Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

F.S. de Boer

A proof rule for process-creation

Computer Science/Department of Software Technology Report CS-R8710 February

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

A Proof Rule for Process-Creation

Frank S. de Boer
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Abstract: A Hoare-style proof system for partial correctness is defined for a language which embodies the kind of parallelism which stems from process-creation.
We make use of the following proof theoretic concepts: Cooperation test, Global invariant, Bracketed section and Auxiliary variables.
These concepts have previously been applied to CSP ([5], [13]), DP ([9]) and to a subset of ADA containing the ADA-rendezvous ([8]).
We shall study the proof theory of a language with a generalisation of the synchronous message-passing mechanism of CSP. The syntactic construct of process creation and its semantic definition is taken from the language POOL ([2]).
One of the main characteristics of proof-theoretical interest of this language is that the communication partner referred to by an output statement (or input statement) is not syntactically identifiable, in contrast with the previously mentioned languages.
Basically our proof system is built upon the way the semantic mechanism of process identification is brought to the syntactic level.
We have proven the proof system to be sound and relative complete.

1980 Mathematics Subject Classification: 70A05.
1986 CR Categories: F.3.1.
Key Words & Phrases: proof theory, pre- and post-conditions, concurrency, dynamic process creation, cooperation test, bracketed section, global invariant.

Report CS-R8710
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. Introduction.

In this paper we show how the concepts of Cooperation test, Bracketed section, Global invariant and Auxiliary variables can be used to formulate a proof rule for Process-creation.
The application of these concepts to CSP we assume known ([5]).
We shall study the proof-theory of a language with process-creation as it occurs in POOL ([2]), an acronym for Parallel Object-Oriented Language.
Processes making up the concurrent systems formulated in this language can interact only by synchronous message passing.
We shall call this language just **P**.
The rest of this introduction will be used to give some idea of the proof theoretical difficulties involved in Process-creation.
In CSP concurrent systems are defined as follows:

$$<P_1 \leftarrow S_1, \ldots, P_n \leftarrow S_n : P_1 \| \cdots \| P_n>,$$

where $S_1, \ldots, S_n$ are sequential programs containing IO-statements (input-output statements).
The execution of the IO-statement $P_i!t$ by process P_j is synchronised with the execution of $P_j?x$ by process P_i and results in assigning the value of expression t to the variable x.
Execution of $P_1 \| \cdots \| P_n$ consists of an interleaving of executions of $S_1, \ldots, S_n$.
Given specifications $\{\phi_i\}S_i\{\psi_i\}$, $1 \leqslant i \leqslant n$, we can specify the input-output behaviour of the total system as follows: as precondition we take the conjunction of the preconditions ϕ_i , as postcondition the conjunction of ψ_i.
In languages which embody Process-creation things are a bit more complicated: we can not view a concurrent system formulated in such a language as consisting of a fixed number of certain processes.
To be more specific consider the following program in **P**:

$U \equiv <C_1 \leftarrow S_1, C_2 \leftarrow S_2, C_3 \leftarrow S_3>$
where
$S_1 \equiv$ if $true \rightarrow x := new(C_2) \,|\, true \rightarrow x := new(C_3)$ *fi*,
$S_2 \equiv R$,
$S_3 \equiv R'$,
R, R' being ordinary sequential programs.

Evaluation of an expression *new(C)* consists of creating a new process and results in a reference to that process.
This new process is going to execute the program associated with the identifier C.
So after an assignment $x := new(C)$ x refers to the newly activated C-process.
Execution of the above system starts with a process, the socalled root-process, which executes S_1.
This process ends with having activated a C_2-process or a C_3-process.
So given postconditions ψ, ψ' of R resp. R′ after execution of U $\psi \vee \psi'$ will hold and not $\psi \wedge \psi'$.
It will be clear that neither as precondition of U can be taken the conjunction of the preconditions for R resp. R'.
The problem is how to define a postcondition resp. precondition for such a concurrent system in general.
Another difficulty consists in the following: during execution of concurrent systems formulated in languages which embody Process-creation there can be active several processes executing the same program, but the sets of memory locations each process associates with the set of variables occurring in that program are disjoint.
Consider the following **P** -program:

The research was partially sponsored by Esprit project 415: Parallel Architectures and Languages for AIP.

$U \equiv < C_1 \leftarrow S_1, C_2 \leftarrow S_2 >$,
where
$S_1 \equiv x := new(C_2); x!0; x := new(C_2); x!1$,
$S_2 \equiv ?y$.

The statements $x!0$, $x!1$ and $?y$ are IO-statements and can be viewed as generalisations of CSP IO-statements.
In this example execution of the output statement $x!0$ resp. $x!1$ is synchronised with the corresponding input statement $?y$ and results in assigning the value 0 resp. 1 to the variable y of the process referred to by x.
Thus after the execution of the above system the value of the variable y of the first activation of S_2 is 0 and that of the second 1.
To be able to prove this we shall define a mechanism of process identification in such a way that its expression at the syntactic level makes a definition of a proof-system possible.
Let $U \equiv < C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n >$ be a program of **P**.
At the semantic level processes will be identified by pairs of natural numbers $<k,m>$, $1 \leqslant k \leqslant n$, such that process $<k,m>$ will be the m^{th} activation of program S_k.
Semantically $C_1, \ldots, C_n$ will be treated as counters, each one counting the number of activations of the corresponding program.
Given this semantic description we can formulate a creation axiom as follows: $\{p[C_k+1 / C_k, <k, C_k+1> / x]\} x := new(C_k)\ \{p\}$ (where $[\cdots]$ denotes substitution). Note that we assume $C_1, \ldots, C_n$ to be variables of the assertion language (the language used to describe states) and that this assertion language contains terms which denote pairs of natural numbers.
We shall treat a variable occurring in, say, S_i, semantically as a function with domain the set of natural numbers and with range the set of integers, booleans etc..
The second component of a process name we use as a pointer to its workspace, that is, the process identified by $<k,m>$ will operate on $f(m)$, f the function denoted by x, x a variable of S_k.
Proof-theoretically this interpretation of program variables lends itself to the treatment of these variables as one-dimensional arrays ([10]).
So we can refer to the value of the variable y w.r.t. the first activation of S_2 (of the example) by the term $y[1]$.
To prove that, w.r.t. the first activation of S_2, the value of y is 0, we must prove that $\{true\}?y[1]\{y[1]=0\}$ holds.
But it will be clear that it is unsatisfactory and, in case of an infinity of activations of some program, impossible, to have to construct a correctness proof for every activation of some program.
We must be able to construct a canonical proof for each program.
This is made possible by abstracting from the specific value of the second component of a process name, the pointer to the workspace of the process denoted by that name.
We introduce a new variable i, not occurring as a program variable, and axioms and proof rules to construct proofs for correctness- assertions like $\{p_k\} S_k^{(i)} \{q_k\}$, $1 \leqslant k \leqslant n$.
$S_k^{(i)}$ denotes $S_k[x_1[i] / x_1, \ldots, x_m[i] / x_m]$, where $\{x_1, \ldots, x_m\}$ is the set of variables occurring in S_k.
Thus the execution of $S_k^{(i)}$ in a particular state consists of the execution of S_k by the process $<k,m>$, m the value of the variable i in that state.
So for the example just given we would have to prove that $\{true\}?y[i]\{i=1 \rightarrow y[i]=0 \wedge i=2 \rightarrow y[i]=1\}$.
In the following section we give a definition of the language **P**.
In section three we give a operational semantics of this language, and a definition of the assertion language.
In section four we show how the proof-theoretic concepts: cooperation test, global invariant, bracketed section and auxiliary variables fit in this framework and present a proof-system for partial

correctness assertions for **P** -programs.
The sections five and six discuss the soundness and completeness of the proof system.
In the first appendix we give a prove of the merging lemma, this lemma is used in the proof of the completeness theorem.
This lemma basically states that given two computations of a **P** program such that a given process is active in both and the sequence of its interactions with the environment in both computations are the same we can replace the local computation of this particular process in the one computation of the main program by the one of this process in the other main computation.
In the second appendix we prove a rather technical lemma, this lemma states that computation sequences can be rearranged in a way which is imperceptible to the processes involved in this sequence.
In the fourth appendix we show how to express in the assertion language a specific way of coding the computations of a **P** program, this expressibility result is used in the proof of the completeness theorem.
In the last appendix we discuss an example of a correctness proof.

2. The language P

Given the following sets of syntactic objects:
$Cname = \{C, \cdots\}$, the set of class names,
$Pvar = \bigcup_{C \in Cname} Pvar^C = \{x, \cdots\}$, the set of program-variables,
such that for any two distinct class names the associated sets of program variables are disjoint,
$Icon = \{n : n \in \mathbf{N}\}$, the set of integer-constants,
$Lab = \{l, \cdots\}$, the set of labels,
the syntax of the language P is given by the following grammar:

$$S ::= nil \mid l:x:=e \mid l:x?y \mid l:?y \mid l:x!t \mid l:!t \mid S_1 ; S_2 \mid l: \overset{n}{\underset{i=1}{\Box}} b_i \rightarrow S_i \mid l:* \overset{n}{\underset{i=1}{\Box}} b_i \rightarrow S_i.$$

$$e ::= t \mid self \mid new(C)$$

$$t ::= \underline{n} \mid x \mid t_1 + t_2 \mid t_1 \times t_2 \mid \cdots$$

$$b ::= t_1 = t_2 \mid t_1 < t_2 \mid \cdots \mid \neg b \mid b_1 \wedge b_2 \mid \cdots$$

$$u ::= <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$$
where, for $1 \leqslant k \leqslant n$, C_k is a class name and $\mathrm{var}(S_k) \subseteq Pvar^{C_k}$.

A syntactic construct u is called a unit. We introduced a set of labels to be able to distinguish between different occurrences of a particular subprogram of a unit. Therefore we demand that all the labels occurring in a unit are different. We shall make use of this labelling in the proofs of the soundness and completeness theorems. Execution of a unit $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$ consists of the execution of S_1. Evaluation of an expression $new(C)$ results in a reference to a new process which associates new memory locations to the variables occurring in the program corresponding with C and which starts executing that program. Evaluation of the expression *self* by a process results in a reference to that process. The communication mechanisms are generalisations of the way processes in CSP communicate. Execution of $x!t$ by a process P is synchronised with the execution of $?y$ or $z?y$ by a process P' if x refers to P' and, in case $z?y$ is being executed, z refers to P. In the same way execution of $x?y$ by a certain process is synchronised with the execution of $!t$ or $z!t$ by another process. So two processes can communicate when at least one refers to the other. Note that we do not allow statements like $x!new(C)$. "$\overset{n}{\underset{i=1}{\Box}} b_i \rightarrow S_i$" and "$* \overset{n}{\underset{i=1}{\Box}} b_i \rightarrow S_i$" we use as notations for the generalised IF resp. DO statement. *Nil* denotes the empty process.

3. Semantics

We shall define an interleaving operational semantics for **P**, making use of the way it is done for POOL ([3]). For general information about this technique for semantic description we refer to [11].
First we define the class of states. A state consists of two components: the first component assigns values to the program variables; the second component assigns (positive) integer values to the class names occurring in *u*. Semantically we will treat these class names as global counters, each counting the number of active instances of that particular class.
The value ω, in the definition below, stands for undefined.

DEFINITION 3.1

We define **states**, elements of which are denoted by σ,... , as follows:
$\sigma \in$ **states** $\Leftrightarrow$
a. $\sigma=(\sigma_0,\sigma_1)$
b. $\sigma_0 \in Pvar \times \mathbf{N}^+ \rightarrow \mathbf{D} \cup \{\omega\}$
c. $\sigma_2 \in Cname \rightarrow \mathbf{N}$
d. when $x \in Pvar^C$, $C \in Cname$, then for $k > \sigma_2(C)$, $\sigma_0(<x,k>)=\omega$.
where
$\mathbf{D}=\mathbf{N} \cup \mathbf{N} \times \mathbf{N}$.
D is the domain of all possible values, elements of which are denoted by *d*,....
N denotes the set of natural numbers.
$\mathbf{N}^+$ denotes the set of natural numbers greater than zero.

REMARK

In the sequel we shall not distinguish the two components of a state, that is we shall just write $\sigma(C)$, $\sigma(<x,n>)$, C a class name , x a program variable. Subscription of states will be used to be able to introduce new states when needed.

Before defining an operational semantics for **P** we define the assertion language L used to describe $\sigma \in$ **states**.

DEFINITION 3.2

Given *lvar* the set of logical variables the syntax of the class of terms of L is described as follows (these terms we denote by f, f_1, $\cdots$):

$term ::= Dt \mid At$
$Dt ::= It \mid Pt$
$It ::= n,\ n \in \mathbf{N} \mid It_1 + It_2 \mid \cdots \mid (Pt)_1 \mid (Pt)_2 \mid At[It] \mid C \mid i,\ i \in Lvar$
$Pt ::= <It_1, It_2> \mid At[It] \mid nil$
$At ::= x,\ x \in Pvar \mid (At;[It]:Dt)$.

We distinguish two kinds of terms: *Dt,* the Data terms, and *At,* the Array terms.
We view every variable occurring in a unit as an one-dimensional array with lowerbound 1, so the basic array terms are program variables.
The array term $(At;[It]:Dt)$ ([10], chapter 5.) denotes the array resulting from assigning the value of the data term Dt to the n^{th} element of the array denoted by At, n the value of the integer term It.
There are two kinds of data terms: *It,* the integer terms, *Pt,* the process terms.
Process terms denote pairs of natural numbers.
The integer term $(Pt)_1$ resp. $(Pt)_2$ denotes the first resp. the second component of the pair of numbers denoted by Pt.
W.r.t. the assertion language we treat class names as variables.

DEFINITION 3.3

Let $\nu \in Env = lvar \to \mathbf{N}$. We define for a term $f \in L$, $\sigma \in$ **states**, $V(f)(\nu)(\sigma)$, the value of term f in state σ, w.r.t. the logical environment ν.

1.

$$V(\underline{n})(\nu)(\sigma) = n, \; n \in \mathbf{N}$$

2.

$$V(C_k)(\nu)(\sigma) = \sigma(C_k)$$

3.

$$V(nil)(\nu)(\sigma) = \omega$$

4.

$$V(i)(\nu)(\sigma) = \nu(i), \; i \in lvar$$

5.

$$V(x)(\nu)(\sigma) = g(x), \; x \in Pvar,$$

where $g \in Pvar \to (\mathbf{N}^+ \to \mathbf{D} \cup \{\omega\})$, such that $g(x)(n) = \sigma(<x,n>)$.

6.

$$V(It_1 + It_2)(\nu)(\sigma) = V(It_1)(\nu)(\sigma) + V(It_2)(\nu)(\sigma) \text{ if } V(It_1)(\nu)(\sigma),\ V(It_2)(\nu)(\sigma) \in \mathbf{N}$$
$$= \omega \text{ otherwise}$$

7.

$$V(<It_1, It_2>)(\nu)(\sigma) = <V(It_1)(\nu)(\sigma), V(It_2)(\nu)(\sigma)> \text{ if } V(It_1)(\nu)(\sigma), V(It_2)(\nu)(\sigma) \in \mathbf{N}$$
$$= \omega \text{ otherwise}$$

8.

$$V((Pt)_1)(\nu)(\sigma) = n \text{ if } V(Pt)(\nu)(\sigma) = <n,m> \in \mathbf{N} \times \mathbf{N}$$
$$= \omega \text{ otherwise}$$

9.

$$V((Pt)_2)(\nu)(\sigma) = m \text{ if } V(Pt)(\nu)(\sigma) = <n,m> \in \mathbf{N} \times \mathbf{N}$$
$$= \omega \text{ otherwise}$$

10.

$$V(At[It])(\nu)(\sigma) = V(At)(\nu)(\sigma)(V(It)(\nu)(\sigma)) \text{ if } V(It)(\nu)(\sigma) \in \mathbf{N}^+$$
$$= \omega \text{ otherwise}$$

11.

$$V((At;[It]{:}Dt))(\nu)(\sigma) = V(At)(\nu)(\sigma) \text{ if } V(It)(\nu)(\sigma) \notin \mathbf{N}^+$$
$$= V(At)(\nu)(\sigma)\{V(Dt)(\nu)(\sigma) \,/\, V(It)(\nu)(\sigma)\} \text{ otherwise.}$$

In clause 3 of the definition just given we used $g\{\cdots / \cdots\}$, g a function, as the variant notation.

