Technical Report

UCAM-CL-TR-78
ISSN 1476-2986

Number 78

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

A complete proof system for SCCS with model assertions

Glynn Winskel

September 1985

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
https://www.cl.cam.ac.uk/

© 1985 Glynn Winskel

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

A Complete Proof System for SCCS with Modal Assertions

Extended Abstract

by

Glynn Winskel

University of Cambridge,
Computer Laboratory,
Corn Exchange Street,
Cambridge CB2 3QG.

Introduction.

This paper presents a proof system for Robin Milner's *Synchronous Calculus of Communicating Systems* (SCCS) with modal assertions. The language of assertions is a fragment of dynamic logic, sometimes called *Hennessy–Milner logic* after they brought it to attention; while rather weak from a practical point of view, its assertions are expressive enough to characterise observation equivalence, central to the work of Milner *et al* on CCS and SCCS. The paper includes a completeness result and a proof of equivalence between an operational and denotational semantics for SCCS. Its emphasis is on the theoretical issues involved in the construction of proof systems for parallel programming languages.

The style of the proof system has been motivated by ideas from denotational semantics. The idea is to model an SCCS process as the set of assertions it satisfies in the modal language. The labelled transition systems of Milner still play a role in determining the logical relationship that exists between the modal assertions. This done, one obtains an *information system* and so a *domain*, in the sense of Dana Scott [S]. But more, the constructs in SCCS induce operations on the information system making it into an algebra. In the framework of information systems these operations are relations of entailment, and presenting proof rules is seen as specifying how to generate these relations effectively. A novelty of the proof system is the way it uses a syntax which mixes assertions from the modal language in with the syntax for SCCS, a perfectly natural thing to do given the way the domain is built–up from assertions. For example, it makes sense to take the parallel composition of two assertions because assertions correspond to finite elements of the domain on which an operation of parallel composition is defined. Recently there has been a great deal of interest in the problem of how to compose modal assertions, in order to deduce the truth of an assertion for a composition of processes from the truth of certain assertions for its components *e.g.* [BKP], [St1,2] and this paper works out those ideas in detail for SCCS with this brand of modal assertions.

This recasts the semantics of SCCS in the traditional framework of Scott–Strachey denotational semantics where one sees, for example, the translation between different semantics for Milner's *bisimulation equivalence* and Hoare's *failure-set equivalence* expressed as an *embedding–projection* pair between domains.

1. The language SCCS.

Assume a set of process variables $x \in \mathrm{Var}$. Assume a set of elementary actions $\alpha \in Act$ forming a *finite* Abelian monoid $(Act, \bullet, 1)$ with *composition* $\bullet$, and *identity* 1. Define $\alpha/\beta = \{\gamma \mid \beta \bullet \gamma = \alpha\}$, the set of β–divisors of α.

The language of SCCS consists of the following terms

$$p ::= \mathbf{O} \mid x \mid \alpha p \mid p + p \mid p \otimes p \mid p\lceil\Lambda \mid recx.p \mid rec^n x.p \mid \Omega$$

where $x \in \mathrm{Var}$, $\alpha \in Act$, Λ is a subset of Act containing 1 and n is a positive integer.

For convenience we have extended SCCS to include numbered terms of the form $rec^n x.p$ and the completely undefined term Ω. Intuitively the label on such a term bounds the number of calls to the recursive definition. This will be useful later when we come to give proofs involving induction on this number. As a useful convention we shall regard $rec^0 x.p$ as being Ω, and sometimes use $rec^\infty x.p$ to mean $recx.p$.

We say a recursive definition $recx.p$ is *well–guarded* when p has the form αq for some term q and action $\alpha \in Act$. However we shall not assume that recursive definitions are well–guarded in general.

Write $\mathbf{P}$ for the set of SCCS terms, and $\mathbf{P}_C$ for the set *closed* of SCCS terms which we shall call *processes*. We call a numbered term a SCCS term in which *all* occurrences of *rec* are labelled by numbers, and write the set of numbered terms as $\mathbf{P}_N$ and the set of closed numbered terms as $\mathbf{P}_{CN}$.

We explain informally the behaviour of the constructs in the language SCCS. The $\mathbf{O}$ term represents the *nil* process which has stopped and refuses to perform any action. The behaviour of Ω will be the same as that of $recx.x$ which is busily doing nothing of interest. A *guarded* process αp first performs the action α to become the process p. A *sum* $p + q$ behaves like p or q. Which branch of a sum is followed will often be determined by the context and what actions the process is restricted to; only in the case when both component processes p and q are able to perform an identity action 1 can the process $p + q$ always choose autonomously, no matter what the context, to behave like p or q. A *product* process $p \otimes q$ behaves like p and q set in parallel but in such a way that they perform their actions synchronously, in "lock–step", together performing the $\bullet$–product of their respective actions. (To avoid confusion later, we have chosen a notation different from Milner's, using $\otimes$ instead of $\times$.) The *restriction* $p\lceil\Lambda$ behaves like the process p but with its actions restricted to lie in the set Λ. Restriction is a surprisingly powerful construction; it determines what kind of communications are allowed between processes, and without it two processes in parallel would behave in a manner completely independent of eachother. We present the formal definition of behaviour in the next section.

Write $\mathrm{FV}(p)$ for the set of free variables of a term p.

A *substitution* is a map $\sigma : \mathrm{Var} \to \mathbf{P}$ assigning SCCS terms to variables. Given an SCCS term p and a substitution σ the term $p[\sigma]$ is the result of substituting $\sigma[\![x]\!]$ for each free occurrence of x in p—we assume changes are made in the naming of bound variables to avoid the binding of free variables in the substituted terms. We use $[p_0/x_1, \cdots, p_m/x_m, \cdots]$ as an abbreviation for the substitution which replaces free occurrences of the variables x_m by the terms p_m while leaving the other free variables the same.

Let p be a term. A *valuation* is a substitution $\vartheta : \mathrm{Var} \to \mathbf{P}_C$ which assigns a *closed* SCCS term to each variable. So, of course, $p[\vartheta]$ is a closed SCCS term.

2. The behaviour of SCCS.

Following Milner [M1,2,3], the behaviour of a process is represented as a labelled transition system. Its states are processes and so the transition system can be given in a syntax–directed way by defining inductively those transitions which are possible from each process term.

2.1 Definition.
Define the labelled transition relations $\xrightarrow{\alpha}$, for $\alpha \in Act$, between closed SCCS terms to be the least relation closed under the following rules:

$$\alpha p \xrightarrow{\alpha} p$$

$$\frac{p \xrightarrow{\alpha} p'}{p + q \xrightarrow{\alpha} p'} \qquad \frac{q \xrightarrow{\alpha} q'}{p + q \xrightarrow{\alpha} q'}$$

$$\frac{p \xrightarrow{\alpha} p' \quad q \xrightarrow{\beta} q'}{p \otimes q \xrightarrow{\alpha \bullet \beta} p' \otimes q'}$$

$$\frac{p \xrightarrow{\lambda} q}{p[\Lambda \xrightarrow{\lambda} q[\Lambda} \quad \text{if } \lambda \in \Lambda$$

$$\frac{p[recx.p/x] \xrightarrow{\alpha} q}{recx.p \xrightarrow{\alpha} q} \qquad \frac{p[rec^n x.p/x] \xrightarrow{\alpha} q}{rec^{n+1} x.p \xrightarrow{\alpha} q}$$

Notice there are no rules for $\mathbf{O}$ or Ω because we do not wish there to be any transitions from such terms. Because the relations $\xrightarrow{\alpha}$, $\alpha \in Act$, are defined to be the least relations given by the rules it follows that *e.g.*

$$p \otimes q \xrightarrow{\alpha} r \Leftrightarrow \exists p', q', \beta, \gamma.\ \beta \bullet \gamma = \alpha \ \&\ r = p' \otimes q' \ \&\ p \xrightarrow{\beta} p' \ \&\ q \xrightarrow{\gamma} q' \quad \text{and}$$
$$recx.p \xrightarrow{\alpha} r \Leftrightarrow p[recx.p/x] \xrightarrow{\alpha} r.$$

A process *diverges* if it can be forever busy performing internal events. In the case of SCCS this can only arise through a process unwinding its recursive definition continually.

