DIVISION OF COMPUTER SCIENCE

A Conservative Extension to CCS for True Concurrency Semantics

Jean Baillie
David Smith

Technical Report No.200

May 1994

A Conservative Extension to CCS for True Concurrency Semantics

Jean Baillie
David Smith

1 Introduction

In this paper we develop the theory of Concurrent CCS, first introduced in [Smi91]. CCCS is a conservative extension to CCS which provides true concurrency semantics. This is achieved by (i) a new action prefix operator, denoted by $\bullet$, which 'ties together' its operands so that they occur simultaneously, (ii) a new parallel composition operator, denoted by $||$, and (iii) the possibility of multiway synchronization.

The notion of simultaneous actions is captured pragmatically by the following condition: in $a \bullet b$, a and b act simultaneously and so $a \bullet b = b \bullet a$. In particular $\tau \bullet a = a \bullet \tau = a$. CCCS allows and, indeed, sometimes forces, multiway synchronization.

Having introduced the notion of simultaneous actions (which we shall refer to as true concurrency, other notions of the term notwithstanding), we need a new parallel composition operator with a different semantics from the interleaving composition of $|$. Apart from the fact that simultaneous actions arise naturally when two or more agents act concurrently, the operator is necessary in order to prevent the deadlock which may occur due to action restriction. For example if we have a process p, say, which contains a term such as $\ldots a \bullet b.p$ and we restrict by $\{a, b\}$, then p may need to communicate with two distinct agents in order for the restricted actions a and b both to occur. We need to develop multiway synchronization to address this. We are not concerned here with synchrony in the sense of wishing agents to proceed in lockstep, as in SCCS [Mil83]. CCCS agents may still proceed at indeterminate relative speeds, just as CCS agents, so we do not need to make explicit the action of *waiting*. There is no assumption of a global clock; CCCS agents may proceed independently or simultaneously. We are *enabling* synchrony rather than assuming or forcing it. One consequence of this is that parallel composition in CCCS does not distribute over summation, whereas in SCCS, the *product* combinator (denoted by $\times$ and corresponding to our parallel composition) does indeed distribute over summation.

The new parallel composition operator, denoted by $||$, allows its operands to run in true concurrency, so far as considerations of synchronization will permit, as well as allowing them to run independently. $||$ will be used anywhere that we wish to admit the possibility of simultaneous execution of processes. So in the case of an agent containing $a \bullet b$ which is composed under $||$ with other agents, a and b will always act in the same context, that is, they act simultaneously with each other and *may* act simultaneously with appropriate multisets of actions from the other agents in the composition.

2 Formal semantics of CCCS

2.1 Notation

Act_c in CCCS is the commutative group $\{Act_c, \tau, \bullet, {}^{-}\}$ with identity τ. We use u, v to range over Act_c. We will use the notation $p \overset{u}{\rightarrow} p'$ to mean that p may engage in the multiset of actions u (which may be empty) and evolve to p'. For each $s \in Act_c$, we define the multiset s' as follows:
(i) if s is atomic, then $s' = \{s\}$
(ii) $\tau' = \{\}$
(iii) if $s = u \bullet v$ then $s' = u' \uplus v'$ where $\uplus$ here denotes multiset union.
So for each $s \in Act_c$, s' denotes the multiset of observable actions that occur simultaneously. Action complementation is defined as follows:

$$\{\}^c = \{\},\ \{a, b\}^c = \{\overline{a}, \overline{b}\},\ \text{where } \overline{\overline{a}} = a.$$

We need to define one more piece of notation: let u and v be any two multisets. Then

$$u \dagger v = \{s \uplus t | s \subseteq u, t \subseteq v, u - s = \{v - t\}^c\}$$

where '$-$' denotes multiset difference. (It can easily be shown that $\dagger$ is associative.) For example, consider the multisets $u = \{a, b, \overline{c}\}$, $v = \{\overline{a}, c\}$. Then

$$u \dagger v = \{\{b\}, \{a, \overline{a}, b\}, \{b, c, \overline{c}\}, \{a, \overline{a}, b, c, \overline{c}\}\}$$

The intuition here is that *either* an action and its complement are both visible, *or* they have communicated internally. For instance, the set $\{b\}$ indicates that both a and c have communicated through their respective complementary ports; the set $\{a, \overline{a}, b\}$ indicates that c has communicated internally but a and its complement are visible. So then, interpreting this as the possible actions of $P||Q$ where $P \overset{def}{=} a \bullet b \bullet \overline{c}$ and $Q \overset{def}{=} \overline{a} \bullet c$, we have

$$P \,||\, Q = b + a \bullet \overline{a} \bullet b + b \bullet c \bullet \overline{c} + a \bullet \overline{a} \bullet b \bullet c \bullet \overline{c}$$

2.2 Transition semantics

We use s to range over actions in Act_c, and P, Q to range over processes.

