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: PHIL₀, PHIL₁, PHIL₂, PHIL₃, PHIL₄.
- 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

- $\alpha_{PHIL_i} = \{i.\text{sitsdown}, i.\text{getsup}, i.\text{picksupfork}.i, i.\text{picksupfork}.(i \oplus 1), i.\text{putsdownfork}.i, i.\text{putsdownfork}.(i \oplus 1)\}$

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

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

- $\alpha_{FORKi} = \{i.\text{picksupfork}.i, (i_1).\text{picksupfork}.i, i.\text{putsdownfork}.i, (i_1).\text{putsdownfork}.i\}$

  (where _ denotes subtraction modulo 5)
Deadlock: The Dining Philosophers

• \( \text{PHIL}_i = (i.\text{sitsdown} \rightarrow i.\text{picksupfork} \rightarrow i.\text{picksupfork(j \oplus 1)} \rightarrow i.\text{putsdownfork} \rightarrow i.\text{putsdownfork(j \oplus 1)} \rightarrow i.\text{getsup} \rightarrow \text{PHIL}_i) \)

• \( \text{FORK}_i = (i.\text{picksupfork} \rightarrow i.\text{putsdownfork} \rightarrow \text{FORK}_i \mid (i_1).\text{picksupfork} \rightarrow (i_1).\text{putsdownfork} \rightarrow \text{FORK}_i) \)

• \( \text{PHILOS} = \| i:{0..4}@ [\alpha \text{PHIL}_i] \text{PHIL}_i \)
• \( \text{FORKS} = \| i:{0..4}@ [\alpha \text{FORK}_i] \text{FORK}_i \)
• \( \text{COLLEGE} = (\text{PHILOS} \sqcup [\alpha \text{PHIL}_i] \| [\alpha \text{FORK}_i] \text{FORKS}) \)
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
\[ t = <0.sitsdown, ..., 4.sitsdown, 0. pksupfork 0, ..., 4.picksupfork 4> \]

\textit{COLLEGE after } \( t = \text{STOP} \)
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

$$U = \bigcup_{i=0}^{4} \{i.\text{getsup}\}$$

$$D = \bigcup_{i=0}^{4} \{i.\text{sitsdown}\}$$
Deadlock: The Dining Philosophers

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

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

- A deadlock free college is defined as:

  $NEWCOLLEGE = (COLLEGE [{\alpha COLLEGE} \parallel \{U,D\}] FOOT_0)$
Deadlock: The Dining Philosophers

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

• If $P$ and $Q$ are processes then their interleaving $P \parallel Q$ 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

\[ P \parallel|\parallel Q \]

– Offer the initial events of both \( P \) and \( Q \), and wait until a communication happens.

– After the communication of an event \( a \) of \( P \) (or \( Q \)) behaves as \( P' \parallel|\parallel Q \) (or as \( P \parallel|\parallel Q' \)), where \( P' \) (or \( Q' \)) behaves as \( P \) (or \( Q \)) after the communication of \( a \).
Interleaving: Example

• A fax machine can be described as:

\[ FAX = \text{accept } d : \text{DOCUMENT} \rightarrow \text{print}!d \rightarrow FAX \]

• A collection of four fax machines can be connected to a single telephone number (with four lines): each one is ready to receive inputs.

\[ FAXES = (FAX \ ||| FAX) \ ||| (FAX \ ||| FAX) \]

• The system can accept four faxes before printing.

• Forks don’t communicate with each other, but neither do the philosophers. So \textit{PHILOS} and \textit{FORKS} can be expressed as

\[
\begin{align*}
\text{PHILOS} &= \ ||| i:\{0..4\}@[\alpha \text{PHIL}_i] \ \text{PHIL}_i \\
\text{FORKS} &= \ ||| i:\{0..4\}@[\alpha \text{FORK}_i] \ \text{FORK}_i 
\end{align*}
\]
Interleaving : Laws

• Interleaving is commutative and associative.

