failure semantic model for concurrent processes, suitable for real-time systems, was presented. The model intended to stress the fact that the actions performed by a process are not instantaneous, but need a certain amount of time to complete.

Following the ideas of Milner's [Mil80], that a concurrent process is not an isolated item, but part of a whole system of interacting machines and users; what is interesting about a process is its external behaviour, i.e. what an external observer (an user or another process) can notice. An ordinary observer takes notice of the visible actions performed by the process (trace); but our observer has a clock, so that he relates each action to the instant when it is executed (timed traces). Moreover he is an active observer, he tries to guess the future behaviour of the process by asking it to perform some actions from a selection (refusals) [BR85]. Finally, as we are interested in differentiating divergence (unbounded sequence of internal actions) from deadlock (doing nothing), we *ideally* suppose that the possibility of divergence is observable (of course, divergence is

*Part of the work was developed during a two-month sojourn of the first author as collaborator in the RWTH-Aachen (Lehrstuhl für Informatik II) (FRG) and a short visit of the second author to this same center.

not computable in general). Therefore the model is a timed and generalized version of failure semantics plus divergences.

The model induced a specification-oriented denotational semantics [OH86] which we applied to TCSP [Old87].

There are several research groups working on timed models. Some of them have produced denotational or compositional semantics like [KSdR*85,GB87,RR87]; some others have developed timed process algebras or calculi, like [Tof88,NRSV90,HR90,BB90] among others. In the present paper we try to combine both approaches and give an equational proof system for the denotational semantics for TCSP processes, given in [OdF90]. This proof system is proven to be correct and complete.

The paper is structured as follows: section 2 introduces the basic notions and definitions in the model. In section 3 we describe the corresponding denotational semantics for TCSP-processes. In section 4 we present the proof system.

2 Timed Observations

The model given here is slightly different from the one presented in [OdF90]. In this section we give its main concepts and definitions. The interested reader is refered to [Ort90].

2.1 Action duration

Let's start defining the kind of things we can observe of a processes. First of all we fix the set of *actions* that the processes may perform: the *finite alphabet* $\mathcal{A}$. These actions are indivisible but they are not instantaneous, i.e. we assume a *duration* function

$$d : \mathcal{A} \longrightarrow \mathbb{N}^+$$

which associates to each action the time it needs to complete its execution (in temporary units).

As actions are no longer instantaneous, hidden actions do not disappear completely from the trace, because its duration time will remain reflected.

2.2 Action Bags

As we are interested in a general model for concurrent processes, we need some way of reflecting the possibility of having several actions executing simultaneously. In order to express this the most natural way[1] is to consider *action bags* (or multisets). One important restriction is that bags are always finite, corresponding to the fact that the execution of an infinite amount of simultaneous actions is unrealistic. Sets (and multisets) appear in several models for concurrent processes like [Mil83a,Mil83b,TV89,Azc89].

[1] Classical theories for concurrency are based on *interleaving* semantics where concurrency is reduced to sequentiality + non-determinism. This is derived from the fact that on each process-step only one action is executed. However more recent works on this topic tend to consider concurrency as a language primitive, giving way to what is known as *true concurrency* [BC88].

Def. 2.1 *The set of* **bags over an alphabet** $\mathcal{A}$ *is:* $\mathcal{B}(\mathcal{A}) = \{B : \mathcal{A} \longrightarrow \mathbb{N}\}$.

Let $a \in \mathcal{A}$, *some interesting bags and bag sets are defined:*

empty bag: $B_\emptyset \stackrel{\text{def}}{=} \forall a \in \mathcal{A}, B_\emptyset(a) = 0$,

bag generated by a**:** $B_a \stackrel{\text{def}}{=} \forall b \in \mathcal{A}, B_a(b) = \begin{cases} 0 & \text{if} \quad b \neq a \\ 1 & \text{if} \quad b = a \end{cases}$

non-empty bags: $\mathcal{B}^+(\mathcal{A}) = \mathcal{B}(\mathcal{A}) - \{B_\emptyset\}$,

bags containing a**:** $\mathcal{B}_a(\mathcal{A}) = \{B \in \mathcal{B}(\mathcal{A}) \mid B(a) > 0\}$,

bags different from B_a**:** $\mathcal{B}_{-a}(\mathcal{A}) = \mathcal{B}(\mathcal{A}) - \{B_a\}$.

When the alphabet is understood, it may be omitted obtaining $\mathcal{B}, \mathcal{B}^+, \mathcal{B}_a, \ldots$. □

2.2.1 Operations on bags

When defining the semantics we will need some operations and definitions on action bags.

Let $a \in \mathcal{A}$; $A \subseteq \mathcal{A}$; $B, B_1, B_2 \in \mathcal{B}(\mathcal{A})$; $n \in \mathbb{N}$.

Alphabet: $\mathcal{A}(B) = \{a \in \mathcal{A} | B(a) > 0\}$.

Size: $|B| = \sum_{a \in \mathcal{A}} B(a)$.

Partial order: $B_1 \leq_{\mathrm{B}} B_2 \stackrel{\text{def}}{=} \forall a \in \mathcal{A}, B_1(a) \leq B_2(a)$.

Addition: $B_1 +_{\mathrm{B}} B_2 \stackrel{\text{def}}{=} \forall a \in \mathcal{A}, B_1 +_{\mathrm{B}} B_2(a) \stackrel{\text{def}}{=} B_1(a) + B_2(a)$,

As an abbreviation we define, $0 * B = B_\emptyset$,
$(n+1) * B = n * B +_{\mathrm{B}} B$.

Hiding: $B \backslash a \stackrel{\text{def}}{=} \forall b \in \mathcal{A}, B \backslash a(b) = \begin{cases} B(b) & \text{if} \quad b \neq a \\ 0 & \text{if} \quad b = a \end{cases}$

Restriction: $B \lceil A \stackrel{\text{def}}{=} \forall b \in \mathcal{A}, B \lceil A(b) = \begin{cases} B(b) & \text{if} \quad b \in A \\ 0 & \text{if} \quad b \notin A \end{cases}$

Synchronization: it can only be applied when $B_1 \lceil A = B_2 \lceil A$,

$$B_1 \oplus_A B_2 \stackrel{\text{def}}{=} \forall a \in \mathcal{A}, B_1 \oplus_A B_2(a) = \begin{cases} B_1(a) & \text{if} \quad a \in A \\ B_1(a) + B_2(a) & \text{if} \quad a \notin A \end{cases}$$

We will use the actions bags to generalise the notion of failure [BR85], which embodies traces and refusals.

2.3 Timed traces

Classical traces reflect some temporal order on the executed actions. Nevertheless this temporal order is insufficient for the study of real-time system behaviour; a more absolute time notion is needed, thereby we relate each action ocurrence to the instant when it is produced (more exactly to the instant when it starts, as actions are not instantaneous), obtaining *timed traces*. Actions can be performed from instant 1 on (instant 0 represents the initial state of the process).

Def. 2.2 *The set of* **timed traces** $\mathcal{TT}$ *is:* $\mathcal{TT} = \{t : \mathbb{N}^+ \longrightarrow \mathcal{B}\}$.
An special and useful trace is the **empty trace** $t_\emptyset$*:* $t_\emptyset \stackrel{\text{def}}{=} \forall i \in \mathbb{N}^+, t_\emptyset(i) = B_\emptyset$. □

The empty bag now plays an important role, as it represents the passing of time with the process doing no externally visible action.[2]

Def. 2.3 *Let* $t \in \mathcal{TT}$,

- *the first non-empty time instant:* $$\begin{array}{rcl} \inf(t_\emptyset) & = & 0, \\ \inf(t) & = & \min\{n | t(n) \neq B_\emptyset\}, t \neq t_\emptyset. \end{array}$$
- *the last non-empty time instant:* $$\begin{array}{rcl} \sup(t_\emptyset) & = & 0, \\ \sup(t) & = & \max\{n | t(n) \neq B_\emptyset\}, t \neq t_\emptyset. \end{array}$$

□

From a practical point of view, processes can only be observed during finite time intervals:

Def. 2.4 *The set of* **finite timed traces** $\mathcal{TTF}$ *is:* $\mathcal{TTF} = \{(t,n) | t \in \mathcal{TT} \wedge n \geq \sup(t)\}$.
Let $tf = (t,n) \in \mathcal{TTF}$*, we define:* $trace(tf) = t$ and $end(tf) = n$. □

Notice that $end(tf) = 0$ implies $trace(tf) = t_\emptyset$. The trace $(t_\emptyset, 0)$ corresponds to the initial state of the process, when the observer has not yet started to observe.

2.3.1 Operations on traces

Let $A \subseteq \mathcal{A}; n \in \mathbb{N}^+; B \in \mathcal{B}; t \in \mathcal{TT}; tf, t_1, t_2 \in \mathcal{TTF}$,

Alphabet: $\mathcal{A}(tf) = \bigcup_{1 \leq i \leq end(tf)} \mathcal{A}(trace(tf)(i))$.

Adding traces: $t_1 +_{\mathrm{T}} t_2 \stackrel{\text{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(t_1 +_{\mathrm{T}} t_2)(i) = trace(t_1)(i) +_{\mathrm{B}} trace(t_2)(i) \\ end(t_1 +_{\mathrm{T}} t_2) = \max\{end(t_1), end(t_2)\} \end{cases}$

[2]In distinction to [TV89] where empty bags are only included to facilitate the definition of the synchronizing operator, but they can be arbitrarily added or eliminated.

Adding a bag to a trace:

$$tf +_{\mathrm{TB}} (B, n) \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(tf +_{\mathrm{TB}} (B,n))(i) = \begin{cases} trace(tf)(i) & \text{if} \quad i \neq n \\ trace(tf)(i) +_{\mathrm{B}} B & \text{if} \quad i = n \end{cases} \\ end(tf +_{\mathrm{TB}} (B,n)) = \max\{end(tf), n\} \end{cases}$$

We can define the trace including only one action: $t_{n,a} \stackrel{\mathrm{def}}{=} trace((t_\emptyset, 0) +_{\mathrm{TB}} (B_a, n))$.

