Exercise 10

Sindre Hansen

April 28, 2016

1 Safety/Deadlock Assigment

1.1 LTSA detection

The built-in safety criterion works as expected, and LTSA detects the deadlock.

1.2 Detecting deadlock in the FSP model

blank

2 Dining philosophers

2.1 Transition diagram for two philosophers

2.2 FSP model of 3 philosophers and 3 forks

The following code snippet shows a system with 3 philosophers and 3 forks.

```
PHIL = (sitdown->right.get->left.get
          ->eat->left.put->right.put
          ->arise->PHIL).

FORK = (get -> put -> FORK).

||DINERS(N=3)=
   forall [i:0..N-1]
   (phil[i]:PHIL
   ||{phil[i].left,phil[((i-1)+N)%N].right}::FORK).

menu RUN = {phil[0..2].{sitdown,eat}}
```

This is the output when checking the system. It clearly ends in a deadlock.

```
Composition:
DINERS = phil.0:PHIL || {phil.0.left,phil.2.right}::FORK ||
         phil.1:PHIL || {phil.1.left,phil.0.right}::FORK ||
         phil.2:PHIL || {phil.2.left,phil.1.right}::FORK
State Space:
 7 * 2 * 7 * 2 * 7 * 2 = 2 ** 12
Analysing...
Depth 7 -- States: 60 Transitions: 155 Memory used: 3361K
Trace to DEADLOCK:
phil.0.sitdown
phil.0.right.get
phil.1.sitdown
phil.1.right.get
phil.2.sitdown
phil.2.right.get
```

3 Communication

3.1 Synchronous communication

```
range X = 0..2
CHAN = (trans[x:X]->CHAN).
PROD = (c.send[0]->c.send[1]->c.send[2]->PROD)/{c.trans/c.send}.
CONS = (c.receive [v:X]->CONS){c.trans/c.send}.
```

3.2 Asynchronous communication

The following snippet models a five slot buffered channel.

```
const N = 5
BUFFER = COUNT[0],c
COUNT[i:0..N] = (when (i<N) put -> COUNT[i+1] |
                 when (i>0) get -> COUNT[i-1]
                )/{send/put, receive/get}.
PRODUCER = (c.send -> PRODUCER).
CONSUMER = (c.receive -> CONSUMER).

|| BOUNDEDCHAN = (PRODUCER || CONSUMER || c::BUFFER).
```