# Stable and timed formats for process algebra

Iain Phillips
Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ
iccp@doc.ic.ac.uk

and

Irek Ulidowski
School of Computing, University of North London
Eden Grove, London N7 8DB
ex13ulidowi@clstr.unl.ac.uk

Abstract

We propose a format of transition rules (*stable* format) for processes which extends De Simone's in that it admits process operators which recognise stability, that is, the inability to perform autonomous actions. The format uses negative premises, but differently from previously proposed formats. We show that refusal testing is the trace congruence generated by the format. We identify a subformat (*timed* format) which serves as a format for discrete-timed process algebra in the style of Hennessy and Regan's TPL.

## 1. Introduction

It is well-known how to use syntax-directed rules to associate labelled transition systems with terms in process algebra. This is often called *structured operational semantics* (SOS) [Plo81], and was pioneered by Milner for CCS [Mil80, Mil89]. The meaning of each operator on processes is given by transition rules. We can classify operators according to the form these may take. We say that an operator is in a certain format if its rules belong to that format.

Formats were first studied by De Simone [DeS85]. Further work, including negative transitions, was carried out in [BIM88], [GrV92], [Gro90], [BoG91], [ABV92], [vaG93]. Formats with silent actions have been studied by Bloom [Blo90], Vaandrager [Vaa91] and the second author [Uli92], [Uli93], [Uli94].

It is customary to measure the power of a format by identifying the congruence associated with it. Formats have variously yielded ready bisimulation [BIM88], strong bisimulation [GrV92], testing equivalence [Vaa91], copy + refusal testing equivalence [Uli92]. Our first aim in this paper is to propose a format (stable format) which has the strength of refusal testing [Phi87]. Information about actions being refused has been incorporated via negative hypotheses into a number of formats, but they have yielded congruences which are finer than refusal equivalence. It was wrongly claimed in [Uli93] that the RSOS format was suitable—we are grateful to Rob van Glabbeek for showing us a counterexample. It therefore seems clear that we must be more restrictive in our use of negative transitions.

Silent actions, generally denoted by $\tau$ as in CCS, are often treated as a generalization of work which has previously been carried out for the case where all actions are visible. In the present work $\tau$ plays an essential role. The only negative information we allow is stability, in other words the inability to perform autonomous actions. An autonomous action is one which can be completed without the participation of the environment. We shall adopt the usual view, as typified by CCS, that $\tau$ is the only autonomous action. The upshot is that we allow negative transitions, but only if they are labelled by $\tau$.

Having considered stable format, we specialize it to obtain a format for discrete-timed processes in the style of Hennessy and Regan [HeR91]. The special action $\sigma$ marks the tick of a global clock. All active processes must participate in $\sigma$ for it to happen. We do not know of any previously proposed format for timed processes.

The plan of the paper is as follows: After some preliminaries in Section 2, Section 3 considers stable format, and Section 4 considers timed format.

## 2. Preliminaries

### 2.1. Transition systems and refusal testing

Let A be some set of actions containing a distinguished element $\tau$. A (labelled) transition system over A is a triple $(P, \rightarrow, p)$, where P is a set of states, the transition relation $\rightarrow$ is a relation on $P \times A \times P$, and $p \in P$ is a distinguished start state. We refer loosely to $(P, \rightarrow, p)$ as p.

Suppose that $w \in A^*$ and $q \in P$. We define $p \xrightarrow{w} q$ in the obvious way. Let $TS_A$ be the class of all transition systems over A. A transition system p is *strongly convergent* if it is impossible to perform an infinite sequence of $\tau$-transitions from any state. It is *finite-branching* if for all states there are only finitely many successor states. It is *sort-finite* if there is a finite set from which all its transition labels are drawn, i.e. there is a finite subset C of A such that if $p \xrightarrow{w} q$ then $w \in C^*$.

We have parametrized the class of transition systems by the set of actions, since there will be times when we wish to move from one set of actions A to another larger set B. There is an obvious embedding, which transforms a member of $TS_A$ into a member of $TS_B$. Let us call this $\eta_{AB}$. It is natural to think of the usual equivalences as being on arbitrary transition systems, rather than depending on the set of actions. So let us call an equivalence ~ on transition systems *generic* if it is actually a family of equivalences $\sim_A$ on $TS_A$ (each set A) satisfying the following conservation condition:

if $A \subseteq B$ and $p, q \in TS_A$ then

$p \sim_A q$ iff $\eta_{AB}(p) \sim_B \eta_{AB}(q)$

The equivalences we shall consider will always be generic.