A diverging process has an unsettled status. In the absence of communication with the environment, it never settles down into a stable state, or settles on the full set of actions it is prepared to do. Viewed behaviourally, from the outside so to speak, it continues to "click and whir" and it never becomes clear whether an action refused now will necessarily be refused later. Mathematically it is the complementary notion of *convergence* which has the more basic definition, by induction.

2.2 Definition. Define the predicates $\downarrow$ on $\mathbf{P}_G$ to be the least predicate such that

$$\begin{gathered}
\mathbf{O}\downarrow, \quad \alpha p\downarrow, \\
p\downarrow \ \& \ q\downarrow \Rightarrow (p+q)\downarrow, \\
p\downarrow \ \& \ q\downarrow \Rightarrow (p \otimes q)\downarrow, \\
p\downarrow \Rightarrow (p\lceil \Lambda)\downarrow, \\
(p[recx.p/x])\downarrow \Rightarrow (recx.p)\downarrow, \\
(p[rec^l x.p/x])\downarrow \Rightarrow (rec^{l+1} x.p)\downarrow .
\end{gathered}$$

where p and q are closed SCCS terms and l is an non–negative integer.

Say a closed SCCS term p is *convergent* iff $p\downarrow$.

Say a closed term p *diverges*, and write $p\uparrow$, when p does not converge.

Intuitively a divergent term is one whose transitions are not completely specified by a finite stage in the recursion. If all recursions were assumed to be well–guarded then all closed terms but Ω would be convergent. Note $\alpha\Omega$ converges and so does $recx.1x$.

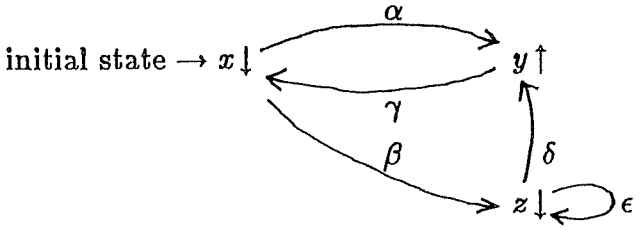
Thus an SCCS process p determines a labelled transition system, $(S, p, \{\xrightarrow{\alpha}\}_{\alpha\in Act}, \uparrow)$, with states S those processes reachable from the initial state p, and in which some states are distinguished as being divergent. For example, the process

$$p = recx.\alpha(\beta x + \gamma\Omega)$$

describes the transition system:

$$\text{initial state} \rightarrow p\downarrow \overset{\beta}{\curvearrowleft} \xrightarrow{\alpha} (\beta p + \gamma\Omega)\downarrow \xrightarrow{\gamma} \Omega\uparrow$$

In fact, it is not hard to see that any finite labelled transition system like this, with labels from the set Act and some states distinguished as divergent, can be described by an SCCS process. For example, the transition system



is described by the process

$$recx.\ (\alpha recy.\ [\Omega + \gamma x] + \beta recz.\ [\delta recy.\ [\Omega + \gamma x] + \epsilon z]),$$

obtained by successively eliminating variables in favour of the recursive definitions they satisfy, making states divergent by summing them with Ω. (We have taken the liberty of using state names as variables). More generally, along these lines one can show:

2.3 Lemma. *Any finite transition system, $(S, p, \{\xrightarrow{\alpha}\}_{\alpha \in Act}, \uparrow)$, can be described by an SCCS process p.*

The number attached to occurrences of *rec* specifies how many times the recursive definition can be unwound when determining the transition system associated with a term. Roughly, the larger the numbers the larger the transition system associated with the term. There corresponds an approximation order between terms which we write as $\leq$.

2.4 Definition. Define $\leq$ to be the least binary relation on $\mathbf{P}_C$ such that

$$\begin{aligned}
&\Omega \leq p, \qquad p \leq p \\
&p \leq q \Rightarrow \alpha p \leq \alpha q \\
&p \leq p' \ \ \& \ \ q \leq q' \Rightarrow p + q \leq p' + q' \\
&p \leq p' \ \ \& \ \ q \leq q' \Rightarrow p \otimes q \leq p' \otimes q' \\
&p \leq q \Rightarrow p\lceil \Lambda \leq q\lceil \Lambda \\
&p \leq q \ \ \& \ \ m \leq n \Rightarrow rec^m x.p \ \leq \ rec^n x.q \\
&p \leq q \Rightarrow rec^n x.p \ \leq \ recx.q.
\end{aligned}$$

The order $\leq$ on terms $\mathbf{P}_C$ respects the language of SCCS, as expressed in the following lemma.

2.5 Lemma. *Let ϑ and ϑ' be valuations in the relation $\vartheta \leq \vartheta' \Leftrightarrow_{def} \forall x \in Var.\ \vartheta[\![x]\!] \leq \vartheta'[\![x]\!]$. Let p, q be terms in the relation $p \leq q$. Then $p[\vartheta] \leq q[\vartheta']$.*

Proof. By structural induction. ∎

3. The assertion language.

Hennessy and Milner defined an equivalence relation between processes called *observational equivalence* in [HM, M1]. For our language of SCCS, two processes are observationally equivalent iff whenever one can do an action to become a process then so can the other do the same action to become an equivalent process. They found an alternative characterisation so that processes were observationally equivalent iff they satisfied the same assertions in a simple language of modal assertions [HM]. However there are inadequacies

in this treatment of processes because it does not take proper account of divergence. So Milner, in [M2], generalised the definition of observational equivalence and the definition of a process satisfying a modal assertion in order to cope with divergence. (See [HP] for a closely related but different extension of observational equivalence to diverging processes.) In this way Milner extended the result he had obtained with Hennessy, so that in SCCS, for example, two processes are observationally equivalent iff they satisfy the same assertions in the modal language of Hennessy and Milner. In future, in this paper, "observational equivalence" shall refer to the more refined equivalence of [M2]. Following [P, St1,2] we have simplified the modal language of Hennessy and Milner a little.

3.1 Definition.
The *assertion language* consists of simple modal expressions built up according to:

$$A ::= \mathit{true} \mid \mathit{false} \mid \bigwedge_{i\in I} A_i \mid \bigvee_{i\in I} A_i \mid \langle\alpha\rangle A \mid [\alpha]A$$

where I is a finite indexing set and $\alpha \in \mathit{Act}$.

We shall call elements of this language *assertions*, and write the set of assertions as **Assn**.

By convention we understand $\bigwedge_{i\in I} A_i$ to be *true* and $\bigvee_{i\in I} A_i$ to be *false* when the indexing set I is null. When the indexing set is $I = \{0, 1\}$ we can write $\bigwedge_{i\in I} A_i$ as $A_0 \wedge A_1$, and $\bigvee_{i\in I} A_i$ as $A_0 \vee A_1$.

The meaning of an assertions is given by specifying the subset $\Pi[\![A]\!]$ of SCCS processes $\mathbf{P}_C$ which satisfy A:

3.2 Definition. Define

$$\begin{aligned}
\Pi[\![\mathit{true}]\!] &= \mathbf{P}_C \\
\Pi[\![\mathit{false}]\!] &= \emptyset \\
\Pi[\![\bigwedge_{i\in I} A_i]\!] &= \bigcap_{i\in I} \Pi[\![A_i]\!] \\
\Pi[\![\bigvee_{i\in I} A_i]\!] &= \bigcup_{i\in I} \Pi[\![A_i]\!] \\
\Pi[\![\langle\alpha\rangle A]\!] &= \{p \in \mathbf{P}_C \mid \exists q.p \xrightarrow{\alpha} q \ \&\ q \in \Pi[\![A]\!]\} \\
\Pi[\![[\alpha]A]\!] &= \{p \in \mathbf{P}_C \mid p\!\downarrow \ \&\ \forall q.p \xrightarrow{\alpha} q \Rightarrow q \in \Pi[\![A]\!]\}
\end{aligned}$$

Write $p \models A \Leftrightarrow_{def} p \in \Pi[\![A]\!]$, where p is a SCCS process and A is an assertion, and say p *satisfies* A.

