CMU-CS-83-134

# A Semantics and Proof System for Communicating Processes

Stephen D. Brookes
Computer Science Department
Carnegie-Mellon University
Pittsburgh, Pa.

May 1983

A version of this paper was presented at the 1983 NSF/ONR Workshop on Logics of Programs, Pittsburgh, May 1983. References should cite the Proceedings, published by Springer Verlag.

The research reported in this paper was supported in part by funds from the Computer Science Department of Carnegie-Mellon University, and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory under Contract F33615-81-K-1539. The views and conclusions contained in it are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

# A SEMANTICS AND PROOF SYSTEM FOR COMMUNICATING PROCESSES.

Stephen D. Brookes
Carnegie-Mellon University
Pittsburgh
Pennsylvania 15213
USA.

## 0. Introduction.

In this paper we describe a semantic model for communicating sequential processes extending the so-called *failures* model of CSP which appears in [B1,2], [HBR] and [R]. We also give a proof system for proving semantic equivalence of processes. This provides an axiomatic characterisation of our semantics.

The failures model of processes represents the behaviour of a sequential process as a set of failures, each of which is a finite piece of information about a possible behaviour of the process. A failure consists of a *trace,* recording a possible finite sequence of actions, and a *refusal set,* representing a set of events which the process *may* decide to refuse at the next step. The idea of using traces to represent process behaviour appeared in [H2], where the inability of traces alone to model deadlock was also noted.

Each failure is essentially a potential result of a nondeterministic decision by the process. This model is well suited to reasoning about the potential deadlock behaviour of a process, since the possibility of deadlock is represented explicitly by the ability to refuse all sets of actions. As described in the above references, the failures model can be given the structure of a complete semi-lattice under the superset ordering, which corresponds to a measure of nondeterminism: if the failure set of process $P$ contains as a subset the failure set of $Q$, then every possible nondeterministic decision of $Q$ is also possible for $P$, and we may say that $P$ is more nondeterministic than $Q$. Since the space of failure sets, under this ordering, has the structure of a complete semi-lattice, we know that limits of chains of processes exists, and every non-empty set of processes has a greatest lower bound. Moreover, applying well-known arguments of fixed point theory (see [LNS] for example), we know that every monotone function on failure sets has a least fixed point; we may therefore give semantics to recursively defined processes, provided that the operations used in constructing processes are all monotone. This is the case for a wide variety of operations whose definitions are suggested by CSP.

The failures model has been used to give a semantics to a language based on Hoare's CSP in [HBR]. However, there are some undesirable aspects to this model. Notably, the behaviour of processes which may *diverge,* or engage in *infinite internal chatter,* is not adequately modelled by a failure set. Internal chatter is the phenomenon of a process continuously engaging in a sequence of actions which are internal or hidden from the environment in which the process is operating: that environment has no control or influence on the process while the process is continuing to perform invisible actions, and the process may never respond to requests from the environment. This type of behaviour arises when, for example, two processes are connected and allowed to communicate along a channel which the outside world cannot affect; the communications which may occur along this channel are hidden from the environment. If there is a potentially unbounded sequence of communications along the hidden link then the two processes may indulge in infinite internal chatter.

Attempts were made to treat divergent behaviour by identifying it with either deadlock (STOP) or wholly arbitrary behaviour (CHAOS), as described in [HBR,B,R], but none of these is satisfactory. It is possible to extend the failures model in a fairly straightforward way which does model divergence more satisfyingly. The new model, whose development first appeared in [B1,R], still enjoys the order-theoretical properties of a complete semi-lattice, and is therefore well suited to provide a semantic basis for a language of processes. Moreover, we find that we can axiomatize the semantics of a language like CSP by giving a set of axioms and inference rules sound and complete for proving semantic equivalence of terms in the language. Thus we will describe a semantics for processes in both the denotational style and axiomatically.

The language with which we are concerned is a derivative of Hoare's CSP [H1]. To begin with, we will discuss a simple sub-language called here FCSP (for "Finite CSP"), whose terms will denote processes with finite behaviour. For these terms the possibility of divergence does not exist, and we will introduce a failure semantics for this language. The associated logical language will contain assertions of the form $P \sqsubseteq Q$ or $P \equiv Q$, with the interpretation that $P$ semantically approximates $Q$ or is semantically equivalent to $Q$. We will give a set of axioms and inference rules for proving such assertions, and show that the system is both sound and complete; under the given interpretation of assertions, every provable assertion is true and every true assertion is provable.

Next we make the transition to a more general language of processes, by allowing recursive terms. Now terms may denote processes with infinite behaviour. We show how to modify the previous proof system to obtain a new system complete and sound for the larger language. Essentially, we use the well known ideas of *syntactic approximation* of terms, and use the fact that the failure semantics of an infinite process is uniquely determined by its finite syntactic approximants. A new inference rule is added which states this fact and basically allows us to reason about infinite terms by manipulating their finite approximations. Crucial to this work is the fact that all of the process operations in the language are continuous with respect to the nondeterminism ordering. Similar techniques were used by Hennessy and de Nicola [HN] to give a proof system for Milner's language CCS [M].

Bearing in mind our earlier problems with the notion of *divergence,* we have made the proof system sufficiently general to cope with divergent processes, by adding to the language a term $\perp$ (representing divergence) and augmenting the failure set model with a divergence set component. A similar augmentation of the failures model was suggested by Roscoe [R]. The special case of well-behaved (divergence–free) processes turns out to correspond to the sublanguage of all terms in which no recursive subterm has an unguarded occurrence of its bound variable, and in which there is no sub-term $\perp$. As a corollary, the proof system is also complete for the old failures ordering on well-behaved processes.

Throughout this paper we will use $P, Q, R$ to stand for terms in the variant of CSP currently under consideration. We are not necessarily assuming that the universal alphabet $\Sigma$ is finite, but every term will only use a finite set of events. As usual, refusal sets are finite; we use $p\Sigma$ for the finite powerset of $\Sigma$. We will use $X, Y, Z$ to range over $p\Sigma$. Finally, $s, t, u$ range over $\Sigma^*$ and $a, b, c$ over $\Sigma$.

## 1. A simple subset of CSP.

Let FCSP (Finite CSP) be the language generated by the following syntax:

$$P ::= \text{STOP} \mid (a \to P) \mid P \Box P \mid P \sqcap P,$$

where $a$ ranges over $\Sigma$. STOP denotes a process which is unable to perform any event, and represents deadlock. The process $(a \to P)$ must first perform event $a$ and thereafter behaves like $P$. The combinator $\Box$ is a *conditional choice* operator: $P \Box Q$ can behave either like $P$ or like $Q$, and the environment of the process can influence this choice by offering an action at the first step which only one of the two is able to perform then. In contrast, $P \sqcap Q$ behaves either like $P$ or like $Q$ but does not allow its decision to be affected by the desires of its environment; this is an *uncontrollable* choice. (The reader familiar with Hoare's CSP will recognise the connection with guarded commands: the two forms of choice arise when the guards are communications or purely boolean.)