For $w \in A^*$, let $\hat{w}$ denote w with all $\tau$s removed (as in [Mil89]). We shall refer to members of $A^*$ as *strings*, and to members of $(A-\{\tau\})^*$ as *traces.*

DEFINITION 2.1 (trace preorder). Let $p, q \in TS_A$. $p \leq_T q$ iff for all $w \in A^*$,

$p \xrightarrow{\hat{w}}$ implies $q \xrightarrow{\hat{w}}$

Also p is trace equivalent to q (written $p \sim_T q$) iff $p \leq_T q$ and $q \leq_T p$.

Refusal testing was introduced in [Phi87]. For simplicity we shall present "may" refusal testing, which may be seen as a generalization of traces to what have been called failure traces [vaG90].

DEFINITIONS 2.2. Let $p \in TS_A$. p is stable iff $p \not\xrightarrow{\tau}$. Let $C \subseteq A-\{\tau\}$. p refuses C (written p ref C) iff p is stable and $p \not\xrightarrow{a}$ for all $a \in C$. For technical convenience we shall abuse notation and also write $p \xrightarrow{C}$.

A *may refusal test* is a string consisting of actions a  A-{τ} and sets of actions (refusals) C  A-{τ}, where we ban adjacent sets. For instance a{b,c}de is allowed, but not a{b}{c}de. With may testing we can combine successive refusals to form sets (with must testing (see [Phi87]) it matters which is the first to be tested). Of course sensible tests would not allow a set C to be followed by a  C. We allow infinite sets. In what follows we refer to may refusal tests simply as tests. We let t,... range over tests.

A *refusal string* is a string of actions drawn from A (this time including τ) and subsets of A-{τ}, where we again ban adjacent sets. In an obvious way we can talk of $p \xrightarrow{w}$ where w is a refusal string. Clearly if w is a refusal string then $\hat{w}$ is a test.

DEFINITION 2.3 (may refusal testing preorder). For any p  $TS_A$ and test t,

p **may** t iff  a refusal string w. $p \xrightarrow{w}$  & $\hat{w} = t$

For any p, q  $TS_A$

$p \sqsubseteq q$ iff  for all t. p **may** t implies q **may** t

It is natural to ask whether we get as much power by restricting to refusal tests where the refusal sets are finite. Say that a refusal test t is finite if all its refusal sets are finite.

DEFINITION 2.4. For any processes p, q  $TS_A$

$p \sqsubseteq_{fin} q$ iff  for all finite t. p **may** t implies q **may** t

Clearly $p \sqsubseteq q$ implies $p \sqsubseteq_{fin} q$. When can we assert the converse? The following proposition is clearly related to the fact that refusal testing is generic.

PROPOSITION 2.5. Suppose that p, q  $TS_A$ are sort-finite. Then

$p \sqsubseteq_{fin} q$ iff $p \sqsubseteq q$  □

Clearly if A is finite then every member of $TS_A$ is sort-finite.

PROPOSITION 2.6. Suppose that p, q are finite-branching and strongly convergent. Then

$p \sqsubseteq_{fin} q$ iff $p \sqsubseteq q$  □

EXAMPLE 2.7. Let $q = \tau a_0 + \tau(\tau a_1 + \tau(\tau a_2 + \ldots))$ and $p = q + \tau$ (using CCS notation). Let $A = \{a_0, a_1, \ldots\}$. Then not q **may** A, but if $C \subseteq_{fin} A$, q **may** C. However p **may** A. So $p \sqsubseteq_{fin} q$ but not $p \sqsubseteq q$.

## 2.2. Formats and congruences

In this section we shall take a fairly abstract view of formats. For our results the important thing is that a format F has an operational meaning in terms of transition systems.

A *language definition* (usually called a transition system specification [GrV92]) is a set $\Sigma$ of function symbols of various arities (a signature) together with a set rules(f) for each $f \in \Sigma$ and a set $A(\Sigma)$ of actions. We shall loosely refer to this as $\Sigma$. A format F is a way of classifying language definitions and giving them operational meaning. Extensionally a format F is a set of language definitions such that if $\Sigma$ belongs to F then for each $f \in \Sigma$ there is a map $O_{F,\Sigma}(f)$: $TS_{A(\Sigma)}{}^n \rightarrow TS_{A(\Sigma)}$, where $n \geq 0$ is the arity of f. What we have called a language definition has been called a transition system specification [GrV92]. The change in terminology is partly because our definition is slightly different, but mainly because we are here concentrating on the operators which are definable in the language, rather than on the global transitions system associated with it (where the states are the closed terms of the language).