Action Prefixing

$s \bullet P \xrightarrow{s'} P$

Choice

$P \xrightarrow{u} P'$ implies $P + Q \xrightarrow{u} P'$
$Q \xrightarrow{u} Q'$ implies $P + Q \xrightarrow{u} Q'$

Constants

$P \xrightarrow{u} P'$ and $Q \overset{def}{=} P$ imply $Q \xrightarrow{u} P'$

Interleaved composition

$P \xrightarrow{u} P'$ implies $P|Q \xrightarrow{u} P'|Q$
$Q \xrightarrow{u} Q'$ implies $P|Q \xrightarrow{u} P|Q'$
$P \xrightarrow{u} P'$, $Q \xrightarrow{u}^{c} Q'$ implies $P|Q \xrightarrow{\{\}} P'|Q'$

The semantics for this operator are substantially those for pure CCS. We are effectively regarding multisets of actions as atomic units; only two-way synchronization is permitted and then only between multisets which are uniquely complementary.

Concurrent Composition

We define this by case analysis.
(i) each component of $P \,||\, Q$ may proceed independently and interact with its environment, or
(ii) agents may proceed concurrently and interact with their environment and/or with each other. Formally,

$P \xrightarrow{u} P'$ implies $P \,||\, Q \xrightarrow{u} P' \,||\, Q$
$Q \xrightarrow{u} Q'$ implies $P \,||\, Q \xrightarrow{u} P \,||\, Q'$
$P \xrightarrow{u} P'$, $Q \xrightarrow{v} Q'$ implies $P \,||\, Q \xrightarrow{r} P' \,||\, Q'$ for every $r \in u \dagger v$.

Restriction

Let L be any set of actions, $\tau \notin L$. Then

$P \xrightarrow{u} P'$ implies $P\backslash L \xrightarrow{u} P'\backslash L$ if $u \cap L = \emptyset$ and $u^c \cap L = \emptyset$.

Relabelling

$P \xrightarrow{u} P'$ implies $P[f] \xrightarrow{f(u)} P'[f]$

where f is a relabelling function and $f(u)$ is the multiset $\{f(a) \,|\, a \in u\}$.

We will give some examples to illustrate the use of the calculus, but first we need a mechanism for expanding agents composed under $||$.

3 The Concurrent Expansion Law

Proposition 1

Let $P \equiv (P_1||P_2||\ldots||P_n),\ \ n \geq 1$. Then

$$\begin{array}{rcl}
P & = & \Sigma\{u_i.(P_1||\ldots||P_i'||\ldots||P_n)\} \text{ where } P_i \xrightarrow{u_i} P_i' \\
 & + & \Sigma\{r_2.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_n) \ \text{ for every } r_2 \in u_i \dagger u_j, \\
 & & \ \text{where } P_i \xrightarrow{u_i} P_i', P_j \xrightarrow{u_j} P_j'\} \\
 & + & \Sigma\{r_3.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_k'||\ldots P_n) \ \text{ for every } r_3 \in (u_i \dagger u_j) \dagger u_k, \\
 & & \ \text{where } P_i \xrightarrow{u_i} P_i', P_j \xrightarrow{u_j} P_j', P_k \xrightarrow{u_k} P_k'\} \\
 & + & \ldots \\
 & + & \Sigma\{r_n.(P_1'||\ldots||P_i'||\ldots||P_n') \ \text{ for every } r_n \in (\ldots(u_1 \dagger u_2)\ldots \dagger u_i)\ldots \dagger u_n), \\
 & & \ \text{where } P_i \xrightarrow{u_i} P_i', 1 \leq i \leq n\}
\end{array}$$

Informally, this means that the P_i may run independently through their observable actions (the first line of the law) or else they may run concurrently in any combination of tuples up to and including the case where all n are running concurrently. As can be imagined, this makes for an explosion of states very early on in the expansion in cases where there are no occurrences of the $\bullet$ operator and no restriction. The operator $\bullet$, however, has the effect of reducing the number of possible states quite significantly, as we shall show.

