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.**

- **&** is associative
  
  
  \[ P \& (Q \& R) = (P \& Q) \& R \]
  
  \[ P \& Q = Q \& P \]
  
  \[ P \& P = P \]
  
  \[ P \& \text{ABORT} = P \]
  
  \[ P \& (\overline{P})^* = (\overline{P})^* \]

- **||** is associative
  
  \[ P || (Q || R) = (P || Q) || R \]
  
  \[ P || Q = Q || P \]
  
  \[ P || (\overline{P})^* = P \]

- **;** is associative
  
  \[ P ; (Q ; R) = (P ; Q) ; R \]
  
  SKIP; P = P
  
  ABORT; P = ABORT
  
  \[ (a \rightarrow P); Q = a \rightarrow (P ; Q) \]
  
  \[ (P ; Q); R = (P ; R) ; (Q ; R) \]

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

  and distributes:

  \[ P ; (Q ; R) = (P ; Q) ; R \]
  
  SKIP; P = P
  
  ABORT; P = ABORT
  
  \[ (a \rightarrow P); Q = a \rightarrow (P ; Q) \]
  
  \[ (P ; Q); R = (P ; R) ; (Q ; R) \]
  
  \[ P >> (Q >> R) = (P >> Q) >> R \]
\[(\text{ABORT} \gg \text{ABORT}) = (\text{ABORT} \gg (?) \in T \rightarrow Q(\in))\]

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

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

\[(!v \; ; P) \gg (!x \; ; Q) = !x \; ; ((!v \; ; P) \gg Q)\]

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

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

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

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

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

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

\[(?v : \{} \rightarrow P(v)) = \text{ABORT}.\]
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 \( U_{i \in T} P_i \). 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 \( \{P_i \mid i \in T\} \)

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

Thm. \( \rightarrow, \emptyset, \parallel, ;, \mathsf{m;} \) are distributive (and hence continuous and monotonic) in each of their arguments.
hm (Tarski, Scott). If \( F \) is continuous, then the least \( p \) satisfying

\[
p = F(p) = (\ldots; p; \ldots; p; \ldots; p)\]

is \( \bigcup_{i \in \mathbb{N}} F^i(\text{ABORT}) = \text{lim} \; \tau. F(p) \)

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: \( \text{RHS} = F \left( \bigcup_{i \in \mathbb{N}} F^i(\text{ABORT}) \right) = \bigcup_{i \in \mathbb{N}} F \left( F^i(\text{ABORT}) \right) \) [continuity]

\[
= \bigcup_{i \in \mathbb{N}} F^{i+1}(\text{ABORT}) \quad \text{[def } F^{i+1} \text{]}
\]

\[= \text{ABORT} \cup \bigcup_{i > 0} F^i(\text{ABORT}) \quad \text{[ABORT } \leq \text{ any]}
\]

\[= \bigcup_{i \in \mathbb{N}} F^i(\text{ABORT}),\]
Unique Fixed Points

If $F(p)$ is an expression in which $p$ appears only to the right of $\rightarrow$, and $F$ does not contain localization, 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, $q \leq p$. So $p = q$.

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

\[
\text{COUNT}_0 = (\text{iszero} \rightarrow \text{COUNT}_0 \uparrow \text{up} \rightarrow \text{COUNT}_1)
\]

\[
\text{COUNT} = (\text{down} \rightarrow \text{COUNT}_n \uparrow \text{up} \rightarrow \text{COUNT}_{n+2})
\]

\[
\text{POS} = (\text{down} \rightarrow \text{SKIP} \uparrow \text{up} \rightarrow \text{POS}; \text{POS})
\]

\[
\text{ZERO} = (\text{iszero} \rightarrow \text{ZERO} \uparrow \text{up} \rightarrow \text{POS}; \text{ZERO}).
\]

Theorem. \text{ZERO} = \text{COUNT}_0.

