Additional notes on a model for communicating sequential processes

C. A. R. Hoare

*University of Wollongong*

Follow this and additional works at: [https://ro.uow.edu.au/compsciwp](https://ro.uow.edu.au/compsciwp)

**Recommended Citation**


Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au
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.**

- **I** is associative
  \[ P \square (Q \square R) = (P \square Q) \square R \]

- **I** is commutative
  \[ P \square Q = Q \square P \]

- **I** is idempotent
  \[ P \square P = P \]

- **I** has unit; and "zero":
  \[ P \square \text{ABORT} = P \]
  \[ P \square (\overline{P})^* = (\overline{P})^* \]

- **II** is associative
  \[ P \ll (Q \ll R) = (P \ll Q) \ll R \]

- **II** is commutative
  \[ P \ll Q = Q \ll P \]

- **II** has unit; and zero:
  \[ P \ll (\overline{P})^* = P \]

- **;** is associative
  \[ P ; (Q ; R) = (P ; Q) ; R \]

- **;** has unit: and zero:
  \[ \text{SKIP} ; P = P \]

- **;** distributes:
  \[ \text{ABORT} ; P = \text{ABORT} \]
  \[ (a \rightarrow P) ; Q = a \rightarrow (P ; Q) \]

- **»** is associative
  \[ P » (Q » R) = (P » Q) » R \]

- **»** distributes:
\[(\text{ABORT} \gg \text{ABORT}) = (\text{ABORT} \gg (? x : T \rightarrow Q(x))) \]

\[= ((! x ; P) \gg \text{ABORT}) = \text{ABORT} \]

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

\[(! v ; P) \gg (! x ; Q) = ! x ; ((! v ; P) \gg Q) \quad \text{for } v \in T.\]

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

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

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

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

\[\emptyset (! x ; ((? v : S \rightarrow P(v)) \gg Q)) \]

\[(? v : S \rightarrow P(v)) = (? v : S \rightarrow P(w)) \]

\[(? v : \{\} \rightarrow P(v)) = \text{ABORT}.\]
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 \( \leq \), with ABORT as bottom and \( A^* \) as top.

Def. A function \( F \) from processes to processes is **distributive** if for all sets \( \{ P_i \mid i \in T \} \):

\[
F(\bigcup_{i \in T} P_i) = \bigcup_{i \in T} F(P_i)
\]

Thm. \( \rightarrow, [], \|, ;, m:, \) are distributive (and hence continuous and monotonic) in each of their arguments.
hm (Tarski, Scott). If \( F \) is continuous, then the least \( p \) satisfying
\[
p = F(p) = (\ldots \ldots p; 1; p; \ldots \ldots p)
\]
is
\[
\bigcup_{i \in \mathbb{N}} F^i(\text{ABORT}) \quad = \mu_p. F(p)
\]
where
\[
F^0(q) = q \\
F^{n+1}(q) = F^n(F(q))
\]

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 \( \text{RHS} = F \left( \bigcup_{i \in \mathbb{N}} F^i(\text{ABORT}) \right) = \bigcup_{i \in \mathbb{N}} F(F^i(\text{ABORT})) \) [continuity]
\[
= \bigcup_{i \in \mathbb{N}} F^{i+1}(\text{ABORT}) \quad \text{[def \( F^{i+1} \)}
\]
\[
= \text{ABORT} \cup \bigcup_{i > 0} F^i(\text{ABORT}) \quad \text{[ABORT \ \subseteq \ \text{any}}
\]
\[
= \bigcup_{i \in \mathbb{N}} F^i(\text{ABORT})
\]
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 unique.

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 $p = F^n(p)$ for all $m$, $s \in F^n(p)$, so $s \in F^n(q)$. Thus $p \leq q$. Similarly, $q \leq p$.

$\therefore p = q$

The same is true for sets of mutually recursive equations.
RECURSION  INDUCTION

COUNT₀ = (iszero → COUNT₀ ⊕ up → COUNT₁)
COUNT = (down → COUNTₙ ⊕ up → COUNTₙ₊₂)
POS = (down → SKIP ⊕ up → POS; POS)
ZERO = (iszero → ZERO ⊕ up → POS; ZERO).

