进程代数

Communicating Sequential Processes

C. A. R. Hoare

VMS—the simple vending machine

VMC—the complex vending machine

and the letters P, Q, R (occurring in laws) stand for arbitrary processes.

3. The letters x, y, z are variables denoting events.
4. The letters A, B, C stand for sets of events.
5. The letters X, Y are variables denoting processes.
6. The alphabet of process P is denoted αP, e.g.,

$$\alpha VMS = \{coin, choc\}$$
$$\alpha VMC = \{in1p, in2p, small, large, out1p\}$$

The process with alphabet A which never actually engages in any of the events of A is called $STOP_A$. This describes the behaviour of a broken object:

Let x be an event and let P be a process. Then

$(x \rightarrow P)$ (pronounced "x then P")

$\alpha(x \rightarrow P) = \alpha P$ provided $x \in \alpha P$

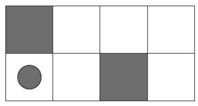
X1 A simple vending machine which consumes one coin before breaking

$(coin \rightarrow STOP_{\alpha VMS})$

X2 A simple vending machine that successfully serves two customers before breaking

$(coin \rightarrow (choc \rightarrow (coin \rightarrow (choc \rightarrow STOP_{\alpha VMS}))))$

X3 A counter starts on the bottom left square of a board, and can move only up or right to an adjacent white square



$\alpha CTR = \{up, right\}$

$CTR = (right \rightarrow up \rightarrow right \rightarrow right \rightarrow STOP_{\alpha CTR})$

$\alpha CLOCK = \{tick\}$

$CLOCK = (tick \rightarrow CLOCK)$

$CLOCK$

$= (tick \rightarrow CLOCK)$ [original equation]

$= (tick \rightarrow (tick \rightarrow CLOCK))$ [by substitution]

$= (tick \rightarrow (tick \rightarrow (tick \rightarrow CLOCK)))$ [similarly]

$tick \rightarrow tick \rightarrow tick \rightarrow \cdots$

X1 A perpetual clock

$CLOCK = \mu X : \{tick\} \bullet (tick \rightarrow X)$

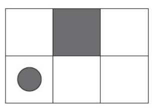
Choice

$(x \rightarrow P \mid y \rightarrow Q)$

$\alpha(x \rightarrow P \mid y \rightarrow Q) = \alpha P$ provided $\{x, y\} \subseteq \alpha P$ and $\alpha P = \alpha Q$

The bar $\mid$ should be pronounced "choice": "x then P choice y then Q"

X1 The possible movements of a counter on the board



are defined by the process

$(up \rightarrow STOP \mid right \rightarrow right \rightarrow up \rightarrow STOP)$

X2 A machine which offers a choice of two combinations of change for 5p (compare 1.1.2 X3 and X4, which offer no choice).

$$CH5C = in5p \rightarrow (out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5C$$
$$\mid out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5C)$$

The choice is exercised by the customer of the machine. □

X3 A machine that serves either chocolate or toffee on each transaction

$$VMCT = \mu X \bullet coin \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$$

□

X4 A more complicated vending machine, which offers a choice of coins and a choice of goods and change

$$\begin{aligned} VMC = (&in2p \rightarrow (large \rightarrow VMC \\ &\quad \mid small \rightarrow out1p \rightarrow VMC) \\ &\mid in1p \rightarrow (small \rightarrow VMC \\ &\quad \mid in1p \rightarrow (large \rightarrow VMC \\ &\qquad \mid in1p \rightarrow STOP))) \end{aligned}$$

X7 A copying process engages in the following events

in.0—input of zero on its input channel
in.1—input of one on its input channel
out.0—output of zero on its output channel
out.1—output of one on its output channel

Its behaviour consists of a repetition of pairs of events. On each cycle, it inputs a bit and outputs the same bit

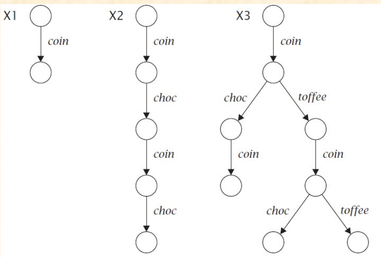
$$COPYBIT = \mu X \bullet (in.0 \rightarrow out.0 \rightarrow X$$
$$\mid in.1 \rightarrow out.1 \rightarrow X)$$

Pictures

$(coin \rightarrow STOP_{\alpha VMS})$

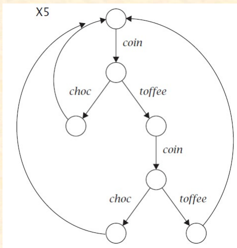
$(coin \rightarrow (choc \rightarrow (coin \rightarrow (choc \rightarrow STOP_{\alpha VMS}))))$