Remark. Let $T = (S, p, \{\xrightarrow{\alpha}\}_{\alpha\in Act}, \dagger)$ be a transition system as described in section 1. It is obvious how to define a satisfaction relation $T \models A$, between transition system T and modal assertions A, in such a way that a process satisfies an assertion iff its associated transition system does.

Clearly $p \models \langle\alpha\rangle A$ means the process p can do an α–action to become a process satisfying A, and $p \models [\alpha]\mathit{false}$ means the process p refuses to do an α–action. The latter kind of properties are important for detecting deadlock. Notice that $\Omega \not\models [\alpha]\mathit{true}$ and $\Omega \not\models [\alpha]\mathit{false}$ because we insist diverging processes, like Ω, cannot satisfy any assertion of the form $[\alpha]A$.

3.3 Proposition. *Let $p \in \mathbf{P}_C$. Then $p\downarrow \Leftrightarrow p \models [\alpha]\mathit{true}$, for any action α.*

Because we insist that a process satisfying a modal assertion $[\alpha]A$ must converge, satisfaction will be effective; if a process p in $\mathbf{P}_C$ satisfies an assertion A then it can be approximated by a numbered version p' which also satisfies the assertion. To show this we must first see how the transition system associated with a term $p' \leq p$ approximates, and simulates, the transition system associated with p.

3.4 Lemma. *For SCCS processes*

(i) For $p, p', q' \in \mathbf{P}_C$
$$p' \xrightarrow{\alpha} q' \;\&\; p' \leq p \Rightarrow \exists q.\; q' \leq q \;\&\; p \xrightarrow{\alpha} q.$$

(ii) For $p, q \in \mathbf{P}_C, q_0 \in \mathbf{P}_{CN}$
$$p \xrightarrow{\alpha} q \;\&\; q_0 \leq q \Rightarrow \exists p', q' \in \mathbf{P}_{CN}.\; p' \leq p \;\&\; p' \xrightarrow{\alpha} q' \;\&\; q_0 \leq q'.$$

(iii) For $p, p' \in \mathbf{P}_C$
$$p'\downarrow \;\&\; p' \leq p \Rightarrow p\downarrow \;\&\; (\forall q.\; p \xrightarrow{\alpha} q \Rightarrow \exists q' \leq q.\; p' \xrightarrow{\alpha} q').$$

(iv) For $p \in \mathbf{P}_C, Y \subseteq \mathbf{P}_{CN}$
$$p\downarrow \;\&\; (\forall q.\; p \xrightarrow{\alpha} q \Rightarrow \exists q_0 \in Y.\; q_0 \leq q)$$
$$\Rightarrow \exists p' \in \mathbf{P}_{CN}.\; p' \leq p \;\&$$
$$p'\downarrow \;\&\; (\forall q.\; p' \xrightarrow{\alpha} q \Rightarrow \exists q_0 \in Y.\; q_0 \leq q).$$

Proof. The proofs follow by induction on the inductive definitions of $\{\xrightarrow{\alpha}\}_{\alpha \in Act}$ and $\downarrow$. ∎

Note that part (iv) above specialises to the result

$$p\downarrow \Rightarrow \exists p' \in \mathbf{P}_{CN}.\; p' \leq p \;\&\; p'\downarrow$$

when we take $Y = \{\Omega\}$. Using parts (i) and (ii) for the modalities $\langle\alpha\rangle$ and (iii) and (iv) for the modality $[\alpha]$ one can prove:

3.5 Theorem. *Let $p \in \mathbf{P}_C$. Then*

$$p \models A \Leftrightarrow \exists p' \in \mathbf{P}_{CN}.\; p' \leq p \;\&\; p' \models A.$$

We can see the results above in topological terms. There is a natural topology on processes which is the Scott–topology.

3.6 Proposition. *The family of sets of the form* $\{p \in \mathbf{P}_C \mid q \leq p\}$ *for* q *a closed numbered term are the basis of a topology on* $\mathbf{P}_C$. *So the open sets have the form*

$$U = \{p \in \mathbf{P}_C \mid \exists p_0 \in X.\ p_0 \leq p\}$$

for a subset X *of closed numbered terms.*

The open sets of $\mathbf{P}_C$ *are those subsets* $U \subseteq \mathbf{P}_C$ *which are*

(i) $\forall p, q.\ p \geq q \in U \Rightarrow p \in U$,

(ii) $\forall$ *directed* $S \subseteq \mathbf{P}_C.\ \bigsqcup S \in U \Rightarrow \exists p \in S.\ p \in U$.

Then theorem 3.5 says each assertion determines an open set of $\mathbf{P}_C$ *i.e.* $\Pi[\![A]\!]$ is open for each assertion A. In fact 3.5 can be generalised, important were we to extend our present language of assertions.

3.7 Theorem. *Let* $\alpha \in Act$. *If* U *is an open set in the topology on processes then so are the sets*

$$\langle\alpha\rangle U =_{def} \{p \in \mathbf{P}_C \mid \exists q \in U.\ q \xrightarrow{\alpha} p\} \quad \textit{and}$$
$$[\alpha]U =_{def} \{p \in \mathbf{P}_C \mid\ p\downarrow\ \&\ \forall q.\ p \xrightarrow{\alpha} q \Rightarrow q \in U\}.$$

This topological view is in line with Dana Scott's development of the theory of domains from neighbourhood systems [S1] and with the ideas of Mike Smyth in [Sm], where it is proposed that computational properties of a topological space be identified with effective open sets. In the approach to domains using neighbourhood systems, to know more information about a process is to know a smaller neighbourhood in which it is contained. These topological ideas have been applied by Gordon Plotkin in [P] to extend the language of assertions and its interpretation to cover intuitionistic negation and implication; their interpretation are those standard for topological models of intuitionistic logic, so in this extension of **Assn** one takes $\Pi[\![A \supset B]\!] = ((\mathbf{P}_C \setminus \Pi[\![A]\!]) \cup \Pi[\![B]\!])^o$ where X^o is the topological interior of the set X (Plotkin's topology is not that here however). One advantage of intuitionistic logic over classical logic is that satisfaction is still effective even for this extended set of assertions. We shall say more on denotational semantics in section 5.

4. The decomposition of assertions.

We are interested in how the goal of proving an assertion holds of a process reduces to the subgoals of proving assertions about its subprocesses, and in the converse problem, of how assertions about subprocesses combine to yield assertions about the compound process. It is clear for example that an assertion $\langle\alpha\rangle A$ holds of a process αp iff A holds of p. Similarly $[\alpha]A$ holds of a process $p + q$ iff $[\alpha]A$ holds of both components p and q. However $p + q \models \langle\alpha\rangle A$ iff $p \models \langle\alpha\rangle A$ or $q \models \langle\alpha\rangle A$; there is not a unique subgoal. Similarly

there are many possible ways in which $p \otimes q \models \langle\alpha\rangle true$; this holds whenever $p \models \langle\beta\rangle true$ and $q \models \langle\gamma\rangle true$ with $\beta \bullet \gamma = \alpha$.

For each unary operation *op* of SCCS we show how for an assertion A there is an assertion $\mathcal{D}_{op}[\![A]\!]$ so that

$$op\,(p) \models A \Leftrightarrow p \models \mathcal{D}_{op}[\![A]\!].$$

For each binary operation *op* of SCCS we show how for an assertion A there is a finite set of pairs of assertions $\mathcal{D}_{op}[\![A]\!]$ so that

$$p\ op\ q \models A \quad \text{iff} \quad \exists (B,C) \in \mathcal{D}_{op}[\![A]\!].\ p \models B\ \ \&\ \ q \models C.$$

Thus we see how, with respect to each operation *op* in SCCS, every assertion has a *decomposition* which reduces the problem of proving the assertion holds of a compound process built-up using *op* to proving assertions about its components. These results provide the foundations of our proof system for SCCS with assertions **Assn**. All the proofs of this section are by structural induction.

