CONSISTENCY AND COMPLETENESS OF A PROOF SYSTEM FOR CSP

by

Neelam Soundararajan

Abstract:

In Soundararajan [6] a proof system for dealing with partial correctness of Communicating Sequential Processes (CSP) was presented. Here we prove the consistency and (relative) completeness of this system, with respect to a formal operational semantics for CSP.

RESEARCH REPORT
INSTITUTE OF INFORMATICS, UNIVERSITY OF OSLO

RESEARCH REPORT IN INFORMATICS

"Research report in Informatics" is a series of publications from University of Oslo, Norway, containing new results from computer science, numerical mathematics, software egineering and other parts of "informatics".

The series serves several objectives:

- to provide a faster way of publication than through journals,
- to allow usage of more space than in normal journal publications,
- to be used as study material in seminars, graduate education etc..

All texts will be in english.

The papers will be exchanged with similar productions from research groups elsewhere having corresponding interests.

Order for copies (duplication price) and other correspondence should be sent to:

Institutt for informatikk,
Universitetet i Oslo,
Postboks 1080, Blindern,
OSLO 3
Norway.

RESEARCH REPORT IN INFORMATICS

No 1	Arne Wang: Generalized Types in High-level Programming Languages.	Jan. 1975
No 2	Tom Lyche: Discrete Polynomial Spline Approximation Methods.	Jan. 1975
No 3	Ole-Johan Dahl and Arne Jonassen: Analysis of an Algorithm for Priority Queue Administration.	Jan. 1975
No 4	Olav Dahl: On the Problem of Solving Linear Algebraic Equations Associated with Matrices that are Polynomial Functions of a Square Matrix.	Jan. 1975
No 5	Tom Lyche: Discrete Cubic Spline Interpolation	Feb. 1975
No 6	Arne Wang: An Axiomatic Basis for Proving Total Correctness of <u>Goto</u>-programs.	April 1975
No 7	Tom Lyche: Computation of B-Spline Gram Matrices.	June 1975
No 8	Arne Jonassen: Additional Notes on the Normal P-tree Forest.	Sept. 1975
No 9	Arne Wang: Correctness of Transformations of Recursion to Iteration in Programs.	Oct. 1975
No 10	Arne Wang: The Semantics of Programs. Weakest Preconditions Revisited.	Nov. 1975
No 11	Tom Lyche: A Note on the Condition Numbers of the B-spline Bases	June 1976
No 12	Olav Dahl: Numerical Solution of Partial Differential Equations on Parallel Computers. A case study.	Nov. 1976
No 13	Ole-Johan Dahl: An Approach to Correctness Proof of Semicoroutines.	Jan. 1977
No 14	Arne Jonassen: Priority Queue Processes with Biased Insertion of Exponentially Distributed Input Keys.	May 1977
No 15	Arne Jonassen: A Formal Description of Data Table Processes.	May 1977

No 16	Ragnar Winther: Iterative Methods for a Class of Linear Equations with Application to Galerkin Approximations of a Parabolic Control Problem.	May 1977
No 17	Stein Gjessing: Compile Time Preparations for Run Time Scheduling in Monitors.	June 1977
No 18	T. Lyche og R. Winther: A Stable Recurrence Relation for Trigonometric B-splines.	Sept. 1977
No 19	Arne Jonassen: A Study on the Reducibility of the Polynomials.$x^m \pm x^n \pm 2p$.	Sept. 1977
No 20	Reiji Nakajima: Proving Abstract Spesifications to be Valid for their Program Implementations.	Nov. 1977
No 21	H. T. Faye-Schjøll, O. Hesjedal, D. F. Langmyhr, O. Owe: The Programming and Specification Language ABEL.	Nov. 1977
No 22	Reiji Nakajima: Sype-Class of Types or Partial Types - for Program Specification Structuring; and their Formalization a First Order Logic.	Nov. 1977
No 23	Arne Jonassen: Analysis of the "Dutch Flag Problem".	Nov. 1977
No 24	Arne Jonassen: Special Priority Queues with General Incident Sequences.	Nov. 1977
No. 25	Ellen Hisdal: Conditional Possibilities, Independence and Noninteraction.	Dec. 1977
No 26	Olaf Owe: Notes on Partial Correctness.	Dec. 1977
No 27	Arne Jonassen: Alternative Solution to the "Dutch National Flag" Problem.	Jan. 1978
No 28	Tom Lyche and L.L. Schumaker: An ALGOL Package for Computing Smooting and Interpolating Spline Functions.	Jan. 1978
No 29	Stein Gjessing: Monitors with Associated Processors	Feb. 1978
No 30	Sven Øivind Wille: Finite Element Formulations of Navier-Stokes Equations.	March 1978
No 31	Stein Gjessing: The Structure of the MUMI Multi Microcomputer System.	March 1978

No 32	Vesko Marinov: Maximal Clause Lenth Problem.	April 1978
No 33	Ole-Johan Dahl: Can Program Proving be Made Practical ?	May 1978
No 34	Arne Jonassen: The Limiting Behaviour of P-trees with Incident Sequence $OI^{n} D^{n}$	August 1978
No 35	Arne Jonassen: The "Chain Gang Schedule", Theoretical Solution.	August 1978
No 36	Arne Jonassen: The Polynomials $x^{m}x^{n}2p$, A new Approach to the Question of Irreducibility.	August 1978
No 37	Tom Lyche: A Newton form for trigonometric Hermite interpolation.	Sept. 1978
No 38	Dag F. Langmyhr and Olaf Owe: Revised Report on the Programming and Specification Language ABEL.	Oct. 1978
No 39	Ellen Hisdal: Acquisition of Information in a Learning Process.	March 1977
No 40	T. Dokken and T. Lyche: A Divided Difference Formula for the Error in Numerical Differentiation.	Dec. 1978
No 41	Arne Jonassen, Per-Arne Rindalsholt, Rolf Sødahl and Gunnar Valsø: A study on some Pseudo-Random Generators for a 36-bit Computer.	Dec. 1978
No 42	Ellen Hisdal: Conditional and Joint Possibilities of Higher Order.	July 1978
No 43	Ellen Hisdal: Concrete and Mathematical Sets, Measures of Second Order Possibilities.	Aug. 1978
No 44	Ellen Hisdal: Particularization - The Theory of Fuzzy Sets versus Classical Theories.	Nov. 1978
No 45	Stein Gjessing: Monitors with arrays of condition variables and proof rules handling local quantities.	June 1979
No 46	Stein Gjessing: Microcomputer Interrupt Servicing in an Environment of Processes and Monitors.	June 1979
No 47	Stein Gjessing: Microcomputer Software Design and Programming using the Concepts of Processes and Monitors - a Case Study.	June 1979

No	48	Ole-Johan Dahl: Time Sequences as a tool for describing program behaviour	Aug.	1979
No	49	Olav Dahl: On the choice of pivot elements in Gaussian elimination	March	1980
No	50	Olaf Owe: A specification Technique with Idealization	March	1980
No	51	Ellen Hisdal: The IF THEN ELSE Statement and Interval-Valued Fuzzy Sets of Higher Type	May	1980
No	52	Per M.A. Bothner: High-level data types in programming languages	Oct.	1980
No	53	Neelam Soundararajan: Semantics of communicating secquential processes	Febr.	1981
No	54	Robert W. Gunderson: Choosing the r-dimension for the FVC family of clustering algorithms	March	1981
No	55	Vesko Marinov: Resolution theorem prover in a program verification environment	May	1981
No	56	Neelam Soundararajan: Axiomatic semantics of communicating sequential processes	May	1981
No	57	Neelam Soundararajan: A note on the Egli-Milner order and Smyth's problem	May	1981
No	58	Neelam Soundararajan: Consistency and completeness of a proof system for CSP	June	1981
No	59	Neelam Soundararajan: Proofs of correctness of CSP programs	June	1981

1. Introduction

One of the most elegant languages for parallel programming is the language of Communicating Sequential Processes (CSP) introduced by Hoare[4]. In [6] we presented an axiomatic semantics for CSP; the sequences of communications exchanged by the processes play a rather prominent role in the semantics specified in [6]. From a formal point of view, the system of [6] has the important advantage that the proofs of the individual processes can be given independently of each other; this may be contrasted with the proof system of Apt et al [3] where the proofs of the individuel processes must be verified to "cooperate" with each other before one can arrive at any conclusion regarding the entire program. From a practical point of view, the fact that proofs of the individual processes of a program may be given independently of each other results in fairly simple proofs of program properties. This has been, we feel, amply demonstrated by the proofs described in [7] of several CSP programs.

Our goal in this paper is a rather theoretical one: we wish to demonstrate that the system of [6] is consistent and complete. The paper is organised as follows: in section 2 we present our axiomatic semantics for CSP. The semantics presented in section 2 is slightly different from the one defined in [6]; in particular, the rule of inference for I/O guarded commands presented in section 2 is different from the corresponding rule in [6]; the reason for the change is that, with the modified rule, the problem of "conspiracy among proofs" of [6] is rather easily handled.

In section 3, we introduce an operational semantics of CSP, consistent with the intuitive semantics given in Hoare[4]. We then prove that the semantics of section 2 is consistent with respect to this operational semantics. In section 4 we try to prove the completeness of the axiomatic system; in order to do this, we first introduce a simpler axiomatic system; we then show that any proof in the new system can be converted into a proof in the earlier system; finally we show that the new system is complete with respect to the operational semantics, thus proving the completeness of the original system.

In this paper, no informal justification is given for the various axioms and rules of inference of the axiomatic system; the interested reader may refer to [6] for the same.

2. Axiomatic semantics of CSP

Consider a CSP program $P::[P_1 // P_2 // \dots // P_n]$, $P_1,\dots,P_n$ being the communicating sequential processes. (We assume that $P_1,\dots,P_n$ do not themselves contain the "//" construct; thus parallelism exists only at the outermost level, the individual processes being strictly sequential). h_i will denote the communication sequence from P_i to the remaining (n-1)processes. Thus h_i will be a sequence of elements of the form (i,j,p), which denotes the number p being sent by P_i to P_j (by an output statement of the form $P_j!p$ in the processes P_i), and elements of the form (j,i,m), denoting the number m being received by P_i from P_j (in response to an input statement of the form $P_j?x$ in P_i); h_i may also include another type of element, of the form (i,j,t) whose purpose will become clear later.

In proving some property of P_i, the system allows us to make assumptions regarding the sequence from P_j to P_i. We shall use the symbol h'_{ji} to denote the sequence from P_j to P_i; as P_i is executed, it will "consume" portions of h'_{ji} (one element per input or output statement of the form $P_j?x$ or $P_j!n$); h_{ji} will denote the portion of the sequence from P_j to P_i that P_i has "already" consumed, h'_{ji} denoting the portion that P_i is "yet" to consume. Thus, at the "start" of P_i, we shall have a predicate of the form $I_i \wedge [h_i = \varepsilon] \wedge [\bigwedge_{j\neq i} h_{ji} = \varepsilon] \wedge C_{ji}(h'_{ji})$, where I_i is a given predicate regarding s_i, the local state of P_i; $[h_i = \varepsilon]$ and $[\bigwedge_{j\neq i} h_{ji} = \varepsilon]$ indicate the facts that, at the start, P_i has neither produced, nor consumed anything. We shall denote the set of objects $s_i, h_i, h_{ij} (j\neq i), h'_{ji} (j\neq i)$, collectively as the state σ_i of P_i.

We are now ready to state the axioms and rules of inference for the various constructs that may appear in the individual processes. We shall state these as applicable to P_i. (Note: We have assumed that the processes are numbered 1 through n; if, instead, processes can be given arbitrary names, we would have to make the obvious changes in the system; thus the communication sequences will be made up of elements of the form (X,Y,n) to denote the number n being sent from the process X to the process Y. Note also that we do not consider declarations in our system. Declarations may be treated in the standard fashion).

A1. Input

$$\{[D(h'_{ji})=(j,i)] \wedge p'\}P_j?\,x\,\{p\}$$

where $p' \equiv p^{x,\;h_i,\;h_{ji},\;h'_{ji}}_{V(h'_{ji}),\;h_i \vdash (j,i,V(h'_{ji})),\;h_{ji} \vdash \mathrm{first}(h'_{ji}),\;\mathrm{rest}(h'_{ji})}$;

first(h) is the leftmost element of the sequence h; rest(h) is the sequence got by deleting the leftmost element of h.
"$\vdash$" denotes concatenation of an element to the right end of a sequence. Similarly, "$\dashv$" will denote concatenation of an element to the left end of a sequence, and "$\vdash\!\dashv$" will denote concatenation of two sequences.

$$D(h) = \begin{cases} t' & \text{if } h=\varepsilon \\ t & \text{if first}(h)=(k,l,t) \text{ for some } k,l. \\ (k,l) & \text{if first}(h)=(k,l,m) \text{ for some integer } m. \end{cases}$$

$$V(h) = \begin{cases} 0 & \text{if } h=\varepsilon, \text{ or if } D(h)=t, \\ m & \text{if first}(h)=(k,l,m). \end{cases}$$

(Note: We assume that only integers are being exchanged between the processes).

The above axiom alone is not sufficient for input; we also need the following rule of inference:

R1. Rule of inference for input

$$\frac{p \Rightarrow \exists \bar{h}'_{ji}.[p' \wedge [D(\bar{h}'_{ji})=(j,i)] \qquad \{p \wedge [D(h'_{ji})=(j,i)]\}P_j?\,x\,\{q\}}{\{p\}\;P_j?\,x\,\{q\}}$$

where $p' \equiv p^{h'_{ji}}_{\bar{h}'_{ji}}$

A2. Output

$$\{[D(h'_{ji})=(i,j)] \wedge [V(h'_{ji})=y] \wedge p'\}P_j!\,y\,\{p\}$$

where

$$p' \equiv p^{h_i,\;h_{ji},\;h'_{ji}}_{h_i \vdash (i,j,y),\;h_{ji} \vdash \mathrm{first}(h'_{ji}),\;\mathrm{rest}(h'_{ji})}.$$

R2. Rule of inference for output

$$\frac{\begin{array}{l} p \Rightarrow \exists \bar{h}'_{ji} \cdot [p' \wedge [D(\bar{h}'_{ji})=(i,j)] \wedge [V(\bar{h}'_{ji}) = y]] \\ \{p \wedge [D(h'_{ji}) = (i,j) \wedge [V(h'_{ji}) = y]\} \; P_j ! \; y \; \{q\} \end{array}}{\{p\} \; P_j ! \; y \; \{q\}}$$

where $p' \equiv p^{h'_{ji}}_{\bar{h}'_{ji}}$

A3. Assignment

$$\{p^x_e\} \; x:=e \; \{p\}$$

A4. Skip

$$\{p\} \; \underline{\text{skip}} \; \{p\}$$

Note: In the rules R3,R4,R5 and R6 below, q' will denote the following predicate:

$$q' \equiv \exists \bar{\sigma}_i \cdot [q^{\sigma_i}_{\bar{\sigma}_i} \wedge [h_i \sqsubseteq \bar{h}_i]]$$

where $h \sqsubseteq \bar{h}$ is true if h is an initial subsequence of $\bar{h}$.

R3. Boolean guarded selection

$$\frac{\begin{array}{l} p \Rightarrow q' \\ \{p \wedge b_r\} S_r \{q\}, \; r = 1,..,m \end{array}}{\{p\} [\square (r=1,..,m) b_r \rightarrow S_r] \{q\}}$$

R4. Boolean repetition

$$\frac{\begin{array}{l} p \Rightarrow q' \\ [p \wedge \neg b] \Rightarrow q \\ \{p \wedge b_r\} S_r \{p\}, \; r = 1,..,m \end{array}}{\{p\} * [\square (r = 1,..,m) b_r \rightarrow S_r] \{q\}}$$

where $b = \bigvee_{r=1}^{m} b_r$.

R5. I/O guarded selection

$$\frac{\begin{array}{l} p \Rightarrow q' \\ \{p \wedge b_r\} \alpha_r \{q_r\}, \{q_r\} S_r \{q\}, \; r = 1,..,m \end{array}}{\{p\} [\square (r= \;\; ,..,m) b_r ; \alpha_r \rightarrow S_r] \{q\}}$$

where α_r are I/O statements.

