Theory of concurrency

Lecture 4

Communicating sequential processes

Processes

Processes: **Introduction**

- <u>Specification of behaviour</u>
  - decide what kinds of event or action will be of interest;
    - choose a different name for each kind.
- *Simple vending machine*
  - *Coin* — the insertion of a coin in the slot of a vending machine; 
  - *Choc* — the extraction of a chocolate from the dispenser of the machine. 
- *Complex vending machine:*
  - *in1p* — the insertion of one penny; 
  - *in2p* — the insertion of a two-penny coin; 
  - *small* — the extraction of a small biscuit or cookie; 
  - *large* — the extraction of a large biscuit or cookie; 
  - *out1p* — the extraction of one penny in change. 

Processes: **Introduction**

- Event name – an event *class*;
  - many occurrences of events in a single class, separated in time.

- *Alphabet* of a system
  - the set of names of *events relevant* for a particular description.
  - Every event *can* happen
  - *No other* events cannot happen
  - Ignore irrelevant properties and events

Processes: **Introduction**

- The occurrence of each event
  - an *atomic* action without duration.

- Extended or time-consuming actions
  - a *pair* of events: start and finish.
  - The duration
    - the internal between the occurrence of the start event and the occurrence of the finish event
      - during such an internal, other events may occur.
  - may *overlap* in time if the start of each one precedes the finish of the other.

Processes: **Introduction**

- No the exact timing of occurrences of events.
  - designs and reasoning about them are simplified
    - can be applied to physical and computing systems of any speed and performance.
  - Cannot reason whether one event occurs simultaneously with another.
    - Synchronisation (simultaneity of a pair of events)
      - as a *single* event occurrence.

- *Critical timing of responses* can be considered
  - independently of the logical correctness of the design.

- No distinction between inside events and outside events.

Processes: **Introduction**

- *Process* is
  - the behaviour pattern of a system described with the limited set of events.
- Specific defined events
  - *coin, choc, in2p, out1p*, a, b, c, d, e.
  - A, B, C for sets of events.
- Specific defined processes
  - *VMS* — the simple vending machine; *VMC* — the complex vending machine
  - P, Q, R for arbitrary processes.
- Variables
  - x, y, z for events.
  - X, Y for processes.
- The *alphabet* of process P is αP
  - $\alpha VMS = \{coin, choc\}$
  - $\alpha VMC = \{in1p, in2p, small, large, out1p\}$

Processes: **Introduction**

- $STOP_A$
  - the process with alphabet A which never deals with events of A.
  - the behaviour of a broken system.

- Systems with different alphabets are distinguished.
  - $STOP_{aVMS}$ might have given out a chocolate
  - $STOP_{aVMC}$ could never give out a chocolate, only a biscuit.

- A customer for either machine knows these facts, even if he does not know that both machines are broken.

$aVMS = \{coin, choc\}$
$aVMC = \{in1p, in2p, small, large, out1p\}$

Processes: Introduction: **Prefix**

x – an event, P – a process.

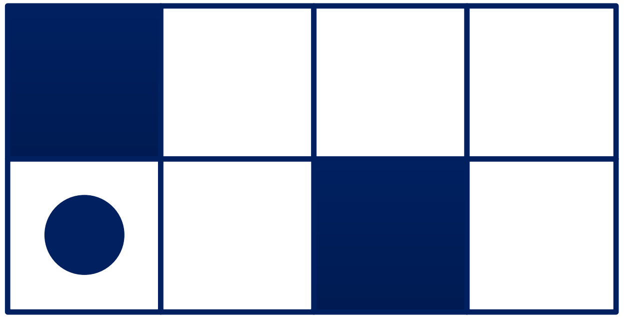
- $(x \to P)$
  - "x then P"
  - a system first engages in the event x and then behaves exactly as P.
  - $\alpha(x \to P) = \alpha P$ provided $x \in \alpha P$

X1. $(coin \to STOP_{\alpha VMS})$

X2. $(coin \to (choc \to (coin \to (choc \to STOP_{\alpha VMS}))))$

X3. $CTR = (right \to up \to right \to right \to STOP_{\alpha CTR})$

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



Processes: Introduction: **Prefix**

- The $\rightarrow$ operator
  - a process on the right and a *single event* on the left.
  - **syntactically *incorrect***
    - for processes $P \rightarrow Q$
    - for events $x \rightarrow y$
      - correctly: $x \rightarrow (y \rightarrow STOP)$

Processes: Introduction: **Recursion**

- Indefinitely long behaviour
  - repetitive behaviours
  - no a prior decision on the length of life of a system;
    - for systems which act and interact with their environment for as long as they are needed.

