CCS with Broadcast Stability

Iain Phillips*

Department of Computing

Imperial College of Science, Technology and Medicine

January 17, 1994

Abstract

We show how adding a new action to Milner's CCS which detects stability (a form of deadlock) allows us to express refusal tests [Phi87] within the system. We explore the close connections between stability and priority. The augmented calculus is given a bisimulation semantics and axiomatized. We prove that the "external" notion of refusal equivalence coincides with the "internal" De Nicola & Hennessy-style testing equivalence for the extended system. Furthermore for strongly convergent processes the associated congruence actually coincides with trace congruence.

1 Introduction

In [Phi87] it was suggested that one could allow observations of "black box" processes to include actions being refused as well as accepted. It was shown that using refusal tests incorporating these observations one obtains a testing equivalence (refusal equivalence) which for CCS processes is strictly finer than De Nicola & Hennessy's testing equivalence [DNH84].

These two equivalences are different in character. Refusal equivalence is an *external* notion. The same tests could be applied to a number of different process calculi - all that would be needed is for the processes to have an operational semantics as a transition system. On the other hand, De Nicola & Hennessy testing equivalence is an *internal* notion. The tests depend on the calculus, since they are just processes with an extra action to represent success. Of course their idea of turning processes into tests can clearly be applied to other process calculi.

*Supported by SERC grant GR/F72475

It is natural to wonder whether we can vary CCS in such a way as to admit refusal tests as processes and recover refusal equivalence as a De Nicola & Hennessy-style testing equivalence.

We suggest how this might be done. Our solution involves adding an extra action to CCS, different in character from the usual visible and invisible actions. The operators and calculus are unchanged, except as they relate to the new action; this can only be performed when the system is stable and behaves differently in parallel compositions, being broadcast rather than participating in handshaking communication as in CCS.

The idea of having an action which tests the system for stability has also been used by Langerak [**?**] in the context of LOTOS. However he does not add the action to the language itself, so that he does not get an internal testing equivalence. Nicollin, Richier, Sifakis & Voiron [NRSV90] and Hennessy & Regan [HR91] have used a special action to represent the tick of a clock in discretely timed process algebra, such a tick only being allowed to occur when all internal actions have occurred (so-called "maximal progress"). So a stability action has been used to model delay. Hennessy & Regan define a kind of testing equivalence for their language TPL. Stability is also relevant to the issue of priority, which has been studied by a number of authors in the context of process algebra (see for instance [BBK85], [CH88], [CW91]). The stability action has lower priority than internal actions.

2 Motivation

Consider the following example:

$$p = ab + ac$$

$$q = a(b + c)$$

p and q are trace equivalent. However Milner [Mil80] has argued that they should be distinguished, since after q has performed a it will always accept b, whereas p cannot be guaranteed to do so. In the formalism of De Nicola & Hennessy testing this may be expressed:

$$q \ \textbf{must satisfy} \ \bar{a}\bar{b}\omega$$

but not

$$p \ \textbf{must satisfy} \ \bar{a}\bar{b}\omega$$

Here ω is a special action which represents success. Another way of expressing the difference is that after a, p might refuse b, whereas q cannot do so. In the formalism of refusal testing:

$$p \ \textbf{may} \ a\tilde{b}\textsc{succ}$$

$$q \ \not\!\textbf{may} \ a\tilde{b}\textsc{succ}$$

Now let us make the test $a\tilde{b}\textsc{succ}$ into a process. Its transition diagram will be something like

$$\bar{a}(\bar{b} + \tilde{b}\omega)$$

Here we borrow some of the De Nicola & Hennessy formalism, but we have a new kind of transition $\tilde{b}$ ("b is refused"). We take the view that trying to observe a refusal of b necessitates requesting b, explaining the b arc. We have argued in [Phi87] that a test should only perform a $\tilde{b}$ transition if the process being tested cannot accept b and is *stable*, that is, it can neither perform a silent action nor diverge. But performing a b would produce a τ from the communication with the $\tilde{b}$ of the test. So we can say that the $\tilde{b}$ should only take place if no τ is possible. But then $\tilde{b}$ clearly does not depend on b. Let us therefore call it σ (short for "stable").

So now the question is: what kind of an action is σ?