R6. I/O quarded repetition

$$\frac{\begin{array}{l} p \Rightarrow q' \\ [p \wedge \neg b] \Rightarrow q \\ [p \wedge b \wedge [\forall_{\ell \leq m} b_\ell \Rightarrow (h'_{c(\alpha_\ell)i} = \varepsilon)] \Rightarrow [q^{h_i}_{h_i \vdash (i,\{j \mid \exists r \leq m.[b_r \wedge c(\alpha_r) = j]\},t)}] \\ \{p \wedge b_r \wedge (h'_{c(\alpha_r)i} \neq \varepsilon)\}\, \alpha_r \{q_r\}, \{q_r\} S_r \{p\}, r = 1,..,m \end{array}}{\{p\}*[\square(r = 1,..,m) b_r; \alpha_r \to S_r]\{q\}}$$

where

$c(\alpha_r) = j$ if α_r is $P_j!y$ or $P_j?x$;

$b \equiv \bigvee_{r=1}^{m} b_r$.

The second line of R6 corresponds to the situation where the loop terminates because the boolean portions of the guards are all false. The third line corresponds to the case when the loop has to terminate despite some of the booleans being true, on account of the fact that the sequences from the other processes are all ε. In this case, we extend the sequence h_i by the term (i,T,t) where T is the set of indices of those processes with which P_i was willing to continue communicating; the "t" is a special symbol which indicates that P_i's loop terminated because the processes mentioned in T had terminated. This element needs to be added to h_i in order to avoid the problem of conspiring proofs of [6]. The first lines of R3 through R6 are needed to ensure consistency of the system.

R7. Sequential composition

$$\frac{\{p\}S_1\{q_1\}, \{q_1\}S_2\{q\}}{\{p\}S_1;S_2\{q\}}$$

R8. Consequence

$$\frac{p \Rightarrow p_1, \{p_1\}S\{q_1\}, \ q_1 \Rightarrow q}{\{p\}\ S\ \{q\}}$$

R9. Conjunction

$$\frac{\{p\}S\{q_1\}, \{p\}S\{q_2\}}{\{p\}S\{q_1 \wedge q_2\}}$$

R10. Disjunction

$$\frac{\{p_1\}S\{q\},\{p_2\}S\{q\}}{\{p_1 \vee p_2\}S\{q\}}$$

R11. Parallel composition

$$\frac{\begin{array}{c} \exists h'_{ji}.C_{ji} \quad , \quad j,i=1,..,n, \quad j \neq i \\ \left[[F_i \wedge [\bar{h}_{ij} = h_i/_{\{(i,j),(j,i)\}}] \Rightarrow C_{ij}\left[\frac{h'_{ij}}{\bar{h}_{ij}}\right] \forall\ 1 \leq j \leq n,\ j \neq i\right],\ i=1,..,n \\ \{I_i \wedge (h_i = \varepsilon) \wedge (\bigwedge_{j \neq i} h_{ji} = \varepsilon) \wedge (\bigwedge_{j \neq i} C_{ji})\}P_i\{F_i\},\ i=1,..,n \end{array}}{\{\bigwedge_{i=1}^{m} I_i\}[P_1 /\!/ \ldots /\!/ P_n]\{\bigwedge_{i=1}^{n} F_i \wedge [\text{Reduce}(h_1,..,h_n) = (\varepsilon,..,\varepsilon)]\}}$$

The first line is needed to ensure consistency of the system. The second line says that the assumption C_{ij} made in the proof of P_j, regarding the sequence from P_i to P_j must be justified by the post-condition of P_i; $\bar{h}_{ij} = h_i/_{\{(i,j),(j,i)\}}$ indicates that $\bar{h}_{ij}$ is obtained from h_i by retaining elements of the form (i,j,m) or (j,i,m), changing elements of the form (i,T,t) into (i,j,t) if $j \in T$, and omitting all other elements.

The predicate $[\text{Reduce}(h_1,..,h_n)=(\varepsilon,..,\varepsilon)]$ expresses a consistency condition on the communication sequences of the various processes. Reduce is a function of n sequences whose value is also n sequences. $\text{Reduce}(h_1,..,h_n)$ is defined as follows: if two of the sequences h_i, h_j are such that $\text{first}(h_i) = \text{first}(h_j)$ then,

$$\text{Reduce}(h_1,..,h_n) = \text{Reduce}(h'_1,..,h'_n),$$

where $h'_i = \text{rest}(h_i)$, $h'_j = \text{rest}(h_j)$, $h'_k = h_k$ for all $k \neq i,j$.

If there is no such pair of sequences, and if $\text{first}(h_i)$ is (i,T,t) and for all $j \in T$, $h_j = \varepsilon$ then,

$$\text{Reduce}(h_1,..,h_n) = \text{Reduce}(h'_1,..,h'_n),$$

where $h'_i = \text{rest}(h_i)$ and $h'_k = h_k$ for all $k \neq i$.

If there are no sequences satisfying either of the above conditions then

$$\text{Reduce}(h_1,..,h_n) = (h_1,..,h_n).$$

(Note: If there is more than one pair of sequences satifying the first condition or more than one sequence satisfying the second, the reduction may be performed in any order, since the final result will be the same).

We could have added another clause to the final post-condition: $\bigwedge_{i=1}^{n}\left[\bigwedge_{j\neq i} h_{ij} = h_i/_{\{(i,j),(j,i)\}}\right]$, to express the fact that the sequence from P_i that P_j has consumed is, in fact, the sequence sent by P_i to P_j; we did not do this, allowing this fact to follow from the notation; thus in any predicate in which both h_i and h_{ij} occur, we may assume tha $h_{ij} = h_i/_{\{(i,j),(j,i)\}}$.

Before concluding this section, we introduce som concepts which will be useful in the next section. These concepts have been modelled on the analogous concepts for parallel programs introduced by Owicki[5]

Suppose P_i is a communicating sequential process. The primary components of P_i are

1) none if p_i is a skip, an assignment, or an I/O statement.

2) $S_1,\ldots,S_r$ if P_i is $S_1;S_2,\ldots,S_r$.

3) $S_1,\ldots,S_m$ if P_i is $[\square(\ell=1,\ldots,m)b_\ell\rightarrow S_\ell]$, or $*[\square(\ell=1,\ldots,m)b_\ell\rightarrow S_\ell]$.

4) $\alpha_1,S_1,\ldots,\alpha_m,S_m$ if P_i is $[\square(\ell=1,\ldots,m)b_i;\alpha_\ell\rightarrow S_\ell]$ or $*[\square(\ell=1,\ldots,m)b_\ell;\alpha_\ell\rightarrow S_\ell]$.

The proper components of P_i are the primary components of P_i, and their proper components. The components of P_i are P_i itself and its proper components.

Suppose pre and post are two functions which map components of P_i to assertions. Then pre and post are assertion functions for $\{p\}P_i\{q\}$ if the following conditions are satified:

1) $p \Rightarrow \mathrm{pre}(P_i)$ and $\mathrm{post}(P_i) \Rightarrow q$.

Conditions (2) through (10) must be satisfied for every component S of P_i.

2) If S is skip then $\mathrm{pre}(S) \Rightarrow \mathrm{post}(S)$.

3) If S is x:=e then $\mathrm{pre}(S) \Rightarrow \mathrm{post}(S)^{x}_{e}$.

4) If S is $P_j!y$ then $\mathrm{pre}(S) \Rightarrow \exists \bar{h}'_{ji} \cdot [\mathrm{pre}(S)^{h'_{ji}}_{\bar{h}'_{ji}} \wedge [D(\bar{h}'_{ji}) = (i,j)] \wedge [V(\bar{h}'_{ji}) = y]]$, and

$[\mathrm{pre}(S) \wedge [D(h'_{ji}) = (i,j)] \wedge [V(h'_{ji}) = y]] \Rightarrow$

$$\mathrm{post}(S)^{h_i,\; h_i,\; h'_{ji}}_{h_i \vdash (i,j,y),\; h_{ji} \vdash \mathrm{first}(h'_{ji}),\; \mathrm{rest}(h'_{ji})}$$

5) If S is $P_j?x$: similar to (4).

6) If S is $[\Box(\ell=1,..,m)b_\ell \to S_\ell]$ then $\text{pre}(S) \Rightarrow \exists\bar{\sigma}_i\cdot[\text{post}(S)^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq \bar{h}_i)]$, and

$\text{pre}(S) \wedge b_r \Rightarrow \text{pre}(S_r)$, $\text{post}(S_r) \Rightarrow \text{post}(S)$, $r=1,\dots,m$.

7) If S is $*[\Box(\ell=1,..,m)b_\ell \to S_\ell]$ then $\text{pre}(S) \Rightarrow \exists\bar{\sigma}_i\cdot[\text{post}(S)^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq \bar{h}_i)]$, and

$\text{pre}(S) \wedge b_r \Rightarrow \text{pre}(S_r)$, $\text{post}(S_r) \Rightarrow \text{pre}(S)$, $r=1,..,m$, and

$[\text{pre}(S) \wedge (\neg(\bigvee_{\ell=1}^{m} b_\ell))] \Rightarrow \text{post}(S)$.

8) If S is $[\Box(\ell=1,..,m)b_\ell;\alpha_\ell \to S_\ell]$ then $\text{pre}(S) \Rightarrow \exists\bar{\sigma}_i\cdot[\text{post}(S)^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq \bar{h}_i)]$, and

$\text{pre}(S) \wedge b_r \Rightarrow \text{pre}(\alpha_r)$, $\text{post}(\alpha_r) \Rightarrow \text{pre}(S_r)$, $\text{post}(S_r) \Rightarrow \text{post}(S)$, $r=1,..,m$.

9) If S is $*[\Box(\ell=1,..,m)b_\ell;\alpha_\ell \to S_\ell]$ then $\text{pre}(S) \Rightarrow \exists\bar{\sigma}_i\cdot[\text{post}(S)^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq h_i)]$, and

$[\text{pre}(S) \wedge (\neg(\bigvee_{\ell=1}^{m} b_\ell))] \Rightarrow \text{post}(S)$,

$[\text{pre}(S) \wedge b_r \wedge (h'_{c(\alpha_r)i} \neq \varepsilon)] \Rightarrow \text{pre}(\alpha_r)$, $\text{post}(\alpha_r) \Rightarrow \text{pre}(S_r)$, $\text{post}(S_r) \Rightarrow \text{pre}(S)$, $r=1,..,m$

and, $[\text{pre}(S) \wedge (\bigvee_{\ell=1}^{m} b_\ell) \wedge [\forall_{\ell\leq m}\, b_\ell \Rightarrow (h'_{c(\alpha_\ell)i} = \varepsilon)] \Rightarrow$

$$[\text{post}(S)^{h_i}_{h_i \nVdash (i,\{j \mid \exists r\leq m.\, b_r \wedge c(\alpha_r)=j\},t)}].$$

10) If S is $S_1;\dots;S_r$, then

$\text{pre}(S) \Rightarrow \text{pre}(S_1)$, $\text{post}(S_\ell) \Rightarrow \text{pre}(S_{\ell+1})$ for $\ell=1,..,r-1$, and $\text{post}(S_r) \Rightarrow \text{post}(S)$.

Suppose now that $\{p\}P_i\{q\}$ is provable using the axioms and rules of inference specified earlier in this section. It should then be clear (and can be formally shown) that there exist assertion functions pre and post for $\{p\}P_i\{q\}$; and, conversely, if pre and post are assertion functions for $\{p\}P_i\{q\}$, then $\{p\}P_i\{q\}$ is provable.

Next suppose $P::[P_1// \dots //P_n]$ is a CSP program, and that using the axioms and rules of inforence we are able to prove $\{I\}P_1//\dots//P_n\{F\}$. Then (R11 being the rule for parallel composition), we may assume that I can be written in the form $\bigwedge_{i=1}^{n} I_i$, I_i being predicates regarding the local state s_i of P_i, F can be written as $[\bigwedge_{i=1}^{n} F_i] \wedge [\text{Reduce}(h_1,\dots,h_n) = (\varepsilon,..,\varepsilon)]$, F_i being a predicate on $s_i, h_i, h_{ji} (j \neq i)$ and h'_{ji}, and that there exist predicates $C_{ij} (i \neq j)$ satisfying the requirements stated in the first two lines of R11, and that $\{I_i \wedge (h_i=\varepsilon) \wedge (\bigwedge_{j\neq i} C_{ji})\}P_i\{F_i\}, i=1,\dots,n$ are provable using our system. Thus there exist (n pairs of) assertion functions $\text{pre}_i, \text{post}_i$ $(i=1,..,n)$, $\text{pre}_i, \text{post}_i$ being assertion functions for $\{I_i \wedge (h_i=\varepsilon) \wedge (\bigwedge_{j\neq i} h_{ji} = \varepsilon) \wedge (\bigwedge_{j=i} C_{ji})\}P_i\{F_i\}$. In the next section we shall assume that any proof of the program P is specified by giving the assertion functions $pre_i, post_i, i = 1,..,n$.

We conclude this section with one further remark: suppose pre_i, $post_i$ are assertion functions for $\{p\}P_i\{q\}$; suppose also that S is a component of P_i. Then the following facts are easily proved:

$$pre(S) \Rightarrow \exists \bar{\sigma}_i \cdot [q^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq \bar{h}_i), \text{ and } post(S) \Rightarrow \exists \bar{\sigma}_i \cdot [q^{\sigma_i}_{\bar{\sigma}_i} \wedge (h_i \sqsubseteq h_i)] .$$

These two results will be of importance in proving the consistency of the system.

3. Consistency of the system

We must first introduce an operational model for the execution of CSP programs. The model we use is based on the model of Apt [1]. The model is defines by specifying the relation "$\rightarrow$" between pairs consisting of a program and a state; essentially "$\rightarrow$" denotes the execution of one step of the program. (If this step is a communication between P_i and P_j, this will be the execution of an input statement $P_j?x$ in P_i and an output statement $P_i!y$ in P_j).

For convenience, we shall allow the empty process E. (Thus after executing the final step of a process, we shall denote the process by E). We shall use the symbol τ (as well as $\tau', \tau'', \bar{\tau}$ etc.) to denote the current state; thus τ will be an n-tuple $(\tau_1,..,\tau_n)$ where τ_i is the state of the ith process. τ_i will have two components: s_i, the local state, specifying the values of the variables of the ith process, and h_i, the communication sequence of the ith process. Thus $(P,\tau) \rightarrow (P',\tau')$ specifies that executing P one step in the state τ can lead to the state τ' with P' being the program yet to be executed. (of course, "$\rightarrow$" does not specify a function; thus there may be more than one (P',τ'), which may arise by execution of one step of P). With these preliminaries taken care of, we can now define "$\rightarrow$".

Definition: $(S_1 // ... // S_n, \tau) \rightarrow (S'_1 // ... // S'_n, \tau')$ if any one of the following seventeen clauses is satisfied:
(In the various clauses, we ought to have included quantifiers of the kind $\exists i. 1 \leq i \leq n$, but these have been omitted; this omission should cause no confusion. Note also that more than one clause cannot be simultaneously satisfied).

1) $S_i \equiv$ skip, and $S'_i \equiv E$, $S'_j \equiv S_j$ for all $j \neq i$; and $\tau' = \tau$.

2) $S_i \equiv x{:=}e$, and $S'_i \equiv E$, $S'_j \equiv S_j$ for all $j \neq i; \tau_i = \tau_i[x \leftarrow e], \tau'_j = \tau_j$ for all $j \neq i$;

where $\tau'_i = \tau_i[x \leftarrow e]$ indicates that τ'_i is the same state as τ_i, except that the value of the variable x has been changed to be equal to the value of the expression e, evaluated in the state τ_i.

In the following, we shall omit clauses of the kind $S'_j \equiv S_j$ for all $j \neq i$; thus $S'_k \equiv S_k$ unless otherwise specified; similarly we shall omit $\tau'_j = \tau_j$ for all $j \neq i$, the understanding being that those components of τ' that are not mentioned are identical to the corresponding components of τ.