Moving along time:

$$\mathrm{mov}(tf, n) \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(\mathrm{mov}(tf,n))(i) = \begin{cases} trace(tf)(i-n) & \text{if} \quad i > n \\ B_\emptyset & \text{if} \quad i \leq n \end{cases} \\ end(\mathrm{mov}(tf, n)) = end(tf) + n \end{cases}$$

Concatenation: $t_1 \cdot_{\mathrm{T}} t_2 \stackrel{\mathrm{def}}{=} t_1 +_{\mathrm{T}} \mathrm{mov}(t_2, end(t_1))$.

Stretch: $Stretch(tf)$[3],

$$Stretch(t_\emptyset, n) = \{(t_\emptyset, m) | m \geq n\},$$
$$Stretch((t,n) +_{\mathrm{TB}} (B, n+1)) =$$
$$\{tf +_{\mathrm{TB}} \textstyle\sum_{k=1}^{m} (B_k, n_k) | \sum_{k=1}^{m} B_k = B \wedge n_k \geq n+1 \wedge tf \in Stretch(t,n)\}.$$

Actions in a bag may be "stretched" and executed in different time instants. The only restriction is to keep the relative order between bags in the original trace.

Hiding: $tf \backslash a \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(tf \backslash a)(i) = trace(tf)(i) \backslash a \\ end(tf \backslash a) = end(tf) \end{cases}$

Restriction to an action set: $tf \lceil A \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(tf \lceil A)(i) = trace(tf)(i) \lceil A \\ end(tf \lceil A) = end(tf) \end{cases}$

Restriction to an instant: $tf \lceil n \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(tf \lceil n)(i) = \begin{cases} trace(tf)(i) & \text{if} \quad i \leq n \\ B_\emptyset & \text{if} \quad i > n \end{cases} \\ end(tf \lceil n) = n \end{cases}$

Synchronization: it can only be applied when $t_1 \lceil A = t_2 \lceil A$,

$$t_1 \oplus_A t_2 \stackrel{\mathrm{def}}{=} \begin{cases} \forall i \in \mathbb{N}^+, trace(t_1 \oplus_A t_2)(i) = trace(t_1)(i) \oplus_A trace(t_2)(i) \\ end(t_1 \oplus_A t_2) = end(t_1) \ (= end(t_2)) \end{cases}$$

Operators on finite timed traces can also be applied to non-finite timed traces.

[3]This operation is borrowed from [TV89].

2.4 Timed failures

Failures consist on traces plus a refusal set. A refusal is a *finite* set of non-empty action bags. These sets can be empty, but empty bags are not considered, because refusing the empty bag means that the process refuses to do nothing, but our model supports "no maximum parallelism"[SM81],i.e. actions can arbitrarily delay its execution, so that every process can always choose to let the time pass doing nothing!

Def. 2.5 *The set of* **refusals** $\mathcal{R}$ *is :* $\mathcal{R} = \{r \in \mathcal{PF}(\mathcal{B}^+)\}$. □

Def. 2.6 *Take* $r_1, r_2 \in \mathcal{R}$ *we define its* **synchronization***:*
$r_1 \oplus_A r_2 \stackrel{\text{def}}{=} \mathcal{PF}(\{B \in \mathcal{B}^+ | \forall B_1, B_2 : B = B_1 \oplus_A B_2 \Rightarrow B_1 \in r_1 \vee B_2 \in r_2\})$. □

Def. 2.7 *The set of* **timed failures**[4] $\mathcal{FT}$ *is:* $\mathcal{FT} = \{\langle tf, r\rangle | tf \in \mathcal{TTF} \wedge r \in \mathcal{R}\}$.
Take $f = \langle tf, r\rangle \in \mathcal{FT}$, *we define:* $T(f) = tf$ and $R(f) = r$. □

2.5 Divergence

A process is said to *diverge* when it is involved in an unbounded sequence of internal actions.

The divergence in our model is not catastrophic; i.e. the feasibility of divergence is not necessarily permanent, it may *disappear* as the process evolves by performing further actions, i.e. by choosing a non-diverging path. Therefore we do not identify a possibly diverging process with *chaos* (the process which has every possible behaviour).[5]

Obviously it is very convenient to be able to distinguish a never diverging process from one which has this possibility. Therefore we suppose that our model is very "skilled" and when the possibility of divergence in a process is detected, this fact is indicated by some externally observable signal.

2.6 Timed observations

Putting all the above concepts together, the observable behaviour of a process consists of a timed failure set plus a divergence set (timed traces which can be extended with an infinite sequence of internal actions).

Def. 2.8 *The set of* **timed observations** $\mathcal{O}$ *is:*

$$\mathcal{O} = \{\langle F, D\rangle | F \subseteq \mathcal{FT} \wedge D \subseteq \mathcal{TTF} \wedge D \subseteq T(F)\}$$

Let $ob = \langle F, D\rangle \in \mathcal{O}$, *we define:* $F(ob) = F$, $D(ob) = D$ and $T(ob) = T(F(ob))$. □

[4] Step Failure Semantics [TV89] is a generalization of classic failure semantics, where action bags (or steps) are considered instead of single actions. Our Timed Failures, although independently developed, could be considered as a timed version of this step failure semantics.

[5] Brookes proved in [Bro83] that the identification of divergence with *chaos* is necessary for the sake of the continuity of the hiding operator in failure semantics. This identification is mainly technical and has no clear intuitive justification.

2.7 Specifications

The notion of *specification* [OH86,Old86] is useful for the design and development of concurrent programs. We start with a specification Sp which denotes a set of behavioural observations that we expect in the behaviour of the program we are looking for. We say a program P is *correct* with respect to Sp (P *sat* Sp), if all the behavioural observations of P are within Sp.

In our model specifications are timed observations ($Sp \in \mathcal{O}$) with some restrictions. To each program P we associate a specification $\mathcal{S}[\![P]\!]$, so that P *sat* Sp when $Sp \leq_{\mathrm{S}} \mathcal{S}[\![P]\!]$, being $\leq_{\mathrm{S}}$ some partial order over $\mathcal{O}$.

- [P1] $(t_{\emptyset}, 0) \in T(Sp)$
- [P2] $(t_1, n_1) \cdot (t_2, n_2) \in T(Sp) \Rightarrow (t_1, n_1) \in T(Sp)$
- [P3] $(t, n) \in T(Sp) \wedge r \in \mathcal{R} \Rightarrow$ $$(\exists B \in r : (t, n) +_{\mathrm{TB}} (B, n+1) \in T(Sp)) \vee \langle (t, n), r \rangle \in F(Sp)$$
- [P4] $\langle tf, r \rangle \in F(Sp) \wedge r' \subset r \Rightarrow \langle tf, r' \rangle \in F(Sp)$
- [P5] $tf \in D(Sp) \Rightarrow \forall m \geq \sup(tf) : tf \lceil m \in D(Sp)$
- [P6] $\langle tf, r \rangle \in F(Sp) \wedge tf' \in \mathit{Stretch}(tf) \Rightarrow \langle tf', r \rangle \in F(Sp)$
- [P7] $tf \in D(Sp) \wedge tf' \in \mathit{Stretch}(tf) \Rightarrow tf' \in D(Sp)$
- [P8] $B \leq_{\mathrm{B}} B' \wedge \langle tf, r \cup \{B\} \rangle \in F(Sp) \Rightarrow \langle tf, r \cup \{B, B'\} \rangle \in F(Sp)$

Def. 2.9 *We define the* **specification space***: $\langle \mathcal{S}, \leq_{\mathrm{S}} \rangle$, where*

$$\mathcal{S} = \{Sp \in \mathcal{O} | Sp \text{ verifies } [\mathrm{P1}] - [\mathrm{P8}]\}$$

is the **set of specifications** *and the* **partial order over timed observations** *is:*

$$ob_1 \leq_{\mathrm{S}} ob_2 \stackrel{\mathrm{def}}{=} F(ob_2) \subseteq F(ob_1) \wedge D(ob_2) \subseteq D(ob_1)$$

with $ob_1, ob_2 \in \mathcal{O}$. $\square$

Props. 2.1 $\langle \mathcal{S}, \leq_{\mathrm{S}} \rangle$ *is a* Complete Partial Order *(CPO).* $\square$

3 Semantics for TCSP-processes

In the previous section we have only given a specification space suitable for defining denotational semantics, but we still must provide, for each syntactic operator in the programming language, a continuous operator over this specification space.

We consider the following set of syntactic operators for TCSP-processes:

$$\Sigma_1 = \{stop\} \cup \{a \rightarrow | a \in \mathcal{A}\} \cup \{\sqcap, \square\} \cup \{\|_A | A \subseteq \mathcal{A}\} \cup \{\backslash a | a \in \mathcal{A}\}$$

Def. 3.1 *A* **TCSP-process** *is a recursive term over* Σ_1 *(*$P \in REC(\Sigma_1)$*):*

$$P ::= stop \mid a \rightarrow P \mid P \sqcap P \mid P \Box P \mid P \|_A P \mid P \backslash a \mid \xi \mid \mu\xi.P$$

where ξ *belongs to some process identifier set Id.* $\Box$

Only recursive closed processes $CREC(\Sigma_1)$ (without free identifiers) are considered.

We now give the semantic operator associated to each syntactic operator. The resulting specification has two components, the first one expressing the timed failures and the second, the divergences. Let $Sp_1, Sp_2 \in \mathcal{S}$,

1. $\mathcal{S}_{stop}[\![\cdot]\!] = \begin{cases} \{\langle (t_\emptyset, m), r\rangle \mid m \in \mathbb{N} \wedge r \in \mathcal{R}\} \\ \emptyset \end{cases}$

2. $\mathcal{S}_{a\rightarrow}[\![Sp]\!] = \begin{cases} \{\langle (t_\emptyset, m), r\rangle \mid m \in \mathbb{N} \wedge r \in \mathcal{PF}(\mathcal{B}^+_{-a})\} \\ \cup\{\langle (t_{n,a}, m), r\rangle \mid n \in \mathbb{N}^+ \wedge n \leq m < nd \wedge r \in \mathcal{R}\} \\ \cup\{\langle (t_{n,a}, nd)\cdot tf, r\rangle \mid n \in \mathbb{N}^+ \wedge \langle tf, r\rangle \in F(Sp)\} \\ \\ \{(t_{n,a}, m) \mid n \in \mathbb{N}^+ \wedge n \leq m < nd \wedge (t_\emptyset, 0) \in D(Sp)\} \\ \cup\{(t_{n,a}, nd)\cdot tf \mid n \in \mathbb{N}^+ \wedge tf \in D(Sp)\} \end{cases}$

   where $nd = n + d(a) - 1$.

