1980

Additional notes on a model for communicating sequential processes

C. A. R. Hoare
University of Wollongong

Additional Notes on

A Model for

COMMUNICATING SEQUENTIAL PROCESSES

C.A.R. Hoare

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

Summary: These notes contain copies of the overhead projector slides presented at the Communicating Sequential Processes Symposium in Wollongong which were not included in the original preprint 80-1 issued at the Symposium.

ALGEBRAIC PROPERTIES.

$\Box$ is associative	$P \Box (Q \Box R) = (P \Box Q) \Box R$
commutative	$P \Box Q = Q \Box P$
idempotent	$P \Box P = P$
with unit:	$P \Box \mathrm{ABORT} = P$
and "zero":	$P \Box (\bar{P})^* = (\bar{P})^*$
$\Vert$ is associative	$P \Vert (Q \Vert R) = (P \Vert Q) \Vert R$
commutative	$P \Vert Q = Q \Vert P$
with unit:	$P \Vert (\bar{P})^* = P$
and zero:	
; is associative	$P ; (Q ; R) = (P ; Q) ; R$
with unit:	$\mathrm{SKIP} ; P = P$
and zero:	$\mathrm{ABORT} ; P = \mathrm{ABORT}$
and distributes:	$(a \to P) ; Q = a \to (P ; Q)$
	$(P \Box Q) ; R = (P ; R) \Box (Q ; R)$
$\gg$ is associative	$P \gg (Q \gg R) = (P \gg Q) \gg R$
and distributes:	

$$(ABORT \gg ABORT) = (ABORT \gg (?x:T \rightarrow Q(x)))$$

$$= ((!x;P) \gg ABORT) = ABORT$$

$$(!v;P) \gg (?x:T \rightarrow Q(x)) = P \gg Q(v) \qquad \text{for } v \in T.$$

$$(\qquad) \gg (\qquad) = ABORT \qquad \text{for } v \tilde{\in} T$$

$$(!v;P) \gg (!x;Q) = !x;((!v;P) \gg Q) \qquad ''$$

$$(?x:S \rightarrow P(x)) \gg (?y:T \rightarrow Q(y)) =$$

$$(?x:S \rightarrow (P(x) \gg (?y:T \rightarrow Q(y))))$$

$$(?v:S \rightarrow P(v)) \gg (!x;Q) =$$

$$(?v:S \rightarrow (P(v) \gg (!x;Q)))$$

$$\square(!x;((?v:S \rightarrow P(v)) \gg Q))$$

$$(?v:S \rightarrow P(v)) = (?w:S \rightarrow P(w))$$

$$(?v:\{\} \rightarrow P(v)) = ABORT.$$

RECURSION

Def A process with alphabet A is a non-empty prefix-closed subset of A^*

Thm. If P_i are processes with alphabet A for all i in T, then so are

$\bigcup_{i \in T} P_i$ Thus processes form a complete lattice under $\subseteq$, with ABORT as bottom and A^* as top

Def. A function F from processes to processes is <u>distributive</u> if for all sets $\{P_i \mid i \in T\}$

$$F\left(\bigcup_{i \in T} P_i\right) = \bigcup_{i \in T} F(P_i)$$

Thm. $\rightarrow$, $[]$, $||$, $;$, $m:$, $\backslash$ are distributive (and hence continuous and monotonic) in each of their arguments.

Thm (Tarski, Scott). If F is continuous, then the least p satisfying

$$p = F(p) = (\ldots; p \ldots \| \cancel{\text{AA}}: p \ /\ldots p \ )$$

is $\bigcup_{i \in NN} F^i(ABORT) \quad =_{df} \mu p.\, F(p)$

where $F^0(q) = q$ $\qquad F(A$

$$F^{n+1}(q) = F^n(F(q))$$

When a process is defined by recursion, we intend it to be the least solution of its defining equation.

And the same is true for sets of mutually recursive equations.

Proof RHS $= F\left(\bigcup_{i \in NN} F^i(ABORT)\right) = \bigcup_{i \in NN} F(F^i(ABORT))$ [continuity

$$= \bigcup_{i \in NN} F^{i+1}(ABORT) \qquad \text{[def } F^{i+1}$$

$$= ABORT \cup \bigcup_{i>0} F^i(ABORT) \qquad \text{[ABORT} \subseteq \text{any}$$

$$= \bigcup_{i \in NN} F^i(ABORT).$$

Unique fixed Points.

If $F(p)$ is an expression in which p appears only to the right of $\rightarrow$, (and F does not contain localisation), then the solution to $p = F(p)$ is <u>unique</u>.

Proof: $F(p)$ always does something before making the recursive call on p. So $F^n(p)$ does at least n things before calling on p; these are the same things as for $F^n(q)$. Suppose $p = F(p)$ & $q = F(q)$. Let $s \in p$ be of length n. Since $p = F^m(p)$ for all m, $s \in F^n(p)$. so $s \in F^n(q)$. Thus $p \subseteq q$. Similarly, $q \subseteq p$. $\therefore$ $p = q$

The same is true for sets of mutually recursive equations.

