Theory of concurrency

Lecture 8

Concurrency

Concurrency: Specifications

- Let P and Q be processes intended to run concurrently, and
  - P sat $S(tr)$ and Q sat $T(tr)$.
- Let tr be an arbitrary trace of $(P \parallel Q)$.
  - $(tr \upharpoonright \alpha P)$ is a trace of P by L1
    - it satisfies S: $S(tr \upharpoonright \alpha P)$
  - $(tr \upharpoonright \alpha Q)$ is a trace of Q by L1
    - it satisfies T: $T(tr \upharpoonright \alpha P)$
- This argument holds for every trace of $(P \parallel Q)$, hence
  - $(P \parallel Q)$ sat $(S(tr \upharpoonright \alpha P) \wedge T(tr \upharpoonright \alpha Q))$
- This reasoning is summarised in the law

L1. If P sat $S(tr)$ **and** Q sat $T(tr)$ **then** $(P \parallel Q)$ sat $(S(tr \upharpoonright \alpha P) \wedge T(tr \upharpoonright \alpha Q))$

Concurrency: Specifications

X1. Let $\alpha P = \{a, c\}$ and $\alpha Q = \{b, c\}$

- $P = (a \rightarrow c \rightarrow P)$ and $Q = (c \rightarrow b \rightarrow Q)$

- We wish to prove that
  - $(P \parallel Q)$ sat $0 \leq tr{\downarrow}a - tr{\downarrow}b \leq 2$

- The proof of *VMS* sat *VMSPEC* (the next slide) can be adapted to show that
  - P sat $(0 \leq tr{\downarrow}a - tr{\downarrow}c \leq 1)$ and Q sat $(0 \leq tr{\downarrow}c - tr{\downarrow}b \leq 1)$
  - *Why?*

- By L1 it follows that

$(P \parallel Q)$ sat

$$(0 \leq (tr \restriction \alpha P) \downarrow a - (tr \restriction \alpha P) \downarrow c \leq 1 \wedge$$
$$0 \leq (tr \restriction \alpha Q) \downarrow c - (tr \restriction \alpha Q) \downarrow b \leq 1)$$
$$\Rightarrow 0 \leq tr{\downarrow}a - tr{\downarrow}b \leq 2$$

[since $(tr \restriction A) \downarrow a = tr{\downarrow}a$ if $a \in A$.]

L1. If P sat $S(tr)$ **and** Q sat $T(tr)$ **then** $(P \parallel Q)$ sat $(S(tr \restriction \alpha P) \wedge T(tr \restriction \alpha Q))$

Concurrency: **Specifications**

X1. Prove that VMS sat $VMSPEC$

1. $STOP$ sat $tr = \langle\rangle$ **[L4A]**
   $\Rightarrow 0 \le (tr \downarrow coin = tr \downarrow choc) \le 1$ **[since ($\langle\rangle \downarrow coin$) = ($\langle\rangle \downarrow choc$) = 0]**

- The conclusion follows by an (implicit) appeal to L3.

2. Assume X sat $(0 \le ((tr \downarrow coin) - (tr \downarrow choc)) \le 1)$, then

   $(coin \rightarrow choc \rightarrow X)$ sat **[L4C]**
   $(tr \le \langle coin, choc\rangle) \vee$
   $(tr \ge \langle coin, choc\rangle \wedge 0 \le ((tr \downarrow coin) - (tr \downarrow choc)) \le 1))$
   $\Rightarrow 0 \le ((tr \downarrow coin) - (tr \downarrow choc)) \le 1$
   since
   $\langle\rangle \downarrow coin = \langle\rangle \downarrow choc = \langle coin\rangle \downarrow choc = 0$ **and**
   $\langle coin\rangle \downarrow coin = (\langle coin, choc\rangle \downarrow coin) = \langle coin, choc\rangle \downarrow choc = 1$ **and**
   $tr \ge \langle coin, choc\rangle \Rightarrow (tr \downarrow coin = tr'' \downarrow coin + 1 \wedge tr \downarrow choc = tr'' \downarrow choc + 1)$

- The conclusion follows by appeal to L3 and L6.

L3. Weak spec
L6. Recursion

Concurrency: **Specifications**

To prove absence of deadlock.

- Impossible to use the laws for *sat* because they allow *STOP* to satisfy every satisfiable specification.
- Try to use careful proof, as for the Dining Philosophers (Lecture 7 page 11-12).
- Try to show that a process defined by the parallel combinator is equivalent to a non-stopping process defined without this combinator (Lecture 6 page 19).
  - Long and tedious algebraic transformations.
- Try to appeal to the general law

L2. If P and Q never stop **and** $|\alpha P \cap \alpha Q| \leq 1$, **then** $(P \parallel Q)$ never stops.