3. $\mathcal{S}_{\sqcap}[\![Sp_1, Sp_2]\!] = \begin{cases} F(Sp_1) \cup F(Sp_2) \\ D(Sp_1) \cup D(Sp_2) \end{cases}$

4. $\mathcal{S}_{\Box}[\![Sp_1, Sp_2]\!] = \begin{cases} \{\langle (t_\emptyset, m), r\rangle \in F(Sp_1) \cap F(Sp_2)\} \\ \cup\{\langle tf, r\rangle \in F(Sp_1) \cup F(Sp_2) \mid trace(tf) \neq t_\emptyset\} \\ \\ D(Sp_1) \cup D(Sp_2) \end{cases}$

5. $\mathcal{S}_{\|_A}[\![Sp_1, Sp_2]\!] = \begin{cases} \{\langle tf_1 \oplus_A tf_2, r\rangle \mid \langle tf_i, r_i\rangle \in F(Sp_i), i = 1,2 \wedge r \in r_1 \oplus_A r_2\} \\ \{tf_1 \oplus_A tf_2 \mid \ (tf_1 \in D(Sp_1) \wedge tf_2 \in T(Sp_2)) \\ \qquad\qquad \vee (tf_1 \in T(Sp_1) \wedge tf_2 \in D(Sp_2))\} \end{cases}$

6. $\mathcal{S}_{\backslash a}[\![Sp]\!] = \begin{cases} \{\langle tf\backslash a, r \cup r'\rangle \mid \langle tf, r\rangle \in F(Sp) \wedge r' \in \mathcal{PF}(\mathcal{B}_a)\} \\ \{tf\backslash a \mid \ \exists tf' : \mathcal{A}(tf') \subseteq \{a\} \wedge tf\cdot tf' \in D(Sp) \\ \qquad \vee \exists \{t_i\}_{i\in\mathbb{N}^+} : \mathcal{A}(t_i) = \{a\} \wedge tf\cdot t_1 \ldots \cdot t_i \in T(Sp)\} \end{cases}$

Props. 3.1 $\forall op \in \Sigma_1, \mathcal{S}_{op}[\![Sp_1, \ldots, Sp_n]\!]$ *is well defined (*$\in \mathcal{O}$*), monotonic, and it preserves properties* $[P1] - [P8]$*.* $\Box$

Props. 3.2 $\forall op \in \Sigma_1 - \{\backslash a\}, \mathcal{S}_{op}[\![Sp_1, \ldots, Sp_n]\!]$ *is continuous.* $\Box$

The hiding operator is not continuous on the specification space, due to the fact that we cannot distinguish between the execution of a unique hidden action and the simultaneous execution of several of them, as the time needed is the same.

Example 3.1 Suppose $\mathcal{A} = \{a, b\}$ with $d(a) = d(b) = 1$ and we define:

$$Q = a \rightarrow b \rightarrow stop \|_{\{b\}} Q$$

This process can perform any amount of a actions, but it never performs any b.

Now we consider the process sequence $\{Q_n\}_{n \in \mathbb{N}}$ where

$$\begin{aligned} Q_0 &= chaos^6 \\ Q_n &= a \rightarrow b \rightarrow stop \|_{\{b\}} Q_{n-1} \end{aligned}$$

For each process Q_{n-1} at least n a actions are executed before performing b.

We define $X = \{\mathcal{S}[\![Q_n]\!]\}_{n \in \mathbb{N}}$, therefore X is a directed set and $\mathrm{lub}(X) = \mathcal{S}[\![Q]\!]$. However when applying the hiding operator we obtain $\mathcal{S}_{\backslash a}[\![\mathrm{lub}(X)]\!] = \mathcal{S}_{\backslash a}[\![Q]\!] \neq \mathrm{lub}(\{\mathcal{S}_{\backslash a}[\![Q_n]\!]\})$. Consider the trace $tf = (t_{2,b}, 2)$, then $\forall n \in \mathbb{N} : tf \in T(\mathcal{S}_{\backslash a}[\![Q_n]\!])$ but $tf \notin T(\mathcal{S}_{\backslash a}[\![Q]\!])$.

□

3.1 Guarded processes

If we restrict the dynamic process generation by requiring that every recursive call is preceded by the execution of at least one action, then in a bounded time interval we will obtain a bounded amount of parallel processes. We will call this class of processes *guarded processes.*

Def. 3.2 Well-defined expression E*:*

$$E ::= stop \mid a \rightarrow E \mid E \sqcap E \mid E \Box E \mid E \|_A E \mid E \backslash a \mid \xi \mid \mu\xi.PG$$

Guarded process $PG \in RECG(\Sigma_1)$*:*

$$PG ::= stop \mid a \rightarrow E \mid PG \sqcap PG \mid PG \Box PG \mid PG \|_A PG \mid PG \backslash a \mid \mu\xi.PG$$

□

In distinction to untimed models, the hiding operator does not affect the guards in processes, as the guard is the duration of the action and not its name.

The hiding operator is well-behaved (on the limits) when dealing with guarded processes.

Let $\sigma \in RECG(\Sigma_1)$ with only one free variable, then $\mathcal{S}[\![\sigma]\!]$ represents a function between specifications denoted by $f_\sigma : \mathcal{S} \longrightarrow \mathcal{S}$.

Props. 3.3 *Let* $\sigma \in RECG(\Sigma_1)$ *be finite and with only one free variable. Let* $X_\sigma = \{f_\sigma^n(\perp_S)\}_{n \geq 0}$ *then* $\mathrm{fix}(f_\sigma) = \mathrm{lub}(X_\sigma)$. □

Proof : the basic idea of the proof is that no recursive guarded process can perform infinitely many simultaneous actions (in a finite time). This is formalized in the following lemma:

[6] The semantics of *chaos* is the bottom of the specification space: $\perp_S = \langle \mathcal{FT}, \mathcal{TTF} \rangle$.

Lemma 3.4 *If* $\sigma \in RECG(\Sigma_1)$ *is finite and with only one free variable, then for each* $Sp \in \mathcal{S}$, $\bigcup_{n\geq i} T(f_\sigma^n(Sp))$ *is bounded in every time instant* $i \in \mathbb{N}^+$*; i.e.*

$$\forall i \in \mathbb{N}^+ \exists k_i = \max\{|t(i)|, t \in \bigcup_{n\geq i} T(f_\sigma^n(Sp))\}$$

with each iteration of the process we bound the size of the bags at one more instant. □

All the proofs are omitted but they can be found in [OdF90,Ort90].

4 Proof System

A denotational semantics defines the meaning of each process and induces a natural equivalence between processes: P and Q are equivalent iff they have the same semantics.

If we are really interested in this equivalence relation, we could establish it, at least in the finite cases, by computing the corresponding denotational semantics and comparing them. But it is clearly preferible to count on some indirect procedure which allows us to make such decisions without leaving the syntactic level in which processes are given. Thus we are interested in having some collection of algebraic laws which allows us to establish any valid equality. This is called *a complete and correct proof system.* Correct in the sense that every derived equality must be semantically valid, and complete in the sense that every semantically valid equality must be derivable by the system.

In order to prove that a system is correct it is sufficient to prove that every axiom and rule in the system is correct. Completeness of the system is more difficult to establish and the usual way is to define some *normal form* (each process can be transformed into a unique normal form[7]) which reflects as well as possible the nature of the semantics, so that two semantically equivalent processes have the same normal form.

This part of the work has been greatly inspired by [Hen88,Bro83,TV89]. The first presents a general theory; the second includes a proof system for a failure semantics for TCSP and the third for a failure semantics with multisets (bags).

We first concentrate on finite processes, i.e. closed processes without recursive calls in their syntactic definitions. Afterwards we extend the results to deal with recursion.

4.1 Finite processes

The main difficulty encountered when looking for the axiom system was defining an appropiate *normal form.* Following the steps of Taubner & Vogler in [TV89] we introduce new and simpler operators, to be able to eliminate the more complicated operators from the normal forms. Therefore we will consider the following syntactic operator set:

$$\Sigma_x = \Sigma_1 \cup \{B \Rightarrow | B \in \mathcal{B}^+(\mathcal{A})\} \cup \{delay, delay*\}$$

[7] To be accurate, unicity of normal forms is not required if the syntactic equivalence between the different normal forms corresponding to a process can be easily established.

Def. 4.1 Extended TCSP term $P \in FIN(\Sigma_x)$*:*

$$P ::= stop \mid a \rightarrow P \mid B \Rightarrow P \mid P \sqcap P \mid P \Box P \mid P\|_A \mid P \backslash a \mid delay(P) \mid delay{*}(P)$$

$\Box$

The new operator $B \Rightarrow$ is similar to $a \rightarrow$ but considering bags instead of single actions, allowing us to eliminate the synchronizing operator. It is assumed that the duration of bag B is just one instant, thus a new operator *delay* is needed to "delay" the following process and to simulate longer durations. Process $delay(P)$ waits one instant before starting the execution of P, thus representing both the execution of internal actions and the "passing of time". Notice how *delay* and $B \Rightarrow$ reflect the temporal nature of the semantics:

- $B \Rightarrow$ represents one time unit in which B is observed,
- *delay* represents one time unit in which nothing is observed.

On the other hand B (in $B \Rightarrow$) is executed immediately, and so a new operator, which expresses the idea of no maximum parallelism, is needed: $delay{*}(P)$ may wait an arbitrary amount of time before starting the execution of P.