Example 1

Consider the agents $P \overset{def}{=} a \bullet b.0$, $Q \overset{def}{=} c.\overline{b}.0$, first of all composed under $|$:

$$\begin{aligned} P\,|\,Q &= c.(P\,|\,\overline{b}.Q) + a \bullet b.(0\,|\,Q) \\ &= c.(a \bullet b.(0|\overline{b}.Q) + \overline{b}.(P\,|0)) \\ &\quad +a \bullet b.c.\overline{b}.(0|0) \\ &= c.(a \bullet b.\overline{b}.0 + \overline{b}.a \bullet b.0) \\ &\quad +a \bullet b.c.\overline{b}0 \end{aligned}$$

If we restrict by b the compositon is deadlocked after the first action c, that is

$$P\,|\,Q\backslash\{b\} = c.(P\,|\,\overline{b}.Q)\backslash\{b\}$$

Considering the same agents composed with $||$ we have

$$\begin{aligned} P\,||\,Q &= c.(P\,||\,\overline{b}.Q) + a \bullet b.(0\,||\,Q) \\ &\quad +a \bullet b \bullet c.(0||\overline{b}.Q) \\ &= c.(a \bullet b.(0||\overline{b}.Q) + \overline{b}.(P\,||0) + a.(0||0)) \\ &\quad +a \bullet b.c.\overline{b}.(0||0) \\ &= c.(a \bullet b.\overline{b}.0 + \overline{b}.a \bullet b.0 + a.0) \\ &\quad +a \bullet b.c.\overline{b}.0 \end{aligned}$$

Restriction by b gives the result $P\,||\,Q\backslash\{b\} = c.a.0\backslash\{b\}$.

Example 2

Consider the agents $P = a \bullet b.P$ and $Q = c.\overline{b}.Q$

$$\begin{aligned} P \mid Q &= c.(P \mid \overline{b}.Q) + a \bullet b.(P \mid Q) \\ &= c.(a \bullet b.(P \mid \overline{b}.Q) + \overline{b}.(P \mid Q)) + a \bullet b.(P \mid Q) \\ &= c.(a \bullet b.a \bullet b.(P \mid \overline{b}.Q) + a \bullet b.\overline{b}.(P \mid Q) + \overline{b}.(P \mid Q)) + a \bullet b.(P \mid Q) \end{aligned}$$

Let $R = P \mid \overline{b}.Q$. Then

$$R = a \bullet b.R + \overline{b}.P \mid Q$$

So we may rewrite

$$P \mid Q = c.R + a \bullet b.(P \mid Q), \text{ where } R = a \bullet b.R + \overline{b}.P \mid Q$$

If we restrict by b, the composition would deadlock following the first term of the first line of the expansion, that is,

$$P \mid Q \backslash \{b\} = c.(P \mid \overline{b}.Q) \backslash \{b\}$$

a and b must act together and also synchronize with $\overline{b}$, which is not permitted by $\mid$.

Now consider P and Q composed with $\mid\mid$:

$$\begin{aligned} P \mid\mid Q &= c.(P \mid\mid \overline{b}.Q + a \bullet b.(P \mid\mid Q) + a \bullet b \bullet c.(P \mid\mid \overline{b}.Q) \\ &= c.(a.(P \mid\mid Q) + a \bullet b.(P \mid\mid \overline{b}.Q) + \overline{b}.(P \mid\mid Q)) + a \bullet b \bullet c.(\overline{b}.(P \mid\mid Q) + a \bullet b.(P \mid\mid \overline{b}.Q)) \end{aligned}$$

Without completing this expansion it can be seen that the combinations are numerous when there is no restriction. However, if we were to restrict by b the result is

$$P \mid\mid Q \backslash \{b\} = c.a.(P \mid\mid Q) \backslash \{b\}$$

Example 3

Consider $P = a \bullet b.P, \quad Q = \overline{a}.c \bullet d.Q$
Again, we consider both compositions, first of all without restrictions:

$$\begin{aligned} P \,|\, Q &= a \bullet b.(P \,|\, Q) + \overline{a}.(P \,|\, c \bullet d.Q) \\ &= a \bullet b.(P \,|\, Q) + \overline{a}.c \bullet d.(P \,|\, Q) \end{aligned}$$

a and $\overline{a}$ cannot perform a silent communication because a occurs in the context of $a \bullet b$ in P. If we were to restrict the composition by a the system would be deadlocked.

