COMP SCI 3SD3 *Virtual Take Home Examination*

Term: December 2021 (December 18, 2021, 12:30-3:30)
Instructor: Dr. R. Janicki
Duration: 2.5h (Exam Time Only)
McMaster University Final Examination **Total: 132 pts**

- THIS EXAMINATION PAPER HAS **5** PAGES AND **9** QUESTIONS.
- The exam starts at 12:30 pm and ends at 3:30 pm, i.e. 180 (3hr) minutes (for students without extra time permissions). This includes the exam time (150 minutes) plus extra time for technology (30 minutes).
- Please submit the solutions via Avenue using the same procedure as for the assignments.
- Also, just in case, send the solutions via e-mail to janicki@mcmaster.ca .
- Any question during this midterm, ask by sending an e-mail to Ryszard Janicki (janicki@mcmaster.ca)

1.[10] Consider the following scenario: An office is used by several workers sharing a single printer. As a simplification of a real life scenario, it is assumed that the printer is able to print any number of jobs, independently of the number of pages those jobs have, before running out of toner. The toner is replaced by a technician whenever necessary.

a.[5] Provide an FSP description of the above scenario. You must provide also a brief description of the intended behavior for each one of the processes you define.
Hint: Possible processes for the above scenario are PRINTER, USER and TECHNICIAN.

b.[5] Provide a Petri nets (any kind) description of the above scenario.

2.[15] Two warring neighbours are separated by a field with wild berries. They agree to permit each other to enter the field to pick berries, but also need to ensure that only one of them is ever in the field at a time. After negotiation, they agree to the following protocol. When a one neighbour wants to enter the field, he raises a flag. If he sees his neighbour's flag, he does not enter but lowers his flag and tries again. If he does not see his neighbour's flag, he enters the field and picks berries. He lowers his flag after leaving the field.

a.[8] Model this algorithm for two neighbours `n1` and `n2`. Specified the required *safety* properties for the field and check that it does indeed ensure mutually exclusive access. Specify the required *progress* properties for the neighbours such that they both get to pick berries given a fair scheduling strategy. Are any adverse circumstances in which neighbours would not make progress? What if the neighbours are greedy?

Hint: The following *FSP* can be used to model the flags.

```
const True = 1 const False = 0
range Bool = False..True
set BoolActions={setTrue,setFalse,[False],[True]}
BOOLVAR = VAL[False],
VAL[v:Bool] = ( setTrue -> VAL[True]
          | setFalse -> VAL[False]
          | [v] -> VAL[v] ).
```

b.[7] Model this algorithm for two neighbours using Petri nets (any kind).

3.[30] A cook puts burgers in a pot. A client checks if there is at least one burger in the pot, and if so, the client must take one.

(a)[20] Assuming two clients, this can be modelled as

```
range Burgers = 0..2
CLIENT = ( check -> get -> CLIENT ).

POT = POT[0],
POT[p: Burgers] = ( when p > 0 check -> POT[p]
     | get -> POT[p-1]
     | fill[n: Burgers] -> POT[n] ).

COOK = ( fill[p: 1..2] -> COOK )+{fill[0]}.
||DS = ( c1: CLIENT || c2: CLIENT || POT || COOK )
/{ {c1, c2}.check/check, {c1, c2}.get/get }.
```

(i)[12] Is there a trace belonging to process DS leading to an error state? If so, give the trace.

(ii)[8] We wish a client to obtain exclusive access to the pot. That is, two clients cannot check the pot at the same time, and when a client checks, he/she must take a burger (if there is at least one).
Show how to modify the given model such that this behaviour is ensured.

(b)[10] Provide a solution with any kind of Petri nets. Note that the cook cannot put a burger in a full pot.

4.[20] 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)[10] Model the behavior of the pot and of the system as FSP processes.
(b)[10] Model the behaviour of the pot with Petri Nets (any kind, your choice)

5.[10] In an operating system, a binary semaphore is used to control access to the console. The console is used by user and system processes. Write down a model with FSP for this system. Discuss when user processes may be denied access to the console.

