THE CONSISTENCY OF THE CALCULUS OF TOTAL CORRECTNESS FOR COMMUNICATING PROCESSES

by

Zhou Chaochen

Technical Monograph PRG-26
February 1982

Oxford University Computing Laboratory,
Programming Research Group,
45, Banbury Road,
OXFORD. OX2 6PE

Institute of Computing Technology,
Chinese Academy of Sciences,
Peking, China.

In [1] Hoare suggests a notation of assertions to describe the total correctness of communicating processes, and a calculus for proving it. But the question of the consistency of the calculus is left open in that paper. In this paper we give an operational model of communicating processes and present two variants of the calculus, which are consistent with this model. One of them cannot deal with livelock, while the other one does.

Contents

Introduction

This paper proves the consistency of two versions of a calculus presented in [1]. This calculus deals with correctness formulas of the form (P sat R), where P is a process and R an assertion.

A process is constructed from a group of subprocesses, intercommunicating on a network of named channels, and communicates with its environment by sending and receiving messages on named channels too. Each communication is denoted as a pair "c.m" where "m" is the value of the message and "c" is the name of the channel along which it passes. In [1], past sequences and ready sets are introduced to describe the behaviour of processes, where past is a finite sequence of communications recording all communications in which a process has engaged up to any given moment, and ready is a set of communications which a process is prepared to communicate at any given moment. The pairs of past and ready are used to denote the states of processes. (s,X) is called a state of the process P, if P may engage in the communications of s, and if, just after doing s, P can accept any communication in X.

For example:

(a) the only possible state of the inactive process STOP is $(\langle\rangle, \emptyset)$. Because STOP never communicates at any moment, the possible record of its history is the empty sequence of communications, and the acceptable communications at any moment constitute an empty set as well.

(b) If P first sends the message e on the channel c, then stops, i.e. P = c!e $\longrightarrow$ STOP. The possible states of P are:

$(\langle\rangle, \{c.e\})$ —— the beginning state of P. At this moment, P has not engaged in any communication, but is prepared to communicate message e on channel c.

$(c.e,\emptyset)$ —— the state after P sends e on c. P will not accept any more communication in future.

(c) If P repeats the sending of message e on channel c for ever, i.e. $P = c!e \longrightarrow P$. The possible states of P are:

$(<>,\{c.e\})$ —— the beginning etate of P.

$(c.e,\{c.e\})$ —— the state after the first sending of e on c.

$(<c.e>^n,\{c.e\})$ —— the state after the nth sending of e on c, where a^n stands for $\underbrace{a^\frown a \ldots ^\frown a}_{n}$.

(d) If P first receives a message of M on channel c, or a message of N on the channel d, then stops, i.e.

$P = c?x{:}M \longrightarrow STOP \;[]\; d?y{:}N \longrightarrow STOP$. The possible states of P are:

$(<>,c.M \cup d.N)$ —— the beginning state of P. P is prepared to receive any message of M on channel c, or any message of N on channel d, where $c.M = \{c.x \mid x \in M\}$

$(c.x,\emptyset)$ —— the state after P receives the message x of M on the channel c.

$(d.y,\emptyset)$ —— the state after P receives the message y of N on the channel d.

The assertions for total correctness of processes suggested in [1] are the predicates of channel variables - "channelname.past" or "channelname.ready". "c.past" stands for a message sequence recording all messages passing along the channel c up to some moment. It can be obtained from "past" by cancelling all communications along the other channels, then dropping the channel name c to get a pure message sequence,

e.g. $c.(\langle c.e\rangle^n) = e^n$,

and $c.(c.e_1 \hat{} d.e_2) = e_1$.

"c.ready" stands for the set of messages which are ready to be passed along the channel c at some moment,

e.g. $c.(c.M \cup d.N) = M \qquad (c \neq d)$.

Thus an assertion defines a set of states:

$$\text{state-set of } R = \left\{(s,X) \,\middle|\, R\left[{}^{s}/_{past}\right]\left[{}^{X}/_{ready}\right]\right\}.$$

For example,

state-set of false = $\emptyset$

state-set of true = the universe of states

$$= \left\{(s,X) \,\middle|\, s \in \Sigma^* \;\&\; X \subseteq \Sigma\right\}.$$

where Σ is the set of all communications.

Let P be a process and R be an assertion. Then P is said to <u>satisfy</u> R (abbreviated by P <u>sat</u> R) if the assertion R correctly describes the states of P, i.e. all the states of P constitute a subset of the state-set of R.

For example,

if $P = c!e \longrightarrow P$, then from (c) we can see

P <u>sat</u> $(c.\text{ready} = \{e\})$.

It implies that P never stops, i.e. P is deadlock-free.

[1], [2] and [3] are recommended to the reader who is interested in the capability of channel predicates for specifying the behaviour of processes. We shall not review them here.

Actually the calculus in [1] is not concerned with the unstable state of process. A process normally changes its states by taking communications with its environment. E.g. in the previous example (b), P changes its beginning state $(<>, \{c.e\})$ to state $(c.e, \emptyset)$ by passing value e on channel c to its environment. However, under certain circumstances a process itself, without participation the environment can also change its states. This phenomenon is called internal transition. Internal transition emerges either from internal communications (communications on hidden channels) or from non-deterministic operators (say, nondeterministic union). The hidden channels of a process are only for the communications between its subprocesses, invisibly to the environment. The nondeterministic union of the processes P and Q, P or Q, behaves either like P or like Q, but the choice between them is wholly nondeterministic, and may be made autonomously by the process, or by its implementor. So the choice cannot be influenced or even observed by the environment. In other words (P or Q) can transform itself autonomously either to P or to Q.

The internal transition may cause an unstable state. A state of a process is said to be unstable, if there is an internal transition which may happen at that moment, and an internal transition will usually bring the process to a new and different state.

E.g. consider the beginning state of $((c!e \longrightarrow P)$ or $(d!m \longrightarrow Q))$, at which the procese may do internal choice, transforming itself to either $(c!e \longrightarrow P)$ or $(d!m \longrightarrow Q)$.

An unstable state may be unreliable. If the environment wants to communicate on c with the process $((c!e \longrightarrow P)$ or $(d!m \longrightarrow Q))$, then sometimes it may succeed, but sometimes it may fail depending on which alternative has been chosen by the process.

Moreover, in most cases, the internal transition can take place at a very high speed, compared with the external communication. This is due to the high speed of modern electronics. So we can suppose that the unstable state is not only unreliable, but also evanescent.

Thus it is claimed in [1] that the unstable states are not taken into account by the calculus. (P sat R) is interpreted more precisely as: "any stable state of P belongs to the state-set of R".

Another problem which can be caused by internal transition is livelock. A process is said to be livelocked if after some moment during its evolution the process may engage in an infinite sequence of internal transitions, and ignore possible external communications completely. We usually wish to prove absence of livelock.

In the following sections we shall give an operational model of communicating processes, and present formal definitions of the previous concepts such as stable state, livelock, etc. Then two calculi, Calculus- and Calculus+, will be developed. Calculus- is simpler, but does not prove absence of livelock. Calculus+ is a little more complicated, but it has the ability to treat livelock. Both are consistent with the operational model in the sense that their rules are theorems in this model.

1. Operational Model

We adopt nearly the same syntax of communicating processes as in [1], which includes the constructs: output, input, nondeterministic union, alternation, channel renaming, parallelism, hiding and recursion. But, instead of the informal descriptions, we are taking a formal

system of the transitions of processes to be their semantics, known as operational semantics, which was proposed by Robin Milner to define the semantics of communicating systems in [4].

Transition is a binary relation over processes. Transition results from an action taken by the process. An action may be observable (external communication ranging over Σ), or it may be unobservable (internal communication or internal choice). Let P and Q be processes. We use

$$P \xrightarrow{c.m} Q$$

to denote "P is transformed to Q under the observable action c.m", where $c.m \in \Sigma$

and

$$P \xrightarrow{\tau} Q$$

to denote "P transforms itself unobservably to Q", where τ is a special symbol not in Σ.

1.1 STOP

STOP is an inactive process, so STOP has no transition.

1.2 Output

If P is a process and $c.e \in \Sigma$, then $(c!e \longrightarrow P)$ is a process, which first outputs e on channel c and then behaves like P. We use the following axiom to define its semantics:

$$(c!e \longrightarrow P) \xrightarrow{c.e} P .$$

This means that the only possible transition of $(c!e \longrightarrow P)$ results from the communication c.e, and transforms $(c!e \longrightarrow P)$ to P.