Theorem. ZERO = COUNT₀.

Proof. Define C₀ = (iszero → C₀ ⊕ up → C₂)
Cₙ₊₁ = POS; Cₙ
Cₙ₊₁ = (down → SKIP; Cₙ ⊕ up → (POS; POS); Cₙ) & distr;
= (down → Cₙ ⊕ up → POS; (POS; Cₙ)) & prop;
= (down → Cₙ ⊕ up → POS; Cₙ₊₁) & def Cₙ₊₂;
= (down → Cₙ ⊕ up → Cₙ₊₂) & def Cₙ₊₂

∴ Cₙ = COUNTₙ  for all n.  ... (1)

but  C₀ = (iszero → C₀ ⊕ up → POS; C₀)
∴  C₀ = ZERO

Conclusion follows from (1), (2).
YET ANOTHER COUNT.

\[ Z = \text{iszero} \rightarrow Z \uplus up \rightarrow (p : Z \parallel X); Z \]

where \( X = (up \rightarrow p \uplus up \rightarrow X \)
\[ \uplus \text{down} \rightarrow (p \text{ iszero } \rightarrow \text{SKIP} \]
\[ \uplus p . \text{down} \rightarrow X \]
\[ ) \]

Theorem. \( Z = \text{ZERO} \)
RELATIONS.

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

\[ P \text{ sat } R = \exists \sigma (s \in P \Rightarrow (\text{ins}(s), \text{outs}(s)) \in R) \]

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

*Example*

let \( f : \) be a monotonic function of traces.

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

If \( P \text{ sat } R_f \), \( P \) is said to be a pipe for \( f \).

We shall often represent \( R \) as a predicate
on the variables "in" and "out".

*Example*

\( P \text{ sat } (\text{out} \leq \text{in}) = P \text{ sat } \{(\text{in}, \text{out}) | \text{out} \leq \text{in}\} \)

means that \( P \) is a buffer

i.e., a pipe for the identity function.

**Theorem.** \( Q \text{ sat } R \& R \subseteq S \Rightarrow Q \text{ sat } S \)

\[ (\forall i \ Q_i \text{ sat } R) \Rightarrow (\bigcup_i Q_i) \text{ sat } R \]

If \( \text{ABORT} \text{ sat } R \& \forall p. \ p \text{ sat } R \Rightarrow F(p) \text{ sat } R \)
then \( (\forall p. F(p)) \text{ sat } R. \) (Fixed point induction)
1. \textbf{ABORT} \texttt{sat } R = R \text{ in } \text{ out} \\
2. (\forall x; P) \texttt{sat } R \equiv P \texttt{sat } R_{\text{out}}^\text{out} \\
   \text{where } R_{\text{out}}^\text{out} = \{(i, o) \mid (i, \langle x \rangle o) \in R\} \\
   \text{i.e. replace } "\text{out}" \text{ by } "\langle x \rangle \text{out}" \text{ in } R \\
3. (\forall x: T \to P(x)) \texttt{sat } R \equiv \forall x: T. (P(x) \texttt{sat } R_{\text{in}}^\text{in}) \\
   \text{i.e. it works for all input values.} \\
\textbf{by } 2. (\forall x; B) \texttt{sat } \text{ out } \leq \langle x \rangle \text{ in} \equiv B \texttt{sat } \langle x \rangle \text{out } \leq \langle x \rangle \text{ in} \\
   \equiv B \texttt{sat } \text{ out } \leq \text{ in.} \\
\textbf{by } 3. (\forall x: T \to !x; B) \texttt{sat } \text{ out } \leq \text{ in} \\
   \equiv \forall x: T. (!x; B) \texttt{sat } \text{ out } \leq \langle x \rangle \text{ in} \\
   \equiv B \texttt{sat } \text{ out } \leq \text{ in} \quad \text{(just proved)} \\
\textbf{by } 2. \text{ ABORT} \texttt{sat } \text{ out } \leq \text{ in.} \quad \text{(because } \text{ outs}(\langle \rangle) = \text{ ins}(\langle \rangle) \Rightarrow \langle \rangle) \\
\therefore \text{ if } B = \text{df. } (\forall x: T \to (!x; B)) \\
\text{ then } B \text{ is a buffer. (Fixed point induction)
\[ T = \text{outs}(\alpha P) = \text{ins}(\alpha Q) \]
\[(P \triangledown R) \land (Q \triangledown S) \Rightarrow \]
\[(P \triangleright Q) \triangledown \exists s \ (R_{s}^{\text{out}} \land S_{s}^{\text{in}}) \]