Now we are obliged to extend the semantic domain, because some of the new operators do not preserve properties [P1]–[P8]. The new domain is $\langle \mathcal{S}x, \leq_{\mathrm{S}} \rangle$:

$$\mathcal{S}x = \{Sp \in \mathcal{O} | Sp \text{ verifies } [\mathrm{P1}] - [\mathrm{P5}], [\mathrm{P6}']\}$$

$$[\mathrm{P6}']\ (t, n) \in T(Sp) \Rightarrow (t, n+1) \in T(Sp)$$

Props. 4.1 $\langle \mathcal{S}x, \leq_{\mathrm{S}} \rangle$ *is a C.P.O.* $\Box$

The corresponding semantic operators are:

- $\mathcal{S}x_{op}[\![Sp_1, \ldots, Sp_n]\!] = \mathcal{S}_{op}[\![Sp_1, \ldots, Sp_n]\!]$ for $op \in \Sigma_1 - \{\backslash a\}$.
- $\mathcal{S}x_{B \Rightarrow}[\![Sp]\!] = \begin{cases} \{\langle (t_\emptyset, 0), r\rangle | r \in \mathcal{PF}(\mathcal{B}^+ - \{B\})\} \cup \{\langle (t_\emptyset, m), r\rangle | m \in \mathrm{I\!N}^+ \wedge r \in \mathcal{R}\} \\ \cup \{\langle \mathrm{mov}(tf, 1) +_{\mathrm{TB}} (B, 1), r\rangle | \langle tf, r\rangle \in F(Sp)\} \\ \\ \{\mathrm{mov}(tf, 1) +_{\mathrm{TB}} (B, 1) | tf \in D(Sp)\} \end{cases}$

  If B is not immediately executed (first instant) the process "loses its chance" of executing its first step and from then on cannot do anything but let the time pass and refuse every bag. This anomalous behaviour is introduced in order to preserve property [P6'], to make process *stop* be the unit of the external choice operator, and to get the distributivity of both choice operators one with respect to the other.

- $\mathcal{S}x_{delay}[\![Sp]\!] = \begin{cases} \{\langle (t_\emptyset, 0), r\rangle | r \in \mathcal{R}\} \cup \{\langle \mathrm{mov}(tf, 1), r\rangle | \langle tf, r\rangle \in F(Sp)\} \\ \{(t_\emptyset, 0) \in D(Sp)\} \cup \{\mathrm{mov}(tf, 1) | tf \in D(Sp)\} \end{cases}$

- $\mathcal{S}x_{delay*}[\![Sp]\!] = \begin{cases} \{\langle (t_\emptyset, m), r\rangle | m \in \mathbb{N} \wedge \forall m' = 0..m, \langle (t_\emptyset, m'), r\rangle \in F(Sp)\} \\ \cup \{\langle \mathrm{mov}(tf, m), r\rangle | m \in \mathbb{N} \wedge trace(tf) \neq t_\emptyset \wedge \langle tf, r\rangle \in F(Sp)\} \\ \\ \{\mathrm{mov}(tf, m) | m \in \mathbb{N} \wedge tf \in D(Sp)\} \end{cases}$

  Notice that $delay{*}(P)$ represents a process which offers to the environment (external choice) the possibility of delaying the execution of P as long as desired. Remember also that in external choice the decision is taken only when the first externally visible action starts. The first part of the semantic definition reflects this idea.

- $\mathcal{S}x_{\backslash a}[\![Sp]\!] = \begin{cases} \{\langle tf\backslash a, r \cup \{B_1, \ldots, B_m\}\rangle | r \in \mathcal{PF}(\mathcal{B}_a) \\ \wedge \forall n_1, \ldots, n_m \in \mathbb{N}^m \langle tf, \bigcup_{i=1}^{m} \{\bigcup_{j=0}^{n_i} \{B_i +_{\mathrm{B}} j * B_a\}\}\rangle \in F(Sp)\} \\ \\ \{tf\backslash a | \ \exists tf' : \mathcal{A}(tf') \subseteq \{a\} \wedge tf{\cdot}tf' \in D(Sp) \\ \qquad \vee \exists \{t_i\}_{i \in \mathbb{N}^+} : \mathcal{A}(t_i) = \{a\} \wedge tf{\cdot}t_1 \ldots {\cdot} t_i \in T(Sp)\} \end{cases}$

  The hiding operator needs a small change due to the absence of property [P8]. The new definition coincides with the old one in the domain $\mathcal{S}$.

Props. 4.2 $\forall op \in \Sigma_x, \mathcal{S}x_{op}[\![Sp_1, \ldots, Sp_n]\!]$ *is well-defined, monotonic, and it preserves properties* [P1]–[P5],[P6']. $\square$

Props. 4.3 $\forall op \in \Sigma_x - \{\backslash a\}, \mathcal{S}x_{op}[\![Sp_1, \ldots, Sp_n]\!]$ *is continuous.* $\square$

As before, the hiding operator is not continuous but behaves well enough for guarded processes.

4.1.1 Basic terms

We consider a subset of TCSP extended terms:

Def. 4.2 Basic term $P \in TBAS$*:*

$$P ::= stop \mid B \Rightarrow P \mid P \sqcap P \mid P \square P \mid delay(P) \mid delay{*}(P)$$

$\square$

We will prove that every TCSP extended term can be transformed into a basic term and every basic term into a normal form.

4.1.2 Proof system

The logic language for the proof system consists of extended TCSP terms and a binary relation $\equiv$.[8]

From now on we consider three different *identity* symbols:

[8]In [Bro83,Hen88,TV89] inequation systems are considered. While we restrict ourselves to finite processes, the results are in fact equivalent, but since it is possible, we consider more elegant to work only with equations. However when dealing with recursive processes inequations will be unavoidable.

(A1)	$P \sqcap P \equiv P$	(A2)	$P \Box P \equiv P$
(A3)	$P \sqcap Q \equiv Q \sqcap P$	(A4)	$P \Box Q \equiv Q \Box P$
(A5)	$P \sqcap (Q \sqcap R) \equiv (P \sqcap Q) \sqcap R$	(A6)	$P \Box (Q \Box R) \equiv (P \Box Q) \Box R$
(A7)	$P \sqcap (Q \Box R) \equiv (P \sqcap Q) \Box (P \sqcap R)$	(A8)	$P \Box (Q \sqcap R) \equiv (P \Box Q) \sqcap (P \Box R)$
(A9)	$P \Box stop \equiv P$		
(A10)	$B \Rightarrow P \sqcap B \Rightarrow Q \equiv B \Rightarrow (P \sqcap Q)$	(A11)	$B \Rightarrow P \Box B \Rightarrow Q \equiv B \Rightarrow (P \sqcap Q)$
(D0)	$delay(stop) \equiv stop$	(D0$*$)	$delay{*}(stop) \equiv stop$
(D1)	$delay(P \sqcap Q) \equiv delay(P) \sqcap delay(Q)$		
(D2)	$delay(P \Box Q) \equiv delay(P) \Box delay(Q)$		
(D2$*$)	$delay{*}(P \Box Q) \equiv delay{*}(P) \Box delay{*}(Q)$		
(D3)	$delay(delay{*}(P)) \equiv delay{*}(delay(P))$		
(D4)	$delay{*}(P) \equiv P \Box delay(delay{*}(P))$		
(D5)	$delay{*}(delay{*}(P)) \equiv delay{*}(P)$		

Table 1: axiom set Δ_1

- $=_S$ represents semantic equality,
- $=$ is syntactic identity,
- $\equiv$ refers to the equivalence induced by the proof system.

By abuse of notation we identify syntactic processes with its semantics, so that we drop $\mathcal{S}x[\![\cdot]\!]$ when the context is obvious. For instance we write $P =_S Q$ for $\mathcal{S}x[\![P]\!] = \mathcal{S}x[\![Q]\!]$.

Tables 1 and 2 contain respectively the lists of axioms (Δ_1) and rules (Δ_2) of the proof system which allow for the transformation of any basic term into a normal form.

Axiom (D4) is like an "expansion" rule for the *delay$*$* operator. It expresses the fact that $delay{*}(P)$ represents a process which offers the possibility of starting P immediately or waiting one instant before choosing again.

It would be nice to have also the following formula:

$$\text{(D1}*\text{)} \quad delay{*}(P \sqcap Q) \equiv delay{*}(P) \sqcap delay{*}(Q)$$

Unfortunately this is false in general. However we will need at least some restricted form of it to complete our axiom system. Therefore we will investigate the grounds for its invalidity.

First of all we have the following lemma:

(O1)	$\dfrac{}{P \equiv P}$	(O2)	$\dfrac{P \equiv Q}{Q \equiv P}$
(O3)	$\dfrac{P \equiv Q \wedge Q \equiv R}{P \equiv R}$	(M1)	$\dfrac{P \equiv Q}{B \Rightarrow P \equiv B \Rightarrow Q}$
(M2)	$\dfrac{P_1 \equiv Q_1 \wedge P_2 \equiv Q_2}{P_1 \sqcap P_2 \equiv Q_1 \sqcap Q_2}$	(M3)	$\dfrac{P_1 \equiv Q_1 \wedge P_2 \equiv Q_2}{P_1 \Box P_2 \equiv Q_1 \Box Q_2}$
(M4)	$\dfrac{P \equiv Q}{delay(P) \equiv delay(Q)}$	(M4$*$)	$\dfrac{P \equiv Q}{delay{*}(P) \equiv delay{*}(Q)}$

Table 2: rule set Δ_2

Lemma 4.4 *Let $Sp_1, Sp_2 \in \mathcal{S}x$: $\mathcal{S}x[\![delay{*}(Sp_1 \sqcap Sp_2)]\!] \leq_{\mathrm{S}} \mathcal{S}x[\![delay{*}(Sp_1) \sqcap delay{*}(Sp_2)]\!]$.* $\Box$

Moreover it is easy to check that the falsehood of (D1$*$) comes only from failures with empty traces. The necessary conditions to make it true are not trivial and some previous concepts and definitions are needed. These will also be useful for the definition of normal forms.