X2. The process $(P \parallel Q)$ with $P = (a \rightarrow c \rightarrow P)$ and $Q = (c \rightarrow b \rightarrow Q)$ never stops, because

- $\alpha P \cap \alpha Q = \{c\}$

Concurrency: **Specifications**

- The proof rule for change of symbol:

L3. If P sat $S(tr)$ **then** $f(P)$ sat $S(f^{-1*}(tr))$

- The use of f^{-1} in the consequent:
  - Let tr be a trace of $f(P)$.
  - Then $f^{-1*}(tr)$ is a trace of P.
  - The antecedent of L3 states that every trace of P satisfies S.
  - Hence $f^{-1*}(tr)$ satisfies S, which is stated by the consequent of L3.

Nondeterminism

Nondeterminism: **Introduction**

- *Deterministic processes*
  - if there is more than one event possible, the choice between them is determined externally *by the environment* of the process.
  - Described by
    - the choice operator $(x : B \rightarrow P(x))$
      - to define a process with a range of possible behaviours;
    - the concurrency operator $\|$
      - to make a selection *by an environment* between the alternatives in set B.
  - The change-giving machine *CH5C* offers its customer the choice of taking his change as three small coins and one large, or two large coins and one small.
- Such processes are determined either
  - in the sense that the environment can actually make the choice, or
  - in the weaker sense that the environment can observe which choice has been made at the very moment of the choice.

Nondeterminism: **Introduction**

- *Nondeterministic processes*
  - has a range of possible behaviours,
  - the *environment cannot influence* the selection between the alternatives.
  - A change-giving machine gives change in either of the combinations uncontrollably.
- The choice is made by the machine itself, in an arbitrary fashion.
- The environment *may infer* which choice was made from the subsequent process behaviour.
- This kind of nondeterminism from ignoring the factor which influence the selection.
  - The combination of change may depend on the number of large and small coins.
    - we have excluded these events (the numbers) from the alphabet.
- Nondeterminism maintains a high level of abstraction in descriptions of the behaviour of physical systems and machines.

Nondeterminism: **Nondeterministic or**

- Nondeterministic **or**
  - $P \sqcap Q$ (P or Q)
  - a process which behaves either like process P or like process Q
    - the selection between them is made arbitrarily,
      - without the knowledge of control of the environment.
- The alphabets of the operands the same
  - $\alpha(P \sqcap Q) = \alpha P = \alpha Q$

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

Nondeterminism: **Nondeterministic or**

X1. A change-giving machine which gives the right change in one of two combinations on each occasion of use.

$$\begin{aligned} CH5D = (in5p \rightarrow ((&out1p \rightarrow out1p \rightarrow out1p \rightarrow out2p \rightarrow CH5D) \\ &\sqcap (out2p \rightarrow out1p \rightarrow out2p \rightarrow CH5D))) \end{aligned}$$

X2. *CH5E* always gives the same combination, but we do not know initially which it will be

$$CH5E = CH5A \sqcap CH5B$$

- After this machine gives its first coin in change, its subsequent behaviour is entirely predictable.
  - $CH5D \neq CH5E$

Nondeterminism: **Nondeterministic or**

Implementation and specification.

- $\sqcap$ is not a useful operator for *implementing* a process.
  - It would be very foolish to build both P and Q, put them in a black bag, make an arbitrary choice between them, and then throw the other one away!
- The main advantage of nondeterminism is in *specifying* a process.
  - To analyze and show the correctness for a range of behaviors.
  - A process specified as $(P \sqcap Q)$ can be implemented
    - either by building P or by building Q.
      - The choice can be made in advance by the implementor on grounds not relevant in the specification,
        - low cost, fast response times, early delivery, etc.

Nondeterminism: Nondeterministic or: **Laws**

- 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:

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

- The occasion on which a nondeterministic choice is made is not significant:

L4. $x \rightarrow (P \sqcap Q) = (x \rightarrow P) \sqcap (x \rightarrow Q)$ (distribution)
  - A process which first does x and then makes a choice is indistinguishable from
    - a process which first makes the choice and then does x.
  - This law states that the prefixing operator distributes through nondeterminism.