The guarded–decomposition of assertions:

4.1 Definition. Let $\alpha \in Act$. Define the assertion $\mathcal{D}_\alpha[\![A]\!]$, for an assertion A, by the structural induction:

$$\begin{aligned}
\mathcal{D}_\alpha[\![true]\!] &= true \\
\mathcal{D}_\alpha[\![false]\!] &= false \\
\mathcal{D}_\alpha[\![\textstyle\bigwedge_{i\in I} A_i]\!] &= \textstyle\bigwedge_{i\in I} \mathcal{D}_\alpha[\![A_i]\!] \\
\mathcal{D}_\alpha[\![\textstyle\bigvee_{i\in I} A_i]\!] &= \textstyle\bigvee_{i\in I} \mathcal{D}_\alpha[\![A_i]\!] \\
\mathcal{D}_\alpha[\![\langle\beta\rangle A]\!] &= \begin{cases} A & \text{if } \beta = \alpha \\ false & \text{if } \beta \neq \alpha \end{cases} \\
\mathcal{D}_\alpha[\![[\beta]A]\!] &= \begin{cases} A & \text{if } \beta = \alpha \\ true & \text{if } \beta \neq \alpha. \end{cases}
\end{aligned}$$

4.2 Theorem. *Let $\alpha \in Act$. Let A be an assertion.*

$$\forall p \in \mathbf{P}_C.\ \alpha p \models A \Leftrightarrow p \models \mathcal{D}_\alpha[\![A]\!].$$

The sum–decomposition of assertions:

4.3 Definition. Define $\mathcal{D}_+[\![A]\!]$ by structural induction on the assertion A:

$$\begin{aligned}
\mathcal{D}_+[\![true]\!] &= \{(true, true)\} \\
\mathcal{D}_+[\![false]\!] &= \{(true, false), (false, true)\} \\
\mathcal{D}_+[\![\bigwedge_{i\in I} A_i]\!] &= \{(\bigwedge_{i\in I} A_{i0}, \bigwedge_{i\in I} A_{i1}) \mid \forall i \in I.(A_{i0}, A_{i1}) \in \mathcal{D}_+[\![A_i]\!])\} \\
\mathcal{D}_+[\![\bigvee_{i\in I} A_i]\!] &= \bigcup_{i\in I} \mathcal{D}_+[\![A_i]\!] \\
\mathcal{D}_+[\![\langle\alpha\rangle A]\!] &= \{(\langle\alpha\rangle A, true), (true, \langle\alpha\rangle A)\} \\
\mathcal{D}_+[\![[\alpha]A]\!] &= \{([\alpha]A, [\alpha]A)\}.
\end{aligned}$$

4.4 Theorem. *For all p and q in $\mathbf{P}_C$*

$$p + q \models A \Leftrightarrow \exists (B, C) \in \mathcal{D}_+[\![A]\!].\ p \models B \ \ \&\ \ q \models C.$$

The parallel-decomposition of assertions:

The problem of decomposition for $\otimes$ is a little more difficult.

4.5 Definition. Define $\mathcal{D}_\otimes[\![A]\!]$ by structural induction on the assertion A:

$$\begin{aligned}
\mathcal{D}_\otimes[\![true]\!] &= \{(true, true)\} \\
\mathcal{D}_\otimes[\![false]\!] &= \{(true, false), (false, true)\} \\
\mathcal{D}_\otimes[\![\bigwedge_{i\in I} A_i]\!] &= \{(\bigwedge_{i\in I} A_{i0}, \bigwedge_{i\in I} A_{i1}) \mid \forall i \in I.(A_{i0}, A_{i1}) \in \mathcal{D}_\otimes[\![A_i]\!])\} \\
\mathcal{D}_\otimes[\![\bigvee_{i\in I} A_i]\!] &= \bigcup_{i\in I} \mathcal{D}_\otimes[\![A_i]\!] \\
\mathcal{D}_\otimes[\![\langle\alpha\rangle A]\!] &= \{(\langle\beta\rangle B, \langle\gamma\rangle C) \mid \beta \bullet \gamma = \alpha \ \ \&\ \ (B, C) \in \mathcal{D}_\otimes[\![A]\!]\} \\
\mathcal{D}_\otimes[\![[\alpha]A]\!] &= \text{ the set of pairs}
\end{aligned}$$

$$\Big(\bigwedge_{\beta\in Act} [\beta] \bigvee_{i\in I_\beta} \bigwedge_{\gamma\in\alpha/\beta} \bigwedge_{j\in J_\gamma} B_{ij}^{\beta\gamma},\quad \bigwedge_{\gamma\in Act} [\gamma] \bigvee_{j\in J_\gamma} \bigwedge_{\beta\in\alpha/\gamma} \bigwedge_{i\in I_\beta} C_{ij}^{\beta\gamma}\Big)$$

such that

$$\beta \bullet \gamma = \alpha \Rightarrow (B_{ij}^{\beta\gamma}, C_{ij}^{\beta\gamma}) \in \mathcal{D}_\otimes[\![A]\!].$$

4.6 Theorem. *For all p and q in $\mathbf{P}_C$*

$$p \otimes q \models A \Leftrightarrow \exists (B, C) \in \mathcal{D}_\otimes[\![A]\!].\ p \models B \ \ \&\ \ q \models C.$$

The restriction–decomposition of assertions:

We can associate with any assertion A an *assertion* $\mathcal{D}_{\lceil\Lambda}[\![A]\!]$ so that A is satisfied by $p\lceil\Lambda$ iff $\mathcal{D}_{\lceil\Lambda}[\![A]\!]$ is satisfied by p.

4.7 Definition. Let Λ be a subset of *Act* containing 1. Define $\mathcal{D}_{\lceil\Lambda}[\![A]\!]$, for an assertion A, by the structural induction:

$$\begin{aligned}
\mathcal{D}_{\lceil\Lambda}[\![\mathit{true}]\!] &= \mathit{true}\\
\mathcal{D}_{\lceil\Lambda}[\![\mathit{false}]\!] &= \mathit{false}\\
\mathcal{D}_{\lceil\Lambda}[\![\bigwedge_{i\in I} A_i]\!] &= \bigwedge_{i\in I} \mathcal{D}_{\lceil\Lambda}[\![A_i]\!]\\
\mathcal{D}_{\lceil\Lambda}[\![\bigvee_{i\in I} A_i]\!] &= \bigvee_{i\in I} \mathcal{D}_{\lceil\Lambda}[\![A_i]\!]\\
\mathcal{D}_{\lceil\Lambda}[\![\langle\alpha\rangle A]\!] &= \begin{cases} \langle\alpha\rangle\mathcal{D}_{\lceil\Lambda}[\![A]\!] & \text{if } \alpha\in\Lambda\\ \mathit{false} & \text{if } \alpha\notin\Lambda \end{cases}\\
\mathcal{D}_{\lceil\Lambda}[\![[\alpha]A]\!] &= \begin{cases} [\alpha]\mathcal{D}_{\lceil\Lambda}[\![A]\!] & \text{if } \alpha\in\Lambda\\ [\alpha]\mathit{true} & \text{if } \alpha\notin\Lambda. \end{cases}
\end{aligned}$$

One clause of the above definition may be puzzling. Why do we take $\mathcal{D}_{\lceil\Lambda}[\![[\alpha]A]\!] = [\alpha]\mathit{true}$ if $\alpha\notin\Lambda$ rather than taking it to be simply the assertion *true*? The answer: because of divergence. For example, because Ω diverges, $\Omega\lceil\Lambda \not\models [\alpha]A$ while $\Omega \models \mathit{true}$.

4.8 Theorem. *Let $p\in\mathbf{P}_C$ and A be an assertion. Then*

$$p\lceil\Lambda \models A \Leftrightarrow p \models \mathcal{D}_{\lceil\Lambda}[\![A]\!].$$

5. From a denotational point of view.

So far our presentation has been based on the operational semantics of SCCS processes. To summarise, we have modelled each SCCS process as a labelled transition system in which some nodes are distinguished as being convergent. Treating the transition system as a Kripke model we defined the satisfaction relation between processes in $\mathbf{P}_C$ and modal assertions **Assn**.