4.1.3 Derived operators

These derived operators are merely syntactic abbreviations. First we give two families of delay operators:

Def. 4.3 General delay operators *$delay(\ell)$ and $delay{*}(\ell)$ with $\ell \in \mathbb{N}$:*

$$\begin{aligned} delay(0, P) &= P \\ delay(\ell+1, P) &= delay(\ell, delay(P)) \\ delay{*}(\ell, P) &= delay(\ell, delay{*}(P)) \end{aligned}$$

Notice that $delay(P) = delay(1, P)$ and $delay{*}(P) = delay{*}(0, P)$. $\Box$

Thanks to axioms (A2), (A4), (A6) and (A9) we can define $\Box_{i \in I} P_i$, where I is any finite index set and $P_i \in \mathcal{S}x$ for $i \in I$:

$$\begin{aligned} \Box_{i \in \emptyset} P_i &= stop \\ \Box_{i \in j \cup I} P_i &= P_j \Box (\Box_{i \in I} P_i) \end{aligned}$$

Analogously we define $\mathrm{OR}_{i\in I}P_i$, but now I cannot be the empty set:

$$\begin{aligned}\mathrm{OR}_{i\in\{j\}}P_i &= P_j\\ \mathrm{OR}_{i\in j\cup I}P_i &= P_j \sqcap (\mathrm{OR}_{i\in I}P_i) \text{ if } I \neq \emptyset.\end{aligned}$$

4.1.4 Prenormal form

As candidates for our normal form we are looking for syntactic formulae which express more clearly the internal structure of the processes denoted by them. We need to know what is happening at each instant.

Def. 4.4 *Action bags are associated to time instants and basic terms:*

timed bags : $\mathcal{BT} = \mathcal{B}(\mathcal{A}) \times \mathbb{N}$,

sets of timed bags : $\mathcal{CBT} = \mathcal{PF}(\mathcal{BT})$,

bag-process pairs: $\mathcal{BP} = \mathcal{B}(\mathcal{A}) \times TBAS$,

sets of bag-process pairs : $\mathcal{CBP} = \mathcal{PF}(\mathcal{BP})$.

Let $BT \in \mathcal{CBT}$, the set of bags in BT is denoted by $\mathcal{B}(BT)$. Analogously for $BP \in \mathcal{CBP}$.
□

Timed bag sets are considered by pairs; the first component refers to *delay* operators and the second to *delay*$*$.

Def. 4.5 *Let $C = (C_1, C_2) \in \mathcal{CBT} \times \mathcal{CBT}$; we define:*

timed bags in C**:** $\mathcal{BT}(C) = C_1 \cup C_2$,

top time : $top(C) = \begin{cases} -1 & \text{if } C = (\emptyset, \emptyset) \\ \max\{t \mid (B,t) \in C_1 \vee (B, t+1) \in C_2\} & \text{otherwise}\end{cases}$

projection : $\Gamma_j(C) = C_j,\ j = 1, 2.$

□

As we will see, the top time spots the limit of changes in the activity of a process. From the top time on, the process will maintain the same options until the first non-empty action bag will be executed.

The above definitions can be generalized to pair sets:

Def. 4.6 *Let $\mathrm{C} = \{C_i\}_{i=1..n} \in \mathcal{PF}(\mathcal{CBT} \times \mathcal{CBT})$; we define:*

timed bags in C**:** $\mathcal{BT}(\mathrm{C}) = \bigcup_{i=1..n} \mathcal{BT}(C_i)$,

top time : $top(\mathrm{C}) = \max\{top(C_i)\}_{i=1..n}$,

projection : $\Gamma_j(\mathrm{C}) = \bigcup_{1 \le i \le n} \Gamma_j(C_i),\ j = 1, 2.$

$\square$

A first step towards the desired normal forms are prenormal forms :

Def. 4.7 *A term is in* **prenormal form** *if it is of form :*

$$\mathrm{OR}_{(C_1,C_2)\in\mathrm{C}} \left[\Box_{(B,t)\in C_1} delay(t, B \Rightarrow P_{(B,t)}) \Box \Box_{(B,t)\in C_2} delay{*}(t, B \Rightarrow P_{(B,t)})\right]$$

where $\mathrm{C} \in \mathcal{PF}(\mathcal{CBT} \times \mathcal{CBT})$ *and each* $P_{(B,t)}$ *is in prenormal form.*

A prenormal form is said **unitary**, *when* C *contains a unique pair of timed bag sets.*
$\square$

Notice that *stop* is in prenormal form.

Subprocesses $P_{(B,t)}$ depend only on B and t, but not on (C_1, C_2). Therefore a prenormal form is completely defined by the pair $\langle \mathrm{C}, fp\rangle$, where fp is a partial function which associates a basic term to each timed bag:

$$fp(B,t) = \begin{cases} P_{(B,t)} & \text{if } (B,t) \in \mathcal{BT}(\mathrm{C}) \\ \perp & \text{otherwise} \end{cases}$$

4.1.5 Expansion and distribution

Prenormal forms are too general to work with it. To a same process correspond too many different prenormal forms. Moreover the combination of prenormal forms with the internal choice operator, not always can be transformed to another prenormal form. Therefore we will define the concepts of expansion and distribution.

Def. 4.8 *Let* $\langle C, fp\rangle$ *be a unitary prenormal form with* $C = \{(C_1, C_2)\}$ *and* $top(C) = K$*; for every* $m \ge K$ *a set of* $m+2$*-tuples* $\mathrm{EX}(\langle C, fp\rangle, m)$ *is defined which is called* **expansion of** $\langle C, fp\rangle$ **until instant m***:*

$$\mathrm{EX}(\langle C, fp\rangle, m) = \langle D(0), \ldots, D(m+1)\rangle \in \mathcal{PF}(\mathcal{CBP} \times \ldots \times \mathcal{CBP})\ (m+1 \text{ times})$$

where $D(j) = \{(B, fp(B,j)) | (B,j) \in C_1\} \cup \{(B, fp(B,t)) | t \le j \wedge (B,t) \in C_2\}$.

When $m = K$ *we write* $\mathrm{EX}(\langle C, fp\rangle)$. $\square$

Expanding until instant m means developing the $delay{*}(\ell)$ operators on $delay(j)$ operators for every $j : \ell \le j \le m$ plus $delay{*}(m+1)$. So that C is separated in temporal groups $D(t), t = 0..m+1$.

Def. 4.9 *Let* $\langle \mathrm{C}, fp\rangle$ *be a prenormal form with* $K = top(\mathrm{C})$*; two families of tuple sets are defined:*

Expansion : $\forall m \ge K, \mathrm{EX}(\langle \mathrm{C}, fp\rangle, m) = \{\mathrm{EX}(\langle C, fp\rangle, m)\}_{C\in\mathrm{C}}$.

When $m = K$ *we write* $\mathrm{EX}(\langle \mathrm{C}, fp\rangle)$.

Distribution : $\forall m = 0..K{+}1, \mathrm{DSTR}(\langle \mathrm{C}, fp\rangle, m) \in \mathcal{PF}(\mathcal{CBP}\times\ldots\times\mathcal{CBP})$ $(K{+}2$ times$)$,

$$\mathrm{DSTR}(\langle \mathrm{C}, fp\rangle, m) = \{D_C(0)\}_{C\in \mathrm{C}}\times\ldots\times\{D_C(m{-}1)\}_{C\in \mathrm{C}}\times\{\langle D_C(m),\ldots,D_C(K{+}1)\rangle\}_{C\in \mathrm{C}}$$

where $\mathrm{EX}(\langle C, fp\rangle) = \langle D_C(0),\ldots,D_C(K+1)\rangle$ *for each* $C \in \mathrm{C}$.
When $m = K+1$ *we write* $\mathrm{DSTR}(\langle \mathrm{C}, fp\rangle)$.

Notice that $\mathrm{DSTR}(\langle \mathrm{C}, fp\rangle, 0) = \mathrm{EX}(\langle \mathrm{C}, fp\rangle)$.

□

Distributing until instant m means obtaining every combination of temporal groups until instant m. If $top(\mathrm{C}) = K$ we build tuples $\langle D(0),\ldots,D(K{+}1)\rangle$ where each $D(i), 0 \leq i < m$ belongs to some element of C, maybe different each time, but every $D(i), m \leq i \leq K+1$ belongs to the same element.

Def. 4.10 *A prenormal form* $\langle \mathrm{C}, fp\rangle$ *is said to be*

expanded until instant m $(m \geq 0)$ *if* $top(\mathrm{C}) = m$ *and* $(B,t) \in \Gamma_2(\mathrm{C}) \Rightarrow t = m+1$.
When $m = top(\mathrm{C})$ *we say* **expanded**.

distributed until instant m $(0 \leq m \leq top(\mathrm{C}){+}1)$ *if* $\mathrm{DSTR}(\langle \mathrm{C}, fp\rangle, m) = \mathrm{EX}(\langle \mathrm{C}, fp\rangle)$.
When $m = top(\mathrm{C})+1$ *we say* **distributed**.

□

We can give now the axiom corresponding to the distribution of the *delay∗* operator with the internal choice and which should be included in the list Δ_1 of table 1. We also include in Δ_1 the *distribution law*, an axiom family which allows for the transformation of any prenormal form into a distributed one. Both axioms appear on table 3.

From now on we write $\vdash P \equiv Q$, when the formula $P \equiv Q$ is provable in $\Delta_1 \cup \Delta_2$.

Theorem 4.5 *For every* $P, Q, R, \ldots$ *TCSP extended terms, the axioms and rules in* $\Delta_1 \cup \Delta_2$ *are correct:* $\vdash P \equiv Q \Rightarrow P =_S Q$. □

4.1.6 Normal form

A normal form is a prenormal form verifying some conditions.

Def. 4.11 C *is* **convex** *if it is non-empty and verifies:*

1. $A, B \in \mathrm{C} \Rightarrow A \cup B \in \mathrm{C}$,
2. $A, D \in \mathrm{C} \wedge A \subseteq B \subseteq D \Rightarrow B \in \mathrm{C}$.

□

if $\langle \mathrm{C}, fp \rangle$ is a distributed prenormal form.