3) $S_i \equiv [\Box(\ell=1,\dots,m) b_\ell \rightarrow T^\ell]$, and

$\models b_r(\tau_i)$, $S'_i \equiv T^r$, and $\tau' = \tau$.

($\models b_r(\tau_i)$ inducates that the boolean b_r has the value true in state τ_i).

4) $S_i \equiv * [\Box(\ell=1,\dots,m) b_\ell \rightarrow T^\ell]$, and

$\models (\bigwedge_{\ell=1}^{m} \neg b_\ell)(\tau_i)$, $S'_i \equiv E$, and $\tau' = \tau$.

(all the guards are false, and hence the loop terminates).

5) $S_i \equiv *[\Box(\ell=1,\dots,m) b_\ell \rightarrow T^\ell]$, and

$\models b_r(\tau_i)$, $S'_i \equiv T^r ; S_i$, and $\tau' = \tau$.

6) $S_i \equiv P_j?x$, $S_j \equiv P_i!y$, and

$S'_i \equiv E, S'_j \equiv E, \tau'_j = \tau_j[h_j \leftarrow h_j \vdash (j,i,\tau_j(y))]$, $\tau'_i = \tau_i[x \leftarrow \tau_j(y), h_i \leftarrow h_i \vdash (j,i,\tau_j(y))]$

(The effect of performing the communication is to assign to x, the current value of y, and extend the sequences h_i, h_j by the term (j,i,m), m being the value of y in the state τ_j).

7) $S_i \equiv [\Box(\ell=1,\dots,m) b_i ; \alpha_\ell \rightarrow T^\ell]$, and

$\models b_r(\tau_i)$, $\alpha_r = P_j?x$, $S_j \equiv P_i!y$, and

$S'_i \equiv \alpha_r ; T^r$, and $\tau' = \tau$.

8) $S_i \equiv [\Box(\ell=1,\dots,m) b_\ell ; \alpha_\ell \rightarrow T^\ell]$, and

$\models b_r(\tau_i)$, $\alpha_r = P_j!x$, $S_j \quad P_i?y$, and

$S'_i \equiv \alpha_r ; T^r$, and $\tau' = \tau$.

9) $S_i \equiv [\Box(\ell=1,\dots,m(b_\ell ; \alpha_\ell \rightarrow T^\ell]$, $S_j \equiv [\Box(\ell'= ,\dots,m') b'_{\ell'} ; \alpha'_{\ell'} \rightarrow T'^{\ell'}]$, and

$\models b_r(\tau_i), \alpha_r = P_j?x$, $\models b'_{r'}(\tau_j)$, $\alpha'_{r'} = P_i!y$, and

$S'_i \equiv \alpha_r ; T^r$, $S'_j \equiv \alpha'_{r'} ; T'^{r'}$, and $\tau' = \tau$.

10) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \to T^\ell]$, and

$\models b_r(\tau_i)$, $\alpha_r = P_j?\,x$, $S_j \equiv P_i!\,y$, and

$S'_i \equiv \alpha_r;\, T^r;\, S_i$, and $\tau' = \tau$.

11) $S_i \equiv *[\square(\ell=1,..,m)\, b_\ell;\alpha_\ell \to T^\ell]$, and

$\models b_r(\tau_i)$, $\alpha_r = P_j!\,x$, $S_j \equiv P_i?\,y$, and

$S'_i \equiv \alpha_r;\, T^r; S_i$, and $\tau' = \tau$.

12) $S_i \equiv *[\square(\ell=1,..,m)\, b_\ell;\alpha_\ell \to T^\ell]$, $S_j \equiv [\square(\ell'=1,..,m') b'_{\ell'};\, \alpha'_{\ell'} \to T'^{\ell'}]$, and

$\models b_r(\tau_i)$, $\models b'_{r'}(\tau_j)$, $\alpha_r = P_j!\,x$, $\alpha'_{r'} = P_i?\,y$, and

$S'_i \equiv \alpha_r; T^r; S_i$, and $S'_j \equiv \alpha'_{r'};\, T'^{r'}$, and $\tau' = \tau$.

13) Same as (12) except that $\alpha_r = P_j?\,x$, and $\alpha'_{r'} = P_i!\,y$.

14) $S_i \equiv *[\square(\ell=1,..,m)\, b_\ell;\alpha_\ell \to T^\ell]$, $S_j \equiv *[\square(\ell'=1,..,m) b'_{\ell'};\alpha'_{\ell'} \to T'^{\ell'}]$, and

$\models b_r(\tau_i)$, $\models b'_r(\tau_j)$, $\alpha_r = P_j!\,x$, $\alpha'_{r'} = P_i?\,y$, and

$S'_i \equiv \alpha_r;\, S_i$, and $S'_j \equiv \alpha'_{r'};\, T'^{r'}; S_j$, and $\tau' = \tau$.

15) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \to T^\ell]$, and

$\models (\bigwedge_{\ell=1}^{m} \neg b_\ell)(\tau_i)$, and $S'_i \equiv E, \tau' = \tau$.

16) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \to T^\ell]$, and

$\models (\bigvee_{\ell=1}^{m} b_\ell)(\tau^i)$, and for all r such that $\models b_r(\tau_i)$, $S_{c(\alpha_r)} \equiv E$, and

$S'_i \equiv E$, and $\tau'_i = \tau_i[h_i \leftarrow h_i \vdash (i,T,t)]$, where $T = \{k \mid \exists\, r.[1 \le r \le m \;\wedge$
$\models b_r(\tau_i) \wedge k = c(\alpha_r)]\}$,

where $c(\alpha_r) = j$ if $\alpha_r = P_j?\,x$ or $P_j!\,x$.

This case corresponds to the termination of an I/O guarded loop in S_i on account of the fact that all processes with which S_i is willing to communicate, have terminated.

17) $S_i \equiv \bar{S}_i\,; S$, and $\bar{S}_i \ne E$, and $(S_1 /\!/ .. /\!/ \bar{S}_i /\!/ .. /\!/ S_n, \tau) \to (\bar{S}'_1 /\!/ .. /\!/ \bar{S}'_i /\!/ .. /\!/ \bar{S}'_n, \tau')$, and

$[\bar{S}'_i \ne E, S'_i \equiv \bar{S}'_i; S]$ or $[\bar{S}'_i \equiv E,\ S'_i \equiv S\,]$, and $S'_j \equiv \bar{S}'_j$.

In other words, this clause says that if

$(S_1 /\!/ .. /\!/ S_i /\!/ .. /\!/ S_n, \tau) \to (S'_1 /\!/ .. S'_n, \tau')$, $S_i \ne E$, then

$(S_1 /\!/ .. /\!/ S_i; S /\!/ .. /\!/ S_n, \tau) \to (S'_1 /\!/ .. /\!/ S'_i\,; S /\!/ .. /\!/ S'_n, \tau')$.

The restriction $\bar{S}_i \neq E$ is needed to ensure the validity of clause (16). Moreover, we may also assume, without loss of generality, that $\bar{S}_i$ is of the form skip, or $x:=e$, or $P_j?x$, or $P_j!y$, or $[\Box b_\ell \to T^\ell]$, or $*[\Box b_\ell \to T^\ell]$, or $[\Box b_\ell; \alpha_\ell \to T^\ell]$, or $*[\Box b_\ell; \alpha_\ell \to T^\ell]$. We are using $E;S \equiv S$ to handle the situation when $\bar{S}_i' \equiv E$ in the above clause. Clause 17 takes care of sequential composition.

Suppose now that we have a CSP program $P:: P //..// P_n$. Suppose $s^I = (s_1^I,..,s_n^I)$ is some given initial state of P. Thus s_i^I is the initial local state of P_i. We shall, in fact, take $\tau^I=(\tau_1^I,..,\tau_n^I)$, where $\tau_i^I=(s_i^I,\varepsilon)$ to be the initial state of P. Recall that the second component of τ_i is the communication sequence h_i of P_i; at the state of P, the communication sequences of all the processes are ε, justifying our definition of τ^I.

Suppose there is some execution of P, starting in the above initial state τ^I and terminating in some final state τ^F, all processes having terminated. Then it should be clear from the definition of "$\rightarrow$" that there exists a sequence of pairs $(S^1,\tau^1),...,(S^u,\tau^u)$ such that $S^1 \equiv P_1//...//P_n$, $\tau^1 = \tau^I$, $S^u \equiv E//...//E$, $\tau^u=\tau^F$, and $(S^w,\tau^w) \rightarrow (S^{w+1},\tau^{w+1})$ for all w, $1 \leq w < u$. The sequence $(S^1,\tau^1),...,(S^u,\tau^u)$ will be called the computation sequence.

Next suppose, we have proved, using the system of the last section the following: $\{\bigwedge_{i=1}^{n} I_i\}P_1//..//P_n\{[\bigwedge_{i=1}^{n} F_i]\wedge[\mathrm{Reduce}(h_1,..,h_n)=(\varepsilon,..,\varepsilon)]\}$.

Let $pre_i, post_i$ $(i=1,..,n)$ be assertion functions for

$$\{I_i \wedge (h_i=\varepsilon)\wedge(\bigwedge_{j\neq i} h_{ji}=\varepsilon)\wedge(\bigwedge_{j\neq i} C_{ji})\}\ P_i\{F_i\},$$

C_{ji} $(j,i=1,..,n, j\neq i)$ satisfying the conditions of the first two lines of R11. We shall define certain assertion functions for $S^v (v=1,..,u)$. Recall that S^v is of the form $S_1^v//...//S_n^v$. Thus the assertion functions associated with S^v will be a set of n pairs of functions pre_i^v, $post_i^v$, $i=1,..,n$. S^1 is of the form $S_1^1//...//S_n^1$, $S_i^1 \equiv P_i$. Hence we define, $pre_i^1 = pre_i$ and $post_i^1 = post_i$. Thus pre_i^1, $post_i^1$ are assertio functions for $\{I_i \wedge (h_i=\varepsilon)\wedge(\bigwedge_{j\neq i} h_{ji}=\varepsilon)\wedge(\bigwedge_{j\neq i} C_{ji})\}\ S_i^1\ \{F_i\}$.

Next consider the assertion functions for S^v, $v>1$. pre_i^v, $post_i^v$ will be defined in terms of pre_i^{v-1}, $post_i^{v-1}$. Before defining pre_i^v, $post_i^v$, we need to take care of a minor point: in our original definition of assertion functions we did not allow for the possibility that the process may be of the form E. We allow for this by adding the following clause to the definition of assertion functions: if P_i is E then $pre(P_i) \Rightarrow post(P_i)$; and if P_i is E, the only component of P_i is E.

Now we are ready to define $pre_i^v, post_i^v$.

1) For every component T of S_i^v, which is also a component of S_i^{v-1} ,

$pre_i^v(T) = pre_i^{v-1}(T)$, $post_i^v(T) = post_i^{v-1}(T)$.

2) If $S_i^{v-1} \not\equiv E$ and $S_i^v \equiv E$, then $pre_i^v(S_i^v), post_i^v(S_i) = post_i^{v-1}(S_i^{v-1})$.

3) If $S_i^{v-1} \equiv *[\square(\ell=1,..,m) b_\ell \to T_\ell]$, and $S_i^v \equiv T_r; S_i^{v-1}$, then

$pre_i^v(S_i^v) = pre_i^{v-1}(T_r)$, $post_i^v(S_i^v) \equiv post_i^{v-1}(S_i^{v-1})$.

4) If $S_i^{v-1} \equiv [\square(\ell=1,..,m) b_\ell;\alpha_\ell \to T_\ell]$, and $S_i^v \equiv \alpha_r;T_r$, then

$pre_i^v(S_i^v) = pre_i^{v-1}(\alpha_r)$, $post_i^v(S_i^v) = post_i^{v-1}(T_r)$.

5) If $S_i^{v-1} \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \to T_\ell]$, and $S_i^v \equiv \alpha_r;T_r;S_i^{v-1}$, then
$pre_i^v(S_i^v) = pre_i^{v-1}(\alpha_r)$, $post_i^v(S_i^v) = post_i^{v-1}(S_i^{v-1})$.

(1) through (5) take care of all clauses in the definition of "$\to$" except for the last one. Now we consider the last clause

6) Suppose $S^{v-1} = [S_1^{v-1} // ... // S_n^{v-1}]$, $S_i^{v-1} = T_1;T_2$ where $T_1 \not\equiv E$. Suppose also that $[S_1^{v-1} // .. // T_1 // S_{i+1}^{v-1} // .. // S_n^{v-1}] \to [S_1^v // .. // T'_1 // .. // S_n^v]$. Then $[S_1^{v-1} // .. // T_1;T_2 // .. // S_n^{v-1}] \to [S_1^v // .. // T'_1;T_2 // .. // S_n^v]$, assuming $T'_1 \not\equiv E$. Denote $[S_1^{v-1} // .. // T_1 // .. // S_n^{v-1}]$ by $\bar{S}^{v-1}$ and $[S_1^v // .. // T'_1 // .. // S_n^v]$ by $\bar{S}^v$.
Given assertion functions $pre_j^{v-1}, post_j^{v-1}$ for S^{v-1}, define assertion functions $\overline{pre}_j^{v-1}, \overline{post}_j^{v-1}$ for $\bar{S}^{v-1}$ as follows:

$\overline{pre}_j^{v-1} = pre_j^{v-1}$, $\overline{post}_j^{v-1} = post_j^{v-1}$ for all $j \neq i$; also every componen of T is a component of S_i^{v-1} . Thus define,

$pre_i^{v-1}(T) = pre_i^{v-1}(T)$, $\overline{post}_i^{v-1}(T) = post_i^{v-1}(T)$ for every component T of T_1.

Define functions $\overline{pre}_j^v, \overline{post}_j^v$ for $\bar{S}^v$ in terms of the functions $\overline{pre}_j^{v-1}, \overline{post}_j^{v-1}$ of $\bar{S}^{v-1}$ using clauses (1) through (5) just specified. Then define $pre_j^v, post_j^v$ as follows:

$pre_j^v = \overline{pre}_j^v$, $post_j^v = \overline{post}_j^v$, for all $j \neq i$;

for every component T of T'_1, $pre_i^v(T) = \overline{pre}_i^v(T)$, $post_i^v(T) = \overline{post}_i^v(T)$;
for every component T of T_2, $pre_i^v(T) = pre_i^{v-1}(T)$, $post_i^v(T) = post_i^{v-1}(T)$;
$pre_i^v(T'_1;T_2) = \overline{pre}_i^v(T'_1)$, $post_i^v(T'_1;T_2) = post_i^{v-1}(T_2)$.

Finally, we have to consider the case when $T'_1 \equiv E$, so that $S^v \equiv [S_1^v // .. // T_2 // .. // S_n^v]$. In this case we simply omit the clause "for every component T of T'_1 ..." above; also $T'_1;T_2$ should be replaced by T_2.

This completes the definition of $\mathrm{pre}_i^v, \mathrm{post}_i^v$. The following results can be shown more or less directly (if somewhat tediously) from the definitions:

a) $\mathrm{post}_i^v(S_i^v) = \mathrm{post}_{i-1}^{v-1}(S_i^{v-1})$ for all $i=1,..,n \quad v=2,..,u$.

b) $\mathrm{pre}_i^v, \mathrm{post}_i^v$ are assertion functions for $\{\mathrm{pre}_i^v(S_i^v)\}S_i^v\{\mathrm{post}_i^v(S_i^v)\}$.

What we shall try to show next is that τ^v, the state of the program after v-1 steps satisfies the predicate,

(A) $\quad [\bigwedge_{i=1}^{n} \mathrm{pre}_i^v(S_i^v)] \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,...,\varepsilon)]$.

This will prove that τ^u, the final state satisfies

$$\left[\bigwedge_{i=1}^{n} \mathrm{pre}_i^u(E)\right] \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,...,\varepsilon)] \text{ , i.e.}$$

$$\left[\bigwedge_{i=1}^{n} \mathrm{post}_i^u(E)\right] \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,...,\varepsilon)], \text{(since pre(E)} \Rightarrow \text{post(E))}$$

i.e. $\left[\bigwedge_{i=1}^{n} \mathrm{post}(P_i)\right] \wedge [\mathrm{Reduce}(h_1,...,h_n) = (\varepsilon,...,\varepsilon)]$, thus proving the consistency of the system.