1.3 Input

Let $P(x)$ be a process for each x in M, and let $c.M$ be a subset of Σ. Then $(c?x{:}M \longrightarrow P(x))$ is a process, which first inputs any message x of M on channel c, and then behaves like $P(x)$.

The only axioms which its transitions obey are:

$$(c?x{:}M \longrightarrow P(x)) \xrightarrow{c.x} P(x) \qquad (x \in M).$$

1.4 (Nondeterministic) Union

If P and Q are processes, $(P \underline{\text{or}}\ Q)$ is a process which behaves either like P or like Q. The choice between them is autonomously made by the process itself. This statement can be axiomatised by

$$(P \underline{\text{or}}\ Q) \xrightarrow{\tau} P ,$$

and $$(P \underline{\text{or}}\ Q) \xrightarrow{\tau} Q .$$

1.5 Alternation

Let $P(x)$ and $Q(y)$ be processes for each x in M and y in N. Let c and d be distinct channel names, and let $(c.M \cup d.N) \subseteq \Sigma$.

Then $(c?x{:}M \longrightarrow P(x) \,[]\, d?y{:}N \longrightarrow Q(y))$ is a process which is prepared to input either on channel c or channel d, and then behaves like $P(x)$ or $Q(y)$ respectively, where x stands for the message just input on channel c and y for the message just input on channel d.

The axioms which define the possible transitions of alternation should be:

$$(c?x{:}M \longrightarrow P(x) \,[]\, d?y{:}N \longrightarrow Q(y)) \xrightarrow{c.x} P(x) \qquad (x \in M)$$

and $$(c?x{:}M \longrightarrow P(x) \,[]\, d?y{:}N \longrightarrow Q(y)) \xrightarrow{d.y} Q(y) \qquad (y \in N) .$$

1.6 Channel Renaming

Let P be a process, and c be a channel, and let d be a channel not in P. Then $P\left[{}^{d}/_{c}\right]$ is a process which behaves exactly like P, except that whenever P usee channel c for communicating, $P\left[{}^{d}/_{c}\right]$ uses channel d instead.

We adopt the following inference rule to define the semantics of $P\left[{}^{d}/_{c}\right]$, because the transition of $P\left[{}^{d}/_{c}\right]$ takes transitions of P to be its premiss.

Let α be an action (external or internal), i.e. $\alpha \in \sum \cup \{\tau\}$, and let $ch(\alpha)$ be the channel name of α :

$$ch(c.m) =_{df} c$$

and $$ch(\tau) =_{df} \tau .$$

$$\frac{P \xrightarrow{\alpha} Q}{P\left[{}^{d}/_{c}\right] \xrightarrow{\alpha} Q\left[{}^{d}/_{c}\right]} \qquad (ch(\alpha) \neq c)$$

and $$\frac{P \xrightarrow{c.m} Q}{P\left[{}^{d}/_{c}\right] \xrightarrow{d.m} Q\left[{}^{d}/_{c}\right]} .$$

1.7 Parallelism

Let A be the set of channel namee occurring in process P, and let B be the set of channel names occurring in process Q. Then $(P \underset{A}{|}\underset{B}{|} Q)$ denotes a network constructed from processes P and Q, which are connected to each other by common channels of A and B. Thus the actions taken by P on channels of $(A \cap B)$ must be synchronized with

the corresponding actions of Q, while the actions taken by P either internally or externally on channels (A-B) can still progress independently of Q.

The actions of Q are similarly constrained.

The inference rules for the semantics of $(P \underset{AB}{||} Q)$ are:

$$\frac{P \xrightarrow{\alpha} P_1}{(P \underset{AB}{||} Q) \xrightarrow{\alpha} (P_1 \underset{AB}{||} Q)} \qquad ch(\alpha) \in (A-B) \cup \{\tau\} \quad ,$$

$$\frac{Q \xrightarrow{\alpha} Q_1}{(P \underset{AB}{||} Q) \xrightarrow{\alpha} (P \underset{AB}{||} Q_1)} \qquad ch(\alpha) \in (B-A) \cup \{\tau\}$$

and $$\frac{P \xrightarrow{\alpha} P_1 \ , \ Q \xrightarrow{\alpha} Q_1}{(P \underset{AB}{||} Q) \xrightarrow{\alpha} (P_1 \underset{AB}{||} Q_1)} \qquad ch(\alpha) \in (A \cap B) \quad .$$

1.8 Hiding

Let b be a channel intended by process P for internal communication. Then (chan b in P) is a process in which all communications along channel b are invisible to the environment.

It can be formulated as follows:

$$\frac{P \xrightarrow{\alpha} Q}{(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P) \xrightarrow{\alpha} (\underline{\text{chan}}\ b\ \underline{\text{in}}\ Q)} \qquad ch(\alpha) \neq b \quad ,$$

and $$\frac{P \xrightarrow{b.m} Q}{(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P \xrightarrow{\tau} (\underline{\text{chan}}\ b\ \underline{\text{in}}\ Q))} \quad .$$

1.9 Recursion

A single recursion is of the form of

$$p \triangleq F(p) \ ,$$

where p is a process variable and $F(p)$ is an expression constructed by p and the previous eight operators. We also use the notation $(\mu p.F)$ (the least fixed point of F) to denote the process defined by recursion $p \triangleq F(p)$. As is well known, $\mu p.F$ must behave exactly like $F(\mu p.F)$. The transitions of $\mu p.F$, since we have given the semantics of expression F with the structural rules (1.1)-(1.8), can be defined by the transitions of $F(\mu p.F)$, i.e.

$$\frac{F(\mu p.F) \xrightarrow{\alpha} Q}{\mu p.F \xrightarrow{\alpha} Q} \ .$$

E.g. (a) Expression $F(p) = (c!e \longrightarrow p)$.

Since $F(\mu p.F) = (c!e \longrightarrow \mu p.F)$

and $(c!e \longrightarrow \mu p.F) \xrightarrow{c.e} \mu p.F$ (1.2) ,

therefore $\mu p.F \xrightarrow{c.e} \mu p.F$ (1.9) .

(b) Expression $I(p) = p$.

Since $I(\mu p.I) = \mu p.I$,

therefore by (1.9) $\mu p.I \xrightarrow{\alpha} Q$ iff $\mu p.I \xrightarrow{\alpha} Q$.

So we cannot derive any transition of $\mu p.I$ by (1.9), and we cannot derive it by (1.1)-(1.8) either. Thus $\mu p.I$ is an inactive process.

Let a mutual recursion take the form

$$p[i:N] \triangleq F(p) \ ,$$

where p and F are arrays. Then the array of processes, $\mu p[i:N].F$, defined by the mutual recursion obeys the following laws:

$$\frac{(F(\mu p[1:N].F))[j] \xrightarrow{\alpha} Q}{(\mu p[1:N].F)[j] \xrightarrow{\alpha} Q} \qquad (j \in N) \ .$$

A formal deduction syetem for traneitions has been established by (1.1)-(1.9) by which transitions of a complex process can be derived from the traneitions of its simpler componente. A transition $P \xrightarrow{\alpha} Q$ ie true iff it is a last line of a deduction in the system. A sequence of transitione (which has length n) generally takes the form

$$P_0 \xrightarrow{\alpha_1} P_1 \xrightarrow{\alpha_2} P_2 \xrightarrow{\alpha_3} \ \ldots \ \xrightarrow{\alpha_n} P_n \ .$$

We shall usually write it as $P_0 \xrightarrow{\alpha_1 \hat{} \alpha_2 \hat{} \ldots \hat{} \alpha_n} P_n$ and introduce a new inference rule for transition eequences:

1.10 $$\frac{P \xrightarrow{s} Q, \ Q \xrightarrow{t} R}{P \xrightarrow{s \hat{} t} R} \ ,$$

where s and t are nonempty finite sequences of $\Sigma \cup \{\tau\}$.

2. Stable States

We proceed to formally describe stable states of processes. A state of a procees consists of an action sequence in which the process has been engaged up to some moment in time, and an action set which the procees is prepared to take at eome given moment.

Each action of a process is described by a transition. So an action sequence of a process corresponds to a transition sequence of the process, and an action set which a proceee is ready to take corresponds to a set of transitions of the process. In other words, let P be a process, let s be a (nonempty) finite sequence of $\Sigma \cup \{\tau\}$,

and let $X \subseteq (\Sigma \cup \{\tau\})$. Then s is an action sequence of P iff $\exists Q.P \xrightarrow{s} Q$, and X is a ready set of P iff $X = \{\alpha \mid \exists Q.P \xrightarrow{\alpha} Q\}$. But, as is well known, we are not concerned with the unobservable action τ in an action sequence, and we are also not concerned with the unstable state at which a process can progress unobservably to another state.