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

\[ \text{Proof: } T = (P \triangleright Q), \ P \triangledown R, \ Q \triangledown S \quad \text{— assume} \]
\[ \exists u \in T \quad \& \quad \exists v \in P \ \exists w \in Q \]
\[ \text{ins}(u) = \text{ins}(v) \quad \& \quad \text{outs}(u) = \text{outs}(v) = \text{outs}(t) \]
\[ \text{ins}(v) = \text{ins}(w) = \text{out}(t) \]
\[ \therefore (\text{ins}(v), \text{out}(v)) \in R \quad \& \quad (\text{ins}(w), \text{out}(w)) \in S \]
\[ \Rightarrow (\text{ins}(v), \text{out}(v)) \in R \quad \& \quad (\text{out}(v), \text{out}(w)) \in S \]
\[ \Rightarrow \exists s \ (\text{ins}(t), s) \in R \quad \& \quad (s, \text{out}(t)) \in S \]
\[ \Rightarrow (\text{ins}(b), \text{out}(b)) \in (R; S). \]
\[ f \text{ and } g \text{ are monotonic:} \\
(P \text{ sat outs} \leq f(\text{ins})) \& (Q \text{ sat outs} \leq g(\text{ins})) \\
\Rightarrow (P \gg Q) \text{ sat outs} \leq f(\text{ins}) \& \text{ outs} \leq g(\text{ins}) \]

\Rightarrow (P \gg Q) \text{ sat outs} \leq g(f(\text{ins}))

therefore, if \( P \) is a pipe for \( f \) and \( Q \) for \( g \)

then \( P \gg Q \) is a pipe for \( g f \).

If \( P \) and \( Q \) are buffers, so is \( P \gg Q \)

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

since \( B_1 = (\forall x: T \to !x; B_1) \) is a buffer.

so is \( B_{n+1} = B_n \gg B_2 \) for all \( n \geq 1 \).

Proof: induction on \( n \)

WARNING: \text{ sat } \text{ defines only a form of partial correctness} \text{ does not prove absence of deadlock, e.g. the following are buffers.} 

\text{ABORT,} \\
\text{; (\forall x: \{3\} \to !x; B_1) \& B_3 \text{, where}} \\
B_3 = (\forall x: T \to (\forall y: T \to !y; \text{ B}_3 \text{, for } x < y))
COMMUNICATIONS.

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

Theorem. If for all \( x : T, P_x \triangleright\triangleright Q_x \) is a buffer, then so is \((\exists x : T \rightarrow (P_x \triangleright\triangleright (\land x ; Q_x))) \) .... (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 \( P_x \triangleright\triangleright Q_x \), which is a buffer.

\[ \therefore \text{outs}(t') \leq \text{ins}(t') \] (3)

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

and \( \text{outs}(t) = \langle \text{first}(\text{outs}(t^2)) \rangle \text{outs}(t) \) (5)

\[ \therefore \text{outs}(t) \leq \text{ins}(t) \]

from \((2, 3, 4, 5)\)
If for all $x \in T$

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

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

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

If $t$ is OK - if outs$(t) \leq$ ins$(t)$, so $t$ is OK.

Assume all $t$ of length $\leq n$ in $P_x \gg Q_x$ are OK (for all $x$).

Now let $t'$ be $P_x \gg Q_x$ be of length $\leq n+1$.