RECURSION INDUCTION

$$COUNT_0 = (iszero \to COUNT_0 \;[]\; up \to COUNT_1)$$

$$COU \quad = (down \to COUNT_n \;[]\; up \to COUNT_{n+2})$$

$$POS = (down \to SKIP \;[]\; up \to POS ; POS)$$

$$ZERO = (iszero \to ZERO \;[]\; up \to POS ; ZERO).$$

Theorem. $ZERO = COUNT_0$.

Proof. Define $C_0 = (iszero \to C_0 \;[]\; up \to C_1)$

$$C_{n+1} = POS ; C_n$$

$$C_{n+1} = (down \to SKIP ; C_n \;[]\; up \to (POS ; POS) ; C_n) \qquad \text{def } C_{n+1} \text{ \& distr ;}$$

$$= (down \to C_n \;[]\; up \to POS ; (POS ; C_n)) \qquad \text{prop ;}$$

$$= (down \to C_n \;[]\; up \to POS ; C_{n+1}) \qquad \text{def } C_{n+1}$$

$$= (down \to C_n \;[]\; up \to C_{n+2}) \qquad \text{def } C_{n+1}$$

$\therefore \quad C_n = COUNT_n$ for all n. ... (1)

but $C_0 = (iszero \to C_0 \;[]\; up \to POS ; C_0)$

$\therefore \quad C_0 = ZERO$ ... (2)

Conclusion follows from (1), (2).

YET ANOTHER COUNT.

$$Z = (\text{iszero} \to Z \,\square\, \text{up} \to (p{:}Z \,\|\, X);Z)$$

where
$$\begin{aligned} X = (&\text{up} \to p.\text{up} \to X \\ &\square\, \text{down} \to (p.\text{iszero} \to \text{SKIP} \\ &\qquad\qquad\quad \square\, p.\text{down} \to X \\ &\qquad\qquad\quad) \\ &) \end{aligned}$$

Theorem. $Z = \text{ZERO}$

RELATIONS.

Let $R : (\mathrm{ins}(\alpha P))^* \leftrightarrow (\mathrm{outs}(\alpha P))^*$

P sat $R =_{df} \forall s\, (s \in P \rightarrow (\mathrm{ins}(s), \mathrm{outs}(s)) \in R)$

– at all times, the sequence of values input by P bears relation R to the sequence of values output by P.

e.g. let f: be a monotonic function of traces.

let $R_f = \{(i, o) \mid o \leq f(i)\}$

If P sat R_f, P is said to be a pipe for f.

We shall often represent R as a predicate on the variables "in" and "out"

eg. P sat $(out \leq in)$ $= P$ sat $\{(in, out) \mid out \leq in\}$

– means that P is a buffer

i.e. a pipe for the identity function.

Theorem. Q sat R & $R \subseteq S \Rightarrow Q$ sat S

$(\forall i\; Q_i \text{ sat } R) \Rightarrow (\bigcup_i Q_i)$ sat R

If ABORT sat R & $\forall p.\ p$ sat $R \Rightarrow F(p)$ sat R

then $(\mu p. F(p))$ sat R. (fixed point induction)

① $\text{ABORT } \underline{\text{sat}}\ R \;\equiv\; R^{\,in\ \ out}_{\,\langle\rangle\ \ \langle\rangle}$

② $(!x ; P)\ \underline{\text{sat}}\ R \;\equiv\; P\ \underline{\text{sat}}\ R^{out}_{\langle x\rangle out}$

where $R^{out}_{\langle x\rangle out} = \{(i,o) \mid (i, \langle x\rangle o) \in R\}$

i.e. replace "out" by "$\langle x\rangle$out" in R

③ $(?x:T \to P(x))\ \underline{\text{sat}}\ R \;\equiv\; \forall x:T.\,(\,P(x)\ \underline{\text{sat}}\ R^{in}_{\langle x\rangle in}\,)$

i.e. it works for all input values.

by ② $((!x ; B)\ \underline{\text{sat}}\ out \le \langle x\rangle in) \equiv B\ \underline{\text{sat}}(\langle x\rangle out \le \langle x\rangle in)$

$\equiv B\ \underline{\text{sat}}\ out \le in.$

by ③ $((?x:T \to !x ; B)\ \underline{\text{sat}}\ out \le in)$

$\equiv \forall x:T\ ((!x ; B)\ \underline{\text{sat}}\ out \le \langle x\rangle in)$

$\equiv B\ \underline{\text{sat}}\ out \le in$ (just proved)

by ① $\text{ABORT } \underline{\text{sat}}\ out \le in.$ (because $outs(\langle\rangle) = ins(\langle\rangle) = \langle\rangle$)

$\therefore$ if $B =_{df.} (?x:T \to (!x ; B))$

then B is a buffer. (Fixed point induction)

$$T = outs(\alpha P) = ins(\alpha Q)$$

$$(P \ \underline{sat}\ R) \ \& \ (Q \ \underline{sat}\ S) \Rightarrow$$

$$(P \gg Q) \ \underline{sat} \ \exists s \, (R^{out}_{s} \ \& \ S^{in}_{s})$$

which is $(R ; S)$ — relational composition of R and S.