Before we can prove (A), we need to face another problem: τ_i^v has two components, s_i^v, the (current) local state of P_i, and h_i^v, the communication sequence of P_i, whereas $\mathrm{pre}_i^v(S_i)$ is a predicate on σ_i, which has the following components: the local state of P_i, s_i; the communication sequence $h_i; h_{ji}$ $(j=1,...,n, j \neq i)$; and h'_{ji} $(j=1,..,n, j\neq i)$. Thus we need to add to τ_i^v, appropriate values for h_{ji} and h'_{ji}, before we can hope to prove (A). Recall that h_{ji} is the sequence from P_j that P_i has consumed, which, as pointed out in the discussion in section 2 following the rule of inference R11 for parallel composition, is identical to $h_j^v/\{(j,i),(i,j)\}$, the sequence that P_j has sent to P_i; and h_j^v is just the second component of τ_j^v. On the other hand h'_{ji} is an entirely mythical variable, introduced by the proof system. Thus what we shall show is that there exists some h'^v_{ji} $(j\neq i)$ such that τ_i^v, h_{ji}^v and h'^v_{ji} together satisfy the predicate (A). This will prove that there exists some h'^u_{ji}, such that τ^u and h'^u_{ji} together satisfy

$$\left[\bigwedge_{i=1}^{n} \mathrm{post}(P_i)\right] \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,...,\varepsilon)]$$

showing that τ^u is consistent with the final post-condition proved using the axiomatic system.

Next, we formalize the notation we have been using so far. τ^v is the state of the program after execution of v-1 steps of the computation sequence. τ_i^v, the ith component of τ^v, is the state of the ith process after execution of v-1 steps of the computation sequence.

τ_i^v consists of two components - s_i^v and h_i^v. We shall take $h_{ji}^v (j\neq i)$ to be equal to $h_j^v/\{(j,i),(i,j)\}$. We then need to show that there exists $h'^v_{ji}(j\neq i)$ such that

(B) $\models \bigwedge_{i=1}^{n} pre_i^v(S_i^v)[\sigma_i \leftarrow \sigma_i^v] \wedge [Reduce(h_1^v,\ldots,h_n^v) = (\varepsilon,\ldots,\varepsilon)]$.

where $\sigma_i^v = (s_i^v, h_i^v, h_{ji}^v(j=i),\ h'^v_{ji}(j=i))$, and $\sigma_i \leftarrow \sigma_i^v$ denotes "replace σ_i by σ_i^v" .

First suppose $v=1$. Then $pre_i^1(S_i^1) = pre(P_i)$; and $pre(P_i)$ is

$I_i \wedge (h_i=\varepsilon) \wedge (\bigwedge_{j\neq i} h_{ji}=\varepsilon) \wedge (\bigwedge_{j\neq i} C_{ji})$ .... (3)

Also $\tau^1=\tau^I$, τ_i^I being (s_i^I,ε), and s_i^I is such that $\models I_i[s_i\leftarrow s_i^I]$. Thus the first two clauses of (3) are satisfied by τ^1, and hence by σ^1. The third clause is also satisfied since $h_j^1=\varepsilon$, and hence $h_{ji}^1 = h_j^1/\{(j,i,),(i,j)\} = \varepsilon$. Moreover, since C_{ji} satisfies the first line of rule R11, there clearly exists some h'^1_{ji} such that $\models C_{ji}[h'_{ji}\leftarrow h'^1_{ji}]$. This proves the required result in the case of $v=1$. (The clause $Reduce(h_1^v,\ldots,h_n^v) = (\varepsilon,\ldots,\varepsilon)$ is trivially true since $h_i^v=\varepsilon, i=1,\ldots,n$).

Next, suppose $v>1$. We shall denote $v-1$ by w. Then we must show that if there exist $h'^w_{ji}(j,i=1,..,n,j\neq i)$ such that

$\bigwedge_{i=1}^{n} pre_i^w(S_i^w)[\sigma_i\leftarrow \sigma_i^w]\wedge[Reduce(h_1^w,\ldots,h_n^w) = (\varepsilon,..,\varepsilon)]$... (4)

is true, then there exist $h'^w_{ji}(j,i=1,..,n,j\neq i)$ such that

$\bigwedge_{i=1}^{n} pre_i^v(S_i^v)[\sigma_i\leftarrow\sigma_i^v]\wedge[Reduce(h_1^v,..,h_n^v) = (\varepsilon,\ldots,\varepsilon)]$...(5)

is true. Moreover, we have $(S_1^w//\ldots//S_n^w,\tau^w)\rightarrow(S_1^v//\ldots//S_n^v,\tau^v)$. Thus we have to prove the required result (5), assuming (4) for each of the seventeen clauses defining the relation "$\rightarrow$".

We shall write $(S^w,\tau^w)\xrightarrow[d]{}(S^v,\tau^v)$ to indicate that (S^w,τ^w) and (S^v,τ^v) are related according to the clause mumbered "d" in the definition of "$\rightarrow$". Thus $(S^w\rightarrow\tau^w)\xrightarrow[1]{}(S^v,\tau^v)$ indicates that

$S_i^w\equiv$ <u>skip</u>, $S_i^v\equiv E$, $S_j^w\equiv S_j^v$ for all $j\neq i$, and $\tau^v=\tau^w$.

We first consider the clause $[Reduce(h^v{}_1,..,h_n^v) = (\varepsilon,\ldots,\varepsilon)]$. A look at the different clauses defining "$\rightarrow$", shows that $(S^w,\tau^w)\xrightarrow[d]{}(S_1^v,\tau^v)$ implies that $(h_1^v,\ldots,h_n^v)=(h_1^w,\ldots,h_n^w)$, unless d is either 6 or 16. Thus, if d is not 6 or 16, the second clause of (4) gives us $Reduce(h_1^w,\ldots,h_n^w)=(\varepsilon,\ldots,\varepsilon)$ i.e. $Reduce(h_1^v,\ldots,h_n^v)=(\varepsilon,\ldots,\varepsilon)$.If d is 6, then there exist i,j such that

$h_k^v=h_k^w$ for all $k\neq i,j$; $h_i^v=h_i^w\vdash(j,i,m)$, $h_j^v=h_j^w\vdash(j,i,m)$ for some m.

Hence $\text{Reduce}(h_1^v,\dots,h_n^v) = \text{Reduce}(h_1^w,..,h_i^w \vdash (j,i,m),..,h_j^w \vdash (j,i,m),..,h_n^w)$
$= \text{Reduce}(\varepsilon,..,<(j,i,m)>,..,<(j,i,m)>,..\varepsilon)$
$= (\varepsilon,..,\varepsilon)$.

(Note: The above proof is valid only if for all $k(\neq i,j)$, (k,T,t) is an element of h_k^w implies $i,j \notin T$; this is, indeed, true since $S_i^1,..,S_i^w$ and $S_j^1,..,S_j^w$ are all different from E).

The analogous proof in the case $d = 16$ is left to the reader.

This verifies that $[\text{Reduce}(h_1^w,\dots,h_n^w)=(\varepsilon,\dots,\varepsilon)]$ implies $[\text{Reduce}(h_1^v,\dots,h_n^v)=(\varepsilon,\dots,\varepsilon)$. Next we must consider the clause $\left[\bigwedge_{i=1}^{n} pre_i^v(S_i^v)[\sigma_i \leftarrow \sigma_i^v]\right]$ of (5).

1. Suppose $(S^w,\tau^w) \xrightarrow[1]{} (S^v,\tau^v)$. Then $S_i^w \equiv skip$, $S_i^v \equiv E$, $S_j^v \equiv S_j^w$ for all $j \neq i$, and $\tau^v = \tau^w$. Moreover, $pre_i^v(S_i^v) = post_i^w(S_i^w)$, and since $S_i^w \equiv$ <u>skip</u>, $pre_i^w(S_i^w) \Rightarrow post_i^w(S_i^w)$; and $pre_j^v(S_j^v) = pre_j^w(S_j^w)$ for all $j \neq i$.
   Let $h'^v_{jk} = h'^w_{jk}$ for all $j,k \neq 1,..,n$ $j = k$. Then, clearly
   $$\left[\bigwedge_{j=1}^{n} pre_j^w(S_j^w)[\sigma_i \leftarrow \sigma_i^w]\right] \Rightarrow \left[\bigwedge_{j=1}^{n} pre_j^v(S_j^v)[\sigma_i \leftarrow \sigma_i^v]\right].$$

2. The proofs in the cases $(S^w,\tau^w) \xrightarrow[d]{} (S^v,\tau^v)$, in the cases $d = 2,3,4,5,7,8,9,15$ are equally straightforward and are left to the reader.

3. The remaining cases involve communication between processes, and are more difficult to deal with. Let us first consider the case when $d = 6$ i.e. $S_i^w \equiv P_j ?\, x$, $S_j^w \equiv P_i !\, y$, $S_i^v \equiv E$, $S_j^v \equiv E$,
   $$\tau_i^v = \tau_i^w[x \leftarrow \tau_j^w(y), h_i \leftarrow h_i \vdash (j,i,\tau_j^w))], \tau_j^v = \tau_j^w[h_j \leftarrow h_j \vdash (j,i,\tau_j^w(y))]$$
   It is trivial to see that $pre_k^w(S_k^w)[\sigma_k \leftarrow \sigma_k^w] \Rightarrow pre_k^v(S_k^v)[\sigma_k \leftarrow \sigma_k^v]$ for all $k \neq i,j$, since $S_k^v \equiv S_k^w$ and $\tau_k^v = \tau_k^w$.

   Next, we shall show $pre_j^w(S_j^w)[\sigma_j \leftarrow \sigma_j^w] \Rightarrow pre_j^v(S_j^v)[\sigma_j \leftarrow \sigma_j^v]$:
   Since $S_j^w \equiv P_i !\, y$, $pre_j^w(S_j^w)$ must satisfy the requirement
   $$pre_j^w(S_j^w) \Rightarrow \exists \bar{h}'_{ij} \cdot [pre_j^w(S_j^w)[h'_{ij} \leftarrow \bar{h}'_{ij}] \wedge [D(\bar{h}'_{ij})=(j,i)] \wedge [V(\bar{h}'_{ij}) = y]].$$

   Hence, we have,
   $$\exists h'^w_{ij} \cdot pre_j^w(S_j^w)[\sigma_j \leftarrow \sigma_j^w] \Rightarrow \exists \bar{h}'^w_{ij} \cdot [pre_j^w(S_j^w)[\sigma_j \leftarrow \bar{\sigma}_j^w] \wedge [D(\bar{h}'^w_{ij})=(j,i)] \wedge [V(\bar{h}'^w_{ij}) = y]] \quad ..(6)$$
   where σ_j^w is essentially (τ_j^w, h'^w_{ij}) and $\bar{\sigma}_j^w$ is essentially $(\tau_j^w, \bar{h}'^w_{ij})$.

   (In fact, of course, σ_j^w and $\bar{\sigma}_j^w$ contain other components such as h_{kj}, $k = 1,..,n, k \neq j$).

Also $\tau_j^V = \tau_j^W[h_j \leftarrow h_j \vdash (j,i,\tau_j^W(y))]$; and, since $S_j^W \equiv P_i ! \, y$,

$$\left[\mathrm{pre}_j^W(S_j^W) \wedge [D(h'_{ij}) = (i,j)] \wedge [V(h'_{ij}) = y]\right] \Rightarrow \mathrm{post}_j^W(S_j^W)[h_j \leftarrow h_j \vdash (j,i,y), h_{ij} \leftarrow h_{ij} \vdash (j,i,y), h'_{ij} \leftarrow \mathrm{rest}(h'_{ij})];$$

and $\mathrm{post}_j^W(S_j^W) = \mathrm{pre}_j^V(S_j^V)$; thus, if we take $\sigma_j^V = (\tau_j^V, \mathrm{rest}(\bar{h}'^W_{ij}))$, $\bar{h}'^W_{ij}$ being from line (6), then, we have shown that there exists h'^W_{ij} such that $\mathrm{pre}_j^V(S_j^V)[\sigma_j \leftarrow \sigma_j^V]$.

Finally, we must show that $\mathrm{pre}_i^V(S_i^V)[\sigma_i \leftarrow \sigma_i^V]$ also holds, for an appropriate σ_i^V. In order to do this, we need to make use of two results: a) There esists a sequence $\bar{h}_j$ such that

$$[h_j^V \le \bar{h}_j] \wedge [\exists \bar{s}_j, \bar{h}_{ij}, \bar{h}'_{ij}. \ \mathrm{post}(P_j)[\sigma_j \leftarrow \bar{\sigma}_j]]$$

where $\bar{\sigma}_j = (\bar{s}_j, \bar{h}_j, \bar{h}_{ij}, \bar{h}'_{ij})$. This follows from the fact that we have shown that there exists $\bar{h}'^V_{ij}$ such that

$$\mathrm{pre}_j^V(S_j^V)[\sigma_j \leftarrow \sigma_j^V], \text{ where } \bar{\sigma}_j^V = (\tau_j^V, h_{ij}^V, \bar{h}'^V_{ij}),$$

and the remarks at the end of the last section.

b) if $\bar{h}_{ji}$ is such that $C_{ji}[h'_{ji} \leftarrow \bar{h}_{ji}]$, and $\bar{h}_{ji} = h_{ji}^W \dashv\vdash \bar{h}'_{ji}$, and there exists h'^W_{ji} such that $\mathrm{pre}_i^W(S_i^W)[\sigma_i \leftarrow \sigma_i^W]$, where $\sigma_i^W = (\tau_i^W, h_{ji}^W, h'^W_{ji})$, then $\mathrm{pre}_i^W(S_i^W)[\sigma_i \leftarrow \bar{\sigma}_i^W]$, where $\bar{\sigma}_i^W = (\tau_i^W, h_{ji}^W, \bar{h}'_{ji})$.

We shall prove this result towards the end of this section.

Using results (a) and (b), we may assume that there exists h'^W_{ji} such that $\mathrm{first}(h'^W_{ji}) = (j,i,\tau_j^W(y))$, and

$$\mathrm{pre}_i^W(S_i^W)[\sigma_i \leftarrow \sigma_i^W], \text{ where } \sigma_i^W = (\tau_i^W, h_{ji}^W, h'^W_{ji}).$$

This implies, by the axiom for $P_i ? \, x$, that the following predicate is true: $\mathrm{post}_i^W(S_i^W)[\sigma_i \leftarrow \bar{\sigma}_i]$, where $\bar{\sigma}_i = (\tau_i^V, h_{ji}^V, \mathrm{rest}(h'^W_{ji}))$, and $\mathrm{pre}_i^V(S_i^V) = \mathrm{post}_i^W(S_i^W)$.

Thus we have shown that there exists an h'^V_{ji} (= $\mathrm{rest}(h'^W_{ji})$), such that $\mathrm{pre}_i^V(S_i^V)[\sigma_i, \sigma_i^V]$, $\sigma_i^V = (\tau_i^V, h_{ji}^V, h'^V_{ji})$.

4) Next, consider the case $d = 10$. Thus,

$S_i^W \equiv *[\square(\ell=1,..,m) b_\ell; \alpha_\ell \rightarrow T^\ell]$, $S_j^W \equiv P_i ! \, y$, $\models b_r(\tau_i^W)$, $\alpha_r = P_j ? \, x$, and $S_i^V \equiv \alpha_r; T^r; S_i^W$.

We have, $\exists h'^W_{ji} \, .\mathrm{pre}_j^W(S_j^W)[\sigma_j \leftarrow \sigma_j^W]$, $\sigma_j^W = (\tau_j^W, h_{ij}^W, h'^W_{ij})$.

Since $S_j^W \equiv P_i ! \, y$, we may assume $\mathrm{first}(h'^W_{ij})$ $(j,i,\tau_j^W(y))$.

