Concurrent Computing

Lecturers: Prof. Majid Mirmehdi (majid@cs.bris.ac.uk)
Dr. Tilo Burghardt (tilo@cs.bris.ac.uk)
Dr. Simon Hollis(simon@cs.bris.ac.uk)
Dr. Daniel Page (page@cs.bris.ac.uk)

Web: http://www.cs.bris.ac.uk/Teaching/Resources/COMS20001

LECTURE 10
(Interactive Lecture)

Bridging the Gap
XC vs CSP

[Many thanks to Kerstin Eder, major parts of the CSP in these lecture slides are taken from or based on materials originally prepared by her.]

Machine Interpretable CSP Notation

CSP_m

…is a machine readable version of CSP. It lacks the full versatility of standard (blackboard) CSP as introduced. However, it can be used as input to tools for automated analysis such as FDR2.

Basic Concept	CSP Examples	CSPm Examples
Processes	PA,PB	PA,PB
Events	e,f	channel e,f
Successful Termination	SKIP	SKIP
Tick Event	√	*not available in CSPm*
Deadlock	STOP	STOP
Alphabet	α(P)={e,f}	alphabetname(P)={e,f}
Behaviour	P=e→P	P = e -> P
Channel Input	P=e?x→P	P = e?x -> P
Channel Output	P=e!x→P	P = e!x -> P

CSP_m : Choice, Parallel, Replication

CSPm External Choice Fragment

```
PA = e1 -> PB [] e2 -> PA
```

CSPm Internal Choice Fragment

```
PA = e1 -> PB |~| e2 -> PA
```

CSPm Interleaved Parallel Fragment

```
P = Q ||| R
P = (Q ||| R) ||| S
```

CSPm Alphabetised Parallel Example Fragment

```
A = {e1,e2}
B = {e2,e3}

P = Q [A||B] R
```

CSPm Interleaved Parallel Replication Fragment

```
I = {0..2}
P = ||| x:I @ Q(x)
```

CSPm Alphabetised Parallel Replication Example

```
A = {0..2}
B = {e1,e2,e3}

P = || x:A @ [B] Q(x)
```

XC *vs.* CSP_m Example: Guarded Choice

- **`select`** statement translates to alternative, external choice
- Note: ~~only input guards~~ are supported by XC

```
//XC Example Fragment
...
chan e,g;
int  a;
select {
  case e :> a :
       F(a);
       break;
  case g :> a :
       F(a);
       break;
}
...
```

```
-- CSPm Example Fragment
...

P = e?a -> F(a) [] g?a -> F(a)

...
```

XC *vs.* CSP_m Example: Channel COM

- XC-CSP conversion possible as long as inputs/outputs are paired
- **Remember:** XC can only synchronize **two** threads per channel

```
//XC Example

int main(void) {
  chan e;
  par {
    // sending process P_IN
    { while (1)
        e <: 1;
    }
    // receiving process P_OUT
    { int buffer;
      while (1)
        e :> buffer;
} } }
```

```
-- CSP Example

P_IN  = e!1 → P_IN
P_OUT = e?buffer → P_OUT
MAIN  = P_IN [|e|] P_OUT
```

```
-- CSPm Example

channel e : {0..1}
P_IN  = e!1 -> P_IN
P_OUT = e?buffer -> P_OUT
MAIN  = P_IN [|{|e|}|] P_OUT
```

XC *vs.* CSP_m Example: Synchronization

- **`par`** statement sequences can be used to implement sync events
- **Note:** XC synchronization is mapped to XCore hardware level

```
//XC Example

int main(void) {
  par {
      a();
      b();
  }
  // synchA
  par {
      c();
      d();
  }
  // synchB
}
```

```
-- CSPm Example

PA  = a -> synchA -> PA1

PA1 = c -> synchB -> SKIP

PB  = b -> synchA -> PB1

PB1 = d -> synchB -> SKIP

MAIN = PA [|{|synchA,synchB|}|] PB
```