Nondeterminism: Nondeterministic or: **Laws**

- A dyadic operator is said to be distributive if
  - it distributes through in both its argument positions independently.
- The following laws state distributivity of some operators defined so far:

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

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

Nondeterminism: Nondeterministic or: **Laws**

- The recursion operator is *not* distributive.
  - Except in the trivial case where the operands of are identical.

- The counterexample:
  - The following processes are different:
    - $P = \mu X \bullet ((a \rightarrow X) \sqcap (b \rightarrow X))$
    - $Q = (\mu X \bullet (a \rightarrow X)) \sqcap (\mu X \bullet (b \rightarrow X))$
  - P can make an independent choice between a and b on each iteration
    - its traces include $\langle a, b, b, a, b \rangle$
  - Q must make a choice between always doing a and always doing b
    - its traces include just repetition of a or repetition of b.
  - P may choose always to do a or always to do b
    - $traces(Q) \subseteq traces(P)$

Nondeterminism: Nondeterministic or: **Laws**

Fairness

- In some theories, *nondeterminism is obliged to be fair*
  - an event that infinitely often *may* happen eventually *must* happen
    - no limit to how long it may be delayed.
- In our theory, *there is no such concept of fairness*.
  - we observe *only finite traces* of the behaviour of a process
    - if an event can be postponed indefinitely, we can never tell whether it is going to happen or not.
  - If the event shall happen eventually
    - there should be a number n such that every trace longer than n contains the event.

Nondeterminism: Nondeterministic or: **Laws**

- The process must be designed *explicitly* to satisfy this fairness constraint.
  - In the process P_0, event a must always occur within n steps of its previous occurrence
    - $P_i = (a \rightarrow P_0) \sqcap (b \rightarrow P_{i+1})$
    - $P_n = (a \rightarrow P_0)$
    - Both Q and P_0 are valid implementations of P.
- If fairness of nondeterminism is required
  - this should be specified and implemented *at a separate stage*
    - for example, by ascribing nonzero probabilities to the alternatives of a nondeterministic choice.
      - Highly desirable to separate complex probabilistic reasoning from concerns about the logical correctness of the process behaviour.

Nondeterminism: Nondeterministic or: **Traces**

- If s is a trace of P (or Q), then
  - s is also a possible trace of $(P \sqcap Q)$ in the case that P (or Q) is selected.
- Conversely, each trace of $(P \sqcap Q)$ must be a trace of one or both alternatives.

L1. $traces(P \sqcap Q) = traces(P) \cup traces(Q)$

- The behaviour of $(P \sqcap Q)$ after s is defined by whichever of P or Q engages in s
  - if both could, the choice remains nondeterministic.

L2.
$$
\begin{aligned}
(P \sqcap Q) / s &= Q / s && \text{if } s \in (traces(Q) - traces(P)) \\
&= P / s && \text{if } s \in (traces(P) - traces(Q)) \\
&= (P / s) \sqcap (Q / s) && \text{if } s \in (traces(P) \cap traces(Q))
\end{aligned}
$$

Nondeterminism: **General choice**

- The environment of $(P \sqcap Q)$ cannot control the choice between P and Q, and must be prepared to deal with either P or Q.
- General choice operation $(P \,\square\, Q)$
  - the environment *can* control which of P and Q will be selected
    - this control is exercised on the very first action.
      - If this action is *not* a possible first action of P, then Q will be selected.
      - If this action is *not* a possible first action of Q, then P will be selected.
      - If the first action is possible for both P and Q, then
        - the choice between them is nondeterministic.
      - If the event is impossible for both P and Q, then it just cannot happen.

Nondeterminism: **General choice**

- The alphabets of P and Q are the same:
  - $\alpha(P \,\square\, Q) = \alpha P = \alpha Q$
- The convention:
  - $\rightarrow$ binds more tightly than $\square$.

- If no initial event is possible for both processes, $\square$ operator is the same as $\mid$ operator:
  - $(c \rightarrow P \,\square\, d \rightarrow Q) = (c \rightarrow P \mid d \rightarrow Q)$ if $c \neq d$.

- If the initial events are the same, $(P \,\square\, Q)$ degenerates to nondeterministic choice:
  - $(c \rightarrow P \,\square\, c \rightarrow Q) = (c \rightarrow P \sqcap c \rightarrow Q)$

Nondeterminism: General choice: **Laws**

- The algebraic laws for $\square$ are similar to those for $\sqcap$:

L1–L3. $\square$ is idempotent, symmetric, and associative.