- A ticking clock
  - $\alpha CLOCK = \{tick\}$
  - $(tick \rightarrow CLOCK)$
    - behaves exactly as the original clock.
  - $\boldsymbol{CLOCK = (tick \rightarrow CLOCK)}$

Processes: Introduction: **Recursion**

- Consequences:
  - $CLOCK$

$$
\begin{aligned}
&= (tick \rightarrow CLOCK) && \text{[original equation]} \\
&= (tick \rightarrow (tick \rightarrow CLOCK)) && \text{[by substitution]} \\
&= (tick \rightarrow (tick \rightarrow (tick \rightarrow CLOCK))) && \text{[similarly]}
\end{aligned}
$$

- The potentially unbounded behaviour of the $CLOCK$
  - $tick \rightarrow tick \rightarrow tick \rightarrow \cdots$

Processes: Introduction: **Recursion**

- A meaningful recursive definition of processes
  - if the right-hand side of the equation starts with at least one event prefixed to all recursive occurrences of the process name.
  - Defines nothing
    - $X = X$
      - everything is a solution.

- A *guarded* process description begins with a prefix.

Processes: Introduction: **Recursion**

- The equation $X = F(X)$
  - has a unique solution in alphabet A iff
    - $F(X)$ as a guarded expression containing the process name X
    - A *as* the alphabet of X.

- This solution is $\mu X : A \bullet F(X)$
  - X is a bound variable
    - can be changed at will, since
      - $\mu X : A \bullet F(X) = \mu Y : A \bullet F(Y)$
        - A solution for the equation $X = F(X)$ is a solution for $Y = F(Y)$.

- The method of substitution informally justify
  - existence and uniqueness of the solution of a guarded equation.

Processes: Introduction: **Recursion**

X1. A perpetual clock

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

X2. A simple vending machine which serves as many chocs as required

- $VMS = (coin \rightarrow (choc \rightarrow VMS))$
  - $VMS = \mu X : \{coin, choc\} \bullet (coin \rightarrow (choc \rightarrow X))$

X3. A machine that gives change for 5p repeatedly

- $CH5A = (in5p \rightarrow out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5A)$
  - $\alpha CH5A = \{in5p, out2p, out1p\}$

X4. A different change-giving machine with the same alphabet

- $CH5B = (in5p \rightarrow out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5B)$

Processes: Introduction: **Choice**

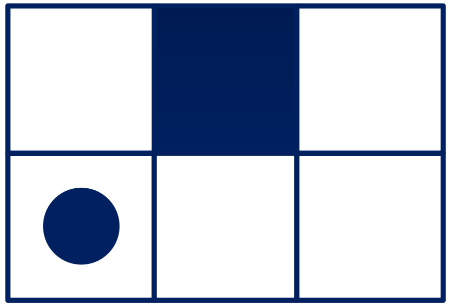
- Behaviour influenced by interaction with the environment.
  - VM may offer a choice of slots for inserting a 2p or a 1p coin.

- $(x \rightarrow P \mid y \rightarrow Q)$
  - "x then P choice y then Q"
  - x and y are *distinct* events
  - the subsequent behaviour is
    - P if the first event was x
    - Q if the first event was y.

- Constancy of alphabets:

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

Processes: Introduction: Choice



X1. The possible movements of a counter on the board

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

X2. A machine which offers a choice of two combinations of change for 5p

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

$$\begin{aligned} CH5C = in5p \rightarrow (&out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5C \\ &\mid out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5C) \end{aligned}$$

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

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

$CH5A = (in5p \rightarrow out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5A)$
$CH5B = (in5p \rightarrow out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5B)$

Processes: Introduction: **Choice**

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

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

X5. A machine that allows its customer to sample a chocolate, and trusts him to pay after.