Proof. $t \in (P \gg Q)$, $P \ \underline{sat}\ R$, $Q \ \underline{sat}\ S$ — assume

$\therefore \ \exists u \ \ t = u \setminus T \ \ \& \ \exists v \in P \ \exists w \in Q$

and (4) $ins(v) = ins(u) \ \& \ outs(w) = outs(u) = outs(t)$

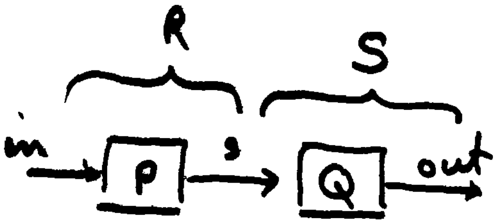
$\quad \& \ outs(v) = ins(w) = u \upharpoonright T$

$\therefore \ (ins(v), outs(v)) \in R \ \& \ (ins(w), outs(w)) \in S$

$\quad \Rightarrow (ins(u), u \upharpoonright T) \in R \ \& \ (u \upharpoonright T, outs(u)) \in S$

$\quad \Rightarrow \exists s \ (ins(t), s) \in R \ \& \ (s, outs(t)) \in S$

$\quad \Rightarrow (ins(t), outs(t)) \in (R ; S)$.



If f and g are monotonic:

$$(P \; \underline{sat} \; outs \le f(ins)) \;\&\; (Q \; \underline{sat} \; outs \le g(ins))$$

$$\Rightarrow (P \gg Q) \; \underline{sat} \; \exists s \;\; s \le f(ins) \;\&\; outs \le g(s))$$

$$\Rightarrow (P \gg Q) \; \underline{sat} \; outs \le g(f(ins))$$

theorem If P is a pipe for f and Q for g

then $P \gg Q$ is a pipe for $g \circ f$.

If P and Q are buffers, so is $P \gg Q$

(a buffer is a pipe for the identity function).

since $B1 = (?x:T \rightarrow !x;\, B1)$ is a buffer.

so is $B_{n+1} = B_n \gg B_1$ for all $n \ge 1$.

Proof. induction on n

WARNING $\underline{sat}$ defines only a form of partial correctness - does not prove absence of deadlock. eg. the following are buffers.

ABORT, ; $(?x:\{3\} \rightarrow !3;\, B_1)$, $B_{\langle\rangle}$ where

$$B_s = (?x:T \rightarrow (?y:T \rightarrow !s_0;\; B_{s'\langle x\rangle\langle y\rangle}))$$

COMMUNICATIONS.

A communication protocol consists of a transmitting process P and a receiving process Q such that $P \gg Q$ is a buffer. i.e. its outputs are at all times an initial segment of its inputs

Theorem. If for all $x{:}T$, $P_x \gg Q_x$ is a buffer then so is $(?x{:}T \rightarrow (P_x \gg (!x; Q_x)))$ ..... (1)

Proof. Let t be a trace of (1)

then $\text{first}(\text{ins}(t)) = \text{first}(\text{outs}(t))$. ..... (2)

Let t' be formed from t by omitting its first input and its first output. t' must be a trace of $P_x \gg Q_x$, which is a buffer.

$\therefore \quad \text{outs}(t') \leq \text{ins}(t')$ (3)

but $\text{ins}(t) = \langle\text{first}(\text{ins}(t))\rangle\, \text{ins}(t')$ (4)

and $\text{outs}(t) = \langle\text{first}(\text{outs}(t))\rangle\, \text{outs}(t')$ (5)

$\therefore \quad \text{outs}(t) \leq \text{ins}(t)$ from (2, 3, 4, 5)

If for all $x:T$

$$P_x \gg Q_x = (?y:T \to P_y \gg (!y\,;\, Q_y))$$

then $P_x \gg Q_x$ is a buffer for all $x:T$.

Proof. induction on length of trace of $P_x \gg Q_x$.

t is OK $=_{df}$ $outs(t) \leq ins(t)$, so $\langle\rangle$ is OK.

assume all t of length $\leq n$ in $P_x \gg Q_x$ are OK (forall x)

now let $t' \in P_x \gg Q_x$ be ~~longer than~~ of length $n+1$.

If t' is all inputs, it's OK

otherwise $t' \in$ RHS, so on removal of its first input and output (which are equal), it is still in $P_y \gg Q_y$ for some y. By induction hypothesis, it's still OK

If $P_1 \gg Q_1$ and $P_2 \gg Q_2$ are buffers then so ~~are~~ is $(P_1 \gg P_2) \gg (Q_2 \gg Q_1)$

(Composition of protocols).