The semantic function $\mathcal{F}$ maps terms to failure sets, and is defined by structural induction as usual, one clause for each syntactic construct of FCSP. This is a *denotational* semantic definition, because the semantics of a term is built up from the semantics of its syntactic constituents.

$$\mathcal{F}[\![\text{STOP}]\!] = \{\, (\langle\rangle, X) \mid X \subseteq \Sigma \,\}$$

$$\mathcal{F}[\![a \to P]\!] = \{\, (\langle\rangle, X) \mid a \not\in X \,\} \cup \{\, (as, X) \mid (s, X) \in \mathcal{F}[\![P]\!] \,\}$$

$$\begin{aligned}\mathcal{F}[\![P \Box Q]\!] = {} & \{\, (\langle\rangle, X) \mid (\langle\rangle, X) \in \mathcal{F}[\![P]\!] \cap \mathcal{F}[\![Q]\!] \,\} \\ & \cup \{\, (s, X) \mid s \neq \langle\rangle \;\&\; (s, X) \in \mathcal{F}[\![P]\!] \cup \mathcal{F}[\![Q]\!] \,\}\end{aligned}$$

$$\mathcal{F}[\![P \sqcap Q]\!] = \mathcal{F}[\![P]\!] \cup \mathcal{F}[\![Q]\!].$$

Note that these clauses capture the intuitive definitions of the syntactic constructs stated above. For instance, $(a \to P)$ must initially perform $a$ because this event does not belong to any of the refusal sets of $\mathcal{F}[\![a \to P]\!]$; and $P \Box Q$ refuses a set initially only if both $P$ and $Q$ can refuse it, but once an event has occurred it behaves either like $P$ or like $Q$.

We will use $\Phi$ to stand for a failure set. Recall that in [HBR] a failure set is a subset of $\Sigma^* \times p\Sigma$ such that

(i) $\text{dom}(\Phi)$ is non-empty and prefix-closed,
(ii) $(s, X) \in \Phi, Y \subseteq X \Rightarrow (s, Y) \in \Phi,$
(iii) $(s, X) \in \Phi, (sa, \emptyset) \not\in \Phi \Rightarrow (s, X \cup \{a\}) \in \Phi.$

These conditions state that (i) the empty trace is always a possible trace of a process, and whenever $st$ is a trace of a process so is $s$; (ii) if $P$ can refuse $X$ after doing $s$ then at the same time it could refuse any subset $Y$ of $X$; (iii) an impossible event can always be included in a refusal set. These are intuitively reasonable properties. It is easy to check that our semantic function does indeed map terms to failure sets satisfying these conditions.

The semantic ordering on failure sets is $\Phi_1 \sqsubseteq \Phi_2 \Leftrightarrow \Phi_1 \supseteq \Phi_2$. We will write $P \sqsubseteq Q$ to mean $\mathcal{F}[\![P]\!] \sqsubseteq \mathcal{F}[\![Q]\!]$, where no confusion can arise. In case this relation holds we say that $P$ is at least as nondeterministic as $Q$, since failures represent consequences of nondeterministic decisions by a process.

Now we introduce the proof system. The logical language is built from FCSP terms and two binary relation symbols $\sqsubseteq$ and $\equiv$. Each formula in the language has the form $P \sqsubseteq Q$ or $P \equiv Q$. The intended interpretation of $P \sqsubseteq Q$ is that $P$ semantically approximates $Q$, and $\equiv$ is interpreted as semantic identity.

We include axioms on idempotence, symmetry, associativity of $\sqcap$ and $\Box$, distribution of these two operators over each other, and some interactions with prefixing. The inference rules assert monotonicity of the operators with respect to $\sqsubseteq$, and state that $\sqsubseteq$ is a partial order and $\equiv$ the associated equivalence. The following table lists the axioms and rules:

(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) $P \Box \mathrm{STOP} \equiv P$

(A10) $P \sqcap Q \sqsubseteq P$

(A11) $(a \to P) \sqcap (a \to Q) \equiv (a \to P \sqcap Q)$

(A12) $(a \to P) \Box (a \to Q) \equiv (a \to P \sqcap Q)$

(O1) $$\frac{P \sqsubseteq Q \sqsubseteq P}{P \equiv Q}$$

(O2) $$\frac{P \equiv Q}{P \sqsubseteq Q \sqsubseteq P}$$

(O3) $$\frac{P \sqsubseteq Q \sqsubseteq R}{P \sqsubseteq R}$$

(M1 ) $$\frac{P \sqsubseteq Q}{(a \to P) \sqsubseteq (a \to Q)}$$

(M2) $$\frac{P_1 \sqsubseteq Q_1 \ \& \ P_2 \sqsubseteq Q_2}{P_1 \sqcap P_2 \sqsubseteq Q_1 \sqcap Q_2}$$

(M3) $$\frac{P_1 \sqsubseteq Q_1 \ \& \ P_2 \sqsubseteq Q_2}{P_1 \Box P_2 \sqsubseteq Q_1 \Box Q_2}$$

**Table 1**

*Soundness.*

In order to prove soundness of the system, it is enough to show that all the axioms are valid and that the inference rules are sound. Each axiom has already appeared in [HBR] and [B1], where proofs of validity were given; similarly all of the operators were shown to be monotonic, so the inference rules (M1)–(M3) are valid. For details the reader is referred to [B1]. We know, therefore, that every provable formula is true. We write $\vdash P \sqsubseteq Q$ when the formula $P \sqsubseteq Q$ is provable. The following theorem states that the proof system is ***sound.***

*Theorem 1.1:* For all terms P,Q

$$\vdash P \sqsubseteq Q \Rightarrow \mathcal{F}[\![P]\!] \supseteq \mathcal{F}[\![Q]\!].$$

*Derived laws.*

The following laws are derivable, and hence valid. They will be useful in establishing completeness. The first states the connection between nondeterministic choice and the ordering. The second says that nondeterministic choice allows more failures in general than conditional choice, in accordance with our intuition and earlier results on these operators. These laws will be heavily used in establishing the existence of normal forms.

*Lemma 1.2:*

The following formulae can be derived in the above proof system:

$$\begin{array}{ll} \text{(D1)} & P \sqcap Q \equiv P \Leftrightarrow P \sqsubseteq Q \\ \text{(D2)} & P \sqcap Q \sqsubseteq P \Box Q \\ \text{(D3)} & P \sqcap (Q \Box R) \sqsubseteq P \Box Q. \end{array}$$

*Proof.* For (D1) we have

$$\vdash P \sqcap Q \sqsubseteq P$$

by (A10). And if we assume $P \sqsubseteq Q$ is provable, we have:

$$P \sqsubseteq Q \vdash P \sqcap P \sqsubseteq P \sqsubseteq Q,$$

by (M2). The result follows by (A1) and (O1).

For (D2):

$$\begin{aligned} (P \sqcap Q) \sqcap (P \Box Q) &\equiv ((P \sqcap Q) \sqcap P) \Box ((P \sqcap Q) \sqcap Q) && \text{by (A7)} \\ &\equiv (P \sqcap Q) \Box (P \sqcap Q) && \text{by (A1)} \\ &\equiv P \sqcap Q && \text{by (A2)} \end{aligned}$$

The result follows from (D1).

For (D3) we have

$$\begin{aligned} P \sqcap (Q \Box R) &\equiv (P \sqcap Q) \Box (P \sqcap R) && \text{by (A7)} \\ &\sqsubseteq (P \Box Q) \Box P && \text{by (M3), (A10) and (D2)} \\ &\equiv (P \Box P) \Box Q && \text{by (A6)} \\ &\equiv P \Box Q && \text{by (A2)} \end{aligned}$$

That completes the proof. ▮

*Completeness.*

We will show that whenever the failures of $P$ include the failures of $Q$ the formula $P \sqsubseteq Q$ is provable. For the proof we use a ***normal form*** theorem. We define a class of normal forms and show that every term is provably equivalent to a unique term in normal form. Moreover, we show that whenever the failures of one normal form include the failures of another, the corresponding formula is provable.

