Introduction to Communicating Sequential Process (CSP) (Lecture 5)

Mannheim, September 2007

Contents

- Concurrency
  - The dining philosophers
  - Interleaving
  - Generalised Parallel Composition
- Relabelling

Deadlock: The Dining Philosophers

- Five philosophers.
- A round table with five chairs labelled with the philosopher's name: $PHIL_0$, $PHIL_1$, $PHIL_2$, $PHIL_3$, $PHIL_4$.
- Between two philosophers there is a fork. There are only five forks. In the center of the table there is a bowl of spaghetti, which is frequently filled.
- In order to eat a philosopher must pick up the forks on either side of him: first the left one and after the right one. A philosopher who cannot pickup one or other fork has to wait.
- After eating the philosophers put down both forks on the table.

Deadlock: The Dining Philosophers

- $\alpha PHIL_i = \{i.sitsdown, i.getsup, i.picksupfork.i, i.picksup fork.(i \oplus 1), i.putsdownfork.i, i.putsdownfork.(i \oplus 1)\}$

- $\oplus$ denotes addition modulo 5, so $i \oplus 1$ identififies the right neighbour of the i^{th} philosopher.

- The philosophers do not interact; their alphabets are disjoint.

- $\alpha FORKi = \{i.picksupfork.i, (i\_1).picksupfork.i, i.putsdown fork.i, (i\_1).putsdownfork.i\}$

  (where _ denotes subtraction modulo 5)

Deadlock: The Dining Philosophers

- $PHIL_i = (i.sitsdown \rightarrow i.picksupfork.i \rightarrow i.picksup fork.(i \oplus 1) \rightarrow i.putsdownfork.i \rightarrow i.putsdownfork.(i \oplus 1) \rightarrow i.getsup \rightarrow PHIL_i)$

- $FORK_i = (i.picksupfork.i \rightarrow i.putsdownfork.i \rightarrow FORK_i \mid (i\_1).picksupfork.i \rightarrow (i\_1).putsdownfork.i \rightarrow FORK_i)$

- $PHILOS = || \; i:\{0..4\}@ \; [\alpha PHIL_i] \; PHIL_i$
- $FORKS = || \; i:\{0..4\}@ \; [\alpha FORK_i] \; FORK_i$
- $COLLEGE = (PHILOS \; [\bigcup \alpha PHIL_i \; || \; \bigcup \alpha FORK_i] FORKS)$

Deadlock: The Dining Philosophers

- Deadlock

  Assume all philosophers get hungry at once. They sit down and pick up their left fork. For then, none can make any progress and the system is deadlocked. The philosophers starve to death.

  Asuming the trace

  t = <0.sitsdown, ..., 4.sitsdown, 0. piksupfork 0, ..., 4.picksupfork 4>

  COLLEGE after t = STOP

Deadlock: The Dining Philosophers

- Solutions:
  - An extra fork? Resources expensive . . .
  - Exactly one philosopher picks up the ‘wrong’ fork first? Asymmetric . . .
  - Footman (monitor) *Fm* who allows at most four philosophers to be seated at once.

  Let

$$U = \bigcup_{i=0}^{4} \{i.getsup\} \qquad D = \bigcup_{i=0}^{4} \{i.sitsdown\}$$

Deadlock: The Dining Philosophers

- $FOOT_j$ define the behaviour of the *footman* with j philosophers seated:

$$FOOT_0 = (x{:}D \rightarrow FOOT_1)$$
$$FOOT_j = (x{:}D \rightarrow FOOT_{j+1} \mid y{:}U \rightarrow FOOT_{j-1})$$
$$FOOT_4 = (y{:}U \rightarrow FOOT_3)$$

- A *deadlock* free *college* is defined as:

$$NEWCOLLEGE = (COLLEGE\ [\{\alpha COLLEGE\} \,||\, \{U,D\}]\ FOOT_0)$$

Deadlock: The Dining Philosophers

- Home exercise: Express *footman* using external and conditional choice.

Interleaving

- If P and Q are processes then their interleaving

$$P \mid\mid\mid Q$$

  is the process which interleaves events from P and Q without synchronisation; when events would synchronise in the parallel composition, now the choice between them is nondeterministic.
- Useful to specify parallel composition of clients in a client-server system.

Interleaving

$P \mid\mid\mid Q$

- Offer the initial events of both P and Q, and wait until a communication happens.
- After the communication of na event a of P (or Q) behaves as $P' \mid\mid\mid Q$ (or as $P \mid\mid\mid Q'$), where P' (or Q') behaves as P (or Q) after the communication of a.