Phase encoding.

$$P = (?x:\{0,1\} \rightarrow (!x;\ !(1-x);\ P))$$

$$(Q = ?x:\{0,1\} \rightarrow (?y:\{1-x\} \rightarrow (!x;\ Q)))$$

$$\begin{aligned}(!0;\ !1;\ R) \gg P &= !0;\ !1;((!1;R) \gg P)\\ &= !0;\ !1;\ !1;\ !0;(R \gg P)\end{aligned}$$

Theorem. $P \gg Q$ is a buffer.

Proof $P \gg Q =$

$= ?x:B \rightarrow (!x;\ !(1-x);P) \gg (?y:B \rightarrow ?z:\{1-y\} \rightarrow (!y;\ Q))$

$= ?x:B \rightarrow ((!(1-x);\ P) \gg (?z:\{1-x\} \rightarrow (!x;\ Q)))$

$= ?x:B \rightarrow (P \gg (!x;\ Q))$

$\therefore$ $P \gg Q$ is a buffer.

NRZ Protocol

$P_0 \gg Q_0$

$$P_0 = ?x:\{0,1\} \to !x \,;\, P_x$$

$$P_1 = ?x:\{0,1\} \to !(1-x) \,;\, P_x$$

x records previous input bit.

$$(!1\,;\,0!\,;\,!0\,;\,!1\,;\,R) \gg P_0 = !1\,;((!0\,;!0\,;1\,;R) \gg P_1)$$
$$= !1\,;\,!1\,;((!0\,;\,!1\,;\,R) \gg P_0)$$
$$= !1\,;\,!1\,;\,!0\,;\,((!1\,;\,R) \gg P_0)$$
$$= !1\,;\,!1\,;\,!0\,;\,!1\,;\,(R \gg P_1)$$

P copies first bit
then outputs 0 if input value remains same
1 if input value changes.

$$Q_0 = ?x:\{0,1\} \to !x \,;\, Q_x$$

$$Q_1 = ?x:\{0,1\} \to !(1-x) \,;\, Q_{(1-x)}$$

$$(!1;!1;\,!0\,;\,!1\,;\,R) \gg Q_0 = !1\,;\,!0\,;\,!0\,;\,!1\,;\,(R \gg Q_1)$$

Q copies first bit
then copies if previous output was 0
inverts if previous output was 1.

Prove that $P_x \gg Q_x$ is a buffer. for $x = 0, 1$

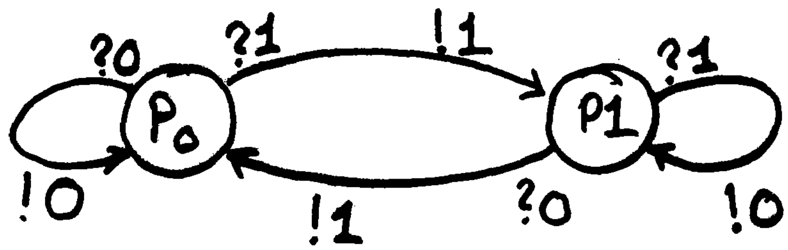
$$P_0 \gg Q_0 = ?x:\{0,1\} \to ((!x; P_x) \gg Q_0)$$
$$= ?x:\{0,1\} \to (P_x \gg (!x; Q_x))$$

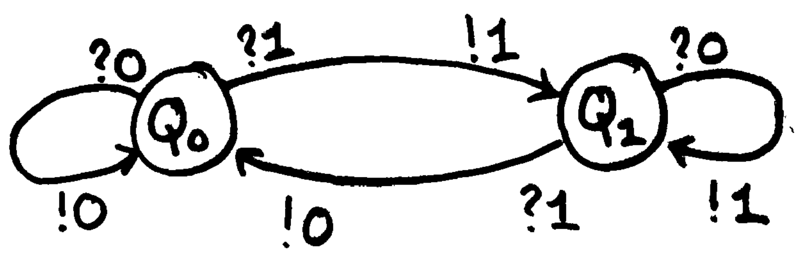
$$P_1 \gg Q_1 = ?x:\{0,1\} \to ((!1-x; P_x) \gg Q_1)$$
$$= ?x:\{0,1\} \to P_x \gg (!(1-(1-x)); Q_{(1-(1-x))})$$
$$= ?x:\{0,1\} \to P_x \gg (!x; Q_x)$$

So $P_y \gg Q_y = (?x:\{0,1\} \to P_x \gg (!x; Q_x))$

for $y = 0, 1$

Therefore they are buffers.