Mini Example 1:
Transition Diagram, Traces, Refusals & Failures

```
//XC Example Fragment
...
chan e;
int  a;
...
e :> a;
...
```

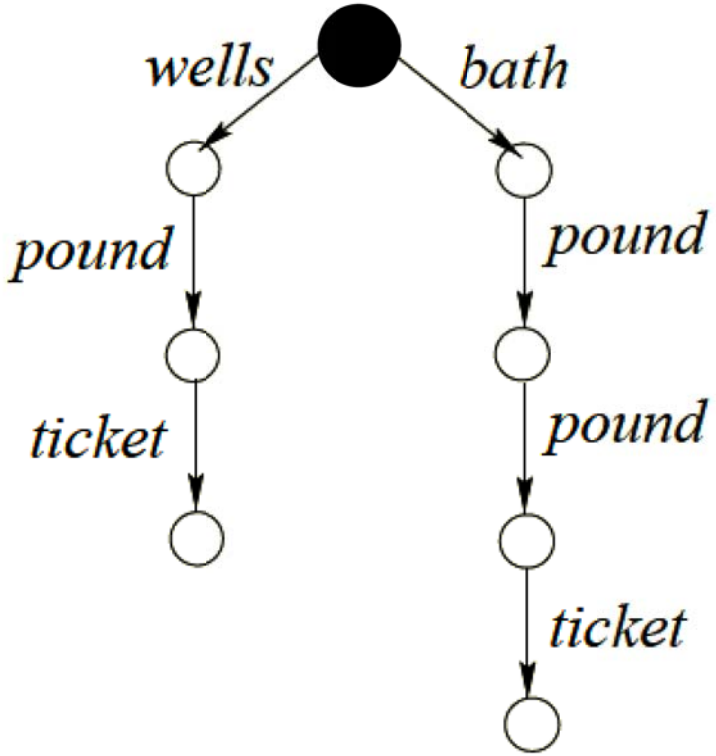
Mini Example 2:
Transition Diagram, Traces, Refusals & Failures

```
//XC Example Fragment
...
chan e,g,h,i;
int  a;
...
select {
  case e :> a :
       h :> void;
       i <: a;
      break;
  case g :> a :
       h :> void;
       h :> void;
       i <: a;
      break;
}
...
```

Example 2

Diagram for simple ticket machine:

- **Circle:** State
- **Labelled Arrow:** Transition
- **Filled Circle:** Initial State
- **Cycles:** Recursion
- **State with no transitions:** Deadlock
- **State with multiple transitions:** Choice



$$TICKETS = wells \rightarrow pound \rightarrow ticket \rightarrow STOP \mid bath \rightarrow pound \rightarrow pound \rightarrow ticket \rightarrow STOP$$

Mini Example 3:
Transition Diagram, Traces, Refusals & Failures

```
...
chan e,g,h,i,j;
int  a,b; ...
while (1) {
  b = 1;
  j :> void;
  while (b) {
    select {
    case j :> void:
       b = 0;
      break;
    case e :> a :
       h :> void;
       i <: a;
      break;
    case g :> a :
       h :> void;
       h :> void;
       i <: a;
      break;
} } } ...
```