L4. $P \,\square\, STOP = P$

- The formalisation of the informal definition of the operation:

L5. $(x : A \rightarrow P(x)) \,\square\, (y : B \rightarrow Q(y)) = (z : (A \cup B) \rightarrow$
$(\text{if } z \in (A - B) \text{ then } P(z)$
$\text{else if } z \in (B - A) \text{ then } Q(z)$
$\text{else if } z \in (A \cap B) \text{ then } (P(z) \sqcap Q(z))))$

- $\sqcap$ distributes through $\square$:

L6. $P \,\square\, (Q \sqcap R) = (P \,\square\, Q) \sqcap (P \,\square\, R)$

- $\square$ distributes through $\sqcap$:

L7. $P \sqcap (Q \,\square\, R) = (P \sqcap Q) \,\square\, (P \sqcap R)$

  - Choices made nondeterministically and choices made by the environment are *independent*.

Nondeterminism: General choice: **Laws**

- John is the agent which makes nondeterministic choices
- Mary is the environment.
- On the left-hand side of the law:
  - John chooses ($\sqcap$) between P and letting Mary choose ($\square$) between Q and R.
- On the right-hand side Mary chooses either:
  - to offer John the choice between P and Q or
  - to offer John the choice between P and R.

L7. $P \sqcap (Q \square R) = (P \sqcap Q) \square (P \sqcap R)$

- On both sides of the equation:
  - if John chooses P, then P will be the overall outcome,
  - if John does not select P, the choice between Q and R is made by Mary.
- Thus the results of the choice left- and right-hand strategies are always equal.
- The same reasoning is applicable to L6.

Nondeterminism: General choice: **Traces**

- Every trace of $(P \square Q)$ must be a trace of P or a trace of Q, and conversely:

L1. $traces(P \square Q) = traces(P) \cup traces(Q)$

- The next law is slightly different from the law L2 for $\sqcap$

L2. $(P \square Q) \,/\, s = P \,/\, s$ if $s \in traces(P) - traces(Q)$

$= Q \,/\, s$ if $s \in traces(Q) - traces(P)$

$= (P \,/\, s) \sqcap (Q \,/\, s)$ if $s \neq \langle\rangle$ and $s \in traces(P) \cap traces(Q)$

L2. $(P \sqcap Q) \,/\, s = Q \,/\, s$ if $s \in (traces(Q) - traces(P))$

$= P \,/\, s$ if $s \in (traces(P) - traces(Q))$

$= (P \,/\, s) \sqcap (Q \,/\, s)$ if $s \in (traces(P) \cap traces(Q))$

Nondeterminism: **Refusals**

The distinction between $(P \sqcap Q)$ and $(P \,\Box\, Q)$

- They cannot be distinguished by their traces.
- But there is an environment in which $(P \sqcap Q)$ can deadlock at its first step, but $(P \,\Box\, Q)$ cannot.
  - Let $x \neq y$ and $P = (x \rightarrow y \rightarrow P)$, $Q = (y \rightarrow x \rightarrow Q)$, $\alpha P = \alpha Q = \{x, y\}$
    - Then $(P \,\Box\, Q) \parallel P = (x \rightarrow y \rightarrow P) = P$
    - But $(P \sqcap Q) \parallel P = (P \parallel P) \sqcap (Q \parallel P) = P \sqcap STOP$
  - In environment P, process $(P \sqcap Q)$ may reach deadlock but process $(P \,\Box\, Q)$ cannot.

Nondeterminism: **Refusals**

- Let X be a set of events offered initially by the environment Q of a process P
  - $\alpha P = \alpha Q$.
- Set X is a *refusal* of P
  - If P *may* go to deadlock on its first step when placed in Q.
- The set of all such refusals of P is *refusals(P)*.
  - A family of sets of events.

- Refusal sets is one of the important indirectly observable aspects of the behaviour.

Nondeterminism: **Refusals**

Formal distinction between deterministic and nondeterministic processes

- A process is *deterministic* if it can never refuse any event in which it can engage:
  - P is deterministic $\Rightarrow (X \in refusals(P) \equiv (X \cap P_0 = \{\}))$, where $P_0 = \{\, x \mid \langle x \rangle \in traces(P)\}$.
- This condition holds after any possible sequence of actions of P:
  - P is deterministic $\equiv \forall s : traces(P) \bullet (X \in refusals(P \,/\, s) \equiv (X \cap (P \,/\, s)^0 = \{\}))$
- For a *nondeterministic* process, there is at some time some event in which
  - it can engage and
  - it may refuse to engage in that event, even though the environment is ready for it
    - as a result of some internal nondeterministic choice.