A MODEL OF NON-DETERMINISM IN COMMUNICATING SEQUENTIAL PROCESSES.

with thanks to

Steve Brooks, Bill Roscoe

March 1980

The problem

Consider $R = (x \to a \to P \mathbin{[]} y \to b \to Q) \setminus \{x, y\}$

Clearly, on its first step, it can accept "a", and it can accept "b". BUT also, it can refuse "a" (if "y" happened) and it can refuse "b" (if "x" happened). In our simple model, $R = (a \to P \mathbin{[]} b \to Q)$, and the possibilities of refusal have not been represented. We need a more complex model.

Let P be a process with finite alphabet A. Let traces(P) be the subset of A^* denoting traces of the possible behaviour of P.

So traces(P) is nonempty & prefix-closed.

Define $P^\circ = \{a \mid \langle a \rangle \in \text{traces}(P)\}$
P° is the set of events possible for P on the very first step.

Let X be a subset of A denoting the events possible for the <u>environment</u> of P. "P <u>canrefuse</u> X" means that P can deadlock in this environment.

So $\quad P$ <u>canrefuse</u> $\{\}$

$$P \ \underline{\text{canrefuse}}\ X \Rightarrow P \ \underline{\text{canrefuse}}\ X \cap Y$$

$$P \ \underline{\text{canrefuse}}\ X \Rightarrow P \ \underline{\text{canrefuse}}\ X \cup (A - P^{\circ}).$$

$(A - P^{\circ})$ is a set which P <u>must</u> refuse

Let s be in traces(P). Then "P <u>after</u> s" denotes the future behaviour of P if s is a trace of its past behaviour.

So $\quad P$ <u>after</u> $\langle\rangle = P$

$$P \ \underline{\text{after}}\ st = (P \ \underline{\text{after}}\ s)\ \underline{\text{after}}\ t$$

Proposition: A process is defined by what it can <u>do</u> and what it can <u>refuse</u>.

So if alphabet (P) = alphabet (Q)

and traces (P) = traces (Q)

and $\forall X$ (P <u>canrefuse</u> X $\equiv$ Q <u>canrefuse</u> X)

and $\forall a$ ($a \in P^0 \Rightarrow$ P <u>after</u> $\langle a \rangle$ = Q <u>after</u> $\langle a \rangle$)

then $P = Q$

We therefore <u>define</u> a process P as a relation:

For s in A^* and $X \subseteq A$

$(s, X) \in P$ <u>means</u> $s \in$ traces(P) & (P <u>after</u> s) <u>canrefuse</u> X.

So traces (P) $=_{df} \{ s \mid (s, \{\}) \in P \}$

P <u>canrefuse</u> X $=_{df}$ $(\langle\rangle, X) \in P$

P <u>after</u> s $=_{df} \{ (t, X) \mid (st, X) \in P \}$

traces (P) must be nonempty & prefix-closed

$\{X \mid$ P <u>canrefuse</u> X$\}$ must be nonempty & left closed and closed under union with traces$(P)^0$

EXAMPLES. with alphabet A.

$STOP_A$ can't do anything
must refuse everything.

$$STOP_A =_{df} \{(\langle\rangle, X) \mid X \subseteq A\}$$

RUN_A can do anything
can't refuse anything.

$$RUN_A =_{df} \{(s, \{\}) \mid s \in A^*\}$$

$CHAOS_A$ can do anything
can refuse anything.

$$CHAOS_A =_{df} \{(s, X) \mid s \in A^* \;\&\; X \subseteq A\}$$

for all $s \in A^*$: RUN_A <u>after</u> $s = RUN_A$

$CHAOS_A$ <u>after</u> $s = CHAOS_A$.

$\longrightarrow$

Let F be a function from A to processes.
Let $B \subseteq A$. Then

$(x : B \to F(x))$ first accepts any x in B, and then behaves like $F(x)$

$$(x : B \to F(x)) =_{df} \{(\langle\rangle, X) \mid X \subseteq A - B\}$$
$$\cup \{(\langle x \rangle s, X) \mid x \in B \;\&\; (s, X) \in F(x)\}$$

$(b \to P)$ is short for $(x : \{b\} \to P)$

$(x : B \to F(x))$ $\underline{\text{after}}$ $b = F(b)$ for all $b \in B$.

$(x : \{\} \to F(x)) = STOP_A$

$(x : B \to F(x)) = (y : B \to F(y))$

PARALLEL COMPOSITION.

P and Q have same alphabet A.

$P \| Q$ can accept anything acceptable
to **both** P and Q
and if P can refuse X and Q
can refuse Y, $P \| Q$ can refuse $X \cup Y$

$$(P \| Q) =_{df.} \{ (s, X \cup Y) \mid (s, X) \in P \ \& \ (s, Y) \in Q \}$$

$$\text{traces}(P \| Q) = \text{traces}(P) \cap \text{traces}(Q)$$

$$(P \| Q) \, \underline{\text{after}} \, s = (P \, \underline{\text{after}} \, s) \| (Q \, \underline{\text{after}} \, s) \quad \text{for } s \in \text{traces}(P \| Q)$$

$\|$ is associative, & commutative,
with unit RUN_A and zero $STOP_A$.

$$(x : B \to F(x)) \| (y : C \to G(y)) = (z : B \cap C \to (F(z) \| G(z)))$$

NONDETERMINISM.