We add two axioms to the system for defining observable action sequence, where $P \overset{s}{\Longrightarrow} Q$ means "P can be transformed to Q under the observable action sequence s ($s \in \Sigma^*$)".

1.11 $P \overset{\langle\rangle}{\Longrightarrow} P$

This axiom says that before P takes any action, P does not change. Thue the empty sequence can be used to record the observable action sequence which a process has taken before it evolves.

1.12 $$\frac{P \xrightarrow{s} Q}{P \overset{s\backslash\tau}{\Longrightarrow} Q}\ ,$$

where $s\backslash\tau$ is obtained from s by cancelling τ. This rule implies that an observable action sequence is simply obtained from an action sequence by ignoring the unobservable action τ.

Now we can precisely define the stable state.

Let P be a process, P^o denote the ready set of P, and st(P) be the stable-state-set of P. Then

$$P^o =_{df} \{\alpha \mid \exists Q.P \xrightarrow{\alpha} Q\},$$

and $$st(P) =_{df} \{(s, Q^o) \mid \exists Q.P \overset{s}{\Longrightarrow} Q \ \&\ \tau \tilde{\in} Q^o\}.$$

This definition is exactly a formulization of the fact that (s,X) is a stable state of P iff s is an observable action sequence of P, and <u>after</u> doing s, P can be stably prepared to take the actions X.

We now show that the stable state of a process can be calculated by structural formulas.

Lemma 1.

(1) $\mathrm{STOP} \overset{s}{\Longrightarrow} Q$ iff $s = \langle\rangle$ & $Q = \mathrm{STOP}$

(2) $(c!e \longrightarrow P) \overset{s}{\Longrightarrow} Q$ iff $s = \langle\rangle$ & $Q = (c!e \longrightarrow P)$
$\vee\ \exists s_1.\ s = c.e\hat{}s_1$ & $P \overset{s_1}{\Longrightarrow} Q$.

(3) $(c?x{:}M \longrightarrow P(x)) \overset{s}{\Longrightarrow} Q$ iff $s = \langle\rangle$ & $Q = (c?x{:}M \longrightarrow P(x))$
$\vee\ \exists x{\in}M, s_1.\ s = c.x\hat{}s_1$ & $P(x) \overset{s_1}{\Longrightarrow} Q$.

(4) $(P \ \underline{\mathrm{or}}\ Q) \overset{s}{\Longrightarrow} R$ iff $s = \langle\rangle$ & $R = (P \ \underline{\mathrm{or}}\ Q)$
$\vee\ P \overset{s}{\Longrightarrow} R \ \vee\ Q \overset{s}{\Longrightarrow} R$.

(5) $(c?x{:}M \longrightarrow P(x) \ \square\ d?y{:}N \longrightarrow Q(y)) \overset{s}{\Longrightarrow} R$
iff $s = \langle\rangle$ & $R = (c?x{:}M \longrightarrow P(x) \ \square\ d?y{:}N \longrightarrow Q(y))$
$\vee\ \exists x{\in}M,\ y{\in}N,\ s_1.\ s = c.x\hat{}s_1$ & $P(x) \overset{s_1}{\Longrightarrow} R$
$\vee\ s = d.y\hat{}s_1$ & $Q(y) \overset{s_1}{\Longrightarrow} R$.

(6) $P\left[{}^{d}/_{c}\right] \overset{s}{\Longrightarrow} Q$ iff $\exists P_1, s_1.\ P \overset{s_1}{\Longrightarrow} P_1$ & $Q = P_1\left[{}^{d}/_{c}\right]$
& $s = s_1\left[{}^{d}/_{c}\right]$.

(7) $(P \underset{AB}{||} Q) \overset{s}{\Longrightarrow} R$ iff $s \in (A \cup B)^{*}$ & $\exists P_1, Q_1.\ P \overset{s\restriction A}{\Longrightarrow} P_1$
& $Q \overset{s\restriction B}{\Longrightarrow} Q_1$ & $R = (P_1 \underset{A}{|} \underset{B}{|} Q_1)$,

where $(A \cup B)^{*}$ stands for the set of finite communication sequences along channels in $A \cup B$, and $s\restriction A$ for the sequence obtained from s by cancelling the communications not along channels in A.

(8) $(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P) \xrightarrow{s} Q$ iff $\exists s_1, P_1.\ s = s_1\backslash b\ \&\ P \xrightarrow{s_1} P_1$

$\&\ Q = (\underline{\text{chan}}\ b\ \underline{\text{in}}\ P_1)$,

where $s\backslash b$ is obtained from s by cancelling the communications on channel b.

(9) $(\mu p[i:N].F)[j] \xrightarrow{s} Q$ iff $(F(\mu p[i:N].F))[j] \xrightarrow{s} Q \quad (j \in N)$,

provided $s \neq \langle\rangle$ or $(Q \neq (\mu p[i:N].F[j]\ \&\ Q \neq (F(\mu p[i:N].F))[j])$.

proof. We only give a part of the proof for parallelism.

$(\Longrightarrow)$ Suppose $(P \underset{AB}{||} Q) \xrightarrow{s} R$.

The axioms from which this may be derived are (1.11) and (1.12).

Then case (1) $s = \langle\rangle\ \&\ R = (P \underset{AB}{||} Q)$ by (1.11).

Let $P_1 = P$ and $Q_1 = Q$.

Since $\langle\rangle\restriction A = \langle\rangle$ and $\langle\rangle\restriction B = \langle\rangle$, $P \xrightarrow{\langle\rangle\restriction A} P_1$ and $Q \xrightarrow{\langle\rangle\restriction B} Q_1$ by (1.11).

Thus RHS of (7) holds.

case (2) $\exists s_1.\ s = s_1\backslash\tau\ \&\ (P \underset{AB}{||} Q) \xrightarrow{s_1} R$ by (1.12).

We prove RHS of (7) by induction on s_1.

When $s_1 = \alpha$, $(P \underset{AB}{||} Q) \xrightarrow{s_1} R$ can be derived only by the rules in (1.7).

Subcases:

(a) $R = (P_1 \underset{AB}{||} Q)\ \&\ P \xrightarrow{\alpha} P_1\ \&\ ch(\alpha) \in (A-B) \cup \{\tau\}$.

Let $Q_1 = Q$. Since $\alpha\restriction B = \langle\rangle$ and $\alpha\restriction A = \alpha\backslash\tau$,

$P \xrightarrow{\alpha\restriction A} P_1$ by (1.12) and $Q \xrightarrow{\alpha\restriction B} Q_1$ by (1.11).

So RHS holds.

(b) $R = (P \underset{AB}{||} Q_1)$ & $Q \xrightarrow{\alpha} Q_1$ & $ch(\alpha) \in (B-A) \cup \{\tau\}$.

Let $P_1 = P$. Then RHS holds.

(c) $R = (P_1 \underset{AB}{||} Q_1)$ & $P \xrightarrow{\alpha} P_1$ & $Q \xrightarrow{\alpha} Q_1$ & $ch(\alpha) \in (A \cap B)$.

Since $\alpha \restriction A = \alpha \restriction B = \alpha = \alpha \backslash \tau$, $P \overset{\alpha \restriction A}{\Longrightarrow} P_1$ and $Q \overset{\alpha \restriction B}{\Longrightarrow} Q_1$ by (1.12). So RHS holds.

Now for $s_1 = t^\frown \alpha$, let $(P \underset{AB}{||} Q) \xrightarrow{t} T$ and $T \xrightarrow{\alpha} R$.

Then $t \backslash \tau \in (A \cup B)^*$ and $\exists P_2, Q_2.\ P \overset{t \restriction A}{\Longrightarrow} P_2$ & $Q \overset{t \restriction B}{\Longrightarrow} Q_2$

& $T = (P_2 \underset{AB}{||} Q_2)$ by induction.

From $(P_2 \underset{AB}{||} Q_2) \xrightarrow{\alpha} R$, repeating (a) (b) and (c), we can get P_1 and Q_1 such that

$$P_2 \overset{\alpha \restriction A}{\Longrightarrow} P_1 \quad \& \quad Q_2 \overset{\alpha \restriction B}{\Longrightarrow} Q_1 \quad \& \quad R = (P_1 \underset{AB}{||} Q_1).$$

Hence RHS holds.

Lemma 2.

(1) $(STOP)^\circ = \emptyset$.

(2) $(c!e \longrightarrow P)^\circ = \{c.e\}$.

(3) $(c?x:M \longrightarrow P(x))^\circ = c.M$.

(4) $(P \ \underline{or}\ Q)^\circ = \{\tau\}$.

