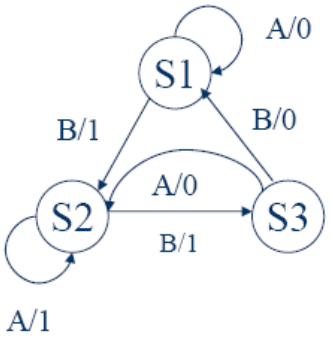
Lecture 10

Exercises:

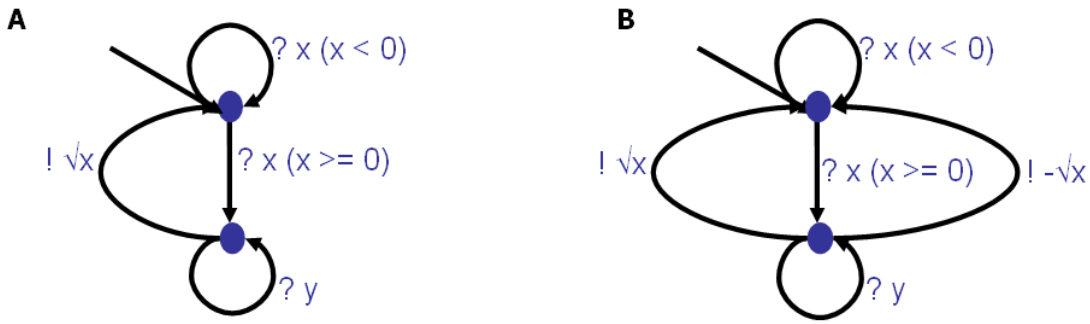
1. A synchronizing sequence (SS) for an FSM state *s* is an input sequence which takes the machine to final state *s* regardless of the output sequence or the initial state.
(1) Proof that not every FSM state has an SS.
(2) Can you suggest some method (or algorithm) to find out synchronizing sequences for a given FSM, say, the following one?

current state	x = 0	x = 1
A	B, 1	C, 0
B	A, 0	D, 1
C	B, 0	A, 0
D	C, 1	A, 1

2. A homing sequence for an FSM is an input sequence such that after it is entered, by observing the output, we know what the final state of the machine is in. Can you suggest some method (or algorithm) to find out a homing sequence for a given FSM, say, the following one?



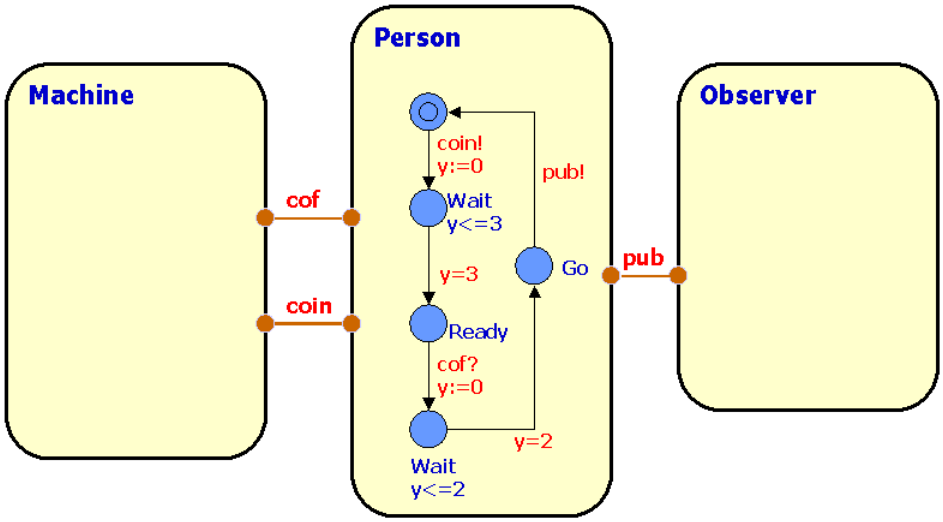
3. In the following figure, why “A **ioco** B” holds whereas “B **ioco** A” does not hold? Suppose B is a specification and A is its implementation. What do you learn from the fact that a partial implementation A is also correct w.r.t. to the **ioco** conformance relation?



4. (modeling exercises: timed coffee machine)
In this exercise you are asked to design the control of a **Machine** (the control program) which will serve a coffee craving **Person** (the environment). As you can see below the person repeatedly (tries to) insert a coin, (tries to) extract coffee after which (s)he will make a publication. Between each action the person requires a suitable time-delay before being ready to participate in the next one.

The machine takes some time for brewing the coffee and will time-out if coffee has not been taken before a certain upper time-limit.

As a requirement we want the overall behaviour to ensure that the indicated **Observer** experiences a constant flow of publications from the system. In particular we want the Observer to complain if at any time more than 8 time-units elapses between two consecutive publications. Model the **Machine** and **Observer** in UPPAAL and analyze the behaviour of the system. Try to determine the maximum brewing time allowed by the Machine in order that the above requirement is met.


Person
Machine
Observer
coin!
y:=0
Wait
y<=3
y=3
Ready
cof?
y:=0
Wait
y<=2
y=2
pub!
Go
cof
coin
pub