First-order formulae are defined in the following way:
$\psi ::= It_1 = It_2 \mid Pt_1 = Pt_2 \mid At_1 = At_2 \mid It_1 \leqslant It_2 \cdots \mid \exists i\psi, \; i \in lvar \mid \cdots$
The truth-definition differs slightly from the usual one w.r.t, for example, the following case:

$\models(It_1 \leqslant It_2)(\nu)(\sigma)$ if and only if $V(It_1)(\nu)(\sigma)$, $V(It_2)(\nu)(\sigma) \in \mathbf{N}$ and $V(It_1)(\nu)(\sigma) \leqslant V(It_2)(\nu)(\sigma)$.

We are now able to give a deduction system for transitions like $<X,\sigma,u> \underset{k}{\overset{h}{\rightarrow}} <X',\sigma',u>$. $<X,\sigma,u>$ and $<X',\sigma',u>$ are called configurations. X, X' are sets of pairs $<\alpha,S>$, $\alpha \in \mathbf{N}\times\mathbf{N}$, S the piece of program still to be executed by the process denoted by α.
The variable k ranges over the set of natural numbers, it denotes the number of "steps" the derivation of the configuration to the right of the arrow from the one to the left takes.
The variable h ranges over the set of histories, a history being a sequence of pairs $<\alpha,\beta>$, and triples $<d,\alpha,\beta>$, α, $\beta \in \mathbf{N}\times\mathbf{N}$, the intended meaning of the first being "α creates β" and that of the second being " the value d is sent by α to β".
The last component of a configuration is a unit.
The information encoded by h and k is used in the proofs of the soundness and the relative completeness of the proof system.
Basic to the following definition of a deduction system for transitions is the coding of processes by pairs $<m,k>$, where m, k are natural numbers, such that the process coded by $<m,k>$ will be the k^{th} activated instance of class C_m and operates on $x[k]$, where $x \in var(S_m)$, the variable x treated as a one-dimensional array.
In the following definition when we write $X \cup \{ \cdots \}$ this implies $\{ \cdots \} \cap X = \varnothing$.

In the sequel $t^{(i)}$, $b^{(i)}$ for the term t and the boolean expression b, i a logical variable, denotes the result of substituting for each program variable x occurring in t resp. b the term $x[i]$.

DEFINITION 3.4

axioms:

[1] $<X \cup \{<\alpha,l:x:=new(C_k)>\},\sigma,u> \underset{1}{\overset{<\alpha,\beta>}{\rightarrow}} <X \cup \{<\alpha,nil>,<\beta,S_k>\},\sigma',u>$
where $\beta = <k, \sigma(C_k)+1>$ and
$\sigma' = \sigma\{nil / <y_1,\beta_2>, \ldots, nil / <y_n,\beta_2>, \beta / <x,\alpha_2>, \beta_2{}^* / C_k\}$
$\{y_1, \ldots, y_n\} = var(S_k)$

[2] $<X \cup \{<\alpha,l:x:=self>\},\sigma,u> \underset{1}{\overset{\epsilon}{\rightarrow}} <X \cup \{<\alpha,nil>\},\sigma',u>$
where $\sigma' = \sigma\{\alpha / <x,\alpha_2>\}$

[3] $<X \cup \{<\alpha,l:x:=t>\},\sigma,u> \underset{1}{\overset{\epsilon}{\rightarrow}} <X \cup \{<\alpha,nil>\},\sigma',u>$
where $\sigma' = \sigma\{d / <x,\alpha_2>\}$
$d = V(t^{(i)})(\nu\{\alpha_2 / i\})(\sigma)$
$\nu \in Lvar \rightarrow \mathbf{N}$ arbitrary

[4] $<X \cup \{<\alpha,nil;S>\},\sigma,u> \underset{1}{\overset{\epsilon}{\rightarrow}} <X \cup \{<\alpha,S>\},\sigma,u>$

[5] $X \cup \{<\alpha,l:x?y>,<\beta,l':z!t>\},\sigma,u> \underset{1}{\overset{<d,\beta,\alpha>}{\rightarrow}} <X \cup \{<\alpha,nil>,<\beta,nil>\},\sigma',u>$
if $\sigma(<x,\alpha_2>) = \beta$ and $\sigma(<z,\beta_2>) = \alpha$
where $\sigma' = \sigma\{d / <y,\alpha_2>\}$
$d = V(t^{(i)})(\nu\{\beta_2 / i\})(\sigma)$
$\nu \in lvar \rightarrow \mathbf{N}$ arbitrary

[6] $<X \cup \{<\alpha,l:x?y>,<\beta,l':!t>\},\sigma,u> \underset{1}{\overset{<d,\beta,\alpha>}{\rightarrow}} <X \cup \{<\alpha,nil>,<\beta,nil>\},\sigma',u>$
if $\sigma(<x,\alpha_2>) = \beta$

*:for $\beta \in \mathbf{N}\times\mathbf{N}$ β_1,β_2 denotes the first resp. the second component of β.

where $\sigma' = \sigma\{d / <y, \alpha_2>\}$
$d = V(t^{(i)})(\nu\{\beta_2 / i\})(\sigma)$
$\nu \in lvar \to \mathbf{N}$ arbitrary

[7] $<X \cup \{<\alpha, l:?y>, <\beta, l':x\,!t>\}, \sigma, u> \overset{<d,\beta,\alpha>}{\underset{1}{\to}} <X \cup \{<\alpha, nil>, <\beta, nil>\}, \sigma', u>$

if $\sigma(<x, \beta_2>) = \alpha$
where $\sigma' = \sigma\{d / <y, \alpha_2>\}$
$d = V(t^{(i)})(\nu\{\beta_2 / i\})(\sigma)$
$\nu \in lvar \to \mathbf{N}$ arbitrary

[8] $<X \cup \{<\alpha, l: \underset{k=1}{\overset{n}{\Box}} b_k \to S_k>\}, \sigma, u> \overset{\epsilon}{\underset{1}{\to}} <X \cup \{<\alpha, S_m>\}, \sigma, u>$

if $\models b_m{}^{(i)}(\nu\{\alpha_2 / i\})(\sigma)$
$\nu \in lvar \to \mathbf{N}$ arbitrary

[9] $<X \cup \{<\alpha, l:* \underset{k=1}{\overset{n}{\Box}} b_k \to S_k\}, \sigma, u> \overset{\epsilon}{\underset{1}{\to}} <X \cup \{<\alpha, S_m; l:* \underset{k=1}{\overset{n}{\Box}} b_k \to S_k>\}, \sigma, u>$

if $\models b_m{}^{(i)}(\nu\{\alpha_2 / i\})(\sigma)$,
$\nu \in lvar \to \mathbf{N}$ arbitrary

[10] $X \cup <\{<\alpha, l:* \underset{k=1}{\overset{n}{\Box}} b_k \to S_k>\}, \sigma, u> \overset{\epsilon}{\underset{1}{\to}} X \cup \{<\alpha, nil>\}, \sigma, u>$

if $\models \underset{k=1}{\overset{n}{\wedge}} \neg b_k{}^{(i)}\ (\nu\{\alpha_2 / i\})(\sigma)$
$\nu \in lvar \to \mathbf{N}$ arbitrary

[11] $<X, \sigma, u> \overset{\epsilon}{\underset{0}{\to}} <X, \sigma, u>$

rules

[1]

$$\frac{<X \cup \{<\alpha, S_1>\}, \sigma, u> \overset{h}{\underset{1}{\to}} <X' \cup \{<\alpha, S_2>\}, \sigma', u>}{<X \cup \{<\alpha, S_1; S>\}, \sigma, u> \overset{h}{\underset{1}{\to}} <X' \cup \{<\alpha, S_2; S>\}, \sigma', u>}$$

[2]

$$\frac{<X, \sigma, u> \overset{h_1}{\underset{k_1}{\to}} <Y, \sigma', u>,\ <Y, \sigma', u> \overset{h_2}{\underset{k_2}{\to}} <Z, \bar{\sigma}, u>}{<X, \sigma, u> \overset{h_1 \circ h_2}{\underset{k_1 + k_2}{\to}} <Z, \bar{\sigma}, u>}$$

Now we are able to state the meaning of a unit u.

DEFINITION 3.5
Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$ then:

$<\sigma, \sigma'> \in M(u) \Leftrightarrow$

$\exists X_0, X_1, h, k$:
a. $X_0 = \{<<1,1>, S_1>\}$
b. $\sigma(C_1) = 1, \sigma(\underset{h}{C_j}) = 0, 1 < j \leqslant n$
c. $<X_0, \sigma, u> \overset{k}{\underset{}{\to}} <X_1, \sigma', u>$
d. $\forall <\alpha, S> \in X_1\ S = nil$

$<X_0,\sigma,u>$ we call an initial configuration.

Having defined the meaning of a unit $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$ we want to define the meaning of R, $R;R'$, $R \| R'$, where R, R' are subprograms of, say, S_k resp. S_m, $1 \leqslant m,k \leqslant n$.
But given a σ, $\sigma \in$ **states**, there can be active several instances of classes C_k, C_m (in case $\sigma(C_m),\sigma(C_k)>0$) all of which are candidates to execute R resp. R'.
One way to specify the particular instance one wants to consider can be described as follows:
We will define, for example, $M(R^{(i)}) \in Env \rightarrow P(\textbf{states} \times \textbf{states})$, R a subprogram of S_k, i a logical variable, such that $<\sigma,\sigma'> \in M(R^{(i)})(\nu)$ if execution of R by the process $<k,\nu(i)>$ starting from σ results in σ'.

DEFINITION 3.6
Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, R a subprogram of S_k, $1 \leqslant k \leqslant n$,
$i \in Lvar$,

$<\sigma,\sigma'> \in M(R^{(i)})(\nu) \Leftrightarrow$

a. $\exists <X_l,\sigma_l,u>_{\substack{h_l \\ l=0,\ldots,m}}$ such that for every $0 \leqslant l < m$:
$<X_l,\sigma_l,u> \underset{1}{\rightarrow} <X_{l+1},\sigma_{l+1},u>$ for some history h_l
and $\sigma_0 = \sigma$, $\sigma_m = \sigma'$
b. $X_0 = \{<<k,\nu(i)>,R>\}$
c. $0 < \nu(i) \leqslant \sigma(C_k)$
d. $\forall 0 \leqslant l \leqslant m \ \forall <\beta,S> \in X_l \ (\beta \neq <k,\nu(i)> \rightarrow <\beta,S> \in X_{l+1})$
e. $<<k,\nu(i)>,nil> \in X_m$.

Clause a of the above definition states the existence of a sequence of configurations, a so called computation, such that every element, but the first, of this sequence is derivable from the previous one in one step.
Clause b expresses that we want to consider the behaviour of the process $<k,\nu(i)>$ executing the subprogram R.
The number of active instances of class C_k in a state σ is given by $\sigma(C_k)$, so clause c must be included.
Clause d states that every step of the computation, mentioned in clause a, is done by the process $<k,\nu(i)>$.
Clause e states that the process $<k,\nu(i)>$ has finished executing R.

DEFINITION 3.7
Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$,
R, R' subprograms of S_k, resp. S_m, $1 \leqslant k,m \leqslant n$,
i, $j \in Lvar$,

$<\sigma,\sigma'> \in M(R^{(i)};R'^{(j)})(\nu) \Leftrightarrow$
$\exists \bar{\sigma} \ (<\sigma,\bar{\sigma}> \in M(R^{(i)}) \wedge <\bar{\sigma},\sigma'> \in M(R'^{(j)}))$.

DEFINITION 3.8
Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$,
R, R' subprograms of S_k, S_m, $1 \leqslant k,m \leqslant n$, i, $j \in lvar$,

$<\sigma,\sigma'> \in M(R^{(i)} \| R'^{(j)})(\nu) \Leftrightarrow$

a. $\exists <X_l,\sigma_l,u>_{\substack{\xrightarrow{h_l} \\ 1}\ l=0,\ldots,p}$ such that for every $0 \leq l < p$:
$<X_l,\sigma_l,u> \xrightarrow[1]{h_l} <X_{l+1},\sigma_{l+1},u>$ for some history h_l
and $\sigma_0 = \sigma, \sigma_p = \sigma'$

b. $X_0 = \{<<k,\nu(i)>,R>,<<m,\nu(j)>,R'>\}$

c. $k = m \rightarrow \nu(i) \neq \nu(j),\ 0 < \nu(i) \leq \sigma(C_k),\ 0 < \nu(j) \leq \sigma(C_m)$

d. $\forall 0 \leq l < p \forall <\beta,S> \in X_l\ (\beta \neq <k,\nu(i)> \wedge \beta \neq <m,\nu(j)> \rightarrow <\beta,S> \in X_{l+1})$

e. $<<k,\nu(i)>,nil>,<<m,\nu(j)>,nil> \in X_p$.

Having defined $M(u)$, $u \equiv <C_1 \leftarrow S_1,..,C_n \leftarrow S_n>$, and $M(R^{(i)})$, $M(R_1^{(i)};R_2^{(j)})$, $M(R_1^{(i)} \| R_2^{(j)})$, where $i, j \in lvar$, R, R_1, R_2 subprograms of one of the programs S_k, $1 \leq k \leq n$, we can define the partial-correctness assertions: $\{p\}u\{q\}, \{p\}R^{(i)}\{q\} \cdots$, such that $\models \{p\}u\{q\}$ iff for every ν, $<\sigma,\sigma'> \in M(u)$ if $\models p(\nu)(\sigma)$ then $\models q(\nu)(\sigma')$,
and $\models \{p\}R^{(i)}\{q\}$ iff for every ν, $<\sigma,\sigma'> \in M(R^{(i)})(\nu)$ if $\models p(\nu)(\sigma)$ then $\models q(\nu)(\sigma)$,
the truth of the other partial correctness assertions is defined in a similar way.
W.r.t. the correctness-assertion $\{p\}u\{q\}$ we do not allow class names occurring in p and q other than those mentioned in u, and, furthermore, every program variable occurring in p or q must belong to one of the classes defined in u.

4. The Proof System

The following axioms and proof rules are modifications of the axioms and proof-rules of the Hoare-style proof-system for sequential programs ([4]). These modifications are introduced because we treat the program variables as one-dimensional arrays. These axioms and proof rules enable one to reason about the correctness of the components of a concurrent system, given assumptions about those parts which depend on the behaviour of the environment.

Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, $i \in Lvar$.

A1. $(u:\{p[(x;[i]:t^{(i)})/x]\}\ (l:x:=t)^{(i)}\{p\})$

A2. $(u:\{p[(x;[i]:<k,i>)/x]\}\ (l:x:=self)^{(i)}\ \{p\})$
where "$x:=self$" occurs in S_k.

R1.

$$\frac{(u:\{p \wedge b_k^{(i)}\}\ R_k^{(i)}\ \{q\}), 1 \leq k \leq m}{(u:\{p\}\ (\overset{m}{\underset{k=1}{\Box}} b_k \rightarrow R_k)^{(i)}\ \{q\})}$$

R2.

$$\frac{(u:\{p \wedge b_k^{(i)}\}\ R_k^{(i)}\{p\}), 1 \leq k \leq m}{(u:\{p\}(*\overset{m}{\underset{k=1}{\Box}} b_k \rightarrow R_k)^{(i)}\ \{p \wedge \overset{m}{\underset{k=1}{\wedge}} \neg b_k^{(i)}\}).}$$

R3.

$$\frac{(u:\{p\}\ R_1^{(i)}\ \{r\}), (u:\{r\}\ R_2^{(i)}\ \{q\})}{(u:\{p\}\ (R_1;R_2)^{(i)}\{q\})}$$

R4.

$$\frac{p \wedge i \leq C_k \rightarrow p_1\ ,\ (u:\{p_1\}R^{(i)}\ \{q_1\}), q_1 \wedge i \leq C_k \rightarrow q}{(u:\{p\}\ R^{(i)}\ \{q\}),\ R \text{ occurs in } S_k}$$

W.r.t. axiom 2 we translated the expression *self* by the term $<k,i>$ because the value of the variable i denotes in the context $(l:x:=self)^{(i)}$ the second component of the name of a process executing $l:x:=self$. Without this modification of the parameterized rule of consequence our proof-system would be incomplete: consider the following valid correctness-assertion, $(u:\{i>C_k\}(l:x:=1)^{(i)}\{x[i]=0\})$. It is valid because $\models i>C_k(\nu)(\sigma)$ means that there is no process $<k,\nu(i)>$ active in state σ. This assertion is deducible from the system consisting of the axioms A1,A2 the rules R1 , . . . , R3 and the parameterized consequence rule if and only if $i>C_k \rightarrow (x;[i]:1)[i]=0$ is valid, which is not the case.

In the following definitions we show how the proof theoretic concepts: Bracketed Section, Global Invariant, Cooperation Test and Auxiliary Variables, can be applied to the language **P**.