$VMCT = \mu X \bullet coin \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)$



X5 A machine that allows its customer to sample a chocolate, and trusts him to pay after. The normal sequence of events is also allowed

$$VMCRED = \mu X \bullet (coin \rightarrow choc \rightarrow X$$
$$\mid choc \rightarrow coin \rightarrow X)$$



Laws

L1 $(x : A \rightarrow P(x)) = (y : B \rightarrow Q(y)) \equiv (A = B \wedge \forall x : A \bullet P(x) = Q(x))$

$(x \rightarrow P \mid y \rightarrow Q) = (y \rightarrow Q \mid x \rightarrow P)$

$(x \rightarrow P) \neq STOP$

$(c \rightarrow P) \neq (d \rightarrow Q) \quad$ if $c \neq d$

$(c \rightarrow P) = (c \rightarrow Q) \equiv P = Q$

$(coin \rightarrow choc \rightarrow coin \rightarrow choc \rightarrow STOP) \neq (coin \rightarrow STOP)$

$\mu X \bullet F(X) = F(\mu X \bullet F(X))$

Traces

$\langle x, y \rangle$ consists of two events, x followed by y.
$\langle x \rangle$ is a sequence containing only the event x.
$\langle \rangle$ is the empty sequence containing no events.

$\langle coin, choc, coin, choc \rangle$

$s \frown t$

$\langle coin, choc \rangle \frown \langle coin, toffee \rangle = \langle coin, choc, coin, toffee \rangle$
$\langle in1p \rangle \frown \langle in1p \rangle = \langle in1p, in1p \rangle$
$\langle in1p, in1p \rangle \frown \langle \rangle = \langle in1p, in1p \rangle$

L1 $s \frown \langle \rangle = \langle \rangle \frown s = s$

L2 $s \frown (t \frown y) = (s \frown t) \frown u$

X1 The only trace of the behaviour of the process *STOP* is $\langle\rangle$. The notebook of the observer of this process remains forever blank

$$traces(STOP) = \{\langle\rangle\}$$

□

X2 There are only two traces of the machine that ingests a coin before breaking

$$traces(coin \rightarrow STOP) = \{\langle\rangle, \langle coin\rangle\}$$

□

X3 A clock that does nothing but *tick*

$$\begin{aligned} traces(\mu X \bullet tick \rightarrow X) &= \{\langle\rangle, \langle tick\rangle, \langle tick, tick\rangle, \ldots\} \\ &= \{tick\}^* \end{aligned}$$

As with most interesting processes, the set of traces is infinite, although of course each individual trace is finite. □

X4 A simple vending machine

$$traces(\mu X \bullet coin \rightarrow choc \rightarrow X) = \{ s \mid \exists n \bullet s \leq \langle coin, choc\rangle^n \}$$

□

L1 $traces(STOP) = \{ t \mid t = \langle\rangle \} = \{\langle\rangle\}$

L2 $traces(c \rightarrow P) = \{ t \mid t = \langle\rangle \vee (t_0 = c \wedge t' \in traces(P)) \}$
$\qquad = \{\langle\rangle\} \cup \{ \langle c\rangle ^\frown t \mid t \in traces(P) \}$

L3 $traces(c \rightarrow P \mid d \rightarrow Q) =$
$\qquad \{ t \mid t = \langle\rangle \vee (t_0 = c \wedge t' \in traces(P)) \vee (t_0 = d \wedge t' \in traces(Q)) \}$

If P and Q are processes with the same alphabet, we introduce the notation

$$P \parallel Q$$

to denote the process which behaves like the system composed of processes P and Q interacting in lock-step synchronisation as described above.

Examples

X1 A greedy customer of a vending machine is perfectly happy to obtain a toffee or even a chocolate without paying. However, if thwarted in these desires, he is reluctantly prepared to pay a coin, but then he insists on taking a chocolate

$$\begin{aligned} GRCUST = (&toffee \rightarrow GRCUST \\ &\mid choc \rightarrow GRCUST \\ &\mid coin \rightarrow choc \rightarrow GRCUST) \end{aligned}$$

When this customer is brought together with the machine *VMCT* (1.1.3 X3) his greed is frustrated, since the vending machine does not allow goods to be extracted before payment. Similarly, *VMCT* never gives a toffee, because the customer never wants one after he has paid

$$(GRCUST \parallel VMCT) = \mu X \bullet (coin \rightarrow choc \rightarrow X)$$