Interleaving : Example

- A fax machine can be describe as:

$$FAX = accept?d : DOCUMENT \text{ -> } print!d \text{ -> } FAX$$

- A colection of four fax machines can be connected in a same telephone number (with four lines): each one is ready to receive inputs.

$$FAXES = (FAX \mid\mid\mid FAX) \mid\mid\mid (FAX \mid\mid\mid FAX)$$

- The system can accept four faxes before printing.
- Forks don't communicate with each other, but neither do the philosophers. So *PHILOS* and *FORKS* can be expressed as

$$PHILOS = \mid\mid\mid\ i{:}\{0..4\} @ \ [\alpha PHILi]\ PHILi$$
$$FORKS = \mid\mid\mid\ i{:}\{0..4\} @ \ [\alpha FORKi]\ FORKi$$

Interleaving : Laws

- Interleaving is commutative and associative.
- It distributes over internal choice

$$P \mid\mid\mid (Q \sqcap R) = (P \mid\mid\mid Q) \sqcap (P \mid\mid\mid R)$$

- The synchronisation is expressed by:

$$\begin{aligned} Assume\ \ P &= ?a : A \text{ -> } P(a) \\ Q &= ?b : B \text{ -> } Q(b) \end{aligned}$$

$$P \mid\mid\mid Q$$

$$=$$

$$\begin{aligned} ?c: A \cup B \text{ -> } & (P(c) \mid\mid\mid Q) \sqcap (P \mid\mid\mid Q(c)) \\ & \lhd c \in A \cap B \rhd \\ & (P(c) \mid\mid\mid Q) \lhd c \in A \rhd (P \mid\mid\mid Q(c)) \end{aligned}$$

Interleaving: Exercise

A garage has two petrol pumps that operate independtly. Write a CSP process of the petrol supply service offered by the garage.

Interleaving: Exercise

A garage has two petrol pumps that operate independtly. Write a CSP process of the petrol supply service offered by the garage.

$SERVICE = PUMP1 \; ||| \; PUMP2$ *or*

$SERVICE = \; ||| \; i{:}\{1..2\} @ \; [\alpha PUMPi] \; PUMPi$

$PUMPi = \ldots$

Generalised Parallel Composition

- Let P and Q be processes and X be a set of events. Then the general parallel composition is

  $P\ [|X|]\ Q$

- P and Q execute in parallel and synchronise the events in X:
  - the other events happens independently

Generalised Parallel Composition

- The other operators can be expressed using general parallel composition:
  - $P \parallel Q = P \; [|Events|] \; Q$
  - $P \; [X||Y] \; Q = P[|X \cap Y|] \; Q$, *if P and Q do not communicate outside X and Y.*
  - $P \; ||| \; Q = P \; [| \; \{\} \; |] \; Q$

Generalised Parallel Composition: Examples

- In a competition, a runner engajes into two events *start* and *finish*:

  RUNNER = start -> finish -> STOP

- Two runners must synchronise in event *start* but can finish independently. The race can be described as

  RACE = RUNNER [|{start}|] RUNNER

Generalised Parallel Composition : Examples

- Two sequences of numbers are received through channels *left1* and *left2*. For each *x* read in *left1* and each *y* read in *left2*, the number *(ax + by)* is output using channel right. The multiplication must occur concurrently.

$$X21 = (left1?x \rightarrow mid!(ax) \rightarrow X21)$$
$$X22 = (left2?y \rightarrow mid?z \rightarrow right!(z+by) \rightarrow X22)$$
$$X2 = (X21\ [|\{mid\}|]\ X22)$$

Generalised Parallel Composition: Laws

- Let P, Q and R CSP processes CSP, X and Y set of events. Then

  $$(P\ [|X|]\ Q)\ [|Y|]\ R \neq P\ [|X|]\ (Q\ [|Y|]\ R)$$

- This associativity is valid only if $X = Y$

  $$(P\ [|X|]\ Q)\ [|X|]\ R = P\ [|X|]\ (Q\ [|X|]\ R)$$

Generalised Parallel Composition: Laws

- The general law is given by:

Assume $C = (A \setminus X)) \cup (B \setminus X)) \cup (A \cap B \cap X),$

$P = ?a : A \text{ -> } P(a)$

$Q = ?b : B \text{ -> } Q(b)$

$P\ [|X|]\ Q$

$=$

$$(P(c)\ [|X|]\ Q(c) \lhd c \in X \rhd ((P(c)\ [|X|]\ Q) \sqcap (P\ [|X|]\ Q(c))$$

$$\lhd c \in A \cap B \rhd$$

$$((P(c)\ [|X|]\ Q)$$

$$\lhd c \in A \rhd (P\ [|X|] Q(c)))))$$

Generalised Parallel Composition: Exercise

- On entering a restaurant, the cloackroom attendant might help the customer off and on with her coat, as captured by the events *coat.off* and *coat.on* respectively, storing and retrieve coats as appropriate. Write a CSP parallel process to model this activity.