DEFINITION 4.1

A BS (Bracketed Section) is of the form $<S>$ where:
$S \equiv R_1;R';R_2$, R' an IO (a communication statement) or a new-statement and in R_1 , R_2 there are no occurrences of IO or new- statements and if $R' \equiv l:x:=new(C)$ then $x \notin change(R_2)$,where $x \in change(R)$ if and only if x occurs at the l.h.s. of an assignment or the r.h.s. of a "?" of an input-statement.

DEFINITION 4.2
The bracketed sections $<R_1>,<R_2>$ match if:
$R_1 \equiv R_3;R;R_4$, $R_2 \equiv R_5;R';R_6$.
1. $R \equiv l:x?z$, $R' \equiv l:y!t$ *or* $R' \equiv l:!t$
2. $R \equiv l:?z$, $R' \equiv l:y!t$.

Note that bracketed sections are syntactic constructs. Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, such that each IO and new-statement occurring in u appears in a bracketed section. To be able to deduce correctness-assertions $\{p\}S_k^{(i)}\{q\}$, $1 \leqslant k \leqslant n$, for some logical variable i, we introduce sets of assumptions A_k, $1 \leqslant k \leqslant n$, an assumption being a correctness-assertion about a bracketed section. We say that two assumptions about bracketed sections containing IO statements match if the corresponding IO statements do. Having established the deducibility of $\{p_k\}S_k^{(i)}\{q_k\}$, $1 \leqslant k \leqslant n$, using the sets of assumptions A_k (notation: $A_k \vdash \{p_k\}S_k^{(i)}\{q_k\}$) we have to show that these assumptions cooperate. The formal definition of this cooperation test is given below, the notion of global invariant, a first-order formula, occurring in this definition is introduced to restrict the cooperation test to the semantically matching assumptions. By semantically matching assumptions we mean a pair of assumptions containing IO-statements execution of which can be synchronised.
In the following definition $FV(p)$, p a formula in the assertion language, denotes the set of free variables, program variables and logical variables, occurring in p, class names are treated as variables too.

DEFINITION 4.3
let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, u bracketed. Given are the proofs $A_k \vdash \{p_k\}S_k^{(i)}\{q_k\}$, $1 \leqslant k \leqslant n$, i some logical variable. These proofs cooperate w.r.t the global invariant I if for some logical variable j distinct from i:

a. $FV(\mathrm{I}) \cap \{i,j\} = \varnothing$,
and there are no free occurrences in I of variables which can be changed outside a bracketed section

b. let $\{pre(R_1)\}<R_1>\{post(R_1)\} \in A_k$,
$\{pre(R_2)\}<R_2>\{post(R_2)\} \in A_m$, be two matching assumptions then :
$\vdash (u:\{\mathrm{I} \wedge pre(R_1) \wedge (pre(R_2)[j/i])\}R_1^{(i)} \| R_2^{(j)} \{\mathrm{I} \wedge post(R_1) \wedge (post(R_2)[j/i])\}$

c. let $\{pre(R)\}<R>\{post(R)\} \in A_m$, $R \equiv <R_1;x:=new(C_k);R_2>$ then:
$\vdash \{\mathrm{I} \wedge pre(R)\}R^{(i)}\{\mathrm{I} \wedge post(R) \wedge p_k[(x[i])_2 / i]\}$

d. For every $\{p\}R\{q\} \in A_k$, $1 \leqslant k \leqslant n$: $FV(p,q,p_k,q_k) \cap (var(S_l) \cup \{C_1, \ldots, C_n\}) = \varnothing$, for $1 \leqslant k \neq l \leqslant n$,
$j \notin FV(p,q)$, and every $x \in FV(p,q,p_k,q_k) \cap var(S_k)$ occurs only subscripted by a free occurrence of i in p,q,p_k,q_k.

Comment:
We do not allow i, j occurring free in the global invariant because this formula is introduced to express some global information. Clause d guarantees freedom from interference. We do not allow j occurring in the formulae mentioned in clause d because we have chosen the variable i to enable

one to relate the values of some program variables of, say S_k, to a particular instance of class C_k. The clauses b and c together establish the invariance of the formula I over the bracketed sections. Clause a implies invariance of I over the remaining parts. Clause c states among others that for every statement $x:=new(C_k)$ we have to establish that the precondition p_k is satisfied by the newly activated instance of class C_k. Because $x \notin change(R_2)$ we can access that instance.

Given the above definition we can now formulate a proof rule for process-creation.

R5.

$$\frac{\textit{there exist proofs } A_k \vdash (u:\{p_k\}S_k^{(i)}\{q_k\}),\ 1 \leqslant k \leqslant n,\ \textit{which cooperate w.r.t the global invariant } I}{\{I \wedge p_1[\underline{1}/i]\}u\{I \wedge \bigwedge_{k=1}^{n} \forall \underline{1} \leqslant i \leqslant C_k\ q_k\}}$$

To derive the correctness assertions mentioned in the clauses b and c of the cooperation test we introduce the following axioms and rules:

Let i, j be distinct logical variables.

A3. $R \equiv \cdots ?x, R' \equiv \cdots !t$:
$(u:\{p[(x;[i]:t^{(j)})/x]\}R^{(i)} \| R'^{(j)}\{p\})$

A4.

$$(u:\{p[C_k+1/C_k, t_1/y_1, \ldots, t_m/y_m, (x;[i]:<\underline{k}, C_k+1>)/x]\}(l:x:=new(C_k))^{(i)}\{p\})$$

where $\{y_1, \ldots, y_m\} = var(S_k)$, and $t_r = (y_r;[C_k+1]:nil)$, $1 \leqslant r \leqslant m$.

R6. Let $R \equiv l:x?y$ occur in S_k and $R' \equiv l':z!t$ in S_m.

$$\text{a. } k \neq m: \frac{(u:\{p \wedge x[i] = <m,j> \wedge z[j] = <k,i> \wedge i \leqslant C_k \wedge j \leqslant C_m\}R^{(i)} \| R'^{(j)}\{q\})}{(u:\{p\}R^{(i)} \| R'^{(j)}\{q\})}$$

$$\text{b. } k = m: \frac{(u:\{p \wedge x[i] = <m,j> \wedge z[j] = <k,i> \wedge i \neq j \wedge i \leqslant C_k \wedge j \leqslant C_m\}R^{(i)} \| R'^{(j)}\{q\})}{(u:\{p\}R^{(i)} \| R'^{(j)}\{q\})}$$

R7. Let $R \equiv l:?y$ occur in S_k and $R' \equiv l':x!t$ in S_m.

$$\text{a. } k \neq m: \frac{(u:\{p \wedge x[j] = <k,i> \wedge i \leqslant C_k \wedge j \leqslant C_m\}R^{(i)} \| R'^{(j)}\{q\})}{(u:\{p\}R^{(i)} \| R'^{(j)}\{q\})}$$

$$\text{b. } k = m: \frac{(u:\{p \wedge x[j] = <k,i> \wedge i \neq j \wedge i \leqslant C_k \wedge j \leqslant C_m\}R^{(i)} \| R'^{(j)}\{q\})}{(u:\{p\}R^{(i)} \| R'^{(j)}\{q\})}$$

R8. $R \equiv l:x?y$, $R' \equiv l':!t$ analogously.

R9.

$$\frac{(u:\{p\}R_1^{(i)};R_3^{(j)}\{p_1\}),\ (u:\{p_1\}R^{(i)}\|R'^{(j)}\{q_1\}),\ (u:\{q_1\}R_2^{(i)};R_4^{(j)}\{q\})}{(u:\{p\}<R_1;R;R_2>^{(i)}\|<R_3;R';R_4>^{(j)}\{q\})}$$

R10.

$$\frac{(u:\{p\}R_1^{(i)}\{q_1\}),\ (u:\{q_1\}R_2^{(j)}\{q\}),}{(u:\{p\}R_1^{(i)};R_2^{(j)}\{q\})}$$

Axioms 3 and 4 model communication resp. creation by (implicit) assignment. The rules 7, 8, 9 state that to prove some correctness assertion about a communication we can use some additional information: the relation, which must hold for the communication to take place, between the values of the variables used as references to the communication partners and the values of the logical variables i and j. We conclude the exposition of the proof-system with the following rules:

R11.

$$\frac{\{p\wedge C_1=1\wedge\bigwedge_{k=2}^{n}C_k=0\}u\{q\}}{\{p\}u\{q\}}$$

R12.

$$\frac{p\rightarrow p_1\ \{p_1\}u\{q_1\}\ q_1\rightarrow q}{\{p\}u\{q\}}$$

R13.

$$\frac{\{p\}u'\{q\}}{\{p\}u\{q\}}$$

Provided $FV(q)\cap AUX=\varnothing$, where u results from u' by deleting all assignments to $x\in AUX$, AUX a set of (auxiliary) variables appearing only in assignments of the form $x:=t$ such that for every occurrence of an assignment $x:=t$ in u' if $FV(t)\cap AUX\neq\varnothing$ then $x\in AUX$.

R14.

$$\frac{\{p\}u\{q\}}{\{p[(x;[1]:Dt)/x]\}u\{q\}}$$

where $x\in Pvar^{C_1}$ and x does not occur in u, and $x\notin FV(q)$.

5. Soundness

In this section we shall sketch a proof of the soundness of the proof-system for **P**. We consider only the case of rule 5, the process-rule, soundness of the other rules and axioms being a routine matter. We will make use of the following definitions and lemma:

DEFINITION 5.1
Let u be a bracketed unit. R, a subprogram of u, we call normal iff $R \equiv nil$ or every bracketed section of u occurs inside or outside of R.

DEFINITION 5.2
let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$. R a (labelled) subprogram of one of the S_k, $1 \leqslant k \leqslant n$. We define $after(R,S_k)$ as follows:
[1] If $R \equiv S_k$ then $after(R,S_k) \equiv nil$
[2] If $S_k \equiv \overset{m}{\underset{k=1}{\square}} b_m \rightarrow R_m$ and R occurs in R_p then $after(R,S_k) \equiv after(R,R_p)$
[3] If $S_k \equiv^* \overset{m}{\underset{k=1}{\square}} b_k \rightarrow R_k$ and R occurs in R_p then $after(R,S_k) \equiv after(R,R_p);S_k$
[4] If $S_k \equiv R_1;R_2$ then $after(R,S_k) \equiv after(R,R_1);R_2$ if R occurs in R_1 else $after(R,S_k) \equiv after(R,R_2)$
Next we define $before(R,S_k)$ such that if $after(R,S_k) \equiv nil;R'$ then $before(R,S_k) \equiv R;R'$ and if $after(R,S_k) \equiv nil$ then $before(R,S_k) \equiv R$.
The intuition behind these concepts should be clear.

LEMMA 5.1
let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, u bracketed, A_k, $1 \leqslant k \leqslant n$, sets of assumptions, then:
$A_k \vdash \{p_k\} S_k^{(i)} \{q_k\}$, i some logical variable, iff there exists for every normal subprogram R assertions $pre(R)$, $post(R)$ such that: (Let R be a subprogram of S_k, $1 \leqslant k \leqslant n$)
[0] $\{pre(R)\} R \{post(R)\} \in A_k$, R a bracketed section,
[1] $p_k \wedge i \leqslant C_k \rightarrow pre(S_k)$, $post(S_k) \wedge i \leqslant C_k \rightarrow q_k$.
[2] $pre(R) \wedge i \leqslant C_k \rightarrow post(R)[(x;[i]:t^{(i)})/x]$, $R \equiv x:=t$, $t \not\equiv self$.
[3] $pre(R) \wedge i \leqslant C_k \rightarrow post(R)[(x;[i]:<k,i>/x]$, $R \equiv x:=self$.
[4] $pre(R) \wedge i \leqslant C_k \rightarrow pre(R_1)$, $post(R_1) \wedge i \leqslant C_k \rightarrow pre(R_2)$, $post(R_2) \wedge i \leqslant C_k \rightarrow post(R)$, $R \equiv R_1;R_2$.
[5] $pre(R) \wedge b_l^{(i)} \wedge i \leqslant C_k \rightarrow pre(R_l)$, $post(R_l) \wedge i \leqslant C_k \rightarrow post(R)$, $l = 1, \ldots, m$, $R \equiv \overset{m}{\underset{k=1}{\square}} b_k \rightarrow R_k$.
[6] $pre(R) \wedge b_l^{(i)} \wedge i \leqslant C_k \rightarrow pre(R_l)$, $post(R_l) \wedge i \leqslant C_k \rightarrow pre(R)$, $l = 1, \ldots, m$, $pre(R) \wedge \overset{m}{\underset{l=1}{\bigwedge}} \neg b_l^{(i)} \wedge i \leqslant C_k \rightarrow post(R)$, $R \equiv^* \overset{m}{\underset{k=1}{\square}} b_k \rightarrow R_k$.
PROOF : routine.

Given $A_k \vdash \{p_k\} S_k^{(i)} \{q_k\}$ we define $VC(\{p_k\} S_k^{(i)} \{q_k\})$ to be the set of assertions corresponding with the clauses $1, \ldots, 6$ of the above mentioned lemma.
Given $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, a first-order formula I, and for $1 \leqslant k \leqslant n$ proofs $A_k \vdash \{p_k\} S_k^{(i)} \{q_k\}$, for some logical variable i, $CP(A_1, \ldots, A_n, I)$ denotes the set of (correctness-) assertions corresponding with the clauses b and c of the cooperation-test.
These (correctness-) assertions are formulated w.r.t to some logical variable j distinct from i, such that $i, j \notin FV(I)$ and j does not occur in the pre (post)-condition of any assumption.
Now we can phrase the soundness of the process-rule as follows: if all assertions of $VC(\{p_k\} S_k^{(i)} \{q_k\})$, $1 \leqslant k \leqslant n$ and $CP(A_1, \ldots, A_n, I)$ are true and for every bracketed section R of, say, S_k, $1 \leqslant k \leqslant n$, $\{pre(R)\} R \{post(R)\} \in A_k$ (and I and the assertions $pre(R)$, $post(R)$, p_k, q_k, R a

bracketed section of S_k, $1 \leq k \leq n$ satisfy the additional syntactic restrictions mentioned in the cooperation test) then $\{p[\underline{1}/i] \wedge I\} u \{\bigwedge_{k=1}^{m} \forall 1 \leq i \leq C_k q_k \wedge I\}$ is true.

It is easy to see that this formulation of the soundness of the process-rule is implied by the following lemma:

LEMMA 5.2

Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, u bracketed. Assume there exists for every normal subprogram R of S_k, $1 \leq k \leq n$, assertions $pre(R)$, $post(R)$ and sets A_k, $1 \leq k \leq n$, of correctness assertions of bracketed sections occurring in u such that all assertions of $VC(\{p_k\} S_k^{(i)} \{q_k\})$, $1 \leq k \leq n$, $CP(A_1, \ldots, A_n, I)$ are true (i some logical variable) and for every bracketed section R of, say, S_k, $1 \leq k \leq n$, $\{pre(R)\} R \{post(R)\} \in A_k$ (and I and the assertions $pre(R)$, $post(R)$, p_k, q_k, R a bracketed section of S_k, $1 \leq k \leq n$, satisfy the syntactic restrictions mentioned in the cooperation test). Let $c: <X_l, \sigma_l, u>_{l=1,\ldots,m}$ such that for $1 \leq l < m$ $<X_l, \sigma_l, u> \xrightarrow[1]{h_l} <X_{l+1}, \sigma_{l+1}, u>$ for some activation, communication record h_l, $<X_1, \sigma_1, u>$ being a initial configuration w.r.t. u, and for every $<\alpha, S> \in X_m$: $S \equiv before(R, S_{\alpha_1})$ or $S \equiv after(R, S_{\alpha_1})$, R a bracketed section, or $S \equiv nil$. Furthermore suppose that: $\models p_1[1/i] \wedge I(\nu)(\sigma_1)$

Then:

$\models I(\nu)(\sigma_m)$ and

$\forall <\alpha, S> \in X_m$:

If $S \equiv before(R, S_{\alpha_1})$ then $\models pre(R)(\nu')(\sigma_m)$,

if $S \equiv after(R, S_{\alpha_1})$ and for some $1 \leq k \leq m$, $<\alpha, before(R, S_{\alpha_1})> \in X_k$ such that for every $k < l \leq m$ $<\alpha, before(R', S_{\alpha_1})> \in X_l$ implies that R' is a subprogram of R, then $\models post(R)(\nu')(\sigma_m)$,

if $S \equiv nil$ then $\models q_{\alpha_1}(\nu')(\sigma_m)$.

