CPDS-Conc Lab 1
Labelled Transition System Analyzer & Finite State Processes

Authors: Jorge Castro and Joaquim Gabarro

Important: Students should have the `ltsa` running in their laptops.

Goal: Twofold.

- Be familiar with the `Labelled Transition System Analyzer` (`ltsa` for short) and the `Finite State Processes` expressions (`FSP` for short).
- Design basic `FSP` processes and test them in the `ltsa`.

Basic material:

- Slides of the course.
- Basic reading: Chapters 2 and 3 of the book: *Concurrency, State Models & Java Programs*, Jeff Magee and Jeff Kramer, Wiley, Second Edition, 2006 (M&K for short).

1.1 Training Exercises

1. (M&K 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`

   More precisely:

   - With pen-and-paper design the FSP process corresponding to `BISTABLE`.
   - Open the `ltsa`, in `EDIT` write the `BISTABLE` process. Press $\mathcal{C}$ to compile. Correct the compilation errors and restart.
   - When no compilation errors exist press `Draw` to display.

- Use Check→Run→DEFAULT to start the Animator. Use the Animator to check the correctness of the possible traces. If there is any mistake try again.

Parlance issue: We summarize the first two steps as "drawing a LTS process". In this case we ask to "draw the LTS for BISTABLE".

2. (M&K 3.5) A roller-coaster system only permits its car to depart when it is full. Passengers arriving at the departure platform are registered with the roller-coaster controller by a turnstile. The controller signals the car to depart when there are enough passengers on the platform to fill the car to its maximum capacity of M passengers. The car goes around the roller-coaster track and then waits for another M passengers. A maximum of M passengers may occupy the platform. Ignore the synchronisation details concerning passengers embarking from the platform and the car departure. The roller-coaster consists of three processes: TURNSTILE, CONTROL and CAR. Processes TURNSTILE and CONTROL interact by the shared action passenger indicating an arrival and CONTROL and CAR interact by the shared action depart signalling car departure. Provide FSP descriptions for each process and the overall composition.

   Hint: Here, the question is almost longer than the solution:

```
const M = 3
TURNSTILE = (passenger -> TURNSTILE).
CAR = (depart -> CAR).

CONTROL = CONTROL[0],
CONTROL[i:0..M] =
  (when (i<M) ... -> CONTROL[i+1]|when (...) ... -> CONTROL[..]).

||ROLLERCOASTER = (TURNSTILE || CONTROL || CAR).
```

1.2 Homework

1.2.1 Exercises

1. (M&K 3.6) A museum allows visitors to enter through the east entrance and leave through its west exit. Arrivals and departures are signaled to the museum controller by the turnstiles at the entrance and exit. At opening time, the museum director signals the controller that the museum is open and then the controller permits both arrivals and departures. At closing time, the director signals that the museum is closed, at which point only departures are permitted by the controller. Given that it consists of the four processes EAST, WEST, CONTROL and DIRECTOR , provide an FSP description for each of the processes and the overall composition.

```
const N = 5
EAST = (arrive -> EAST).
WEST = (leave  -> WEST).
DIRECTOR = (open -> close -> DIRECTOR).
```

```
CONTROL        = CLOSED[0],
CLOSED[...] = (when (i==0) open -> OPENED[...]
               |when (...) leave -> CLOSED[...]
                 ),
OPENED[i:0..N] = (close -> CLOSED[i]
                 |... arrive -> OPENED[...]
                 |... leave  -> OPENED[...]
                 ).

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

2. (Time-Out Client Server) Consider the following client behavior in a client-server system:

```
CLIENT = (call -> WAIT),
WAIT = (answer -> continue -> CLIENT | timeout -> CLIENT).
```

In words, the client, after calling the server, can give up obtaining an answer from the server by issuing a timeout action. This means that the client gets bored waiting the server answer (the server may be working slowly because it is overloaded), aborts throwing a timeout and tries later. Assume the following naive server behavior:

```
SERVER = (request -> service -> reply -> SERVER).
```

Solve the following questions:

(a) Define the CLIENT_SERVER process by composing CLIENT and SERVER synchronizing actions call/request and answer/reply.

```
||CLIENT_SERVER = (CLIENT || SERVER) /{.../..., .../...}.
```

In order to execute the parallel composition || in ltsa you need to press the || button next to the $\mathcal{C}$ button.

(b) Draw the LTS for CLIENT_SERVER. Do you observe in the diagram something strange pointing out a possible bad behavior in the CLIENT_SERVER system? Write down and explain what you have found.

(c) Provide a new SERVER definition to overcome previous drawbacks.

Hint: Read section 3.1.4 of M&K

1.2.2 How to deliver the homework

Use the *Raco deliver feature* in order to hand in these exercises. *Groups of two people are mandatory.* Names of all participants in the group have to appear at the beginning of the document. Each group has to submit only one file containing solutions for all exercises. More precisely, submit a `Surname-Surname.txt` file, where the surnames of the participants should appear in alphabetical order. Deadline for the submission is shown at the Raco. In order to guide the correction, please fill the following schema code:

```
/* Homework: LTS & FSP
*
*
*
* Name: ...
* Surname: ...
*
* Name: ...
* Surname:  ...
*
*/

/* museum */

const N = 5
EAST = (arrive -> EAST).
WEST = (leave  -> WEST).
DIRECTOR = (open -> close -> DIRECTOR).
CONTROL        = CLOSED[0],
CLOSED[...] = ....
||MUSEUM = (EAST || WEST || DIRECTOR || CONTROL).

/* Time-Out Client server */

/* write your solution and answer the questions */

/* end homework */
```