One possibility is to make it a new silent action. This would accord with the view that σ is a spontaneous action on the part of the observer rather than a communication with the process. It would clearly have to be different from τ, since it can only happen when no τ is possible. We can take it to be a silent action with lower priority than τ, recalling the notion of priority (there between visible actions) of [BBK87]. However it is possible to get a sort of "deadlock" when a system is stable except that two observers are trying to observe a refusal and neither can proceed because they have to wait for each other's σ. After all if σ is internal a process which might perform it cannot be deemed stable. A calculus can be worked out along these lines, but it has the drawback that it does not correspond exactly to refusal testing, since in [Phi87] we took the view that the fact that an observer was waiting for a process to stabilise should not inhibit the process from performing internal actions, whereas here the test might deadlock with the process.

Another possibility is to make σ into a visible handshaking communication, so that σ and $\bar{\sigma}$ together produce τ in the usual CCS fashion. This would correspond to the view that observing stability is a two-sided affair — the process giving a signal (Milner [Mil81] suggested that the black box might switch off a green light) and the test observing it. Here we again have difficulties if there is more than one observer — either this leads to livelock as above or one excludes the other leading to a loss of either commutativity or associativity of the parallel composition operator.

We therefore take σ to be a new kind of "broadcast" action. If a system of processes/observers is stable then any observers who are interested (can perform σ) must observe the fact simultaneously. It is debatable whether σ is visible or not. It is visible in that the observer can synchronise on it with a process. However we take the view that it is invisible: Firstly the process can perform σ without the observer being aware of the fact. Secondly although the observer can synchronise with a process on σ, the observer cannot tell whether he is performing σ on his own, or together with the process. Thus

$$p \stackrel{\tau^*\sigma}{\rightarrow} \ \textbf{implies}\ p\ \textbf{may}\ \sigma\omega$$

but not conversely.

To clarify: Our view is that stability is observable (to observers equipped with the right capability). However the act of observing stability is not observable.

Broadcast communication is used in CSP [Hoa85], where processes are required to synchronize on events in their common alphabet, but our communication discipline is more reminiscent of Pnueli's operator $\bowtie$ [Pnu85]. Broadcasting is also studied in [Pra91].

Returning to our example, the test is now $t = \bar{a}(\bar{b} + \sigma\omega)$. We can see that p may pass this test since the following is a computation within our augmented calculus:

$$t|p \xrightarrow{\tau} (\bar{b} + \sigma\omega)|c \xrightarrow{\sigma} \omega|c$$

On the other hand, q cannot pass t.

Remarks. (1) We note that the preceding shows that $t|p$ can perform ω (after σ), whereas $t|q$ cannot. Thus although p, q are trace equivalent, this is no longer the case when they are placed in the context $t|\cdot$. We shall return to this in Section 5.

(2) We have two invisible actions, τ and σ. However the usual argument that any two invisible actions can be identified does not hold, because of the different communication disciplines. An example:

$$\tau + b \ \mathbf{may} \ \bar{b} + \sigma\omega$$

$$\sigma + b \ \mathbf{m\!\!/ay} \ \bar{b} + \sigma\omega$$

3 The system CCSσ

In this section we first describe the process language CCSσ informally. We then give the precise syntax and transitional semantics and establish that it is conservative over CCS.

The syntax of CCSσ is just (pure) CCS in the version of [DNH84] with a new action σ. We admit the totally divergent or undefined process Ω. We denote the set of visible actions (other than σ) by Vis (ranged over by $a, b, c, \ldots$) and let μ range over $Act = Vis \cup \{\sigma, \tau\}$. We let $u, v, \ldots$ range over terms and $p, q, \ldots$ over the set of processes $Proc$ (that is, terms with no free variables). There is a variable-binding recursion operator rec. We assume that all recursions are guarded. We do not allow σ to be restricted or renamed.