Where $\nu' = \nu\{\alpha_2 / i\}$

PROOF :

Induction to $|h|$, the length of the history of the computation c.

$|h| = 0$:

We note that in this case $\models I(\nu)(\sigma_m)$ because $\models I(\nu)(\sigma_1)$ and for every $x \in FV(I) \cup \{C_1, \ldots, C_n\}$ $\sigma_1(x) = \sigma_m(x)$.

We prove with induction to m, the length of the computation c that:

if $S \equiv before(R, S_1)$ and R is minimal w.r.t the relation $''$ is a normal subprogram of $''$ then $\models pre(R)(\nu')(\sigma_m)$,

if $S \equiv after(R, S_1)$ and for some $1 \leq k \leq m$ $<<1,1>, before(R, S_1)> \in X_k$ such that for every $k \leq l \leq m$ $<<1,1>, before(R', S_1)> \in X_l$ imlies that R' is a subprogram of R then $\models post(R)(\nu')(\sigma_m)$ and

if $S \equiv nil$ then $\models q_1(\nu')(\sigma_m)$,

where $\nu' = \nu\{1/i\}$.

m = 1: $X_1 = \{<<1,1>, S_1>\}$.

Given is that: $\models p_1 \wedge i \leq C_1 \rightarrow pre(S_1)$, $\models p_1 \wedge i \leq C_1[\underline{1}/i](\nu)(\sigma_1)$ so we can conclude that $\models pre(S_1)(\nu')(\sigma_1)$.

Now let R be minimal such that $before(R, S_1) \equiv S$ then $S_1 \equiv R_1; \cdots; R_k$, $R_1 \equiv R$.

From $pre(S_1) \wedge i \leq C_1 \rightarrow pre(R)$ we conclude $\models pre(R)(\nu')(\sigma_m)$.

m > 1: we have to consider two main cases:

1. $X_m = \{<<1,1>, nil; S>\}$.

   1.1 $X_{m-1} = \{<<1,1>, l:x:=t; S>\}$:

   Ind.hyp.: $\models pre(l:x:=t)(\nu')(\sigma_{m-1})$,

   Given: $\models pre(l:x:=t) \wedge i \leq C_1 \rightarrow post(l:x:=t)[(x;[i]:t^{(i)})/x]$.

   So: $\models post(l:x:=t)[(x;[i]:t^{(i)})/x](\nu')(\sigma_{m-1})$.

   $\sigma_m = \sigma_{m-1}\{V(t^{(i)})(\nu')(\sigma_{m-1}) / <x, 1>\}$.

   Thus: $\models post(l:x:=t)(\nu')(\sigma_m)$.

Let $nil;S \equiv after(R,S_1)$ and for some $1 \leq k < m$, $X_k = \{<<1,1>, before(R,S_1)>\}$ such that for every $k \leq l < m$ if $<\alpha, before(R',S_1)> \in X_l$ then R' a subprogram of R. An easy induction to the complexity of R establishes that $\models post(l:x:=t) \wedge i \leq C_1 \rightarrow post(R)$.

1.2 $X_{m-1} = \{<<1,1>, before(R,S_1)>\}$ where $R \equiv l:* \overset{k}{\underset{l=1}{\square}} b_l \rightarrow R_l$, and $\models \overset{k}{\underset{l=1}{\wedge}} \neg b_l^{(i)}(\nu')(\sigma_{m-1})$.

Ind.hyp: $\models pre(R)(\nu')(\sigma_{m-1})$

Given: $\models pre(R) \wedge \overset{k}{\underset{l=1}{\wedge}} \neg b_l^{(i)} \wedge i \leq C_1 \rightarrow post(R)$

$\sigma_{m-1} = \sigma_m$, thus $\models post(R)(\nu')(\sigma_m)$

Let $nil;S \equiv after(R',S_1)$ and for some $1 \leq n < m$, $X_n = \{<<1,1>, before(R',S_1)>\}$ such that for every $n \leq l < m$ if $<\alpha, before(\overline{R},S_1)> \in X_l$ then $\overline{R}$ a subprogram of R'. With induction to R' one can prove $\models post(R) \wedge i \leq C_1 \rightarrow post(R')$.

2. $X_m = \{<<1,1>, S>\}$, $S \not\equiv nil;S'$ for any S'

2.1. $X_{m-1} = \{<<1,1>, before(R,S_1>\}$, $R \equiv \overset{k}{\underset{i=1}{\square}} b_i \rightarrow R_i$ and $S \equiv before(R_n,S_1)$, $\models b_n^{(i)}(\nu')(\sigma_{m-1})$, for some $1 \leq n \leq k$.

Ind.hyp.: $\models pre(R)(\nu')(\sigma_{m-1})$.

Given: $\models pre(R) \wedge b_n^{(i)} \wedge i \leq C_1 \rightarrow pre(R_n)$. Thus: $\models pre(R_n)(\nu')(\sigma_{m-1})$. $\sigma_{m-1} = \sigma_m$ so $\models pre(R_n)(\nu')(\sigma_m)$.

Let R' be minimal such that $S \equiv before(R',S_1)$, then $R' \equiv R_n$ or $R_n \equiv R_1'; \cdots ; R_l'$, where $R_1' \equiv R'$.

We know that $\models pre(R_n) \wedge i \leq C_1 \rightarrow pre(R_1')$.

So $\models pre(R')(\nu')(\sigma_m)$.

2.2. $X_{m-1} = \{<<1,1>, before(R,S_1)>\}$, $R \equiv * \overset{k}{\underset{i=1}{\square}} b_i \rightarrow R_i$ and $S \equiv before(R_n,S_1)$, $\models b_n^{(i)}(\nu')(\sigma_{k-1})$, for some $1 \leq n \leq k$.

Analogously to case 2.1..

2.3. $X_{m-1} = \{<<1,1>, nil;S>\}$.

Let $S \equiv before(R,S_1)$, R minimal.

Let R' be maximal such that: $after(R',S_1) \equiv nil;S$, and for some $1 \leq k \leq m-1$ $<<1,1>, before(R',S_1) \in X_k$, and for every $k < l \leq m$, $<\alpha, before(\overline{R},S_{\alpha_1})> \in X_l$ implies that $\overline{R}$ a subprogram of R'.

Ind.hyp.: $\models post(R')(\nu')(\sigma_{m-1})$. There are the following two cases to consider:

a. $R';R \in subprog(S_1)$ then: $post(R') \wedge i \leq C_1 \rightarrow pre(R)$.

b. $R \equiv * \overset{k}{\underset{j=1}{\square}} b_j \rightarrow R_j$, $R' \equiv R_p$, $1 \leq p \leq k$. Then: $post(R') \wedge i \leq C_1 \rightarrow pre(R)$.

This concudes the basis of the induction.

REMARK :
In the sequel we will write $<X,\sigma>$ instead of $<X,\sigma,u>$ when it is clear from the context which unit u is meant.
End of remark.

$|h| > 0$:

1. $h = h_1 \circ <\alpha,\beta>$:

It is not difficult to prove that we can assume that
$c = <X_1,\sigma_1>, \ldots, <X_{k_1},\sigma_{k_1}>, \ldots, <X_{k_2},\sigma_{k_2}>, \ldots, <X_{k_3},\sigma_{k_3}>, \ldots, <X_m,\sigma_m>$,
where:

1. $<\alpha, before(R_1;x:=new(C_{\beta_1});R_2, S_{\alpha_1})> \in X_{k_1}$, $<\alpha, after(R_1;x:=new(C_{\beta_1});R_2, S_{\alpha_1})> \in X_{k_2}$,
2. from configuration $<X_{k_1},\sigma_{k_1}>$ to $<X_m,\sigma_m>$ only the processes α, β are active.

3. from $<X_{k_1},\sigma_{k_1}>$ to $<X_{k_3},\sigma_{k_3}>$ only the process α is active, the result of this activity being the activation of β.
4. from $<X_{k_3},\sigma_{k_3}>$ to $<X_m,\sigma_m>$ only β is performing.

(see appendix 2)

Let $<\gamma,before(R,S_{\gamma_1})>\in X_{k_1}$, R a bracketed section, ind.hyp. (note that for every $<\gamma,S>\in X_k$ $S\equiv before(R,S_{\gamma_1})$ or $S\equiv after(R,S_{\gamma_1})$, R a bracketed section, or $S\equiv nil$):
$\models pre(R)(\nu\{\gamma_2/i\})(\sigma_{k_1})$, in case $\gamma\neq\alpha$, β: $\sigma_{k_1}(<x,\gamma_2>)=\sigma_m(<x,\gamma_2>)$, $x\in var(S_{\gamma_1})$, so $\models pre(R)(\nu\{\gamma_2/i\})(\sigma_m)$
(remember that all the free program-variables of $pre(R)$ are subscripted by a free occurrence of i and that $C_1,\ldots,C_n$ do not occur in $pre(R)$).
The same reasoning applies to the case that $<\gamma,after(R,S_{\gamma_1})>\in X_{k_1}$, (R a bracketed section) or $<\gamma,nil>\in X_{k_1}$.
In particular: $\models I\wedge pre(S)(\nu\{\alpha_2/i\})\ (\sigma_{k_1}\ S\equiv R_1;x:=new(C_{\beta_1});R_2$.
Furthermore: $\models\{I\wedge pre(S)\}S^{(i)}\ \{I\wedge post(S)\wedge p_{\beta_1}[(x[i])_2/i]\}$ holds.
We know that $<\sigma_{k_1},\sigma_{k_2}>\in M(S^{(i)})(\nu\{\alpha_2/i\})$.
So $\models I\wedge post(S)\ \wedge p_{\beta_1}[(x[i])_2/i]\ (\nu\{\alpha_2/i\})(\sigma_{k_2})$.
Following the same pattern of reasoning used in the case of $|h|=0$ we conclude that $\models pre(R)(\nu\{\alpha_2/i\})(\sigma_m)$, $\models post(R)(\nu\{\alpha_2/i\})(\sigma_m)$, resp. $\models q_{\alpha_1}(\nu\{\alpha_2/i\})\ (\sigma_m)$ in case $<\alpha,before(R,S_{\alpha_1})>$, $<\alpha,after(R,S_{\alpha_1})>$, R a bracketed section, resp. $<\alpha,nil>\in X_m$.
Because the free program-variables of I are not changed by the computation $<X_{k_2},\sigma_{k_2}>,\ldots,<X_m,\sigma_m>$ we conclude that $\models I(\nu)(\sigma_m)$.
Finally because $\models p_{\beta_1}[(x[i])_2/i]\ (\nu\{\alpha_2/i\})(\sigma_{k_2})$ we know that $\models p_{\beta_1}\ (\nu\{\beta_2/i\})(\sigma_{k_2})$ ($x\notin change(R_2)$!).
Thus: $\models p_{\beta_1}(\nu\{\beta_2/i\})\ (\sigma_{k_3})$. Again reasoning as in the case $|h|=0$ gives us the desired result.

2. $h=h_1\circ<d,\alpha,\beta>$:

It is not difficult to prove that we may assume that:
$c=<X_1,\sigma_1>,\ldots,<X_k,\sigma_k>,\ldots,<X_l,\sigma_l>,\ldots,<X_{l'},\sigma_{l'}>,\ldots,<X_m,\sigma_m>$,
where,

1. $<\alpha,before(R,S_{\alpha_1})>$, $<\beta,before(R',S_{\beta_1})>\in X_k$,
   (α, β are to enter the bracketed sections R resp. R' execution of which consists of the transfer of the value d from α to β)
2. from $<X_k,\sigma_k>$ to $<X_m,\sigma_m>$ only α and β are performing steps.
3. $<\alpha,after(R,S_{\alpha_1})>$, $<\beta,after(R',S_{\beta_1})>\in X_l$.
4. From $<X_l,\sigma_l>$ to $<X_{l'},\sigma_{l'}>$ only the process α is executing.
5. From $<X_{l'},\sigma_{l'}>$ only β is performing.

(see appendix 2)
For $\gamma\neq\alpha$, β we reason as in the previous case.
Ind.hyp.: $\models I(\nu)(\sigma_k)$, $\models pre(R)(\nu\{\alpha_2/i\})(\sigma_k)$ $\models pre(R')(\nu\{\beta_2/i\})(\sigma_k)$.
So for some logical variable j distinct from i, such that $i,j\notin FV(I)$, $j\notin FV(pre(R),pre(R'))$, $\models I\wedge pre(R)\wedge pre(R')[j/i]\ (\nu\{\alpha_2,\beta_2/i,j\})(\sigma_k)$.
Furthermore: $<\sigma_k,\sigma_l>\in M(R^{(i)}\|R'^{(j)})\ (\nu\{\alpha_2,\beta_2/i,j\})$.
Thus: $\models I\wedge post(R)\wedge post(R')[j/i]\ (\nu\{\alpha_2,\beta_2/i,j\})(\sigma_l)$
From which we conclude: $\models I(\nu)(\sigma_l)$, $\models post(R)(\nu\{\alpha_2/i\})(\sigma_l)$, $\models post(R')(\nu\{\beta_2/i\})(\sigma_l)$.
Because for $x\in FV(I)\cup\{C_1,\ldots,C_n\}$ $\sigma_l(x)=\sigma_m(x)$, we deduce that $\models I(\nu)(\sigma_m)$.
Finally reasoning as in the case $|h|=0$ gives us the desired result, (making use of the fact that the assertions we must show to hold in the last state of the computation, w.r.t. some particular process, are invariant over the actions of all the other processes).

6. Completeness

Let $\models\{p\}u\{q\}$, u a unit. We want to prove $\vdash\{p\}u\{q\}$ along the lines of [6]. We want to extend u by substituting, for example, for a IO statement $x?y$ the statement $x?y;h:=h\circ<x,self,y>$, where h is a new variable used to record the sequence of communications and activations each process of a particular class has executed.
In CSP this sequence contains only communication records and can be coded into an integer. But this is not the case w.r.t. the language **P**: at the programming level we can not manipulate a variable which refers to a process as a pair of integers.
We therefore extend our language to be able to manipulate sequences of data in the following way.
Introduce for each class name C a set of new program variables $hist^C$.
Define: $r::=t \mid self$
$Trace::=h,h\in\bigcup_C hist^C \mid <r_1,\ldots,r_n> \mid trace_1\circ trace_2$.
Extend the class of statements by:
$S::=l:h:=trace,\ h\in\bigcup_C hist^C$.

We extend the set of terms of our assertion language as follows:
$Tt::=h[It],\ h\in\bigcup_C hist^C \mid <Dt_1,\ldots,Dt_n> \mid Tt_1\circ Tt_2$
$ATt::=(h;[It]:Tt),\ h\in\bigcup_C hist^C$
$It::=|Tt| \mid Tt(It)$
$Pt::=Tt(It)$

We will prove completeness w.r.t. this extended programming language (note that for each program u' formulated in this language we have that $M(u')=M(u)$ modulo the set of history variables occurring in u' u obtained from u' by deleting all assignments to the socalled trace variables).
Now let $\models\{p\}u\{q\}$. We will show that this correctness assertion is deducible. Say $u\equiv<C_1\leftarrow S_1,\ldots,C_n\leftarrow S_n>$. We introduce the auxiliary (trace) variables $h_1,\ldots,h_n$. Each variable h_k will record for each activation of C_k the sequence of communications, activations it has executed. For each variable $x\in var(S_1)$ we introduce a corresponding fresh program variable $z_x\in Pvar^{C_1}$ (not occurring in p, q, u). These variables are used to freeze the initial values of the root proces.
Without loss of generality we assume that $FV(q)\cap Pvar^{C_i}\subseteq var(S_i)$, $1\leqslant i\leqslant n$.
We extend u to u' as follows:
Substitute
$x?y$ in S_k by $<x?y;h_k:=h_k\circ<0,y,x,self>>$,
$x!t$ in S_k by $<x!t;h_k:=h_k\circ<0,e,self,x>>$,
$!t$ in S_k by $<!t;h_k:=h_k\circ<0,e,self,nil>>$,
$?x$ in S_k by $<?x;h_k:=h_k\circ<0,x,nil,self>>$,
$x:=new(C)$ in S_k by $<x:=new(C);h_k:=h_k\circ<1,self,x>>$.
We define $nil\circ<0,d,\alpha,\beta>=<0,d,\alpha,\beta>$, and $nil\circ<1,\alpha,\beta>=<1,\alpha,\beta>$. The empty process *nil* is used to mark that the communication patner is not known.
It is obvious how to view each variable h_k for each active process of class C_k as a sequence of communication and activation records.
Let $u'\equiv<C_1\leftarrow S'_1,\ldots,C_n\leftarrow S'_n>$.

