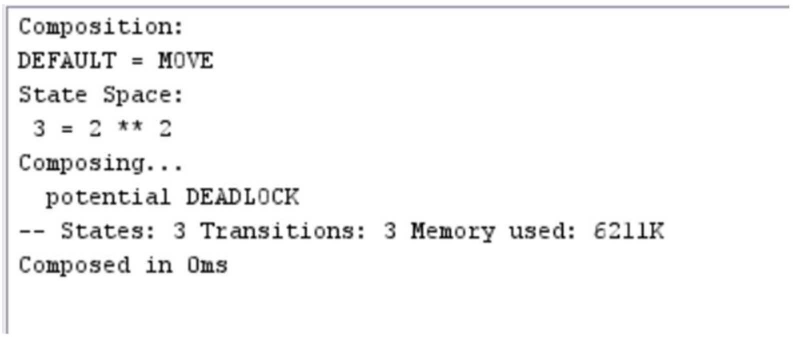
Part 1 – Deadlock

1. An easy example of of a system with a deadlock is:
   MOVE = (up->(down->MOVE|down->STOP)).

   LTSA catches the potential deadlock:



```
Composition:
DEFAULT = MOVE
State Space:
 3 = 2 ** 2
Composing...
  potential DEADLOCK
-- States: 3 Transitions: 3 Memory used: 6211K
Composed in 0ms
```

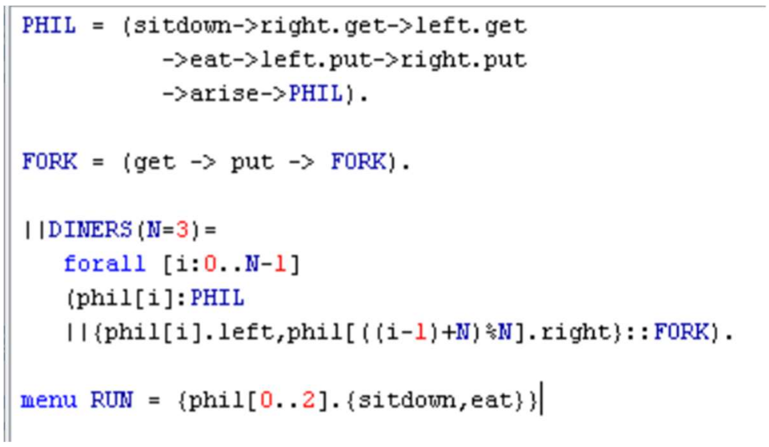
2. The second MOVE state does not have an exit, which is clearly visible in the FSP-model

Part 2 – Livelock

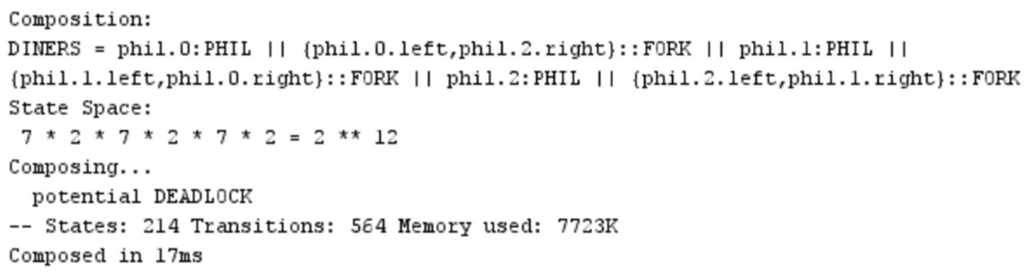
1. A livelock is similar to a deadlock, except that the states of the processes involved in the livelock constantly change with regard to one another, none progressing. Livelock is a special case of resource starvation; the general definition only states that a specific process is not progressing.
   Couldn't create a livelock scenario, but a hostage negotiation is a good example, because the police has the ransom, but is waiting for the hostage to be released, and the criminal has the hostage, but is waiting for the ransom to be paid.