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

X6. To prevent loss, an initial payment is extracted for using *VMCRED*

$$VMS2 = (coin \rightarrow VMCRED)$$

Processes: Introduction: **Choice**

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

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

Processes: Introduction: **Choice**

- <u>The extended definition of choice</u>
  - $(x \rightarrow P \mid y \rightarrow Q \mid \ldots \mid z \rightarrow R)$
- Choice is *not* an operator on processes.
  - **Incorrect:** $\boldsymbol{P \mid Q}$ for processes P and Q.
  - The reason is to avoid giving a meaning to
    - $(x \rightarrow P \mid x \rightarrow Q)$: not a choice of first event.
- A single operator with three arguments P, Q, R, if x, y, z are distinct events:
  - $(x \rightarrow P \mid y \rightarrow Q \mid z \rightarrow R)$
  - $\neq (x \rightarrow P \mid (y \rightarrow Q \mid z \rightarrow R))$
    - syntactically incorrect

Processes: Introduction: **Choice**

- The general definition of choice
  - $(x : B \rightarrow P(x))$
    - B is any set of events,
      - the initial *menu* of the process
    - $P(x)$ is an expression defining a process for each different x in B.
  - a choice of any event y in B, and then behaves like $P(y)$.
  - "x from B then P of x".
  - x is a local variable
    - $(x : B \rightarrow P(x)) = (y : B \rightarrow P(y))$

X8. A process which at all times can engage in any event of its alphabet A

- $\alpha RUN_A = A$

$$RUN_A = (x : A \rightarrow RUN_A)$$

Processes: Introduction: **Choice**

- The menu is $\{e\}$:
  - $(x : \{e\} \rightarrow P(x)) = (e \rightarrow P(e))$
- The menu is $\{\ \}$:
  - $(x : \{\} \rightarrow P(x)) = (y : \{\} \rightarrow Q(y)) = STOP$
- The binary choice in the general notation
  - $(a \rightarrow P \mid b \rightarrow Q) = (x : B \rightarrow R(x))$
    - $B = \{a, b\}$
    - $R(x) = \textbf{if } x = a \textbf{ then } P \textbf{ else } Q$
- Choice between three or more alternatives can be similarly expressed.
- Special cases of the general choice notation:
  - choice, prefixing, and *STOP*.

Processes: Introduction: **Mutual recursion**

- Recursion
  - a single process as the solution of a single equation.

- *Mutual recursion* is generalisation:
  - the solution of sets of simultaneous equations in more than one unknown.
  - Correctness
    - all the right-hand sides must be guarded,
    - each unknown process must appear exactly once on the left-hand side of one of the equations.

Processes: Introduction: **Mutual recursion**

X1. A drinks dispenser has two buttons labelled ORANGE and LEMON.

- The actions of pressing the two buttons:
  - *setorange* and *setlemon*.
- The actions of dispensing a drink:
  - *orange* and *lemon*.
- The choice of drink that will next be dispensed is made by pressing the corresponding button.
- Before any button is pressed, no drink will be dispensed.
- $\alpha DD = \alpha O = \alpha L = \{setorange, setlemon, orange, lemon\}$

$$
\begin{aligned}
DD &= (setorange \rightarrow O \mid setlemon \rightarrow L) \\
O &= (orange \rightarrow O \mid setlemon \rightarrow L \mid setorange \rightarrow O) \\
L &= (lemon \rightarrow L \mid setorange \rightarrow O \mid setlemon \rightarrow L)
\end{aligned}
$$

Processes: Introduction: **Mutual recursion**

- With indexed variables, it is possible to specify infinite sets of equations.

X2. An object starts on the ground.

- On the ground, it may move *around* and *up*.
- Not on the ground, it may move *up* and *down*.
- Let $n \in \{0, 1, 2, \ldots\}$.

$$CT_0 = (up \rightarrow CT_1 \mid around \rightarrow CT_0)$$
$$CT_{n+1} = (up \rightarrow CT_{n+2} \mid down \rightarrow CT_n)$$

- An ordinary inductive definition
  - validity: the right-hand side of each equation uses only indices less than that of the left-hand side.
- An infinite set of mutually recursive definitions
  - CT_{n+1} is defined in terms of CT_{n+2},
  - validity : the right-hand side of each equation is guarded.

Processes: Introduction: **Laws**

- There are many different ways of describing the same behaviour.
  - $(x \rightarrow P \mid y \rightarrow Q) = (y \rightarrow Q \mid x \rightarrow P)$
    - Presentation of an order of a choice does not matter.
  - $(x \rightarrow P) \neq STOP$
    - A process that can do something is not the same as one that cannot do anything
- We must learn to recognise which expressions describe
  - *the same object and which do not.*
- *Identity* of processes with the same alphabet may be proved or disproved
  - by appeal to *algebraic laws*.

