

## CS 2SD3

**Assignment #2. Due March 13 (Monday), 2023, 23:59 via Avenue. Do not hesitate to discuss with TA or instructor all the problems as soon as you discover them. This assignment labour consuming. Start early!**

**Total: 122 pts**

**Instructions: For all assignments, the students must submit their solution to Avenue → Assessments → Assignment #**

**Students can simply solve the exercises on a paper and use a smartphone app called [Microsoft Lens - PDF Scanner](#) and convert their entire solution into a single PDF file and submit it to avenue. The maximum upload file size is 2Gb in avenue for each submission. Please also attach your LTSA and JAVA files separately.**

**Please make sure that the final PDF file is readable.**

**Students, who wish to use Microsoft word and do not have Microsoft Word on their computer, are suggested to use google document editor ([Google Docs](#)). This online software allows you to convert your final file into PDF file.**

**There will be a mark deduction for not following the submission instruction.**

**Please first finish the assignment on your local computer and at the end, and attach your solution as a PDF file.**

**You will have unlimited number of submissions until the deadline.**


## Submission recommendations for students.


❖ You must submit Java files for java question.


❖ You must submit LTSA file for your LTSA questions.

❖ For hand drawn diagrams, you can attach a photo, we recommend JPG format or insert your solution inside a pdf file.

❖ We must be able to copy and past your code into LTSA tool.

 Final PDF file.pdf

 Q#.Java

 Q#.Its

**Students must submit their assignments to [Avenue](#). Any problem with Avenue, please discuss with Mahdee Jodayree <[mahdijaf@yahoo.com](mailto:mahdijaf@yahoo.com)>, the lead TA for this course.**

1.[40] Consider a group of  $k$  philosophers. Each philosopher either think or eats cookies (one serving at a time) or drinks cola (one bottle at a time). The cookie dispenser has a capacity of  $M$  servings, and cola dispenser has a capacity of  $N$  bottles. When a dispenser is empty a philosopher waits until a servant refills it. If any dispenser is empty, the servant refills it with either  $M$  servings of cookies or  $N$  bottles of cola, whichever the case. Refilling must be done as soon as possible, so they have priority over dispensing cookies and cola. There is no partial refilling of the dispensers. Assume that initially, both dispensers are full.

(a)[10] Model the behaviour of the system as FSP processes. Write a safety property that when composed with your system will check **if no cookies are dispensed when there is no cookies and no cola is dispensed when there is no cola**. Compose this property with your system and verify if this it holds.

There is no standard model of priorities in Petri nets, so you have the freedom to define them to fit your solution. This is a comment for points (b), (c) and (d).

(b)[5] Model the behaviour of the system as an Elementary Petri net (see Lecture Notes 3).

(c)[5] Model the behaviour of the system as a Place/Transition Petri net (see Lecture Notes 9, pages 21, 22).

(d)[5] Model the behaviour of the system as a Coloured Petri net (see Lecture Notes 9, pages 23-34).

(e)[5] Discuss the differences between the FSP and various Petri Net solutions.

(f)[10] Implement the system in Java program.

2.[17] A museum allows visitors to enter through the east entrance and leave through its west exit. Arrivals and departures are signalled to the museum controlled 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.

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

(a)[7] Draw the structure diagram for the museum and provide an FSP description for each of the processes and the overall composition.

(b)[5] Model the above scenario with Petri nets (any kind, your choice)

(c)[5] Provide Java classes that implement each one of the above FSP processes.

3.[10] Specify a *safety property* for the car park problem of Lecture Notes 7 or Chapter 5 of the textbook, which asserts that the car park does not overflow.  
 Also, specify a *progress property* which asserts that cars eventually enter the car park.  
 If car departure is *lower priority* than car arrival, does starvation occur?

4.[15] For ‘The Dining Philosophers Problem’, simultaneous picking up of both forks is an abstraction of a general rule that ‘the act of picking up both forks is atomic’. In other words, an order ‘pick right’, ‘pick left’ is arbitrary but once the process of picking starts, it cannot be interrupted. In practice quite often *conceptual simultaneity* is implemented as *atomicity*.

Provide a solution with FSP for Dining Philosophers with ‘atomic act of (sequential) picking up both forks’.

5.[5] Provide a Petri Nets solution (any kind of nets can be used) to asymmetric Dining Philosophers discussed in Lecture Notes 9 page 13 (or Chapter 6.2.2 of the textbook).

6.[5] A lift has a maximum capacity of ten people. In the model of the lift control system, passengers entering a lift are signalled by an enter action and passengers leaving the lift are signalled by an exit action. Specify a *safety property* in FSP which when composed with the lift will check that the system never allows the lift that it controls to have more than ten occupants.

7.[5] Consider the safety property:

$$\text{property } P = (a \rightarrow (b \rightarrow P \mid a \rightarrow P) \mid b \rightarrow a \rightarrow P)$$

Provide LTS generated by the property  $P$  and a standard process  $SP$  such that  $LTS(P)=LTS(SP)$

8.[25] Consider the formulation of Smokers' A 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)[10] Provide a straightforward FSP model of Smokers similar to that of 'Hungry Simple Minded Philosophers'. In principle add a 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 its 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 leads to the deadlock, if not, provide some arguments why not.

(b)5] 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 the 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)[5] An elegant deadlock-free solution to the Smokers can be constructed by applying 'ask first, do later' paradigm. Assume that the smokers are not so hungry for smoking and look on the table first, before making any movement. Then each smoker starts picking ingredients only if he has the ingredient that he does not see on the table. Otherwise, he waits patiently. Provide this solution using FSPs.

(d)[5] 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.