Part 3 – Dining Philosophers

1. 3 Philosophers with 3 forks, system and output

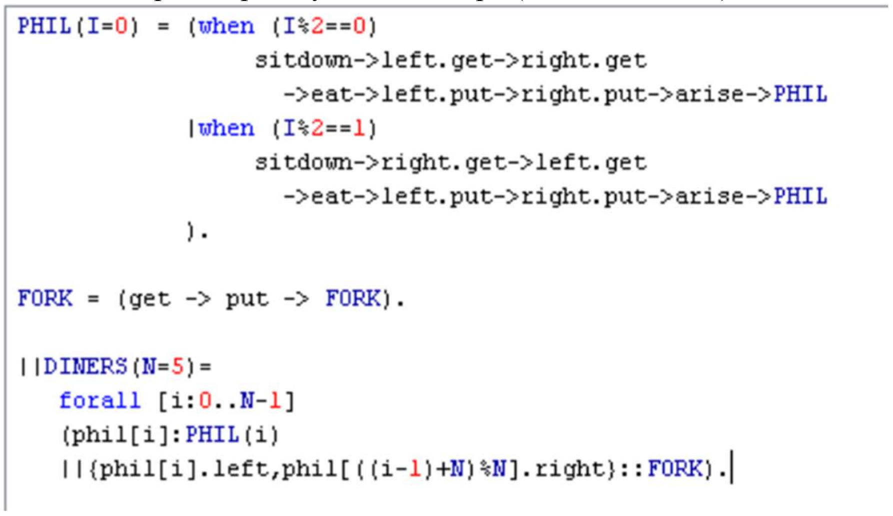


```
PHIL = (sitdown->right.get->left.get
          ->eat->left.put->right.put
          ->arise->PHIL).

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}}
```

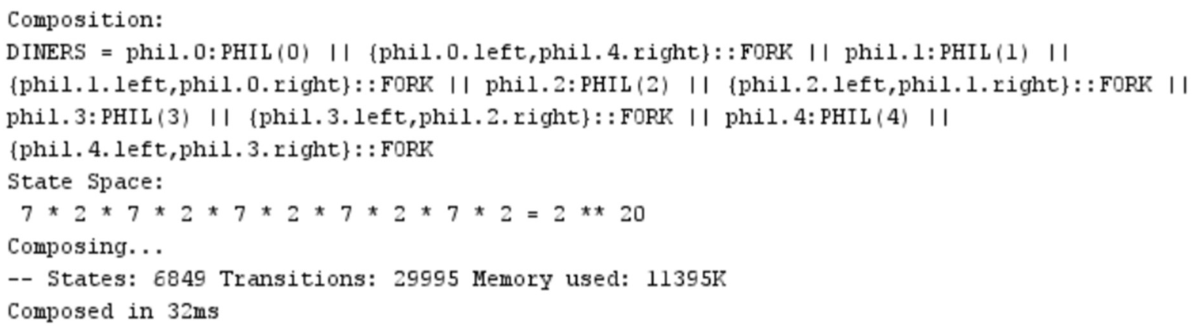


```
Composition:
DINERS = phil.0:PHIL || {phil.0.left,phil.2.right}::FORK || phil.1:PHIL ||
{phil.1.left,phil.0.right}::FORK || phil.2:PHIL || {phil.2.left,phil.1.right}::FORK
State Space:
 7 * 2 * 7 * 2 * 7 * 2 = 2 ** 12
Composing...
  potential DEADLOCK
-- States: 214 Transitions: 564 Memory used: 7723K
Composed in 17ms
```

2. LTSA can handle a few hundred philosophers. Changed the N in DINERS above to test.
3. Left-handed philosopher system and output(deadlock free <3):



```
PHIL(I=0) = (when (I%2==0)
                    sitdown->left.get->right.get
                      ->eat->left.put->right.put->arise->PHIL
             |when (I%2==1)
                    sitdown->right.get->left.get
                      ->eat->left.put->right.put->arise->PHIL
             ).

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

||DINERS(N=5)=
   forall [i:0..N-1]
   (phil[i]:PHIL(i)
   ||{phil[i].left,phil[((i-1)+N)%N].right}::FORK).
```



```
Composition:
DINERS = phil.0:PHIL(0) || {phil.0.left,phil.4.right}::FORK || phil.1:PHIL(1) ||
{phil.1.left,phil.0.right}::FORK || phil.2:PHIL(2) || {phil.2.left,phil.1.right}::FORK ||
phil.3:PHIL(3) || {phil.3.left,phil.2.right}::FORK || phil.4:PHIL(4) ||
{phil.4.left,phil.3.right}::FORK
State Space:
 7 * 2 * 7 * 2 * 7 * 2 * 7 * 2 * 7 * 2 = 2 ** 20
Composing...
-- States: 6849 Transitions: 29995 Memory used: 11395K
Composed in 32ms
```

4. By using asymmerty, deadlocks can be avoided. In this case, you could make half of the philosophers pick up the left fork first, while the other half picks up the right first. This is a fair solution in this scenario.