Let $\Sigma$ be a language definition. $\Sigma$-terms are formed from variables and function symbols from $\Sigma$. We let X,... range over variables and u,... range over $\Sigma$-terms. We may display the variables by writing $u(X_1,\ldots,X_n)$, and unless stated otherwise this will imply that u uses some, but not necessarily all, of the variables $X_1,\ldots,X_n$. Let $p_1,\ldots,p_n$ be members of $TS_{A(\Sigma)}$. By $u(p_1,\ldots,p_n)$ we mean the interpretation of $u(X_1,\ldots,X_n)$ in $TS_{A(\Sigma)}$ where each function symbol f occurring in u is interpreted by $O_{F,\Sigma}(f)$ and $X_i$ is interpreted by $p_i$, i = 1,...,n. When we come to internalize refusal testing we shall also need to extend the definition of $u(p_1,\ldots,p_n)$ by allowing $p_1,\ldots,p_n$ to be members of $TS_A$, where $A \supseteq A(\Sigma)$. It then means $u(\eta_{A,A(\Sigma)}(p_1),\ldots,\eta_{A,A(\Sigma)}(p_n))$.

DEFINITION 2.8. Let F be a format and let ~ be a generic congruence on transition systems. Then ~ is an F-*congruence* if the following holds for all $\Sigma \in F$, $A \supseteq A(\Sigma)$, $p,q \in TS_A$,

$u(X_1,\ldots,X_n)$ a $\Sigma$-term (any n), and $r_2,\ldots,r_n$ $TS_{A(\Sigma)}$:

if $p \sim_A q$ then $u(p,r_2,\ldots,r_n) \sim_{A(\Sigma)} u(q,r_2,\ldots,r_n)$

The ~-congruence *generated by* F (notation $\sim^F$) is defined to be the largest F-congruence contained in ~. In a standard way we may characterize $\sim^F$ as follows: For p, q in $TS_A$, $p \sim^F q$ iff for all language definitions $\Sigma$ (with action set $A(\Sigma)$ A) in F-format, and all $\Sigma$-terms $u(X_1,\ldots,X_n)$ and all $r_2,\ldots,r_n$ in $TS_{A(\Sigma)}$,

$u(p,r_2,\ldots,r_n) \sim u(q,r_2,\ldots,r_n)$

We refer to $\sim_T^F$ as the trace congruence generated by F.

In the above we can replace the equivalence ~ by a preorder $\leq$, and change congruence to precongruence.

### 3. Stable format

In this section we define a format which we call *stable* format (SF). We start by reviewing De Simone's format [DeS85, Definition 1.9], which we refer to as DeS. The allowable language definitions are as follows: Let $\Sigma$ be a signature, and let $\alpha$,... range over $A(\Sigma)$ while a,... range over $A(\Sigma)$-$\{\tau\}$. Each f $\Sigma$ is defined by a (possibly empty) set of rules of the form

$$\frac{\{X_i \xrightarrow{\alpha_i} X_i'\}_{i \in I}}{f(\mathbf{X}) \xrightarrow{\alpha} u(\mathbf{X}')}$$

Here **X** is an abbreviation for $X_1,\ldots,X_n$ and I $\{1,...,n\}$. Note that each $X_i$ occurs at most once in the premises. $X_i'$ is defined to be $X_i$ for i I. u is a $\Sigma$-term built from the variables $X_i'$. We impose the condition on u that each $X_i'$ must occur at most once. De Simone calls this "linearity".

The map $O_{DeS,\Sigma}(f)$: $TS_{A(\Sigma)}{}^n \rightarrow TS_{A(\Sigma)}$ is defined in a straightforward way. Let $p_1,\ldots,p_n$ be members of $TS_{A(\Sigma)}$. Introduce new constants into $\Sigma$ for the states of $p_1,\ldots,p_n$ to get $\Sigma'$ and introduce axioms for the transitions (cf [Pin93]). The states of the new TS are the terms over $\Sigma'$, and the transitions are those deducible from the rules of $\Sigma$ together with the new axioms. The initial state is of course $f(p_1,\ldots,p_n)$.