If $t'$ is all inputs, it's OK

Otherwise $t'$ is 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 is $(P_2 \gg P_1) \gg (Q_2 \gg Q_1)$

(Composition of protocols).
Phase encoding.

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

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

\[(O; !1; R) \gg P = ![O; !1; (!!1; R) \gg P] = ![O; !1; !1; !0; (R \gg P) \]

**Theorem.** \( P \gg Q \) is a buffer.

**Proof.** \( P \gg Q = \)

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

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

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

\[ \therefore P \gg Q \text{ is a buffer.} \]
NRZ Protocol

\[ P_0 = \chi : \{0,1\} \to \{0,1\} ; P_x \]
\[ P_1 = \chi : \{0,1\} \to \{0,1\} ; P_x \]

\[ (!1; 0; 0; 1; R) \gg P_0 = !1; ((!0; 0; 1; R) \gg P_1) \]
\[ = !1; !1; ((!0; 1; R) \gg P_0) \]
\[ = !1; !1; !0; ((!1; R) \gg P_0) \]
\[ = !1; !1; !0; !1; (R \gg P_1) \]

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

\[ Q_0 = \chi : \{0,1\} \to \{0,1\} ; Q_x \]
\[ Q_1 = \chi : \{0,1\} \to \{0,1\} ; Q_x \]

\[ (!1; 1; 0; 1; R) \gg Q_0 = !1; !0; !0; !1; (R \gg Q_1) \]

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 $x = 0, 1$

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

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

So $P_y \gg Q_y = (\? x: \{0, 1\} \rightarrow P_x \gg (\! x; Q_{\neq}))$
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 \( R = (x \rightarrow a \rightarrow P \parallel y \rightarrow b \rightarrow Q) \setminus \{x, y\} \)

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, \( R = (a \rightarrow P \parallel b \rightarrow Q) \), 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 \( \text{traces}(P) \) be the subset of \( A^* \) denoting traces of the possible behaviours of \( P \).

So \( \text{traces}(P) \) is nonempty & prefix-closed.

Define \( P^0 = \{a | \langle a \rangle \in \text{traces}(P)\} \)

\( P^0 \) 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 environment of $P$.

"$P$ can refuse $X$" means that $P$ can deadlock in this environment.

So $P$ can refuse $\emptyset$

$P$ can refuse $X \Rightarrow P$ can refuse $X \cup Y$

$P$ can refuse $X \Rightarrow P$ can refuse $X \cup (A - P^o)$.

$(A - P^o)$ is a set which $P$ must refuse.

Let $s$ be in traces $(P)$. Then "$P_{after}$" denotes the future behaviour of $P$ if $s$ is a trace of its past behaviour.

So $P_{after} < > = P$

$P_{after} st = (P_{after} s) after t$
Proposition: A process is defined by what it can do and what it can refuse.

So if \( \text{alphabet}(P) = \text{alphabet}(Q) \)
and \( \text{traces}(P) = \text{traces}(Q) \)
and \( \forall X \ (P \text{ canrefuse } X \equiv Q \text{ canrefuse } X) \)
and \( \forall a \ (a \in P^0 \Rightarrow P \text{ after } a\downarrow = Q \text{ after } a\downarrow) \)
then \( P = Q \)

We therefore define a process \( P \) as a relation:

For \( s \) in \( A^* \) and \( X \subseteq A \)
\( (s, X) \in P \) means \( s \in \text{traces}(P) \& (P \text{ after } s) \text{ canrefuse } X \).

So \( \text{traces}(P) = \text{df} \{ s \mid (s, \emptyset) \in P \} \)
\( P \text{ canrefuse } X = \text{df} \ (\emptyset, X) \in P \)
\( P \text{ after } s = \text{df} \ \{ (t, X) \mid (st, X) \in P \} \)
\( \text{traces}(P) \) must be nonempty & prefix-closed
\{ \emptyset \} \text{canrefuse } X \} must be nonempty & left closed
and closed under union with \( \text{traces}(P) \)
EXAMPLES. with alphabet \( A \).