The transition relation $\rightarrow \subseteq Proc \times Act \times Proc$ is defined as usual except for σ. First let $DF(p)$ hold iff p has no unguarded occurrence of Ω (intuitively if p can diverge without engaging in any transition). Then define a stability predicate by

$$stab(p) \iff DF(p) \& p \stackrel{\tau}{\not\rightarrow}$$

We stipulate that $p \xrightarrow{\sigma} p'$ cannot hold unless $stab(p)$. This ensures that τ has priority over σ; for any p, if $p \xrightarrow{\tau}$ then $p \stackrel{\sigma}{\not\rightarrow}$. Apart from this stipulation the rules for $\xrightarrow{\sigma}$ are as usual except for $|$ where we have:

- if $p \xrightarrow{\sigma} p'$, $q \xrightarrow{\sigma} q'$, $stab(p|q)$ then $p|q \xrightarrow{\sigma} p'|q'$
- if $p \xrightarrow{\sigma} p'$, $q \stackrel{\sigma}{\not\rightarrow}$, $stab(p|q)$ then $p|q \xrightarrow{\sigma} p'|q$, $q|p \xrightarrow{\sigma} q|p'$

Now we give the explicit definitions. We let L range over subsets of Vis and as usual actions $a \in Vis$ have complements $\bar{a}$. A renaming is a function from Act to Act which respects complements and leaves τ, σ unchanged. We let $S, \ldots$ range over renamings. To handle recursion we consider open terms. We let $x, \ldots$ range over process variables. The syntax of CCSσ terms is as follows:

$$u ::= x|\mathsf{nil}|\Omega|\mu u|u_1 + u_2|u_1|u_2|u\backslash L|u[S]|\mathsf{rec}x.u$$

First of all we define the predicate DF by structural induction.

Definition 1 • *$DF(\mathsf{nil})$, $DF(\mu u)$*

- *If $DF(u)$ then $DF(u\backslash L)$, $DF(u[S])$, $DF(\mathsf{rec}x.u)$*
- *If $DF(u)$, $DF(v)$ then $DF(u+v)$, $DF(u|v)$*

Notice that this allows $DF(\tau\Omega)$; DF simply identifies *immediate* divergence.

We now give the definition of the transition relation $\rightarrow$. Recall that we let

$$stab(u) \iff DF(u) \& u \stackrel{\tau}{\not\rightarrow}$$

We define $\rightarrow$ by structural induction.

Definition 2

$$\mu u \xrightarrow{\mu} u$$

$$\frac{u \xrightarrow{\mu} u'}{u + v \xrightarrow{\mu} u'} \qquad \textit{provided } stab(v) \textit{ if } \mu = \sigma$$

$$\frac{v \xrightarrow{\mu} v'}{u + v \xrightarrow{\mu} v'} \qquad \textit{provided } stab(u) \textit{ if } \mu = \sigma$$

$$\frac{u \xrightarrow{a} u' \quad v \xrightarrow{\bar{a}} v'}{u|v \xrightarrow{\tau} u'|v'}$$

$$\frac{u \xrightarrow{\mu} u'}{u|v \xrightarrow{\mu} u'|v} \qquad \frac{v \xrightarrow{\mu} v'}{u|v \xrightarrow{\mu} u|v'} \qquad \mu \neq \sigma$$

$$\frac{u \xrightarrow{\sigma} u' \quad v \xrightarrow{\sigma} v'}{u|v \xrightarrow{\sigma} u'|v'} \qquad \textit{provided } \not\exists a \in Vis \textit{ such that } u \xrightarrow{a}, v \xrightarrow{\bar{a}}$$

$$\frac{u \xrightarrow{\sigma} u' \quad v \stackrel{\sigma}{\not\rightarrow}}{u|v \xrightarrow{\sigma} u'|v} \qquad \frac{v \xrightarrow{\sigma} v' \quad u \stackrel{\sigma}{\not\rightarrow}}{u|v \xrightarrow{\sigma} u|v'} \qquad \textit{provided } \not\exists a \in Vis \textit{ such that } u \xrightarrow{a}, v \xrightarrow{\bar{a}}$$

$$\frac{u \xrightarrow{\mu} u'}{u\backslash L \xrightarrow{\mu} u'\backslash L} \qquad \textit{provided } \mu \notin L \cup \bar{L}$$

$$\frac{u \xrightarrow{\mu} u'}{u[S] \xrightarrow{S(\mu)} u'[S]}$$

$$\frac{u \xrightarrow{\mu} u'}{\mathsf{rec}x.u \xrightarrow{\mu} u'[\mathsf{rec}x.u/x]}$$

The following can be checked easily by structural induction:

Proposition 3 *If* $u \xrightarrow{\sigma}$ *then* $stab(u)$.