$P \sqcap Q$ behaves non-deterministically, either like P or like Q.

It can do anything that P <u>or</u> Q can do.

It can refuse anything that P <u>or</u> Q can refuse.

$$P \sqcap Q =_{df} P \cup Q$$

$\sqcap$ is associative, commutative, and idempotent with zero $CHAOS_A$.

$$traces(P \sqcap Q) = traces(P) \cup traces(Q)$$

$$\begin{aligned}(P \sqcap Q) \,\underline{after}\, s &= P \,\underline{after}\, s && \text{if } s \in traces(P) - traces(Q)\\ &= Q \,\underline{after}\, s && \text{if } s \in traces(Q) - traces(P)\\ &= (P \,\underline{after}\, s) \sqcap (Q \,\underline{after}\, s) && \\ & && \text{if } s \in traces(P) \cap traces(Q)\end{aligned}$$

If we admit the EMPTY relation as a process, it w'd be the unit of $\sqcap$.

UNION

$P \,\Box\, Q$ behaves like P or like Q; the choice can be influenced by its environment, but only on the first step.

It can do anything P <u>or</u> Q can do.

It can refuse anything that <u>both</u> P and Q can refuse.

$$P \,\Box\, Q = \{(\langle\rangle, X) \mid (\langle\rangle, X) \in P \cap Q\}$$
$$\cup \{(s, X) \mid s \neq \langle\rangle \;\&\; (s, X) \in (P \cup Q)\}$$

$\Box$ is associative, commutative and idempotent with unit $STOP_A$.

$$\text{traces}(P \,\Box\, Q) = \text{traces}(P) \cup \text{traces}(Q)$$

$$(x : B \to F(b)) \,\Box\, (y : C \to G(c)) = (z : B \cup C \to$$
$$\underline{\text{if}}\ z \in B - C\ \underline{\text{then}}\ F(z)\ \underline{\text{else}}\ \underline{\text{if}}\ z \in C - B\ \underline{\text{then}}\ G(z)$$
$$\underline{\text{else}}\ F(z) \sqcap G(z))$$

LIMITS.

$P \sqsubseteq Q$ means Q is more deterministic than P.
& therefore more predictable, controllable, useful.

Everything Q can do so can P
Everything Q can refuse, so can P.

$P \sqsubseteq Q \;=_{df}\; Q \subseteq P$ or $P \sqcap Q = P$

e.g. $P \sqcap Q \sqsubseteq P \,[]\, Q$ $\qquad$ $CHAOS_A \sqsubseteq P$

$P \sqcap (P \,[]\, Q \,[]\, R) \sqsubseteq P \,[]\, Q$.

If $P_i \sqsubseteq P_{i+1}$ for all i, then we write

$$\bigsqcup_i P_i \;=_{df}\; \bigcap_i P_i$$

The relation $\sqsubseteq$ is a complete partial order with $CHAOS_A$ as its bottom.

If we add EMPTY, processes form a complete lattice with EMPTY as an isolated top.

SYMBOL CHANGE.

Let F be a total function from alphabet B onto alphabet A

Let P have alphabet A.

Then $F^{-1}(P)$ can do b (in B) whenever P can do $F(b)$, and can refuse X $(\subseteq B)$ whenever P can refuse $F(X) = \{F(x) \mid x \in X\}$

$F^{-1}(P) = \{(s, X) \mid (F(s), F(X)) \in P\}$ with alphabet B.

$traces(F^{-1}(P)) = \{s \mid F(s) \in traces(P)\}$

$(F^{-1}(P))^{\circ} = F^{-1}(P^{\circ})$

$(F^{-1}(P))\ \underline{after}\ s = F^{-1}(P\ \underline{after}\ F(s))$

$F^{-1}(x : C \rightarrow F(x)) =$

$F^{-1}(P \| Q) =$

$F^{-1}(P \sqcap Q) =$

If F is one-one, write $F(P)$ for $(F^{-1})^{-1}(P)$

ALPHABET EXTENSION

Let P be a process with alphabet A then P <u>ext</u> B is a process with alphabet $A \cup B$, which behaves like P, except that it is always prepared for any event in $B - A$, which it then ignores.

$$P \,\underline{ext}\, B = \{(s, X) \mid s \in (A \cup B)^* \ \&\ (s_A, X) \in P\}$$

where s_A is formed from s by omitting all symbols outside A.

$$(P \,\underline{ext}\, B)^\circ = P^\circ \cup (B - A)$$

$$(P \,\underline{ext}\, B) \,\underline{canrefuse}\, X \equiv P \,\underline{canrefuse}\, X$$

$$(P \,\underline{ext}\, B) \,\underline{after}\, s = (P \,\underline{after}\, s_A) \,\underline{ext}\, B$$

$$P \,\underline{ext}\, A = P, \quad (P \,\underline{ext}\, B) \,\underline{ext}\, C = P \,\underline{ext}\, (B \cup C)$$

If Q has alphabet B then

$$P \| Q =_{df} (P \,\underline{ext}\, B) \| (Q \,\underline{ext}\, A)$$

$\|$ is associative and commutative, etc.