DEFINITION 6.1
We define $[h]_\gamma$, $\gamma\in\mathbf{N}\times\mathbf{N}$, h a sequence of communication, activation records as follows:
$length(h)=0$: $[h]_\gamma=nil$

length$(h)=n+1$:

$h=h'\circ<<d,\alpha,\beta>>$:

$[h]_\gamma=[h']_\gamma\circ<<d,\alpha,\beta>>$ if $\gamma=\alpha$ *or* $\gamma=\beta$,

$=[h']_\gamma$ *otherwise*

$h=h'\circ<<\alpha,\beta>>$:

$[h]_\gamma=[h']_\gamma\circ<<,\alpha,\beta>>$ if $\gamma=\alpha$,

$=[h']_\gamma$ *otherwise.*

DEFINITION 6.2

Let $x,y\in\mathbf{D}^3$ or $x,\ y\in\mathbf{D}^2$:

$x=_U y$ *iff* $(|x|=|y|=2 \wedge x=y) \vee (|x|=|y|=3 \wedge x_1=y_1 \wedge (x_i=y_i\neq nil \vee x_i=nil \vee y_i=nil),\ i=2,3)$

So, for example, $<d,\alpha,\beta>=_U<d,\alpha,nil>$.
This notion is introduced to cope with the situation that locally the communication partner is not always known.

DEFINITION 6.3

Let $h,\ h'$ be sequences of communication, activation records:
$h=_U h'$ *iff* *length*$(h)=$*length*(h') and $\forall 1\leqslant i\leqslant$ *length*(h) $(h_i=_U h'_i)$.

Using the expressiveness of the underlying domain of values we will construct proof-outlines for the components of the unit u' and a global invariant.
Let $p'\equiv p\wedge h_1[1]=nil \wedge \underset{x\in var(S_1)}{\wedge} x[1]=z_x[1]$.

DEFINITION 6.4

$\vDash I(\nu)(\sigma)$ iff $\exists X,\ Y,\ h,\ k,\ \bar{\sigma},\ \sigma'$:

[a] $<X,\bar{\sigma},u'>\xrightarrow[k]{h}<Y,\sigma',u'>$, $<X,\bar{\sigma},u'>$ an initial configuration.

[b] $\vDash p'(\nu)(\bar{\sigma})$

[c] $\sigma(h_k)=\sigma'(h_k)$, $1\leqslant k\leqslant n$ and $\sigma(z_x)(1)=\sigma'(z_x)(1)$, $x\in var(S_1)$

[d] $\sigma(C_k)=\sigma'(C_k)$, $1\leqslant k\leqslant n$

[e] $\forall 1\leqslant i\leqslant C_k(\sigma'(h_k)(i)=_U [h]_{<k,i>})$, $k=1,\ldots,n$.

Stated very roughly this formula I collects all those states in which all the activated processes occur outside a bracketed section.

LEMMA 6.1

The above mentioned clauses $a,\ldots,e$ are expressible in the assertion language such that the resulting formula I contains as free only the variables of the set $\{h_1,\ldots,h_n\}\cup\{z_x: x\in var(S_1)\}$.

PROOF :

See appendix 3.

DEFINITION 6.5

Let R be a normal (see definition 5.1) subprogram of, say, S'_k, i some logical variable,
$\vDash pre(R)^{(i)}(\nu)(\sigma)$ iff $\exists X,\ Y,\ h,\ k,\ \bar{\sigma},\ \sigma'$:

[a] $<X,\bar{\sigma},u'>\xrightarrow[k]{h}<Y,\sigma',u'>$, $<X,\bar{\sigma},u'>$ an initial configuration.

[b] $\models p'(\nu)(\overline{\sigma})$
[c] $<<k,\nu(i)>,before(R,S'_k)>\in Y$
[d] $\sigma(x)(\nu(i))=\sigma'(x)(\nu(i)),\ x\in var(S'_k)$.
[e] $\sigma(z_x)(1)=\sigma'(z_x)(1),\ x\in var(S_1)$

This formula $pre(R)^{(i)}$ collects all those states σ where the process $<k,\nu(i)>$ is going to execute R.

LEMMA 6.2
The previously mentioned clauses $a,\ldots,e$ are expressible in the assertion language such that the free variables of the resulting formula $pre(R)$ are contained in $var(S'_k)\cup\{i\}\ \cup\{z_x:\ x\in var(S_1)\}$ and all the program variables of $var(S'_k)$ occur only subscripted by a free occurence of the logical variable i.
PROOF :
See appendix 3.

DEFINITION 6.6
Let R be a normal subprogram of S'_m, i some logical variable,
$\models post(R)^{(i)}(\nu)(\sigma)$ iff $\exists X,\ Y,\ h,\ k,\ \overline{\sigma},\ \sigma'$ such that:
[a] $<X,\overline{\sigma},u'>$ is an initial configuration,
$<X,\overline{\sigma},u'>\xrightarrow[k]{h}<Y,\sigma',u'>$
[b] $\models p'(\nu)(\overline{\sigma})$.
[c] $<<m,\nu(i)>,after(R,S'_m)>\in X_k$,
[d] $\sigma(x)(\nu(i))=\sigma_k(x)(\nu(i)),\ x\in var(S'_m),\ \sigma(z_x)(1)=\sigma'(z_x)(1),\ x\in var(S_1)$

This formula $post(R)^{(i)}$ collects all those states σ in which the process $<m,\nu(i)>$ has just finished executing R.

LEMMA 6.3
The clauses $a,\ldots,e$ of the previous definition are expressible in the assertion language such that the free variables of the resulting formula $post(R)$ are contained in $var(S'_m)\cup\{i\}\ \cup\{z_x:\ x\in var(S_1)\}$ and all the program variables of $var(S'_m)$ occur only subscripted by a free occurence of i.
PROOF :
See appendix 3.

We consider a routine matter to check that w.r.t a particular component of u' the assertions, $pre(R)^{(i)}$, $post(R)^{(i)}$, i some logical variable, R a normal subprogram of that componenent, constitute a valid proof-outline (use lemma 5.1). What remains to be shown is that these proof-outlines cooperate. This is done by first establishing the truth of the following correctness assertions:
$\models\{pre(R_1)^{(i)}\ \wedge pre(R_2)^{(j)}[j\ /\ i]\wedge I\}\ R_1^{(i)}\|R_2^{(j)}\ \{post(R_1)^{(i)}\ \wedge post(R_2)^{(j)}[j\ /\ i]\wedge I\}$, R_1, R_2 being matching bracketed sections, j a fresh logical variable, and $\{I\wedge pre(R)^{(i)}\}R^{(i)}\{I\wedge post(R)^{(i)}\ \wedge pre(S_k)^{(i)}[(x[i])_2\ /\ i]\}$, R a bracketed section containing a new-statement of the form: $x:=new(C_k)$. After that we show that arbitrary true correctness assertions like the ones mentioned are deducible. To establish the truth of these correctness assertions we need the following definition and lemma, the so called "merging lemma".

DEFINITION 6.7
Let A be a finite (non-empty) set of pairs of natural numbers (= processes), σ a state such that for $\alpha\in A$, $1\leqslant\alpha_1\leqslant n$, $1\leqslant\alpha_2\leqslant\sigma(C_{\alpha_1})$. Let for $\alpha\in A$ R_α be a subprogram of S'_{α_1} or be equal to *nil*, and $\nu\in Env\rightarrow$N.

Then we call the set $\{R_\alpha\colon \alpha\in A\}$ $(\sigma,\nu,A)-$*reachable*
iff
$\exists Y,\ h,\ k,\ \sigma,\ \sigma'$:

[a] $<X,\bar{\sigma},u'>\xrightarrow[h]{k}<Y,\sigma',u'>$, $<X,\bar{\sigma},u'>$ an initial configuration.

[b] $\vDash p'(\nu)(\bar{\sigma})$

[c] $\forall\alpha\in A\ \exists<\alpha,S>\in Y$ such that: if $R_\alpha\equiv nil$ then $S\equiv nil$ else $S\equiv before(R_\alpha,\ S'_{\alpha_1})$ and $\sigma(x)(\alpha_2)=\sigma'(x)(\alpha_2)$, $x\in var(S'_{\alpha_1})$.

[d] $\sigma(z_x)(1)=\sigma'(z_x)(1)$, $x\in var(S_1)$

[e] If furthermore $\sigma(x)=\sigma'(x)$, $x\in\{h_1,\ldots,h_n,C_1,\ldots,C_n\}$ and $\sigma'(h_k)(i)=_U[h]_{<k,i>}$, $1\leqslant k\leqslant n$, $1\leqslant i\leqslant\sigma'(C_k)$
we speak of $(I,\sigma,\nu,A)-$*reachability* of $\{R_\alpha\colon \alpha\in A\}$.

Note that $\vDash pre(R)^{(i)}(\nu)(\sigma)$ and $\nu(i)\leqslant\sigma(C_k)$ implies that R is $(\sigma,\nu,<k,\nu(i)>)-$*reachable* (R occurring in S_k). The following lemma states the conditions under which we can fuse different computations of u' into one computation.

LEMMA 6.4 (merging-lemma):
Let A be a finite set of processes such that for every $\alpha\in A$ R_α is a normal subprogram and $(\sigma,\nu,\alpha)-$*reachable* (and for $\alpha\in A$, $1\leqslant\alpha_1\leqslant n$, $1\leqslant\alpha_2\leqslant\sigma(C_{\alpha_1})$). Let furthermore $\vDash I(\nu)(\sigma)$.
then:
$\{R_\alpha\colon \alpha\in A\}$ $(I,\sigma,\nu,A)-$*reachable*

PROOF :
See appendix 1..

Given this lemma we can prove the following lemma:

LEMMA 6.5
Let R_1, R_2 be two matching bracketed sections, occurring in, say, S'_l resp. S'_m, i, j some logical variables such that $i,\ j\notin FV(I)$.
Then:
$\vDash\{pre(R_1)^{(i)}\wedge pre(R_2)^{(i)}[j\ /\ i]\wedge I\}\ R_1^{(i)}\|R_2^{(j)}\ \{post(R_1)^{(i)}\wedge post(R_2)^{(i)}[j\ /\ i]\wedge I\}$.

PROOF :
Let $(\vDash pre(R_1)^{(i)}\wedge pre(R_2)^{(i)}[j\ /\ i]\wedge I)(\nu)(\sigma)$ and $<\sigma,\sigma_0>\in M(R^{(i)}\|R^{(j)})(\nu)$.
From the definition of the formulae $pre(R_1)^{(i)}$, $pre(R_2)^{(i)}$ and the merging-lemma it follows that (note that $pre(R)^{(i)}[j\ /\ i]\Leftrightarrow pre(R)^{(j)}$ and $\nu(i)\leqslant\sigma(C_l)$, $\nu(j)\leqslant\sigma(C_m)$):
$\exists X,\ Y,\ \bar{\sigma},\ \sigma',\ h,\ k$ such that

a. $<X,\bar{\sigma},u'>\xrightarrow[k]{h}<Y,\sigma',u'>$ $<X,\bar{\sigma},u'>$ an initial configuration.

b. $\vDash p'(\nu)(\bar{\sigma})$

c. $\sigma'(x)=\sigma(x)$, $x\in FV(I)\cup\{C_1,\ldots,C_n\}$.

d. $\sigma'(x)(\nu(i))=\sigma(x)(\nu(i))$, $x\in var(S'_l)$

e. $\sigma'(x)(\nu(j))=\sigma(x)(\nu(j))$, $x\in var(S'_m)$

f. $\sigma(z_x)(1)=\sigma'(z_x)(1)$, $x\in var(S_1)$

g. $<<l,\nu(i)>,before(R_1,S_l)>$, $<<m,\nu(j)>,before(R_2,S_m)>\in Y$

h. $\sigma'(h_k)(i)=_U[h]_{<k,i>}$, $1\leqslant k\leqslant n$, $1\leqslant i\leqslant\sigma'(C_k)$.

Note that: $\sigma_0(C_j)=\sigma(C_j)=\sigma'(C_j)$, $j=1,\ldots,n$.
We define the state σ_1 as follows:

$$\sigma_1(x)(s)=\sigma_0(x)(s),\ x\in var(S'_r),\ (r=l\wedge s=\nu(i))\vee(r=m\wedge s=\nu(j)).$$

For the remaining cases σ_1 agrees with σ'. It follows that:
$<X,\bar{\sigma},u'> \xrightarrow[k]{h} <Y,\sigma',u'> \xrightarrow[k']{h'} <Y',\sigma_1,u'>$, for some h', k',
where
$Y' = Y / \{<<l,v(i)>,before(R_1,S'_l)>, <<m,v(j)>,before(R_2,S'_m)>\} \cup Z$,
($Z = \{<<l,v(i)>,after(R_1,S'_l)>, <<m,v(j)>,after(R_2,S'_m)>\}$).
We conclude: $\models (post(R_1)^{(i)} \wedge post(R_2)^{(i)}[j/i] \wedge I)(v)(\sigma_1)$ ($i,j \notin FV(I)$).
So: $\models (post(R_1)^{(i)} \wedge post(R_2)^{(i)}[j/i] \wedge I)(v)(\sigma_0)$ (making use of c. above).
Which finishes the proof.

We shall now prove that arbitrary true correctness assertions about I/O bracketed sections are deducible.

LEMMA 6.6
Let R, R' be two matching bracketed sections, occurring in S'_k resp. S'_m. Then $\models \{p\} R^{(i)} \| R'^{(j)} \{q\}$ implies $\vdash \{p\} R^{(i)} \| R'^{(j)} \{q\}$.

PROOF :
Let: $\models p_1(v)(\sigma) \Leftrightarrow \exists \sigma' (\models p(v)(\sigma') \wedge <\sigma',\sigma> \in M(R_1^{(i)}; R_3^{(j)}))$, and
$\models p_2(v)(\sigma) \Leftrightarrow \forall \sigma' (<\sigma,\sigma'> \in M(R_2^{(i)}; R_4^{(j)}) \rightarrow \models q(v)(\sigma'))$,
where $R \equiv R_1; C; R_2$, $R' \equiv R_3; A; R_4$.
Using the coding techniques as presented in appendix 3 it can be shown that p_1 and p_2 can be formulated in the assertion language. It follows that: $\models \{p\} R_1^{(i)}; R_3^{(j)} \{p_1\}$ and $\models \{p_2\} R_2^{(i)}; R_4^{(j)} \{q\}$.
From the completeness of the Hoare logic for sequential programs (which carries over to its parameterized version) it follows that: $\vdash \{p\} R_1^{(i)}; R_3^{(j)} \{p_1\}$ and $\vdash \{p_2\} R_2^{(i)}; R_4^{(j)} \{q\}$.
Furthermore: $\models \{p_1\} C^{(i)} \| A^{(j)} \{p_2\}$.
Let: $C \equiv x?y$, $A \equiv z!t$, $k \neq m$, the other cases are treated similar. It is easily shown that:
$\models p_1 \wedge x[i] = <m,j> \wedge z[j] = <k,i> \wedge i \leq C_k \wedge j \leq C_m \rightarrow p_2[(y;[i]:t^{(j)})/y]$
Thus: $\vdash \{p_1\} C^{(i)} \| A^{(j)} \{p_2\}$. Finally apply the formation rule (R9).