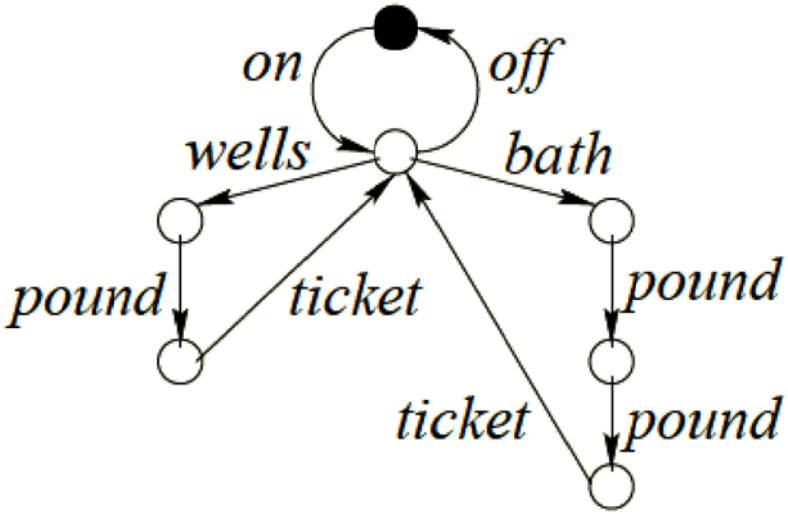
Example 3

Connection between transition diagram of a process, and its traces.

- $MACHINE = on \rightarrow TICKETS$

$$TICKETS = wells \rightarrow pound \rightarrow ticket \rightarrow TICKETS$$
$$\mid\ bath \rightarrow pound \rightarrow pound \rightarrow ticket \rightarrow TICKETS$$
$$\mid\ off \rightarrow MACHINE$$

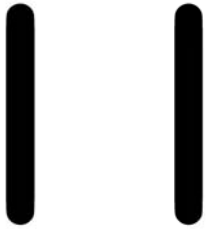
Transition diagram:



traces(*MACHINE*) is the set of traces corresponding to the paths in the diagram starting from the filled-in (black or white) state.

Mini Example 4:
Transition Diagram, Traces, Refusals & Failures

```
...
chan e,g,h,i,j;
...
int  a;
h :> void;
select {
  case e :> a :
      i <: a;
      break;
  case g :> a :
      j <: a;
      break;
} ...
```



```
...
int  b;
h <: 1;
e <: 1;
i :> b;
...
```

Example 4

- Vending machine and customer

$$VM = coin \rightarrow (choc \rightarrow STOP \mid toffee \rightarrow STOP)$$
$$CUST = coin \rightarrow choc \rightarrow STOP$$

Interfaces: $\alpha(VM) = \alpha(CUST) = \{coin, choc, toffee\} = A$

$$VM \;\; {}_A\|_A \;\; CUST$$

$$\downarrow coin$$

$$(choc \text{ -> } STOP \mid toffee \text{ -> } STOP) \;\; {}_A\|_A \;\; choc \text{ -> } STOP$$

$$\downarrow choc$$

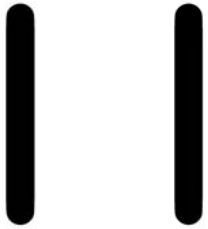
$$STOP \;\; {}_A\|_A \;\; STOP$$

Note: Both components continue to end of (potential) behaviour.

What happens if we change $\alpha(CUST)$ *to* $\{coin, choc\}$*?*

Mini Example 5:
Transition Diagram, Traces, Refusals & Failures

```
...
chan h,i,j;
int  c;
...
h :> void;
if (c) i <: 1;
else j <: 1;
...
```