If the language of modal assertions captures all the nature of a process that is of interest to us then it is natural to regard two processes as equivalent iff they satisfy exactly the same assertions. We write

$$p\approx q \quad \text{iff} \quad \forall A\in\mathbf{Assn}.\ p\models A \Leftrightarrow q\models A$$

for closed SCCS terms p and q. Because we assume the set of actions *Act* is finite, it follows from Milner's result in [M2] that $\approx$ coincides with his extended notion of *observational equivalence* and that of *bisimulation equivalence* due to Park [M3, Pa], again extended to cope with divergence. In fact for the language SCCS this equivalence is a congruence too.

We can go further and take the more radical view that a process can be identified with the set of assertions it satisfies, so a process p is identified with the set

$$\mathcal{A}[\![p]\!] = \{A \in \mathbf{Assn} \mid p \models A\}.$$

This step takes us into the realm of denotational semantics, with its own approach and techniques.

To see this first notice the obvious relation of entailment between (open) sets of processes, thought of as *properties*. One property $U \subseteq \mathbf{P}_C$ entails another $V \subseteq \mathbf{P}_C$ iff $U \subseteq V$. Spelt out, $U \subseteq V$ simply says that every process which satisfies U also satisfies V. This induces an entailment relation between assertions.

5.1 Definition. Let $\succcurlyeq$ be the binary relation between assertions given by:

$$A \succcurlyeq B \Leftrightarrow_{def} \Pi[\![A]\!] \subseteq \Pi[\![B]\!].$$

Now as is well known from *e.g.* the work on *information systems* [S, LW], and the specific applications in *e.g.* [Gol], there is a *domain* associated with this entailment on modal assertions. Its elements, like $\mathcal{A}[\![p]\!]$, are sets of assertions which are consistent and closed under entailment. Ordered by inclusion these form a Scott domain of information in which more information corresponds to more assertions being true. For our purposes it is more natural to allow sets of assertions inconsistent assertions which entail *false*.

5.2 Definition. Define the partial order $\mathbf{D}$ to consist of the following elements ordered by inclusion. The *elements* of $\mathbf{D}$ are those subsets $a \subseteq \mathbf{Assn}$ which are $\succcurlyeq$*–closed:*

$$\forall B \forall X \subseteq a.\ \bigwedge X \succcurlyeq B \Rightarrow B \in a.$$

5.3 Proposition. *The set* $\mathbf{D}$ *ordered by inclusion forms an* ω*–algebraic complete lattice. Its finite elements are precisely those elements of the form* $\overline{A} =_{def} \{B \in \mathbf{Assn} \mid A \succcurlyeq B\}$ *for some assertion* A*. The least element* $\bot$ *is* $\overline{true}$ *and the greatest element is* $\top = \overline{false}$*.*

Let p *be an SCCS process. The set* $\{A \mid p \models A\}$ *is an element of* $\mathbf{D}$*.*

Of course the method by which we have obtained the *denotation* $\mathcal{A}[\![p]\!] = \{A \in \mathbf{Assn} \mid p \models A\}$ of a process p is rather roundabout, in contrast to the usual direct way of giving denotational semantics. Generally in denotational semantics

one reflects the constructs in the language as operations on a complete partial order and uses least fixed points to give a meaning to recursive definitions. To follow this more traditional route we must first define operations on the elements **D** to correspond to the syntactic operations of guarding, sum, product and restriction. The operations are determined by further relations of entailment between properties which we define first.

5.4 Definition. For U and V be open sets in the topology on processes, and $\alpha \in Act$ and $\Lambda \subseteq Act$ with $1 \in \Lambda$, define

$$\begin{aligned} \alpha U &= \{\alpha p \mid p \in U\} \\ U + V &= \{p + q \mid p \in U \ \& \ q \in V\} \\ U \otimes V &= \{p \otimes q \mid p \in U \ \& \ q \in V\} \\ U\lceil\Lambda &= \{p\lceil\Lambda \mid p \in U\}. \end{aligned}$$

From the definition of $\leq$ and the topology on processes we see the above are indeed well–defined operations on open sets. In section 5 we have seen how the truth of an assertion in a compound process reduces to certain assertions holding of the component processes. For example $p \otimes q \models A$ iff $p \models B$ and $q \models C$ for some $(B, C) \in \mathcal{D}_{\otimes}[\![A]\!]$. Thus in particular if B is true of process p and C is true of process q then A is true of $p \otimes q$. Clearly this relation can be expressed as $\Pi[\![B]\!] \otimes \Pi[\![C]\!] \subseteq \Pi[\![A]\!]$, which we can write as $B \otimes C \gg A$. Following this style we define relations between assertions. (They correspond to *approximable mappings* on the information system of assertions. And then these induce continuous operations on the domain of elements **D**, making it into a continuous algebra.)

5.5 Definition. Let A, B, C range over **Assn**. Let $\alpha \in Act$ and let $\Lambda \subseteq Act$ be such that $1 \in \Lambda$.

(i)Define a unary relation on assertions by $\mathbf{O} \gg A \Leftrightarrow_{def} \mathbf{O} \models A$.
(ii)Define a binary relation on assertions by $\alpha B \gg A \Leftrightarrow_{def} \alpha\Pi[\![B]\!] \subseteq \Pi[\![A]\!]$.
(iii)Define a ternary relation on assertions by $B + C \gg A \Leftrightarrow_{def} \Pi[\![B]\!] + \Pi[\![C]\!] \subseteq \Pi[\![A]\!]$.
(iv)Define a ternary relation on assertions by $B \otimes C \gg A \Leftrightarrow_{def} \Pi[\![B]\!] \otimes \Pi[\![C]\!] \subseteq \Pi[\![A]\!]$.
(v)Define a binary relation on assertions by $B\lceil\Lambda \gg A \Leftrightarrow_{def} \Pi[\![B]\!]\lceil\Lambda \subseteq \Pi[\![A]\!]$.

The domain **D** consists of sets of assertions. Once we extend the relations to continuous operations on **D**, we can give a denotation to each term of **P** with respect to an environment in the standard way. However note that we have an added freedom. Not only do processes sit naturally in **D**, so too of course do assertions; any assertion A can be identified with the element $\overline{A} =_{def} \{B \mid A \gg B\}$. Consequently we can make sense in **D** of syntax like $A \otimes B$ where A and B are assertions, or even more free mixes of the assertion language and the programming language like, for example, $recx.\ (x \otimes \langle\alpha\rangle[\beta]true + \gamma true)$. Note in the latter example assertions only occur at the roots of the syntax tree. Here we shall not take the more liberal step of having a complete mix of program syntax and assertion language syntax. But this should not be taken to indicate any deep prejudice against such things, or as far as I can see any real difficulties—it's easy to extend the

modal operators to elements of $\mathbf{D}$, *e.g.* take $[\alpha]a = \{B \mid \exists A \in a.\ [\alpha]A \geqslant B\}$. The work of the next section only requires the following enrichment of SCCS by assertions **Assn**.

5.6 Definition. Define the language of SCCS *with assertions* by

$$p ::= A \mid \mathbf{O} \mid x \mid \alpha p \mid p+p \mid p \otimes p \mid p\lceil\Lambda \mid recx.p \mid rec^n x.p \mid \Omega$$

where $A \in \mathbf{Assn}$, and $x \in \mathrm{Var}$, $\alpha \in Act$, Λ is a subset of Act containing 1 and n is a positive integer.

Write **AP** for the language of SCCS with assertions.

5.7 Definition. *Denotational semantics of SCCS with assertions:*

Define an *environment* to be a function $\rho : \mathrm{Var} \to \mathbf{D}$. Write $\rho[a/x]$, where ρ is an environment, $x \in \mathrm{Var}$ and $a \in \mathbf{D}$, for the environment which results by replacing ρ's value on x by a.

