CS 3SD3

Assignment #2. Due November 9 (Froday), 2018, 23:59 via the course' svn depository. Do not hesitate to discuss with TA or instructor all the problems as soon as you discover them. This assignment is more labour consuming than the first one. Start early!

Instructions: For the solutions that involve FSPs you should provide a plain text file with the corresponding LTSA source code (properly commented). The LTSA solution for each one of the questions must be in a separate file. For questions that do not involve using LTSA or producing FSPs (like questions 5b, 7, 8), hand drawn pictures are allowed, but a solution should be in PDF format (you may do a scan or photo). You must upload your solutions in the course' svn repository. *Any problem with svn repository please discuss with Duy Vu <vud1@mcmaster.ca>, a TA for this course.*

For the implementation questions you should provide a Java program together with the corresponding instructions on how to compile it and run some test cases; good programming style will also be marked.

Furthermore, for each one of the questions involving Java, you must also submit either a plain text file or PDF file explaining in what way your Java program implements your model. Use properly structure diagrams for your FSP processes and class diagrams for your Java programs in your explanation.

1. The dining savages: A tribe of savages eats communal dinners from a large pot capable of holding *M* servings of stewed missionaries. When a savage wants to eat, he helps himself from the pot, unless it is empty, in which case he waits until the cook refills the pot. If the pot is empty, the cook refill the pot with *M* servings.
   The behavior of the savages and the cook is described by:

```
SAVAGE = ( get_serving -> SAVAGE ) .
COOK = ( fill_pot -> COOK ) .
```

   (a) Model the behavior of the pot and of the system as FSP processes.
   (b) Model the behaviour of the pot as and of the system as an Elementary Petri net (see Lecture Notes 3).
   (c) Model the behaviour of the pot as and of the system as an Place/Transition Petri net (see Lecture Notes 9, pages 21, 22).
   (d) Model the behaviour of the pot as and of the system as an Coloured Petri net (see Lecture Notes 9, pages 23-34).
   (e) Discuss the differences between the FSP and the various Petri Net solutions.
   (f) Implement the system as a Java program.

2. A museum allows visitors to enter through the east entrance and leave through its west exit. Arrivals and departures are signalled 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 departuresare permitted by the controller.

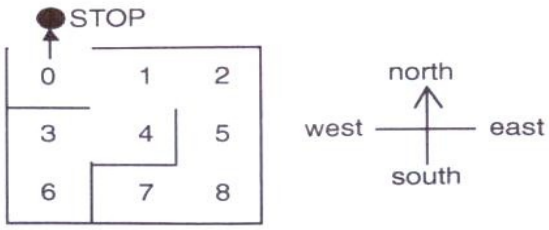
   For Process Algebra models (as FSP), the museum system consists of processes EAST, WEST, CONTROL and DIRECTOR.

   (a) Draw the structure diagram for the museum and provide an FSP description for each of the processes and the overall composition.
   (b) Model the above scenario with Petri nets (any kind, your choice)
   (c) Provide Java classes which implement each one of the above FSP processes.

3. A roller-coaster control 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. Ignore the synchronization detail of passengers embarking from the platform and car departure. The roller-coaster consists of three processes: *TURNSTILE*, *CONTROL* and *CAR*. *TURNSTILE* and *CONTRO*L interact by the shared action *passenger* indicating an arrival and *CONTROL* and *CAR* interact by the shared action *depart* signalling the car departure.

   (a) Provide FSP description for each process and the overall composition.
   (b) Model the above system with Petri nets (any kind, your choice)

4. The figure below depict a maze.



   Write a description of the maze in FSP, which using deadlock analysis provided by LTSA, finds the shortest path out of the maze starting at any square.
   (*Hint*) At each numbered square in the maze, a directional action can be used to indicate an allowed path to another square.

5. Consider the formulation of Smokers' Problem in plain English given in Lecture Notes 10, pages 5-7. The formulation of Dining Philosophers in the same style is in Lecture Notes 9 on page 7. A straightforward FSP model of Dining Philosophers is presented in Lecture Notes 9 on page 9 ('Hungry Simple Minded Philosophers')

(a) Provide a straightforward FSP model of Smokers similar to that of 'Hungry Simple Minded Philosophers'. In principle add supplier to the processes described on page 6 and represent the system using FSP. Use both the compact FSP notation (as upper part of page 9 of LN 9, above the horizontal line) and it expanded version (as lower part of page 9 of LN 9, below the horizontal line). The smoker with for example tobacco could be modelled by the process (but other solutions are also possible):

```
SMOKER_T=( get_paper -> get_match->roll_cigarrette ->
smoke_cigarrette ->SMOKER_T)
```

The resource 'tobacco' could be modelled for example by the process:

```
TOBACCO = ( delivered -> picked -> TOBACCO)
```

etc. If your solution deadlock, provide the shortest trace that lead to the deadlock, if not, provide some arguments why not.

(b) Write (safety) *property* process (syntax `property CORRECT_PICKUP = ...`) that verifies correct sequences of picking resources, i.e. picking up the paper by the smoker with tobacco must be followed by picking up match by the same smoker, picking up the tobacco by the process with paper must be followed by picking up match by the same smoker, and picking up tobacco by the smoker with matches must be followed by picking the paper by the same smoker.

Then compose `CORRECT_PICKUP` with your solution to (a) above and use the system provided by the textbook to verify if this safety property is violated.

(c) An elegant deadlock free solution to the Smokers can be constructed by applying 'ask first, do later' paradigm. Assume that the supplier informs smokers *explicitly* about the ingredient that is *not* supplied, for example it supplies paper, matches and a sign 'no tobacco'. Each smoker reads the sign first and then start picking ingredients only if he has the ingredient that is mentioned on the sign.
Provide this solution using FSPs.

(d) Compose your solution from (c) with the property `CORRECT_PICKUP` from (b) and use the system provided by the textbook to verify if this safety property is *not* violated.

6. Pages 16-18 of Lecture Notes 9, provide a solution to the Dining Philosophers problem with Elementary Petri Nets assuming that simultaneous picking forks (i.e. resources) is allowed, while pages 23-24 provide a solution to the same problem using Coloured Petri Nets, also assuming simultaneous picking forks (i.e. resources is allowed).

   (a) Provide a solution to Smokers problem with Elementary Petri Nets and assuming that simultaneous picking (and delivery) resources is allowed (just mimic the solution for Dining Philosophers).

   (b) Provide a solution to Smokers problem with Coloured Petri Nets and assuming that simultaneous picking (and delivery) resources is allowed (just mimic the solution for Dining Philosophers).