We would like the new transition relation for the augmented language CCSσ to be *conservative* over CCS, in the sense that we obtain exactly the same transitions as usual when we restrict ourselves to terms of CCS. This is indeed the case. For clarity in the following proposition let $\rightarrow$ denote the usual transition relation of CCS and let $\rightarrow_\sigma$ denote the new transition relation of CCSσ .

Proposition 4 $\rightarrow_\sigma$ *is conservative over* $\rightarrow$*, that is, if* u *is a term of CCS then*

- *if* $u \xrightarrow{\mu} u'$ *then* $u \xrightarrow{\mu}_\sigma u'$
- *if* $u \xrightarrow{\mu}_\sigma u'$ *then* $\mu \neq \sigma$*,* u' *is a term of CCS, and* $u \xrightarrow{\mu} u'$

In view of the proposition we drop the subscript from $\rightarrow_\sigma$.

4 The correspondence between the internal and external forms of testing

We recall the definition of refusal equivalence from [Phi87]. Refusal tests were defined inductively by:

- SUCC, FAIL are tests
- if t_i, t_j are tests then so is $\sum_{i \in I} a_i t_i + \sum_{j \in J} \tilde{A} t_j$

where $A = \{a_i | i \in I\}$ and $J \neq \emptyset$. Transition relations $\xrightarrow{a}$, $\xrightarrow{\tilde{A}}$ are defined in the obvious fashion, the latter meaning that all actions in A are refused. We defined a *computation* of a test t and a process p to be a maximal sequence

$$t|p \rightarrow \cdots$$

according to the rules

- CP1 $p \xrightarrow{\tau} p'$ implies $t|p \xrightarrow{\tau} t|p'$
- CP2 $p \xrightarrow{a} p', t \xrightarrow{a} t'$ implies $t|p \xrightarrow{(aa)} t'|p'$
- CP3 $p \stackrel{a}{\not\rightarrow}$ for all $a \in A, stab(p), t \xrightarrow{\tilde{A}} t'$ implies $t|p \xrightarrow{\tilde{A}\cdot} t'|p'$, where $p \xrightarrow{\sigma} p'$ or $p' = p$ if $p \stackrel{\sigma}{\not\rightarrow}$
- CP4 $\neg DF(p)$ implies $t|p \xrightarrow{\cdot\Omega} t|\Omega$

We have altered CP3 to take account of the fact that in CCSσ p may contain observers who have to synchronise with the $\tilde{A}$ transition. Of course for CCS processes we always have $p = p'$. The *outcome* of a computation is success if it reaches $\textsc{succ}|p'$, some p'. Otherwise it is failure.

As in [DNH84] let CCSσ observers be CCSσ processes where a new action ω is allowed, representing success. Define a translation of refusal tests into CCSσ observers by:

$$g(\textsc{succ}) = \omega.\mathsf{nil}$$

$$g(\textsc{fail}) = \mathsf{nil}$$

$$g(\sum a_i t_i + \sum \tilde{A} t_j) = \sum \bar{a}_i g(t_i) + \sum \sigma g(t_j)$$

The following valid rules of CCSσ correspond to CP1–4:

CP1$'$ $p \xrightarrow{\tau} p'$ implies $q|p \xrightarrow{\tau} q|p'$

CP2$'$ $p \xrightarrow{a} p'$, $q \xrightarrow{\bar{a}} q'$ implies $q|p \xrightarrow{\tau} q'|p'$

CP3$'$ $stab(q|p)$, $q \xrightarrow{\sigma} q'$ implies $q|p \xrightarrow{\sigma} q'|p'$, where $p \xrightarrow{\sigma} p'$ or $p' = p$ if $p \not\xrightarrow{\sigma}$

There is no counterpart to CP4. A computation of a CCSσ test t on a process p is a maximal sequence of σ/τ transitions starting from $t|p$. The *result* is success if the computation reaches some $t'|p'$ such that $t' \xrightarrow{\omega}$. It is failure if it is not successful or if it passes through some state $t''|p''$ such that $\neg DF(p)$ before succeeding. Notice that a single computation can be both a success and a failure, the added complication being because we are not allowed CP4. The definition of computation and of its result is exactly as in [DNH84] except that we allow $\xrightarrow{\sigma}$ transitions in a computation as well as $\xrightarrow{\tau}$. Define the *result set* of test t applied to p to be