Define $\mathcal{A}[\![p]\!]\rho$ for a term $p \in \mathbf{AP}$ and any environment ρ by the structural induction:

$$\begin{aligned}
&\mathcal{A}[\![A]\!]\rho = \{B \mid A \geqslant B\} \quad \text{where } A \in \mathbf{Assn}\\
&\mathcal{A}[\![\mathbf{O}]\!]\rho = \{A \mid \mathbf{O} \geqslant A\}\\
&\mathcal{A}[\![x]\!]\rho = \rho[\![x]\!]\\
&\mathcal{A}[\![\alpha p]\!]\rho = \{A \mid \exists B \in \mathcal{A}[\![p]\!]\rho.\ \alpha B \geqslant A\} \quad \text{where } \alpha \in Act\\
&\mathcal{A}[\![p+q]\!]\rho = \{A \mid \exists B \in \mathcal{A}[\![p]\!]\rho, C \in \mathcal{A}[\![q]\!]\rho.\ B+C \geqslant A\}\\
&\mathcal{A}[\![p \otimes q]\!]\rho = \{A \mid \exists B \in \mathcal{A}[\![p]\!]\rho, C \in \mathcal{A}[\![q]\!]\rho.\ B \otimes C \geqslant A\}\\
&\mathcal{A}[\![p\lceil\Lambda]\!]\rho = \{A \mid \exists B \in \mathcal{A}[\![p]\!]\rho.\ B\lceil\Lambda \geqslant A\} \quad \text{where } 1 \in \Lambda \subseteq Act\\
&\mathcal{A}[\![\Omega]\!]\rho = \bot\\
&\mathcal{A}[\![rec^n x.p]\!]\rho = \Gamma^n(\bot)\\
&\mathcal{A}[\![recx.p]\!]\rho = \mathrm{fix}\ \Gamma
\end{aligned}$$

where $\bot = \{A \mid \mathit{true} \geqslant A\}$ and $\Gamma : \mathbf{D} \to \mathbf{D}$ is given by $\Gamma(a) = \mathcal{A}[\![p]\!]\rho[a/x]$ and fix is the least fixed point operator $\mathrm{fix}\ \Gamma = \bigcup_{i \in \omega} \Gamma^i(\bot)$.

Note how the numbered term $rec^n x.p$ is the syntactic counterpart of the nth iteration of the functional in the construction of the least fixed point of $a \mapsto \mathcal{A}[\![p]\!]\rho[a/x]$. Of course the above denotational semantics specialises to one for pure SCCS by just ignoring the clause for assertions.

5.8 Proposition. *(The denotational semantics is well-defined)*

We have $\mathcal{A}[\![p]\!]\rho \in \mathbf{D}$ for all $p \in \mathbf{P}$ and any environment ρ.

The function $\Gamma : \mathbf{D} \to \mathbf{D}$ in the above definition is continuous, so $\mathrm{fix}\ \Gamma$ is the least fixed point of Γ.

It becomes vital to check that the denotational semantics agrees with the operational semantics we have given earlier. The decomposition results of section 4 play a key role, as

does theorem 3.5 saying a process satisfies an assertion iff there is some numbered term approximating it which satisfies it too.

5.9 Theorem. *Let ϑ be a valuation. Let $\hat{\vartheta}$ be the associated environment*

$$\hat{\vartheta}[\![x]\!] = \{A \mid \vartheta[\![x]\!] \models A\}.$$

Then for $p \in \mathbf{P}$,

$$\mathcal{A}[\![p]\!]\hat{\vartheta} = \{A \mid p[\vartheta] \models A\}.$$

In particular if p is a closed term then

$$\mathcal{A}[\![p]\!]\rho = \{A \mid p \models A\}$$

for an arbitrary environment ρ.

Proof. By structural induction on p, with an inner induction on n in the case where p has the form $rec^n x.q$, and by invoking 3.5 when p has the form $recx.q$. ∎

The decomposition results of section 4 play a vital role in the above proof. They can be seen as ensuring the language of assertions meets an *expressiveness criterion*; that there are enough assertions so that any differences in those assertions satisfied by the composition of processes is detected as differences in the assertions satisfied by the components (a property whose failure for CCS led Stirling to introduce another modality which he wrote as ○ [St2]).

Thus the denotational semantics is in perfect agreement with the work of the previous sections. Of course the entailment relation and operations on the information system of assertions have been derived from the satisfaction relation between processes and one would like an independent construction of them. But this is the job of a proof theory and is tackled in the next section.

We can generalise the satisfaction relation to elements of **AP** , assured it is consistent with our previous use.

5.10 Definition. Let $p \in \mathbf{AP}$. Let $A \in \mathbf{Assn}$. Define

$$p \models A \Leftrightarrow \forall \text{ environments } \rho.\ A \in \mathcal{A}[\![p]\!]\rho.$$

Of course one can take issue with the view that the assertions **Assn** capture all those basic properties of importance that can be noted about a process. The work of Milner et al, *e.g.* [M1,3], shows how much can be done with the observational and bisimulation equivalence induced by the assertions. This argues that the assertions are sufficiently rich to capture a great many of the properties of interest. This should not seem so surprising.

Remember a process denotes the set of assertions it satisfies so is essentially modelled as an (infinite) conjunction of these assertions; only for a finite process could a single assertion in **Assn** capture its full behaviour.

Although the assertions may make it possible to distinguish all the processes one could wish, this is not to say the logic is as expressive as one would like from all points of view. Clearly it is rather primitive. For example one would like the ability to specify infinite behaviours by finite assertions.

Quite possibly there are other properties of interest to which the language of assertions is blind. However it is interesting that two other well-known notions of equivalence can be induced by taking fragments of the assertion language **Assn**. They are *trace equivalence* and *failure-set equivalence*. Strictly speaking the failure-set equivalence has not been defined on SCCS but the definition that follows has been based on the work of [HBR] modified to take proper account of divergence. The use of traces and their associated equivalence is widespread, see *e.g.* [H] and [HdeN]. As far as these two equivalences are concerned **Assn** is certainly expressive enough. It is a pleasing fact that the domains of assertions for these two equivalences are related to the domain **D** and to each other by the classical notion of embedding-projection pairs. This is because the embedding-projection pairs between domains are associated with restriction in their representation as information systems (see [LW]). We define the corresponding fragments of **Assn**. Define $\mathbf{Assn}_T$, *the trace assertions*, to consist of all those assertions of the form

$$\langle\alpha_0\rangle\langle\alpha_1\rangle\cdots\langle\alpha_{j-1}\rangle \mathit{true}.$$

Define $\mathbf{Assn}_{FS}$, *the failure-set assertions*, to consist of all those assertions of the form

$$\langle\alpha_0\rangle\langle\alpha_1\rangle\cdots\langle\alpha_{j-1}\rangle(\bigwedge_{\beta\in I}[\beta]\mathit{false}).$$

6. A proof system for SCCS with modal assertions.

In this section we take advantage of the observation we made in the last section that we can make perfectly good sense of terms which mix the syntax of assertions in with the syntax of the programming language. We use terms in **AP** to define a proof system.

First we define a proof system for the entailment relation $\geq$ between assertions. I am grateful to Colin Stirling for sending [St3] which showed me how to do the completeness proof for the assertion language, without SCCS operators.