Essentially, a normal form will be a term with a ***uniform*** structure, rather like a nondeterministic composition of a collection of ***guarded*** terms. In order to get uniqueness of normal forms we will require certain closure conditions on the sets of guards appearing at each position in the term; these conditions amount to a ***convexity*** requirement. In addition, we will require that in a normal form every subterm guarded by a particular event be identical and also in normal form. This means that every normal form is itself built up from normal forms in a simple way that facilitates proofs. Formally, these constraints are defined as follows.

*Normal forms.*

*Definition 1.3:* A subset $\mathcal{B}$ is ***convex*** iff it is non-empty and

$$\begin{array}{ll}\text{(i)} & A, B \in \mathcal{B} \Rightarrow A \cup B \in \mathcal{B}, \\ \text{(ii)} & A, C \in \mathcal{B} \ \& \ A \subseteq B \subseteq C \Rightarrow B \in \mathcal{B}.\end{array}$$

Note that a convex set is closed under (finite) unions as well as the convex containment relation (ii). We will write $\operatorname{con}(\mathcal{B})$ for the smallest convex set containing $\mathcal{B}$, and refer to the ***convex closure*** of $\mathcal{B}$.

There are clear connections between this form of convexity on sets of sets of events and the "saturated" condition of [HN], a fact which is not surprising in view of the close connections which can be found between their models and ours (see [B1,2] for example).

***Examples.***

*Example 1.* The set $\mathcal{A} = \{\emptyset, \{a, b\}\}$ is not convex, because

$$\emptyset \subseteq \{a\} \subseteq \{a, b\}$$

but $\{a\}$ is not in $\mathcal{A}$.

*Example 2.* The set $\mathcal{B} = \{\emptyset, \{a\}, \{b\}\}$ is not convex, because it does not contain $\{a, b\}$.

*Example 3.* The smallest convex set containing $\mathcal{A}$ and $\mathcal{B}$ is the set

$$\mathcal{C} = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}.$$

We have $\operatorname{con}(\mathcal{A}) = \operatorname{con}(\mathcal{B}) = \mathcal{C}$.

*Example 4.* For any set $B \subseteq \Sigma$ the powerset of $B$ is convex.

Now we can define normal form:

*Definition 1.4:* A term $P$ is in ***normal form*** iff it has the structure

$$\begin{array}{ll} \text{either } & P = \text{STOP} \\ \text{or} & P = \sqcap_{B\in\mathcal{B}} \Box_{b\in B}(b \to P_b) \end{array}$$

for some convex set $\mathcal{B}$, and each $P_b$ is also in normal form.

Note that although a normal form $P$ may have "disjuncts " $P_B$ and $P_C$ with some initials in common, say

$$P_B = \Box_{b\in B}(b \to P_b)$$
$$P_C = \Box_{c\in C}(c \to P_c)$$

the definition forces these two processes to have identical derivatives $P_a$ for all $a \in B \cap C$. Some examples will help.

*Examples.*

*Example 5.* $P = \text{STOP} \sqcap (a \to \text{STOP})$ is in normal form: here $\mathcal{B}$ is the convex set $\{\emptyset, \{a\}\}$ and $P_a = \text{STOP}$.

*Example 6.* $P = (a \to (b \to \text{STOP})) \sqcap ((a \to \text{STOP}) \Box (b \to \text{STOP}))$ is not in normal form, because the two subterms guarded by $a$ are distinct.

The next result is the basis of our completeness theorem.

*Lemma 1.5:* Any term $P$ can be transformed using the proof system into a normal form.

*Proof.* By induction on the length of the term.

The base case, when $P = \text{STOP}$, is trivial; the case when $P = (a \to Q)$ is also straightforward. In the remaining two cases, we must show that if $P$ and $Q$ are normal forms then $P \sqcap Q$ and $P \Box Q$ can be put into normal form. To this end, suppose the two normal forms are:

$$P = \sqcap_{B\in\mathcal{B}} \Box_{b\in B}(b \to P_b)$$
$$Q = \sqcap_{C\in\mathcal{C}} \Box_{c\in C}(c \to Q_c).$$

Write $P_B = \Box_{b\in B}(b \to P_b)$ and $Q_C = \Box_{c\in C}(c \to Q_c)$, so that

$$P = \sqcap_{B\in\mathcal{B}} P_B$$
$$Q = \sqcap_{C\in\mathcal{C}} Q_C.$$

Then it is easily provable that

$$P \sqcap Q \equiv \sqcap_{B\in\mathcal{B}} \sqcap_{C\in\mathcal{C}} P_B \sqcap Q_C.$$

Let $\mathcal{A} = \mathcal{B} \cup \mathcal{C}$, and define the terms $R_a$ for $a \in \mathcal{A}$ by:

$$\begin{array}{lll} R_a & = P_a & \text{if } a \in B{-}C, \\ & = Q_a & \text{if } a \in C{-}B, \\ & = P_a \sqcap Q_a & \text{if } a \in B \cap C. \end{array}$$

Using the obvious notation, it is clear that the statements

$$P_B \sqcap Q_C \equiv R_B \sqcap R_C$$

are provable, and hence that

$$\vdash\ P \sqcap Q \equiv \sqcap_{A \in \mathcal{A}} R_A.$$

To complete the proof in this case we use the convexity laws to replace $\mathcal{A}$ by its convex closure. The following identities are deducible from (D1)–(D3) and show that replacement of $\mathcal{A}$ by con($\mathcal{A}$) is valid here:

$$R_A \sqcap R_B \equiv R_A \sqcap R_B \sqcap R_{A \cup B},$$
$$R_A \sqcap R_C \equiv R_A \sqcap R_B \sqcap R_C, \qquad \text{if } A \subseteq B \subseteq C.$$

Finally we must reduce $P \Box Q$ to normal form. Again it is easy to show that

$$\vdash\ P \Box Q \equiv \sqcap_{B \in \mathcal{B}} \sqcap_{C \in \mathcal{C}} P_B \Box Q_C,$$

and (using the same notation as above) that

$$\vdash\ P_B \Box Q_C \equiv R_{B \cup C}.$$

It follows that

$$\vdash\ P \Box Q \equiv \sqcap_{A \in \mathcal{A}} \Box_{a \in A} (a \rightarrow R_a),$$

where $\mathcal{A} = \text{con}(\{\, B \cup C \mid B \in \mathcal{B}, C \in \mathcal{C} \,\})$. ∎

The following result states that every true statement about normal forms is provable.

*Lemma 1.6:* Given two normal forms $P^*$ and $Q^*$,

$$\mathcal{F}[\![P^*]\!] \supseteq \mathcal{F}[\![Q^*]\!] \Rightarrow \vdash\ P^* \sqsubseteq Q^*.$$

*Proof.* Let the two normal forms be

$$P^* = \sqcap_{B \in \mathcal{B}} \Box_{b \in B} (b \rightarrow P_b)$$
$$Q^* = \sqcap_{C \in \mathcal{C}} \Box_{c \in C} (c \rightarrow Q_c).$$

Write $P_B$ and $Q_C$ for the subterms:

$$P_B = \Box_{b \in B} (b \rightarrow P_b),$$
$$Q_C = \Box_{c \in C} (c \rightarrow Q_c).$$

Then $P^* = \sqcap_{B \in \mathcal{B}} P_B$ and $Q^* = \sqcap_{C \in \mathcal{C}} Q_C$.