We are interested in treating $\tau$ as a silent and autonomous action. To reflect the fact that it is silent, we shall not allow it in the premises, so that we get

$$\frac{\{X_i \xrightarrow{a_i} X_i'\}_{i \in I}}{f(\mathbf{X}) \xrightarrow{\alpha} u(\mathbf{X}')}$$

We shall refer to rules of this form as DeS rules. Note that $\tau$ is still allowed in the conclusion. To reflect the fact that it is autonomous we must allow the subprocesses to perform $\tau$ freely and without being detected by outer levels. We nominate a set active(f) $\subseteq$ {1,...n} of active arguments and stipulate that rules(f) must contain the rule $\tau_i$ for each $i \in$ active(f), where $\tau_i$ is

$$\frac{X_i \xrightarrow{\tau} X_i'}{f(\mathbf{X}) \xrightarrow{\tau} f(\mathbf{X}')}$$

We insist that active(f) must contain every i such that there is a DeS rule with $X_i$ in the premises.

We refer to the revised format as $DeS_\tau$. The ideas of the $\tau$ rules and active arguments have been used by Bloom [Blo90] and Vaandrager [Vaa91].

PROPOSITION 3.1 (as in [Vaa91, Theorem 4.7]). Trace equivalence $\sim_T$ is a congruence for $DeS_\tau$. So it is the trace congruence generated by $DeS_\tau$.

The format is rich enough to define languages such as CSP [Hoa85]. It allows all the operators of CCS except for summation, which does not have the necessary $\tau$ rules. Usually trace equivalence is regarded as too abstract in that it neglects deadlock and liveness, so that testing equivalence [DNH84] or failures equivalence [BHR84] are used instead. These are also congruences for $DeS_\tau$ [Vaa91].

We now augment the format with the following rules:

$$\frac{\{X_i \xrightarrow{a_i} X_i'\}_{i \in I} \quad f(\mathbf{X}) \not\xrightarrow{\tau}}{f(\mathbf{X}) \xrightarrow{a} u(\mathbf{X}')}$$

The conditions on I, u($\mathbf{X}'$) are exactly as for DeS rules. We shall refer to these new rules as s-rules. They are in addition to DeS and $\tau$-rules. Notice that the conclusion must have a visible action. We call the new format *stable* format (SF). The s-rules involve a negative

premise, but they are structured in such a way that this does not cause problems in deciding their operational meaning.

DEFINITION 3.2. A language definition $\Sigma$ is in *finitely overlapping* SF-format (FOSF) if it is in SF-format and for every $f \in \Sigma$ and $\alpha \in A(\Sigma)$, there are only finitely many rules with conclusion $f(\mathbf{X}) \xrightarrow{\alpha} \ldots$

The finite overlapping format excludes renaming functions with infinite pre-images and hiding of infinite sets of actions. This is perhaps not too much of a loss. More seriously, it also excludes CCS parallel composition | (with infinite action set), in view of the following rule:

$$\frac{X \xrightarrow{a} X' \qquad Y \xrightarrow{\bar{a}} Y'}{X|Y \xrightarrow{\tau} X'|Y'}$$

However we could replace | by a family of finite-sorted operators $|_B$, where $B \in_{fin} A$. The rules would be:

$$\frac{X \xrightarrow{a} X' \qquad Y \xrightarrow{\bar{a}} Y'}{X|_B Y \xrightarrow{\tau} X'|_B Y'} \quad a \in B \qquad\qquad \frac{X \xrightarrow{a} X'}{X|_B Y \xrightarrow{a} X'|_B Y} \quad a \notin A$$

together with a rule similar to the right-hand one for Y and $\tau$-rules.

THEOREM 3.3. (i) Refusal testing preorder $\sqsubseteq$ is an SF-precongruence.

(ii) $\sqsubseteq_{fin}$ is an FOSF-precongruence.

