Chapter 1 - exercises

This is not really meant as an exercise, but as a way for you to get a first contact with the LTSA tool, which you will be using extensively in this course.

1.1 Start the LTSA, and type the following in the Edit window:

```
BOMB = (start -> timeout -> explode -> STOP).
```

This is a simplified model of a bomb. The timer of the bomb is started, and when it expires (action timeout) the bomb explodes. Now from the "Build" menu option, select "Parse". This lets you know if your specification contains syntax errors.

If you have no syntax errors, then select "Check – Compile". This will generate the state machine (Labelled Transition System - LTS) that corresponds to your specification. You want to see what it looks like? Select "Window – Draw". Is this what you expected?

You can also experiment with actually "animating" the model of the bomb. Select "Check - Run – Default". An animator window comes up. On the right, you have the actions that can be performed by the Bomb. The "ticked" ones are those that are eligible at the current state. Select an action that you would like the bomb to perform (by clicking in its corresponding box). Can all actions be selected? What happens when you select an eligible action? Does anything change on the displayed LTS?

1.2 Perform all of the above steps for the following specification of a lamp:

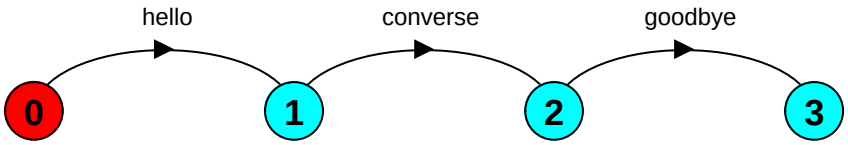
```
LAMP = (switch_on -> switch_off -> LAMP).
```

Can you see an important difference from the first model?

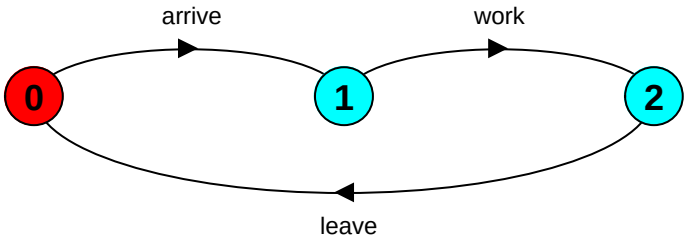
Chapter 2 - exercises

2.1 For each of the following processes, give the Finite State Process (FSP) description of the Labeled Transition System (LTS) graph. The FSP process descriptions may be checked by generating the corresponding state machines using the analysis tool, *LTSA*.

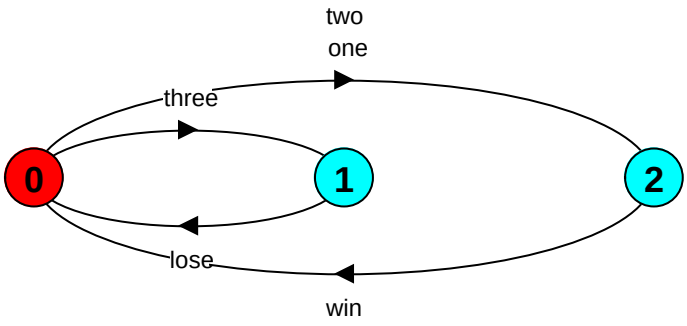
I. MEETING:



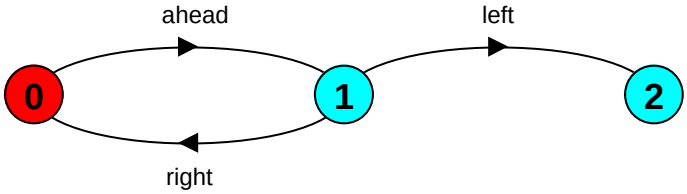
II. JOB:



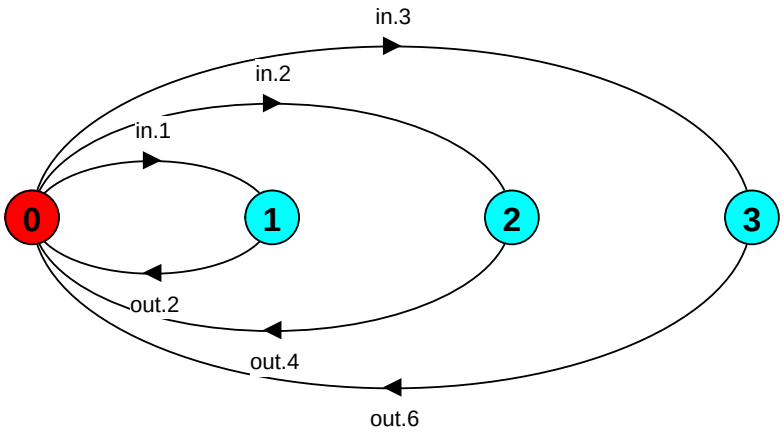
III. GAME:



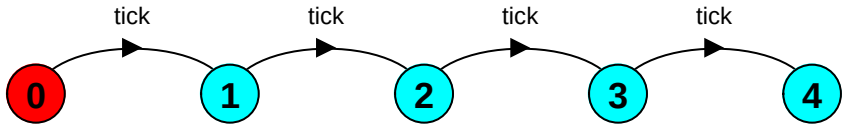
IV. MOVE:



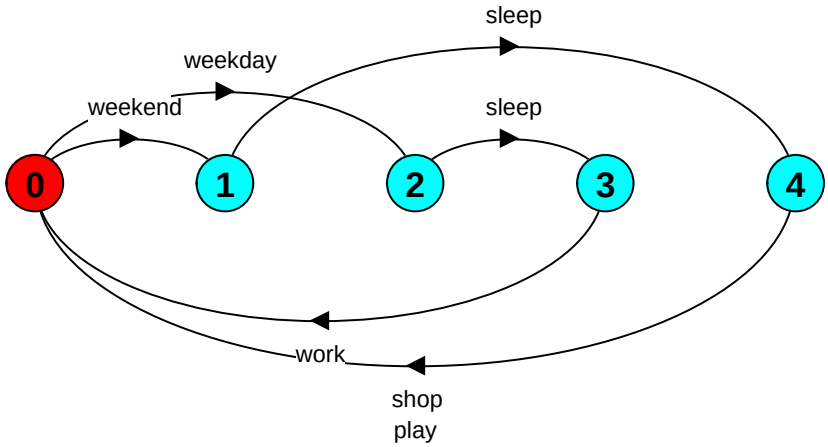
V. DOUBLE



VI. FOURTICK:



VII. PERSON:



For each of the following exercises 2.2 to 2.6, draw the state machine diagram that corresponds to your *FSP* specification and check that it can perform the required actions. The state machines may be drawn manually or generated using the analysis tool, *LTSA*. *LTSA* may also be used to animate (run) the specification to produce a trace.

2.2 A variable stores values in the range 0..*N* and supports the actions *read*

and *write*. Model the variable as a process, `VARIABLE`, using *FSP*.

For N=2, check that it can perform the actions given by the trace:

write.2 → read.2 → read.2 → write.1 → write.0 → read.0

2.3 A bistable digital circuit receives a sequence of *trigger* inputs and alternately outputs 0 and 1. Model the process `BISTABLE` using *FSP*, and check that it produces the required output i.e. it should perform the actions given by the trace:

trigger → 1 → trigger → 0 → trigger → 1 → trigger → 0 …

(Hint: the alphabet of `BISTABLE` is *[0],[1],trigger*).

2.4 A sensor measures the water *level* of a tank. The level (initially 5) is measured in units 0..9. The sensor outputs a *low* signal if the level is less than 2 and a *high* signal if the level is greater than 8 otherwise it outputs *normal*. Model the sensor as an *FSP* process, `SENSOR`.

(Hint: the alphabet of `SENSOR` is *level[0..9], high, low, normal*).

2.5 A drinks dispensing machine charges 15p for a can of Sugarola. The machine accepts coins with denominations 5p, 10p and 20p and gives change. Model the machine as an *FSP* process, `DRINKS`.

2.6 A miniature portable FM radio has three controls. An on/off switch turns the device on and off. Tuning is controlled by two buttons *scan* and *reset* which operate as follows. When the radio is turned on or *reset* is pressed, the radio is tuned to the top frequency of the FM band (108 MHz). When *scan* is pressed, the radio scans towards the bottom of the band (88 MHz). It stops scanning when it *locks* on to a station or it reaches bottom (*end*). If the radio is currently tuned to a station and *scan* is pressed then it starts to scan from the frequency of that station towards bottom. Similarly, when *reset* is pressed the receiver tunes to top. Using the alphabet (*on, off, scan, reset, lock, end*) model the FM radio as an *FSP* process, `RADIO`.

2.7 Program the radio of 2.6 in Java, complete with graphic display.