Concurrency, Parallelism and Distributed Systems (CPDS)
Module I: Concurrency
Facultat d'Informàtica de Barcelona
Final On-Line Exam
June 8, 2020

Answer the questions concisely and precisely.
Duration: 2 hour
A solution of the Exam will be publish on the Racó on June 10, Wednesday
A preliminary grading will be publish on the Racó on June 15, Monday

Exercise 1 (3 Points) FSP & LTS
Let us consider a small vending system.

1. (1 Point) Design the `TWO_DRINKS` process working as follows:
   - Initailly the machine is empty and needs to be filled (execute the action `fill`) before deliver a drink)
   - The machine can deliver just two drinks before to be filled again (execute action `fill`).
   - The drinks delivered can be `coffee` or `tea`.
   - When the machine is empty, it is refilled again (the action `fill` is executed).
2. (1/2 Point) Define a WORKER just executing the action `fill` forever.
3. (1/2 Point) Define a CLIENT as follows: first it chooses between `coffee` and `tea`, later on it takes a `tea` for sure. Starts again. That is, choose between `coffee` and `tea`, later take `tea`, later choose between `coffee` and `tea`,...
4. (1 Point) Given the system `||HOT_DRINKS = (CLIENT||WORKER||TWO_DRINKS)` give discription of `HOT_DRINKS`, called `STRAIGHT`, just using prefixing and recursion (with no parallel composition).

Exercise 2 (2 Points) Stressed Systems.
Let us give high priority to `coffee` over `tea`.

1. (1 Point) First, given the system

   ```
   ||BETTER_COFFE_HOT_DRINKS = (CLIENT||WORKER||TWO_DRINKS)<<{coffee}.
   ```

   Give a description of it, called `BETTER_COFFE_ONE` without parallel composition. Explain intuitivelly the behaviour of LTS.
2. (1 Point) Given `||LIKE_COFFEE = TWO_DRINKS << {coffee}` consider the system

   ```
   ||OTHER_NEW_HOT_DRINKS = (CLIENT||WORKER||LIKE_COFFEE).
   ```

   Give a descriction of the LTS a `OTHER_NEW_HOT_DRINKS` called `BETTER_COFFE_TWO` without parallel composition describing such a system. Explain shortly the behaviour of `OTHER_NEW_HOT_DRINKS`.

Exercise 3 (1 Points) Safety & Liveness properties.

1. (1/2 Point) Define the safety property `NEVER_TWO_FILL` assuring that never one action `fill` is immediatlly followed by other action `fill`. Explain shortly, how do you test that `HOT_DRINKS` verifies the safety property.
2. (1/2 Point) Define the liveness property that: It is always possible to take a tea. Do you think that process `BETTER_COFFE_HOT_DRINKS` verifies this property? Justify the answer.

Exercise 4 (2 Points) Java
Define JAVA monitor corresponding to `TWO_DRINKS` follow M&K approach.

Exercise 5 (2 Points) Erlang
Remind the `server5` given in the Exam Preparation class:

```
-module(server5).
-export([start/0, rpc/2]).

start() -> spawn(fun() -> wait() end).

wait() ->
   receive
     {become, F} -> F()
   end.

rpc(Pid, Q) ->
    Pid ! {self(), Q},
    receive
        {Pid, Reply} -> Reply
    end.
```

started with

```
Pid=server5:start().
```

Suppose that `server5` is currently running as a factorial server. Imagine that you need it to become a quicksort server.

- (1 Point) Design a module `my_quicksort_server` to do the job.
- (1/2 Point) Complete the following instruction in order to update the server.

  ```
  Pid!{... , ...}
  ```

- (1/2 Point) Write the instruction (or instructions) needed to ask the `server5` to sort the $L = [5.0, 1.0, 10.0, 2.0, 7.0, 6.0]$.