Proof. Define \[
\text{C}_0 = (\text{iszero} \rightarrow \text{C}_0 \uparrow \text{up} \rightarrow \text{C}_2)
\]

\[
\text{C}_{n+1} = \text{POS}; \text{C}_n
\]

\[
\text{C}_{n+1} = (\text{down} \rightarrow \text{SKIP}; \text{C}_n \uparrow \text{up} \rightarrow (\text{POS}; \text{POS}); \text{C}_n) \quad \text{def} \text{C}_{n+1}
\]

\[
= (\text{down} \rightarrow \text{C}_n \uparrow \text{up} \rightarrow \text{POS}; (\text{POS}; \text{C}_n)) \quad \text{prop;}
\]

\[
= (\text{down} \rightarrow \text{C}_n \uparrow \text{up} \rightarrow \text{POS}; \text{C}_{n+1}) \quad \text{def} \text{C}_{n+1}
\]

\[
= (\text{down} \rightarrow \text{C}_n \uparrow \text{up} \rightarrow \text{C}_{n+2}) \quad \text{def} \text{C}_{n+2}
\]

\[
\therefore \text{C}_n = \text{COUNT}_n \quad \text{for all } n. \quad \ldots \ (1)
\]

but \[
\text{C}_0 = (\text{iszero} \rightarrow \text{C}_0 \uparrow \text{up} \rightarrow \text{POS}; \text{C}_0)
\]

\[
\therefore \text{C}_0 = \text{ZERO} \quad \ldots \ (2)
\]

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

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

where \( X = (\text{up} \rightarrow p.\text{up} \rightarrow X \)

\( \text{down} \rightarrow (p.\text{iszero} \rightarrow \text{SKIP} \)

\( p.\text{down} \rightarrow X \)

\)

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

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

\[ P \text{ sat } R = \exists s (s \circ P \Rightarrow (\text{ins}(s), \text{oute}(s)) \circ R) \]

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

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

e.g. \( P \text{ sat } (\text{oute} \leq \text{in}) = P \text{ sat } \{\text{(in, out)} \mid \text{out} \leq \text{in}\} \)

- means that \( P \) is a buffer
  i.e. a pipe for the identity function.

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

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

If \( \text{ABORT sat } R \land \forall p. \ p \text{ sat } R \Rightarrow F(p) \text{ sat } R \)

then \( (\forall p. F(p)) \text{ sat } R. \) (fixed point induction)
1. \textbf{ABORT} \text{ sat } R = R \text{ in out }

2. \((!x ; P) \text{ sat } R = P \text{ sat } R^{\text{out}}\)
   \begin{align*}
   \text{where } R^{\text{out}} &= \{(i, o) \mid (i, <x> o) \in R\} \\
   \text{i.e. replace "out" by "<x>out" in } R
   \end{align*}

3. \((?x : T \rightarrow P(x)) \text{ sat } R = \forall x : T. (P(x) \text{ sat } R^{\text{in}})\)
   \begin{align*}
   \text{i.e. it works for all input values.}
   \end{align*}

4. \((!x ; B) \text{ sat } \text{ out } \leq <x> \text{ in }) \equiv B \text{ sat } <x> \text{ out } \leq <x> \text{ in }
   \equiv B \text{ sat } \text{ out } \leq \text{ in.}

5. \((?x : T \rightarrow !x ; B) \text{ sat } \text{ out } \leq \text{ in })
   \begin{align*}
   &\equiv \forall x : T. (!x ; B) \text{ sat } \text{ out } \leq <x> \text{ in }
   \equiv B \text{ sat } \text{ out } \leq \text{ in } \quad \text{(just proved)}
   \end{align*}

6. \text{by (4) } \text{ ABORT sat } \text{ out } \leq \text{ in.} \quad \text{(because outs (<>) = ins (<>) = <>)}

   \text{if } B = \text{df. } (?x : T \rightarrow (!x ; B))

   \text{then } B \text{ is a buffer.} \quad \text{(founded point induction)}
\[ T = \text{outs}(\alpha P) = \overline{\text{ins}}(\alpha Q) \]

\[(P \setminus R) \land (Q \setminus S) \Rightarrow \]
\[(P \Rightarrow Q) \setminus \exists s \in (R^+ \setminus S) \]

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

\[ \text{prove } T = (P \Rightarrow Q), \; P \setminus R, \; Q \setminus S \quad \text{— assume} \]

\[ \exists u \in T \; \land \; \exists v \in P \; \exists w \in Q \]
\[ \text{ins}(v) = \overline{\text{ins}}(u) \; \land \; \text{outs}(u) = \text{outs}(v) = \text{outs}(w) \]
\[ \land \; \text{ins}(v) = \text{ins}(u) = \overline{\text{ins}}(T) \]

\[ \Rightarrow (\text{ins}(v), \text{outs}(v)) \circ R \; \land \; (\text{ins}(v), \text{outs}(w)) \circ S \]
\[ \Rightarrow (\text{ins}(v), \overline{\text{ins}}(T)) \circ R \; \land \; (\overline{\text{ins}}(T), \text{outs}(w)) \circ S \]
\[ \Rightarrow \exists s \in (\text{ins}(v), T) \circ R \; \land \; (s, \text{outs}(w)) \circ S \]
\[ \Rightarrow (\text{ins}(v), \text{outs}(w)) \circ (R ; S). \]
If \( f \) and \( g \) are monotonic:

\[(P \text{ sat outs} \leq f(\text{ins})) \land (Q \text{ sat outs} \leq g(\text{ins}))\]