$$\begin{aligned} P \,||\, Q \;=\;& a \bullet b.(P \,||\, Q) + b.(P \,||\, c \bullet d.Q) \\ & +\overline{a}.(P \,||\, c \bullet d.Q) \\ =\;& a \bullet b.(P \,||\, Q) + b.(c \bullet d.(P \,|\, Q) + a \bullet b.(P \,||c \bullet d.Q) \\ & +a \bullet b \bullet c \bullet d.(P||Q)) \\ & +\overline{a}.(a \bullet b.(P \,||\, c \bullet d.Q) + c \bullet d.(a \bullet b.P \,||\, Q) \\ & +a \bullet b \bullet c \bullet d.(P \,|\, Q)) \end{aligned}$$

Again we find that without restriction, the states are numerous. However, if we restrict by a the result is $P \,||\, Q \backslash \{b\} = b.c \bullet d.(P \,||\, Q) \backslash \{b\}$

Example 4

Lastly we consider a composition of four terms: $P = a.b.P$, $Q = a.\overline{b}.Q$ and two copies of $R = c \bullet \overline{a}.R$. We restrict by a and b. We find that the composition $P \,|\, Q \,|\, R \,|\, R \backslash \{a, b\}$ cannot proceed; it is deadlocked from the beginning. On the other hand

$$\begin{aligned}
P \,||\, Q \,||\, R \,||\, R \backslash \{a,b\} &= c.(P \,||\, \overline{b}.Q \,||\, R \,||\, R) + c.(b.P \,||\, Q \,||\, R \,||\, R) \\
&\quad + c \bullet c.(b.P \,||\, \overline{b}.Q \,||\, R \,||\, R) \backslash \{a,b\} \\
&= c.c.(b.P \,||\, \overline{b}.Q \,||\, R \,||\, R) + c.c.(b.P \,||\, \overline{b}.Q \,||\, R \,||\, R) \\
&\quad + c \bullet c.(P \,||\, Q \,||\, R \,||\, R) \backslash \{a,b\} \\
&= c.c.(P \,||\, Q \,||\, R \,||\, R) + c \bullet c.(P \,||\, Q \,||\, R \,||\, R) \backslash \{a,b\}
\end{aligned}$$

What we see from these examples is that composition under $||$ gives rise to a larger state space than composition with $|$. This is what we should expect, particularly where there is no restriction of actions and no use of $\bullet$. Care is needed in the design of agents which are to be allowed to proceed in true concurrency; there will clearly be undesirable 'clashes' and certain time orderings to be avoided. Agents which will eventually become part of a truly concurrent system will need to be specified in conjunction with one another and not as more or less independent units. It can be argued that the operator $|$ is unnecessary in CCCS; where agents are specified using the $\bullet$ prefix operator, it is unlikely that we should then wish to restrict their composition to interleaving only. However, without $|$, CCCS is not a conservative extension to CCS—that is, unless it is possible to embed CCS in CCCS using only the $||$ operator for composition, though on the face of it this seems unlikely. More work is needed on the development of the calculus before we decide we can dispense with $|$.

4 Proof of the Concurrent Expansion Law

We recall the law from section 3.

Proposition 1

Let $P \equiv (P_1||P_2||\dots||P_n),\;\; n \geq 1$. Then

$$
\begin{array}{rcl}
P & = & \Sigma\{u_i.(P_1||\dots||P'_i||\dots||P_n) :\; P_i \overset{u_i}{\to} P'_i\} \\
 & + & \Sigma\{r_2.(P_1||\dots||P'_i||\dots||P'_j||\dots||P_n) : \\
 & & \text{for every } r_2 \in u_i \dagger u_j, \text{ where } P_i \overset{u_i}{\to} P'_i, P_j \overset{u_j}{\to} P'_j\} \\
 & + & \Sigma\{r_3.(P_1||\dots||P'_i||\dots||P'_j||\dots||P'_k||\dots P_n) : \\
 & & \text{for every } r_3 \in (u_i \dagger u_j) \dagger u_k, \text{ where } P_i \overset{u_i}{\to} P'_i, P_j \overset{u_j}{\to} P'_j, P_k \overset{u_k}{\to} P'_k\} \\
 & + & \dots \\
 & + & \Sigma\{r_n.(P'_1||\dots||P'_i||\dots||P'_n) : \\
 & & \text{for every } r_n \in ((\dots(u_1 \dagger u_2)\dots \dagger u_i)\dots \dagger u_n), \text{ where } P_i \overset{u_i}{\to} P'_i, 1 \leq i \leq n\}
\end{array}
$$