Then, from the conditions satisfied by pre_j^W, we have

$$post_j^W(S_j^W)[\sigma_j \leftarrow \bar{\sigma}_j^W],\ \bar{\sigma}_j^W = (s_j^W, h_j^W \vdash (j,i,\tau_j^W(y)),\ h_{ij}^W \vdash (j,i,\tau_j^W(y)), \bar{h}'^W_{ij}),$$

where $\bar{h}'^W_{ij} = rest(h'^W_{ij})$.

Hence, by the remark at the end of the last section, we may assume that there exists $\bar{h}'_{ji}$ such that

$$\left[\exists \bar{\bar{h}}'_{ji}\,.\,h_{ji}^W \vdash (j,i,\tau_j^W(y)) \dashv \bar{\bar{h}}'_{ji} = \bar{h}'_{ji}\right] \wedge C_{ji}[h'_{ji} \leftarrow \bar{h}'_{ji}].$$

Then, by the result (b) above, we have,

$$pre_i^W(S_i^W)[\sigma_i \leftarrow \sigma_i^W],\ \text{where}\ \sigma_i^W = (\tau_i^W, h_{ji}^W, (j,i,\tau_j^W(y)) \dashv \bar{h}'_{ji}).$$

This, in conjunction with $\models b_r(\tau_i^W)$ gives us $pre_i^W(T^r)[\sigma_i \leftarrow \sigma_i^W]$, which gives us $pre_i^V(S_i^V)[\sigma_i \leftarrow \sigma_i^V]$.

The remaining cases may be dealt with in essentially similar fashion. Thus we have shown that for all v, $1 \leq v \leq u$, there exist h'^V_{jk} $(j,k=1,..,n,\ j \neq k)$, such that the following predicate is true: $\left[\bigwedge_{i=1}^{n} pre_i^V(S_i^V)[\sigma_i \leftarrow \sigma_i^V]\right]$, $[Reduce(h_1^V,..,h_n^V) = (\varepsilon,...,\varepsilon)]$, where $\sigma_i^V = (\tau_i^V, h_{ji}^V, h'^V_{ji})$, where $h_{ji}^V = h_j^V/_{\{(i,j),(j,i)\}}$. Thus, in particular, taking $v=u$, we have shown that there exists h'^u_{jk} such that

$$\bigwedge_{i=1}^{n} pre_i^u(S_i^i)[\sigma_i \leftarrow \sigma_i^u] \wedge [Reduce(h_1^u,..,h_n^u) = (\varepsilon,...,\varepsilon)]$$

is true; and since, $S_i^u \equiv E$ for all $i=1,..,n$, and hence $pre_i^u(S_i^u) = post_i^u(S_i^u)$, and $post_i^V(S_i^V) = post(P_i) = F_i$ for all $i=1,..,n$, and $v=1,..,u$ we have shown that the following predicate is true:

$$\left[\bigwedge_{i=1}^{n} F_i[\sigma_i \leftarrow \sigma_i^u]\right] \wedge [Reduce(h^i,..) = (\varepsilon,..)],$$

thus proving the consistency of the axiomatic system.

Two further remarks are in order: 1) In the above discussion, we have not considered the case $d=17$. The reason for this is the following: we may consider clause 17 in the definition of "$\rightarrow$" as an abbreviation of several other clauses, each of which will be quite similar to one of the other sixteen clauses. An example of such a clause would be:

$S_i \equiv [x:=e;\bar{S}_i]$; $S'_i \equiv \bar{S}_i$; $S'_j \equiv S_j$ for all $j \neq i$; $\tau'_i = \tau_i[x \leftarrow e]$; $\tau'_j = \tau_j$ for all $j \neq i$.

Another example would be:

$S_i \equiv [\square(\ell=1,..,m)\,b_\ell;\alpha_\ell \rightarrow T^\ell];\bar{S}_i$, and

$\models b_r(\tau_i)$, $\alpha_r = P_j?\,x$, $S_j \equiv P_i!\,y;\bar{S}_j$, and

$S'_i \equiv \alpha_r;T^r;\bar{S}_i$ and $S'_k \equiv S_k$ for all $k \neq i$, and $\tau' = \tau$.

Each of these cases may be handled in exactly the same fashion as the corresponding case among the first sixteen clauses. Note also that we do not need to consider the case where

$S_i \equiv [T_i;T_i'];\bar{S}_i$, since this may be rewritten as $T_i;[T_i';\bar{S}_i]$.

2) Finally, we still need to verify the result (b) stated earlier in the section and used extensively in the proof of consistency. Recall that the result states the following: if for some i,j, there exists $\bar{h}_{ji}'$ such that $C_{ji}[h_{ji}' \leftarrow \bar{h}_{ji}']$ is true, and $h_{ji}^{v} \dashv\vdash \bar{\bar{h}}_{ji}' = \bar{h}_{ji}'$, where $h_{ji}^{v} = h_j^{v}/\{(i,j),(j.i)\}$, and there exist $h_{k\ell}'^{v}(k, = 1,..,n,k \neq \ell)$ such that $\bigwedge_{m=1}^{n} pre_m^v(S_m^v)[\sigma_m \leftarrow \sigma_m^v]$ is true, where $\sigma_m^v = (\tau_m^v, h_{km}^v, h_{km}'^{v})$, then $\bigwedge_{m=1}^{n} pre_m^v(S_m^v)$ $[\sigma_m \leftarrow \bar{\sigma}_m^v]$ is also true, where $\bar{\sigma}_m^v = \sigma_m^v$ for all $m \neq i$, and $\bar{\sigma}_i^v = \sigma_i^v[h_{ji}'^{v} \leftarrow \bar{\bar{h}}_{ji}']$

The result will be proved by induction on v. The result is trivially true if $v = 1$, since $pre_m^1(S_m^1) = I_m \wedge (h_m = \varepsilon) \wedge (\bigwedge_{k \neq m} h_{km} = \ ) \ (\bigwedge_{k \neq m} C_{km})$ so that any $\bar{\sigma}_i^1 = (\tau_i^1, h_{ki} = \varepsilon, \bar{h}_{ki}')$, where $C_{ki}[h_{ki}' \leftarrow \bar{h}_{ki}']$, satisfies $pre_i^1(S_i^1)$ Next suppose the result is true for $v - 1, v > 1$. (Again, we shall denote $v - 1$ by w), and $(S^w, \tau^w) \rightarrow (S^v, \tau^v)$. If $S_i^w \equiv S_i^v$, i.e. the progress in the computation took place in some process other than i, the result is again trivial, since $pre_i^v(S_i^v) \equiv pre_i^w(S_i^w)$, and $\tau_i^v = \tau_i^w$, and $h_{ki}^v = h_{ki}^w$ for all $k \neq i$.

Next, suppose $S_i^w \equiv x:=e$, $S_i^v \equiv E$, $\tau_i^v = \tau_i^w[x \leftarrow e]$. Then $h_i^v = h_i^w$, $h_{ki}^v = h_{ki}^w$ for all $k \neq i$. Then, since the result is true for w, we may assume that $pre_i^w(S_i^w)[\sigma_i \leftarrow \bar{\sigma}_i^w]$ is true, where $\bar{\sigma}_i^w = (\tau_i^w, h_{ki}^w, \bar{h}_{ki}'^{w})$, where $\bar{h}_{ji}'^{w} = \bar{\bar{h}}_{ji}'$. This implies $pre_i^v(S_i^v)[\sigma_i \leftarrow \bar{\sigma}_i^v]$, where $\bar{\sigma}_i^v = (\tau_i^v, h_{ki}^v, \bar{h}_{ki}'^{w})$, which is the required result. All other cases not involving communication between P_i and P_j may be treated similarly.

Finally, suppose $S_i^w \equiv P_j?\,x$, $S_j^w \equiv P_i!\,y$, $S_i^v \equiv E$, $S_j^v \equiv E$, $\tau_i^v = \tau_i^w[x \leftarrow \tau_j^w(y), h_i \leftarrow h_i \vdash (j,i,\tau_j^w(y))]$, $\tau_j^v = \tau_j^w[h_j \leftarrow h_j \vdash (j,i,\tau_j^w(y))]$.
We also have $\bar{h}_{ji}' = h_{ji}^v \dashv\vdash \bar{\bar{h}}_{ji}' = h_{ji}^w \vdash (j,i, {}_j^w(y)) \dashv\vdash \bar{\bar{h}}_{ji}'$. Thus, by the inductive hypothesis the following predicate is true:

$pre_i^w(S_i^w)[\sigma_i \leftarrow \bar{\sigma}_i^w]$, where $\bar{\sigma}_i^w = (\tau_i^w, h_{ki}^w, h_{ki}'^{w})$, $h_{ji}'^{w} = (j,i,\tau_j^w(y)) \dashv \bar{\bar{h}}_{ji}'$
This gives, $post_i^w(S_i^w)[\sigma_i \leftarrow \bar{\sigma}_i^v]$, $\bar{\sigma}_i^v = (\tau_i^v, h_{ki}^v, h_{ki}'^{v})$, $h_{ji}'^{v} = \bar{\bar{h}}_{ji}'$; which, $post_i^w(S_i^w)$ being identical to $pre_i^v(S_i^v)$, is the required result. The other cases involving communication are dealt with similarly.

Thus we have proved result (b), and hence the consistency of the axiomatic system.

4. Completeness of the axiomatic system

This section is divied into three subsections: in 4.1 we introduce a new axiomatic system, and show that any proof in the new system can be converted into an equivalent proof in the system of section 2. In 4.2 we define a new operational semantics for CSP and show that it is equivalent to the operational semantics defined in section 3. In 4.3 we prove that the new axiomatic system is complete with respect to the operational semantics of section 4.2, thus proving the completeness of the system of section 2.

4.1. A new axiomatic semantics for CSP

Again, consider a CSP program $P::[P_1//\ldots//P_n]$, $P_1,\ldots P_n$ being the communicating sequential processes. h_i will denote, as before, the communication sequence of process P_i. $h_1,\ldots h_n$ will be the only sequences we shall deal with in the new axiomatic semantics. For ease of reference, we shall refer to the system of section 2 by the name SYS1, the system to be introduced in this section being named SYS2.

As in section 2, the axioms and rules of inference for constructs appearing in the individual processes will be stated in a form applicable to P_i.

A1. Input

$$\{p'\}P_j?\,x\,\{p\}$$

where $p' \equiv \left[\forall z.\ p^{x,h_i}_{z,h_i \vdash (j,i,z)}\right]$

The universal quantification is over the set of all integers, since we assume that only integers are being communicated. The reason for the universal quantification is that we have no information on what P_j will send to P_i, and hence must consider all possible values.

A2. Output

$$\left\{p^{h_i}_{h_i \vdash (i,j,y)}\right\}\ P_j!\,y\,\{p\}.$$

A3. Assignment

$$\{p^x_e\}x:=e\{p\}$$

A4. Skip

$$\{p\}\underline{\text{skip}}\{p\}$$

R1. Boolean guarded selection

$$\frac{\{p \wedge b_r\}\ S_r\ \{q\},\ r=1,..,m}{\{p\}[\square(\ell=1,..,m)b_\ell \rightarrow S_\ell]\{q\}}$$

R2. Boolean guarded repetition

$$\frac{\{p \wedge b_r\}\ S_r\ \{p\},\ r=1,..,m}{\{p\}*[\square(\ell=1,..,m)b_\ell \rightarrow S_\ell]\{p \wedge [\bigwedge_{\ell=1}^{m} \neg b_\ell]\}}$$

R3. I/O guarded selection

$$\frac{\{p \wedge b_r\}\alpha_r;S_r\{q\},\ r=1,..,m}{\{p\}[\square(\ell=1,..,m)b_r;\alpha_r \rightarrow S_r]\{q\}}$$

R4. I/O guarded repetition

$$\frac{\begin{array}{l}(p \wedge \neg\, b) \Rightarrow q \\ (p \wedge b) \Rightarrow q^{h_i}_{h_i} \vdash (i,\{j \mid \exists r \leq m.[b_r \wedge c(\alpha_r)=j]\},t) \\ \{p \wedge b_r\}\alpha_r;S_r\{p\},\ r=1,..,m\end{array}}{\{p\}*[\square(\ell=1,..,m)b_\ell;\alpha_\ell \rightarrow S_\ell]\{q\}}$$

where $b = \bigvee_{\ell=1}^{m} b_\ell$, and $c(\alpha_r)=j$ if α_r is $P_j?x$ or $P_j!y$.

R5. Sequential composition

$$\frac{\{p\}\ S_1\{q'\},\{q'\}S_2\{q\}}{\{p\}\ S_1;S_2\{q\}}$$

R6. Consequence

$$\frac{p \Rightarrow p',\{p'\}S\{q'\},\ q' \Rightarrow q}{\{p\}\ S\ \{q\}}$$

R7. Conjunction

$$\frac{\{p\}\ S\ \{q_1\},\{p\}\ S\ \{q_2\}}{\{p\}\ S\ \{q_1 \wedge q_2\}}$$

R8. Disjunction

$$\frac{\{p_1\}S\{q\},\{p_2\}S\{q\}}{\{p_1 \vee p_2\}\ S\ \{q\}}$$

R9. Parallel composition

$$\frac{\{I_i \wedge (h_i = \varepsilon)\}P_i\{F_i\},\ i = 1,..,n}{\left\{\bigwedge_{j=1}^{n} I_j\right\}[P_1 /\!/ \ldots /\!/ P_n]\left\{\bigwedge_{j=1}^{n} F_j \wedge [\mathrm{Reduce}(h\ ,\ldots,h_n) = (\varepsilon,\ldots,\varepsilon)]\right\}}$$

where I_i is a predicate on the local state s_i.

Next, we must show that, given a proof of

$$\left\{\bigwedge_{j=1}^{n} I_j\right\}[P_1 /\!/ \ldots /\!/ P_n]\left\{\bigwedge_{j=1}^{n} F_j \wedge [\mathrm{Reduce}(h_1,\ldots,h_n) = (\varepsilon,\ldots,\varepsilon)]\right\} \ldots (7)$$

in the above system, we can find an equivalent proof in the system SYS1 of section 2. We may assume that the proof of (7) is given by giving the proofs of

$$(\mathrm{II})\{I_i \wedge (h_i=\varepsilon)\}P_i\{F_i\},\ i = 1,..,n \quad \ldots. \ (8)$$

(The (II) denotes that the proof is based on SYS2. In general, we shall write (II) {p}S{q} to denote that {p}S{q} can be proved using SYS2. A similar notation will be used for SYS1). If we can show that (II) {p}S{q} ⇒ (I){p}S{q} , then our problem would be trivially solved; however, the above implication is not true for arbitrary constructs S that may appear in P_i, and we have to modify our approach. What we shall show is the following:

$$(\mathrm{II})\{p\}S\{q\} \Rightarrow (\mathrm{I})\{p \vee R_i\}\ S\ \{q \vee R_i\} \quad \ldots. \ (9)$$

where $R_i \equiv \bigwedge_{j \neq i} [f(h_{ji}) = g(h_i,i,j)+1]$,

where $f(h_{ji}) = 0$, if $h_{ji} = \varepsilon$

$f(\mathrm{rest}(h_{ji}))$, if $\mathrm{first}(h_{ji}) = (j,i,t)$

$f(\mathrm{rest}(h_{ji}))+1$, if $\mathrm{first}(h_{ji}) = (j,i,m)$ or $\mathrm{first}(h_{ji}) =$ (i,j,m) for some m.

and,

$g(h_i,i,j) = 0$, if $h_i = \varepsilon$

$g(\mathrm{rest}(h_i))+1$, if $\mathrm{first}(h_i) = (j,i,m)$ or (i,j,m) for some m

$g(\mathrm{rest}(h_i))$ otherwise.