\( \text{STOP}_A \) can't do anything
must refuse everything.

\( \text{STOP}_A = \{ (\_ , X) \mid X \subseteq A \} \)

\( \text{RUN}_A \) can do anything
can't refuse anything.

\( \text{RUN}_A = \{ (s, \{ ? \}) \mid s \in A^* \} \)

\( \text{CHAOS}_A \) can do anything
can refuse anything.

\( \text{CHAOS}_A = \{ (\_ , X) \mid A^* \text{ & } X \subseteq A \} \)

for all \( s \in A^* \):
\( \text{RUN}_A \text{ after } s = \text{ RUN}_A \)
\( \text{CHAOS}_A \text{ after } s = \text{ CHAOS}_A \).
Let \( F \) be a function from \( A \) to processes.
Let \( B \subseteq A \). Then

\[
(x : B \rightarrow F(x)) \quad \text{first accepts any } x \text{ in } B,
\]

and then behaves like \( F(x) \)

\[
(x : B \rightarrow F(x)) = \text{df } \{(\langle x \rangle, X) \mid X \subseteq A - B\} \\
\cup \{(x : s, X) \mid x \in B \land (s, X) \in F(x)\}
\]

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

\( (x : B \rightarrow F(x)) \text{ after } b = F(b) \quad \text{for all } b \in B. \)

\( (x : \{x\} \rightarrow F(x)) = \text{STOP}_A \)

\( (x : B \rightarrow F(x)) = (y : B \rightarrow F(y)) \)
PARALLEL COMPOSITION.

P and Q have same alphabet A.
P I I Q can accept anything acceptable to both P and Q
and if P can refuse X and Q can refuse Y, P I I Q can refuse X u Y

\[(P \parallel Q) = \text{df. } \{ (s, X \cup Y) \mid (s, X) \in P \land (s, Y) \in Q \} \]

\[\text{traces } (P \parallel Q) = \text{traces } (P) \cap \text{traces } (Q) \]

\[(P \parallel Q)_{\text{after } s} = (P_{\text{after } s}) \parallel (Q_{\text{after } s}) \text{ for } s \in \text{traces } (P \parallel Q)\]

\[\parallel \text{ is associative and commutative,} \]

with unit \(\text{RUN}_A\) and zero \(\text{STOP}_A\).

\[\langle x : B \rightarrow F(x) \rangle \parallel \langle y : C \rightarrow G(y) \rangle = \langle z : B \cap C \rightarrow (F(z) \parallel G(z)) \rangle\]
**NONDETERMINISM.**

\( P \cap Q \) behaves non-deterministically, either like \( P \) or like \( Q \). It can do anything that \( P \) or \( Q \) can do. It can refuse anything that \( P \) or \( Q \) can refuse.

\[ P \cap Q = \text{df} \ P \cup Q \]

\( \cap \) is associative, commutative, and idempotent with zero \( \text{CHAOS}_A \).

\[ \text{traces} \ (P \cap Q) = \text{traces} \ (P) \cup \text{traces} \ (Q) \]

\[ (P \cap Q)_{\text{after}} s = P_{\text{after}} s \quad \text{if} \ s \in \text{traces} \ (P) - \text{traces} \ (Q) \]

\[ = Q_{\text{after}} s \quad \text{if} \ s \in \text{traces} \ (Q) - \text{traces} \ (P) \]

\[ = (P_{\text{after}} s) \cap (Q_{\text{after}} s) \quad \text{if} \ s \in \text{traces} \ (P) \cap \text{traces} \ (Q) \]

If we admit the **EMPTY** relation as a process, it will be the unit of \( \cap \).
P \uplus Q \text{ behaves like } P \text{ or like } Q; \text{ the choice can be influenced by its environment, but only on the first step.}

It can do anything \( P \) or \( Q \) can do.
It can refuse anything that both \( P \) and \( Q \) can refuse.