Generalised Parallel Composition: Exercise

- On entering a restaurant, the cloackroom attendant might help the customer off and on with her coat, as captured by the events *coat.off* and *coat.on* respectively, storing and retrieve coats as appropriate. Write a CSP parallel process to model this activity.

ATT = coat.off -> store -> ATT [] retrieve -> coat.on -> ATT
CUST = enter -> coat.off -> eat -> coat.on -> CUST

REST = ATT [|{coat.on, coat.off}] CUST

Generalised Parallel Composition: Home Exercise

- Which is the equation that express the traces of $P \; [|X|] \; Q$?
- Let $V = coin \;\text{->}\; choc \;\text{->}\; V$. Do the following processes have the same behaviour?

$$(V\ [|\{choc\}|]\ V)\ [|\{coin\}|]\ V$$

$$V\ [|\{choc\}|]\ (V\ [|\{coin\}|]\ V)$$

Relabelling

An *injective* (or one-to-one) function from Σ to Π is one for which

$$\forall\, x , y : \;\bullet\, fx = fy \Rightarrow x = y.$$

If f : $\Sigma \longrightarrow \Pi$ is injective and P is a process over then the injective relabelling $f\,P$ of P by f, is the process over which performs event $fe : \Pi$ iff P performs event $e : \Sigma$.

$$traces(f\,P) = \{map\, f\, t \mid t \in traces\ P\}$$

Relabelling: Example - Inflation

Vending machine Vct has universe $\Sigma = \{coin, choc, toffee\}$

$$Vct = coin \rightarrow (choc \rightarrow Vct \mid toffee \rightarrow Vct).$$

After inflation the universe becomes $\Pi = \{fiver, minichoc, microtoffee\}$ with injective relabelling function f

$f\ coin = fiver$
$f\ choc = minichoc$
$f\ toffee = microtoffee.$

Relabelling: Example - Inflation

The new vending machine is

$$
\begin{aligned}
fVct = \mathit{fiver} \rightarrow (&\mathit{minichoc} \rightarrow fVct \\
\mid\ &\mathit{minitoffee} \rightarrow fVct).
\end{aligned}
$$

Relabelling: Laws

Injective relabelling distributes through every operator.
In particular,

$$f\, STOP = STOP$$
$$f\,(x : A \rightarrow P(x\,)) = f\,(x\,) : f\,(A) \rightarrow f\,P(x\,)$$
$$f\,(P\ [|X|]\ Q) = f\,P\ [|\{fX\}|]\ f\,Q\ \ldots$$

Labelling

If P is a process on Σ then its l labelling $l\,.P$ is the l relabelling of P, $l\,P$, defined by the injective relabelling function which simply places l in front of each event

$$l : \Sigma \longrightarrow l\,.\,\Sigma = \{l\,.\sigma \mid \sigma \in \Sigma\}$$

where

$$l\,\sigma = l.\sigma$$

Functional Relabelling: Example

Example: Entry costs *20p* for a child and *50p* for an adult:

$$\Sigma = \{20p, 50p\},$$

$$Enter = (20p \rightarrow Enterchild\ )[](50p \rightarrow Enteradult).$$

Revision of entry fee, to a single charge of *40p*, results in abstraction which can be achieved by a non-injective (many-to-one) relabelling:

$$\Pi = \{40p\}$$

with f : $\Sigma \longrightarrow \Pi$ defined by

$$f\ 20p = 40p$$

$$f\ 50p = 40p$$

Functional Relabelling: Example

Then

$$f\,Enter = (40p \longrightarrow f\,Enterchild\,)[](40p \longrightarrow f\,Enteradult)$$

= law of []

$$(40p \longrightarrow f\,Enterchild\,) \sqcap (40p \longrightarrow f\,Enteradult)$$

= law of $\sqcap$

$$40p \longrightarrow (f\,Enterchild \sqcap f\,Enteradult).$$

Nondeterminism!

Functional Relabelling

If P is a process and f is a function from Σ to Π then the functional relabelling

$$f\,P$$

of P by f is the process which performs $fe : \Pi$ when P performs $e : \Sigma$. Now P performing distinct events in Σ may result in $f\,P$ performing the same event in Π with nondeterministic choice of consequent.

$$traces(f\,P) = \{map\ f\ t \mid t \in traces\ P\}$$

Home exercise: Study the laws

Relational Relabelling

Example: Recall, with $\Sigma = \{coin, choc\}$,

$$V = coin \rightarrow choc \rightarrow V.$$

A clock with a choice of chimes is obtained by setting

$$\Pi = \{tick, chime, chime'\},$$

taking R : $\Sigma \leftrightarrow \Pi$ to be the (one-to-many) relation

$$coin\ R\ tick$$
$$choc\ R\ chime$$
$$choc\ R\ chime',$$

Relational Relabelling

and considering the R relabelling of V

$$R\,V = tick \rightarrow (chime \rightarrow R\,V \\ \mid chime' \rightarrow R\,V).$$