We shall prove (9) by induction on the structure of S. Suppose, S ≡ skip. Then, (II) {p}S{q} ⇒ [p ⇒ q] ⇒ (I){p}S{q}. Also, we have $R_i \Rightarrow R_i$, and hence (II) $\{R_i\}S\{R_i\}$; using the rules of disjunction and consequence, this gives, (II) {p}<u>skip</u>{q} ⇒ (I){p ∨ R_i}<u>skip</u> {q ∨ R_i}. If S ≡ x:=e, the result follows in similar fashion, using the fact $R_{i\,e}^{x} \equiv R_i$. Next, suppose

$S \equiv P_j ! y$. Then, from the axiom for output, we have, $(\Pi)\ \{p\}P_j ! y \{q\} \Rightarrow \left[p \Rightarrow q^{h_i}_{h_i \vdash (i,j,y)}\right]$. In order to be able to prove $(I)\ \{p \vee R_i\}P_j ! y \{q \vee R_i\}$, we must have the following:

$$[p \vee R_i] \Rightarrow \exists \bar{h}'_{ji} \cdot [[p \vee R_i]^{h'_{ji}}_{\bar{h}'_{ji}} \wedge [D(\bar{h}'_{ji}) = (i,j)] \wedge [V(\bar{h}'_{ji}) = y]] \quad \ldots (10)$$

and, $[[p \vee R_i] \wedge [D(h'_{ji}) = (i,j)] \wedge [V(h'_{ji}) = y]] \Rightarrow [q \vee R_i]^{h_i,\ h_{ji},\ h'_{ji}}_{h_i \vdash (i,j,y),\ h_{ji} \vdash (i,j,y),\ \mathrm{rest}(h'_{ji})} \ldots$ (1

(10) is trivially true, since p and R_i do not impose any restrictions on h'_{ji}. We may rewrite $[q \vee R_i]^{h_i, \ldots}_{\ldots}$ of (11) as follows:

$$q^{h_i}_{h_i \vdash (i,j,y)} \vee R_i{}^{h_i,\ h_{ji}}_{h_i \vdash (i,j,y),\ h_{ji} \vdash (i,j,y)},$$ since q has no references to h_{ji}, h'_{ji}, and R_i has no references to h'_{ji}. From the definition of R_i, we have $R_i \Rightarrow R_i{}^{h_i,\ h_{ji}}_{h_i \vdash (i,j,y),\ h_{ji} \vdash (i,j,y)}$. Also, we have $p \Rightarrow q^{h_i}_{h_i \vdash (i,j,y)}$, since $(\Pi)\ \{p\}P_j ! y \{q\}$ is true. From these results, it is easy to see that (11) is true, and hence we have $(I)\ \{p\}P_j ! Y \{q\}$. The case $S \equiv P_j ? y$ may be handled in similar fashion. The proof in case $S \equiv T;T'$ is straightforward. Next, consider the case $S \equiv [\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]$. $(\Pi)\ \{p\}S\{q\}$ implies $(\Pi)\ \{p \wedge b_r\}T^r\{q\}$, $r = 1, \ldots, m$ Hence, we have, $(I)\ \{(p \wedge b_r) \vee R_i\}T^r\{q \vee R_i\}$. Then, we can conclude $(I)\ \{p \vee R_i\}S\{q \vee R_i\}$, provided we can show the following:

$$[p \vee R_i] \Rightarrow \exists \bar{\sigma}_i \cdot [[q \vee R_i]^{\sigma_i}_{\bar{\sigma}_i} \wedge [h_i \sqsubseteq \bar{h}_i]] \quad \ldots (12)$$

The above implication is easily seen to be true, since any $\bar{\sigma}_i$ will satisfy the predicate $R_i{}^{\sigma_i}_{\bar{\sigma}_i}$ if $f(\bar{h}_{ji}) = g(\bar{h}_i, i, j)+1$ for all $j \neq i$.

This proves $(I)\ \{p \vee R_i\}[\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]\{q \vee R_i\}$. The other cases can be handled in exactly the same fashion.

Thus, we have shown, $(\Pi)\ \{p\}S\{q\} \Rightarrow (I)\ \{p \vee R_i\}S\{q \vee R_i\}$, for all constructs S which may appear in P_i. Hence, we have,

$$(\Pi)\ \{I_i \wedge (h_i = \varepsilon)\}P_i\{F_i\} \Rightarrow (I)\ \{[I_i \wedge (h_i = \varepsilon)] \vee R_i\}P_i\{F_i \vee R_i\} \Rightarrow$$
$$(I)\ \{I_i \wedge [h_i = \varepsilon] \wedge [h_{ji} = \varepsilon]\}P_i\{F_i \vee R_i\}. \quad \ldots (13)$$

Also, we have, $(II)\ \{\bigwedge_{i=1}^{n} I_i\}P\{\bigwedge_{i=1}^{n} F_i \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,\ldots,\varepsilon)\} \Rightarrow$

$$(\Pi)\ \{I_i \wedge (h_i = \varepsilon)\}P_i\{F_i\},\ i = 1,..,n \quad \Rightarrow$$
$$(I)\ \{I_i \wedge [h_i = \varepsilon] \wedge [h_{ji} = \varepsilon]\}P_i\{F_i \vee R_i\},\quad i = 1,..,n \quad \Rightarrow$$
$$(I)\{\bigwedge_{i=1}^{n} I_i\}P\{\bigwedge_{i=1}^{n} [F_i \vee R_i] \wedge [\mathrm{Reduce}(h_1,..,h_n) = (\varepsilon,\ldots,\varepsilon)]\} \ldots (14$$

From the definition of Reduce and R_i, it is see to see the following:

$$[\mathrm{Reduce}(h_1,\ldots,h_n)=(\varepsilon,\ldots,\varepsilon)] \Rightarrow [\bigwedge_{j\neq i} f(h_{ji})=g(h_i,i,j)] \Rightarrow [\bigwedge_{i=1}^{n} \neg R_i].$$

Thus (14) gives us,

$$(\text{II})\ \{\bigwedge_{i=1}^{n} I_i\}P\{\bigwedge_{i=1}^{n} F_i \wedge [\mathrm{Reduce}(h_1,\ldots,h_n) = (\varepsilon,\ldots,\varepsilon)]\} \Rightarrow$$

$$(\text{I})\{\bigwedge_{i=1}^{n} I_i\}P\{\bigwedge_{i=1}^{n} F_i \wedge [\mathrm{Reduce}(h_1,\ldots,h_n)=(\varepsilon,\ldots,\varepsilon)]\} \qquad \ldots(1$$

which proves that given a proof of a program P in SYS2, we can give an equivalent proof in SYS2.

4.2. An alternative operational semantics for CSP

In this subsection, we shall define a new operational semantics for CSP. To distinguish this semantics from the semantics defined in section 3, we shall refer to the semantics to be introduced now as SEM2, and the semantics of section 3 as SEM1. SEM2 will be defined by specifying a relation corresponding to the various CSP constructs. The relation corresponding to a construct S, which will be also denoted by S will be a set of pairs of the kind (τ,τ'), indicating that if we execute S starting in the initial state τ, then one of the possible final states is τ'.

Suppose $S \equiv$ skip Then, clearly the relation corresponding to S is $\{(\tau,\tau)\}$. We shall write this as,

$$\underline{\text{skip}} = \{(\tau,\tau)\}.$$

Next, consider the assignment statement:

$$x:=e = \{(\tau,\tau') \mid \tau'=\tau[x \leftarrow e]\}$$

where $\tau[x\leftarrow e]$ is the state obtained from τ by replacing the value of the variable x by the value of the expression e.

$$T;T' = \{(\tau,\tau'') \mid \exists\tau'.[(\tau,\tau')\in T \wedge (\tau',\tau'')\in T']\}.$$

$$P_j!\,y = \{(\tau,\tau') \mid \tau'=\tau[h_i \leftarrow h_i \vdash (i,j,y)]\},$$

where $\tau[h_i \leftarrow h_i \vdash (i,j,y)]$ is the state got by extending the sequence h_i by (i,j,m), m being the value of y in the state τ. (We are, of cours defining the relations corresponding to constructs appearing the process P_i).

$$P_j?\,x = \{(\tau,\tau') \mid \exists z.\tau'=\tau[x\leftarrow z, h_i \leftarrow h_i \vdash (j,i,z)]\}$$

$$[\square(\ell=1,\ldots,m)b_\ell \rightarrow T^\ell] = \{(\tau,\tau') \mid \exists r\leq m.[b_r(\tau) \wedge (\tau,\tau')\in T^r]\}$$

where $b_r(\tau)$ indicates that the boolean expression b_r has the value "true" in the state τ.

$$[\square(\ell=1,..,m)b_\ell;\alpha_\ell \rightarrow T^\ell]=\{(\tau,\tau')\,|\,\exists r\leq m.[b_r(\tau)\wedge(\tau,\tau')\in[\alpha_r;T^r]]\}.$$

In order to define boolean guarded loops, we introduce a preliminary definition:

$$**[\square(\ell=1,..,m)b_\ell \rightarrow T^\ell]=\{(\tau,\tau')\,|\,\exists k.\exists\tau_0,\dots,\tau_k.[[\tau=\tau_0]\wedge[\tau=\tau_k]\wedge [\forall k'<k.[(\tau_{k'},\tau_{k'+1})\in[\square(\ell=1,..,m)b_\ell\rightarrow T^\ell]]]]\}$$

$$*[\square(\ell=1,..,m)b_\ell \rightarrow T^\ell)=\{(\tau,\tau')\,|\,[(\tau,\tau')\in**[\square(\ell=1,..,m)b_\ell\rightarrow T^\ell]]\wedge \left[\bigwedge_{r=1}^{m}\neg b_r\right](\tau')\}.$$

Similarly, to define I/O guarded loops, we first introduce an auxiliary definition:

$$**[\square(\ell=1,..,m)b_\ell;\alpha_\ell \rightarrow T^\ell]=\{(\tau,\tau')\,|\,\exists k.\exists\tau_0,..,\tau_k.[[\tau=\tau_0]\wedge[\tau'=\tau_k]\wedge [\forall k'<k.[(\tau_{k'},\tau_{k'+1})\in[\square(\ell=1,..,m)b_\ell;\alpha_\ell\rightarrow T^\ell]]]]\}$$

and

$$*[\square(\ell=1,..,m)b_\ell;\alpha_\ell \rightarrow T^\ell]=\{(\tau,\tau'')\,|\,\exists\tau'.[[(\tau,\tau')\in**[\square(\ell=1,..,m)b_\ell;\alpha_\ell\rightarrow T^\ell]] [[[\tau'=\tau'']\wedge\left[\bigwedge_{r=1}^{m}\neg b_r\right](\tau')]\vee \left[\left[\bigvee_{r=1}^{m}b_r\right](\tau')\wedge[\exists T.[[\tau''=\tau'[h_i\leftarrow h_i\vdash(i,T,t)]]\wedge[T=\{j\,|\,\exists r\leq m.b_r(\tau')\wedge j=c(\alpha_r)\}]]]]\right]\}.$$

Finally,

$$[P_1/\!/../\!/P_n]=\{((\tau_1,..,\tau_n),(\tau'_1,..,\tau'_n))\,|\,\forall k\leq n[[h_k=\varepsilon]\wedge[(\tau_k,\tau'_k)\in P_k]]\wedge [\text{Reduce}(h'_1,..,h'_n)=(\varepsilon,\dots,\varepsilon)]\}$$

where h_k is the sequence component of τ_k, and h'_k that of τ'_k.

Next, we must show the equivalence of the operational semantics SEM2 just introduced, and the semantics SEM1 defined in section 3. The reader who is not interested in the proof of equivalence of SEM1 and SEM2 may skip the rest of this section and proceed to section 4.3.

We first introduce a relation "$\rightarrow$" which will be analogous to the relation defined in section 3; the difference between the two is that the relation to be defined next corresponds to the execution of one step of a process considered in isolation, whereas the relation defined earlier deals with all the processes.

Definition: $(S_i,\tau_i)\rightarrow(S'_i,\tau'_i)$ if any of the following clauses is satisfied:

1) $S_i\equiv$ <u>skip</u> , $S'_i\equiv E$, $\tau'_i=\tau_i$.

2) $S_i\equiv x:=e$, $S'_i\equiv E$, $\tau'_i=\tau_i[x\leftarrow e]$.

3) $S_i \equiv P_j ! y$, $S'_i \equiv E$, $\tau'_i = \tau_i[h_i \leftarrow h_i \vdash (i,j,y)]$.

4) $S_i \equiv P_j ? x$, $S' \equiv E$, $\tau'_i = \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)$, for some m.

5) $S_i \equiv [\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]$, $\models b_r(\tau_i)$, $S'_i \equiv T^r$, $\tau'_i = \tau_i$.

6) $S_i \equiv *[\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]$, $\models b_r(\tau_i)$, $S'_i \equiv T^r;S_i$, $\tau'_i = \tau_i$.

7) $S_i \equiv *[\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]$, $\models \bigwedge_{\ell=1}^{m} \neg b_\ell(\tau_i)$, $S'_i \equiv E$, $\tau'_i = \tau_i$.

8) $S_i \equiv [\square(\ell=1,..,m) b_\ell;\alpha_\ell \rightarrow T^\ell]$, $\models b_r(\tau_i)$, $S'_i \equiv \alpha_r;T^r$, $\tau'_i = \tau_i$.

9) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \rightarrow T^\ell]$, $\models b_r(\tau_i)$, $S'_i \equiv \alpha_r;T^r;S_i$, $\tau'_i = \tau_i$.

10) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \rightarrow T^\ell]$, $\models \bigwedge_{\ell=1}^{m} \neg b_\ell(\tau_i)$, $S'_i \equiv E$, $\tau'_i = \tau_i$.

11) $S_i \equiv *[\square(\ell=1,..,m) b_\ell;\alpha_\ell \quad T^\ell]$, $\models \bigvee_{\ell=1}^{m} b_\ell(\tau_i)$,

$$S'_i \equiv E, \tau'_i = \tau_i[h_i \leftarrow h_i \vdash (i,T,t)], T = \{j \mid \exists r \leq m. [\models b_r(\tau_i) \wedge j = c(\alpha_r)]\}$$

2) $S_i \equiv T;T'$, and $(T,\tau_i) \rightarrow (T'',\tau'_i)$, $T'' \equiv E$ and $S'_i \equiv T'$ or $T'' \not\equiv E$ and $S'_i \equiv T'';T'$.

Next, we state a useful lemma:

<u>Lemma:</u> If $(\tau_i^I,\tau_i^F) \in P_i$, then there exists a sequence of pairs $(\tau_i^1,S_i^1),\ldots,(\tau_i^k,S_i^k)$ such that $\tau_i^1 = \tau^I_i$, $S_i^1 \equiv P_i$, $\tau^k_i = \tau^F_i$, $S^k_i \equiv E$, and for all $r < k$: $(\tau^r_i, S^r_i) \rightarrow (\tau_i^{r+1}, S_i^{r+1})$.

The proof is by a simple induction on the structure of P_i, and we omit the details. A sequence of the kind introduced in the above lemma will be called a "local computation sequence for P_i" to distinguish it from the computation sequence to be called a "global computation sequence", for the entire program introduced in section 3. Also, in what follows we use "→" to denote the relation defined above, as well as the relation defined in section 3; the context will make clear which relation is intended; thus in $(S_i,\tau_i) \rightarrow (S'_i,\tau_i)$, the intende relation is the one defined above; and, in $(S_1//..//S_n,\tau) \rightarrow (S'_1//..//S'_n,\tau')$ the intended relation is the one of section 3. And, $(S_i,\tau_i) \xrightarrow[d]{} (S'_i,\tau'_i)$ will indicate that (S_i,τ_i) and (S',τ'_i) are related according to clause (d) in the definition of "→". The next lemma states the equivalence of SEM1 and SEM2.