(D1*) $\quad delay{*}(\langle \mathrm{C}, fp \rangle) \equiv \mathrm{OR}_{C \in \mathrm{C}} delay{*}(\langle C, fp \rangle)$

Let $R = (P_1 \Box delay(k, Q_1)) \sqcap (P_2 \Box delay(k, Q_2))$
with $P_i = \Box_{0 \leq j < k} \Box_{(B,P) \in c_{ij}} delay(j, B \Rightarrow P)$

(D6) $\quad R \equiv R \sqcap (P_1 \Box delay(k, Q_2)) \sqcap (P_2 \Box delay(k, Q_1))$

Table 3: distribution axiom and law.

Def. 4.12 *A term is in* **normal form** *if it is in distributed prenormal form* $\langle \mathrm{C}, fp \rangle$ *with* C *convex and every* $fp(B, t)$ *in normal form.* $\Box$

We will give now some lemmas and propositions to transform any prenormal form into a normal one. First lemma states the *convexity laws*, already presented in [Hen88]:

Lemma 4.6 *The following formulas are provable in* $\Delta_1 \cup \Delta_2$*:*

$$\text{(DV1)} \quad P \sqcap Q \equiv P \sqcap Q \sqcap (P \Box Q)$$

$$\text{(DV2)} \quad P \sqcap (P \Box Q \Box R) \equiv P \sqcap (P \Box Q) \sqcap (P \Box Q \Box R)$$

$\Box$

Next lemma states that in any prenormal form, C can be replaced by its convex closure ($\mathrm{CONV}(\mathrm{C})$).

Lemma 4.7 $\vdash \langle \mathrm{C}, fp \rangle \equiv \langle \mathrm{CONV}(\mathrm{C}), fp \rangle$.
Proof : We apply the convexity laws (lemma 4.6). $\Box$

Lemma 4.8 *Let* $P = P_1 \sqcap P_2$*, the following formulas are provable in* $\Delta_1 \cup \Delta_2$*:*

$$\text{(DV3)} \quad (delay(\ell, B \Rightarrow P_1) \Box Q_1) \sqcap (delay(\ell, B \Rightarrow P_2) \Box Q_2) \equiv (delay(\ell, B \Rightarrow P) \Box Q_1) \sqcap (delay(\ell, B \Rightarrow P) \Box Q_2)$$

$$\text{(DV3*)} \quad (delay{*}(\ell, B \Rightarrow P_1) \Box Q_1) \sqcap (delay{*}(\ell, B \Rightarrow P_2) \Box Q_2) \equiv (delay{*}(\ell, B \Rightarrow P) \Box Q_1) \sqcap (delay{*}(\ell, B \Rightarrow P) \Box Q_2)$$

The above rules are called *expansion laws* and are used to transform any prenormal form into an expanded one. $\Box$

Lemma 4.9 *Let* $\langle \mathrm{C}, fp \rangle$ *be a prenormal form,*

1. $\forall m \geq top(\mathrm{C}), \exists\langle \mathrm{C}', fp'\rangle$ *expanded until instant* m *so that* $\vdash \langle \mathrm{C}, fp\rangle \equiv \langle \mathrm{C}', fp'\rangle$,

2. $\forall m = 0..top(\mathrm{C})+1, \exists\langle \mathrm{C}', fp'\rangle$ *distributed until instant* m *and expanded, which verifies* $top(\langle \mathrm{C}', fp'\rangle = top(\langle \mathrm{C}, fp\rangle)$ *so that* $\vdash \langle \mathrm{C}, fp\rangle \equiv \langle \mathrm{C}', fp'\rangle$.

□

Finally we will prove that the convexity closure preserves distribution.

Lemma 4.10 *If* $\langle \mathrm{C}, fp\rangle$ *is a distributed prenormal form, then* $\langle \mathrm{CONV}(\mathrm{C}), fp\rangle$ *is distributed too.* □

Corollary 4.11 *Let* $\langle \mathrm{C}, fp\rangle$ *be a prenormal form, there exists* $\langle \mathrm{C}', fp'\rangle$ *normal form so that* $\vdash \langle \mathrm{C}, fp\rangle \equiv \langle \mathrm{C}', fp'\rangle$.
Proof : We apply lemmas 4.9, 4.7 and 4.10. □

Props. 4.12 *Let* P *be a basic term, there exists a normal form* $\langle \mathrm{C}, fp\rangle$ *so that* $\vdash P \equiv \langle \mathrm{C}, fp\rangle$. □

4.1.7 Proof system completeness

The next proposition states that two semantically equivalent normal forms are syntactically equivalent too.[9]

Props. 4.13 *Let* $\langle \mathrm{C}, fp\rangle, \langle \mathrm{C}', fp'\rangle$ *be normal forms, then*
$\langle \mathrm{C}, fp\rangle =_S \langle \mathrm{C}', fp'\rangle \Rightarrow \vdash \langle \mathrm{C}, fp\rangle \equiv \langle \mathrm{C}', fp'\rangle$. □

Theorem 4.14 *Let* P, Q *be basic terms:* $P =_S Q \Rightarrow \vdash P \equiv Q$.
Proof : It is a corollary of propositions 4.12 and 4.13. □

4.1.8 Transformation of finite TCSP terms into basic terms

Table 4 contains the axiom list Δ_3, which added to $\Delta_1 \cup \Delta_2$ allows for the transformation of any finite TCSP term into a basic one. We obtain then a proof system for finite TCSP processes, which is correct and complete with respect to the denotational semantics given in section]3.

The more complex axioms are a consequence of the non-distributivity of the synchronization and hiding operators with the external choice.

Theorem 4.15 *For any extended TCSP term* $P, Q, R, \ldots$ *and* $C, D \in \mathcal{R}$ *the axioms in* Δ_3 *are correct.* □

Props. 4.16 *Any extended TCSP term can be transformed, by means of the axioms and rules in* Δ_1, Δ_2 *and* Δ_3 *into a basic term.* □

Now if we consider that $\vdash P \equiv Q$ means that the formula $P \equiv Q$ is provable in $\Delta_1 \cup \Delta_2 \cup \Delta_3$, we shall give the last theorem of this part devoted to finite processes:

Theorem 4.17 *Let* P, Q *be finite TCSP terms,* $P =_S Q \Leftrightarrow \vdash P \equiv Q$.
Proof : It is a corollary of proposition 4.16 and theorems 4.5, 4.14 and 4.15. □

[9]We cannot assert that they are syntactically identical, due to the fact that our normal forms are not unique. We could have defined some order over the set of normal forms for a process, obtaining a unique least normal form, but this is unnecessary.

(X) $a \to P \equiv delay{*}(0, B_a \Rightarrow delay(d(a)-1, P))$

(S0) $stop\|_A stop \equiv stop$

(S1) $P\|_A Q \equiv Q\|_A P$

(S2) $P\|_A(Q \sqcap R) \equiv (P\|_A Q)\sqcap(P\|_A R)$

(S3) $delay(P)\|_A delay(Q) \equiv delay(P\|_A Q)$

(S4) Let $P = \Box_{B\in C} B \Rightarrow P_B \,\Box\, delay(P')$ and $Q = \Box_{E\in D} E \Rightarrow Q_E \,\Box\, delay(Q')$

$$\begin{array}{lll} P\|_A Q \equiv & & \Box_{B\lceil A = E\lceil A} B \oplus_A E \Rightarrow (P_B\|_A Q_E) \\ & \Box & \\ & & \Box_{B\lceil A = B_\emptyset} B \Rightarrow (P_B\|_A Q') \\ & \Box & \\ & & \Box_{E\lceil A = B_\emptyset} E \Rightarrow (P'\|_A Q_E) \\ & \Box & \\ & & delay(P'\|_A Q') \end{array}$$

(S5) Let $P = delay{*}(\Box_{B\in C} B \Rightarrow P_B)$ and $Q = delay{*}(\Box_{E\in D} E \Rightarrow Q_E)$, then

$$\begin{array}{lll} P\|_A Q \equiv delay{*}(& & \Box_{B\lceil A = E\lceil A} B \oplus_A E \Rightarrow (P_B\|_A Q_E) \\ & \Box & \\ & & \Box_{B\lceil A = B_\emptyset} B \Rightarrow (P_B\|_A Q) \\ & \Box & \\ & & \Box_{E\lceil A = B_\emptyset} E \Rightarrow (P\|_A Q_E)) \end{array}$$

(H0) $stop\backslash a \equiv stop$

(H1) $(P\sqcap Q)\backslash a \equiv P\backslash a \sqcap Q\backslash a$

(H2) $delay(P)\backslash a \equiv delay(P\backslash a)$

(H3) Let $C \cap \mathcal{B}_a = \emptyset$, then

$$\left[\Box_{B\in C} B \Rightarrow P_B \,\Box\, delay(P)\right]\backslash a \equiv \Box_{B\in C} B\backslash a \Rightarrow P_B\backslash a \,\Box\, delay(P)\backslash a$$

(H4) Let $C \cap \mathcal{B}_a \neq \emptyset$, then

$$\left[\Box_{B\in C} B \Rightarrow P_B \,\Box\, delay(P)\right]\backslash a \equiv \Box_{B\in C-\mathcal{B}_a} B\backslash a \Rightarrow P_B\backslash a \,\Box\, delay(P \sqcap \mathrm{OR}_{B\in C\cap\mathcal{B}_a} P_B)\backslash a$$

(H5) Let $C \cap \mathcal{B}_a = \emptyset$, then

$$delay{*}(\Box_{B\in C} B \Rightarrow P_B)\backslash a \equiv delay{*}(\Box_{B\in C} B\backslash a \Rightarrow P_B\backslash a)$$

(H6) Let $C \cap \mathcal{B}_a \neq \emptyset$, then

$$\begin{array}{ll} delay{*}(\Box_{B\in C} B \Rightarrow P_B)\backslash a \equiv & \Box_{B\in C-\mathcal{B}_a} B\backslash a \Rightarrow P_B\backslash a \,\Box \\ & delay(delay{*}(\Box_{B\in C-\mathcal{B}_a} B\backslash a \Rightarrow P_B\backslash a \sqcap \mathrm{OR}_{B\in C\cap\mathcal{B}_a} P_B\backslash a)) \end{array}$$

Table 4: transformation axiom set Δ_3

4.2 Recursive processes

In order to deal with recursive processes we need their *syntactic approximations* corresponding to the semantic approximations deduced from the fix-point theory.