$R(t,p) = \{\top|$ some computation $t|p \rightarrow$ is a success$\} \cup \{\bot|$ some computation $t|p \rightarrow$ is a failure$\}$

if t is a CCSσ process. The *outcome set* $O(t,p)$ is defined similarly if t is a refusal test. Define *refusal equivalence* by

$$p \sim_R q \iff \text{for all refusal tests } t, O(t,p) = O(t,q)$$

and define CCSσ testing equivalence by

$$p \sim_\sigma q \iff \text{for all CCS}\sigma \text{ processes } t, R(t,p) = R(t,q)$$

It is clear that there is a correspondence between computations in the two systems so that for any refusal test t and any process p

$$O(t,p) = R(g(t),p)$$

Hence

Proposition 5 *For any CCSσ processes p, q, if $p \sim_\sigma q$ then $p \sim_R q$.*

We wish to establish the converse to this proposition. This we do by showing that a restricted set of CCSσ processes is adequate for CCSσ testing.

First we recall from [DNH84] that testing equivalence can be decomposed as follows. Define

$$p \ \mathbf{may} \ t \iff \top \in R(t,p)$$

$$p \ \mathbf{must}\ t \iff R(t,p) = \{\top\}$$

Then $p \sim_\sigma q$ iff for all t,

$$p \ \mathbf{may}\ t \iff q \ \mathbf{may}\ t$$

$$p \ \mathbf{must}\ t \iff q \ \mathbf{must}\ t$$

Now we examine which processes are really needed for testing. First notice that tests of infinite depth are unnecessary, since successful computations are finite. The argument is much as for De Nicola & Hennessy testing; for may testing it is simple, and for must one uses König's Lemma. Furthermore any occurrences of the undefined process Ω are unnecessary, since for may one just omits them and for must one curtails the test to nil wherever an Ω occurs. Since $\rightarrow$ is finite branching any CCSσ process can be expressed as

$$p = \sum_{i \in I} \mu_i p_i$$

Hence any test can be expressed as

$$t = \sum_{i\in I} a_i t_i + \sum_{j \in J} \tau t_j + \sum_{k \in K} \sigma t_k \tag{1}$$

where if $J \neq \emptyset$ then $K = \emptyset$. Now we argue somewhat as in [Phi87]. For may testing such a test can be replaced by a disjunction of tests as follows: Let $L = \{a_i | i \in I\}$. There are three cases:

Case 1. $J \neq \emptyset$, $K = \emptyset$. Then for any p

$$p \ \mathbf{may}\ t \iff \bigvee_i p \ \mathbf{may}\ \sigma + a_i t_i \vee \bigvee_j p \ \mathbf{may}\ t_j$$

Case 2. $J = \emptyset$, $K \neq \emptyset$. Then for any p

$$p \ \mathbf{may}\ t \iff \bigvee_i p \ \mathbf{may}\ \sigma + a_i t_i \vee \bigvee_k p \ \mathbf{may}\ \sigma t_k + \sum_{a \in L} a$$

Case 3. $J = K = \emptyset$. Define

$$t_0 = \sigma + \sum_{i \in I} a_i t_i$$

$$t_{n+1} = \sigma t_n + \sum_{i \in I} a_i t_i$$

Then for any p

$$p \ \mathbf{may}\ t \iff \bigvee_{n < \omega} p \ \mathbf{may}\ t_n$$

The added complication in this case is that p might perform some σ's on its own. For example p **may** $a\omega$ does not imply p **may** $\sigma + a\omega$.

It can then be seen that for may testing it is sufficient to use tests generated as follows:

(may) $$t ::= \omega | \sigma + at | \sigma t + \sum_{a \in L} a$$

(we allow $L = \emptyset$). For must testing we have special cases when $t = \mathsf{nil}$ or when $t \xrightarrow{\omega}$ (in which case t may as well be replaced by ω). Excluding these cases, suppose that t is in the form of (1)). There are the same three cases as before:

Case 1. $J \neq \emptyset$. Then for any p

$$p \textbf{ must } t \iff \bigwedge_i p \textbf{ must } \sigma\omega + a_i t_i \wedge \bigwedge_j p \textbf{ must } t_j$$

Case 2. $J = \emptyset$, $K \neq \emptyset$. Then for any p

$$p \textbf{ must } t \iff \bigwedge_i p \textbf{ must } \sigma\omega + a_i t_i \wedge \bigwedge_k p \textbf{ must } \sigma t_k + \sum_{a \in L} a\omega$$

where L is as before.

