Assigment

Diego Garcia-Olano, Rajagopal Hariharan, Roberto E. Vargas Caballero

October 31, 2013

- 5.3: The LTS definition of the problem is:

Listing 1: Museum LTS definition

```
const N = 5

DIRECTOR = (open.west -> open.east -> close.east ->
    close.west -> DIRECTOR).
EAST = (open.east -> EASTOPEN),
EASTOPEN = (arrive -> EASTOPEN | close.east -> EAST).

WEST = (open.west -> WESTOPEN),
WESTOPEN = (leave -> WESTOPEN | close.west -> WEST).

CONTROL = CLOSED,
CONTROL[i:0..N] = (when (i < 1) close.west -> CLOSED
                  |when (i < N) arrive -> CONTROL[i+1]
                  |when (i > 0) leave -> CONTROL[0]),
CLOSED = (open.west -> open.east -> CONTROL[0]).

||MUSEUM = (EAST || WEST || DIRECTOR || CONTROL).
```

We can see that the only process without free actions is CONTROL, so it is the unique monitor in our system:

Listing 2: Control.java

```
public class Control {
        int count = 0;
        boolean west = false, east = false;
        public final static int MAX = 20;
        public final static int WEST = 0, EAST = 1;

        public synchronized void arrive()
                        throws InterruptedException {
                while (count == MAX || east == false)
                        wait();
                ++count;
                System.out.println("arrive:count = " + count);
                notifyAll();
        }
        public synchronized void leave()
                        throws InterruptedException {
                while (count == 0 || west == false)
                        wait();
                --count;
                System.out.println("leave:count = " + count);
                notifyAll();
        }

        public synchronized void open(int which) {
                if (which == WEST)
                        west = true;
                else if (which == EAST)
                        east = true;
                notifyAll();
        }

        public synchronized void close(int which)
                        throws InterruptedException {
                if (which == WEST) {
                        while (count != 0)
                                wait();
                        System.out.println("Museum closed");
                        west = false;
                } else if (which == EAST) {
                        east = false;
                }
                notifyAll();
        }
}
```

- 5.4: The LTS definition of the problem is:

Listing 3: Dinning LTS definition

```
const N = 5

SAVAGE = (getserving -> SAVAGE)+{fillpot}.
COOK = (fillpot -> COOK)+{getserving}.
POT = POT[0],
POT[i:0..N] = (when (i == 0) fillpot -> POT[N]
              |when (i > 0) getserving -> POT[i - 1]
              ).

||DINING_SAVAGES = ( s0:SAVAGE || s1:SAVAGE || c:COOK ||
    {s0,s1,c}::POT ).
```

POT has not free actions, so it can be implemented with a monitor:

Listing 4: Pot.java

```

public class Pot {
        int count = 0;
        private final static int MAX = 20;
        public synchronized void getserving()
                        throws InterruptedException {
                while (count == 0)
                        wait();
                System.out.println("Serving num: " + count);
                --count;
                notifyAll();
        }

        public synchronized void fillpot()
                        throws InterruptedException {
                while (count != 0)
                        wait();
                count = MAX;
                System.out.println("Fillng " + count);
                notifyAll();
        }
}
```