So as to work with approximations we need to introduce some changes in the proof system given for finite processes. On the one hand it is no longer sufficient to deal with equivalences. Some partial order between processes is needed. Consequentely a new symbol is introduced: $\sqsubseteq$, from which the syntactic equivalence $\equiv$ is deduced. The other essential element is the syntactic equivalent of the least element of the semantic domain: *chaos*.

4.2.1 Superextended TCSP

We shall introduce the constant operator *chaos*. Thus the new set of syntactic operators is:

$$\Sigma_X = \Sigma_x \cup \{chaos\}$$

We can maintain the same specification space $\langle \mathcal{S}x, \leq_{\mathrm{S}} \rangle$. The semantics of *chaos* is:

$$\mathcal{S}x_{chaos}[\![\cdot]\!] = \langle \mathcal{FT}, \mathcal{TTF} \rangle$$

Obviously it is well-defined, it verifies [P1]–[P5] and [P6'], and it is continuous.

4.2.2 Syntactic approximations

The semantics of a recursive process P is the limit of the sequence generated by unwinding the recursive calls, and starting with *chaos*:

$$\begin{aligned} P^o &= chaos \\ P^{n+1} &= P[P^n] \end{aligned}$$

At a syntactic level each P^n is a superextended TCSP term ($P^n \in FIN(\Sigma_X)$).

Def. 4.13 *The set of* **finite approximations** *for process P is* $\mathrm{APX}(P) = \{P^n | n \in \mathbb{N}\}$. □

4.2.3 New axioms and rules

We must include, on the one hand, the axioms concerning the behaviour of *chaos* with respect to the other operators (Ω_1); on the other hand, the rules refering to recursion (Ω_2); and finally the rules justifying that $\sqsubseteq$ is a partial order (Ω_3). The tables 5, 6 and 7 contain these sets respectively. Notice that the rules in Δ_2 are obtained from the rules in Ω_2.

We write $\vdash P \sqsubseteq Q$ when the formula $P \sqsubseteq Q$ is provable in $\Delta_1 \cup \Omega_1 \cup \Omega_2 \cup \Omega_3$.

(A12)	$P \sqcap Q \sqsubseteq P$
(CH0)	$chaos \sqsubseteq P$
(CH1)	$P \sqcap chaos \equiv chaos$
(CH2)	$P \Box chaos \equiv chaos$
(CH3)	$delay{*}(chaos) \equiv chaos$
(CH4)	$chaos \|_A P \equiv chaos$
(CH5)	$chaos \backslash a \equiv chaos$

Table 5: axiom set Ω_1

(O1')	$\dfrac{P \sqsubseteq Q \sqsubseteq P}{P \equiv Q}$	(O2')	$\dfrac{P \equiv Q}{P \sqsubseteq Q \sqsubseteq P}$
(O3')	$\dfrac{P \sqsubseteq Q \sqsubseteq R}{P \sqsubseteq R}$	(M1')	$\dfrac{P \sqsubseteq Q}{B \Rightarrow P \sqsubseteq B \Rightarrow Q}$
(M2')	$\dfrac{P_1 \sqsubseteq Q_1 \wedge P_2 \sqsubseteq Q_2}{P_1 \sqcap P_2 \sqsubseteq Q_1 \sqcap Q_2}$	(M3')	$\dfrac{P_1 \sqsubseteq Q_1 \wedge P_2 \sqsubseteq Q_2}{P_1 \Box P_2 \sqsubseteq Q_1 \Box Q_2}$
(M4')	$\dfrac{P \sqsubseteq Q}{delay(P) \sqsubseteq delay(Q)}$	(M4*')	$\dfrac{P \sqsubseteq Q}{delay{*}(P) \sqsubseteq delay{*}(Q)}$

Table 6: rule set Ω_2

(R1)	$P[\mu\xi.P/\xi] \sqsubseteq \mu\xi.P$
(R2)	$\dfrac{\forall Q \in \mathrm{APX}(P) : Q \sqsubseteq R}{P \sqsubseteq R}$

Table 7: recursion rules Ω_3

4.2.4 Correctness of the new proof system

First of all we must confess that the new system is not completely correct, as axioms (CH2), (CH4) and (CH5) are not correct. However "don't panic". As we are only interested in dealing with *chaos* in relation to the finite approximations for recursive guarded processes, the inclusion of these "incorrect" axioms is justified by the fact that $P\Box chaos$; $P\|_A chaos$ and $chaos\backslash a$ have such a "chaotic" behaviour that it is convenient to identify them with *chaos* when they appear in the approximations.

We shall formalize the concept of *partial correctness* and we will see that by means of this concept we can prove the total correctness of the system restricted to guarded TCSP processes.

First we prove that the rest of axioms and rules are correct.

Lemma 4.18 *Let $P, Q, R, \ldots$ be extended and guarded TCSP terms; then the axioms in $\Omega_2 \cup \Omega_3 \cup \{(CH0), (CH1), (CH4)\}$ are correct.* $\Box$

A family of semantic orders, based on the length of the observed traces, is introduced.

Def. 4.14 *Let $Sp \in \mathcal{S}x$ and $n \in \mathbb{N}$, we define:*

$$\begin{array}{lcl} F_n(Sp) & = & \{\langle tf, \emptyset\rangle \in F(Sp) | end(tf) = n\} \cup \{\langle tf, r\rangle \in F(Sp) | end(tf) < n\}, \\ D_n(Sp) & = & \{tf \in D(Sp) | end(tf) \leq n\}, \\ OB_n(Sp) & = & \langle F_n(Sp), D_n(Sp)\rangle. \end{array}$$

$\Box$

Def. 4.15 *Let $Sp_1, Sp_2 \in \mathcal{S}x$ and $n \in \mathbb{N}$: $Sp_1 \leq_n Sp_2 \Leftrightarrow OB_n(Sp_2) \subseteq OB_n(Sp_1)$.* $\Box$

We shall write $P \leq_n Q$ when $\mathcal{S}x[\![P]\!] \leq_n \mathcal{S}x[\![Q]\!]$. These orders are only partial orders which induce the corresponding equivalences $=_n$ between processes.

The relation between these new orders and the partial order $\leq_{\mathrm{S}}$ is given by the following lemma:

Lemma 4.19 *Let $Sp_1, Sp_2 \in \mathcal{S}x$, then $Sp_1 \leq_{\mathrm{S}} Sp_2 \Leftrightarrow \forall n \in \mathbb{N} : Sp_1 \leq_n Sp_2$.* $\Box$

Lemma 4.20 *Let $P \in CRECG(\Sigma_1)$ then $\forall n \in \mathbb{N} : P =_n P^n$.* $\Box$

Lemma 4.21 (Partial correctness) *Let $P, Q \in CRECG(\Sigma_1)$: $P^n \sqsubseteq Q \Rightarrow P^n \leq_n Q$.*
Proof : As P is guarded it is easily proven that each ocurrence of *chaos* in P^n is guarded n times. The application of any axiom or rule in our present proof system preserves this property. The application of correct axioms preserves $\leq_{\mathrm{S}}$ and therefore each $\leq_n$. While the application of "incorrect" axioms, as it is done "under" n guards, preserves $\leq_n$ too. $\Box$

Now we can prove the correctness with respect to guarded processes.

Props. 4.22 *Let P, Q, then $\vdash P \sqsubseteq Q \Rightarrow P \leq_{\mathrm{S}} Q$.*
Proof : For every $P^n \in \mathrm{APX}(P)$ it is proved that $P^n \sqsubseteq P \sqsubseteq Q$. The partial correctness lemma 4.21 is applied and we have that $P^n \leq_n Q$. Therefore by lemma 4.20 we obtain that $\forall n \in \mathbb{N} : P \leq_n Q$. We conclude by lemma 4.19. $\Box$

4.2.5 Completeness of the new system

We shall start with the completeness with respect to finite terms. We only need to "extend" all the results obtained for extended TCSP terms, introducing *chaos* in the normal forms.

Def. 4.16 Superbasic term $P \in STBAS$:

$$P ::= chaos \mid stop \mid B \Rightarrow P \mid P \sqcap P \mid P \square P \mid delay(P) \mid delay{*}(P)$$

□

Def. 4.17 *A term is in* **superprenormal form** *if it is of any of the following forms:*

- $\langle \mathrm{C}, fp\rangle$ *prenormal form with each* $fp(B,t)$ *in superprenormal form,*
- $delay(t, chaos) \square \langle \mathrm{C}, fp\rangle$ *where* $\langle \mathrm{C}, fp\rangle$ *is a prenormal form with each* $fp(B,t)$ *in superprenormal form and* $\Gamma_2(\mathrm{C}) = \emptyset$*, and* $top(\mathrm{C}) < t$.

□

Last conditions reflect the special nature of *chaos*: when it starts executing, then everything else does not matter anymore. Next lemma states that these conditions can always be obtained:

Lemma 4.23 *Let P be a superextended TCSP term; in the new proof system the following formulas are true:*

1. *if* $t \leq t'$*, then* $delay(t, chaos) \square delay(t', P) \equiv delay(t, chaos)$*,*
2. $delay(t, chaos) \square delay{*}(t', P) \equiv delay(t, chaos) \square \Box_{t' \leq \ell < t} delay(\ell, P)$.

□

Every normal form is a superprenormal form too. Superprenormal forms are determined by the instant t associated to the chaotic component ($delay(t, chaos)$), if it exists, and by the set C and the function fp associated to the ordinary component. We use the following notation: $\langle t, \mathrm{C}, fp\rangle$. When a superprenomal form does not have chaotic component on the first level, we take $t = \infty$.