Case 3. $J = K = \emptyset$. Define t_n as before. Then by a König's Lemma argument, for any p

$$p \textbf{ must } t \iff \bigvee_{n<\omega} p \textbf{ must } t_n$$

It can then be seen that for must testing we need only tests generated as follows:

$$\text{(must)} \qquad t ::= \sigma\omega | \mathsf{nil} | \sigma\omega + at | \sigma t + \sum_{a \in L} a\omega$$

(again we allow $L = \emptyset$). But of course the tests generated as in (may), (must) are already (translations of) refusal tests. Hence

Proposition 6 *For any p, q, if $p \sim_R q$ then $p \sim_\sigma q$.*

Together with Proposition 5 this shows that we have succeeded in our aim of characterising refusal testing by an internal notion of testing. In particular, for any CCS processes p, q

$$p \sim_R q \iff p \sim_\sigma q$$

Remark. In [Phi87] we were able to simplify the tests even further than (may), (must) above, since we could replace the simultaneous refusal of a set of actions by the sequence of refusing them one by one. This is not possible here, since processes may perform transitions as a result of refusals.

5 Traces and congruences

When defining the trace of a process it is usual to omit internal or silent actions. Since we have decided to regard σ as unobservable, define a *trace* of a process p to be any finite sequence of actions in which p can engage, omitting any σ's and τ's. Say that $p \sim_T q$ iff

$$\text{for all } s, s \text{ is a trace of } p \iff s \text{ is a trace of } q$$

This trace equivalence is appealing in its simplicity. However as recalled above, it has been rejected because of its insensitivity to deadlock. In [HBR84] the concept of trace was refined to that of a failure, being a trace followed by a set of actions which the process could refuse simultaneously. [DNH84] followed a different line, introducing the "must" modality in addition to "may", which of course corresponds to trace for their testing.

For every equivalence $\sim$ on CCSσ there is a corresponding congruence, defined by $p \sim^c q$ iff for all CCSσ contexts $C[\cdot], C[p] \sim C[q]$. As with other testing equivalences, CCSσ testing equivalence *eqts* is nearly a congruence, except that we have to take into account initial τ and σ actions to cope with +-contexts.

Proposition 7 *For any p, q*

$$p \sim_\sigma^c q \iff p \sim_\sigma q \& (p \xrightarrow{\tau} \iff q \xrightarrow{\tau}) \& (p \xrightarrow{\sigma} \iff q \xrightarrow{\sigma})$$

□

A version of this was shown by Liebert [Lie].

In this section we show that trace congruence is actually as discriminating as $\sim_\sigma^c$ on the strongly convergent processes, which include all finite processes.

Definition 8 *p is* convergent *iff $DF(p) \& p \stackrel{\tau^\omega}{\not\rightarrow}$. p is* strongly convergent *iff for all p' reachable from p, p' is convergent.*

The tests we defined in (may), (must) may be thought of as finitely branching trees with either nil or ω at the leaves. It should be clear that in a computation of such a test t and a strongly convergent process p it is guaranteed that the test will eventually reach a leaf node. Thus every computation is either a success or a finite failure. In dealing with strongly convergent processes it is not hard to check that we can omit the test $\sigma\omega$ from (must). Just as in [Phi87] we can define a dual map between (may) and (must):

$$\begin{array}{ll}
d(\omega) = \mathsf{nil} & d(\mathsf{nil}) = \omega \\
d(\sigma + at) = \sigma\omega + ad(t) & d(\sigma\omega + at) = \sigma + ad(t) \\
d(\sigma t + \sum_{a \in L} a) = \sigma d(t) + \sum_{a \in L} a\omega & d(\sigma t + \sum_{a \in L} a\omega) = \sigma d(t) + \sum_{a \in L} a
\end{array}$$

This is simply swapping ω's and nil's at leaf nodes.