By definition of normal form, the sets $\mathcal{B}$ and $\mathcal{C}$ are convex, and each $P_b$ and $Q_c$ is also in normal form. We will use an induction on the length of the normal forms. The base case, when both $P$ and $Q$ have zero length, is trivial; both terms are STOP. For the inductive step, we argue as follows. First we show that

$$\mathcal{F}[\![P^*]\!] \supseteq \mathcal{F}[\![Q^*]\!] \Rightarrow \mathcal{C} \subseteq \mathcal{B} \ \&\ \forall c \in C \in \mathcal{C}.\ \mathcal{F}[\![P_c]\!] \supseteq \mathcal{F}[\![Q_c]\!] \quad (1).$$

To this end, assume that $\mathcal{F}[\![P^*]\!] \supseteq \mathcal{F}[\![Q^*]\!]$. Let $B_0 = \bigcup \mathcal{B}$ and $C_0 = \bigcup \mathcal{C}$ be the initials of $P^*$ and $Q^*$. Then we know that

$$B_0 \supseteq C_0.$$

Since $P^*$ and $Q^*$ have unique $c$-derivatives $P_c$ and $Q_c$ respectively, for all $c \in B_0 \cap C_0$, we must have

$$\mathcal{F}[\![P_c]\!] \supseteq \mathcal{F}[\![Q_c]\!]$$

for all such $c$. All we need to show now is that $\mathcal{C} \subseteq \mathcal{B}$. If this does not hold, let $X = B_0 - C$. Then $(\langle\rangle, X)$ must be a failure of $P^*$. By hypothesis, this is also a failure of $Q^*$. But this happens only if there is a $B \in \mathcal{B}$ with

$$B \cap X = B \cap (B_0 - C) = \emptyset.$$

Equivalently, $B \subseteq C$. But $C \subseteq C_0 \subseteq B_0$, and the sets $B$ and $B_0$ belong to $\mathcal{B}$ ($B$ by assumption and $\mathcal{B}$ by convexity). Thus we find that $C \in \mathcal{B}$, contradicting our assumption. It must therefore be the case that $\mathcal{C} \subseteq \mathcal{B}$, as required. The truth of (1) has now been established.

Now the inductive hypothesis applied to the terms $P_c$ and $Q_c$ gives

$$\vdash P_c \sqsubseteq Q_c,$$

for all $c \in \bigcup \mathcal{C}$. This implies that, for each $C \in \mathcal{C}$,

$$\vdash P_C \sqsubseteq Q_C.$$

Then, since $\mathcal{C} \subseteq \mathcal{B}$, we may use (A10) and (M2) to show

$$\vdash P^* \sqsubseteq Q^*,$$

as required. That completes the proof. ∎

*Corollary 1.7:* For all terms $P$ and $Q$,

$$\mathcal{F}[\![P]\!] \supseteq \mathcal{F}[\![Q]\!] \Rightarrow \vdash P \sqsubseteq Q.$$

*Proof.* By Lemmas 1.5 and 1.6. ∎

## 2. Extending to infinite processes.

In this section we modify the language FCSP, adding process variables, recursion and a new constant $\perp$, which is intended to denote a process whose only capability is to diverge. Such a pathological process will turn out to correspond precisely to the terms in which a badly constructed recursion appears. We will be mainly interested in terms without free process variables, so-called *closed terms.* The semantics we use for this language is based on failure sets but has an extra component called a *divergence set* in order to allow us to distinguish between deadlock and divergence.

Let RCSP ("Recursive CSP") be the language generated by the following syntax:

$$P ::= \text{STOP} \mid (a \rightarrow P) \mid P \Box P \mid P \sqcap P \mid \perp \mid x \mid \mu x.P$$

where $a \in \Sigma$ and $x$ ranges over a set of process variables or *identifiers.*

Let **F** be the domain of failure sets, ordered by $\supseteq$ . Now we introduce **D**, the domain of *divergence sets,* which is just the powerset $\mathcal{P}(\Sigma^*)$, ordered also by $\supseteq$ . The semantics of terms in RCSP will be given via two semantic functions, one for failures and one for divergence. Since terms may contain occurrences of identifiers we will use an *environment* in the semantics, which binds each identifier to the failure set and divergence set it is intended to denote. Let Ide be the

Thus $\delta_n = a^n\Sigma^*$, for each $n$, and the intersection of these sets is empty: $\mathcal{D}[\![\mu x.(a \rightarrow x)]\!]u$ is the empty set.

*Example 8.* The recursion $\mu x.x$ is obviously not well-guarded. This process diverges on all traces:

$$\mathcal{D}[\![\mu x.x]\!]u = \Sigma^*.$$

*Example 9.* The term $\mu x.((a \rightarrow x)\Box(\mu y.y))$ is not well-guarded, because of the subterm $\mu y.y$. One can check that the divergence set of this term is $\Sigma^*$.

The semantic function for failures is also given by structural induction, and it makes use of $\mathcal{D}$. For the most part, the definition is exactly as in [HBR,B1,B2,R], but extended to make it consistent with the notion that divergence is catastrophic: when a process is diverging we can guarantee no aspect of its behaviour; thus we make the operations (except prefixing) ***strict***, so that a process constructed from divergent components can diverge too.

*Definition 2.2:* The failures semantic function is:

$$\mathcal{F} : \mathrm{RCSP} \rightarrow U \rightarrow \mathbf{F}$$

$$\mathcal{F}[\![\mathrm{STOP}]\!]u = \{ (\langle\rangle, X) \mid X \subseteq \Sigma \}$$

$$\mathcal{F}[\![\bot]\!]u = \Sigma^* \times p\Sigma$$

$$\mathcal{F}[\![a \rightarrow P]\!]u = \{ (\langle\rangle, X) \mid a \not\in X \} \cup \{ (as, X) \mid (s, X) \in \mathcal{F}[\![P]\!]u \}$$

$$\begin{aligned} \mathcal{F}[\![P \Box Q]\!]u &= \Sigma^* \times p\Sigma, \qquad\qquad \text{if } \langle\rangle \in \mathcal{D}[\![P \Box Q]\!]u, \\ \mathcal{F}[\![P \Box Q]\!]u &= \{ (\langle\rangle, X) \mid (\langle\rangle, X) \in \mathcal{F}[\![P]\!]u \cap \mathcal{F}[\![Q]\!]u \} \\ &\qquad \cup \{ (s, X) \mid s \neq \langle\rangle \ \& \ (s, X) \in \mathcal{F}[\![P]\!]u \cup \mathcal{F}[\![Q]\!]u \} \\ &\qquad\qquad \text{otherwise} \end{aligned}$$

$$\mathcal{F}[\![P \sqcap Q]\!]u = \mathcal{F}[\![P]\!]u \cup \mathcal{F}[\![Q]\!]u$$

$$\mathcal{F}[\![x]\!]u = (u[\![x]\!])_1$$

$$\mathcal{F}[\![\mu x.P]\!]u = \mathrm{fix}(\lambda\phi.\mathcal{F}[\![P]\!](u + [x \mapsto \phi]))$$

Again we can prove that all operations on failure sets used here are continuous, and therefore the least fixed point of any construction exists and is given by the limit:

$$\mathcal{F}[\![\mu x.P]\!]u = \bigcap_{n=0}^{\infty} \Phi_n,$$

$$\text{where} \quad \Phi_0 = \Sigma^* \times \mathcal{P}\Sigma,$$

$$\text{and} \quad \Phi_{n+1} = \mathcal{F}[\![P]\!](u + [x \mapsto \Phi_n]).$$

set of identifiers. Then the domain of environments is

$$U = \mathrm{Ide} \to (\mathbf{F} \times \mathbf{D}).$$

For an environment $u$ which maps identifiers to pairs, we will use the conventional notation $(u[\![x]\!])_1$ and $(u[\![x]\!])_2$ to refer to the components of pairs.