We omit the proof for lack of space, but make a few brief remarks. The essential reason why refusal testing is a congruence is because positive information (a transition) is deduced in SF from positive and negative information (the latter being stability and refusals), but negative information comes entirely from negative information. To make this explicit we can transform the format into one with rules of the following form:

$$\frac{X_1 \text{ ref } C_1 \quad \ldots \quad X_n \text{ ref } C_n}{f(\mathbf{X}) \text{ ref } C}$$

A single rule of SF will in general translate into many rules of the above kind. Given a term $u(\mathbf{X})$ and a refusal test t we can find refusal tests such that if $\mathbf{p}$ may pass these tests then $u(\mathbf{p})$ **may** t.

We now show how to internalize refusal testing in the format.

PROPOSITION 3.4. Let A be a set of actions. There are language definitions $\Sigma$ in SF and $\Sigma'$ in FOSF such that for any p, q in $TS_A$,

(i) $p \sqsubseteq q$ iff for all $\Sigma$-terms u(X), $u(p) \leq_T u(q)$

(ii) $p \sqsubseteq_{fin} q$ iff for all $\Sigma'$-terms u(X), $u(p) \leq_T u(q)$

Proof. We define $\Sigma$ as follows: It has function symbols $r_C(X)$ (each $C \subseteq A$), and a parallel composition T|X, which is to model tests being applied to processes. This will be reminiscent of CCS |, though not identical. The action set $A(\Sigma)$ will be $A \cup \{\sigma, \omega\}$. We also use action prefixing $\alpha.T$ and a constant $\mathbf{0}$, if these are not already included. The rules are:

$$\frac{T \xrightarrow{a} T' \quad X \xrightarrow{a} X'}{T|X \xrightarrow{\tau} T'|X'} \quad a \in A \qquad \frac{T \xrightarrow{\sigma} T' \quad T|X \not\xrightarrow{\tau}}{T|X \xrightarrow{\sigma} T'|X}$$

$$\frac{T \xrightarrow{\omega} T'}{T|X \xrightarrow{\omega} \mathbf{0}} \qquad \frac{}{r_C(T) \xrightarrow{a} \mathbf{0}} \quad a \in C \qquad \frac{}{r_C(T) \xrightarrow{\sigma} T}$$

(plus $\tau$-rules). Using a CCS-like notation, $r_C(T)$ is in effect $\Sigma_{a \in C} a.\mathbf{0} + \sigma.T$. Notice that if we allow C to be infinite then $r_C(T)$ is not finite branching. Also if A is infinite then | is not finitely overlapping.

Now that we have defined $\Sigma$, we translate our tests t into $\Sigma$-terms $t^\dagger$ as in the following inductive definition:

$\varepsilon^\dagger = \omega.\mathbf{0}$

$(at)^\dagger = a.t^\dagger$

$(Ct)^\dagger = r_C(t^\dagger)$

Here $\varepsilon$ is the empty test. We need something like $\omega$ to recognize that the test has been completed. It is not hard to see that for any p in $TS_A$, and any test t,

p may t iff $t^\dagger|p \xrightarrow{w}$ for some string w containing $\omega$.

Together with the previous theorem this is enough to establish (i).

The definition of $\Sigma'$ is as for $\Sigma$ except that we only allow finite refusal sets C when forming $r_C(X)$, and we replace | by a family of operators $|_B$ for each $B \subseteq_{fin} A$ (this is much as

earlier, when we replaced CCS | by $|_B$'s, and is to ensure that $\Sigma'$ is in finitely overlapping format). The new rules are:

$$\frac{T \xrightarrow{a} T' \quad X \xrightarrow{a} X'}{T|_B X \xrightarrow{\tau} T'|_B X'} \quad a \in B \qquad \frac{T \xrightarrow{\sigma} T' \quad T|_B X \not\xrightarrow{\tau}}{T|_B X \xrightarrow{\sigma} T'|_B X} \qquad \frac{T \xrightarrow{\omega} T'}{T|_B X \xrightarrow{\omega} \mathbf{0}}$$

(plus $\tau$-rules). Take any finite test t. Let B be large enough to include all actions mentioned in t (only finitely many). Then for any TS p

p may t iff $(t^\dagger |_B p) \xrightarrow{w}$ for some string w containing $\omega$.

Again this is enough to establish (ii). □

COROLLARY 3.5. (i) $p \sqsubseteq q$ iff $p \leq_T^{SF} q$.