Notice that the above defined concepts of expansion and distributivity do not depend on the internal structure of the $fp(B,t)$, but only on the "surface" of the prenormal form. Therefore both concepts and the corresponding results (like any prenormal form can be expanded, etc), can be directly extended to superprenormal forms without chaotic component ($\langle \infty, \mathrm{C}, fp\rangle$) and from these to the general ones, as any superprenormal form can be written:

$$\langle t, \mathrm{C}, fp\rangle \equiv delay(t, chaos) \square \langle \infty, \mathrm{C}, fp\rangle$$

Def. 4.18 *A term is in* **supernormal form** *if it is in distributed superprenormal form* $\langle t, \mathrm{C}, fp\rangle$ *with* C *convex and each* $fp(B,t)$ *in supernormal form.* □

Props. 4.24 *Let P be a superbasic term, then there exists a supernormal form $\langle t, \mathrm{C}, fp\rangle$ so that $\vdash P \equiv \langle t, \mathrm{C}, fp\rangle$.* $\square$

Props. 4.25 *Let $\langle t, \mathrm{C}, fp\rangle, \langle t', \mathrm{C}', fp'\rangle$ be supernormal forms, then*

$$\langle t, \mathrm{C}, fp\rangle \leq_{\mathrm{S}} \langle t', \mathrm{C}', fp'\rangle \Rightarrow \vdash \langle t, \mathrm{C}, fp\rangle \sqsubseteq \langle t', \mathrm{C}', fp'\rangle$$

$\square$

Theorem 4.26 *Let P, Q be superbasic terms, $P =_S Q \Rightarrow \vdash P \equiv Q$.*
Proof : It is a corollary of propositions 4.24 and 4.25. $\square$

Props. 4.27 *Any superextended and finite TCSP term can be transformed into a superbasic term.* $\square$

4.2.6 Semantic aproximations

In order to prove the completeness with respect to recursive (guarded) processes, we will use the concept of syntactic aproximation (APX(P)). Now some way of "measuring" the degree of semantic aproximation for these syntactic aproximations is needed. To this purpose we define a second family of semantic orders, which depend on the number of instants in which visible actions are executed[10]. First of all we count the number of non-empty *steps* in a trace:

Def. 4.19 *Let $tf \in \mathcal{TTF}$, the number of* **non-empty steps** *in tf is:*

$$st((t_\emptyset, 0)) = 0,$$
$$st(tf +_{\mathrm{TB}} (B, end(tf)+1)) = \begin{cases} st(tf) & \text{if } \; B = B_\emptyset \\ st(tf)+1 & \text{if } \; B \neq B_\emptyset \end{cases}$$

$\square$

We need observation sets with restrictions on the number of steps for the contained traces.

Def. 4.20 *Let $Sp \in \mathcal{S}x$ and $n \in \mathbb{N}$, we define:*

$$\begin{array}{lcl} F^n(Sp) & = & \{\langle tf, \emptyset\rangle \in F(Sp) | st(tf) = n \wedge trace(tf)(end(tf)) \neq B_\emptyset\} \\ & & \cup\{\langle tf, r\rangle \in F(Sp) | st(tf) < n\}, \\ D^n(Sp) & = & \{tf \in D(Sp) | st(tf) \leq n \wedge trace(tf)(end(tf)) \neq B_\emptyset\}, \\ OB^n(Sp) & = & \langle F^n(Sp), D^n(Sp)\rangle. \end{array}$$

$\square$

The corresponding semantic orders are given:

Def. 4.21 *Let $Sp_1, Sp_2 \in \mathcal{S}x$ and $n \in \mathbb{N}$: $Sp_1 \leq^n Sp_2 \Leftrightarrow OB^n(Sp_2) \subseteq OB^n(Sp_1)$.* $\square$

[10] The above defined orders refering to the trace length are insufficient, due to the relation between hidden actions and the delay operators.

Lemma 4.28 *Let $Sp_1, Sp_2 \in \mathcal{S}x$, then $Sp_1 \leq_{\rm S} Sp_2 \Leftrightarrow \forall n \in \mathbb{N} : Sp_1 \leq^n Sp_2$.* $\Box$

We shall write $P \leq^n Q$ when $\mathcal{S}x[\![P]\!] \leq^n \mathcal{S}x[\![Q]\!]$.

Lemma 4.29 *Let P be a finite superextended TCSP term, and Q a guarded TCSP term:*

$$P \leq_{\rm S} Q \Rightarrow \exists n \in \mathbb{N} : P \leq_{\rm S} Q^n$$

Proof : It is proved first that $\forall n \in \mathbb{N}\, \exists m_n : P \leq^n Q^{m_n}$. As P is syntactically finite, there is some instant when every computation of P reaches *stop* or *chaos*. In both cases, from that instant on, $OB^n(P)$ is constant, and by lemma 4.28 we conclude the proof. $\Box$

Theorem 4.30 (Completeness and correctness for guarded TCSP processes) . *Let $P, Q \in CRECG(\Sigma_1)$, then $P \leq_{\rm S} Q \Leftrightarrow \vdash P \sqsubseteq Q$*
Proof :

1. Correctness ($\Leftarrow$) is proven by proposition 4.22.

2. In order to prove the completeness ($\Rightarrow$), take $P' \in \mathrm{APX}(P)$ then $P' \leq_{\rm S} P \leq_{\rm S} Q$; by lemma 4.29 there exists $Q' \in \mathrm{APX}(Q)$ so that $P' \leq_{\rm S} Q'$; and by propos.4.26, we have that $\vdash P' \sqsubseteq Q'$. Therefore for any $Q' \in \mathrm{APX}(Q)$ it is true that $P' \sqsubseteq Q' \sqsubseteq Q$ and by rule (R2) we have that $\vdash P \sqsubseteq Q$.

$\Box$

All omitted proofs can be found in [Ort90].

5 Conclusion and Future Work

We have presented a timed model suitable for real-time systems, where time is explicitely expressed by providing a global clock, so that we can see what happens at each instant and observe how the processes evolve in time.

The model induces denotational failure semantics with divergences for TCSP processes, for which we have provided a correct and complete proof system. This axiomatic system provides an interesting insight on the nature and properties of the new model.

As pointed out in [OdF90,Ort90], the model studied in this paper provides a too liberal treatment of time, inadequate to specify real time processes with time-outs and/or maximum parallelism requirements. Nevertheless we already introduced there (in the above cited works) two extensions of the model to cope with these features, and although more sophisticated (specially the second one) they are based on similar ideas to the original model. Therefore several of the formal problems related to the future axiomatisation of these modified versions are already solved by the results obtained in the present work. Presently we are working on the remainder details.

6 Acknowledgements

We thank Prof. K. Indermark (Lehrstuhl für Informatik II / RWTH-Aachen) for his support and kind hospitality. Thanks also to Paz for her help.

References

[Azc89] A. Azcorra. *Modelado Formal de Sistemas Síncronos.* PhD thesis, ETSI Telecomunicación, Univ.Politécnica de Madrid, 1989.

[BB90] J.C.M. Baeten and J.A. Bergstra. *Real Time Process Algebra.* Technical Report, CWI/ Amsterdam, 1990.

[BC88] G. Boudol and I. Castellani. Concurrency and atomicity. *TCS*, (59):25–84, 1988.

[BR85] S.D. Brookes and A.W. Roscoe. An improved failures model for communicating processes. In *Pittsburgh Seminar on Concurrency*, Springer Verlag, 1985.

[Bro83] S.D. Brookes. *A Model for Communicating Sequential Processes.* PhD thesis, Oxford Univ., 1983.

[GB87] R. Gerth and A. Boucher. A timed failures model for extended communicating processes. In *ICALP 87*, Springer Verlag, 1987.

[Hen88] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[HR90] M. Hennessy and T. Regan. *A Temporal Process Algebra.* Technical Report, University of Sussex (UK), 1990.

[KSdR*85] R. Koymans, R.K. Shyamasundar, W.P. de Roever, R. Gerth, and S. Arun-Kumar. Compositional semantics for real-time distributed computing. In *Conference on Logics of Programs*, Springer Verlag, 1985.

[Mil80] R. Milner. *A Calculus of Communicating Systems. LNCS 92*, Springer Verlag, 1980.

[Mil83a] G.J. Milne. *CIRCAL and the representation of Communication, Concurrency and Time.* Technical Report, Dept.of Computer Science / Univ.Edinburgh (UK), 1983. CSR 151-83.

[Mil83b] R. Milner. Calculi for synchrony and asynchrony. *TCS*, (25):267–310, 1983.

[NRSV90] X. Nicollin, J.L. Ritchier, J. Sifakis, and J. Voiron. ATP-an algebra for timed processes. In M. Broy and C.B. Jones, editors, *TC2-Working Conference on Programming Concepts and Methods*, North-Holland, 1990.

[OdF90] Y. Ortega-Mallén and D. de Frutos-Escrig. Timed observations: a semantic model for real-time concurrency. In M. Broy and C.B. Jones, editors, *TC2-Working Conference on Programming Concepts and Methods*, North-Holland, 1990.

[OH86] E.R. Olderog and C.A.R. Hoare. Specification-oriented semantics for communicating processes. *Acta Informatica*, (23):9–66, 1986.

[Old86] E.R. Olderog. Process theory : semantics, specification and verification. In *Advanced School on Current Trends in Concurrency*, Springer Verlag, 1986.

[Old87] E.R. Olderog. TCSP : theory of communicating sequential processes. In *Advances in Petri Nets 86*, Springer Verlag, 1987.

[Ort90] Y. Ortega-Mallén. *En Busca del Tiempo Perdido.* PhD thesis, Fac. CC. Matem['aticas, Univ.Complutense de Madrid, 1990.

[RR87] G.M. Reed and A.W. Roscoe. Metric spaces as models for real-time concurrency. In *Mathematical Foundations of Programming Language Semantics*, Springer Verlag, 1987.

[SM81] A. Salwicki and T. Muldner. On the algoritmic properties of concurrent programs. In *Logic of Programs 1979*, Springer Verlag, 1981.

[Tof88] C. Tofts. *Temporal Ordering for Concurrency.* Technical Report, Dept.of Computer Science / Univ.Edinburgh (UK), 1988. CSR 261-88.

[TV89] D. Taubner and W. Vogler. The step failure semantics and a complete proof system. *Acta Informatica*, (27):125–156, 1989.