The semantic function $\mathcal{D}$ maps terms to divergence sets, relative to an environment. It is defined in the usual way, by structural induction:

*Definition 2.1:* The divergence semantic function is:

$$\begin{aligned} \mathcal{D} &: \mathrm{RCSP} \to U \to \mathbf{D} \ . \\ \mathcal{D}[\![\mathrm{STOP}]\!]u &= \emptyset \\ \mathcal{D}[\![a \to P]\!]u &= \{\, as \mid s \in \mathcal{D}[\![P]\!]u \,\} \\ \mathcal{D}[\![P \,\Box\, Q]\!]u &= \mathcal{D}[\![P]\!]u \cup \mathcal{D}[\![Q]\!]u \\ \mathcal{D}[\![P \sqcap Q]\!]u &= \mathcal{D}[\![P]\!]u \cup \mathcal{D}[\![Q]\!]u \\ \mathcal{D}[\![\perp]\!]u &= \Sigma^* \\ \mathcal{D}[\![x]\!]u &= (u[\![x]\!])_2 \\ \mathcal{D}[\![\mu x.P]\!]u &= \mathrm{fix}(\lambda\delta.\mathcal{D}[\![P]\!](u + [x \mapsto \delta])). \end{aligned}$$

It is easy to see that all of the operations induced on divergence sets by the above definitions are continuous with respect to the superset ordering and hence that the fixed point used in the semantics of recursion will always exists. The usual fixpoint characterisation as a limit is expressed in:

$$\begin{aligned} \mathcal{D}[\![\mu x.P]\!]u &= \bigcap_{n=0}^{\infty} \delta_n, \\ \text{where} \quad \delta_0 &= \Sigma^* = \mathcal{D}[\![\perp]\!]u, \\ \text{and} \quad \delta_{n+1} &= \mathcal{D}[\![P]\!](u + [x \mapsto \delta_n]) \quad \text{for } n \geq 0. \end{aligned}$$

Notice also that the only terms with a non-trivial divergence set are those with a subterm $\perp$ or with an *unguarded* recursion, *i.e.* a subterm of the form $\mu x.P$ in which there is an occurrence of $x$ appearing in $P$ without a guard. This fact could be proved by a structural induction, once we have defined rigorously the notion of well-guardedness. Finally, our definition guarantees that whenever a particular trace $s$ belongs to a divergence set then all extensions of that trace are also included:

$$s \in \mathcal{D}[\![P]\!]u \Rightarrow st \in \mathcal{D}[\![P]\!]u, \quad \text{for all } t.$$

*Examples.*

*Example 7.* The recursion $\mu x.(a \to x)$ is guarded. Applying the previous definition, we have

$$\begin{aligned} \mathcal{D}[\![\mu x.(a \to x)]\!]u &= \cap_{n=0}^{\infty}\delta_n, \\ \text{where} \quad \delta_0 &= \Sigma^*, \\ \text{and, for each } n, \quad \delta_{n+1} &= \{\, as \mid s \in \delta_n \,\}. \end{aligned}$$

*Example.*

*Example 10.* If the alphabet $\Sigma$ is finite, then the term

$$P = \mu x.\, \sqcap_{B \subseteq \Sigma} (\Box_{b \in B}(b \to x))$$

is expressible in RCSP. This denotes a process which never diverges, but which can perform or refuse to perform any sequence of events. This is the same process as was called CHAOS in [HBR]. We have, for this term $P$,

$$\mathcal{F}[\![P]\!]u = \Sigma^* \times p\Sigma,$$
$$\mathcal{D}[\![P]\!]u = \emptyset.$$

We say that $P$ may diverge on $s$ if $s$ is a trace in the divergence set of $P$. Notice that we have defined the semantics of terms in such a way that the following conditions hold:

$$\begin{array}{ll} \text{(i)} & s \in \mathcal{D}[\![P]\!]u \Rightarrow \forall t, X.(st, X) \in \mathcal{F}[\![P]\!]u. \\ \text{(ii)} & s \in \mathcal{D}[\![P]\!]u \Rightarrow \forall t.st \in \mathcal{D}[\![P]\!]u. \end{array}$$

Intuitively, (i) says that a divergent process is totally unpredictable: we cannot be sure that it will or will not ever stop diverging and allow some sequence of actions. Condition (ii) says that once a process starts to diverge it cannot "recover" by performing a visible action: divergence continues forever. Thus, a pair $(\Phi, \delta)$ is a reasonable model for a process iff the following conditions hold:

$$\begin{array}{lrl} (1) & \multicolumn{2}{c}{\text{dom}(\Phi) \text{ is non-empty and prefix-closed}} \\ (2) & (s, X) \in \Phi,\ Y \subseteq X & \Rightarrow (s, Y) \in \Phi \\ (3) & (s, X) \in \Phi,\ (sa, \emptyset) \not\in \Phi & \Rightarrow (s, X \cup \{a\}) \in \Phi \\ (4) & s \in \delta & \Rightarrow st \in \delta, \quad \text{for all } t \\ (5) & s \in \delta & \Rightarrow (st, X) \in \Phi, \quad \text{for all } t, X. \end{array}$$

This more general model of processes is thus seen to be derived from the old failures model by adding divergence sets and requiring a kind of *consistency* between failures and divergences. Indeed, the processes with empty divergence sets form a space isomorphic to the failures model. Notice that the limit of a directed set of pairs $(\Phi_i, \delta_i)$ is the intersection and the greatest lower bound of a non-empty set of pairs is again the union. As with the set of failures, the new model forms a complete semi-lattice with respect to the (pairwise) superset ordering. We will write $P \sqsubseteq Q$ to mean that the failures of $P$ contain those of $Q$ and the divergence set of $P$ contains the divergence set of $Q$. All of the operations considered in this section are continuous with respect to this ordering. This fact justifies our use of fixpoints in the semantics of recursively defined processes.

*Syntactic approximation.*

Before we introduce a complete axiom system for the new model, we will need some important results which allow us to reason about a (possibly) infinite process in terms of its (finite) approximations. Beginning with the standard definition of *syntactic approximation* on terms, we define the set of finite approximants of an arbitrary term and show that the semantics of any term is uniquely determined from the semantics of its finite approximants.

The notion of *syntactic approximation* on terms is well known (see, for example, [Gu]). The following presentation is typical of the general style.

*Definition 2.3:* The relation $\prec$ on terms is the smallest relation satisfying:

(i) $\perp \prec P$

(ii) $P \prec P$

(iii) $P \prec Q \prec R \Rightarrow P \prec R$

(iv) $P \prec Q \Rightarrow (a \rightarrow P) \prec (a \rightarrow Q)$

(v) $P_1 \prec Q_1,\ P_2 \prec Q_2 \Rightarrow P_1 \Box P_2 \prec Q_1 \Box Q_2$

(vi) $P_1 \prec Q_1,\ P_2 \prec Q_2 \Rightarrow P_1 \sqcap Q_1 \prec P_2 \sqcap Q_2$

(vii) $P[(\mu x.P) \setminus x] \prec \mu x.P$

We have used the notation $P[Q \setminus x]$ to denote the result of replacing every free occurrence of $x$ in $P$ by $Q$, taking care to avoid name clashes.