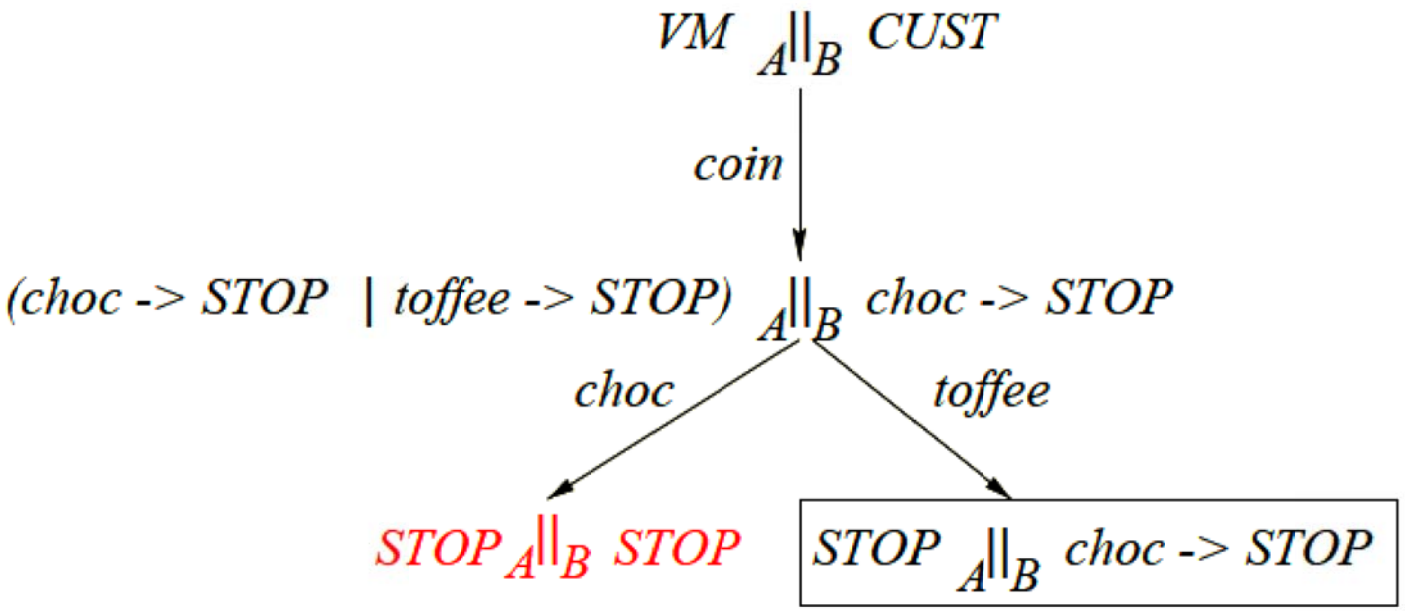
```
...
int  b;
h <: 1;
i :> b;
...
```

Example 5

$$VM = coin \rightarrow (choc \rightarrow STOP \mid toffee \rightarrow STOP)$$
$$CUST = coin \rightarrow choc \rightarrow STOP$$

Interface of VM: $\alpha(VM) = \{coin, choc, toffee\} = A$

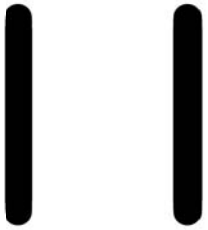
Interface of $CUST$: $\alpha(CUST) = \{coin, choc\} = B$



Note: Customer might get stuck!

Mini Example 6:
Transition Diagram, Traces, Refusals & Failures

```
...
chan h, i;
out port speaker =
          PORT_SPEAKER;
out port led0 =
          PORT_CLOCKLED_0;
...
while (1) {
  h :> void;
  1 :> speaker;
  i <: 1;
} ...
```



```
...
while {1) {
  h <: 1;
  1 :> led0;
  i :> void;
} ...
```

Example 6

A trace of P and a trace of Q can be used to form a trace of $P \,{}_A\|_B\, Q$ *as long as* events in $A \cap B$ appear in the same order in both traces.

- Processes VM and $CUST$ with

$$\alpha(VM) = \{coin, choc, beep\} = A \qquad \alpha(CUST) = \{coin, choc, shout\} = B$$
$$VM = coin \rightarrow beep \rightarrow choc \rightarrow VM$$
$$CUST = coin \rightarrow shout \rightarrow choc \rightarrow CUST$$

$\langle coin, beep, choc \rangle$ is a trace of VM.

$\langle coin, shout, choc \rangle$ is a trace of $CUST$.

Events common in both alphabets are $coin$ and $choc$.

Their order needs to be preserved in $VM_A\|_B CUST$!