6.[9] Let

```
property Alpha = (a->b->Alpha | b->Beta)+{d}
         Beta  = (c->b->Alpha | b->Beta)
```

and

```
Alpha = (a->b->Alpha | b->Beta)+{d}
Beta  = (c->b->Alpha | b->Beta)
```

Let $LTS_{propertyAlpha}$ be the Labelled Transition System of the process `property Alpha`, and LTS_{Alpha} be the Labelled Transition System of the process `Alpha`.

a.[3] Produce $LTS_{propertyAlpha}$.

b.[3] Produce LTS_{Alpha}.

c.[3] Write a process `Alpha1` (but not property process) such that
$LTS_{propertyAlpha} = LTS_{Alpha1}$.

7.[10] *Simplified Multidimensional Semaphores* are defined as follows:

- The extended primitives *edown* and *eup* are atomic (indivisible) and each operates on a set of semaphore variables which must be initiated with non-negative integer value.

- *edown*(S_1,...,S_n):
  if for all i, $1 \le i \le n$, $S_i > 0$ then for all i, $1 \le i \le n$, $S_i := S_i - 1$
  else block execution of calling processes

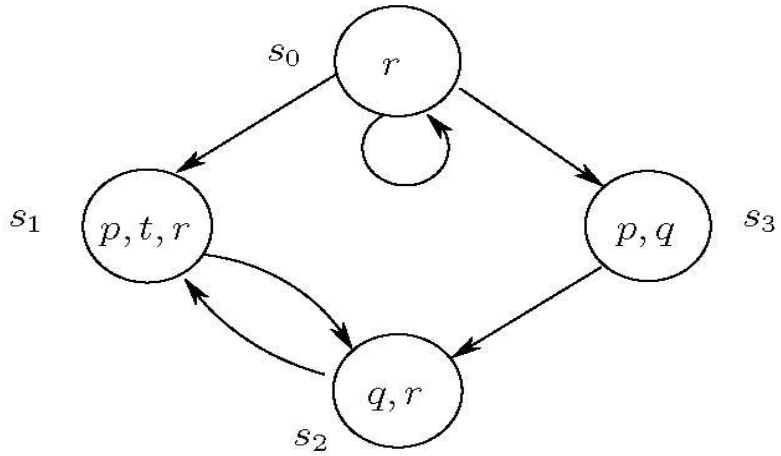
- *eup*(S_1,...,S_n):
  if processes blocked on (S_1,...,S_n) then awaken one of them
  else for all i, $1 \le i \le n$, $S_i := S_i + 1$

Model the *Simplified Multidimensional Semaphores* by FSPs for n=2 and maximal value of S_1, S_2 equal to 3.

Hint. *edown*(S_1,S_2) could be interpreted as just a synchronized execution of *down*(S_1) and *down*(S_2). Similarly for *eup*(S_1,S_2

8.[18] This question deals with Model Checking.

(a)[6] Consider the system *M* defined below:



Determine whether $M, s_0 \models \varphi$ and $M, s_2 \models \varphi$ hold and justify your answer, where φ is the LTL or CTL formula:

(i)[3] EG *r*

(ii)[3] G(r ∨ q)

(b)[3] Express in LTL:

*If the process is **enabled** infinitely often, then it **runs** infinitely often.*

(c)[3] Express in CTL:

*If the process is **enabled** infinitely often, then it **runs** infinitely often.*

(d)[6] Express in LTL and CTL:

A passenger entering the elevator at 5^{th} floor and pushing 2^{nd} floor button, will never reach 6^{th} floor, unless 6^{th} floor button is already lighten or somebody will push it, no matter if she/he entered an upwards or upward travelling elevator.

In this case you might use the following atomic predicates: *floor=2, direction=up, direction=down, ButtonPres2, floor=6, etc.*

9.[10] 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. 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.[5] 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.[5] 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).

END OF EXAM