<u>Lemma</u>: If $(S_1^1,\tau_1^1),\ldots,(S_1^{u_1},\tau_1^{u_1})$,..., $(S_n^1,\tau_n^1),\ldots,(S_n^{u_n},\tau_n^{u_n})$ are local computation sequences for $P_1,..,P_n$, so that $S_i^1 \equiv P_i$, $h_i^1 = \varepsilon$, $S_i^{u_i} \equiv E$, and $[\text{Reduce}(h_1^{u_1},\ldots,h_n^{u_n}) = (\varepsilon,\ldots,\varepsilon)]$, then there exists a global computation sequence $[(S^1,\tau^1),\ldots,(S^u,\tau^u)]$, such that $S^1 \equiv S_1^1//..//S_n^1$, $\tau^1 = (\tau_1^1,\ldots,\tau_n^1)$, $S^u \equiv E//..//E$, $\tau^u = (\tau_1^{u_1},\ldots,\tau_n^{u_n})$, and for all $v \leq u$, there exist $v_1,\ldots,v_n$, such that $v_i \leq u_i$, $i = 1,..,n$, and

$S_i^u \equiv S_i^{v_i}$, $\tau^v = (\tau_1^{v_1},\dots,\tau_n^{v_n})$, [Reduce $h_1^v,\dots,h_n^v) = (\varepsilon,\dots,\varepsilon)$] , and there exists $\bar{h}_1^v,\dots,\bar{h}_n^v$ such that,

$h_i^{u_i} = h_i^v \dashv\vdash \bar{h}_i^v$ for all $i = 1,\dots,n$, and [Reduce$(\bar{h}_1^v,\dots,\bar{h}_n^v) = (\varepsilon,\dots,\varepsilon)$].

Proof: We shall prove the result by constructing the sequence $[(S^1,\tau^1),\dots,(S^u,\tau^u)]$. First, define $S^1 \equiv S_1^1 /\!/ \dots /\!/ S_n^1$, $\tau^1 = (\tau_1^1,\dots,\tau_n^1)$. Thus the result stated in the lemma clearly holds for $v = 1$.

Next, suppose we have constructed (S^v,τ^v), and that

$S^v \equiv S_1^{v_1} /\!/ \dots /\!/ S_n^{v_n}$, $\tau^v = (\tau_1^{v_1},\dots,\tau_n^{v_n})$, Reduce$(h_1^v,\dots,h_n^v) = (\varepsilon,\dots,\varepsilon)$],

$h_i^{u_i} = h_i^v \dashv\vdash \bar{h}_i^v$, and [Reduce$(\bar{h}_1^v,\dots,\bar{h}_n^v) = (\varepsilon,\dots,\varepsilon)$].

Then, we have the following possibilities:

1) $v_i = u_i$ for all $i = 1,\dots,n$. Then $S_i^{v_i} \equiv E$, $\tau_i^v = \tau_i^{v_i} = \tau_i^{u_i}$, and the construction of the global computation sequence with the required properties is complete.

2) If (1) does not apply and there exists some j such that $(S_j^{v_j},\tau_j^{v_j}) \xrightarrow[d]{} (S_j^{v_j+1},\tau_j^{v_j+1})$ for $d = 1,2,5,6,7$ or 10, then define

$S^{v+1} \equiv (S_1^{v_1} /\!/ \dots /\!/ S_j^{v_j+1} /\!/ \dots /\!/ S_n^{v_n})$, $\tau^{v+1} = (\tau_1^{v_1},\dots,\tau_j^{v_j+1},\dots,\tau_n^{v_n})$,

and $\bar{h}_i^{v+1} = \bar{h}_i^v$. If there is more than one j satisfying the stated condition, we choose any one of them, and define (S^{v+1},τ^{v+1}), and $\bar{h}_i^{v+1}$ as above. It is then easy to see that the lemma holds for $v+1$.

3) If neither (1) nor (2) holds, then note first that for all j such that $\bar{h}_j^v = \varepsilon$, we have $S_j^{v_j} \equiv E \equiv S_j^{u_j}$ and $\tau_j^v = \tau_j^{u_j}$. Also, the facts that $h_i^{u_i} = \bar{h}_i^v \dashv\vdash \bar{h}_i^v$, $i = 1,\dots,n$, and [Reduce$(\bar{h}_1^v,\dots,\bar{h}_n^v) = (\varepsilon,\dots,\varepsilon)$], in conjunction imply that at least one of the following clauses is true:

a) There exists j such that $S_j^{v_j} \equiv *[\square(\ell=1,\dots,m)b_\ell;\alpha_\ell \to T^\ell]$, $\models \bigvee_{\ell=1}^{m} b_\ell(\tau_j^{v_j})$,

$S_j^{v_j+1} \equiv E$, first$(\bar{h}_j^v) = (j,T,t)$, $T = \{k \mid \exists r \cdot [r \leq m \wedge \models b_r(\tau_j^{v_j}) \wedge (k = c(\alpha_r))]\}$,

and $\bar{h}_k^v = \varepsilon$ for all $k \in T$ and hence $S_k^{v_k} \equiv E$ for all $k \in T$.

In this case, define $S^{v+1} \equiv (S_1^{v_1} /\!/ \dots /\!/ E /\!/ \dots /\!/ S_n^{v_n})$, (i.e. all components of S^{v+1} except the jth one is the same as the corresponding component of S^v, the jth component of S^{v+1} being E); and $\tau^{v+1} = (\tau_1^{v_1},\dots,\tau_j^{v_j+1},\dots,\tau_n^{v_n})$; and $\bar{h}_i^{v+1} = \bar{h}_i^v$ for all $i \neq j$, and $\bar{h}_j^{v+1} =$ rest$(\bar{h}_j^v$

Again, if this clause applies, it should be clear that the lemma is satisfied for $v+1$.

b) There exists i,j such that $S_i^{v_i} \equiv P_j ? x$, $S_j^{v_j} = P_i ! y$, $\tau_j^{v_j}(y) = m$,

first$(\bar{h}_i^v) = (j,i,m)$, first$(\bar{h}_j^v) = (j,i,m)$.

In this case, define $S_i^{v+1} \equiv E$, $S_j^{v+1} \equiv E$, $S_k^{v+1} \equiv S_k^v \equiv S_k^{v_k}$ for all $k \neq i,j$; and $\tau_i^{v+1} = \tau_i^{v_i+1}$, $\tau_j^{v+1} = \tau_j^{v_j+1}$; and $\bar{h}_i^{v+1} = \text{rest}(\bar{h}_i^v)$, $\bar{h}_j^{v+1} = \text{rest}(\bar{h}_j^v)$, $\bar{h}_k^{v+1} = \bar{h}_k^v$ for all $k \neq i,j$.

Once again, it is straightforward to check that the lemma is satisfied for v+1, if this clause applies.

c) There are several other possible cases corresponding to I/O guarded selections, and I/O guarded loops; we consider a typical case, leaving the remaining ones, which may be dealth with similarly, to the interested reader:

There exist i,j such that $S_i^{v_i} \equiv *[\square(\ell=1,..,m)\, b_\ell;\alpha_\ell \rightarrow T^\ell]$, $S_j^{v_j} \equiv *[\square(p= ,..,m')\, b'_p;\alpha'_p \rightarrow T'^p]$, $\models b_r(\tau_i^{v_i})$, $\models b_{r'}(\tau_j^{v_j})$, $\alpha_r = P_j?\,x$, $\alpha'_{r'} = P_i!\,y$, $\tau_j^{v_j}(y) = m$, $S_i^{v_i+1} \equiv \alpha_r;T^r;S_i^{v_i}$, $S_j^{v_j+1} \equiv \alpha_{r'};T'^{r'};S_j^{v_j}$ and first$(\bar{h}_j^{v_j}) = (j,i,m)$ and first$(\bar{h}_i^{v_i}) = (j,i,m)$. In this case, define,

$S_i^{v+1} \equiv \alpha_r;T^r;S_i^{v_i}$, $S_j^{v+1} \equiv \alpha'_{r'};T'^{r'};S_j^{v_j}$, and $S_k^{v+1} \equiv S_k^{v_k}$ for all $k \neq i,j$ and $\tau^{v+1} = \tau^v$; $\bar{h}_k^{v+1} = \bar{h}_k^v$ for all k.

Again, it is routine to verify that the lemma is satisfied for $v+1$, if this clause applies.

Note also that irrespective of whether S^{v+1} is defined according to clause (2), or according to one of the clauses in (3), the following is true:

if $S^v \equiv (S_1^{v_1} // .. // S_1^{v_n})$, and $S^{v+1} \equiv (S_1^{v'_1}, .., S_n^{v'_n})$, then $v'_i \geq v_i$ for all $i = 1,..,n$; and there exists $j \leq n$ such that $v'_j > v_j$.

Hence, $u_1,..,u_n$ each being finite, and v_i being less than or equal to u_i, we may conclude that ultimately clause (1) of the above constructio will apply, giving us the global computation sequence

$(S_1^1 // .. // S_n^1, \tau^1), .., (S_1^u // .. // S_n^u, \tau^u)$ such that $S_i^1 \equiv P_i$ for $i = 1,..,n$; $S_i^u \equiv E$ for $i = 1,..,n$; $\tau^1 = (\tau_1^I,..,\tau_n^I)$, $\tau^u = (\tau_1^f,..,\tau_n^F)$. This concludes the proof of the lemma.

The lemma along with the earlier one shows that if $((\tau^I,..,\tau_n^I),(\tau_1^F,..,\tau_n^F))$ belongs to the relation corresponding to the program $P_1 // .. // P_n$ as defined by SEM2, then there exists a (global) computation sequence of $P_1 // .. // P_n$, as defines by SEM1, such that starting in the initial state $(\tau_1^I,..,\tau_n^I)$, and following the computation sequence leads to the final state $(\tau_1^F,..,\tau_n^F)$.

In order to complete the proof of equivalence of SEM1 and SEM2, we ought to prove the converse of the above result. This converse result is not, however, needed for the purposes of this paper, as the following argument shows: given a program $P_1//..//P_n$, and a predicate p on the initial state of $P_1//..//P_n$, we shall define (in secteion 4.3) a predicate $sp(p,P_1//..//P_n)$ such that for every state τ' satisfying $sp(p,P_1//..//P_n)$, there exists a τ satisfying p, and (τ,τ') belongs to the relation corresponding to $P_1//..//P_n$ as defined by SEM2; and show that $\{p\}P_1//..//P_n\{sp(p,P_1//..//P_n)\}$ is provable using the axiomatic system SYS2. This will prove the (relative) completeness of SYS2 with respect to SEM2, and hence, by the result stated in the last paragraph, with respect to SEM1.

4.3 Completeness of the axiomatic system

In this section we prove the (relative) completeness of SYS2 with respect to SEM2. First we define the notion of the "strongest post-condition" sp(p,S), given a predicate (the precondition) p and a construct S that may appear in the process P_i.

Definition: Given a predicate p and a construct S which may appear in P_i, sp(p,S) is defined as follows:

1) sp(p,skip) = p;

2) $sp(p,x:=e)=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge \tau_i' = \tau_i[x\leftarrow e]]\}$;

(Note:We treat predicates as characterising sets of states; thus p above characterises the set $\{\tau_i \mid \tau_i \in p\}$, and we define sp(p,x:=e) by specifying the set that it characterises).

3) $sp(p,P_j!\,y)=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge \tau_i' = \tau_i[h_i \vdash (i,j,y)]]\}$;

4) $sp(p,P_j?\,x)=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge \exists m.\tau_i' = \tau_i[x\leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)]]\}$;

5) $sp(p,[\Box(\ell=1,..,m)b_\ell \rightarrow T^\ell])=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge [\exists r\leq m.[\ b_r(\tau_i)] \wedge [\tau_i' \in sp(\tau_i,T^r)]]]]\}$;
where, in $sp(\tau_i,T^r)$, τ_i denotes a set consisting of the single state τ_i.

6) $sp(p,**[\Box(\ell=1,..,m)b_\ell \rightarrow T^\ell])=\{\tau_i' \mid \exists\tau_i.\exists r.[\exists \tau_1,..,\tau_r.[\tau_1=\tau_i \wedge \tau_r=\tau_i' \wedge$

$\forall r'<r.[\exists r''<m.[b_{r''}(\tau_{r'}) \wedge \tau_{r'+1} \in sp(\tau_{r'},T^{r''})]]]]\}$;

7) $sp(p,*[\Box(\ell=1,..,m)b_\ell \rightarrow T^\ell])=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge \tau_i' \in sp(\tau_i,**[(\ell=1,..,m)b_\ell \rightarrow T^\ell]) \wedge$

$\left[\bigwedge_{\ell=1}^{m} \neg b_\ell(\tau_i')\right]]\}$;

8) $sp(p,[\Box(\ell=1,..,m)b_\ell;\alpha_\ell \rightarrow T^\ell])=\{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge [\exists r\leq m.[\ b_r(\tau_i) \wedge \tau_i' \in sp(\tau_i,\alpha_r;T^r)]]]\}$

9) $sp(p, **[\square(\ell=1,..,m) b_\ell ; \alpha_\ell \rightarrow T^\ell]) = \{\tau_i' \mid \exists \tau_i . \exists r . [\exists \tau_1, .., \tau_r . [\tau_1 = \tau_i \wedge \tau_r = \tau_i' \wedge$

$\forall r' < r . [\exists r'' < m . [\, b_{r''}(\tau_{r'}) \wedge \tau_{r'+1} \in sp(\tau_{r'}, \alpha_{r''} ; T^{r''})]]]]\}$

10) $sp(p, *[\square(\ell=1,..,m) b_\ell ; \alpha_\ell \rightarrow T^\ell]) = \{\tau_i' \mid \exists \tau_i . [\tau_i \in p \wedge \tau_i' \in sp(\tau_i, **[\square(\ell=1,..,m) b_\ell ; \alpha_\ell \rightarrow T^\ell])$

$\left[\bigwedge_{\ell=1}^{m} \neg b_\ell(\tau_i')\right]]\} \quad \cup$

$\{\tau_i' \mid \exists \tau_i, \tau_i'' . [\tau_i \in p \wedge \tau_i'' \in sp(\tau_i, **[\square(\ell=1,..,m) b_\ell ; \alpha_\ell \rightarrow T^\ell]) \wedge$

$\left[\bigvee_{\ell=1}^{m} b_\ell(\tau_i'')\right] \wedge \left[\exists T . \left[\tau_i' = \tau_i''[h_i \leftarrow h_i \vdash (i,T,t)] \wedge T = \{j \mid \exists r \leq m .\ b_r(\tau_i'')\right.\right.$

$\left.\left. j = c(\alpha_r)\}\right]\right]]\} \;;$

11) $sp(p, T^1 ; T^2) = \{\tau_i' \mid \exists \tau_i, \tau_i'' . [\tau_i \in p \wedge \tau_i'' \in sp(\tau_i, T^1) \wedge \tau_i' \in sp(\tau_i'', T^2)]\}$.

Next, we extend the definition of sp to handle a set of parallel processes.

<u>Definition</u>: Suppose $I_1, .., I_n$ are given predicates characterising the initial states (<u>not</u> including the communication sequences) of the processes $P_1, ..., P_n$ of CSP program $P_1 // ... // P_n$, then

$$sp(I_1 \wedge \cdots \wedge I_n, P_1 // .. // P_n) = \{(\tau_1', .., \tau_n') \mid \exists \tau_1, .., \tau_n . [[\forall i \leq n . (s_i \in I_i \wedge h_i = \varepsilon)] \wedge$$
$$[\forall i \leq n . \tau_i' \in sp(\tau_i, P_i)] \wedge [Reduce(h_1', .., h_n') = (\varepsilon, .., \varepsilon)]]\}$$

where s_i is the state component of τ_i, h_i is the sequence component of τ_i, and h_i' the sequence component of τ_i' .

The following two theorems are the key results of this section.

<u>Theorem</u>: If S is a CSP program or a construct appearing in one of the processes of such a program, and $\tau' \in sp(p,S)$, p being a given predicate then there exists a $\tau \in p$ such that $(\tau, \tau') \in S$ (i.e. (τ, τ') is a member of the relation corresponding to S.

The proof is by a straightforward induction on the structure of S. First we consider the case when S is a construct appearing in a process. Then we have the following possibilities:

1) $S \equiv$ <u>skip</u>. Suppose $\tau_i' \in sp(p,S)$. Then, since $sp(p,$<u>skip</u>$)=p$, $\tau_i' \in p$. Also by definition (of SEM2), <u>skip</u>$=\{(\sigma_i, \sigma_i)\}$. Hence, $(\tau_i', \tau_i') \in$ <u>skip</u>.