If $P \prec Q$ we say that $P$ *approximates* $Q$. An easy structural induction shows that for all $P$ and $Q$ syntactic approximation implies semantic approximation:

$$P \prec Q \Rightarrow P \sqsubseteq Q.$$

A term $P$ is *finite* iff it does not contain any subterm of the form $\mu x.Q$. For any term $P$, the set of finite approximants is

$$\mathrm{FIN}(P) = \{\, Q \mid Q \prec P \ \&\ Q \text{ is finite} \,\}.$$

It should be noted that FIN($P$) is *directed* with respect to $\prec$ .

The common notion of *unrolling* or *unwinding* a recursive term is intimately connected with finite approximation. The result of unrolling the term $P$ $n$ times will be denoted $P^{(n)}$. The formal definition is:

(i) $P^{(0)} = \perp$

(ii) $\mathrm{STOP}^{(n+1)} = \mathrm{STOP}$

(iii) $(a \rightarrow P)^{(n+1)} = (a \rightarrow P^{(n+1)})$

(iv) $(P \Box Q)^{(n+1)} = P^{(n+1)} \Box Q^{(n+1)}$

(v) $(P \sqcap Q)^{(n+1)} = P^{(n+1)} \sqcap Q^{(n+1)}$

(vi) $x^{(n+1)} = x$

(vii) $(\mu x.P)^{(n+1)} = P[(\mu x.P)^{(n)} \setminus x].$

Every finite approximation to a term $P$ is also a finite approximation to some unrolling of that term:

*Lemma 2.4:* If $Q \in \mathrm{FIN}(P)$ then there is an $n$ such that $Q \prec P^{(n)}$.

*Proof.* See [Gu]. ∎

*Corollary:* For all $P$, $\mathrm{FIN}(P) = \bigcup_{n=0}^{\infty} \mathrm{FIN}(P^{(n)})$.

*Lemma 2.5:*

$$\begin{array}{ll}
\text{(i)} & \mathrm{FIN}(\bot) = \{\bot\} \\
\text{(ii)} & \mathrm{FIN}(\mathrm{STOP}) = \{\bot, \mathrm{STOP}\} \\
\text{(iii)} & \mathrm{FIN}(P \,\Box\, Q) = \{\bot\} \cup \{P' \,\Box\, Q' \mid P' \in \mathrm{FIN}(P) \;\&\; Q' \in \mathrm{FIN}(Q)\} \\
\text{(iv)} & \mathrm{FIN}(P \sqcap Q) = \{\bot\} \cup \{P' \sqcap Q' \mid P' \in \mathrm{FIN}(P) \;\&\; Q' \in \mathrm{FIN}(Q)\} \\
\text{(v)} & \mathrm{FIN}(a \to P) = \{\bot\} \cup \{(a \to P') \mid P' \in \mathrm{FIN}(P)\} \\
\text{(vi)} & \mathrm{FIN}(x) = \{\bot, x\}.
\end{array}$$

*Proof.* Elementary. ▮

In a sense, a term $P$ is the "syntactic limit" of its finite approximation set FIN($P$). Recall that this set is directed with respect to the syntactic relation $\prec$, and therefore the semantic images of the finite approximations to $P$ form a directed set with respect to the semantic order $\sqsubseteq$ . The following results show that the semantics of a term is uniquely determined by the semantics of its finite approximations, and allow us to deduce that the semantics of a term $P$ is in fact the limit of the semantics of its finite approximations. We omit proofs, as they follow standard lines. More details can be found in [B1].

*Lemma 2.6:* If $P$ is finite and $P \sqsubseteq Q$, then there is a finite approximation $R$ of $Q$ such that $P \sqsubseteq R$.

*Theorem 2.7:* For all $P$ and $u$,

$$\mathcal{D}[\![P]\!]u = \bigcap\{\, \mathcal{D}[\![Q]\!]u \mid Q \in \mathrm{FIN}(P)\,\}.$$

*Proof.* By structural induction on $P$. ▮

*Theorem 2.8:* For all $P$ and $u$,

$$\mathcal{F}[\![P]\!]u = \bigcap\{\, \mathcal{F}[\![Q]\!]u \mid Q \in \mathrm{FIN}(P)\,\}.$$

*Proof.* By structural induction. ▮

***Proof system.***

All of the axioms and inference rules of Table 1 are still valid. Let $\mathcal{L}$ be the proof system containing all axioms and rules of the earlier system together with the following additions:

$$\begin{array}{ll}
\text{(B1)} & P \,\Box\, \bot = \bot \\
\text{(B2)} & P \sqcap \bot = \bot \\
\text{(B3)} & \bot \sqsubseteq P \\
\text{(B4)} & P[(\mu x.P) \setminus x] \sqsubseteq \mu x.P
\end{array}$$

$$\text{(R)} \qquad \frac{\forall Q \in \mathrm{FIN}(P).\ Q \sqsubseteq R}{P \sqsubseteq R}$$

The new axioms state that the two conditional combinators are *strict,* and that $\perp$ is the bottom element with respect to $\sqsubseteq$ . The new inference rule essentially says that any property of a term is deducible from the properties of its finite approximations. This is an *infinitary rule,* because a term may have an infinite set of syntactic approximants; in such a case one would need an infinite number of premises in order to use rule (R). It seems unlikely that a finitary proof system could be found which was still complete, although some interesting sublanguages (in which use of recursion is constrained) will presumably have decidable proof systems. This remains a topic for future work. Now we are concerned with the soundness and completeness of our enlarged proof system.

*Soundness.*

Under the interpretation that for closed terms $P$ and $Q$, $P \sqsubseteq Q$ means

$$\mathcal{F}[\![P]\!]u \supseteq \mathcal{F}[\![Q]\!]u \ \& \ \mathcal{D}[\![P]\!]u \supseteq \mathcal{D}[\![Q]\!]u$$

for all environments $u$, the proof system $\mathcal{L}$ is sound. We need merely to check that the axioms are valid and the proof rules sound. Since the semantics was defined to make the conditional operators strict, the new axioms are clearly valid. Soundness of rule (R) follows from Theorems 2.7 and 2.8. It is easy to check validity of the old axioms and rules. Thus we have:

*Theorem 2.9:* For all closed terms $P$ and $Q$, and all $u$,

$$\vdash_{\mathcal{L}} P \sqsubseteq Q \Rightarrow P \sqsubseteq Q.$$

*Completeness.*

In order to establish that the new proof system is complete, we must first modify the definition of normal form. Essentially, we just allow $\perp$ as well as STOP in building up normal forms.

*Definition 2.10:* A term $P$ in RCSP is in normal form iff it has the structure:

either $P = \text{STOP}$,
or $P = \perp$,
or $P = \sqcap_{B\in\mathcal{B}}\square_{b\in B}(b \rightarrow P_b)$

where $\mathcal{B}$ is convex and each $P_b$ is in normal form.

It is easy to modify the proof of Lemmas 1.5 and 1.6 to show that any *finite* term can be reduced to normal form using the axioms and rules, and that whenever $P$ and $Q$ are normal forms

$$P \sqsubseteq Q \Rightarrow \vdash_{\mathcal{L}} P \sqsubseteq Q.$$

The completeness theorem relies on Lemma 2.6, which states that whenever $P$ is finite and $P \sqsubseteq Q$ there is a finite term $R \in \text{FIN}(Q)$ such that $P \sqsubseteq R$.