• It distributes over internal choice

\[ P ||| (Q \sqcap R) = (P ||| Q) \sqcap (P ||| R) \]

• The synchronisation is expressed by:

Assume \( P = \ ?a : A \rightarrow P(a) \)
\( Q = \ ?b : B \rightarrow Q(b) \)

\[ P ||| Q \]
=  
\( ?c : A \cup B \rightarrow (P(c) ||| Q) \sqcap (P ||| Q(c)) \)
\[ \langle c \in A \cap B \rangle \]
\( (P(c) ||| Q) \langle c \in A \rangle (P ||| Q(c)) \)
Interleaving: Exercise

A garage has two petrol pumps that operate independently. Write a CSP process of the petrol supply service offered by the garage.
Interleaving: Exercise

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

\[
\text{SERVICE} = PUMP1 ||| PUMP2 \text{ or } \\
\text{SERVICE} = ||| i:\{1..2\}@ [\alphaPUMPi] PUMPi
\]

\[
PUMPi = ...
\]
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 happen independently
Generalised Parallel Composition

• The other operators can be expressed using general parallel composition:
  
  $- P || Q = P [\mid \text{Events} \mid ] Q$

  $- P [X||Y] Q = P[\mid X \cap Y \mid ] Q$, if $P$ and $Q$ do not communicate outside $X$ and $Y$.

  $- P \|\| Q = P [\mid \{ \} \mid ] Q$
Generalised Parallel Composition: Examples

• In a competition, a runner engages into two events start and finish:
  \[ \text{RUNNER} = \text{start} \rightarrow \text{finish} \rightarrow \text{STOP} \]

• Two runners must synchronise in event start but can finish independently. The race can be described as
  \[ \text{RACE} = \text{RUNNER} \left[\left\{\text{start}\right\}\right] \text{RUNNER} \]
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.

\[
X_{21} = (left1?x → mid!(ax) → X21)
\]
\[
X_{22} = (left2?y → mid?z → right!(z+by) → X22)
\]
\[
X2 = (X21 ⋄ \{mid\} ⋄ X22)
\]
Generalised Parallel Composition: Laws

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

$$(P \mid X \mid Q) \mid Y \mid R \neq P \mid X \mid (Q \mid Y \mid R)$$

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

$$(P \mid X \mid Q) \mid X \mid R = P \mid X \mid (Q \mid X \mid R)$$
Generalised Parallel Composition: Laws

- The general law is given by:
  
  Assume\[ C = (A \setminus X) \cup (B \setminus X) \cup (A \cap B \cap X), \]
  \[ P = ?a : A \rightarrow P(a) \]
  \[ Q = ?b : B \rightarrow Q(b) \]

\[
P \mid X \mid Q
= (P(c) \mid X\mid Q(c) \triangleleft c \in X \triangleright ((P(c) \mid X\mid Q) \sqcap (P \mid X\mid Q(c))) \]
  \[ \triangleleft c \in A \cap B \triangleright ((P(c) \mid X\mid Q) \]
  \[ \triangleleft c \in A \triangleright (P \mid X\mid Q(c)))\]
Generalised Parallel Composition: Exercise

- On entering a restaurant, the cloakroom 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 cloakroom 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 \( V = \text{coin} \rightarrow \text{choc} \rightarrow V \). Do the following processes have the same behaviour?

\[
(V [|\{\text{choc}\}|] V) [|\{\text{coin}\}|] V \\
V [|\{\text{choc}\}|] (V [|\{\text{coin}\}|] V)
\]
Relabelling

An injective (or one-to-one) function from $\Sigma$ to $\Pi$ is one for which

$$x, y : \bullet fx = fy \quad x = y.$$ 

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

$$traces(fP) = \{\text{map } f t \mid t \quad traces \ P\}$$
Relabelling: Example - Inflation

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

$$Vct = \text{coin} \rightarrow (\text{choc} \rightarrow Vct$$

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$| \text{toffee} \rightarrow Vct).$$