2) $S \equiv x := e$. Suppose $\tau_i' \in sp(p,S)$. Then there exists τ_i such that $[\tau_i \in p] \wedge [\tau_i' = \tau_i[x \leftarrow e]]$; and $[x := e] = \{(\sigma_i, \sigma_i') \mid \sigma_i' = \sigma_i[x \leftarrow e]\}$, thus proving the required result.

3) $S \equiv P_j?x$. Then, $\tau_i' \in sp(p,S)$ implies there exists τ_i and m such that $[\tau_i \in p] \wedge [\tau_i' = \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)]]$; hence $(\tau_i,\tau_i') \in P_j?x$, sinc

$$P_j?x = \{(\tau_i,\tau_i') \mid \exists m.\ \tau_i' = \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)]\}.$$

4) $S \equiv [\square(\ell=1,..,m) b_\ell \rightarrow T^\ell]$. Then $\tau_i' \in sp(p,S)$ implies that there exists τ_i and r such that $[b_r(\tau_i)] \wedge [\tau_i \in p] \wedge [\tau_i' \in sp(\tau_i,T^r)]$; hence $(\tau_i,\tau_i') \in S$, since $S = \{(\tau_i,\tau_i') \mid \exists r.[b_r(\tau_i) \wedge (\tau_i,\tau_i') \in T^r]\}$.

5) $S \equiv T^1;T^2$. Then, $\tau_i' \in sp(p,S) \Rightarrow \exists \tau_i.[\tau_i \in p \wedge \exists \tau_i''[\tau_i'' \in sp(\tau_i,T^1) \wedge$

$$\tau_i' \in sp(\tau_i'',T^2)]].$$

Hence $(\tau_i,\tau_i') \in S$, since $S = \{\tau_i' \mid \exists \tau_i,\tau_i''.[(\tau_i,\tau_i'') \in T^1 \wedge (\tau_i'',\tau_i') \in T^2]\}$.

The remaining cases are equally simple and are left to the reader. Finally, consider the case $S \equiv P_1//..//P_n$. We shall assume that p is of the form $\bigwedge_{i=1}^{n} I_i$, where I_i characterises the initial state (not includin the communication sequence) of P_i.

Then, $\tau' = (\tau_1',..,\tau_n') \in sp(p,S)$ implies that there exists $\tau = (\tau_1,..,\tau_n)$ such that $[\forall i \leq n.[s_i \in I_i \wedge h_i = \varepsilon \wedge \tau_i' \in sp(\tau_i,P_i)]] \wedge [\text{Reduce}(h_1',..,h_n') =$

$$(\varepsilon,..,\varepsilon)].$$

And $P_1//..//P_n = \{((\sigma_1,..,\sigma_n'), (\sigma_1',..,\sigma_n')) \mid [\forall i \leq n.[h_i = \varepsilon \wedge (\sigma_i,\sigma_i') \in P_i]] \wedge$

$$[\text{Reduce}(h_1',..,h_n') = (\varepsilon,..,\varepsilon)]\}.$$

Hence $((\tau_1,..,\tau_n), (\tau_1',...,\tau_n')) \in P_1//..//P_n$, thus proving the theorem.

Corollary. If $\models \{p\}S\{q\}$ is true (in SEM2) for given predicates p,q and CSP program (or CSP construct appearing in a process) S, then $sp(p,S) \Rightarrow q$.

The corollary follows directly from the above theorem, and the intended meaning of "$\models \{p\}S\{q\}$ is true".

The name sp(for strongest post-condition) was, of course, chosen in anticipation of the above corollary. The next theorem proves the completeness of SYS2 (with respect to SEM2, and hence, by the result of section 4.2, with respect to SEM1). By the corollary, what we need to show, in order to establish completeness, is the following: $\vdash \{p\}S\{sp(p,S)\}$. Note also that since, by the results of section 4.1, any proof in SYS2 can be converted into an equivalent proof in SYS1, we are in fact proving completeness of SYS2 as well as of SYS1 .

Theorem: for any given predicate p, and CSP construct S, $\{p\}S\{sp(p,S)\}$ is provable in SYS2.

Proof: The proof is again by induction the structure of S.

1) $S \equiv \text{skip}$: $sp(p,S)=p$; hence, by axiom A1 of section 4.1, $\vdash \{p\}S\{sp(p,S)\}$

2) $S \equiv x:=e$: $sp(p,S)=\{\tau_i' \mid \exists \tau_i.[\tau_i \in p \wedge \tau_i' = \tau_i[x \leftarrow e]]\}$. In order to show $\vdash \{p\}S\{sp(p,S)\}$, we need to show $p \Rightarrow \left[sp(p,S)\right]_e^x$. is do this, we must first define, given the predicate of, the set corresponding to q_e^x:

$$q_e^x = \{\sigma \mid \sigma[x \leftarrow e] \in q\}$$

Then $\left[sp(p,x:=e)\right]_e^x = \{\sigma_i \mid \sigma_i[x \leftarrow e] \in sp(p,x:=e)\}$.

Suppose $\tau_i \in p$; suppose also that $\tau_i' = \tau_i[x \leftarrow e]$; (Note: we assume that τ_i' is a well-defined state; i.e. the expression e has a proper value in state τ_i). Then by definition of $sp(p,x:=e)$, $\tau_i' \in sp(p,x:=e)$. Hence, by definition of $\left[sp(p,x:=e)\right]_e^x$, $\tau_i \in \left[sp(p,x:=e)\right]_e^x$. Thus we have $(\tau_i \in p) \Rightarrow \left(\tau_i \in \left[sp(p,x:=e)\right]_e^x\right)$ i.e. $p \Rightarrow \left[sp(p,x:=e)\right]_e^x$. Hence, $\vdash \{p\}\ x:=e\ \{sp(p,x:=e)\}$ by axiom A2.

3) $S \equiv P_j?x$: $sp(p,S) = \{\tau_i' \mid \exists \tau_i, m.[\tau_i \in p \wedge \tau_i' = \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)]]\}$

$$\left[sp(p,S)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i} = \{\tau_i \mid \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)] \in sp(p,P_j?x)\}$$

We need to show, that $p \Rightarrow \forall m.\left[sp(p,P_j?x)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i}$.

Suppose $\tau_i \in p$. We must show that $\forall m.\tau_i \in \left[sp(p,P_j?x)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i}$

Let $\tau_i' = \tau_i[x \leftarrow m, h_i \leftarrow h_i \vdash (j,i,m)]$. Then, by definition of $sp(p,P_j?x)$, $\tau_i' \in sp(p,P_j?x)$. Hence, by definition of $\left[sp(p,P_j?x)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i}$, $\tau_i \in \left[sp(p,P_j?x)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i}$; and this argument is independent of the value of m. Hence, $\forall m.\left[\tau_i \in \left[sp(p,P_j?x)\right]_{m,h_i\ (j,i,m)}^{x,h_i}\right]$. Hence, $p \Rightarrow \forall m.\left[sp(p,P_j?x)\right]_{m,h_i \vdash (j,i,m)}^{x,h_i}$. Therefore, by axioms A3, we have $\vdash \{p\}P_j?x\{sp(p,P_j?x)\}$.

4) $S \equiv P_j!y$: this case is similar to (3) and we omit the details.

5) $S \equiv [\square(\ell=1,..,m)b_\ell \rightarrow T^\ell]$. The inference rule of SYS2 for this case is,

$$\frac{\{p \wedge b_r\}\ T^r\ \{q\},\quad r = 1,..,m}{\{p\}[\square(\ell=1,..,m)b_\ell \rightarrow T^\ell]\{q\}}$$

Also $sp(p,S)=\{\tau_i' \mid \exists\tau_i,r.[\tau_i \in p \wedge b_r(\tau_i) \wedge \tau_i' \in sp(\tau_i,T^r)]\}$

$$= sp(p \wedge b_1,T^1) \cup \cdots \cup sp(p \wedge b_m,T^m)$$
$$= \bigvee_{\ell=1}^{m} sp(p \wedge b_\ell,T^\ell).$$

By the inductive hypothesis we have, for $\ell=1,..,m$, $\vdash\{p \wedge b_\ell\}\,T^\ell\{\,sp(p\wedge b_\ell,T^\ell$
and, hence, by the rule of consequence $\vdash\{p\wedge b_\ell\}T^\ell\{\bigvee_{r=1}^{m} sp(p\wedge b_r,T^r)\}, \ell=1,..$
Hence, by the rule of inference above, we have,

$$\vdash\{p\}[\square(\ell=1,..,m)b_\ell \to T^\ell]\{\bigvee_{\ell=1}^{m} sp(p\ b_\ell,T^\ell)\}\ \text{i.e.} \vdash\{p\}S\{sp(p,S)\}.$$

6) $S \equiv *[\square(\ell=1,..,m)b_\ell \to T^\ell]$. The rule of inference is,

$$\frac{\{q \wedge b_r\}\ T^r\ \{q\}\ ,\quad r=1,..,m}{\{q\}*[\square(\ell=1,..,m)b_\ell \to T^\ell]\{q \wedge \neg b\}}$$

where $\neg b \equiv \bigwedge_{\ell=1}^{m} \neg b_\ell$. q is usually called the "loop invariant".
On account of the rule of consequence, we may rewrite the above rule as

$$\frac{\begin{array}{l} p \Rightarrow q \\ q_r \Rightarrow q,\ r=1,..,m \\ \{q \wedge b_r\}T^r\ \{q_r\},\ r=1,..,m \end{array}}{\{p\}*[\square(\ell=1,..,m)b_\ell \to T^\ell]\{q \wedge \neg b\}}$$

Now $sp(p,S) = \{\tau_i' \mid \exists\tau_i.[\tau_i \in p \wedge \tau_i' \in sp(\tau_i,**[\square(\ell=1,..,m)b_\ell \to T^\ell]) \wedge \neg b(\tau_i')$
$$= \{\tau_i' \mid \tau_i' \in sp(p,**[\square(\ell=1,..,m)b_\ell \to T^\ell]) \wedge \neg b(\tau_i')\}.$$
Define $q = sp(p,**[\square(\ell=1,..,m)b_\ell \to T^\ell])$.
Suppose $\tau_i \in p$. Then, it is easy to see, from the definition of $sp(p,**[\square(\ell=1,..,m)b_\ell \to T^\ell])$, that $\tau_i \in q$. Thus $p \Rightarrow q$. Next, define,
$q_r = sp(q \wedge b_r,T^r)$. Hence, by the inductive hypothesis, $\vdash\{q \wedge b_r\}T^r\{q_r\}$.
Moreover, it is easy to see from the definition of q, that
$[\tau_i \in q_r \Rightarrow \tau_i \in q]$ i.e. $[q_r \Rightarrow q]$. Finally, from the definition of
$sp(p,*[\square(\ell=1,..,m)b_\ell \to T^\ell]) \equiv q \wedge \neg b$. Hence, we have $\vdash\{p\}S\{sp(p,S)\}$.

7) $S \equiv T^1;T^2$. Then, $sp(p,S)=\{\tau_i' \mid \exists\tau_i,\tau_i''.[\tau_i \in p \wedge \tau_i'' \in sp(\tau_i,T^1) \wedge \tau_i' \in sp(\tau_i'',T^2)]\}$
$$= sp(sp(p,T^1),T^2).$$
Then, by the rule for sequential composition, we have $\vdash\{p\}T^1;T^2\{sp(p,T^1;T^2)\}$

The cases for I/O guarded selection and repetition are handled as in (5) and (6) respectively, and we omit the details.

8) Finally, consider the case $S \equiv P_1 // .. // P_n$, $p \equiv \bigwedge_{i=1}^{n} I_i$.

Then $sp(p,S) = \{\tau_1',..,\tau_n') \mid \exists \tau_1,..,\tau_n . [\bigwedge_{i=1}^{n} [s_i \in I_i \wedge (h_i=\varepsilon) \wedge \tau_i' \in sp(\tau_i,P_i)]] \wedge$

$[\text{Reduce}(h_1',..,h_n') = (\varepsilon,...,\varepsilon)]\}$

$= \{(\tau_1',..,\tau_n') \mid \bigwedge_{i=1}^{n} [\tau_i' \in sp((I_i \wedge h_i=\varepsilon),P_i)] \wedge [\text{Reduce}(h_1',..,h_n')=(\varepsilon,..,\varepsilon)]\}$.

By the inductive hypothesis, we may assume,

$$\vdash \{I_i \wedge (h_i = \varepsilon)\} P_i \{sp((I_i \wedge h_i = \varepsilon), P_i\}$$

Hence, by the rule for parallel composition, we have

$$\vdash \{\bigwedge_{i=1}^{n} I_i\} P_1 // .. // P_n \ \bigwedge_{i=1}^{n} [sp((I_i \wedge h_i = \varepsilon), P_i] \wedge (\text{Reduce}(h_1,..,h_n) = (\varepsilon,..,\varepsilon))\}$$

i.e. $\vdash \{p\}\ P_1 // .. // P_n \{sp(p, P_1 // .. // P_n)\}$.

That completes the proof of the theorem.

Before concluding, some remarks are in order: the reader familiar with Apt et al[2] would not have failed to notice how closely our proof of completeness resembles their proof of completeness (of a simpl sequential language); our proof was, in fact, inspired by the proof in Apt et al[2]. Note also that, as in Apt et al[2],we may restrict our assertions to be recursively enumerable; the system would still be complete, since the intermadiate assertions are all r.e., if the given pre and post conditions are r.e. Also. the system is complete if we restrict our assertions to be first order, since all intermediate assertions can be converted to first order predicates; the only difficult case is the loop invariants and they may be handled by using an appropriate godelization of tuples of states such as $<\tau_i^1,..,\tau_i^r>$. It should be noted that if we restrict our assertions to be recursive then, as follows from the arguments in Apt et al[2], our system is <u>not</u> complete, since no auxiliary variables, which can serve as loop counters, are allowed in our system.

Finally we would like to consider the following question: why have we introduced SYS1, when SYS2 is simpler, complete, and can be easily proved to be consistent (independently of the consistency of SYS1)? The answer is that, in our experience with actual examples, SYS1 has, occasionally, proved to be easier to use than SYS2. Surprisingly, however, in the case of programs where each of the processes receives, as well as sends numbers to the same set of processes, the proofs in the two systems are quite similar to each other. It is only in the case of programs where each (or,at least,most) of the processes receive numbers from one set of processes, and send out numbers to another set of processes, that proofs in SYS1 are usually simpler than in SYS2. The set partitioning and gcd programs of [7] are examples of the former kind, and the matrix multiplication and prime number generation programs of [7] are examples of the latter kind, and the reade may wish to look at the proofs given in [7] of these programs to see th validity of our remarks.

Acknowledgements

The author is highly obliged to Prof.O.-J.Dahl for a number of discussions and suggestions; in particular, the axioms A1, and A2, of SYS2 were suggested by Prof. Dahl. A number of other members of the Informatics group at the University of Oslo also participated in some of these discussions. The author sincerely thanks Mrs.Madsen for her usual efficient typing of the manuscript. The research reported i this paper was performed while the author was supported by a post-doctorate fellowship of the Royal Norwegian Council for Scientific and Industrial Research (NTNF).

References

1. Apt,K.R., Formal justification of a proof System for Communicating Sequential Processes, draft,Eramus University, Rotteram, The Netherlands, 1981.
2. Apt.K.R.,Bergstra,J.A.,and Meertens,L.G.K.T., Recursive assertions are not enough - or are they ?, Theoratical Computer SC.,8(1979),73-7
3. Apt.K.R.,Francey,N.,and de Roever,W.P., A proof system for communicating sequential processes, ACM TOPLAS,2(1980),359-385.
4. Hoare,C.A.R., Communicating Sequential Processes, CACM,21(1978),666-677
5. Owicki,S., Axiomatic proof techniques for parallel programs, Computer Science Dept. Cornell University,Ph.D.thesis(1975).
6. Soundararajan,N., Axiomatic Semantics of communicating Sequential processes, research report,Institute for Informatics, University of Oslo (1981).
7. Soundararajan,N., Proofs of correctness of CSP programs, research report, Institute for Informatics, University of Oslo (1981