Proof

The proof is by induction on n. For $n = 1$ we need to prove that

$$P_1 = \Sigma\{u_1.P_1' : \ P_1 \overset{u_1}{\rightarrow} P_1'\}$$

which follows immediately. Assume the result for n and consider $R \equiv P||P_{n+1}$. From the transition rules we have that

$$\begin{array}{rcl} R & = & \Sigma\{u_i.(P'||P_{n+1}) : P \overset{u_i}{\rightarrow} P'\} \\ & & +\Sigma\{u_{n+1}.(P||P_{n+1}') : P_{n+1} \overset{u_{n+1}}{\rightarrow} P_{n+1}'\} \\ & & +\Sigma\{r_2.(P'||P_{n+1}') : \text{ for every } r_2 \in u \dagger u_{n+1}, \text{ where } P \overset{u}{\rightarrow} P',\ P_{n+1} \overset{u_{n+1}}{\rightarrow} P_{n+1}'\} \\ & = & R_1 + R_2 + R_3 \end{array}$$

From the inductive hypothesis, the first two terms of R expand to give

$$\begin{array}{rcl} R_1 + R_2 & = & \Sigma\{u_i.(P_1||\ldots||P_i'||\ldots||P_n||P_{n+1}) : \ P_i \overset{u_i}{\rightarrow} P_i',\ 1 \leq i \leq n\} \\ & + & \Sigma\{r_2.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_n||P_{n+1}) : \ \text{ for every } r_2 \in u_i \dagger u_j, \\ & & \text{where } P_i \overset{u_i}{\rightarrow} P_i', P_j \overset{u_j}{\rightarrow} P_j'\} \\ & + & \Sigma\{r_3.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_k'||\ldots P_n||P_{n+1}) : \\ & & \text{for every } r_3 \in (u_i \dagger u_j) \dagger w, \text{ where } P_i \overset{u_i}{\rightarrow} P_i', P_j \overset{u_j}{\rightarrow} P_j', P_k \overset{w}{\rightarrow} P_k'\} \\ & + & \ldots \\ & + & \Sigma\{r_n.(P_1'||\ldots||P_i'||\ldots||P_n'||P_{n+1}) : \\ & & \text{for every } r_n \in (\ldots(u_1 \dagger u_2)\ldots \dagger u_i)\ldots \dagger u_n), \text{ where } P_i \overset{u_i}{\rightarrow} P_i', 1 \leq i \leq n\} \\ & + & \Sigma\{u_i.(P_1||\ldots||P_i'||\ldots||P_n||P_{n+1}) : \ P_{n+1} \overset{u_{n+1}}{\rightarrow} P_{n+1}',\} \end{array}$$

The first and last lines of the above can be combined, giving the required extension to the first part of the law to accommodate P_{n+1} (that is, the part that says that each agent may act independently). So far P_{n+1} has taken no part in any communication; the third term of R gives us this. For P_{n+1} to communicate with P it may be communicating with just one of the P_i, or with two of them, or three, etc., as follows:

$$\begin{array}{rl} R_3 = & \Sigma\{r_2.(P_1||\ldots||P_i'||\ldots||P_n||P_{n+1}') : \ \text{ for every } r_2 \in u_i \dagger u_{n+1}, \\ & +\Sigma\{r_3.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots P_n||P_{n+1}') : \ \text{ for every } r_3 \in (u_i \dagger u_j) \dagger u_{n+1}, \\ & \ldots \\ & +\Sigma\{r_{n+1}.(P_1'||\ldots||P_i'||\ldots||P_n'||P_{n+1}') : \\ & \text{for every } r_{n+1} \in ((\ldots(u_1 \dagger u_2)\ldots \dagger u_i)\ldots \dagger u_n) \dagger u_{n+1}, \\ & \text{where } P_i \overset{u_i}{\rightarrow} P_i', 1 \leq i \leq n+1\} \end{array}$$