\[
P \uplus Q = \{ (\langle \cdot, X \rangle \mid (\langle \cdot, X \rangle \in P \cap Q)
\cup \{ (s, X) \mid s \neq \langle \cdot \rangle \& (s, X) \in (P \cup Q)\}
\]

\( \uplus \) is associative, commutative and idempotent with unit \( \text{STOP} \).

\[
\text{traces } (P \uplus Q) = \text{traces } (P) \cup \text{traces } (Q)
\]

\[
(x: B \to F(b)) \uplus (y: C \to G(c)) = (z: B \cup C \to
\begin{cases}
F(z) & \text{if } z \in B - C \\
G(z) & \text{if } z \in C - B \\
F(z) \cap G(z) & \text{else}
\end{cases}
\]

LIMITS.

$P \leq Q$ means $Q$ is more deterministic than $P$, and therefore more predictable, controllable, useful. Everything $Q$ can do, so can $P$. Everything $Q$ can refuse, so can $P$.

$P \leq Q \iff Q \subseteq P \quad \text{or} \quad P \cap Q = P$

e.g. $P \cap Q \subseteq P \cap Q$

$\text{CHAOS}_A \subseteq P$

$P \cap (P \cap Q \cap R) \subseteq P \cap Q$.

If $P_i \in P_i$ for all $i$, then we write

$$\bigcup_i P_i = \downarrow \bigcap_i P_i$$

The relation $E$ is a complete partial order with $\text{CHAOS}_A$ as its bottom.

If we add $\text{EMPTY}$, processes form a complete lattice with $\text{EMPTY}$ as an isolated top.
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 $f(X) = \{f(x) | x \in X\}$

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

$\text{traces}(F^{-1}(P)) = \{s | f(s) \in \text{traces}(P)\}$

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

$(F^{-1}(P))_{\text{after } s} = F^{-1}(P_{\text{after } f(s)})$

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

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

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

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

Let $P$ be a process with alphabet $A$ and $P \preceq B$ 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.

$$P \preceq B = \{(s, X) \mid s \in (A \cup B)^* \text{ and } (s_A, X) \in P\}$$

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

$$(P \preceq B)^o = P^o \cup (B - A)$$

$$(P \preceq B) \text{ canrefuse } X \equiv P \text{ canrefuse } X$$

$$(P \preceq B) \text{ after } s = (P \text{ after } s_A) \preceq B$$

$$(P \preceq A = P, (P \preceq B) \preceq C = P \preceq (B \cup C)$$

If $Q$ has alphabet $B$, then

$$P || Q = \text{if } (P \preceq B) || (Q \preceq A)$$

$||$ is associative and commutative, etc.
Let $P$ have alphabet $A$. Let $B$ be a set of events to be regarded as internal to $P$. Then $P \setminus B$ 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 $P$.

$$(P \setminus B)^* \supseteq P^* - B$$

$P$ can refuse $X$ & $X \cap B = \emptyset \Rightarrow (P \setminus B)$ can refuse $X$

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

$(P \setminus B)$ after $s_{A-B} \equiv (P$ after $s) \setminus B$.

These properties are satisfied by

$$[P \setminus B] = \{(s_{A-B}, X) \mid X \cap B = \emptyset \land (s, X \cup B) \in P\}$$

But $\{(b^n, X) \mid X \subseteq \{\alpha\}\ \& \ {\text{not a process}}\}$
The trouble lies in the infinite trace consisting of hidden symbols. The process may choose to follow this path forever and never 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.

\[ P \setminus B = \left[ P \setminus B \right] \cup \{(st, X) \mid \{u \mid u \in P \& u_{R-B} = s\} \text{ is infinite}\} \]

\[ P \setminus \emptyset = P \]

\[ (P \setminus B) \setminus C = P \setminus (B \cup C) \]

**NOTE** — we rely on finitude of alphabets
MONOTONICITY

PER 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.

\[ F(P) \leq F(R) \text{ whenever } \text{PER}. \]

All functions defined so far are monotonic.
DISTRIBUTIVITY.

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

$$F(P) \cap F(Q)$$

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

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

All functions defined so far are distributive.
RECURSION.