Nondeterminism: Refusals: **Laws**

- The process *STOP* does nothing and refuses everything:

L1. $refusals(STOP_A)$ = all subsets of A (including A itself)

- A process $c \rightarrow P$ refuses every set that does not contain the event c:

L2. $refusals(c \rightarrow P) = \{X \mid X \subseteq (\alpha P - \{c\})\}$

- A generalisation:

L3. $refusals(x : B \rightarrow P(x)) = \{X \mid X \subseteq (\alpha P - B)\}$

- If P (Q) can refuse X, so will $P \sqcap Q$ if P (Q) is selected:

L4 $refusals(P \sqcap Q) = refusals(P) \cup refusals(Q)$

- If *both* P and Q can refuse X, so can $(P \,\square\, Q)$:

L5. $refusals(P \,\square\, Q) = refusals(P) \cap refusals(Q)$

- Comparison of L5 with L4 shows the distinction between $\square$ and $\sqcap$.

Nondeterminism: Refusals: **Laws**

- The law for concurrency:

L6. $refusals(P \parallel Q) = \{X \cup Y \mid X \in refusals(P) \wedge Y \in refusals(Q)\}$

- For symbol change:

L7. $refusals(f(P)) = \{f(X) \mid X \in refusals(P)\}$

- A process can refuse only events in its own alphabet:

L8. $X \in refusals(P) \Rightarrow X \subseteq \alpha P$

- A process deadlocks when the environment offers no events:

L9. $\{\} \in refusals(P)$

- If a process refuses a nonempty set, it can also refuse any subset of that set:

L10. $(X \cup Y) \in refusals(P) \Rightarrow X \in refusals(P)$

- Any event x which cannot occur initially may be added to any set X already refused:

L11. $X \in refusals(P) \Rightarrow (X \cup \{x\}) \in refusals(P) \vee \langle x \rangle \in traces(P)$

Nondeterminism: **Specifications**

- Specification of indirectly observable aspects of the behaviour.
  - describe the desired properties of process refusal sets as well as its traces.
- The variable *ref* denotes an arbitrary refusal set of a process.
- Specification of nondeterministic process *P*:
  - P sat $S(tr, ref)$ iff
    - $\forall tr, ref \bullet tr \in traces(P) \wedge ref \in refusals(P\ /\ tr) \Rightarrow S(tr, ref)$

$$VMCRED = \mu X \bullet (coin \to choc \to X \mid choc \to coin \to X)$$
$$VMS2 = (coin \to VMCRED)$$
$$VMS = (coin \to (choc \to VMS))$$

Nondeterminism: **Specifications**

X1. When a vending machine has ingested more coins than it has dispensed chocolates,

- it must not refuse to dispense a chocolate $FAIR = (tr \downarrow choc < tr \downarrow coin \Rightarrow choc \notin ref)$
- *Every* trace *tr* and *every* refusal *ref* of the specified process *at all times*
  - should satisfy the specification.

X2. When a vending machine has given out as many chocolates as have been paid for

- it must not refuse a further coin $PROFIT = (tr \downarrow choc = tr \downarrow coin \Rightarrow coin \notin ref)$

X3. A simple vending machine should satisfy the combined specification

$$NEWVMSPEC = FAIR \wedge PROFIT \wedge (tr \downarrow choc \leq tr \downarrow coin)$$

- This specification is satisfied by
  - *VMS*
  - *VMS2*
    - may accept several coins in a row, and then give out several chocolates.

$L3.$ $refusals(x : B \rightarrow P(x)) = \{X \mid X \subseteq (\alpha P - B)\}$

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

$VMS2 = (coin \rightarrow VMCRED)$

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

Nondeterminism: **Specifications**

X4. A limit on the balance of coins which may be accepted in a row

$$ATMOST2 = (tr \downarrow coin - tr \downarrow choc \leq 2)$$

X5. The machine accept at least two coins in a row:

$$ATLEAST2 = (tr \downarrow coin - tr \downarrow choc < 2 \Rightarrow coin \notin ref)$$

X6. The process *STOP* refuses every event in its alphabet.

- The predicate specifies that a process with alphabet *A* never stops
  - $NONSTOP = (ref \neq A)$

$$NEWVMSPEC \Rightarrow ref \neq \{coin, choc\}$$

- If *P* sat *NONSTOP* and an environment allows all events in *A*,
  - *P* must perform one of them.
- Since any process which satisfies *NEWVMSPEC* will never stop.

$$NEWVMSPEC = FAIR \wedge PROFIT \wedge (tr \downarrow choc \leq tr \downarrow coin)$$