Concurrency, Parallelism and Distributed Systems (CPDS)
Module I: Concurrency
Facultat d'Informàtica de Barcelona
Final Exam
November 17, 2017

Answer the questions concisely and precisely
Answer each problem in a separate page (remember to put your name)
Closed-book exam
Duration: 2 hour

Exercise 1 SERVER (5 Points)
We consider a possible extensions of the well known `CLIENT_SERVER`:

```
CLIENT = (call -> answer -> continue -> CLIENT).
SERVER = (request -> service -> reply -> SERVER).
||CLIENT_SERVER = (CLIENT || SERVER)/{call/request, answer/reply}.
```

Let us proceed in small steps. We consider two clients `a` (Alice) and `b` (Bob).

(1 Point) First, explain shortly why the following `MY_SERVER` and `OTHER_SERVER` give the same `LTS`. Second, give a picture of such `LTS`.

```
MY_SERVER =(a.call->CALL | b.call->CALL),
CALL = (service->SERVICE),
SERVICE =(a.answer->MY_SERVER | b.answer->MY_SERVER).

||OTHER_SERVER = SERVER/{{a,b}.call/request, {a,b}.answer/reply}.
```

Now consider the following extension of the server:

```
||CLIENT_SERVER_EXT
   = (a:CLIENT || b: CLIENT || SERVER)/{{a,b}.call/request, {a,b}.answer/reply}.
```

(1/2 Points) Give a picture of the `LTS` corresponding to `a:CLIENT`.

(1/2 Points) In `CLIENT_SERVER_EXT`, is it the following trace is a valid one?

```
b.call
service
b.answer
a.call
b.continue
service
a.answer
```

Give a short explanation.

(2 Points) Define a property process `SAFE` in order to check that in all valid traces of `CLIENT_SERVER_EXT` first answer after a `a.call` action is `a.answer` and first answer after a `b.call` action is `b.answer`. Therefore a trace like

```
a.call
service
a.answer
```

is a valid one because the first answer to a `a.call` is a `a.answer` but

```
a.call
service
b.answer
```

in not a valid trace because the first answer to a `a.call` is a `b.answer`. We ask for

- Give the `FSP` corresponding to the property `SAFE`. Avoid unnecessary guards. The alphabet of `SAFE` has to be $\{a, b\}\{call, answer\}$.
- Give the corresponding `LST`.

(1 Point) Discuss the progress property `BOB_CALL` on the `GREEDY_ALICE` process below

```
||GREEDY_ALICE = CLIENT_SERVER_EXT << {a.call}.

progress BOB_CALL = {bob.call}
```

Exercise 2 ONEBUF (2 Points)
A single-slot buffer may be modelled by:

```
ONEBUF = (put -> get -> ONEBUF).
```

Write a Java class, `OneBuf`, that implements this one-slot buffer as a monitor.

```
public class OneBub {
     Object slot = null;
     ...
}
```

Exercise 3 OneLine (1 Points)
Let `L1` and `L2` two lists with no repeated elements (encoding sets). In Python you have the following easy function to find the intersection:

```
# L1 and L2 have no repeted elements
def intersection(L1,L2):
    L=[]
    for x in L1:
        if x in L2:
            L.append(x)
    return L

if __name__ == "__main__":
    L1=[5, 2, 8]
    L2 = [1, 2, 7, 8]
    print(intersection(L1,L2))
```

Complete in Erlang the following one line encoding of the preceding approach:

```
intersection(L1,L2)->[X || .... ].
```

Use `lists:member(X,L)` to encode `if x in L`.

Exercise 4 ParallelDotProduct (2 Points)
Given to arrays $\mathtt{X} = [\mathtt{x}_1, ..., \mathtt{x}_n]$ and $\mathtt{Y} = [\mathtt{y}_1, ...\mathtt{y}_n]$. Give a parallel implementation of the dot product

$$\mathtt{dot\_product}(\mathtt{X}, \mathtt{Y}) = \mathtt{x}_1 * \mathtt{y}_1 + \cdots + \mathtt{x}_n * \mathtt{y}_n$$