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

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 \( B_1 = (?x : T \rightarrow !x ; B_1) \) is a buffer.

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

Proof: induction on \( n \)

**WARNING** sat defines only a form of partial correctness - does not prove absence of deadlock. e.g. the following are buffers.

**ABORT**, \( ; (?x : \{33 \rightarrow !3 ; B_1 \} \gg B_6 \) where

\( B_6 = (?x : T \rightarrow (?y : T \rightarrow !y ; \ B_5' < x < y >)) \)
COMMUNICATIONS

A communications protocol consists of a transmitting process P and a receiving process Q such that P >> 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 >> Q_x is a buffer

then so is (∀x ∈ T → (P_x >> (1; Q_x))) .... (1)

Proof. Let t be a trace of (1)

then first (ins(t)) = first (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 >> Q_x, which is a buffer.

∴ outs(t') ≤ ins(t') (3)

but ins(t) = < first (ins(t)) > ins(t') (4)

and outs(t) = < first (outs(t')) > outs(t) (5)

∴ outs(t) ≤ ins(t) from (2, 3, 4, 5)
If for all $x : T$

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

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

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

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

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 $(P_1 \gg P_2) \gg (Q_2 \gg Q_1)$ (composition of protocols).
Phase encoding.

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

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

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

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

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

Proof. \( P \gg Q = \)

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

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

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

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

\[ P_0 = \chi: \{0,1\} \rightarrow \!\chi; P_x \]
\[ P_1 = \chi: \{0,1\} \rightarrow (1-\chi); P_x \]

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

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

\[ Q_0 = \chi: \{0,1\} \rightarrow \!\chi; Q_0 \]
\[ Q_1 = \chi: \{0,1\} \rightarrow !(1-\chi); Q_1 \]

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

\( Q \) 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 $x = 0, 1$

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

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

$$P_1 \gg Q_1 = \exists x: \{0, 1\} \rightarrow (((1-x); P_x) \gg Q_1)$$

$$= \exists x: \{0, 1\} \rightarrow (P_x \gg ((1-x); Q_x))$$

$$= \exists x: \{0, 1\} \rightarrow (P_x \gg ((1-x); Q_x))$$

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 \( R = (x \rightarrow a \rightarrow P \sqcup y \rightarrow b \rightarrow Q) \setminus \{x, y\} \)

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, \( R = (a \rightarrow P \sqcup b \rightarrow Q) \), 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 \( P^\circ = \{a \mid \langle a \rangle \in \text{traces}(P)\} \)

\( P^\circ \) 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 \( \emptyset \)

\[
P \text{can refuse } X \Rightarrow P \text{ can refuse } X \cup (A - P^0).
\]

\( (A - P^0) \) is a set which \( P \) must refuse.

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

So \( P \text{ after } s \) = \( P \)

\[
P \text{ after } st = (P \text{ after } s) \text{ after } t
\]
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 \( \forall a (a \in P^O \Rightarrow P \text{after} a = Q \text{after} a) \)
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 \( \text{traces}(P) = \text{def} \{ s \mid (s, \{ \}) \in P \} \)
\( P \text{canrefuse} X = \text{def} \langle \rangle, X \rangle \in P \)
\( P \text{after} s = \text{def} \{ (t, X) \mid (st, X) \in P \} \)

\( \text{traces}(P) \) must be nonempty \& prefix-closed
\( \{ X \mid P \text{canrefuse} X \} \) must be nonempty \& left-closed
and closed under \( \text{unim} \) with \( \text{traces}(P) \).
EXAMPLES. with alphabet $A$.

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

$\text{STOP}_A \overset{\text{df}}{=} \{(\langle \ast \rangle, X) \mid X \in A\}$

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

$\text{RUN}_A \overset{\text{df}}{=} \{(s, \{\rangle \} \mid s \in A^*\}$

$\text{CHAOS}_A$ can do anything
can refuse anything.