When we place the summand R_3 together with $R_1 + R_2$ above, all the terms in R_3 except the last term are absorbed. So we have

$$\begin{array}{rcl}
R & = & \Sigma\{u_i.(P_1||\ldots||P_i'||\ldots||P_n||P_{n+1}) : P_i \overset{u_i}{\rightarrow} P_i'\} \\
& + & \Sigma\{r_2.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_n||P_{n+1}) : \text{ for every } r_2 \in u_i \dagger u_j, \\
& & \text{where } P_i \overset{u_i}{\rightarrow} P_i', P_j \overset{u_j}{\rightarrow} P_j'\} \\
& + & \Sigma\{r_3.(P_1||\ldots||P_i'||\ldots||P_j'||\ldots||P_k'||\ldots P_n||P_{n+1}) : \text{ for every } r_3 \in (u_i \dagger u_j) \dagger w, \\
& & \text{where } P_i \overset{u_i}{\rightarrow} P_i', P_j \overset{u_j}{\rightarrow} P_j', P_k \overset{w}{\rightarrow} P_k'\} \\
& + & \ldots \\
& + & \Sigma\{r_n.(P_1'||\ldots||P_i'||\ldots||P_n'||P_{n+1}) : \text{ for every } r_n \in ((\ldots(u_1 \dagger u_2)\ldots \dagger u_i)\ldots \dagger u_n), \\
& & \text{where } P_i \overset{u_i}{\rightarrow} P_i', 1 \leq i \leq n\} \\
& + & \Sigma\{r_{n+1}.(P_1'||\ldots||P_i'||\ldots||P_n'||P_{n+1}') : \\
& & \text{for every } r_{n+1} \in ((\ldots(u_1 \dagger u_2)\ldots \dagger u_i)\ldots \dagger u_n) \dagger u_{n+1}, \\
& & \text{where } P_i \overset{u_i}{\rightarrow} P_i', 1 \leq i \leq n+1\} \;\; \square
\end{array}$$

The proof may be extended to accommodate restriction and relabelling. The Expansion Law of CCS remains substantially unchanged except that occurrences of α, representing atomic action, are replaced by u, representing multiset action. Internal communication may only occur between complementary multisets (which may be simply complementary atomic actions).

5 Equivalence in Concurrent CCS

5.1 Bisimulation

We need to define a notion of equivalence over CCCS agents; the model which suggests itself immediately is a generalized bisimulation.

Definition 1: A binary relation $\mathcal{S} \in \mathcal{P} \times \mathcal{P}$ over agents is a (weak) *bisimulation* if $(P, Q) \in \mathcal{S}$ implies, for all $u \in Act_c$,
(i) Whenever $P \overset{u}{\rightarrow} P'$ then, for some Q', $Q \overset{\hat{u}}{\Rightarrow} Q'$ and $(P', Q') \in \mathcal{S}$
(ii) Whenever $Q \overset{u}{\rightarrow} Q'$ then, for some P', $P \overset{\hat{u}}{\Rightarrow} P'$ and $(P', Q') \in \mathcal{S}$

In fact, the more general form of the transition relation $\overset{\hat{u}}{\Rightarrow}$ is only necessary in the case when u is atomic; all occurrences of τ in multisets of actions are absorbed by definition. In this model there can be no equivalence between pairs of agents where one contains simultaneity (i.e., the operator $\bullet$) and the other does not; equivalences between standard CCS agents are preserved and so the relation is a conservative extension of weak bisimulation.

This model, while seemingly a natural extension of bisimulation, is nevertheless suspect. Bisimulation models *observation* equivalence; the u referred to in the definition are multisets of actions occurring simultaneously, not all of which, therefore, can be observed. We need to permit *sets* of observers if we wish to adopt this model of equivalence. Notwithstanding, we illustrate the notion with two examples.

Example 5

Consider the agents
$P = a \bullet b.P$
$Q = a.Q$
$R = b.R$
The composition $Q||R$ yields

$$Q||R = a.b.(Q||R) + b.a.(Q||R) + a \bullet b.(Q||R),$$

clearly not equivalent to P; but if we use the signal c to synchronize Q and R as follows
$Q' = a \bullet c.Q'$
$R' = b \bullet \overline{c}.R'$
and restrict by c we have

$$Q'||R' \backslash \{c\} = a \bullet b.(Q'||R') \backslash \{c\} = P$$

Example 6

We now consider the agent *Spec* defined as
$Spec \overset{def}{=} a.Spec + a \bullet a.Spec$
and implemented by *Imp*, defined by
$Imp \overset{def}{=} P||P\backslash\{b\}$, where $P \overset{def}{=} a.P + b.Q$, $Q \overset{def}{=} a.P$
The expansion of *Imp* gives

$$\begin{aligned} Imp &= a.(P||P) + a.(P||P) + a \bullet a.(P||P) + \tau.(Q||Q) \\ &= a.(P||P) + a \bullet a.(P||P) + \tau.(a.(P||P) + a \bullet a.(P||P)) \\ &= \tau.(a.(P||P) + a \bullet a.(P||P)) \\ &\approx Spec \end{aligned}$$

Imp is equivalent to its specification *Spec*. It is not congruent to *Spec*, owing to the instability of *Imp*.