After inflation the universe becomes

$\Pi = \{\text{fiver, minichoc, microtoffee}\}$

with injective relabelling function $f$

$$f \text{ coin} = \text{fiver}$$

$$f \text{ choc} = \text{minichoc}$$

$$f \text{ toffee} = \text{microtoffee}.$$
Relabelling: Example - Inflation

The new vending machine is

\[ f_{Vct} = \text{fiver} \rightarrow (\text{minichoc} \rightarrow f_{Vct} \mid \text{minitoffee} \rightarrow f_{Vct}). \]
Relabelling: Laws

Injective relabelling distributes through every operator. In particular,

\[
\begin{align*}
&f \text{ STOP} = \text{ STOP} \\
&f (x : A \to P(x)) = f (x) : f (A) \to f P(x) \\
&f (P [\{X\} fX] Q) = f P [\{fX\}] f Q \ldots
\end{align*}
\]
Labelling

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

$$l : \Sigma \rightarrow l \cdot \Sigma = \{l \cdot \sigma \mid \sigma \in \Sigma\}$$

where

$$l \sigma = l \cdot \sigma$$
Functional Relabelling: Example

Example: Entry costs 20p for a child and 50p for an adult:
\[ \Sigma = \{20p, 50p\}, \]
\[ \text{Enter} = (20p \rightarrow \text{Enterchild})[](50p \rightarrow \text{Enteradult}). \]

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 \( f : \Sigma \rightarrow \Pi \) defined by
\[ f\ 20p = 40p \]
\[ f\ 50p = 40p \]
Functional Relabelling: Example

Then

\[
f \text{Enter} = (40p \to f \text{Enter}_\text{child})[](40p \to f \text{Enter}_\text{adult})
\]

= law of []

\[
(40p \to f \text{Enter}_\text{child}) \sqcap (40p \to f \text{Enter}_\text{adult})
\]

= law of \sqcap

\[
40p \to (f \text{Enter}_\text{child} \sqcap f \text{Enter}_\text{adult}).
\]

Nondeterminism!
Functional Relabelling

If $P$ is a process and $f$ is a function from $\Sigma$ to $\Pi$ then the functional relabelling

$$fP$$

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

$$\text{traces}(fP) = \{ \text{map } f t \mid t \text{ traces } P \}$$

Home exercise: Study the laws
Relational Relabelling

Example: Recall, with $\Sigma = \{\text{coin, choc}\}$, $V = \text{coin} \rightarrow \text{choc} \rightarrow V$.

A clock with a choice of chimes is obtained by setting $\Pi = \{\text{tick, chime, chime}',\}$, taking $R : \Sigma \leftrightarrow \Pi$ to be the (one-to-many) relation

$\text{coin} R \text{tick}$
$\text{choc} R \text{chime}$
$\text{choc} R \text{chime}'$, 
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 $\Sigma$ to $\Pi$ then the relational relabelling

$$R \ P$$

of $P$ by $R$ is the process which, when $P$ performs event $e : \Sigma$, offers the events $f : \Pi$ satisfying $e R f$.

$P$ performing a single event in $\Sigma$ may result in $R \ P$ performing an external choice of several events in $\Pi$.

$$\text{traces}(R \ P) = \{u : \Pi^* | \ t \ \text{traces} \ P \cdot \ i \cdot t_i \ R \ u_i\}$$

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

Introduces

External choice

Internal choice

$\Sigma \quad R \quad \Pi$
Relabelling

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

\[
(a \rightarrow \text{STOP})[[a \leftarrow a, a \leftarrow b]]
\]

\[
= \quad \text{(a \rightarrow \text{STOP}) \[\] (b \rightarrow \text{STOP})}
\]
Relabelling

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

\[(a \rightarrow P \ [\] \ b \rightarrow Q)[[a \leftarrow b, b \leftarrow b]] = (b \rightarrow P \ |\sim| \ b \rightarrow Q)\]