Relational Relabelling

If P is a process and R is a relation from Σ to Π then the relational relabelling

$$R\ P$$

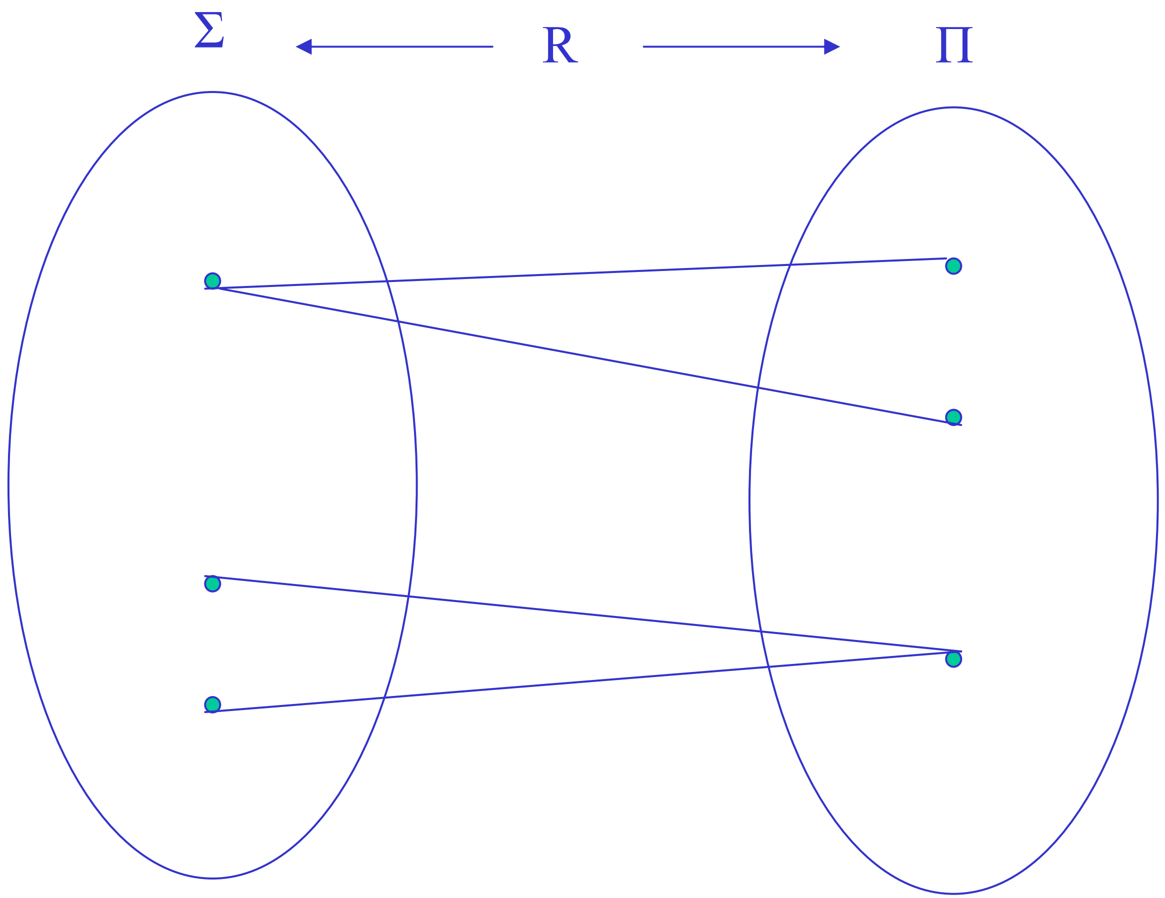
of P by R is the process which, when P performs event e : Σ, offers the events $f : \Pi$ satisfying $e\ R\ f$.
P performing a single event in Σ may result in $R\ P$ performing an external choice of several events in Π.

$$traces(R\ P) = \{u : \Pi^* \mid \square\ t\ \square\ traces\ P \bullet \square\ i \bullet t_i\ R\ u_i\}$$

• Roscoe uses generalised substitution notation $P[R]$ to express relabelling.
• Home Exercise: Study the laws.

Relabelling

Introduces



External choice

Internal choice

Relabelling

- One old event mapped to two new events. (External choice)

```
(a -> STOP)[[a <- a, a <- b]]
=
(a -> STOP) [] (b -> STOP)
```

old <- new

Relabelling

- Two old events mapped to one new event. (Internal choice)

```
(a -> P [] b -> Q)[[a <- b, b <- b]]
=
(b -> P |~| b -> Q)
```