6.1 Definition. In the following let $A, B, \cdots$ stand for assertions, and $X, Y, \cdots$ for a finite subset of assertions. Let $\vdash_A \subseteq \mathrm{Fin}\,(\mathbf{Assn}) \times \mathbf{Assn}$ be the least relation between finite sets of assertions and assertions closed under the following rules:

Structural rules:

refl. rule $\quad X \vdash_A A \quad \text{if } A \in X$

tran. rule $\quad \dfrac{X \vdash_A \bigwedge Y \quad Y \vdash_A A}{X \vdash_A A}$

Logical rules:

true r. rule $\quad \vdash_A true$

false l. rule $\quad false \vdash_A A$

$\bigwedge r.$ rule $\quad \{A_i \mid i \in I\} \vdash_A \bigwedge_{i \in I} A_i$

$\bigwedge l.$ rule $\quad \bigwedge_{i \in I} A_i \vdash_A A_i$

$\bigvee r.$ rule $\quad A_i \vdash_A \bigvee_{i \in I} A_i$

$\bigvee l.$ rule $\quad \dfrac{\{X, A_i \vdash_A B \mid i \in I\}}{X, \bigvee_{i \in I} A_i \vdash_A B}$

Modal rules:

$$\frac{A \vdash_A B}{\langle\alpha\rangle A \vdash_A \langle\alpha\rangle B}$$

$$\langle\alpha\rangle \bigvee_{i \in I} A_i \vdash_A \bigvee_{i \in I} \langle\alpha\rangle A_i$$

$$\frac{A \vdash_A B}{[\alpha] A \vdash_A [\alpha] B}$$

$$\bigwedge_{i \in I} [\alpha] A_i \vdash_A [\alpha] \bigwedge_{i \in I} A_i \quad \text{where } I \neq \emptyset$$

$$\langle\alpha\rangle false \vdash_A false$$

$$[\alpha] A \wedge \langle\alpha\rangle B \vdash_A \langle\alpha\rangle (A \wedge B)$$

Convergence rules $\quad [\alpha] true \vdash_A [\beta] true$

$$[\alpha] true \vdash_A \langle\beta\rangle true \vee [\beta] false.$$

The first convergence rule simply expresses the fact that a process converges iff it satisfies any assertion of the form $[\alpha]\mathit{true}$. The second convergence rule says that a convergent process either can perform an action α or refuses to perform an action α. The other rules are fairly intuitive. Note we must insist the indexing set I is nonempty in the rule expressing how conjunctions interact with $[\alpha]$ because $[\alpha]\mathit{true}$ is not always satisfied.

6.2 Theorem. *(Soundness and completeness)*

$$X \vdash_A A \Leftrightarrow \bigwedge X > A.$$

Proof. The proof of soundness is routine. The proof of completeness follows the following scheme: If $A \not\vdash_A B$ then a labelled transition system $H = (S, s_0, \{\xrightarrow{\alpha}\}_{\alpha \in Act}, \uparrow)$ is built from the proof system, such that $H \models A$ and $H \not\models B$. The construction of this Henkin model is closely based on that of Stirling [St3]. Then the technique of filtration, using the set of subformulae of A and B together with $[\alpha]\mathit{true}$ (for the convergence structure), reduces this model to a finite transition system which satisfies A but does not satisfy B. By lemma 2.3 this can be described by an SCCS process p. Hence $p \models A$ and $p \not\models B$ so $A \not\models B$. Thus $A \models B \Rightarrow A \vdash B$, so completeness follows directly. ∎

Now we define the proof rules to generate a relation $p \vdash_{AP} A$, meaning $p \models A$, *i.e.* $A \in \mathcal{A}[\![p]\!]\rho$ for p a term in **AP**—and so a mix of SCCS and **Assn**—and A an assertion, and ρ an arbitrary environment.

6.3 Definition. In the following let $A, B, C, \cdots$ stand for assertions, X for a finite subset of assertions and p, q terms in **AP**. Define $\vdash_{AP}$ to be the least relation between elements of **AP** and assertions **Assn** given by the rules:
Structural rules:

$$\frac{A \vdash_A B}{A \vdash_{AP} B}$$

$$\frac{p \vdash_{AP} A, C[A] \vdash_{AP} B}{C[p] \vdash_{AP} B} \quad \text{where } C[\ ] \text{ is any context in } \mathbf{AP}\ .$$

$$\frac{\{p \vdash_{AP} A \mid A \in X\}, X \vdash_A B}{p \vdash_{AP} B}$$

$$\frac{\{C[A] \vdash_{AP} B \mid A \in X\}}{C[\bigvee X] \vdash_{AP} B} \quad \text{where } C[\ ] \text{ is any proper context in } \mathbf{AP}\ .$$

(Note because $\bigvee \emptyset = \mathit{false}$, this includes the rule $\mathit{op}\ (\mathit{false}) \vdash_{AP} A$ where op is any derived unary operator.)

Correctness rules:

O–$[\alpha]$ rule $\quad \mathbf{O} \vdash_{AP} [\alpha]A$

α–$\langle\alpha\rangle$ rule $\quad \alpha A \vdash_{AP} \langle\alpha\rangle A$

α–$[\alpha]$ rule $\quad \alpha A \vdash_{AP} [\alpha]A$

α–$[\beta]$ rule $\quad \alpha A \vdash_{AP} [\beta]B \quad \text{if } \beta \neq \alpha$

$+$–$\langle\alpha\rangle$ rule $\quad \langle\alpha\rangle A + B \vdash_{AP} \langle\alpha\rangle A \quad \text{and} \quad B + \langle\alpha\rangle A \vdash_{AP} \langle\alpha\rangle A$

$+$–$[\alpha]$ rule $\quad [\alpha]A + [\alpha]A \vdash_{AP} [\alpha]A$

$\otimes$–$\langle\alpha\rangle$ rule $\quad \dfrac{B \otimes C \vdash_{AP} A}{\langle\beta\rangle B \otimes \langle\gamma\rangle C \vdash_{AP} \langle\beta \bullet \gamma\rangle A}$

$\otimes$–$[\alpha]$ rule $\quad \dfrac{\{B_\beta \otimes C_\gamma \vdash_{AP} A \mid \beta \bullet \gamma = \alpha\}}{(\bigwedge_\beta [\beta]B_\beta) \otimes (\bigwedge_\gamma [\gamma]C_\gamma) \vdash_{AP} [\alpha]A}$

$\lceil\Lambda$–$\langle\lambda\rangle$ rule $\quad \dfrac{A\lceil\Lambda \vdash_{AP} B}{(\langle\lambda\rangle A)\lceil\Lambda \vdash_{AP} \langle\lambda\rangle B} \quad \text{if } \lambda \in \Lambda$

$\lceil\Lambda$–$[\lambda]$ rule $\quad \dfrac{A\lceil\Lambda \vdash_{AP} B}{([\lambda]A)\lceil\Lambda \vdash_{AP} [\lambda]B} \quad \text{if } \lambda \in \Lambda$

$\lceil\Lambda$–$[\mu]$ rule $\quad ([\alpha]\mathit{true})\lceil\Lambda \vdash_{AP} [\mu]B \quad \text{if } \mu \notin \Lambda$

rec. rule $\quad \dfrac{p[\mathit{rec}x.p/x] \vdash_{AP} A}{\mathit{rec}x.p \vdash_{AP} A}$

$$\frac{p[\mathit{rec}^n x.p/x] \vdash_{AP} A}{\mathit{rec}^{n+1} x.p \vdash_{AP} A} \quad \text{for } n \in \omega$$

Remark. The requirement above that $C[\ ]$ is a proper context (*i.e.* a context with a real "hole") is made for one of the structural rules above so that $C[\mathit{false}] \models \mathit{false}$. This is clearly not the case for a non–proper context like $C[\ ] = \mathit{true}$. For the latter context, with X empty, the rule is clearly invalid.

The proof of completeness depends on the following lemma which shows how the decomposition rules of section 4 are captured in the proof system.

6.4 Lemma. *For assertions A, B, C*

(i) $\alpha\mathcal{D}_\alpha[\![A]\!] \vdash_{AP} A$,
(ii) $B + C \vdash_{AP} A$ *if* $(B, C) \in \mathcal{D}_+[\![A]\!]$,
(iii) $B \otimes C \vdash_{AP} A$ *if* $(B, C) \in \mathcal{D}_\otimes[\![A]\!]$,
(iv) $\mathcal{D}_{\lceil\Lambda}[\![A]\!]\lceil\Lambda \vdash_{AP} A$.

Proof. By structural induction on the assertion which is decomposed. ∎

Using this lemma we can show that the entailment relations $A \rhd B$, $A \otimes B \rhd C$ *etc.* on the domain **D** are all provable.