(5) $(c?x:M \longrightarrow P(x) \ [] \ d?y:N \longrightarrow Q(y))^\circ = c.M \cup d.N$.

(6) $(P[{}^{d}/_{c}])^\circ = P^\circ[{}^{d}/_{c}]$.

(7) $(P \underset{AB}{||} Q)^\circ = (P^\circ \restriction A \cap Q^\circ \restriction B) \cup P^\circ \restriction ((A-B) \cup \{\tau\}) \cup Q^\circ \restriction ((B-A) \cup \{\tau\})$,

where $P^\circ \restriction A$ stands for the set obtained from P° by including only communications on channels in A.

(8) $(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P)^\circ = P^\circ\left[\tau/b\right]$.

(9) $((\mu p[i:N].F)[j])^\circ = ((F(\mu p[i:N].F))[j])^\circ \qquad (j \in N)$.

proof. We give the proof only for (7).

$(P \underset{AB}{||} Q)^\circ = \left\{ \alpha \;\middle|\; \exists R.\ (P \underset{AB}{||} Q) \xrightarrow{\alpha} R \right\}$ by definition.

Since $(P \underset{AB}{||} Q) \xrightarrow{\alpha} R$ can only be derived by the rules in (1.7), and

(a) $(\exists P_1.P \xrightarrow{\alpha} P_1)\ \&\ ch(\alpha) \in (A-B) \cup \{\tau\}$.

iff $\alpha \in P^\circ \restriction (A-B) \cup \{\tau\}$. (by definition of P°).

(b) $(\exists Q_1.Q \xrightarrow{\alpha} Q_1)\ \&\ ch(\alpha) \in (B-A) \cup \{\tau\}$.

iff $\alpha \in Q^\circ \restriction (B-A) \cup \{\tau\}$. (by definition of Q°)

(c) $(\exists P_1, Q_1\ \ P \xrightarrow{\alpha} P_1\ \&\ Q \xrightarrow{\alpha} Q_1)\ \&\ ch(\alpha) \in (A \cap B)$

iff $\alpha \in P^\circ \restriction (A \cap B)\ \&\ \alpha \in Q^\circ \restriction (A \cap B)$,

i.e. $\alpha \in (P^\circ \restriction A \cap Q^\circ \restriction B)$,

therefore (7) is proved.

Theorems.

2.1 $st(STOP) = \{(<>, \emptyset)\}$.

2.2 $st(c!e \longrightarrow P) = \{(<>, \{c.e\})\} \cup \{(c.e\hat{}s, X) \mid (s,X) \in st(P)\}$.

2.3 $st(c?x:M \longrightarrow P(x)) = \{(<>, c.M)\} \cup \{(c.x\hat{}s, X) \mid x \in M\ \&\ (s,X) \in st(P(x))\}$.

2.4 $st(P\ \underline{\text{or}}\ Q) = st(P) \cup st(Q)$.

2.5 $st(c?x:M \longrightarrow P(x) \,[]\, d?y:N \longrightarrow Q(y))$

$= \{(<>, c.M \cup d.N)\}$

$\cup \{(c.x\hat{}s, X) \mid x \in M\ \&\ (s,X) \in st(P(x))\}$

$\cup \{(d.y\hat{}s, X) \mid y \in N\ \&\ (s,X) \in st(Q(y))\}$.

2.6 $st(P[^{d}/_{c}]) = st(P)[^{d}/_{c}]$.

2.7 $st(P_{AB}||Q) = \{(s, (X\restriction A \cap Y\restriction B) \cup X\restriction(A-B) \cup Y\restriction(B-A)) \mid s \in (A \cup B)^{*} \& (s\restriction A, X) \in st(P) \& (s\restriction B, Y) \in st(Q)\}$.

2.8 $st(\underline{chan}\ b\ \underline{in}\ P) = \{(s\backslash b, X) \mid (s,X) \in st(P) \& X\restriction\{b\} = \emptyset\}$.

2.9 $\forall j \in N, n.\ st((\mu p[i:N].F)[j]) = st((F^{n}(\mu p[i:N].F))[j])$.

proof. We give the proofs of (2.7) and (2.9).

(2.7) $(P_{AB}||Q) \overset{s}{\Longrightarrow} R$ iff $s \in (A \cup B)^{*}$ & $\exists P_1, Q_1.\ P \overset{s\restriction A}{\Longrightarrow} P_1$ & $Q \overset{s\restriction B}{\Longrightarrow} Q_1$ & $R = (P_1||_{AB}Q_1)$ (by Lemma 1 (7)).

and $\tau\tilde{\epsilon}(P_1||_{AB}Q_1)^{\circ}$ iff $\tau\tilde{\epsilon}P_1^{\circ}$ & $\tau\tilde{\epsilon}Q_1^{\circ}$ (by Lemma 2 (7)).

But $(s,R^{\circ}) \in st(P_{AB}||Q)$ iff $(P_{AB}||Q) \overset{s}{\Longrightarrow} R$ & $\tau\tilde{\epsilon}R^{\circ}$ (by def. of stable state)

hence $(s,R^{\circ}) \in st(P_{AB}||Q)$ iff there exists P_1 and Q_1 such that

$$s \in (A \cup B)^{*} \ \&\ P \overset{s\restriction A}{\Longrightarrow} P_1 \ \&\ Q \overset{s\restriction B}{\Longrightarrow} Q_1 \ \&\ \tau\tilde{\epsilon}P_1^{\circ} \ \&\ \tau\tilde{\epsilon}Q_1^{\circ} \ \&\ R = (P_1||_{AB}Q_1),$$

i.e. $s \in (A \cup B)^{*}$ & $(s\restriction A, P_1^{\circ}) \in st(P)$ & $(s\restriction B, Q_1^{\circ}) \in st(Q)$ & $R = (P_1||_{AB}Q_1)$ (by def. of stable state)

Moreover $H^{\circ} = (P_1||_{AB}Q_1)^{\circ} = (P_1^{\circ}\restriction A \cap Q_1^{\circ}\restriction B) \cup P_1^{\circ}\restriction(A-B) \cup Q_1^{\circ}\restriction(B-A)$

(since Lemma 2 (7) and $(\tau\tilde{\epsilon}P_1^{\circ}$ & $\tau\tilde{\epsilon}Q_1^{\circ})$), the proof is completed.

(2.9) Let us just take the case of single recursion:

$$\forall n.\ st(\mu p.F) = st(F^n(\mu p.F)) ,$$

which we prove from a stronger conclusion: $\mu p.F$ and $F^n(\mu p.F)$ not only have the same stable states, but also have the same unstable states.

Let ast(P) be the set of all states (stable and unstable) of process P.

$$ast(P) =_{df} \left\{ (s,Q^o) \;\middle|\; P \stackrel{s}{\Longrightarrow} Q \right\}.$$

Then we can show:

(a) $ast(\mu p.F) = ast(F(\mu p.F))$ by Lemma 1 (9) and Lemma 2 (9); and

(b) for any process expression H(p) and processes P and Q,

$$(ast(P) = ast(Q)) \Longrightarrow (ast(H(P)) = ast(H(Q))),$$

which can be proved by routine induction on the structure of H.

Thus $ast(\mu p.F) = ast(F(\mu p.F))$ by (a),

so $ast(F(\mu p.F)) = ast(F^2(\mu p.F))$ by (b), and so on.

3. Calculus-

We now present the weaker calculus, named Calculus-, in which livelock is not taken into account. Calculus- is nearly the same as that given in [1], with the exception of the inference rule for hiding. Recall that the assertion which we use to specify the behaviour of a process is a predicate with channel variables - "channelname.past" and "channelname.ready", which are intended to describe the stable states.

Let P be a process, and let R be a channel predicate.
Then "P satisfies R" means that all stable states of P satisfy R.
Formally

$$(P \underline{\text{sat}}\ R) =_{df} \forall(s,X) \in st(P).\ R\left[s/_{past}\right]\left[X/_{ready}\right] \quad \text{----} \quad (I-)$$

where c.s is a message sequence obtained from s by cancelling all communications not on channel c, and then dropping channel name c from the resulting sequence; and c.X is a message set obtained from X in the same way.

In what follows $R\left[s/_{past}\right]\left[X/_{ready}\right]$ is abbreviated as Rs,X.

We list the inference rules of Calculus- followed by the proofs of their consistency; i.e. every rule is a theorem under the interpretation (I-) of (P sat R).