$\text{CHAOS}_A \overset{\text{df}}{=} \{\langle \ast \rangle, X \rangle \mid A^* \land X \in A\}$

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

$$(x : B \rightarrow F(x)) \text{ first accepts any } x \text{ in } B, \text{ and then behaves like } F(x)$$

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

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

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

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

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

P and Q have same alphabet A.

P \parallel Q can accept anything acceptable to both P and Q
and if P can refuse X and Q can refuse Y, \( P \parallel Q \) can refuse \( X \cup Y \)

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

traces \( (P \parallel Q) \) = traces \( (P) \) \& traces \( (Q) \)

\( (P \parallel Q) \text{ after } s = (P \text{ after } s) \parallel (Q \text{ after } s) \) for \( s \in \text{traces}(P \parallel Q) \)

\( \parallel \) is associative & commutative,

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

\[
(x : B \rightarrow F(x)) \parallel (y : C \rightarrow G(y)) = (z : B \cap C \rightarrow (F(z) \parallel G(z)))
\]
P ∩ 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 ∩ Q = df P ∪ Q

∩ is associative, commutative, and idempotent
with zero CHAOS.

traces (P ∩ Q) = traces (P) ∪ traces (Q)

(P ∩ Q) after s = P after s if s ∈ traces (P) - traces (Q)
= Q after s if s ∈ traces (Q) - traces (P)
= (P after s) ∩ (Q after s)
if s ∈ traces (P) ∩ traces (Q)

If we admit the EMPTY relation as a process,
it would be the unit of ∩.
$P \sqcup Q$ behaves like $P$ or like $Q$; the choice can be influenced by its environment, but only on the first step. It can do anything $P$ or $Q$ can do. It can refuse anything that both $P$ and $Q$ can refuse.

$$P \sqcup Q = \{(\leftarrow, X) \mid (\leftarrow, X) \in P \cap Q\}$$

$$\cup \{(s, X) \mid s \neq \leftarrow \& (s, X) \in (P \cup Q)\}$$

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

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

$$(x : B \rightarrow F(b)) \sqcup (y : C \rightarrow G(c)) = (z : B \cup C \rightarrow$$

if $z \in B - C$ then $F(z)$ else if $z \in C - B$ then $G(z)$

else $F(z) \cap G(z))$$
LIMITS.

P ≼ 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.

PEQ if Q ⊆ P or PnQ = P

e.g. PnQ ∈ P ∩ Q

CHAOSP ∈ P

P n (P ∩ Q) ∈ P ∩ Q.

If Pi ∈ Pi+1 for all i, then we write

∩i Pi = ∩i Pi

The relation ≼ is a complete partial order with CHAOSA as its bottom.

If we add EMPTY, processes form a complete lattice with 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 $B$.

\[ \text{traces}(F^{-1}(P)) = \{ s \mid F(s) \in \text{traces}(P) \} \]

\[ (F^{-1}(P))^o = F^{-1}(P^o) \]

\[ (F^{-1}(P)) \text{ after } s = F^{-1}(P \text{ after } f(s)) \]

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

\[ F^{-1}(P \parallel Q) = \]

\[ F^{-1}(P \circ Q) = \]

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

\[ F(P) \]
ALPHABET EXTENSION

Let $P$ be a process with alphabet $A$, then $\text{Point } B$ 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.

$$\text{Point } B = \{(s, X) \mid s \in (A \cup B)^* \land (s, X) \in P\}$$

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

$$(\text{Point } B)^* = P^* \cup (B - A)$$

$$(\text{Point } B) \text{ can refuse } X \equiv P \text{ can refuse } X$$

$$(\text{Point } B) \text{ after } s = (\text{after } s_A) \text{ ext } B$$

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

If $Q$ has alphabet $B$ then

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

$||$ 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\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.

\((P\setminus B)^* \geq P^* - B\)

\(P\) cannot refuse \(X\) & \(X \cap B = \{\}\) \(\Rightarrow (P\setminus B)\) cannot refuse \(X\)

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

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

These properties are satisfied by

\[
|P\setminus B| = \{ (s_{A-B}, X) | X \cap B = \{\} \& (s, X \cup B) \in P \}\]

But \(\{(b^n, X) | X \subseteq \{a, b\}\}\) is empty, i.e. 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.