Processes: Introduction: **Laws**

- Here and elsewhere, we assume that the alphabets of the processes on each side of an equation are the same.

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

- Consequences:

L1A. $STOP \neq (d \rightarrow P)$

$$
\begin{aligned}
Proof: LHS &= (x : \{\} \rightarrow P) && \text{by def. GenCh} \\
&\neq (x : \{d\} \rightarrow P) && \text{because } \{\} \neq \{d\} \\
&= RHS && \text{by def. GenCh}
\end{aligned}
$$

L1B. $(c \rightarrow P) \neq (d \rightarrow Q)$ **if** $c \neq d$

$Proof: \{c\} \neq \{d\}$

Processes: Introduction: **Laws**

L1C. $(c \rightarrow P \mid d \rightarrow Q) = (d \rightarrow Q \mid c \rightarrow P)$

$$
\begin{array}{lll}
\textit{Proof}: \text{Define } R(x) = P & & \text{if } x = c \\
\qquad\qquad\qquad\qquad\quad = Q & & \text{if } x = d \\
LHS = (x : \{c, d\} \rightarrow R(x)) & & \text{by definition} \\
\qquad\; = (x : \{d, c\} \rightarrow R(x)) & & \quad \text{because } \{c, d\} = \{d, c\} \\
\qquad\; = RHS & & \quad \text{by definition}
\end{array}
$$

L1D. $(c \rightarrow P) = (c \rightarrow Q) \equiv P = Q$

$\textit{Proof}: \{c\} = \{c\}$

Processes: Introduction: **Laws**

- These laws permit proof of simple theorems.

X1. $(coin \rightarrow choc \rightarrow coin \rightarrow choc \rightarrow STOP) \neq (coin \rightarrow STOP)$

  - *Proof* : by L1D then L1A.

X2. $\mu X \bullet (coin \rightarrow (choc \rightarrow X \mid toffee \rightarrow X)) = \mu X \bullet (coin \rightarrow (toffee \rightarrow X \mid choc \rightarrow X))$

  - *Proof* : by L1C.

L1A. $STOP \neq (d \rightarrow P)$
L1C. $(c \rightarrow P \mid d \rightarrow Q) = (d \rightarrow Q \mid c \rightarrow P)$
L1D. $(c \rightarrow P) = (c \rightarrow Q) \equiv P = Q$

Processes: Introduction: **Laws**

- Every properly guarded recursive equation has only one solution.

L2. $(Y = F(Y)) \equiv (Y = \mu X \bullet F(X))$

  - If $F(X)$ is a guarded expression.

- The corollary states that $\mu X \bullet F(X)$ is indeed a solution.

L2A. $\mu X \bullet F(X) = F(\mu X \bullet F(X))$

$VMS = (coin \rightarrow (choc \rightarrow VMS))$

Processes: Introduction: **Laws**

X3. Let $VM1 = (coin \rightarrow VM2)$ and $VM2 = (choc \rightarrow VM1)$

- Required to prove $VM1 = VMS$.
- *Proof* : $VM1 = (coin \rightarrow VM2)$ definition of $VM1$

  $= (coin \rightarrow (choc \rightarrow VM1))$ definition of $VM2$
- Therefore $VM1$ is a solution of the same recursive equation as VMS.
- Since the equation is guarded, there is only one solution.
- So $VM1$ and VMS are just different names for this unique solution.

Processes: Introduction: **Laws**

- The law L2 can be extended to mutual recursion.
- The general form for a set of mutually recursive equations:
  - $X_i = F(i, \boldsymbol{X})$ for all i in S
    - S is an indexing set with one member for each equation,
    - $\boldsymbol{X}$ is an *array* of processes with indices ranging over the set S,
    - $F(i, \boldsymbol{X})$ is a guarded expression.
- There is only one array $\boldsymbol{X}$ whose elements satisfy all the equations:

L3. if$(\forall i : S \bullet (X_i = F(i, \boldsymbol{X}) \land Y_i = F(i, \boldsymbol{Y})))$ **then** $\boldsymbol{X} = \boldsymbol{Y}$.

L2. $(Y = F(Y)) \equiv (Y = \mu X \bullet F(X))$