*Theorem 2.11:* For all terms $P$ and $Q$,

$$P \sqsubseteq Q \Rightarrow \vdash_{\mathcal{L}} P \sqsubseteq Q.$$

*Proof.* Let $P'$ be a finite approximation to $P$ and suppose $P \sqsubseteq Q$. Then

$$P' \sqsubseteq P \sqsubseteq Q.$$

By Lemma 2.6 there is a finite approximation $Q'$ to $Q$ such that $P' \sqsubseteq Q'$. But then

$$\vdash P' \sqsubseteq Q'.$$

Since for every $Q' \in \mathrm{FIN}(Q)$ the formula $Q' \sqsubseteq Q$ is provable, we have

$$\vdash P' \sqsubseteq Q.$$

The result follows by an application of rule (R). ∎

## 3. Adding more CSP operations.

We may extend the proof system to encompass other CSP operations provided we add enough axioms and rules to allow normal form reductions. We must introduce failure sets and divergence sets for the new forms of processes, by extending the definition of $\mathcal{D}$ and $\mathcal{F}$ accordingly. We must also add axioms and inference rules corresponding to these definitions, in such a way that Theorems 2.7 and 2.8 still hold. This will keep the proof system complete and consistent.

In keeping with the notion that a divergent process is totally unpredictable, and that divergence of a component process should also give rise to divergence of the compound process (so that a process built from divergent components diverges) we stipulate that all operations should be ***strict***, in that they map $\perp$ to $\perp$. Now we consider extending the proof system and the semantic defnitions to include parallel composition, interleaving, and hiding. It should be clear how to include the other operations of [HBR], with these examples as illustration of the general method. Essentially, we make each operation strict, and include axioms for strictness and for distribution over $\sqcap$ and guarded terms.

### *Parallel composition.*

For the parallel composition $P||Q$, for example, we require divergence when either $P$ or $Q$ diverges. This combination performs an event only if both component processes perform it, and can refuse an event if either component can refuse it; and thus we specify:

(i) $\mathcal{D}[\![P||Q]\!]u = \{\, st \mid s \in (\mathcal{D}[\![P]\!]u \cup \mathcal{D}[\![Q]\!]u) \cap (\mathrm{traces}(P) \cap \mathrm{traces}(Q))\,\}$

(ii) $\mathcal{F}[\![P||Q]\!]u = \{\,(s, X \cup Y) \mid (s,X) \in \mathcal{F}[\![P]\!]u \;\&\; (s,Y) \in \mathcal{F}[\![Q]\!]u\,\}$
$\{\,(st, X) \mid s \in \mathcal{D}[\![P||Q]\!]u\,\}.$

It should be evident that the semantic definition captures the intuition stated above. Again we should check that our definition does yield a divergence set and failure set satisfying the conditions (1)–(5) of page 12. The details are left as an exercise.

Extending the syntactic approximation relation in the obvious way, we add the clause

$$P_1 \prec P_2,\ Q_1 \prec Q_2 \ \Rightarrow P_1||Q_1 \prec P_2||Q_2$$

to Definition 2.3. Then the finite approximations of $P\|Q$ are built up as parallel compositions of finite approximations of $P$ and $Q$ :

$$\mathrm{FIN}(P\|Q) = \{\perp\} \cup \{P'\|Q' \mid P' \in \mathrm{FIN}(P) \ \&\ Q' \in \mathrm{FIN}(Q)\}.$$

It is clear from this that Theorems 2.7 and 2.8 still hold.

We add axioms for strictness and manipulation of normal forms. In each case the axiom is either a restatement of an earlier result which clearly still holds in the extended model, or is self-evident. In the axioms we adopt the convention that a term $P_B$ stands for

$$\Box_{b\in B}(b \rightarrow P_b).$$

(PAR 0) $\quad P\|\perp \equiv \perp$

(PAR 1) $\quad P\|(Q \sqcap R) \equiv (P\|Q) \sqcap (P\|R)$

(PAR 2) $\quad P_B\|Q_C \equiv \Box_{a\in B\cap C}(a \rightarrow P_a\|Q_a).$

It is easy to check that these axioms enable any parallel composition of normal forms to be reduced to normal form. Note also the special case of (PAR 2) when $C = \emptyset$ : the axiom reduces in this case to the identity $P_B\|\mathrm{STOP} = \mathrm{STOP}$.

*Interleaving.*

For the interleaving operation $P|||Q$ we want divergence when either component process can diverge. And at any stage a trace of $P|||Q$ is to be an interleaving of a trace of $P$ with a trace of $Q$, and the process can refuse an event only if both components refuse it.

(i) $\mathcal{D}[\![P|||Q]\!]u = \mathrm{merge}(\mathcal{D}[\![P]\!]u, \mathrm{traces}(Q)) \cup \mathrm{merge}(\mathrm{traces}(P), \mathcal{D}[\![Q]\!]u)$

(ii) $\mathcal{F}[\![P|||Q]\!]u = \{(u,X) \mid \exists s,t.\,(s,X) \in \mathcal{F}[\![P]\!]u \ \&\ (t,X) \in \mathcal{F}[\![Q]\!]u \ \&\ u \in \mathrm{merge}(s,t)\}$
$\qquad \cup \{(st,X) \mid s \in \mathcal{D}[\![P|||Q]\!]u\}.$

Here we have used the merge function on traces and its natural extension to sets of traces. It can be defined inductively as follows:

$$\mathrm{merge}(\langle\rangle, t) = \mathrm{merge}(t, \langle\rangle) = \{t\}$$
$$\mathrm{merge}(as, bt) = \mathrm{merge}(bt, as) = \{au, bv \mid u \in \mathrm{merge}(s, bt),\ v \in \mathrm{merge}(as, t)\}.$$

For syntactic approximation we add to Definition 2.3:

$$P_1 \prec P_2,\ Q_1 \prec Q_2 \Rightarrow P_1|||Q_1 \prec P_2|||Q_2.$$

Again the finite approximations of an interleaved process are formed by interleaving finite approximations to the components:

$$\mathrm{FIN}(P|||Q) = \{\perp\} \cup \{P'|||Q' \mid P' \in \mathrm{FIN}(P) \ \&\ Q' \in \mathrm{FIN}(Q)\}.$$

Again Theorems 2.4 and 2.5 are still true.

Our definition yields a strict operation, since $\mathcal{D}[\![\bot \,|||\, Q]\!]u = \Sigma^*$. Otherwise it has similar properties to the interleaving operation of [HBR]. We add axioms:

(INT 0) $\quad P|||\bot \equiv \bot$

(INT 1) $\quad P|||(Q \sqcap R) \equiv (P \sqcap Q)|||(P \sqcap R)$

(INT 2) $\quad P_B|||Q_C \equiv (\Box_{b\in B}(b \to (P_b|||Q_C)))\,\Box\,(\Box_{c\in C}(c \to (P_B|||Q_c)))$.

Again it is easy to check the validity of these axioms, and to verify that an interleaving of two normal forms can be reduced to normal form. In particular, the special case of (INT 2) when $C$ is empty is to be interpreted as the identity: $P_B|||\mathrm{STOP} \equiv P_B$.

*Hiding.*

For the hiding operator, we have to model the fact that hiding a potentially infinite sequence of actions produces divergence: we are identifying the phenomenon of infinite internal chatter with divergence. This version of hiding is closely related to the second form of hiding introduced in [B] and in [HBR], where infinite chatter was identified with CHAOS; here a process which is chattering has the same failure set as CHAOS, but (unlike CHAOS) can also diverge. It is simple to alter the proofs given in [HBR] for the chaotic version of hiding, to show that this form enjoys similar properties, such as continuity.