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

\[ P\setminus \emptyset = P \]

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

\[ \text{NOTE - we rely on finitude of alphabets} \]
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.

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

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.
A function $F$ from processes to processes is **continuous** if for all ascending chains $\{P_i | i \in \mathbb{N} \text{ & } \forall i \ P_i \in P_{i+1}\}$

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

If $F$ is continuous, the least solution of

$$p = F(p)$$

is given by $p = \bigsqcup_i F^{i}(\text{CHAOS}_P)$

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

All functions defined so far are continuous.
## Symposium on Communicating Sequential Processes

### Programme

**Saturday March 22, 1980.**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.30 - 9.30</td>
<td>Registration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.30 - 10.30</td>
<td>Hoare C.A.R.</td>
<td>CSP Lecture I</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>Dromey R.G.</td>
<td>Text Searching</td>
<td></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>Hoare C.A.R.</td>
<td></td>
<td>CSP Language and Implementation</td>
</tr>
<tr>
<td>15.00 - 15.30</td>
<td>Afternoon Tea</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.30 - 16.30</td>
<td>Stanton R.B.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.30 - 17.30</td>
<td>Happy Hour</td>
<td>Barter C.J.</td>
<td></td>
</tr>
<tr>
<td>17.30 - 18.00</td>
<td>Dinner</td>
<td>Reinfelds J.</td>
<td>Software Science</td>
</tr>
<tr>
<td>18.00 - 21.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Sunday March 23, 1980.**

<table>
<thead>
<tr>
<th>Time</th>
<th>Event</th>
<th>Speaker</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.30 - 10.30</td>
<td>Hoare C.A.R.</td>
<td>CSP Lecture IV</td>
<td></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>Correctness</td>
<td>Proof of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Indenting Pro-</td>
</tr>
<tr>
<td>12.00 - 13.00</td>
<td>Lunch</td>
<td></td>
<td>gram</td>
</tr>
<tr>
<td>13.00 - 14.00</td>
<td>Hoare C.A.R.</td>
<td></td>
<td>CSP Lecture VI</td>
</tr>
<tr>
<td>14.00 - 15.00</td>
<td>Afternoon Tea</td>
<td>Tobias J.M.</td>
<td>Single User</td>
</tr>
<tr>
<td>15.00 - 15.30</td>
<td>Afternoon Tea</td>
<td></td>
<td>Multi Processor</td>
</tr>
<tr>
<td>15.30 - 16.30</td>
<td></td>
<td></td>
<td>Around High</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Level Language</td>
</tr>
</tbody>
</table>

---
<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>BAILLY Thomas</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>BARTER Chris J</td>
<td>University of Adelaide</td>
</tr>
<tr>
<td>BEROLDI 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>CADI John</td>
<td>CSIRO</td>
</tr>
<tr>
<td>CANTER Don</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>CARRINGTON David</td>
<td>University of New South Wales</td>
</tr>
<tr>
<td>CHROMAT Nuri</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>CHROMAT Trevor</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>CLARK Neville</td>
<td>New South Wales Institute of Technology</td>
</tr>
<tr>
<td>COLVILLE J</td>
<td>Australian National University</td>
</tr>
<tr>
<td>CREASY P</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>CROIOT Christopher</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>CUOCCO Anthony</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>DIX Trevor</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>DRONEY 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>University of Wollongong</td>
</tr>
<tr>
<td>FONG Meng Wai</td>
<td>University of Sydney</td>
</tr>
<tr>
<td>GERRER Anthony J</td>
<td>Royal Military College Duntroon</td>
</tr>
<tr>
<td>GERRITY George W</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>HALIMAH Lillian</td>
<td>Australian Iron and Steel Pty Ltd</td>
</tr>
<tr>
<td>HAMMONDS Wal</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>HAYES Ian 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>HERREPOE 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 Les</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>HUMPHRIFS George</td>
<td>Royal Australian Naval College HMAS Creswell</td>
</tr>
<tr>
<td>JAFRAJ Joxan</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>JORDAN Julian</td>
<td>University of Wollongong</td>
</tr>
<tr>
<td>KANDIL Ashraf Fl-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>LTIU 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>MATTEI Prahbaker</td>
<td>University of Melbourne</td>
</tr>
<tr>
<td>MAYAPAS David</td>
<td>University of Wollongong</td>
</tr>
</tbody>
</table>