3.1 STOP

$$R_{<>,\emptyset} \equiv (\text{STOP}\ \underline{\text{sat}}\ R)$$

proof.
$$(\text{STOP}\ \underline{\text{sat}}\ R) \equiv \forall(s,X) \in st(P).\ Rs,X \qquad (\text{def.})$$
$$\equiv \forall(s,X) \in \{(<>,\emptyset)\}.Rs,X \qquad (2.1)$$
$$\equiv R_{<>,\emptyset}$$

3.2 Output

$$R_{<>}\left[\{e\}/_{c.ready}\right]_{\emptyset} \quad \& \quad (P\ \underline{\text{sat}}\ R\left[e\,\hat{}\,c.past/_{c.past}\right])$$
$$\equiv (c!e \longrightarrow P)\ \underline{\text{sat}}\ R$$

where $R_{<>}\left[\{e\}/_{c.ready}\right]\emptyset$ stands for

$$R\left[<>/_{past}\right]\left[\{e\}/_{c.ready}\right]\left[\emptyset/_{ready}\right].$$

proof. RHS $\equiv \forall(s,X) \in st(c!e \longrightarrow P)\ .\ Rs,X$ (def.)

$\equiv \forall(s,X) \in (\{(\langle\rangle, \{c.e\})\} \cup \{(c.e\hat{}s, X) \mid (s,X) \in st(P)\}).\ Rs,X$ (2.2)

$\equiv R_{\langle\rangle,\{c.e\}}\ \&\ \forall(s,X) \in st(P).\ R_{c.e\hat{}s,X}$

$\equiv R_{\langle\rangle}\left[{}^{\{e\}}/_{c.ready}\right]_{\emptyset}\ \&\ (P\ \underline{sat}\ R\left[{}^{e\hat{}c.past}/_{past}\right])$

(since $c.\{c.e\} = \{e\}$ and $d.\{c.e\} = \emptyset$ if $d \neq c$, and $c.(c.e\hat{}s) = e\hat{}c.s$ and $d.(c.e\hat{}s) = d.s$ if $d \neq c$.)

3.3 Input

$$R_{\langle\rangle}\left[{}^{M}/_{c.ready}\right]_{\emptyset}\ \&\ \forall x \in M.\ (P(x)\ \underline{sat}\ R\left[{}^{x\hat{}c.past}/_{c.past}\right])$$

$$\equiv (c?x{:}M \longrightarrow P(x))\ \underline{sat}\ R,$$

provided R does not contain x as a free variable.

proof. Omitted.

3.4 Union

$$(P\ \underline{sat}\ R)\ \&\ (Q\ \underline{sat}\ R) \equiv (P\ \underline{or}\ Q)\ \underline{sat}\ R.$$

proof. RHS $\equiv \forall(s,X) \in st(P\ \underline{or}\ Q).\ Rs,X$ (def.)

$\equiv \forall(s,X) \in (st(P) \cup st(Q)).\ Rs,X$ (2.4)

$\equiv (\forall(s,X) \in st(P).Rs,X)\ \&\ (\forall(s,x) \in st(Q).Rs,X)$

$\equiv (P\ \underline{sat}\ R)\ \&\ (Q\ \underline{sat}\ R)$ (def.)

3.5 Alternation

$$R_{<>}\left[{}^{M}/_{c.ready}\right] \quad \left[{}^{N}/_{d.ready}\right]_{\emptyset}$$

$$\& \ \forall x \in M.\ P(x) \ \underline{sat}\ R\left[{}^{x \hat{} c.past}/_{c.past}\right]$$

$$\& \ \forall y \in N.\ Q(y) \ \underline{sat}\ R\left[{}^{y \hat{} d.past}/_{d.past}\right]$$

$$\equiv\ (c?x{:}M \longrightarrow P(x) \ [\!] \ d?y{:}N \longrightarrow Q(y)) \ \underline{sat}\ R,$$

provided $c \neq d$ and R does not contain x or y freely.

proof. Omitted.

3.6 Renaming

$$(P \ \underline{sat}\ R\left[{}^{c}/_{d}\right]\ ) \equiv (P\left[{}^{d}/_{c}\right] \ \underline{sat}\ R)\ ,$$

provided that R does not mention c, and d does not occur in P.

proof. Omitted.

3.7 Parallelism

If S only mentions channel names of A

and R only mentions channel names of B,

and $A \cap B = \{c_1, c_2 \ldots, c_n\}$, then

$$(P \ \underline{sat}\ S) \ \& \ (Q \ \underline{sat}\ R) \Longrightarrow (P \underset{AB}{||} Q) \ \underline{sat}$$

$$\exists_{i=1}^{n} X_i, Y_i.\ \ S\left[{}^{X_i}/_{c_i.ready}\right]_{i=1}^{n} \ \& \ R\left[{}^{Y_i}/_{c_i.ready}\right]_{i=1}^{n}$$

$$\&_{i=1}^{n} c_i.ready = X_i \cap Y_i \ .$$

proof. LHS $\equiv \forall (s,X) \in st(P).\ S_{s,X}\ \&\ \forall (s,Y) \in st(Q).R_{s,Y}$ (def.)

$\Rightarrow \forall (s\restriction A,X) \in st(P).\ S_{s,X}\ \&\ \forall (s\restriction B,Y) \in st(P).R_{s,Y}$

(since $S_{s,X} \equiv S_{s\restriction A,X}$ and $R_{s,Y} \equiv R_{s\restriction B,Y}$

by the assumptions of S and R)

$\Rightarrow \forall s, X, Y.\ (s\restriction A,X) \in st(P)\ \&\ (s\restriction B,Y) \in st(Q)$

$\Rightarrow S_{s,X}\ \&\ R_{s,Y}$

$\Rightarrow \forall s, X, Y, Z.\ (s\restriction A,X) \in st(P)\ \&\ (s\restriction B,Y) \in st(Q)$

$\&\ Z = (X\restriction A \cap Y\restriction B) \cup X\restriction(A-B) \cup Y\restriction(B-A)$

$\Rightarrow S_{s,Z\cup X\restriction(A\cap B)}\ \&\ R_{s,Z\cup Y\restriction(A\cap B)}$

(since $(Z \cup X\restriction(A\cap B))\restriction A = X\restriction A$, $(Z\cup Y\restriction(A\cap B))\restriction B = Y\restriction B$

and $S_{s,X} \equiv S_{s,X\restriction A}$ and $R_{s,Y} \equiv R_{s,Y\restriction B}$)

$\Rightarrow \forall s,X,Y,Z.\ (s\restriction A,X) \in st(P)\ \&\ (s\restriction B,Y) \in st(Q)$

$\&\ Z = (X\restriction A \cap Y\restriction B) \cup X\restriction(A-B) \cup Y\restriction(B-A)$

$\Rightarrow \exists_{i=1}^{n} X_i,Y_i.\ S\left[{}^{X_i}/_{c_i.ready}\right] s,Z$

$\&\ R\left[{}^{Y_i}/_{c_i.ready}\right] s,Z$

$\&_{i=1}^{n}\ (c_i.ready = X_i \cap Y_i)\ s,Z$

(let $X_i = X\restriction\{c_i\}$ and $Y_i = Y\restriction\{c_i\}$, i=1,2,...,n)

$\Rightarrow$ RHS (2.7)

3.8 Hiding

$$(P \text{ sat } R) \Longrightarrow (\underline{\text{chan}}\ b\ \underline{\text{in}}\ P)\ \underline{\text{sat}}\ \exists b.\text{past}.\ R\left[\emptyset/_{b.\text{ready}}\right]$$

Note. This proof rule is simpler than the corresponding one in [1]. This is because we ignore livelock here, so infinite chatter on the hidden channel b is allowed in this rule.

proof. $(P \text{ sat } R) \equiv \forall (s,X) \in st(P).\ R_{s,X}$ (def.)

$\Longrightarrow \forall (s,X) \in st(P).\ \exists b.\text{past}.\ R_{s,X}$ (let b.past = b.s)

$\Longrightarrow \forall (s,X) \in st(P), s_1.\ s_1 = s\backslash b\ \&\ X\lceil\{b\} = \emptyset$

$\Longrightarrow \exists b.\text{past}.R\left[\emptyset/_{b.\text{ready}}\right]\ s_1,X$

(since $c.s = c.s_1$ provided $c \neq b$)

$\Longrightarrow$ RHS (2.8)

3.9 Recursion

Let R be an assertion with channel names $\{a,\ldots,z\}$, and let $\#s$ stand for the length of the sequence s. Then we define $R\lceil n$ as the same assertion of R if only the first (n-1) communications are considered:

$$R\lceil n =_{df} (\#a.\text{past} + \ldots + \#z.\text{past} \geq n) \vee R.$$

Thus $R\lceil 0 \equiv \text{true}$ and $R \equiv \forall n.R\lceil n$ are always true for any assertion R.