Proposition 9 *If p is strongly convergent and t is a (may) or (must) test then*

$$p \ \mathbf{must}\ t \iff p \ \cancel{\mathbf{may}}\ d(t)$$

This shows that we only need may testing when considering strongly convergent processes. Take any (may) test t and any p. Then

$$p \ \mathbf{may}\ t \iff \omega \text{ is a trace of } t|p \tag{2}$$

(Notice that we do not need ω to be a special action here; we can rename it to any action not in the alphabet of p,t). On the other hand, clearly for any $s \in Act^*$

$$s \text{ is a trace of } p \iff p \ \mathbf{may}\ \bar{s}\omega \tag{3}$$

where $\bar{s}$ is defined in the obvious fashion. Hence combining Proposition 9 and (2), (3) we obtain:

Proposition 10 *For strongly convergent processes p,q*

$$p \sim^c_T q \iff p \sim^c_\sigma q$$

6 Bisimulation for CCSσ

We now address two issues arising from our formulation of CCSs.

- σ is "semi-invisible". A process σP can observe σ, but cannot be sure whether any other process took part. How can we make sense of this?
- We should like the addition of s to be fairly "orthogonal" to the rest of CCS, and in particular $\tau$and the τ-laws. Ideally we should just add axioms to deal with σ.

Some of the difficulty arises from the fact that $\sim_\sigma$deals with both σ and τ. We can clarify matters by looking at more refined equivalences, namely strong and weak bisimulation.

Since we have a transition relation *tran* on CCSσ the definition of strong bisimulation $\sim$ [Mil89] is immediate. It will be a congruence, just as for CCS. However we would like to reflect the "semi- invisible" character of σ. Consider the processes b and $b+\sigma b$. The latter can perform σ, but the former is stable. If we wish to observe stability by performing σ then these processes are equivalent. Also in both cases b can occur immediately without waiting for stability. In order to identify these processes we borrow the idea of idling from Hennessy & Regan [HR91]. Formally we define a new transition system derived from the operational semantics of CCSs in much the same way that Milner [Mil89] obtains the weak transition relation *wtran* for CCS by hiding τs.

Definition 11 *For* $\mu \in Vis \cup \{\tau\}$

$$p \xrightarrow{\mu}_\sigma q \iff p \xrightarrow{\mu}$$

$$p \xrightarrow{\sigma}_\sigma q \iff p \xrightarrow{\sigma} \text{ or } (p = q \text{ and } p \stackrel{\sigma}{\not\rightarrow}, stab(p))$$

Strong and weak bisimulation on CCSσ with respect to transS are denoted by $\sim_1$, $\approx_1$ *respectively.*

It is easily seen that $b \sim_1 b + \sigma b$. However note that b and σb are distinguished, since the former can perform b as well as σ. Both $\sim_1$ and *wbisS* are congruences with respect to parallel composition. However neither is a congruence with respect to $+$. In fact for any p, q

$$p \sim^c_1 q \text{ iff } p \sim_1 q \text{ and } (p tran\sigma \iff q tran\sigma)$$

We do really mean $tran\sigma$ rather than $tranS\sigma$ in the above.

Note that when we restrict attention to CCS processes *wbisS* is stronger than the usual weak bisimulation $\approx$. For instance *wbisS* can distinguish between the processes τ and $\mathsf{rec}\, x.\tau x$, which are equated in the usual theory. The example illustrates the fact that *wbisS* is sensitive to divergence in a way that $\approx$ is not. However

$$\tau \approx_1 \mathsf{rec}\, x.\tau(x+\tau)$$

whereas these would be distinguished by testing because the right-hand process can diverge.

Proposition 12 *For strongly convergent p, q*

- $p \approx q$ *implies* $p \approx_1 q$
- $p \approx_1 q$ *implies* $p \sim_\sigma q$

$\square$

6.1 Axiomatizations

We now discuss axiomatizations of bisimulation for CCSσ . We restrict our attention to finite processes. We show that our weak equivalence has the same axioms as in CCS, except that we add two new σ-axioms.