5.2 Testing

We recall Hennessy's tests from [Hen88]. Tests of the form

$$1w + b_1(1w + \ldots + b_n(1w + a)\ldots)$$

or

$$1w + b_1(1w + b_2(1w + \ldots + b_n(a_1w + \ldots + a_kw)\ldots)$$

are sufficient to distinguish between any two processes which are not testing equivalent. However it is not clear that there is a test to distinguish between $P = a.0 + b.0$ and $Q = a \bullet b.0$. The test $aw + bw$ is certainly passed by P and intuitively we might feel that Q ought also to pass such a test since it can deliver either a or b. In order to distinguish between them (which we clearly wish to do) tests of a different form need to be designed.

Related to this question, we might ask how we are to extend the notion of traces to include simultaneity and what sort of trees might be the denotation of CCCS terms. We suggest the following model, exemplified by the agent $P \overset{def}{=} a \bullet b \bullet c.d.0$ in Figure 1.

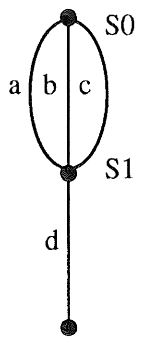

Figure 1: The tree for the CCCS agent $Q \overset{def}{=} a \bullet b \bullet c.d.0$

What we see from this is that in order for P to evolve from the state S_0 to the state S_1 all three actions a, b and c must have occurred. However the observer can take only one path through the tree, though he has three choices for this, namely, ad, bd, cd. This suggests trace (or *may*) equivalence with the agent $Q \overset{def}{=} ad + bd + cd$, represented by the tree shown in Figure 2.

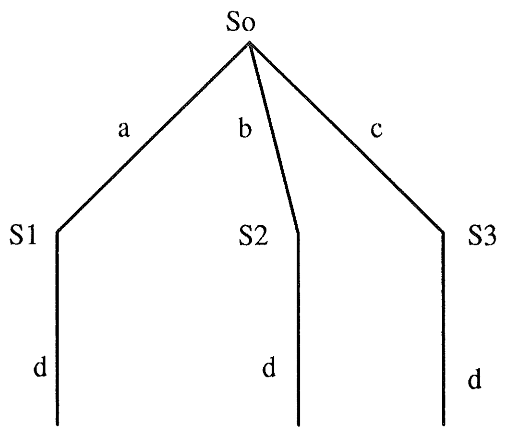

Figure 2: The tree for the CCCS agent $Q \overset{def}{=} ad + bd + cd$

Indeed we may well not wish to distinguish between these two in the trace model; an experimenter who can only observe one thing at a time (in keeping with the relativistic view) is subject to severe limitations on his capability. In that case, it is intuitively right to identify P and Q in this model. We then have that

$$\mathcal{L}(P) = \mathcal{L}(Q) = \{\varepsilon, a, b, c, ad, bd, cd\} \quad *$$

However, if we feel that, notwithstanding the real limitations of experimenters, this identification is too generous and we wish to distinguish between processes such as P and Q then we might express the language of P as

$$\mathcal{L}'(P) = \{\varepsilon, \{a,b,c\}, \{a,b,c\}d\} \quad **$$

to suggest the choice of any element of the set $\{a,b,c\}$ but also implying a difference from $\mathcal{L}(P)$ above. In this case, unfortunately, language inclusion is no longer the model for *may* preordering. The intrinsic weakness of the trace model together with the nice properties it enjoys lead us to think that there is no need for this distinction and to accept $\mathcal{L}(P)$ as a model of the traces of P, identifying as it does the agents P and Q. Unfortunately, this means that we need to define a different class of test for the explicit purpose of establishing *must* preordering only, and only where CCCS agents are involved (the tests are redundant otherwise); we shall explore this a little further.