The inference rule for recursion is:

$$(\forall p,n.\ (\forall j \in N.\ p[j]\ \underline{\text{sat}}\ R[j]\lceil n) \Longrightarrow (\forall j \in N.\ F(p)[j]\ \underline{\text{sat}}\ R[j]\lceil n+1))$$

$$\Longrightarrow \forall j \in N.\ (\mu p[i:N].F)[j]\ \underline{\text{sat}}\ R[j]\ .$$

proof. We only prove the case of single recursion:

$$(\forall p,n.\ p \ \underline{sat}\ R\lceil n \longrightarrow F(p) \ \underline{sat}\ R\lceil n+1) \Longrightarrow (\mu p.F \ \underline{sat}\ R).$$

Since $R\lceil 0 \equiv$ true, $(\mu p.F \ \underline{sat}\ R\lceil 0)$ by the definition.

Then $(F(\mu p.F) \ \underline{sat}\ R\lceil 1)$ by the premiss (let $p = \mu p.F$ and $n=0$).

Hence for any n $(F^n(\mu p.F) \ \underline{sat}\ R\lceil n)$ by repeated use of the premiss.

(2.9) shows that for any n $st(\mu p.F) = st(F^n(\mu p.F))$, so $(\mu p.F \ \underline{sat}\ R\lceil n)$ holds for any n. Thus $(\mu p.F \ \underline{sat}\ \forall n.R\lceil n)$ by the definition and $(\mu p.F \ \underline{sat}\ R)$ follows from $R = \forall n.R\lceil n$.

Beside the rules (3.1)-(3.9), of course, we need the familiar inference rule:

3.10 Consequence

If $(R \Longrightarrow S)$ is a theorem, then so is

$$(P \ \underline{sat}\ R) \Longrightarrow (P \ \underline{sat}\ S).$$

proof. Omitted; it follows directly from the definition.

The inference rules presented here are <u>more</u> complete than those in [1], in the sense that more expressible truths can be proved. For example, the unprovable proposition of [1]

$$(\underline{chan}\ b\ \underline{in}\ \mu p.(b!0 \longrightarrow p)) \ \underline{sat}\ \text{true}$$

becomes provable now.

But, meanwhile, for any assertion R

$$(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.(b!0 \longrightarrow p))\ \underline{\text{sat}}\ R$$

can unfortunately also be proved. Since

$$\mu p.(b!0 \longrightarrow p)\ \underline{\text{sat}}\ b.\text{ready} \neq \emptyset$$

can be derived by (recursion), then

$$(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.\ (b!0 \longrightarrow p))\ \text{sat}\ \emptyset \neq \emptyset \quad \text{by (hiding).}$$

Hence

$$(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.\ (b!0 \longrightarrow p))\ \underline{\text{sat}}\ R \quad \text{by (consequence).}$$

In other words, the process $(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.\ (b!0 \longrightarrow p))$ can meet any assertion under the interpretation (I-). So the theorem P sat R deduced in Calculus- is only conditionally correct. The trouble here is that (I-) does not deal with livelock, and $(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.\ (b!0 \longrightarrow p))$ is livelock everywhere. Thus its stable state set is empty. That is why it can meet any assertion.

4. Liveness

We now take up the question of livelock. A process gets into a livelock after some moment during its evolution, if from this moment the process may engage in an infinite sequence of internal transitions. This concept can be precisely described in the model as follows.

Let P be a process. We first use trace(P) to denote all possible action sequences of P (both finite and infinite action sequences).

$$\text{trace}(P) =_{df} \left\{ s \;\middle|\; s \neq \langle\rangle \,\&\, \forall i \in \{0..\#s\}\ \exists P_i.\ P_0 = P \;\&\; (i \neq \#s \Longrightarrow P_i \xrightarrow{(s)_{i+1}} P_{i+1}) \right\},$$

where $(s)_i$ stands for the ith action of s and $\#s = \infty$ if s is an infinite sequence.

Let s be a finite action sequence, and let τ^∞ stand for an infinite sequence of τ. Then $s\tau^\infty \in \text{trace}(P)$ means that P can take actions of s and after doing s P can engage in an infinite internal transition sequence, i.e. P is livelocked after the moment of just finishing s. So if we use live(P) to mean "P is livelock-free", then

$$\text{live}(P) =_{df} \neg \exists s \in (\Sigma \cup \{\tau\})^* \,.\, s\tau^\infty \in \text{trace}(P)\ .$$

Suppose $P \xrightarrow{s} Q$ and $\tau^\infty \in \text{trace}(Q)$. This implies that P can reach a state by doing s which will never get stable by the internal transitions of P, even if we are so patient as to wait for an arbitrarily long time before further communication with it. So this is a state which is unreliable and cannot recover by itself.

But if P is live and $P \xrightarrow{s} Q$, then $\tau^\infty \notin \text{trace}(Q)$ by the definition of live(P). Thus even if P enters into an unstable state, sooner or later it will be stable again, and perhaps sooner, due to the high speed of internal transition. In other words, if P is live, then the communication with its environment can be expected to go on reliably at any time. This reasoning is formalized as follows.

Theorem. Let P be a process and live(P). Then

$$\forall s.(\exists P_1.P \stackrel{s}{\Longrightarrow} P_1) \Longrightarrow \exists Q.\ P_1 \stackrel{\langle\rangle}{\Longrightarrow} Q \;\&\; (s, Q^\circ) \in \text{st}(P).$$

proof. Suppose $P \stackrel{s}{\Longrightarrow} P_1$. Since $s\tau^{\infty}\tilde{\epsilon}$ trace(P) by live(P), there must exist n and Q such that $P_1 \xrightarrow{\tau^n} Q$ and $\tau\tilde{\epsilon}Q^o$. Thus $(s,Q^o) \in st(P)$ from the definition of st(P).

As a preparation for Calculus+ we need rules to build livelock-free processes.

The fact that all operators except the hiding and recursion preserve liveness can be directly proved from the definition of liveness.

Theorems

4.1 live(STOP)

4.2 $live(c!e \longrightarrow P) \equiv live(P)$

4.3 $live(c?x{:}M \longrightarrow P(x)) \equiv \forall x \in M.\ live(P(x))$

4.4 $live(P \underline{\text{or}}\ Q) \equiv live(P)\ \&\ live(Q)$

4.5 $live(c?x{:}M \longrightarrow P(x)\ []\ d?y{:}N \longrightarrow Q(y))$
$\equiv \forall x \in M.\ live(P(x))\ \&\ \forall y \in N.\ live(Q(y))$

4.6 $live(P[c/d]) \equiv live(P)$

4.7 $live(P)\ \&\ live(Q) \Longrightarrow live(P \underset{AB}{\downarrow} Q)$

proofs. Omitted.

The hiding operator can preserve liveness under a certain assumption - the original process will not be involved in infinite consecutive communications on the hidden channel.

Let P be a process and let b be a channel.

Then

$$live_b(P) = {}_{df}\ \neg\exists s.\ P \stackrel{s}{\longrightarrow} Q\ \&\ \forall i\, \exists \alpha_i.\ ch(\alpha_i) \in \{b,\tau\}\ \&$$
$$\alpha_1 \hat{}\ \alpha_2 \hat{}\ \ldots \in trace(Q).$$

Theorem

4.8 $\text{live}_b(P) \Longrightarrow \text{live}(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P)$

proof. Since $s\tau^\infty \in \text{trace}(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P)$

iff there exists $s_1 \hat{} \alpha_1 \hat{} \alpha_2 \hat{} \ldots$ in trace (P) such that

$$s_1\left[\tau/b\right] = s \quad \text{and} \quad \forall i.\ ch(\alpha_i) \in \{b,\tau\}.$$

We now must face the fact that a process defined by recursion is not always live. Can we find a syntactic restriction on recursion which can guarantee the liveness of the defined process? Ths answer is positive. However there is a variety of such restrictions, and we prefer here to present a simple but useful one.

Let us first see some examples.

(a) $F(p) = \underline{\text{chan}}\ b\ \underline{\text{in}}\ (b!e \longrightarrow p)$.

Since $(b!e \longrightarrow \mu p.F) \xrightarrow{b.e} \mu p.F$ by (1.2),

$\underline{\text{chan}}\ b\ \underline{\text{in}}\ (b!e \longrightarrow \mu p.F) \xrightarrow{\tau} (\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F)$ by (1.8).

Then $\mu p.F \xrightarrow{\tau} (\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F)$ by (1.9).