It is a simple matter to modify the Expansion Theorem [Mil89] to handle σ, so that every finite process can be converted to one involving only nil, prefixing and $+$. To axiomatize $\sim$ we need only add

$\sigma 1.$ $\tau p + \sigma q = \tau p$

to the usual axioms. To handle $\sim_1$ we further add

$\sigma 2.$ $\mu p = \mu(p + \sigma p)$ where $p \stackrel{\sigma}{\not\rightarrow}$

For $\approx_1$ we add the usual τ-laws [Mil89]. Let $=$ denote $\approx_1^c$, and let $\vdash p = q$ denote that p can be proved equal to q in the axiom system we have just described.

Proposition 13 *(Soundness) If $\vdash p = q$ then $p = q$.* $\square$

We now sketch the completeness proof, which largely follows the methods of [Mil89]. We use two lemmas:

Proposition 14 *(Hennessy Lemma) If $p \approx_1 q$ then $p = q$ or $p = \tau q$ or $\tau p = q$ or p,q are stable and $(q \stackrel{\sigma}{\not\rightarrow},\ p = q + \sigma q)$ or $(p \stackrel{\sigma}{\not\rightarrow},\ p + \sigma p = q)$*

Proof. Cases according to whether $p, q \stackrel{\tau}{\rightarrow}, \stackrel{\sigma}{\rightarrow}$. $\square$

Proposition 15 *(Saturation Lemma) If $p \stackrel{\mu}{\Rightarrow}_1 p'$ then $\vdash p = p + \mu p'$* $\square$

Milner uses this lemma to get a normal form by saturating processes. For our proof it is not clear how to obtain a normal form, since saturation using $\sigma 2$ from left to right will not terminate. So we don't bother, and instead show that if $p = q$ and $\mu q'$ is a summand of q then $\vdash p = p + \mu q'$. Applying this repeatedly gives $\vdash p = p + q$. Similarly $q = q + p$ and we are done.

Theorem 16 *(Completeness) If $p = q$ then $\vdash p = q$* $\square$

Axiomatisation of testing equivalence $\sim_\sigma$ has been investigated by Liebert [Lie].

7 Conclusion

8 Acknowledgements

I am grateful to Samson Abramsky, Matthew Hennessy, Bent Thomsen, Jos Baeten for helpful comments and suggestions. This paper has also benefited from Sarah Liebert's work on CCSσ . I also wish to thank K.V.S. Prasad and the organisers of the 1991 Chalmers Workshop on Concurrency, where some of this material was presented.

References

[BBK85] J. C. M. Baeten, J. Bergstra, and J.-W. Klop. Syntax and defining equations for an interrupt mechanism in process algebra, Technical Report CS-R8503, Department of Computer Science, Centre for Mathematics and Computer Science, Amsterdam. 1985.

[BBK87] J. C. M. Baeten, J. Bergstra, and J.-W. Klop. Ready-trace semantics for concrete process algebra with priority operator. *The Computer Journal*, 30:498–506, 1987.

[CH88] R. Cleaveland and M. Hennessy. Priorities in process algebra, LICS. 1988.

[CW91] J. Camilleri and G. Winskel. CCS with priority choice, LICS. 1991.

[DNH84] R. De Nicola and M. C. B. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 34:83–134, 1984.

[HBR84] C. A. R. Hoare, S. D. Brookes, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the Association for Computing Machinery*, 31:560–599, 1984.

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

[HR91] M. Hennessy and T. Regan. A process algebra for timed systems, Dept. of Computer Science, University of Sussex, Tech. Report 5/91. 1991.

[Lan92] R. Langerak. Transformations and semantics for LOTOS, PhD thesis, University of Twente. 1992.

[Lie] S.A.J. Liebert. forthcoming PhD thesis, Imperial College, University of London.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[Mil81] R. Milner. A modal characterisation of observable machine-behaviour. *Lecture Notes in Computer Science*, 112:25–34, 1981.

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

[NRSV90] X. Nicollin, J.-L. Richier, J. Sifakis, and J. Voiron. ATP: an algebra for timed processes. 1990.

[Phi87] I. C. C. Phillips. Refusal testing. *Theoretical Computer Science*, 50:241–284, 1987.

[Pnu85] A. Pnueli. Linear and branching structures in the semantics and logics of reactive systems. *Lecture Notes in Computer Science*, 194:15–32, 1985.

[Pra91] K.V.S. Prasad. A calculus of broadcasting systems, CAAP. *Lecture Notes in Computer Science*, 493:338–358, 1991.