5.2.1 Extending the set of tests: restriction

In order to distinguish $P \overset{def}{=} a \bullet b$ from $Q \overset{def}{=} a + b$ we need to design tests which exploit the simultaneity of a and b in P; though it may be true that we can see either a or b (but not both) in P, we *cannot* see a without b also occurring, or b without a. This suggests tests which include *restriction*. The question whose answer will enable us to distinguish between P and Q is: can we see a if b is restricted? or b if a is restricted? We could design an experiment, e_0, such that $e_0 = aw\backslash\{b\}$; this test—and its complement $e_1 = bw\backslash\{a\}$—is passed by Q (both *may* and *must*) but not by P (neither *may* nor *must*). If we wish to retain the trace model *, then, we need to confine the use of this type of test to *must* testing only.

It is difficult to think of any test that P should pass but not Q, bearing in mind that we are interested in observable results rather than purely theoretical ones. This would give us the orderings $P \underset{\sim}{\sqsubset}_{may} Q$ and $P \overset{\sqsubset}{\sim}_{must} Q$, with the *must* preordering here *not* relating to internal nondeterminism. It was suggested in [Bai92] that the conjunction of these two orderings might be regarded as a satisfaction relation between CCS implementations and their specifications; here, it would be a very *un*satisfactory satisfaction relation for, say, a vending machine that purported to give the choice of tea or coffee but, regardless of the choice, delivered both every time. The fact that the observer of the machine can see only one of the outputs will be of little comfort to the manufacturer. On the other hand, when we visit a cinema with two screens, we choose just one to watch, though both will run concurrently regardless of our choice. In this case, the satisfaction relation might indeed be perfectly satisfactory.

We now consider the agents $R \overset{def}{=} a \bullet b + a$ and $S \overset{def}{=} a \bullet b + a + b$; to recap, we have
$P \overset{def}{=} a \bullet b$
$Q \overset{def}{=} a + b$
$R \overset{def}{=} a \bullet b + a$
$S \overset{def}{=} a \bullet b + a + b$
and the tests $e_0 = aw\backslash\{b\}$ and $e_1 = bw\backslash\{a\}$. We find that $Q\, must\, e_0$, $R\, must\, e_0$, $S\, must\, e_0$, $Q\, must\, e_1$, $S\, must\, e_1$ but $R\, m\cancel{u}st\; e_1$ giving

$$P \overset{\sqsubset}{\sim}_{must} R \overset{\sqsubset}{\sim}_{must} Q \underset{\sim}{\sqsubset}_{must} S$$

and accepting the usual trace model, as * above, we have

$$P \underset{\sim}{\sqsubset}_{may} R \underset{\sim}{\sqsubset}_{may} Q \underset{\sim}{\sqsubset}_{may} S$$

The type of test exemplified by e_0 and e_1 does not distinguish between Q and S. In the case of Q, the choice of a, say, implicitly precludes b; if we don't want b to occur, we simply choose a (or make no choice at all) and we can be sure of no b action. In S, by contrast, we should need to exclude b explicitly, otherwise s contains implicit nondeterminism. Since *must* testing is intended to detect this and order process at least partly on that basis, this suggests that we would want to distinguish between Q and S; if so, then the new tests are still insufficient.

5.2.2 Extending the set of tests: multiple experimenters

So far we have avoided using tests which include the operator $\bullet$; this is because the experimenter has always been assumed to be singular and therefore unable to observe simultaneous actions. In now introducing this operator into tests we make an implicit assumption that there is one experimenter to observe each simultaneous action. We should still like to keep the trace model so use these new tests only for *must* testing.

Tests of the form $a_1 \bullet a_2 w + a_1 \bullet a_2 \bullet a_3 w + \ldots + a_1 \bullet a_2 \ldots \bullet a_n w$ in addition to Hennessy's set of tests will distinguish between CCCS agents, and also between CCCS and standard

CCS agents. In particular, the test $e_2 = a \bullet bw$ will distinguish between Q and S in 5.2.1. This, alas, give us $Q \overset{\sqsubset}{\sim}_{must} S$: not what we want. Intuitively, *must* testing distinguishes between processes on the basis of nondeterminism and divergence; S contains a degree of nondeterminism which ought to be detected under *must* testing and which should result in S being below Q in the *must* preordering. We need a test which formalizes the question: can we engage in action a successfully and be sure that no other action has occurred? Tests that involve hiding do not quite answer this question. This is the subject of future work.

References

[Bai92] E. J. Baillie. *Towards a Satisfaction Relation between CCS Specifications and their Refinements.* PhD thesis, University of Hertfordshire, 1992.

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

[Mil83] Robin Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science,* 25:267–310, 1983.

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

[Smi91] David Smith. Concurrent CCS: An introduction. Technical Report 126, Hatfield Polytechnic, 1991.