The only thing left to do is to prove the following lemma:

LEMMA 6.7
Let $<R> \equiv <x := new(C_k); h_m := h_m \circ <self, x>>$ in S'_m.
Then $\vdash \{I \wedge pre(R)^{(i)}\}\ R^{(i)}\ \{I \wedge post(R)^{(i)} \wedge pre(S'_k)^{(i)}[(x[i])_2 / i]\}$.
Where $i \in Lvar$, $i \notin FV(I)$.
PROOF :
define:

$$f_0 = C_k + 1$$

$$f_i = (y_i;[C_k+1]:nil),\ i = 1, \ldots, l,\ \{y_1, \ldots, y_l\} = var(S_k)$$

$$f_{l+1} = (x;[i]:<k, C_k+1>)$$

$$f_{l+2} = (h_m;[i]:h_m[i] \circ <<m,i>, x[i]>)$$

Let $\psi \equiv (I \wedge post(R)^{(i)} \wedge pre(S'_k)^{(i)}[(x[i])_2 / i])[f_{l+2} / h_m]$.
We prove that $\models I \wedge pre(R)^{(i)} \wedge i \leq C_m \rightarrow \psi[f_0 / C_k, f_1 / y_1, \ldots, f_l / y_l, f_{l+1} / x]$ as follows:
Let $\models I \wedge pre(R)^{(i)} \wedge i \leq C_m(v)(\sigma)$.
Then (merging lemma): $\exists X, Y, \bar{\sigma}, \sigma', h, k$:

$$a.\ <X,\bar{\sigma},u'> \xrightarrow[k]{h} <Y,\sigma',u'>$$

b. $<<m,\nu(i)>,before(R,S'_m)>\in Y$

c. $\sigma'(x)(\nu(i))=\sigma(x)(\nu(i))$, $x\in var(S'_m)$

d. $\sigma(x)=\sigma'(x)$, $x\in FV(I)$

e. $\sigma'(h_k)(i)=[h]_{<k,i>}$, $1\leqslant k\leqslant n$, $1\leqslant i\leqslant\sigma'(C_k)$.

f. $\sigma(z_x)(1)=\sigma'(z_x)(1)$, $x\in var(S_1)$

Let $<Y,\sigma',u'>\underset{k'}{\overset{h'}{\rightarrow}}<Z,\sigma_0,u'>$, such that all the k'-steps are taken by process $<m,\nu(i)>$ and $<<m,\nu(i)>,after(R,S'_m)>\in Z$.
Let $\sigma_1=\sigma\{V(f_0)(\nu)(\sigma)/C_k,\ V(f_1)(\nu)(\sigma)/y_1,\ \ldots,\ V(f_l)(\nu)(\sigma)/y_l, V(f_{l+1})(\nu)(\sigma)/x\}$
Then:

$$\models\psi[f_0/C_k,f_1/y_1,\ldots,f_l/y_l,f_{l+1}/x](\nu)(\sigma)\Leftrightarrow$$

$$\models\psi(\nu)(\sigma_1)\Leftrightarrow$$

$$\models I\wedge post(R)^{(i)}\wedge pre(S'_k)^{(i)}[(x[i])_2/i](\nu)(\sigma_1\{V(f_{l+2})(\nu)(\sigma_1)/h_m\}).$$

This last assertion follows from $<X,\bar{\sigma},u'>\underset{k+k'}{\overset{h\circ h'}{\rightarrow}}<Z,\sigma_0,u'>$.

We conclude $\vdash\{I\wedge pre(R)^{(i)}\}(x:=new(C_k))^{(i)}\{\psi\}$,
$\vdash\{\psi\}(h_m:=h_m\circ<self,x>)^{(i)}\ \{I\wedge post(R)^{(i)}\wedge pre(S'_k)^{(i)}[(x[i])_2/i]\}$.
Finally apply the rule of sequential composition.

This finishes the cooperation test.
We may apply now rule 7: $\vdash\{I\wedge pre(S'_1)^{(i)}[1/i]\}u'\ \{I\wedge\bigwedge_{k=1}^{n}(\forall 1\leqslant i\leqslant C_k post(S'_k)^{(i)})\}$.
It is easy to see that: $\models p'\wedge C_1=1\wedge\bigwedge_{k=2}^{n}C_k=0\rightarrow I\wedge pre(S'_1)^{(i)}[1/i]$ (use axiom 12 of the transition system).
We next prove that:
$\models I\wedge\bigwedge_{k=1}^{n}(\forall 1\leqslant i\leqslant C_k post(S'_k)^{(i)})\rightarrow q$.
Let $\models I\wedge\bigwedge_{k=1}^{n}(\forall 1\leqslant i\leqslant C_k post(S'_k)^{(i)})(\nu)(\sigma)$.
Define: $A=\{<k,l>:1\leqslant k\leqslant n,\ 1\leqslant l\leqslant\sigma(C_k)\}$.
Apply the merging-lemma for $R_\alpha\equiv nil$, $\alpha\in A$.
So for some X, Y, h, k, $\bar{\sigma}$, σ': $<X,\bar{\sigma},u'>\underset{h}{\overset{k}{\rightarrow}}<Y,\sigma',u'>$, ($<X,\bar{\sigma},u'>$ a initial configuration w.r.t. u') $\alpha\in A$ implies $<\alpha,nil>\in Y$ and
$\sigma(x)(\alpha_2)=\sigma'(x)(\alpha_2)$, $x\in var(S'_{\alpha_1})$, furthermore: $\sigma'(C_k)=\sigma(C_k)$, $k=1,\ldots,n$, and $\models p'(\nu)(\bar{\sigma})$.
So $\forall<\alpha,R>\in Y\ R\equiv nil$, in other words $<\bar{\sigma},\sigma'>\in M(u')$.
From $\models\{p\}u'\{q\}$ (which follows from $\models\{p\}u\{q\}$) and $\models p(\nu)(\bar{\sigma})$ we conclude that $\models q(\nu)(\sigma')$.
But σ' agrees with σ w.r.t. the free variables of q ($FV(q)\cap Pvar^{C_i}\subseteq var(S_i)$, $1\leqslant i\leqslant n$). Thus: $\models q(\nu)(\sigma)$. Applying rule 12 and rule 11 gives $\{p'\}u'\{q\}$. Application of rule 13: $\{p'\}u\{q\}$. Application of rule 14: $\{p\}u\{q\}$ (substituting for z_x, $(z_x;[1]:x[1])$, $x\in var(S_1)$, and for h_1 $(h_1;[1]:nil)$).

1. Conclusion.

We have shown in this paper how we can apply the concepts of cooperation test, global invariant, bracketed section and auxiliary variables to the proof-theory of a language containing process creation.
We have proven the proof system to be sound and (relative) complete.
The following remains to be studied:
a formalisation of a more general class of safety properties;
how to prove absence of deadlock.
Another problem is the construction of a proof system which abstracts from the specific mechanism of process identification. Such a proof system is developed for a sequential object-oriented language in [1].

acknowledgement

The idea of the language the proof-theory of which we studied in this paper has been inspired by the language POOL designed by Pierre America. We wish to thank Pierre America, Jaco de Bakker, Joost Kok, John-Jules Meyer, Jan Rutten and Erik de Vink for their part in the discussion of this proof-system.

references

[1] Pierre America, A proof theory for a sequential version of POOL, ESPRIT 415, Doc. No. 188, Philips Research Labs, October 1986.
[2] P.America, Definition of the trogramming language POOL-T, ESPRIT project 415, Doc No. 0091, Philips Research Laboratories, Eindhoven, the Netherlands, June 1985.
[3] P.America, J.W. de Bakker, J.N. Kok, J.J.M.M. Rutten,
Operational semantics of a parallel object-oriented language,
13th ACM symposium on principles of programming Languages, St. Petersburg, Florida, January 13-15, 1986.
[4] K.R.Apt, Ten years of hoare's logic:a survey -
part 1, TOPLAS 3 (4) (1981) 431 -484 .
[5] K.R.Apt, N.Francez, W.P. de Roever, A proof system for CSP,
TOPLAS 2 (3) (1980) 359 -385.
[6] K.R.Apt, Formal justification of a proof system for CSP,
J.ASSOC. COMPUT. MACH., 30 1 (1983) 197 -216.
[7] J.W. de Bakker: Mathematical theory of program correctness.
Prentice-Hall International, 1980.
[8] R. Gerth, A proof system for concurrent ADA programs
in: Science of computer programming 4 (1984) 159-204,
North-Holland.
[9] R.Gerth, W.P.de Roever, M.Roncken, Procedures and concurrency: a study in proof,
in:Lecture notes in computer science;
International Symposium On Programming, nr.137, Turin, April, 1982.
[10] D.Gries, The science of programming,
Springer-Verlag, New York Heidelberg Berlin, 1981.
[11] G.D. Plotkin, A structural approach to operational semantics,
Report DAIMI FN-19, Comp.Sci.dept., Aarhus Univ. 1981.
[12] G.D.Plotkin, An operational semantics for CSP,
in:Formal description of programming concepts 2 (D.Bjorner ed.), North Holland, Amsterdam

(1983) 199-223

[13] J. Zwiers, W.P. de Roever, P. van Emde Boas: Compositionality and concurrent nerworks: soundness and completeness of a proof system. br Proceedings of the 12th International Colloquium on Automata, Languages and Programming (ICALP), Nafplion, Greece, July 15-19, 1985, Springer-Verlag, Lecture Notes in Computer Science, Vol. 194, pp. 509-519.

Appendix 1

LEMMA (merging lemma):
Let for $\alpha \in A$ (A a non-empty finite set of processes) R_α, a normal subprogram, be $(\sigma,\nu,\alpha)-$*reachable* and $\models I(\nu)(\sigma)$ and for $<k,l> \in A$: $1 \leqslant k \leqslant n$, $1 \leqslant l \leqslant \sigma(C_k)$ then:
$\{R_\alpha: \alpha \in A\}$ $(I,\sigma,\nu,A)-$*reachable.*

PROOF :
Let $A = \{<k_1,l_1>, \ldots, <k_m,l_m>\}$.
From $\models I(\nu)(\sigma)$ and the $(\sigma,\nu,\alpha)-$*reachability* of R_α, $\alpha \in A$, follows the existence of the following computations (transitions) (these computations are defined w.r.t. u' as defined on page 17):

$c = <X_0,\sigma_0> \xrightarrow[k]{h} <X,\sigma'>$ according $\models I(\nu)(\sigma)$,

$c_\alpha = <X_0,\sigma_0> \xrightarrow[k_\alpha]{h_\alpha} <Y_\alpha,\sigma_\alpha>$ according R_α $(\sigma,\alpha)-$*reachability*, $\alpha = <k_p,l_p>$, $1 \leqslant p \leqslant m$.

such that:
$\sigma'(h_k)(l) = \sigma_{<k,l>}(h_k)(l)$, $<k,l> \in A$,
$\forall 1 \leqslant m \leqslant \sigma'(C_k)[h]_{<k,m>} =_U \sigma'(h_k)(m)$, $k = 1, \ldots, n$,
$<\alpha,$*before*$(R_\alpha,S_{\alpha_1})> \in X_\alpha$,
$[h_\alpha]_\alpha =_U \sigma_\alpha(h_{\alpha_1})(\alpha_2)$, $\alpha \in A$ (R_α being normal).

To proceed we need the following definitions:
Let B be a finite set of processes.
Given states σ, σ_α, $\alpha \in B$ we define $\sigma\{\sigma_\alpha: \alpha \in B\}$ s.t.:

$$\sigma\{\sigma_\alpha: \alpha \in B\}(x)(n) = \sigma_\alpha(x)(n), \text{ if } x \in var(S'_{\alpha_1}) \text{ and } n = \alpha_2$$
$$= \sigma(x)(n), \text{ otherwise}$$

$$\{\sigma_\alpha: \alpha \in B\}(C) = \sigma(C).$$

Let X be a set of pairs $<\alpha,S>$. Suppose we are given for each $\alpha \in B$ a program R_α.
We define:
$X\{R_\alpha: \alpha \in B\} = \{<\alpha,S>: (\alpha \notin B \wedge <\alpha,S> \in X) \vee (\alpha \in B \wedge S \equiv S_\alpha)\}$,
where

$S_\alpha \equiv R_\alpha$, if $R_\alpha \equiv nil$

$\equiv before(R_\alpha, S_{\alpha_1})$ otherwise

We will show with induction to the length of h, h the history of the above mentioned computation c, that:

$<X_0,\sigma_0> \xrightarrow[k']{h} <X\{R_\alpha: \alpha\in A\}, \sigma'\{\sigma_\alpha: \alpha\in A\}>$., for some k'.

It is easily checked that this suffices.

Basis:the length of h is zero:

In this case the computation c consists only of an activation of S_1, no other processes being activated and so no communications being executed.

We conclude $A = \{<1,1>\}$

($\sigma(C_1)=1$, $\sigma(C_k)=0$, $1<k\leq n$).

$R_{<1,1>}$ is a normal subprogram so:

$nil=[h]_{<1,1>} =_U \sigma'(h_1)(1)=\sigma(h_1)(1)=\sigma_{<1,1>}(h_1)(1)=_U [h_{<1,1>}]_{<1,1>}$.

Thus: $<X_0,\sigma_0> \xrightarrow[k_{<1,1>}]{\epsilon} <X\{R_\alpha: \alpha\in A\}, \sigma'\{\sigma_\alpha: \alpha\in A\}>$ equals $c_{<1,1>}$.

$h=h'\circ<\alpha,\beta>$:

It can be shown that we may assume that (see appendix 2.):

$d_1 = <X_0,\sigma_0> \xrightarrow[k_1]{h'} <X_1,\sigma_1>$,

$d_2 = <X_1,\sigma_1> \xrightarrow[k_2]{<\alpha,\beta>} <X,\sigma'>$, such that

$<\alpha, before(R, S'_{\alpha_1})> \in X_1$,

$R\equiv x:=new(C_{\beta_1})$ and all the k_2 steps are taken by α or β.

We distinguish the following four cases:

1. $\alpha, \beta\notin A$:

   Ind.Hyp.:

   $<X_0,\sigma_0> \xrightarrow[k']{h'} <X_1\{R_\alpha: \alpha\in A\}, \sigma_1\{\sigma_\alpha: \alpha\in A\}>$, for some k'.

   But:

   $<X_1\{R_\alpha: \alpha\in B\}, \sigma_1\{\sigma_\alpha: \alpha\in A\}> \xrightarrow[k_2]{<\alpha,\beta>} <X\{R_\alpha: \alpha\in A\}, \sigma'\{\sigma_\alpha: \alpha\in A\}>$.

   And we are done.

2. $\alpha\in A$, $\beta\notin A$; say $\alpha=<p,q>$:

   Let

   $d_3 = <X_0,\sigma_0> \xrightarrow[k'_1]{h'_1} <Y_1,\sigma_2>$ and

   $d_4 = <Y_1,\sigma_2> \xrightarrow[k'_2]{h'_2} <X_\alpha,\sigma_\alpha>$,

   such that:

   $<\alpha, before(R, S'_p)> \in X_1$, $R\equiv x:=new(C_{\beta_1}); h_p:=h_p\circ<1, self, x>$, $h'_1\circ h'_2 = h_\alpha$,

   and $\sigma_2(h_p)(q)=\sigma_1(h_p)(q)$.

   Ind.Hyp.:

   $<X_0,\sigma_0> \xrightarrow[k']{h'} <X_1\{R'_\alpha: \alpha\in A\}, \sigma_1\{\sigma'_\alpha: \alpha\in A\}>$, for some k'.

   Where

   $R'_\gamma \equiv R_\gamma$, $\gamma\neq\alpha$

$\equiv R$, otherwise.

$\sigma'_\gamma = \sigma_\gamma$, $\gamma \neq \alpha$

$= \sigma_2$, otherwise.

Now proceed from this last configuration by executing α along the lines of d_4 and β of d_2.
Note that because $[h_\alpha]_\alpha =_U \sigma_\alpha(h_p)(q)$ (R_α being a normal program) $[h'_2]_\alpha = \ll \alpha, \beta \gg$.

3. $\alpha \notin A$, $\beta \in A$:
Ind.Hyp.:
$<X_0, \sigma_0> \underset{k'}{\overset{h'}{\rightarrow}} <X_1\{R_\alpha: \alpha \in A / \{\beta\}\}, \sigma_1\{\sigma_\alpha: \alpha \in A / \{\beta\}\}>$, for some k'.
Proceed from this last configuration by executing α along the lines of d_2 and β of c_β.
Note that $nil = [h]_\beta =_U \sigma'(h_{\beta_1})(\beta_2) = \sigma_\beta(h_{\beta_1})(\beta_2) =_U [h_\beta]_\beta$, because R_β is normal.

4. $\alpha, \beta \in A$.:
Similar to case 2.

$h = h' \circ <d, \alpha, \beta>$:
Let (see appendix 2.):

$$d_1 = <X_0, \sigma_0> \underset{k_1}{\overset{h'}{\rightarrow}} <X_1, \sigma_1>,$$

$$d_2 = <X_1, \sigma_1> \underset{k_2}{\overset{<d,\alpha,\beta>}{\rightarrow}} <X, \sigma'>,$$

such that computation d_2 consists only of steps taken by α or β, and in $<X_1, \sigma_1>$ α, β are to enter the bracketed sections execution of which consists of the transfer of the value d from α to β.
We, again , distinguish four cases:

1. $\alpha, \beta \notin A$: apply the same reasoning as used in the above mentioned case 1..
2. $\alpha \in A$, $\beta \notin A$: say: $\alpha = <p, q>$.
Let

$$d_3 = <X_0, \sigma_0> \underset{k'_1}{\overset{h'_1}{\rightarrow}} <Y_1, \sigma_2>,$$

$$d_4 = <Y_1, \sigma_2> \underset{k'_2}{\overset{h'_2}{\rightarrow}} <X_\alpha, \sigma_\alpha>.$$

where,
$<\alpha, before(R, S'_p)> \in Y_1$, R, a bracketed section corresponding with the communication record $<d, \alpha, nil>$ or $<d, \alpha, \beta>$, depending on whether the last record of $\sigma'(h_p)(q)$ equals $<d, \alpha, nil>$ or $<d, \alpha, \beta>$,
furthermore, $\sigma_2(h_p)(q) = \sigma_1(h_p)(q)$, and $h'_1 \circ h'_2 = h_\alpha$.
Ind.Hyp.:
$<X_0, \sigma_0> \underset{k'}{\overset{h'}{\rightarrow}} <X_1\{R'_\alpha: \alpha \in A\}, \sigma_1\{\sigma'_\alpha: \alpha \in A\}>$.
Where

$R'_\gamma \equiv R_\gamma$, $\gamma \neq \alpha$

$\equiv R$, otherwise.