(i) $\quad \mathcal{D}[\![P/b]\!]u = \{\, (s\backslash b)t \mid s \in \mathcal{D}[\![P]\!]u \,\} \cup \{\, (s\backslash b)t \mid \forall n.\ sb^n \in \mathrm{traces}(P) \,\}$.

(ii) $\quad \mathcal{F}[\![P/b]\!]u = \{\, (s\backslash b, X) \mid (s, X \cup \{\, b \,\}) \in \mathcal{F}[\![P]\!]u \,\} \cup \{\, (st, X) \mid s \in \mathcal{D}[\![P/b]\!]u \,\}$.

For finite approximations, we again add to Definition 2.3:

$$P' \prec P \Rightarrow P'/b \prec P/b.$$

The finite approximations to a process formed by hiding are again formed by hiding:

$$\mathrm{FIN}(P/b) = \{\, \bot \,\} \cup \{\, P'/b \mid P' \in \mathrm{FIN}(P) \,\}.$$

Our new hiding operator is strict. We add axioms:

(HIDE 0) $\quad \bot/b \equiv \bot$

(HIDE 1) $\quad (P \sqcap Q)/b \equiv (P/b) \sqcap (Q/b)$

(HIDE 2) $\quad (b \to P)/c \equiv (b \to P/c) \quad$ if $b \neq c$,
$\qquad\qquad\qquad\qquad \equiv P/c \quad$ if $b = c$.

(HIDE 3) $\quad P_B/c \equiv \Box_{b\in B}((b \to P_b)/c), \quad$ if $c \not\in B$,
$\qquad\qquad\qquad \equiv (P_c/c) \sqcap \Box_{b\in B}((b \to P_b)/c) \quad$ if $b \in C$.

Again the validity of these axioms is easy to check, and one can use the axioms to produce normal forms.

*Examples.*

*Example 11.* The term $P = \mu x.(a \to x)$ has finite approximations

$$P_n = (a^n \to \bot), \qquad \text{for all } n,$$

using the obvious abbreviations. Thus, the term $P/a$ has finite approximations

$$\bot, \text{ and } P_n/a = (a^n \to \bot)/a,$$

for all $n$. Using (HIDE 2) we see that, for each $n$,

$$\vdash (a^n \to \bot)/a \equiv \bot/a,$$

and so, by (HIDE 1) every finite approximation to $P/a$ is provably equivalent to $\bot$. By rule (R), it follows that $P/a$ is equivalent to $\bot$, as expected because $P/a$ diverges.

*Example 12.* A slightly more complicated argument shows that for the term

$$Q = \mu x.((a \to x)\,\Box\,(b \to \text{STOP}))$$

$Q/a$ is also equivalent to $\bot$. One can also show that $Q/b$ is equivalent to $\mu x.(a \to x)$.

## 4. Conclusions.

We have introduced a semantics for processes based on the concepts of failures and divergence. The semantic mapping from terms to meanings has been described in the denotational style, in which the denotation of a complex term is built up from the meanings of its parts. In addition, we gave a set of axioms and proof rules characterising this semantics in the sense that two terms in the language of processes denote identical values (have the same meaning) if and only if this fact is provable within the formal system. The proof system contained an infinitary axiom to the effect that the semantics of an arbitrary term is determined uniquely by its syntactically finite approximations. It does not appear true that a finitary (and decidable) proof system exists for the language including recursion, although we do not investigate this issue here. Our axiomatic presentation demonstrates that the denotational semantics can be characterised by means of algebraic relations between processes, a fact of interest in itself. An attempt to treat the failures model (without divergence) in a similar way is reported in [N].

We have not tried in this paper to apply this proof system to problems involving large processes. It is certainly possible to represent some interesting behavioural properties of processes within our framework. For instance, the potential for deadlocking after performing a sequence of actions $s$ in an environment represented by $Q$ would correspond to an assertion of the form

$$P \| (s \to Q) \sqsubseteq (s \to \text{STOP}),$$

and the inability to refuse any event in the sequence $s$ would be represented by

$$P \| (s \to \text{STOP}) \equiv (s \to \text{STOP}).$$

The possibility of divergence after performing $s$ is captured by the assertion

$$P \| (s \to Q) \equiv (s \to \perp).$$

It remains to be seen how useful our proof system is in helping to formalise proofs for complex processes. Nevertheless, it is clear that the semantics given here both denotationally and axiomatically can serve as the foundation of a theory of processes, and can be used to justify reasoning about the behaviour of processes as in [R,B1,HBR].

It would be very interesting to extend our work to cover a language more directly derived from Hoare's original CSP, for which there are existing Hoare-style axiom systems. In its original formulation, CSP processes were able to perform essentially two different kinds of event: communication with another process, and assignment to a local variable. We have not specified the nature of events in our model, but one can certainly specialize the model to cases where the events are of particular forms. It is to be hoped that such an approach would help to bridge the gap between abstract languages (such as FCSP and RCSP) and their more concrete counterparts (CSP). In doing so, we would hope to shed some light on the existing proof systems for CSP [AFR,LG], and on the relationships between them.

## 5. Acknowledgements.

I am grateful to many people for advice and encouragement during the development of this work, which is based on part of my Ph.D. thesis. Particular thanks are due to Prof. C.A.R. Hoare, the author's thesis advisor; and to Bill Roscoe, who commented in detail and suggested some improvements and corrections. I have also been influenced by the work of Robin Milner, Matthew Hennessy and Rocco de Nicola.

## 6. References.

[AFR] Apt, K.R., Francez, N., and de Roever, W.P., *A Proof System for Communicating Sequential Processes,* ACM Transactions on Programming Languages and Systems, Vol 2. No. 3 (July 1980).

[B1] Brookes, S.D., *A Model for Communicating Sequential Processes,* Ph.D thesis, Oxford University (submitted 1983).

[B2] Brookes, S.D., *On the Relationship of CCS and CSP,* CMU Technical Report CMU-CS-83-111, also to appear in Proceedings of ICALP 1983 (pub. Springer).

[Gu] Guessarian, I., *Algebraic Semantics,* Springer-Verlag Lecture Notes in Computer Science Vol. 99 (1981).

[H1] Hoare, C.A.R., *Communicating Sequential Processes,* Communications of the ACM (August 1978).

[H2] Hoare, C.A.R., *A Model for Communicating Sequential Processes,* Technical Report PRG-22, Oxford University Computing Laboratory, Programming Research Group (1981).

[HBR] Hoare, C.A.R., Brookes, S.D., and Roscoe, A.W., *A Theory of Communicating Sequential Processes,* Technical Report PRG-16, Oxford University Computing Laboratory, Programming Research Group (May 1981). (an extended version will appear in JACM)

[HN] Hennessy, M.C.B., and de Nicola, R., *Testing Equivalences for Processes,* to appear in Proceedings of ICALP 1983.

[LG] Levin, G.M., and Gries, D., *A Proof Technique for Communicating Sequential Processes,* Acta Informatica 15 (1981).

[LNS] Lassez, J.-L., Nguyen, V.L., and Sonenberg, E.A., *Fixed Point Theorems and Semantics: A Folk Tale,* Information Processing Letters, Vol. 14 No. 3, May 1982.

[M] Milner, R., *A Calculus of Communicating Systems,* Springer-Verlag Lecture Notes in Computer Science Vol. 92 (1980).

[N] de Nicola, R., *A Complete Set of Axioms for a Theory of Communicating Sequential Processes,* Department of Computer Science Technical Monograph, University of Edinburgh (1983).

[R] Roscoe, A.W., *A Mathematical Theory of Communicating Processes,* Ph. D. thesis, Oxford University (1982).