A function $F$ from processes to processes is **continuous** if for all ascending chains

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

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

If $F$ is continuous, the least solution of

$$p = F(p)$$

is given by

$$p = \bigsqcup_i F^i(\text{CHAOSEP}_p)$$

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

All functions defined so far are continuous.
### PROGRAMME

**Pentagon Lecture Theatre 2.**

Saturday March 22, 1980.

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker</th>
<th>Lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.30 - 9.30</td>
<td>Registration</td>
<td></td>
<td>CSP Lecture I</td>
</tr>
<tr>
<td>9.30 - 10.30</td>
<td>Hoare C.A.R.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.30 - 11.00</td>
<td>Morning Tea</td>
<td>Hoare C.A.R.</td>
<td>CSP Lecture II</td>
</tr>
<tr>
<td>11.00 - 12.00</td>
<td>Hoare C.A.R.</td>
<td>Dromey R.C.</td>
<td>Text Searching</td>
</tr>
<tr>
<td>12.00 - 13.00</td>
<td>Lunch</td>
<td>Hoare C.A.R.</td>
<td>CSP Lecture III</td>
</tr>
<tr>
<td>13.00 - 14.00</td>
<td>Afternoon Tea</td>
<td>Stanton R.B.</td>
<td>Primitives for Concurrency</td>
</tr>
<tr>
<td>14.00 - 15.00</td>
<td>Happy Hour</td>
<td>Barter C.J.</td>
<td>CSP Language and Implementation</td>
</tr>
<tr>
<td>15.00 - 15.30</td>
<td>Dinner</td>
<td>Reinfelds J.</td>
<td>Software Science</td>
</tr>
<tr>
<td>15.30 - 16.30</td>
<td>Afternoon Tea</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker</th>
<th>Lecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.30 - 10.30</td>
<td>Hoare C.A.R.</td>
<td></td>
<td>CSP Lecture IV</td>
</tr>
<tr>
<td>10.30 - 11.00</td>
<td>Morning Tea</td>
<td>Hoare C.A.R.</td>
<td>CSP Lecture V</td>
</tr>
<tr>
<td>11.00 - 12.00</td>
<td>Mateti P.</td>
<td></td>
<td>Correctness</td>
</tr>
<tr>
<td>12.00 - 13.00</td>
<td>Lunch</td>
<td>Hoare C.A.R.</td>
<td>Proof of Indenting Program</td>
</tr>
<tr>
<td>13.00 - 14.00</td>
<td>Afternoon Tea</td>
<td>Tobias J.M.</td>
<td>Single User</td>
</tr>
<tr>
<td>14.00 - 15.00</td>
<td></td>
<td></td>
<td>Multi Processor</td>
</tr>
<tr>
<td>15.00 - 15.30</td>
<td></td>
<td></td>
<td>Around High</td>
</tr>
<tr>
<td>15.30 - 16.30</td>
<td></td>
<td></td>
<td>Level Language</td>
</tr>
</tbody>
</table>
# LIST OF PARTICIPANTS