6.5 Lemma. *For assertions A, B, C*

(i) $\alpha A \rhd B \Rightarrow \alpha A \vdash_{AP} B$,
(ii) $B + C \rhd A \Rightarrow B + C \vdash_{AP} A$,
(iii) $B \otimes C \rhd A \Rightarrow B \otimes C \vdash_{AP} A$,
(iv) $A\lceil\Lambda \rhd B \Rightarrow A\lceil\Lambda \vdash_{AP} B$.

Proof. All the proofs follow a similar line. We show (iii). Suppose $E \otimes F \rhd A$. Let $p \in \Pi[\![E]\!]$ and $q \in \Pi[\![F]\!]$. Then there is $(B_{p,q}, C_{p,q}) \in \mathcal{D}_\otimes[\![A]\!]$ such that $p \models B_{p,q}$ and $q \models C_{p,q}$. Thus $p \models \bigwedge_{q\in\Pi[\![F]\!]} B_{p,q}$ and $q \models \bigwedge_{p\in\Pi[\![E]\!]} C_{p,q}$, where both are conjunctions of finite sets as $\mathcal{D}_\otimes[\![A]\!]$ is finite. Hence $E \models \bigvee_{p\in\Pi[\![E]\!]} \bigwedge_{q\in\Pi[\![F]\!]} B_{p,q}$, where the disjunction is of a finite set. Thus

$$E \vdash_A \bigvee_{p\in\Pi[\![E]\!]} \bigwedge_{q\in\Pi[\![F]\!]} B_{p,q}, \text{ and similarly } F \vdash_A \bigvee_{q\in\Pi[\![F]\!]} \bigwedge_{p\in\Pi[\![E]\!]} C_{p,q}.$$

By lemma 6.4 each $B_{p,q} \otimes C_{p,q} \vdash_{AP} A$, so by the structural rules we obtain $E \otimes F \vdash_{AP} A$. ∎

6.6 Theorem. *Let $p \in$ **AP** and $A \in$ **Assn**. Then $p \models A$ iff $p \vdash_{AP} A$.*

Proof. The proof of soundness is routine. To show completeness use structural induction on p and induction on iterates in the case of a recursive definition to show

$$\forall A.\ p \models A \Rightarrow p \vdash_{AP} A.$$

The cases of the structural induction use the above lemma. ∎

7. Related work.

Colin Stirling has produced a related proof system for SCCS but without restriction and in the case where recursive definitions are well-guarded. His proof system captures the concept of *relative satisfaction*, so he has proof rules which generate the relation $p \models^{St}{}_B A$ with this interpretation: if a process q satisfies B then $p \otimes q$ satisfies A; so relative satisfaction takes account of the environment. Clearly we can translate relative satisfaction into our notation by noting that $p \models^{St}{}_B A$ iff $p \otimes B \models A$.

There is some overlap in the work of this paper and that in [W] which presented the decomposition results of section 4 though in the more restricted case where the monoid of actions is a group. The proof system in [W] was unsatisfactory in several respects: it did not achieve strong completeness and relied on process variables in a somewhat *ad hoc* way.

The use of mixed assertions, mixing program syntax with assertions, makes the proof system and proofs about it much smoother. This realisation seems to have occurred independently to a number of people, though it was present some time ago in the paper [OH] by Olderog and Hoare. It is central in the work Graf and Sifakis [GS], which does not treat parallel composition however, and in the work of Brookes [B] which uses assertions built from synchronisation trees to survey a range of proof systems.

Acknowledgements

I have been strongly influenced by the work of Colin Stirling. I owe the proof system $\vdash_A$ and a large part of its proof of completeness to Colin's work.

References

[Ac] Aczel, P., An introduction to inductive definitions. In the handbook of Mathematical Logic, Ed. Barwise, J., North–Holland (1983).

[B] Brookes, S. D., On the axiomatic treatment of concurrency. In the proceedings of the joint US–UK seminar on the semantics of concurrency, July 1984, Carnegie–Mellon University, Pittsburgh, Springer–Verlag Lecture Notes in Comp. Sc. 197 (1984).

[BKP] Barringer H., Kuiper R. and Pnueli A., Now you may compose temporal logic specifications. In the proceedings of STOC 84 (1984).

[deNH] de Nicola, R. and Hennessy, M.C.B., Testing Equivalences for Processes, Lecture Notes in Comp. Sc. vol. 154 (1983) and in Theoretical Computer Science (1984).

[Gol] Golson, W. G., Denotational models based on synchronously communicating processes: refusal, acceptance, safety. In the proceedings of the joint US–UK seminar on the semantics of concurrency, July 1984, Carnegie–Mellon University, Pittsburgh, Springer–Verlag Lecture Notes in Comp. Sc. 197 (1984).

[GS] Graf, S., and Sifakis, J., A logic for the specification and proof of controllable processes of CCS. Advanced Seminar on logics and models for verification and specification of concurrent systems, La Colle–sur–Loup, France, to appear in Springer–Verlag Lecture Notes in Comp. Sc. (1984).

[H] Hoare, C.A.R., A model for communicating sequential processes. Monograph of the Programming Research Group, Oxford University (1981).

[HBR] Hoare, C.A.R., Brookes, S.D., and Roscoe, A.W., A Theory of Communicating Processes, Technical Report PRG-16, Programming Research Group, University of Oxford (1981); appears also in JACM (1984).

[HM] Hennessy, M.C.B. and Milner, R., On observing nondeterminism and concurrency, Springer LNCS Vol. 85. (1979).

[HP][HM] Hennessy, M.C.B. and Plotkin, G.D., A term model for CCS. Springer Lecture Notes in Comp. Sc., vol. 88 (1980).

[LW] Larsen, K. and Winskel, G., Using Information Systems to solve Recursive Domain Equations Effectively. Springer Lecture Notes in Comp. Sc., vol. 173 (1984). A full version appears as report No 51 of the Computer Laboratory, University of Cambridge.

[M1] Milner, R., A Calculus of Communicating Systems. Springer Lecture Notes in Comp. Sc. vol. 92 (1980).

[M2] Milner, R., A modal characterisation of observable machine–behaviour. Springer Lecture Notes in Comp. Sc. vol. 112 (1981).

[M3] Milner, R., Calculi for synchrony and asynchrony, Theoretical Computer Science, pp.267–310 (1983).

[OH] Olderog, E., and Hoare, C.A.R., Specification–oriented semantics for communicating processes. ICALP 83, Springer Lecture Notes in Comp. Sc. vol. 154 (1983).

[P] Plotkin, G. D., Some comments on Robin's "A modal characterisation of observable machine–behaviour". Handwritten notes, Comp. Sc. Dept., University of Edinburgh (1983).

[Pa] Park, D., Concurrency and automata on infinite sequences. Springer Lecture Notes in Comp. Sc. vol. 104 (1981).

[S] Scott, D. S., Domains for Denotational Semantics. ICALP 1982.

[S1] Scott, D. S., Lectures on a mathematical theory of computation. Oxford University Computing Laboratory Technical Monograph PRG–19 (1981).

[Sm] Smyth, M.B., Power domains and predicate transformers: a topological view. Proc. of ICALP 83, Springer Lecture Notes in Comp. Sc. vol. 154 (1983).

[St1] Stirling, C., A complete modal proof system for a subset of SCCS. Research report, Dept. of Comp. Sci., Edinburgh University (1984).

[St2] Stirling, C., A proof theoretic characterisation of observational equivalence. Research report, Dept. of Comp. Sci., Edinburgh University, CSR–132–83 (1983). A version

also appears in the proceedings of the Bangalore conference, India (1983) and is to appear in Theoretical Computer Science.

[St3] Stirling, C., A Complete Intuitionistic Hennessy–Milner Logic. Handwritten note, Comp. Sc. Dept, University of Edinburgh (Sep. 84).

[W] Winskel, G., On the composition and decomposition of assertions. In the proceedings of the joint US–UK seminar on the semantics of concurrency, July 1984, Carnegie–Mellon University, Pittsburgh, Springer–Verlag Lecture Notes in Comp. Sc. 197, and appears as a report of the Computer Laboratory, University of Cambridge (1984).