ALPHABET CONTRACTION

Let P have alphabet A. Let B be a set of events to be regarded as internal to P. Then $P \backslash B$ is the process which behaves like P, but events in B may occur whenever they are possible, without participation or even the knowledge of the environment of P.

$$(P \backslash B)^\circ \supseteq P^\circ - B$$

$$P \ \underline{\text{can refuse}}\ X \ \& \ X \cap B = \{\} \Rightarrow (P \backslash B) \ \underline{\text{can refuse}}\ X$$

$$s \in \text{traces}(P) \Rightarrow s_{A-B} \in \text{traces}(P \backslash B)$$

$$(P \backslash B) \ \underline{\text{after}}\ s_{A-B} \sqsubseteq (P \ \underline{\text{after}}\ s) \backslash B.$$

These properties are satisfied by

$$\boxed{P \backslash B} = \{(s_{A-B}, X) \mid X \cap B = \{\} \ \& \ (s, X \cup B) \in P\}$$

But $\{(b^n, X) \mid X \subseteq \{a\}\} \backslash \{b\}$ is empty, i.e. <u>not a process</u>

The trouble lies in the <u>infinite</u> trace consisting of hidden symbols. The process <u>may</u> choose to follow this path forever and <u>never</u> engage in any further external interactions. or it may not. But you can't rely on anything. It's as bad as CHAOS. So let's make it so.

$$P\backslash B = \boxed{P\backslash B} \cup \{(st, X) \mid \{u \mid u \in P \;\&\; u_{A-B} = s\} \text{ is infinite}\}$$

$$P\backslash\{\} = P$$

$$(P\backslash B)\backslash C = P\backslash(B \cup C)$$

NOTE – we rely on <u>finitude</u> of alphabets

MONOTONICITY

$P \sqsubseteq R$ means that for all purposes R is better than P. Let F be a function on processes. Regard $F(P)$ as an assembly into which P has been plugged. We would like that replacement of P by a better component can only improve the assembly. For this, F must be monotonic, i.e.

$$F(P) \sqsubseteq F(R) \quad \text{whenever} \quad P \sqsubseteq R.$$

All functions defined so far are monotonic.

DISTRIBUTIVITY.

Let F be a monotonic function of processes. Suppose we wish to implement

$$F(P) \sqcap F(Q)$$

An easy way to do this may be first to implement $(P \sqcap Q)$ and then apply F to the result. This is valid only if F is <u>distributive</u>. i.e.

$$F(P) \sqcap F(Q) = F(P \sqcap Q)$$

All functions defined so far are distributive.

RECURSION.

A function F from processes to processes is <u>continuous</u> if for all ascending chains

$$\{P_i \mid i \in \mathbb{N} \ \& \ \forall i: P_i \sqsubseteq P_{i+1}\}$$

$$F(\bigsqcup_i P_i) = \bigsqcup_i F(P_i).$$

If F is continuous, the least solution of

$$p = F(p)$$

is given by $p = \bigsqcup_i F^i(CHAOS_A)$

where F^i is the i-fold composition of F

All functions defined so far are continuous.

THE UNIVERSITY OF WOLLONGONG

DEPARTMENT OF COMPUTING SCIENCE

symposium on

COMMUNICATING SEQUENTIAL PROCESSES

PROGRAMME — Pentagon Lecture Theatre 2.

Saturday March 22, 1980.

8.30 - 9.30	Registration		
9.30 - 10.30		Hoare C.A.R.	CSP Lecture I
10.30 - 11.00	Morning Tea		
11.00 - 12.00		Hoare C.A.R.	CSP Lecture II
12.00 - 13.00		Dromey R.G.	Text Searching
13.00 - 14.00	Lunch		
14.00 - 15.00		Hoare C.A.R.	CSP Lecture III
15.00 - 15.30	Afternoon Tea		
15.30 - 16.30		Stanton R.B.	Primitives for Concurrency
16.30 - 17.30		Barter C.J.	CSP Language and Implementation
17.30 - 18.00	Happy Hour		
18.00 - 21.00	Dinner	Reinfelds J.	Software Science

++++++++++++++++++++++++++++

Sunday March 23, 1980.