$\sigma'_\gamma = \sigma_\gamma$, $\gamma \neq \alpha$

$= \sigma_2$, otherwise.

Proceed from this last configuration by first performing the communication between α and β and

after that executing α along the lines of d_4 and β of d_2.

3. $\alpha \in A$, $\beta \notin A$:
   analoguosly to the previous case.
4. α, $\beta \in A$ (let $\alpha = <p,q>$, $<\beta = <r,s>$):
   Let

$$d_3 = <X_0,\sigma_0> \xrightarrow[k'_1]{h'_1} <Y_1,\sigma_2>,$$

$$d_4 = <Y_1,\sigma_2> \xrightarrow[k'_2]{h'_2} <X_\alpha,\sigma_\alpha>.$$

where,
$<\alpha, before(R,S'_p)> \in Y_1$, R, a bracketed section corresponding with the communication record $<d,\alpha,nil>$ or $<d,\alpha,\beta>$, depending on whether the last record of $\sigma'(h_p)(q)$ equals $<d,\alpha,nil>$ or $<d,\alpha,\beta>$,
furthermore, $\sigma_2(h_p)(q) = \sigma_1(h_p)(q)$, and $h'_1 \circ h'_2 = h_\alpha$,
and

$$d_5 = <X_0,\sigma_0> \xrightarrow[\bar{k}_1]{\bar{h}_1} <\bar{Y}_1,\bar{\sigma}_2>,$$

$$d_6 = <\bar{Y}_1,\bar{\sigma}_2> \xrightarrow[\bar{k}_2]{\bar{h}_2} <X_\beta,\sigma_\beta>.$$

where,
$<\beta, before(R',S'_r) \in \bar{Y}_1$, R', a bracketed section corresponding with the communication record $<d,nil,\beta>$ or $<d,\alpha,\beta>$, depending on whether the last record of $\sigma'(h_r)(s)$ equals $<d,nil,\beta>$ or $<d,\alpha,\beta>$,
furthermore, $\bar{\sigma}_2(h_r)(s) = \sigma_1(h_r)(s)$, and $\bar{h}_1 \circ \bar{h}_2 = h_\beta$.
Ind.Hyp.:
$<X_0,\sigma_0> \xrightarrow[k']{h'} <X_1\{R'_\alpha : \alpha \in A\}, \sigma_1\{\sigma'_\alpha : \alpha \in A\}>$. Where

$$R'_\gamma \equiv R_\gamma, \gamma \notin \{\alpha,\beta\}$$
$$\equiv R, \ \gamma = \alpha$$
$$\equiv R', \ \gamma = \beta$$
$$\sigma'_\gamma = \sigma_\gamma, \ \gamma \notin \{\alpha,\beta\}$$
$$= \sigma_2, \ \gamma = \alpha$$
$$= \bar{\sigma}_2, \ \gamma = \beta$$

Now proceed from this last configuration by first transfering d from α to β and then executing α along the lines of d_4 and β of d_6.

Appendix 2

In this sextion we will write $<X,\sigma>\overset{h}{\rightarrow}<X',\sigma'>$ in stead of $<X,\sigma>\underset{k}{\overset{h}{\rightarrow}}<X',\sigma'>$ for some k.

LEMMA :

Let $<X_0,\sigma_0>\overset{h}{\rightarrow}<X,\sigma>$ be a computation (transition) w.r.t. a bracketed unit $u\equiv<C_1\leftarrow\underset{h'}{S_1},\ldots,C_n\leftarrow S_n>$, $h=h'\circ<\alpha,\beta>$, then there exists four computations:

$<X_0,\sigma_0>\rightarrow <X_1,\sigma_1>$,

$<X_1,\sigma_1>\overset{<\alpha,\beta>}{\rightarrow} <X_2,\sigma_2>$,

$<X_2,\sigma_2>\overset{\epsilon}{\rightarrow} <X_3,\sigma_3>$,

$<X_3,\sigma_3>\overset{\epsilon}{\rightarrow} <X,\sigma>$

such that $<\alpha,before(R,S_{\alpha_1})>\in X_1$, α is about to enter the backeted section execution of which consists of the activation of β, $<\alpha,after(R,S_{\alpha_1})>\in X_2$, and in the second and third computation only α is performing, in the last one only β is performing.

PROOF :

Let $<X_0,\sigma_0>\overset{h'}{\rightarrow} <X',\sigma'>\overset{<\alpha,\beta>}{\rightarrow} <\overline{X},\overline{\sigma}>\overset{\epsilon}{\rightarrow} <X,\sigma>$,
such that $<\alpha,before(R,S_{\alpha_1})>\in X'$ and $<\alpha,after(R,S_{\alpha_1})\in\overline{X}$.
Let $X_1=X/\{<\gamma,S>:<\gamma,S>\in X\wedge\gamma\in\{\alpha,\beta\}\}\cup\{<\alpha,before(R,S_{\alpha_1})>\}$,
σ_1 such that:

$$\sigma_1(x)(m)=\sigma'(x)(m),\ \text{if}\ (x\in var(S_{\alpha_1})\wedge m=\alpha_2)\vee(x\in var(S_{\beta_1})\wedge m=\beta_2)$$

$$=\sigma(x)(m)\ \text{otherwise}$$

$$\sigma_1(C)=\sigma'(C)$$

Then: $<X',\sigma'>\overset{\epsilon}{\rightarrow} <X_1,\sigma_1>$,

because from $<X',\sigma'>$ to $<X,\sigma>$ all the processes but α are involved in some internal action, formally: induction to k, where $<X',\sigma'> \overset{<\alpha,\beta>}{\underset{k}{\rightarrow}} <X,\sigma>$.

Thus: $<X_0,\sigma_0> \overset{h'}{\rightarrow} <X_1,\sigma_1>$.

Let $X_2 = X_1 / \{<\alpha, before(R,S_{\alpha_1})>\} \cup \{<\alpha, after(R,S_{\alpha_1})>, <\beta,S_{\beta_1}>\}$,

σ_2 such that:

$$\sigma_2(x)(m) = \omega, \text{ if } x \in var(S_{\beta_1}) \wedge m = \beta_2$$

$$= \bar{\sigma}(x)(m), \text{ if } x \in var(S_{\alpha_1}) \wedge m = \alpha_2$$

$$= \sigma_1(x)(m) \text{ otherwise}$$

$$\sigma_2(C) = \sigma(C)$$

Then: $<X_1,\sigma_1> \overset{<\alpha,\beta>}{\rightarrow} <X_2,\sigma_2>$,

because executing α starting from $<X_1,\sigma_1>$ along the lines of $<X',\sigma'> \overset{<\alpha,\beta>}{\rightarrow} <\bar{X},\bar{\sigma}>$ results in $<X_2,\sigma_2>$,

formally: induction to k, where $<X',\sigma'> \overset{<\alpha,\beta>}{\underset{k}{\rightarrow}} <\bar{X},\bar{\sigma}>$.

Let

$X_3 = X_2 / \{<\alpha, after(R,S_{\alpha_1})>\} \cup \{<\alpha,S>: <\alpha,S> \in X\}$ and

σ_3 such that:

$$\sigma_3(x)(m) = \sigma(x), \text{ if } x \in var(S_{\alpha_1}) \wedge m = \alpha_2$$

$$= \sigma_2(x)(m), \text{ otherwise}$$

$$\sigma_3(C) = \sigma(C).$$

Then: $<X_2,\sigma_2> \overset{\epsilon}{\rightarrow} <X_3,\sigma_3> \overset{\epsilon}{\rightarrow} <X,\sigma>$,

because from $<X_2,\sigma_2>$ to $<X_3,\sigma_3>$ resp. $<X_3,\sigma_3>$ to $<X,\sigma>$ one gets by executing α resp. β along the lines of $<\bar{X},\bar{\sigma}> \overset{\epsilon}{\rightarrow} <X,\sigma>$, during which both are involved in some internal action.

LEMMA :

Let $<X_0,\sigma_0> \overset{h}{\rightarrow} <X,\sigma>$ be a computation w.r.t. a bracketed unit u, $h = h' \circ <d,\alpha,\beta>$, then there exists the following computations:

$<X_0,\sigma_0> \overset{h'}{\rightarrow} <X_1,\sigma_1>$,

$<X_1,\sigma_1> \overset{<d,\alpha,\beta>}{\rightarrow} <X_2,\sigma_2>$,

$<X_2,\sigma_2> \overset{\epsilon}{\rightarrow} <X_3,\sigma_3>$,

$<X_3,\sigma_3> \overset{\epsilon}{\rightarrow} <X,\sigma>$,

such that

$before(R,S_{\alpha_1})$, $before(R',S_{\beta_1}) \in X_1$, α, β are to enter the bracketed sections execution of which consists the tansfer of d from α to β,

and $after(R,S_{\alpha_1})$, $after(R',S_{\beta_1}) \in X_2$,

and, finaly, from $<X_2,\sigma_2>$ to $<X_3,\sigma_3>$ only α is performing and from $<X_3,\sigma_3>$ to $<X,\sigma>$ only β is performing.

PROOF :
Analoguosly to the proof of the previous lemma.

Appendix 3

Let $u \equiv <C_1 \leftarrow S_1, \ldots, C_n \leftarrow S_n>$, p, q such that $\models \{p\}u\{q\}$ and $FV(p,q) \cap Pvar \subseteq \bigcup_{1 \leq i \leq n} Pvar^{C_i}$.

Let $Y_i = \{x_1^i, \ldots, x_{k_i}^i\} = Var(S_i) \cup (FV(p,q) \cap Pvar^{C_i})$, $1 \leq i \leq n$.

We assume some coding of our programming language: for statement S , $\overline{S}$ denotes the natural number encoding S.

Let p_n denote the n^{th} prime number.

We define for $n \in \mathbf{N}$, $\overline{n} = p_1^{n+1}$, and for $<n,m> \in \mathbf{N} \times \mathbf{N}$, $\overline{<n,m>} = p_1^{n+1} \times p_2^{m+1}$.

Furthermore: $\overline{<\alpha_1, \ldots, \alpha_n>} = \prod_{i=1}^{n} p_i^{\overline{\alpha_i}}$, $\alpha_i \in \mathbf{D} \cup \{\omega\}$,

where $\overline{\omega} = 0$.

Next we define for state σ, $\overline{\sigma} \in \mathbf{N}$:

$$\overline{\sigma} = \prod_{i=1}^{n} p_i^{a_i} \times \prod_{i=1}^{n} p_{n+i}^{b_i}$$

where:

$$a_i = \prod_{m=1}^{k_i} p_m^{a^{i,m}} \text{ and}$$

$$a^{i,m} = \prod_{l=1}^{k} p_l^{a_l^{i,m}},\ k = \sigma(C_i) \text{ and}$$

$$a_l^{i,m} = \overline{\sigma(<x_m^i, l>)} \text{ and}$$

$$b_i = \sigma(C_i).$$

We define for every configuration $<X, \sigma>$:

$$<X, \sigma> = \overline{\sigma} \times \prod_{i=1}^{n} p_{2n+i}^{d_i}$$

where

$$d_i = \prod_{j=1}^{k} p_j^{e_j},\ k = \sigma(C_i),\ e_j = \overline{S},\ s.t.\ <<i,j>, S> \in X$$

Now let $h \in \{<\alpha,\beta>, <\alpha,\beta,\gamma> : \alpha, \beta, \gamma \in D\}^*$.
We define:

$$h = \epsilon \rightarrow \bar{h} = 1$$

$$h = h_1 \circ <<\alpha,\beta>> \rightarrow \bar{h} = \overline{h_1} \times p_{k+2}^{\bar{\alpha}} \times p_{k+3}^{\bar{\beta}}$$

$$h = h_1 \circ <<\alpha,\beta,\gamma>> \rightarrow \bar{h} = \overline{h_1} \times p_{k+1} \times p_{k+2}^{\bar{\alpha}} \times p_{k+3}^{\bar{\beta}} \times p_{k+4}^{\bar{\gamma}}$$

Where $k = \max\{n : p_n \text{ divides } \bar{h}\}$ ($\max(\varnothing) = 0$).
Next we give our basic definition:

DEFINITION :
transition(n_1, n_2, n_3, n_4) iff
$n_1 = \overline{<X_1,\sigma_1>}$, $n_2 = \overline{<X_2,\sigma_2>}$, $n_3 = \bar{h}$, $<X_1,\sigma_1> \xrightarrow[n_4]{h} <X_2,\sigma_2>$ for some configurations $<X_1,\sigma_1>$, $X_2,\sigma_2>$ and history h.
LEMMA :
Transition is a recursive predicate and thus representable in Peano Arithmetic.
PROOF : omitted (straightforward , but tedious)

To proceed we need the following list of predicates which are easily shown to be recursive.

- *init*(n) iff $n = \overline{<X,\sigma>}$, $<X,\sigma>$ a initial configuration (w.r.t. u).
- *state*$(n) = m$ iff $n = \overline{<X,\sigma>}$, $m = \bar{\sigma}$.
- *int*(n) iff $n = p_1^m$, $m > 0$.
- *proc*(n) iff $n = p_1^k \times p_2^l$, $k,l > 0$.
- $proc_1(n) = m$ iff $(n = p_1^k \times p_2^l \wedge k,l > 0 \wedge m = p_1^k)$
$\vee (\neg proc(n) \wedge m = 0)$
- $proc_2(n) = m$ iff $(n = p_1^k \times p_2^l \wedge k,l > 0 \wedge m = p_1^l)$
$\vee (\neg proc(n) \wedge m = 0)$
- $mkint(n) = m$ iff $m = p_1^{n+1}$
- $mkproc(k,l) = m$ iff $(m = p_1^{k'} \times \underline{p_2^{l'}} \wedge k = p_1^{k'} \wedge l = p_1^{l'} \wedge k',k' > 0) \vee (m = 0 \wedge \neg(int(k) \wedge int(l))$
- $proj(k,l,m) = n$ iff $k = \bar{h} \wedge n = \underline{[h]_{<l,m>}}$, for some $h \in \{<\alpha,\beta>, <\alpha,\beta,\gamma> : \alpha, \beta, \gamma \in D\}^*$
- $|n| = m$ iff $m = \max\{k : p_k \text{ divides } n\}$
- $elt(k,l) = m$ iff $m = max\{n : p_l^n \text{ divides } k\}$
- $m \oplus n = k$ iff $(m = p_1^{m_1} \wedge n = p_1^{n_1} \wedge m_1, n_1 > 0 \wedge k = p_1^{m_1+n_1-1})$
$\vee (\neg(int(m) \wedge int(n)) \wedge k = 0)$
- $elem(n,m) = k$ iff $(m = p_1^{m'+1} \wedge 1 \leq m' \leq |n| \wedge elt(n,m') = k)$
$\vee ((\neg int(m) \vee (m = p_1^{m'+1} \wedge (m' > |n| \vee m' = 0))) \wedge k = 0)$.
- $newarray(k,l,m) = n$ iff $(l = p_1^{l'+1} \wedge l' \geq 1 \wedge \forall i \neq l'\ elt(k,i) = elt(n.i) \wedge elt(n,l') = m)$
$\vee (\neg int(l) \wedge k = n)$.
- $before(k,l) = m$ iff $k = \overline{S_1}$, $l = \overline{S_2}$, $m = \overline{S_3}$, $before(S_1,S_2) = S_3$
- $after(k,l) = m$ iff $k = \overline{S_1}$, $l = \overline{S_2}$, $m = \overline{S_3}$ $after(S_1,S_2) = S_3$.
- $conc(k,l) = m$ iff $|m| = |k| + |l| \wedge \forall 1 \leq i \leq |k|\ elt(k,i) = elt(m,i) \wedge$
$\forall |k| < i \leq |m|\ elt(l,i) = elt(m,i))$
- $n =_U m$ iff $n = \overline{h_1} \wedge m = \overline{h_2} \wedge h_1 =_U h_2$,
for some $h \in \{<\alpha,\beta>, <\alpha,\beta,\gamma> : \alpha, \beta, \gamma \in D\}^*$.