Thus $(\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F) \xrightarrow{\tau} \underline{\text{chan}}\ b\ \underline{\text{in}}\ (\underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F)$

and $\underbrace{(\underline{\text{chan}}\ b\ \underline{\text{in}} \ldots \underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F)}_{n} \xrightarrow{\tau} \underbrace{(\underline{\text{chan}}\ b\ \underline{\text{in}} \ldots \underline{\text{chan}}\ b\ \underline{\text{in}}\ \mu p.F)}_{n+1}$

by (1.8).

So $\mu p.F$ is not live.

The reason for its livelock is obvious, because $\mu p.F$ has not got any external channel. The only possible transition of $\mu p.F$ is internal, so if it is to progress any longer, it must be livelocked.

Moreover why and how should we be concerned with a process which cannot communicate with its environment?

Thus it is suggested that the recursion which defines a live process must have <u>external channels</u>. Let F be a recursion (single or mutual). Then the external channels of F are channels specifically for external communications, never hidden or renamed in F.

$$xch(F) =_{df} \{c \mid c \text{ does not occur in the hiding operator of F as a hidden channel, nor in the renaming operator of F as a renamed channel}\}.$$

(b) $F(p) = (c?e \longrightarrow p)$ <u>or</u> p

In this recursion, F has an external channel c. But $\mu p.F$ is still not live. Since $F(\mu p.F) \xrightarrow{\tau} \mu p.F$ by (1.4), then $\mu p.F \xrightarrow{\tau} \mu p.F$ by (1.9).

The trouble here is that one of the occurrences of p in F is not <u>guarded</u> by external channel c.

Hence the restriction has to be strengthened by the further requirement that the recursion which we use to define process must be <u>externally guarded</u>. Precisely for single recursion $p \triangleq F(p)$, we say that F(p) is externally guarded iff every occurrence of p in F(p) has an enclosing external guard, i.e. there exists G, H_i, and g_i such that

$$F(p) = G(g_1 \longrightarrow H_1(p), \ldots, g_n \longrightarrow H_n(p)),$$

where every g_i is an input or output command on channel of xch(F).

The definition of the externally guarded mutual recursion can be given in the following way.

For recursion $p[i:N] \triangleq F(p)$, we say that $p[i]$ is <u>not</u> externally guarded in $F(p)[j]$, if $p[i]$ occurs in $F(p)[j]$ without an enclosing guard of $xch(F)$.

Then $F(p)$ is externally guarded iff there is no infinite sequence $p[i_1]$, $p[i_2]$, ... ; such that for each j, $p[i_{j+1}]$ is not externally guarded in $F(p)[i_j]$. (cf. p. 72, [4]).

We end this section by giving a theorem that the process defined by an externally guarded recursion is live. The proof of the theorem is only shown for single recursion. It holds for mutual recursion as well, but is more tedious.

We first need four lemmas. They are mainly proved by structural induction on the process expression, so we will not give the details of the proofs.

Let $p \triangleq F(p)$ be a single recursion concerned.

Lemma 1. For any natural number n and expression H,

$$trace(H(\mu p.F)) = trace\ (H(F^n(\mu p.F))).$$

proof. (1) $trace(\mu p.F) = trace(F(\mu p.F))$ by the definition of trace and (1.9)

(2) By structural induction on expressions, we can prove

$$\forall p,q,H.(trace(p) = trace(q)) \Longrightarrow (trace(H(p)) = trace(H(q))).$$

Then the lemma follows (1) and (2) directly.

Lemma 2. If F(p) is externally guarded, then

$$F(p) \xrightarrow{\alpha} H(p) \quad \text{iff} \quad F(q) \xrightarrow{\alpha} H(q) \ ,$$

for any p, q, α and H.

proof. Since F(p) is externally guarded, F(p) must be of the form of $G(g_1 \longrightarrow H_1(p), \ldots, g_n \longrightarrow H_n(p))$, where g_i (i=1, ..., n) are guards. Then by making an induction on the structure of G, the lemma can be proved.

Let inch(F) be the set of internal channels of F(p), i.e.

$$\text{inch}(F) =_{df} \left\{ c \;\middle|\; \begin{array}{l} c \text{ occurs in a hiding operator of F as a hidden} \\ \text{channel or in a renaming operator of F as a} \\ \text{renamed channel} \end{array} \right\} .$$

Lemma 3. If F(p) is externally guarded and $F(p) \xrightarrow{\alpha} H(p)$, then

$$\text{inch}(F) \supseteq \text{inch}(H).$$

proof. By the routine induction on the structure of G too,

where $F(p) = G(g_1 \longrightarrow H_1(p), \ldots, g_n \longrightarrow H_n(p))$.

One of the corollaries of this lemma which we shall use in the proof of the desired theorem is that:

if $H_o(p) \xrightarrow{\alpha_1} H_1(p) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_n} H_n(p)$ and $\forall i<n.\ (H_i(p)$ is externally guarded), then $H_n(H_o(p))$ is externally guarded.

This is because the guards of every occurrence of p in $H_n(H_o(p))$ are not less than the guards of the corresponding occurrence of p in $H_o(p)$, and $\text{inch}(H_n) \subseteq \text{inch}(H_o)$ by the lemma. Thus we can prove the conclusion from the hypothesis: $H_o(p)$ is externally guarded.

Lemma 4. If

(1) $F(p) \xrightarrow{\alpha} H(p)$,

(2) $F(p)$ is externally guarded,

and (3) $H(p)$ is externally unguarded,

then $\alpha \neq \tau$.

proof. Roughly speaking, since $F(p)$ is externally guarded, while $H(p)$ is not, an external communication must be referred.

The rigorous proof can be given by induction on the transition rules (1.1)~(1.8).

We prove the theorem itself now.

Theorem 4.9 (single recursion)

If $p \triangleq F(p)$ is externally guarded,

and for any process p, $\text{live}(p) \Longrightarrow \text{live}(F(p))$,

then $\text{live}(\mu p.F)$.

proof. Suppose $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots) \in \text{trace}(\mu p.F)$. Let us prove that $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots)$ cannot take τ^∞ as its tail.

By lemma 1, $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots) \in \text{trace}(F(\mu p.F))$. Say

$$F(\mu p.F) \xrightarrow{\alpha_1} H_1(\mu p.F) \xrightarrow{\alpha_2} H_2(\mu p.F) \xrightarrow{\alpha_3} \ldots$$

If all $H_j(p)$ is guarded, then, by lemma 2, for any p $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots) \in \text{trace}(F(p))$. So $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots) \in \text{trace}(F(\text{STOP}))$.

But $\text{live}(F(\text{STOP}))$ by $\text{live}(\text{STOP})$ and $\text{live}(p) \Longrightarrow \text{live}(F(p))$, therefore $(\alpha_1 \hat{} \alpha_2 \hat{} \ldots)$ is as required.

If $\exists k.\ (\forall i<k.\ H_i(p)$ is externally guarded) & ($H_k(p)$ is not externally guarded), then $\alpha_k \neq \tau$ by lemma 4.

Furthermore

$F^2(p) \xrightarrow{\alpha_1} H_1(F(p)) \xrightarrow{\alpha_2} \ldots \xrightarrow{\alpha_k} H_k(F(p))$ (by lemma 2)

$H_i(F(p))$ $(i \leq k)$ are guarded (by lemma 3)

and $(\alpha_{k+1} \hat{}\ \alpha_{k+2} \hat{}\ \ldots) \in \mathrm{trace}(H_k(F(\mu p.F)))$ (by lemma 1).

Say $H_k(F(\mu p.F)) \xrightarrow{\alpha_{k+1}} G_1(\mu p.F) \xrightarrow{\alpha_{k+2}} G_2(\mu p.F) \xrightarrow{\alpha_{k+3}} \ldots$

It implies $F^2(\mu p.F) \xrightarrow{\alpha_1} \ldots \xrightarrow{\alpha_{k+1}} G_1(\mu p.F) \xrightarrow{\alpha_{k+2}} G_2(\mu p.F) \xrightarrow{\alpha_{k+3}} \ldots$

Then repeating the previous reasoning, we can get α_m $(m>k)$ such that $\alpha_m \neq \tau$.

Hence in this way we can prove that $\forall n\ \exists m.\ m>n$ & $\alpha_m \neq \tau$; i.e. $(\alpha_1 \hat{}\ \alpha_2 \hat{}\ \ldots)$ cannot take τ^∞ as its tail.

The version of Theorem 4.9 for mutual recursion should be

Theorem 4.9 (mutual recursion).

If $P[i:N] \triangleq F(p)$ is externally guarded,

and $(\forall j \in N.\ \mathrm{live}(p[j])) \Longrightarrow (\forall j \in N.\ \mathrm{live}(F(p)[j]))$,

then $\forall j \in N.\ \mathrm{live}((\mu p[i:N].F)[j])$.