at the

COMMUNICATING SEQUENTIAL PROCESSES SYMPOSIUM

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>AGUERO Alex</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>ALLEN Murray W</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>ANDERSON Alastair</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>BAILEY Thomas</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>BARTEL Chris J</td>
<td>University of Adelaide</td>
</tr>
<tr>
<td>BERTOLDI C</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>BLAIR John A</td>
<td>University of Newcastle</td>
</tr>
<tr>
<td>BLATT David</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>BRAND Richard</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>BURTON Peter</td>
<td>New South Wales Institute of Technology</td>
</tr>
<tr>
<td>CADDY John</td>
<td></td>
</tr>
<tr>
<td>CAMERON Don</td>
<td>CSIRO</td>
</tr>
<tr>
<td>CANNING Lyn</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>CARRINGTON David</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>CHORVAT Nuri</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>CHORVAT Trevor</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>CLARK Neville</td>
<td></td>
</tr>
<tr>
<td>COLVILLE J</td>
<td>New South Wales Institute of Technology</td>
</tr>
<tr>
<td>CREASY P</td>
<td>Australian National University</td>
</tr>
<tr>
<td>CROOK Christopher</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>CUOCO Anthony</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>DIX Trevor</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>DROEY R Geoffrey</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>EDWARDS S</td>
<td>Australian National University</td>
</tr>
<tr>
<td>ELLIOTT Liz</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>EVANS David</td>
<td></td>
</tr>
<tr>
<td>FONG Meng Wai</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>GERRER Anthony J</td>
<td>University of Sydney</td>
</tr>
<tr>
<td>GERRITY George W</td>
<td>Royal Military College Duntroon</td>
</tr>
<tr>
<td>HALIMAH Lillian</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>HAMMONDS Wai</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>HAYES Jan J</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>HELISTRAND Graham</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>HERBERT C</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>HEXT Jan R</td>
<td>University of Sydney</td>
</tr>
<tr>
<td>HILL Lee</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>HOARE C A R Prof</td>
<td>Oxford University</td>
</tr>
<tr>
<td>HOWARTH Bruce R</td>
<td>New South Wales Institute of Technology</td>
</tr>
<tr>
<td>HUMPHREYS George</td>
<td>Royal Australian Naval College HMAS Creswell</td>
</tr>
<tr>
<td>JAFFAR Jozan</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>JORDAN Julian</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>KANDIL Ashraf FI-Sayed</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>KELLY R</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>KONTOLEON J</td>
<td>University of Sydney</td>
</tr>
<tr>
<td>KUMMERFELD Robert J</td>
<td>University of Sydney</td>
</tr>
<tr>
<td>LEONG Andre</td>
<td>Perkin-Elmer</td>
</tr>
<tr>
<td>LIONS John</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>LIU Tze-Jian</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>LLOYD John W Dr</td>
<td>Canberra College of Advanced Education</td>
</tr>
<tr>
<td>MATETI Prabhaker</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>MAYAPAS David</td>
<td>University of Wollongong</td>
</tr>
</tbody>
</table>
MCGRATH Anthony
MCKENZIE Lyn
MILLER Richard
MOLINARI R
MONTAGNER John
MOUGAN Carroll
MURRAY Bede
McKERRAW Phillip
NEALON Ross
NICHOLSON Paul
NUFFAN T
O’HARA Denis
ORSZANSKI Roman
PIPER Ian
PIRIE Ian
POOLE David
RAO K R
REINFELDS Juris
ROBINSON Ken
ROBINSON Michael
ROBSON M
ROSE Greg
SALVADORI Antonio
SAHANEK J
SAMMUT Claude
SANCHEZ David
SAUNDERS Munro R
SEPT Chern
SHEPHERD John
SOPLI Ronald M
STAMRUL Linda
STANLEY P
STANTON Robin
STAZIC Don
TODIAS Jeffrey
TOFC John
TOROSSIAN James
VEREL J
WALSH Peter
WATKIN J
WILSON Lawrence
WISE Michael
WISHART P
YASTRUBOFF Mike
YUDKIN M
ZOSZAK Kas P

University of Wollongong
Australian Iron and Steel Pty Ltd
University of Wollongong
Australian National University
University of Wollongong
ASCOMP Pty Ltd
Wollongong Institute of Advanced Education
University of Wollongong
University of Wollongong
University of Western Australia
Australian Iron and Steel Pty Ltd
University of Wollongong
University of Melbourne
University of Wollongong
University of New South Wales
University of Wollongong
Australian National University
University of New South Wales
University of Melbourne
New South Wales Institute of Technology
Australian Iron and Steel Pty Ltd
Australian Iron and Steel Pty Ltd
Australian National University
Australian Atomic Energy Commission
University of Wollongong
Australian Iron and Steel Pty Ltd
University of Wollongong
Australian Iron and Steel Pty Ltd
University of New South Wales
Australian National University
Australian Iron and Steel Pty Ltd
Australian National University
University of Wollongong