(ii) $p \sqsubseteq_{fin} q$ iff $p \leq_T^{FOSF} q$. □

## 4. A format for timed process algebra

We present a format for process algebra with discrete time. The action set has a special action $\sigma$ which represents the passage of time (the tick of the global clock). Following many authors we require $\tau$ actions to be "urgent", so that time cannot pass until the system is stable. This is often called the maximal progress assumption. Our format is inspired by TPL [HeR91], which was in turn influenced by [Phi89]. It is a subformat of SF, where we make a restriction on the s-rules. Apart from TPL, other timed process algebras have been proposed in [ReR88], [BaB91], [NRSV90], [MoT90], [Wan90] among others.

DEFINITION 4.1 (Timed format). Let a language definition $\Sigma$ have action set A containing special actions $\tau$, $\sigma$. Then $\Sigma$ is in timed format (TF) if it is in SF and satisfies the following further conditions:

(1) $\sigma$ may not appear in either the premises or conclusion of a DeS rule.

(2) Each operator f has no s-rules apart from the single $\sigma$-rule as follows, which is compulsory:

$$\frac{\{X_i \xrightarrow{\sigma} X_i'\}_{i \in active(f)} \quad f(\mathbf{X}) \not\xrightarrow{\tau}}{f(\mathbf{X}) \xrightarrow{\sigma} u}$$

The σ-rules have the effect of saying that the passage of time will be marked by all processes when the system is stable.

DEFINITION 4.2. An operator f in a language definition Σ in TF is said to be *untimed* if its σ-rule is

$$\frac{\{X_i \xrightarrow{\sigma} X_i'\}_{i \ \mathrm{active}(f)} \quad f(\mathbf{X}) \overset{\tau}{\not\rightarrow}}{f(\mathbf{X}) \xrightarrow{\sigma} f(\mathbf{X}')}$$

EXAMPLES 4.3. The following operators are in TF. We omit the τ-rules for active arguments.

(1) A delay operator

$$\frac{}{d(X) \xrightarrow{\sigma} X}$$

(2) A "timeout" operator (cf [NiS91], [HeR91])

$$\frac{X \xrightarrow{a} X'}{X \rhd Y \xrightarrow{a} X'} \qquad \frac{X \xrightarrow{\sigma} X' \qquad X \rhd Y \overset{\tau}{\not\rightarrow}}{X \rhd Y \xrightarrow{\sigma} Y}$$

This can be varied to give a "watchdog" operator (again cf [NiS91])

$$\frac{X \xrightarrow{a} X'}{X \rhd Y \xrightarrow{a} X' \rhd Y} \qquad \frac{X \xrightarrow{\sigma} X' \qquad X \rhd Y \overset{\tau}{\not\rightarrow}}{X \rhd Y \xrightarrow{\sigma} Y}$$

In both cases X is active and Y is not. In the σ−rules the X' is thrown away.

(3) Languages like CCS (except for +) or CSP can be straightforwardly put into TF by adding the necessary σ-rules to make their operators untimed. For instance, action prefixing has the "idling" rule

$$\frac{}{a.X \xrightarrow{\sigma} a.X} \qquad a \neq \tau$$

just as in TPL (the X is inactive).

TPL is not actually in TF because it uses CCS-style choice. However it can be put into TF with a few adjustments.

We can place a restriction on transition systems to get those which model timed processes:

DEFINITION 4.4. Let $\sigma,\tau$ A. Then $(P,\rightarrow,p)$ $TS_A$ is a *timed* TS if for all q P

(1) if $q \xrightarrow{\sigma} r$, $q \xrightarrow{\sigma} r'$ then r = r'.

(2) either $q \xrightarrow{\sigma}$ or $q \xrightarrow{\tau}$, but not both.

Denote the set of such timed TSs by $TTS_A$. (1) has been called time determinism [NiS91].

The next result says that using operators in TF keeps us within timed TSs:

PROPOSITION 4.5. Let $\mathbf{p} = p_1,..,p_n$ $TTS_A$ and let $\Sigma$ be in TF with $A(\Sigma)$ A. Then for any $\Sigma$-term $u(\mathbf{X})$, $u(\mathbf{p})$ $TTS_{A(\Sigma)}$. □

DEFINITION 4.6. A $\sigma$-test is a refusal test where every refusal is immediately followed by a $\sigma$.

$p \sqsubseteq_\sigma q$ iff for every $\sigma$-test t, p may t implies q may t.

REMARK 4.7. Hennessy and Regan [HeR91] characterise testing equivalence over TPL by means of an ordering based on "barbs". It is not hard to show that (at least for strongly convergent processes) their definition coincides with $\sqsubseteq_\sigma$.