We are now in a position to modify calculus- into calculus+, which takes liveness into account.

5. Calculus+

In this calculus (P sat R) has a stronger interpretation:

$$(P \text{ sat } R) =_{df} \text{live}(P) \;\&\; \forall (s,X) \in \text{st}(P) \,.\, R_{s,X} \qquad \ldots \quad (I_+) \,.$$

Under this interpretation, for any assertion R (P sat R) implies live (P), and for no process P (P sat false) holds. It is because Theorem in 4 says that any livelock-free process after doing any observable action sequence can get into a stable state. Meanwhile the empty sequence $\langle\rangle$ is an observable action sequence of every process by (1.11). Thus the stable-state-set of a live process is always nonempty. Hence (P sat false) cannot hold for any P by its interpretation.

So the calculus consistent with this interpretation can be expected to prove the liveness of process, and reject the ridiculous formula (P sat false) as well.

Calculus+ is obtained from Calculus- by replacing (4.8) and (4.9) by the following rules (5.8) and (5.9). But all the proofs for the consistency of the rules must use the interpretation (I_+) instead. Here we only list (5.8) and (5.9) with their consistency proofs.

5.8 Hiding

Let f be a total function from message strings to natural numbers, and let $\{c, \ldots, z\}$ be a finite set of channel names. Then

$$(P \text{ sat } (R \;\&\; \#b.past \leq f(c.past, \ldots, z.past)))$$
$$\Longrightarrow ((\underline{\text{chan}}\ b\ \underline{\text{in}}\ P)\ \underline{\text{sat}}\ \exists b.past.R\left[\not{}_{b.ready}\right]) .$$

proof. The only thing which we need to prove, given (3.8), is that:

$$P\ \underline{\text{sat}}\ \#b.past \leq f(c.past, \ldots, z.past)$$
$$\Longrightarrow \text{live}(\underline{\text{chan}}\ b\ \underline{\text{in}}\ P).$$

By (4.8), it will be enough for us to show

$$P\ \underline{\text{sat}}\ \#b.past \leq f(c.past, \ldots, z.past)$$
$$\Longrightarrow \text{live}_b(P).$$

This last proposition can be proved by

$$\text{LHS} \Longrightarrow \forall s \in (\Sigma \cup \{\tau\})^*.\ \ s \in \text{trace}(P) \Longrightarrow (\#b.s \leq f(c.s, \ldots, z.s))$$
(by Theorem in 4)

$$\Longrightarrow \text{live}_b(P) \qquad \text{(by def. of live}_b(P)).$$

5.9 Recursion

If $p[i:N] \triangleq F(p)$ is externally guarded,

then $(\forall p,n.\ (\forall j \in N.\ p[j]\ \underline{\text{sat}}\ R[j]\restriction n) \Longrightarrow (\forall j \in N.\ F(p)[j]\ \underline{\text{sat}}\ R[j]\restriction n+1))$

$$\Longrightarrow \forall j \in N.\ (\mu p[i:N].F)[j]\ \underline{\text{sat}}\ R[j] .$$

proof. We only need, by (3.9), to prove that

$\forall j \in N.\text{live}((\mu p[i:N].F)[j])$ is true under these antecedents.

By (4.9), this can be reduced to showing

$$(\forall j \in N.\ \mathrm{live}(p\lceil j\rceil)) \Longrightarrow (\forall j \in N.\ \mathrm{live}(F(p)\lceil j\rceil)).$$

This proposition is true, because $R\lceil j\rceil\restriction_0 \equiv \mathrm{true}$ and by the definition of P sat R, $\mathrm{live}(p\lceil j\rceil) \Longrightarrow (p\lceil j\rceil \text{ sat } \mathrm{true})$.

Thus

$$(\forall j \in N.\ \mathrm{live}(p\lceil j\rceil)) \Longrightarrow (\forall j \in N.\ p\lceil j\rceil \text{ sat } R\lceil j\rceil\restriction_0)$$

$$\Longrightarrow (\forall j \in N.\ F(p)\lceil j\rceil \text{ sat } R\lceil j\rceil\restriction_1)$$

(by the antecedent)

$$\Longrightarrow \forall j \in N.\ \mathrm{live}(F(p)\lceil j\rceil)$$

(by the def. of (P sat R)).

5.10 Consequence

If $R \Longrightarrow S$, then

$$(P \text{ sat } R) \longrightarrow (P \text{ sat } S).$$

6. Discussion

We have presented an operational model for two variants of Hoare's calculus, but we still do not know if there is a model for Hoare's calculus itself. The difficulty of finding that model seems to arise from that the predicate of past and ready variables is not able to describe the liveness of process, while the proof rule for recursion in Hoare's calculus has no antecedent to guarantee the liveness of the process concerned. So in the desired model an infinite internal transition sequence should always take a "stable" process as its limit. One of the possible candidates of that model can be found in [5], but unfortunately it rejects infinite hiding.

Acknowledgements

This paper results from the author's efforts to understand the calculus of Tony Hoare. It is almost a cooperative work with him. The operational model is strongly influenced by Robin Milner's elegant work on communicating systems. Finally I am grateful for fruitful discussions with and suggestions from, Steve Brookes, Bill Roscoe, Cliff Jones and Ernst-Rüdiger Olderog.

References

[1] C.A.R. Hoare A Calculus for Total Correctness of Communicating Processes, Science of Computer Programming 1 (1981) 49-72.

[2] Zhou Chaochen and C.A.R. Hoare Partial Correctness of Communicating Sequential Processes, Proc. of the Second International Conference on Distributed Computing System, April 1981.

[3] Zhou Chaochen and C.A.R. Hoare Partial Correctness of Communication Protocols, Protocol Testing - towards Proving?, INWG/NPL Workshop. May 1981.

[4] Robin Milner A Calculus of Communicating Systems, Vol. 92 LNCS, 1980

[5] C.A.R. Hoare, S.D. Brookes and A.W. Roscoe A Theory of Communicating Sequential Processes, Technical Monograph PRG-16, May 1981.

OXFORD UNIVERSITY COMPUTING LABORATORY
PROGRAMMING RESEARCH GROUP TECHNICAL MONOGRAPHS

MARCH 1982

This is a series of technical monographs on topics in the field of computation. Copies may be obtained from the Programming Research Group, (Technical Monographs), 45 Banbury Road, Oxford, OX2 6PE, England. Prices include surface postage.

PRG-2 Dana Scott
Outline of a Mathematical Theory of Computation

PRG-3 Dana Scott
The Lattice of Flow Diagrams

PRG-5 Dana Scott
Data Types as Lattices

PRG-6 Dana Scott and Christopher Strachey
Toward a Mathematical Semantics for Computer Langueges

PRG-7 Dana Scott
Continuous Lattices

PRG-8 Joseph Stoy and Christopher Strachey
OS6 – an Experimental Operating System for a Small Computer

PRG-9 Christopher Strachey and Joseph Stoy
The Text of OSPub

PRG-10 Christopher Strachey
The Varieties of Programming Language

PRG-11 Christopher Strachey and Christopher P Wadsworth
Continuations: A Mathamatical Semantics for Handling Full Jumps

PRG-12 Peter Mosses
The Mathematical Semantics of Algol 60

PRG-13 Robert Milne
The Formal Semantics of Computer Languages and their Implementations

PRG-14 Shan S. Kuo, Michael H. Linck and Sohrab Saadat
A Guide to Communicating Sequential Processes

PRG-15 Joseph Stoy
The Congruence of Two Programming Language Definitions

PRG-16 C. A. R. Hoare, S D Brookes and A. W. Roscoe
A Theory of Communicating Sequential Processes

PRG-17 Andrew P Black
Report on the Programming Notation 3R

PRG-18 Elizabeth Fielding
The Specification of Abstract Mappings and their Implementation as B^+-trees

PRG-19 Dana Scott
Lectures on a Mathematical Theory of Computation

PRG-20 Zhou Chao Chen and C. A. R. Hoare
Partial Correctness of Communicating Processes and Protocols

PRG-21 Bernard Sufrin
Formal Specification of a Display Editor

PRG-22 C. A. R. Hoare
A Model for Communicating Sequential Processes

PRG-23 C. A. R. Hoare
A Calculus of Total Correctness for Communicating Processes

PRG-24 Bernard Sufrin
Reading Formal Specifications

PRG-25 C. B. Jones
Development Methods for Computer Programs including a Notion of Interference

PRG-26 Zhou Chao Chen
The Consistency of the Calculus of Total Correctness for Communicating Processes