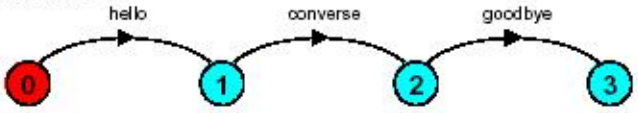
PMR5230 - Sistemas Computacionais para Automação

2a. Lista de Exercícios - Versão 2017

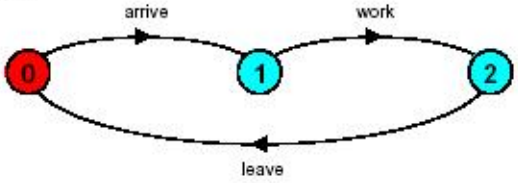
Capítulo 2

[Ex. 1] Para cada um dos seguintes processos descritos através de grafos LTS (Labelled Transition System), descreva-os utilizando FSP (Finite State Process).

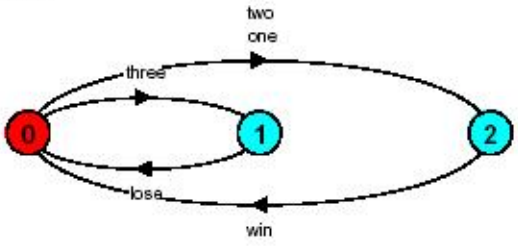
I. MEETING:



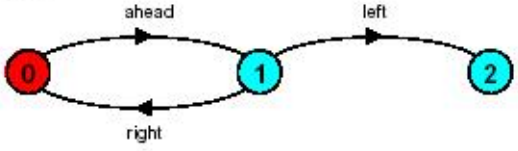
II. JOB:



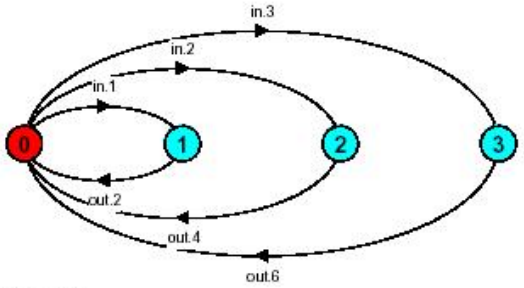
III. GAME:



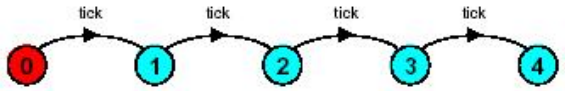
IV. MOVE:



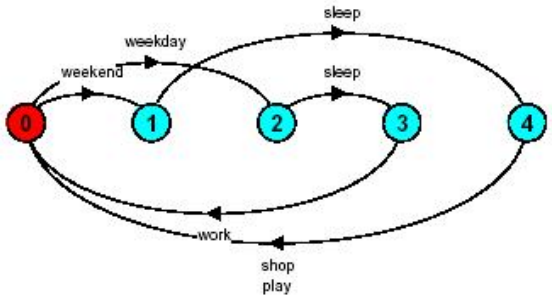
V. DOUBLE



VI. FOURTICK:



VII. PERSON:



[Ex. 2] Um sensor mede o nível de água de um tanque. O nível (inicialmente 5) é medido em unidades discretas 0..9 o sensor coloca como saída um sinal *low* se o nível é menor que 2 e um sinal *high* se o nível é maior que 8, caso contrário ele fornece um sinal *normal*. Modele este sensor como um processo FSP de nome SENSOR. Considere que o alfabeto de SENSOR é `{level[0..9], high, low, normal}`

Capítulo 3

[Ex. 1] Mostre que os processos S1 e S2 são equivalentes.

```
P = (a->b->P).
Q = (c->b->Q).
||S1 = (P||Q).

S2=(a->c->b->S2 | c->a->b->S2).
```

[Ex. 2] A museum allows visitors to enter through the east entrance and leave through its west exit. Arrivals and departures are signaled to the museum controller by the turnstiles at the entrance and exit. At opening time, the museum director signs that the museum is closed, at which point only departures are permitted by the controller. Given that it consists of the four processes EAST, WEST, CONTROL and DIRECTOR. Now provide an FSP description for each of the processes and the overall composition.

Capítulo 4 e 5

[Ex. 1] The dining savages: A tribe of savages eats communal dinners from a large pot that can hold M servings of stewed missionary. When a savage wants to eat, he helps himself from the pot unless it is empty in which case he waits for the pot to be filled. If the pot is empty, the cook refills the pot with M servings. The behavior of the savages and the cook are described by:

```
SAVAGE = (getserving -> SAVAGE).
COOK = (fillpot -> COOK).
```

Model the behavior of the pot as an FSP processes and then skecth a solution in Java using monitors. (OBS: only sketch not detailed).

Capítulo 6

[Ex. 1] Quais são as condições necessárias e suficientes para ocorrência de *deadlock* ?

[Ex. 2] No sistema descrito a seguir é possível a ocorrência de *deadlock*. Explique como isto pode ocorrer, e correlacione com umas das quatro condições necessárias e suficientes para a ocorrência de *deadlock*

```
Alice  = (call.bob -> wait.chris -> Alice).
Bob    = (call.chris -> wait.alice -> Bob).
Chris  = (call.alice -> wait.bob -> Chris).
||S = (Alice || Bob || Chris) / {call/wait}.
```

[Ex. 3] O modelo a seguir é uma tentativa para resolver o problema de *deadlock*. Neste modelo permite-se um tempo de *timeout* de uma tentativa de chamada. O *deadloack* ainda é possível ? Se possível descreva como o deadlock pode ocorrer evidenciando um traço (trace) de execução que leve ao *deadlock*.

```
Alice = (call.bob -> wait.chris -> Alice
         | timeout.alice -> wait.chris -> Alice ).
Bob   = (call.chris -> wait.alice -> Bob
         | timeout.bob -> wait.alice - Bob).
Chris = (call.alice -> wait.bob -> Chris
         | timeout.chris -> wait.bob -> Chris ).
||S = (Alice || Bob || Chis) / {call/wait}.
```

Capítulo 7

[Ex. 1] A lift has a maximum capacity of ten people. In the model of the lift control system, passengers entering the 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.

[Ex. 2] For the following FSP model of the car park problem of Chapter 5:

```
CARPARKCONTROL(N=4) = SPACES[N],
SPACES[i:0..N]= ( when(i>0) arrive->SPACES[i-1] |
                  when(i<N) depart->SPACES[i+1]
                ).
ARRIVALS = (arrive->ARRIVALS).
DEPARTURES = (depart->DEPARTURES).

||CARPARK = (ARRIVALS||CARPARKCONTROL(4)||DEPARTURES).
```

(a) specify and check a safety property OVERFLOW(N=4) which asserts that the car park does not overflow. Now check the carpark against property OVERFLOW(3). What happens in this case ?

(b) specify a progress property, which asserts that cars eventually enter the car park. Which situation is reflected if we make car departure lower priority than car arrival? Do we get starvation as a result?