PROPOSITION 4.8. (i) $\sqsubseteq_\sigma$ is a TF-precongruence.
(ii) $p \sqsubseteq_\sigma q$ iff $p \leq_T^{TF} q$.

## Conclusions

We have defined a format for discrete-timed process algebra. We have seen how a restricted form of refusal testing is a congruence for the format. The format is a special case of a more general format which admits operators that detect stability. Throughout the work silent actions have been considered in an essential way. Further work might include defining a timed format for non-discrete time.


## Acknowledgements

We thank Rob van Glabbeek for a helpful conversation and pointing out an error in our previous work. We have also benefited from discussion with Sophie Pinchinat. The first author was supported by SERC grant GR/F72475.

**References**


[ABV92] L. Aceto, B. Bloom, and F.W. Vaandrager. Turning SOS rules into equations. LICS, 1992.

[BaB91] J.C.M. Baeten and J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing* **3** (142-188), 1991.

[BHR84] S.D. Brookes, C.A.R. Hoare, and A.W. Roscoe. A Theory of Communicating Sequential Processes. *JACM* **31** (560-599), 1984.

[BIM88] B. Bloom, S. Istrail, and A.R. Meyer. Bisimulation can't be traced: Preliminary report. POPL, 1988. Accepted by *JACM*.

[Blo90] B. Bloom. Strong process equivalence in the presence of hidden moves. Preliminary report, 1990.

[BoG91] R.N. Bol and J.F. Groote. The meaning of negative premises in transition systems specifications (extended abstract). ICALP. *LNCS* **510** (481-494). Springer, 1991.

[DeS85] R. de Simone. Higher-level synchronising devices in MEIJE-SCCS. *Theoretical Computer Science* **37** (245-267), 1985.

[DNH84] De Nicola, R. and M.C.B. Hennessy. Testing Equivalences for Processes. *Theoretical Computer Science* **34** (83-134), 1984.

[Gro90] J.F. Groote. Transition systems specifications with negative premises (extended abstract). CONCUR 90. *LNCS* **458** (332-341). Springer, 1990.

[GrV92] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation* **100** (202-260), 1992.

[HeR91] M. Hennessy and T. Regan. A process algebra for timed systems. Dept. of Computer Science, University of Sussex, Tech. Report 5/91. 1991.

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

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

[Mil89] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[MoT90] F. Moller and C. Tofts. A temporal calculus of communicating systems. CONCUR'90. *LNCS* **458** (401-415). Springer, 1990.

[NiS91] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. CAV'91. *LNCS* **575** (376-398). Springer, 1991.

[NRS90] X. Nicollin, J.-L. Richier, J. Sifakis and J. Voiron. ATP: an algebra for timed processes. Proc. IFIP TC 2 Working conference on programming concepts and methods. 1990.

[Phi87] I.C.C. Phillips. Refusal testing. *Theoretical Computer Science* **50** (241-284), 1987.

[Phi89] I.C.C. Phillips. CCS with broadcast stability. Manuscript, 1989.

[Pin93] S. Pinchinat. Des bisimulations pour la sémantique des systèmes réactifs. Thesis, Grenoble, 1993.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[ReR88] G.M. Reed and A.W. Roscoe. A timed model for communicating sequential processes. *Theoretical Computer Science* **58** (249-261), 1988.

[Uli92] I. Ulidowski. Equivalences on observable processes. LICS, 1992.

[Uli93] I. Ulidowski. Congruences for τ-respecting formats of rules. Theory and Formal Methods 1993, ed. G. Burn, S. Gay and M. Ryan. Workshops in Computing. Springer, 1993.

[Uli94] I. Ulidowski. Local testing and implementable concurrent processes. PhD Thesis, Imperial College, London. Forthcoming.

[Vaa91] F.W. Vaandrager. On the relationship between process algebra and input/output automata (extended abstract). LICS, 1991.

[vaG90] R. van Glabbeek. The linear time – branching time spectrum. CONCUR '90. *LNCS* **458** (278-297). Springer, 1990.

[vaG93] R. van Glabbeek. Full abstraction in Structural Operational Semantics (extended abstract). 1993.

[Wan90] Wang Yi. Real-time behaviour of asynchronous agents. CONCUR '90. *LNCS* **458** (502-520). Springer, 1990.