X2 A foolish customer wants a large biscuit, so he puts his coin in the vending machine *VMC*. He does not notice whether he has inserted a large coin or a small one; nevertheless, he is determined on a large biscuit

$$\begin{aligned} FOOLCUST = (&in2p \rightarrow large \rightarrow FOOLCUST \\ &\mid in1p \rightarrow large \rightarrow FOOLCUST) \end{aligned}$$

Unfortunately, the vending machine is not prepared to yield a large biscuit for only a small coin

$$(FOOLCUST \parallel VMC) = \mu X \bullet (in2p \rightarrow large \rightarrow X \mid in1p \rightarrow STOP)$$

L1 $P \parallel Q = Q \parallel P$

The next law shows that when three processes are assembled, it does not matter in which order they are put together

L2 $P \parallel (Q \parallel R) = (P \parallel Q) \parallel R$

Thirdly, a deadlocked process infects the whole system with deadlock; but composition with $RUN_{\alpha P}$ (1.1.3 X8) makes no difference

L3A $P \parallel STOP_{\alpha P} = STOP_{\alpha P}$

L3B $P \parallel RUN_{\alpha P} = P$

The next laws show how a pair of processes either engage simultaneously in the same action, or deadlock if they disagree on what the first action should be

L4A $(c \rightarrow P) \parallel (c \rightarrow Q) = (c \rightarrow (P \parallel Q))$

L4B $(c \rightarrow P) \parallel (d \rightarrow Q) = STOP \qquad \text{if } c \neq d$

L4 $(x : A \rightarrow P(x)) \parallel (y : B \rightarrow Q(y)) = (z : (A \cap B) \rightarrow (P(z) \parallel Q(z)))$

Example

X1 Let $P = (a \rightarrow b \rightarrow P \mid b \rightarrow P)$

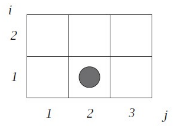
and $Q = (a \rightarrow (b \rightarrow Q \mid c \rightarrow Q))$

Then

$$\begin{aligned} &(P \parallel Q) = \\ &\quad = a \rightarrow ((b \rightarrow P) \parallel (b \rightarrow Q \mid c \rightarrow Q)) && \text{[by L4A]} \\ &\quad = a \rightarrow (b \rightarrow (P \parallel Q)) && \text{[by L4A]} \\ &\quad = \mu X \bullet (a \rightarrow b \rightarrow X) && \text{[since the recursion is guarded.]} \end{aligned}$$

□

X2



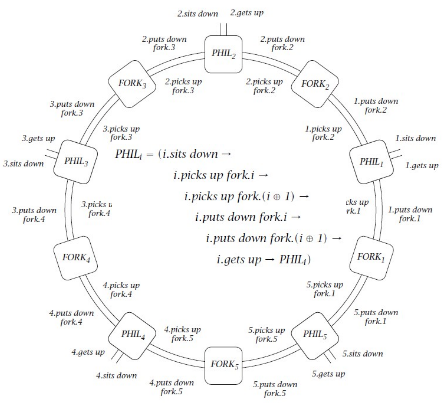
A counter starts at the middle bottom square of the board, and may move within the board either *up*, *down*, *left* or *right*. Let

$$\alpha P = \{up, down\}$$
$$P = (up \rightarrow down \rightarrow P)$$
$$\alpha Q = \{left, right\}$$
$$Q = (right \rightarrow left \rightarrow Q \mid left \rightarrow right \rightarrow Q)$$

哲学家进餐问题

$$\alpha PHIL_i = \{i.sits\ down, i.gets\ up,\ i.picks\ up\ fork.i, i.picks\ up\ fork.(i \oplus 1),\ i.puts\ down\ fork.i, i.puts\ down\ fork.(i \oplus 1)\}$$

$$\alpha FORK_i = \{i.picks\ up\ fork.i, (i \ominus 1).picks\ up\ fork.i,\ i.puts\ down\ fork.i, (i \ominus 1).puts\ down\ fork.i\}$$



$$FORK_i = (i.picks\ up\ fork.i \rightarrow i.puts\ down\ fork.i \rightarrow FORK_i \mid (i \ominus 1).picks\ up\ fork.i \rightarrow (i \ominus 1).puts\ down\ fork.i \rightarrow FORK_i)$$

$$PHILOS = (PHIL_0 \parallel PHIL_1 \parallel PHIL_2 \parallel PHIL_3 \parallel PHIL_4)$$
$$FORKS = (FORK_0 \parallel FORK_1 \parallel FORK_2 \parallel FORK_3 \parallel FORK_4)$$
$$COLLEGE = PHILOS \parallel FORKS$$

expensive. The solution finally adopted was the appointment of a footman, whose duty it was to assist each philosopher into and out of his chair. His alphabet was defined as

$$\bigcup_{i=0}^{4} \{i.sits\ down, i.gets\ up\}$$

This footman was given secret instructions never to allow more than four philosophers to be seated simultaneously. His behaviour is most simply defined by mutual recursion. Let

$$U = \bigcup_{i=0}^{4} \{i.gets\ up\} \qquad D = \bigcup_{i=0}^{4} \{i.sits\ down\}$$