9.30 - 10.30		Hoare C.A.R.	CSP Lecture IV
10.30 - 11.00	Morning Tea		
11.00 - 12.00		Hoare C.A.R.	CSP Lecture V
12.00 - 13.00		Mateti P.	Correctness Proof of Indenting Pro-gram
13.00 - 14.00	Lunch		
14.00 - 15.00		Hoare C.A.R.	CSP Lecture VI
15.00 - 15.30	Afternoon Tea		
15.30 - 16.30		Tobias J.M.	Single User Multi Processor Around High Level Language

LIST OF PARTICIPANTS

at the

COMMUNICATING SEQUENTIAL PROCESSES SYMPOSIUM

Name	Affiliation
AGUERO Alex	University of Wollongong
ALLEN Murray w	University of New South Wales
ANDERSON Alastair	University of Wollongong
BAILEY Thomas	University of Wollongong
BARTER Chris J	University of Adelaide
BERTOLDI C	Australian Iron and Steel Pty Ltd
BLAIR John A	
BLATT David	University of Newcastle
BRAND Richard	Australian Iron and Steel Pty Ltd
BURTON Peter	University of Wollongong
CADY John	New South Wales Istitute of Technology
CAMERON Don	CSIRO
CANNING Lyn	University of Melbourne
CARRINGTON David	University of New South Wales
CHORVAT Nuri	Australian Iron and Steel Pty Ltd
CHORVAT Trevor	Australian Iron and Steel Pty Ltd
CLARK Neville	
COLVILLE J	New South Wales Istitute of Technology
CREASY P	Australian National University
CROOT Christopher	University of Wollongong
CUOCO Anthony	Australian Iron and Steel Pty Ltd
DIX Trevor	University of Melbourne
DROMEY R Geoffrey	University of Wollongong
EDWARDS S	Australian National University
ELLIOTT Liz	Australian Iron and Steel Pty Ltd
EVANS David	
FONG Meng Wai	University of Wollongong
GERBER Anthony J	University of Sydney
GERRITY George W	Royal Military College Duntroon
HALIMAH Lillian	Australian Iron and Steel Pty Ltd
HAMMONDS Wal	University of Wollongong
HAYES Ian J	University of New South Wales
HELISTRAND Graham	University of New South Wales
HERBERT G	Australian Iron and Steel Pty Ltd
HEXT Jan B	University of Sydney
HILL Les	University of New South Wales
HOARE C A R Prof	Oxford University
HOWARTH Bruce R	New South Wales Istitute of Technology
HUMPHRIES George	Royal Australian Naval College HMAS Creswell
JAFFAR Joxan	University of Melbourne
JORDAN Julian	University of Wollongong
KANDIL Ashraf El-Sayed	University of Wollongong
KELLY B	Australian Iron and Steel Pty Ltd
KONTOLEON J	University of Wollongong
KUMMERFELD Robert J	University of Sydney
LEONG Andre	Perkin-Elmer
LIONS John	University of New South Wales
LIU Tze-Jian	University of Melbourne
LLOYD John W Dr	Canberra College of Advanced Education
MATETI Prabhaker	University of Melbourne
MAYADAS David	University of Wollongong

MCGRATH Anthony	University of Wollongong
MCKENZIE Lyn	Australian Iron and Steel Pty Ltd
MILLER Richard	University of Wollongong
MOLINARI B	Australian National University
MONTAGNER John	University of Wollongong
MORGAN Carroll	ASCOMP Pty Ltd
MURRAY Bede	Wollongong Institute of Advanced Education
McKERROW Phillip	University of Wollongong
NEALON Ross	University of Wollongong
NICHOLSON Paul	University of Western Australia
NUNAN T	Australian Iron and Steel Pty Ltd
O'HARA Denis	University of Wollongong
ORSZANSKI Roman	University of Adelaide
PIPER Ian	University of Wollongong
PIRIE Ian	University of Wollongong
POOLE David	Australian National University
RAO K R	University of Melbourne
REINFELDS Juris	University of Wollongong
ROBINSON Ken	University of New South Wales
ROBINSON Michael	Australian Iron and Steel Pty Ltd
ROBSON M	Australian National University
ROSE Greg	University of New South Wales
SALVADORI Antonio	University of Wollongong
SAMAŇEK J	University of New South Wales
SAMMUT Claude	University of New South Wales
SANCHEZ David	University of Wollongong
SAUNDERS Munro R	University of New South Wales
SEET Chern	University of Melbourne
SHEPHERD John	University of Melbourne
SORLI Ronald M	New South Wales Istitute of Technology
STAMBUL Linda	Australian Iron and Steel Pty Ltd
STANLEY P	Australian Iron and Steel Pty Ltd
STANTON Robin	Australian National University
STAZIC Don	
TOBIAS Jeffrey	Australian Atomic Energy Commission
TONG John	University of Wollongong
TOROSSIAN James	
VEBEL J	Australian Iron and Steel Pty Ltd
WALSH Peter	University of Wollongong
WATKIN J	Australian Iron and Steel Pty Ltd
WILSON Lawrence	University of Wollongong
WISE Michael	University of New South Wales
WISHART P	Australian National University
YASTRUBOFF Mike	Australian Iron and Steel Pty Ltd
YUDKIN M	Australian National University
ZOSZAK Kas P	University of Wollongong