We call a formula arithmetical if all the terms occurring in it are built from logical variables and arithmetical operations only.
We are going to associate with each term f and formula p such that $FV(f,p) \subseteq \bigcup_{1 \leq i \leq n} Y_i$ a arithmetical

term $\overline{f(z)}$ and formula $\overline{p}(z)$ such that for arbitrary ν, σ $\models p(\nu)(\sigma)$ iff $\models \overline{p}(z)(\nu\{\overline{\sigma}/z\})$ and $\overline{V(f)(\nu)(\sigma)} = V(\overline{f}(z))(\nu\{\overline{\sigma}/z\})$, where we define
$\overline{g} = \prod_{i=1}^{k} p_i^{\overline{g(i)}}$, for $g \in \mathbf{N}^+ \to \mathbf{D} \cup \{\omega\}$, such that $\forall j \geq k\ g(j) = \omega$.

- $\overline{n} \equiv p_1^{n+1}$
- $\overline{It_1 + It_2} \equiv \overline{It_1} \oplus \overline{It_2}$
- $\overline{(Pt)_1}, \overline{(Pt)_2} \equiv \overline{proc_1(Pt)}, \overline{proc_2(Pt)}$
- $\overline{At[It]} \equiv elem(\overline{At}, \overline{It})$
- $\overline{ATt[It]} \equiv elem(\overline{ATt}, \overline{It})$
- $\overline{C_k} \equiv elt(z, n+k)$
- $i \in lvar,\ \overline{i} \equiv p_1^{i+1}$
- $\overline{<It_1, It_2>} \equiv mkproc(\overline{It_1}, \overline{It_2})$
- $\overline{nil} \equiv 0$
- $\overline{x_j^i} \equiv elt(elt(z,i),j)$
- $\overline{(At;[It]:Dt)} \equiv newarray(\overline{At}, \overline{It}, \overline{Dt})$
- $\overline{Tt_1 \circ Tt_2} \equiv conc(\overline{Tt_1}, \overline{Tt_2})$
- $\overline{Tt[It]} \equiv elem(\overline{Tt}, \overline{It})$
- $\overline{(ATt;[It]:Dt)} \equiv newarray(\overline{ATt}, \overline{It}, \overline{Dt})$
- $\overline{<Dt_1, \ldots, Dt_m>} \equiv \prod_{i=1}^{m} p_i^{\overline{Dt_i}}$

An easy induction on the complexity of the term f shows that $\overline{V(f)(\nu)(\sigma)} = V(\overline{f}(z)(\nu\{\overline{\sigma}/z\})$, for arbitrary ν and σ.

This translation of terms is extended to formulae in the natural way:
for example, $\overline{It_1 = It_2} \equiv \overline{It_1} = \overline{It_2}$,
$\overline{It_1 \leq It_2} \equiv int(\overline{It_1}) \wedge int(\overline{It_2}) \wedge \overline{It_1} \leq \overline{It_2}$,
$\overline{\psi_1 \wedge \psi_2} \equiv \overline{\psi_1} \wedge \overline{\psi_2}$.
Again a straightforward induction to the complexity of p shows that for arbitrary ν and σ $\models p(\nu)(\sigma)$ iff $\models \overline{p}(z)(\nu\{\overline{\sigma}/z\})$.

Before we are able to give formulations in the assertion language of the assertions I, $pre(R)$, $post(R)$, we need the following definition:

DEFINITION
Let f_1 be a data term ($f_1 \in Dt$) and f_2 a integer term ($f_2 \in It$) then
$f_1 \doteq f_2 \equiv f_1 = nil \to f_2 = 0 \wedge \forall i,j (f_1 = <i,j> \to f_2 = <i,j> \wedge f_1 = i \to f_2 = i)$.
In case f_1 is a trace term ($f_1 \in Tt$)
$f_1 \doteq f_2 \equiv |f_1| = |f_2| \wedge \forall 1 \leq i \leq |f_1|\, f_1[i] \doteq elt(f_2, i)$.

DEFINITION :
I can be formulated as follows:

$$\exists n_1, n_2, n_3, n_4 (transition(n_1,n_2,n_3,n_4) \wedge init(n_1) \wedge \overline{p'}[state(n_1)/z] \wedge$$

$$\bigwedge_{x_j^i \in F'} x_j^i[1] \doteq elt(elt(elt(state(n_2),i),j),1) \wedge \bigwedge_{x_j^i \in F} \forall 1 \leq k \leq C_i x_j^i[k] \doteq elt(elt(elt(state(n_2),i),j),k) \wedge$$

$$\bigwedge_{1 \leq k \leq n} C_k = elt(state(n_2), n+k) \wedge$$

$$\bigwedge_{1 \leq k \leq n} \forall 1 \leq i \leq C_k \exists m (h_k[i] \doteq m \wedge m = {}_U proj(n_3,k,i))$$

where $F=\{h_1, \ldots, h_n\}$ and $F'=\{z_x: x \in var(S_1)\}$.

DEFINITION :

Pre(R) can be formulated as follows: (we assume that R occurs in the k^{th} component of U)

$$\exists n_1, n_2, n_3, n_4(transition(n_1,n_2,n_3,n_4) \wedge init(n_1) \wedge \overline{p'}[state(n_1)/z] \wedge$$

$$before(\overline{R},\overline{S_k}) = elt(elt(n_2,k+2n),i) \wedge \bigwedge_{x_j^k \in var(S'_k)} x_j^k[i] \doteq elt(elt(elt(state(n_2),k),j),i) \wedge$$

$$\bigwedge_{x_j^l \in F'} x_j^l[1] \doteq elt(elt(elt(state(n_2),l),j),1))$$

where $F'=\{z_x: x \in var(S_1)$.

post(R) is defined analogously.

Appendix 4

An example: a parallel prime generator.
This program is a translation, modification of a program written in the language POOL by L. Augusteijn.
The original program generates all prime numbers and is thus a non-terminating program. We have modified the original program to make it terminating, that is we introduced a variable m such that the program generates all prime numbers not greater than m. Let $u \equiv <C_1 \leftarrow S_1, C_2 \leftarrow S_2>$ where

S_1:
$c := new(C_2)$
; $n := 2$
; *do* $n \leq m \rightarrow c!n$; $n := n+1$ *od*
; $c!0$

S_2:
$?p$;
if $p \neq 0 \rightarrow$ $next := new(C_2)$
; *do* $q \neq 0 \rightarrow$ $?q$
; if q *MOD* $p \neq 0 \vee q = 0 \rightarrow$ $next!q$
otherwise $\rightarrow$ *skip*
fi
od
$p = 0 \rightarrow$ *skip*
fi

We want to prove the following partial correctness assertion:
$\{m[1] \geq 2\} u \{\forall 1 \leq i < C_2\ p[i] = '\ i^{th}\ prime\ number' \wedge C_2 - 1 = |\{p : prime(p) \wedge p \leq m[1]\}|\}$.
(for X a set $|X|$ denotes the cardinality of X.)
We extend u to a annotated and bracketed unit u' as follows:

S'_1:
$\{\phi_0\}$
$<c := new(C_2)$
;*bool* := *false*
;$n := 2>$

$\{\phi_1\}$
;*do* $n \leqslant m \rightarrow$ $\{\phi_2\}$
$<c!n;n:=n+1>$
$\{\phi_1\}$
od
$\{\phi_3\}$
; $<c!0>$
$\{\phi_4\}$

Where,
1. $\phi_0 \equiv bool[i]=nil \wedge m[i] \geqslant 2$,
2. $\phi_1 \equiv c[i]=<2,1> \wedge n[i] \leqslant m[i]+1 \wedge bool[i] \neq nil$,
3. $\phi_2 \equiv n[i] \leqslant m[i] \wedge c[i]=<2,1> \wedge bool[i] \neq nil$,
4. $\phi_3 \equiv n[i]=m[i]+1 \wedge c[i]=<2,1> \wedge bool[i] \neq nil$,
5. $\phi_4 \equiv n[i]=m[i]+1 \wedge bool[i] \neq nil$.

S'_2:
$\{\psi_0\}$
$<?p>$;
$\{\psi_1\}$
if $p \neq 0 \rightarrow$ $\{\psi_2\}$
$<next:=new(C_2)$
;$last:=false>$
$\{\psi_3\}$
; *do* $q \neq 0 \rightarrow$ $\{\psi_3\}$
$<?q;r:=true>$
$\{\psi_4\}$
; if q *mod* $p \neq 0 \vee q=0 \rightarrow$ $\{\psi_5\}$
$<next!q;r:=false>$
$\{\psi_3\}$
otherwise $\rightarrow$ $\{\psi_3\}skip\{\psi_3\}$
fi
$\{\psi_3\}$
od
$\{\psi_6\}$
$p=0 \rightarrow$ $\{\psi_6\}skip\{\psi_6\}$
fi
$\{\psi_6\}$

Where,
1. $\psi_0 \equiv last[i]=r[i]=p[i]=nil$,
2. $\psi_1 \equiv last[i]=r[i]=nil \wedge (p[i]=0 \vee p[i]=i^{th}$ *prime number*$)$
3. $\psi_2 \equiv last[i]=r[i]=nil \wedge p[i]=i^{th}$ *prime number*
4. $\psi_3 \equiv r[i]=true \rightarrow \neg(prime(q[i]) \vee q[i]=0) \wedge next[i]=<2,i+1> \wedge last[i]=false$,
5. $\psi_4 \equiv next[i]=<2,i+1> \wedge r[i]=true \wedge last[i]=false$,
6. $\psi_5 \equiv (p[i] \nmid q[i] \vee q[i]=0) \wedge next[i]=<2,i+1> \wedge last[i]=false \wedge r[i]=true$,
7. $\psi_6 \equiv r[i] \neq true \wedge last[i]=nil \rightarrow p[i]=0$.

We define for X a finite set of prime numbers $NP(X)$ the least prime number greater than all the numbers of X. Note that $NP(\varnothing)=2$.
Let $n \lhd NP(X)$ iff $n \leqslant NP(X)$ and $\forall x \in X\ n>x$.
Next we define our global invariant I.

1. $\forall 1 \leq i \leq C_2 (p[i] = i^{th}\ prime\ number \vee (i = C_2 \wedge (p[i] = nil \vee p[i] = 0)))$,
2. $\forall 1 \leq i \leq C_2 (r[i] = true \wedge q[i] \neq 0 \rightarrow \forall 1 \leq j < i\ p[j] \nmid q[i])$,
3. $\forall 1 \leq i \leq C_2 (r[i] = true \wedge q[i] \neq 0 \rightarrow q[i] \lhd NP(\{p[k]: 1 \leq k \leq C_2 \wedge prime(p[k])\} \cup \{q[k]: i < k \leq C_2 \wedge r[k] = true \wedge prime(q[k])\})$,
4. $bool[1] \neq nil \rightarrow n[1] \lhd NP(\{p[i]: 1 \leq i \leq C_2 \wedge prime(p[i])\} \cup \{q[i]: 1 \leq i \leq C_2 \wedge r[i] = true \wedge prime(q[i])\})$,
5. $bool[1] = nil \rightarrow C_2 = 0$,
6. $C_1 = 1$,
7. $\forall 1 \leq i \leq C_2 (last[i] = nil \Leftrightarrow i = C_2)$.

I is the conjunction of the formulas 1, . . . , 7.
Next we have to prove that these proof outlines cooperate. We have selected the following two cases, the others are left to the reader to verify.
We prove that:
$\{I \wedge \phi_0\}(c := new(C_2); bool := false; n := 2)^{(i)} \{I \wedge \phi_1\}$.

PROOF :
a. $I \wedge \phi_0 \wedge 0 < i \leq C_1 \rightarrow C_1 = 1 \wedge C_2 = 0 \wedge i = 1 \wedge m[i] \geq 2$,
b. $C_1 = 1 \wedge C_2 = 0 \wedge i = 1 \wedge m[i] \geq 2 \rightarrow$
$C_1 = 1 \wedge C_2 + 1 = 1 \wedge i = 1 \wedge m[i] \geq 2 \wedge (c;[i]: <2, C_2 + 1>)[i] = <2, C_2 + 1> \wedge \bigwedge_{x \in var(S'_2)} (x;[C_2 + 1]; nil)[C_2 + 1] = nil$
c. $\{I \wedge \phi_0 \wedge 0 < i \leq C_2\}$
$(c := new(C_2))^{(i)}$
$\{C_1 = 1 \wedge C_2 = 1 \wedge i = 1 \wedge m[i] \geq 2 \wedge c[i] = <2,1> \wedge \bigwedge_{x \in var(S'_2)} x[1] = nil\} (\equiv \chi)$ (application of A4 and R4),
d. $\chi \rightarrow \chi \wedge (n;[i]:2)[i] = 2 \wedge (bool;[i]; false)[i] = false$
e. $\{\chi\}(bool := false; n := 2)^{(i)} \{\chi \wedge bool[i] = false \wedge n[i] = 2\}$
(application of A1 and R3),
f. $\{I \wedge \phi_0 \wedge 0 < i \leq C_2\}(c : new(C_2); bool := false; n := 2)^{(i)} \{\chi \wedge bool[i] = false \wedge n[i] = 2\}$
(application of R3),
g. $\chi \wedge bool[i] = false \wedge n[i] = 2 \rightarrow I \wedge \phi_1$
h. $\{I \wedge \phi_0\}(c := new(C_2); bool := false; n := 2)^{(i)} \{I \wedge \phi_1\}$
(application of R4).

Next we prove that:
$\{I \wedge \psi_5 \wedge \psi_3[j / i]\}\ (next!q; r := false)^{(i)} \| (?q; r := true)^{(j)}\ \{I \wedge \psi_3 \wedge \psi_4[j / i]\}$.

PROOF :
Let $\psi_i \equiv last[i] = false$ and $\psi_j \equiv next[j] = <2, j + 1> \wedge last[j] = false$.
a. $I \wedge \psi_5 \wedge \psi_3[j / i] \wedge 0 < i \leq C_2 \wedge 0 < j \leq C_2 \wedge next[i] = <2, j> \rightarrow$
$I[(q;[j]: q[i]) / q, ((r;[i]: false);[j]: true) / r] \wedge \psi_i \wedge \psi_j)$
b. $\{I \wedge \psi_5 \wedge \psi_3[j / i]\}\ (next!q)^{(i)} \| (?q)^{(j)}\ \{I[((r;[i]: false);[j]: true) / r)] \wedge \psi_i \wedge \psi_j\}$
(application of A3 and R7)
c. $\{I[((r;[i]: false);[j]: true) / r] \wedge \psi_i \wedge \psi_j\}\ (r := false)^{(i)}\ \{I[(r;[j]: true) / r] \wedge \psi_3 \wedge \psi_j\}$
(application of A1),
d. $\{I[(r;[j]: true) / r] \wedge \psi_3 \wedge \psi_j\}\ (r := true)^{(j)}\ \{I \wedge \psi_3 \wedge \psi_4[j / i]\}$.
e. One application of R9 finishes the proof.

Remark:the truth of the implication mentioned in clause a. follows from the following observation; if $r[i + 1] = true$ then $q[i + 1]$ is not a prime number so

$NP(\{p[k]:1 \leqslant k \leqslant C_2 \wedge p[k] \neq nil\} \cup \{q[k]:i < k \leqslant C_2 \wedge r[k] = true \wedge prime(q[k])\})$
equals $NP(\{\cdots\} \cup \{q[k]:i+1 < k \leqslant C_2 \wedge r[k] = true \wedge prime(q[k])\})$.
End of remark.

Applying rule 5. yields:
$\{I \wedge \phi_0[1/i]\}u'\ \{I \wedge \forall 1 \leqslant i \leqslant C_1 \phi_4 \wedge \forall 1 \leqslant i \leqslant C_2 \psi_6\}$.
Now the following implications can be shown to hold:
1. $C_1 = 1 \wedge C_2 = 0 \wedge \phi_0[1/i] \rightarrow I \wedge \phi_0[1/i]$
2. $I \wedge \forall 1 \leqslant i \leqslant C_1 \phi_4 \wedge \forall 1 \leqslant i \leqslant C_2 \psi_6 \rightarrow$
$\forall 1 \leqslant i < C_2 p[i] = i^{th}\ prime\ \ number \wedge C_2 - 1 = |\{p : prime(p) \wedge p \leqslant m[1]\}|$.

Applying the rules 11, 12, 13 and 14 in that order gives us the desired result.