$FOOT_j$ defines the behaviour of the footman with j philosophers seated

$$FOOT_0 = (x : D \rightarrow FOOT_1)$$
$$FOOT_j = (x : D \rightarrow FOOT_{j+1} \mid y : U \rightarrow FOOT_{j-1}) \qquad \text{for } j \in \{1, 2, 3\}$$
$$FOOT_4 = (y : U \rightarrow FOOT_3)$$

A college free of deadlock is defined

$$NEWCOLLEGE = (COLLEGE \parallel FOOT_0)$$

3.2 Nondeterministic or

If P and Q are processes, then we introduce the notation

$$P \sqcap Q \qquad (P \text{ or } Q)$$

to denote a process which behaves either like P or like Q, where the selection between them is made arbitrarily, without the knowledge of control of the external environment. The alphabets of the operands are assumed to be the same

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

Examples

X1 A change-giving machine which always gives the right change in one of two combinations

$$CH5D = (in5p \rightarrow ((out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5D) \sqcap (out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5D)))$$

□

X2 $CH5D$ may give a different combination of change on each occasion of use. Here is a machine that always gives the same combination, but we do not know initially which it will be (see 1.1.2 X3, X4)

$$CH5E = CH5A \sqcap CH5B$$

3.2.1 Laws

The algebraic laws governing nondeterministic choice are exceptionally simple and obvious. A choice between P and P is vacuous

L1 $P \sqcap P = P$ (idempotence)

It does not matter in which order the choice is presented

L2 $P \sqcap Q = Q \sqcap P$ (symmetry)

A choice between three alternatives can be split into two successive binary choices. It does not matter in which way this is done

L3 $P \sqcap (Q \sqcap R) = (P \sqcap Q) \sqcap R$ (associativity)

The occasion on which a nondeterministic choice is made is not significant. A process which first does x and then makes a choice is indistinguishable from one which first makes the choice and then does x

L4 $x \to (P \sqcap Q) = (x \to P) \sqcap (x \to Q)$ (distribution)

L5 $(x : B \to (P(x) \sqcap Q(x))) = (x : B \to P(x)) \sqcap (x : B \to Q(x))$

L6 $P \parallel (Q \sqcap R) = (P \parallel Q) \sqcap (P \parallel R)$

L7 $(P \sqcap Q) \parallel R = (P \parallel R) \sqcap (Q \parallel R)$

L8 $f(P \sqcap Q) = f(P) \sqcap f(Q)$

However, the recursion operator is *not* distributive, except in the trivial case where the operands of $\sqcap$ are identical. This point is simply illustrated by the difference between the two processes

$$P = \mu X \bullet ((a \to X) \sqcap (b \to X))$$

$$Q = (\mu X \bullet (a \to X)) \sqcap (\mu X \bullet (b \to X))$$

4.2 Input and output

Let v be any member of $\alpha c(P)$. A process which first outputs v on the channel c and then behaves like P is defined

$$(c!v \to P) = (c.v \to P)$$

The only event in which this process is initially prepared to engage is the communication event $c.v$.

A process which is initially prepared to input any value x communicable on the channel c, and then behave like $P(x)$, is defined

$$(c?x \to P(x)) = (y : \{ y \mid channel(y) = c \} \to P(message(y)))$$

Example

X1 Using the new definitions of input and output we can rewrite 1.1.3 X7

$$COPYBIT = \mu X \bullet (in?x \to (out!x \to X))$$

where $\alpha in(COPYBIT) = \alpha out(COPYBIT) = \{0, 1\}$ □

We shall observe the convention that channels are used for communication in only one direction and between only two processes. A channel which is used only for output by a process will be called an output channel of that process; and one used only for input will be called an input channel. In both cases, we shall say loosely that the channel name is a member of the alphabet of the process.

When drawing a connection diagram (Section 2.4) of a process, the channels are drawn as arrows in the appropriate direction, and labelled with the name of the channel (Figure 4.1).

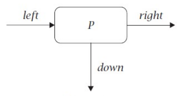


Figure 4.1

L2 $((c!v \to P) \parallel (c?x \to Q(x))) \setminus C = (P \parallel Q(v)) \setminus C$

where $C = \{ c.v \mid v \in \alpha c \}$

L1 $(c!v \rightarrow P) \parallel (c?x \rightarrow Q(x)) = c!v \rightarrow (P \parallel Q(v))$