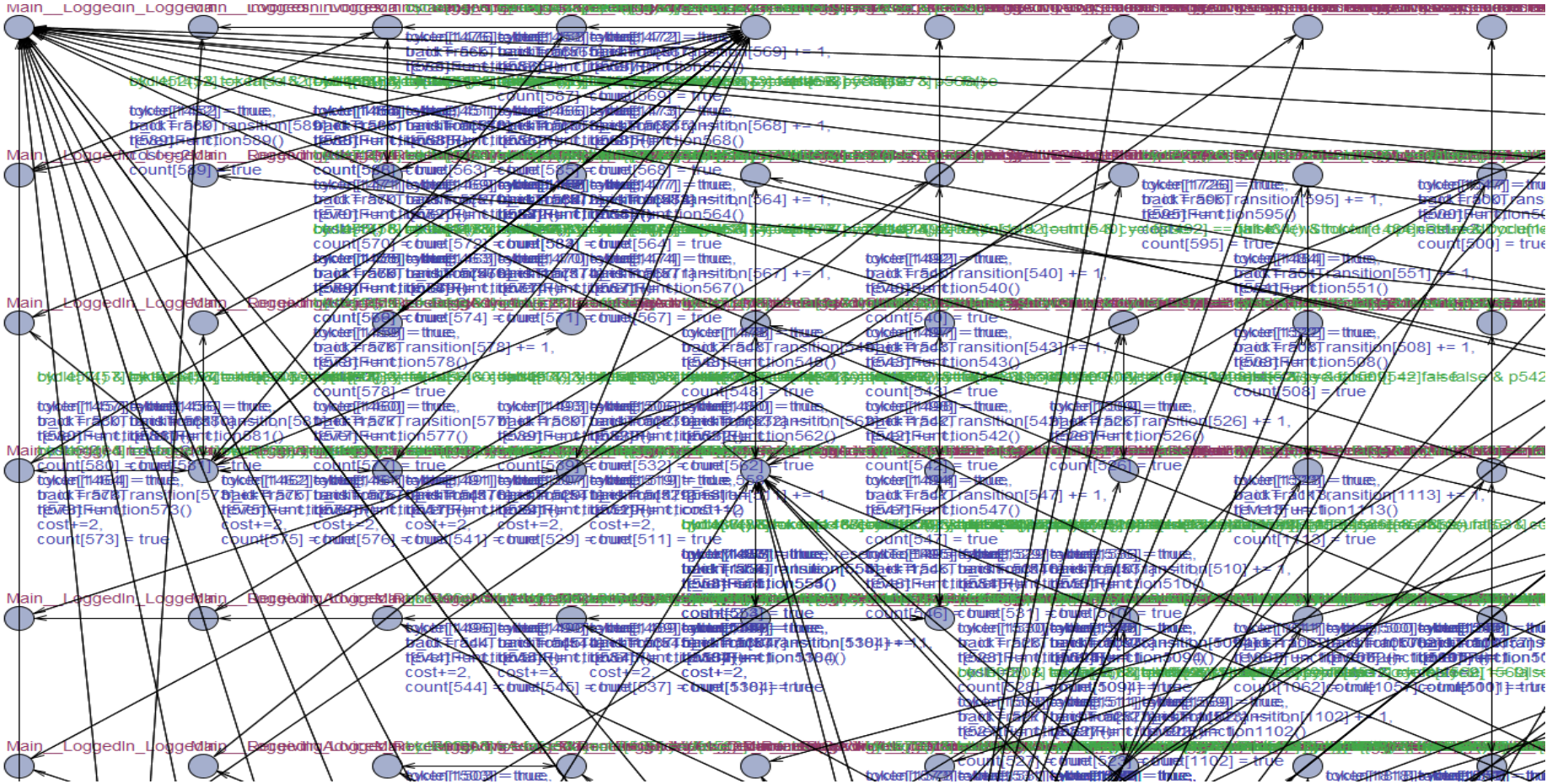


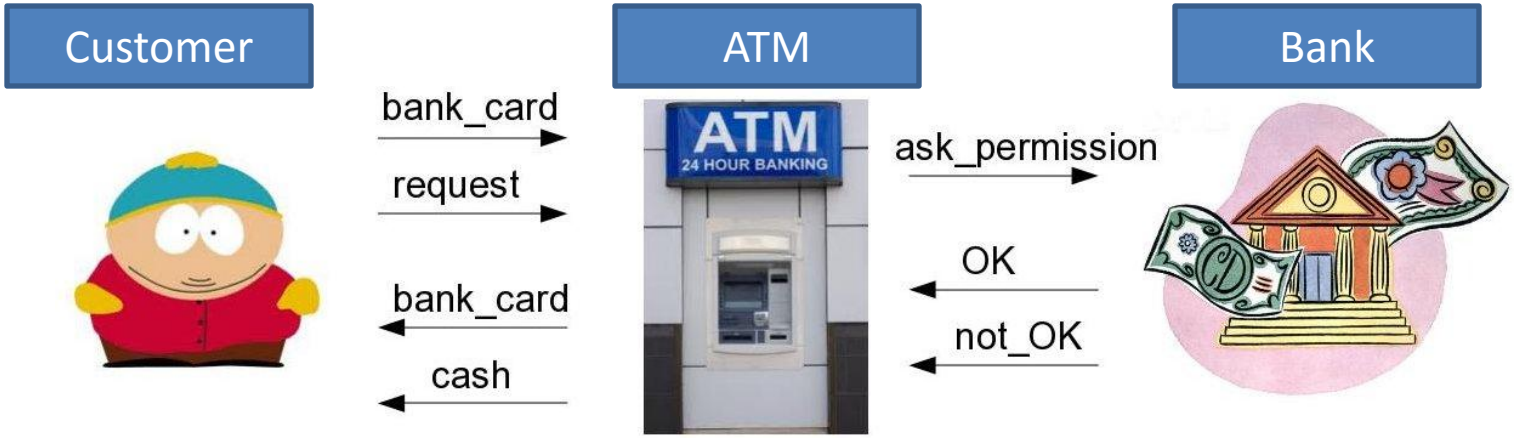
Uppaal Lab Assignments

Deepak Pal

3-March-2018

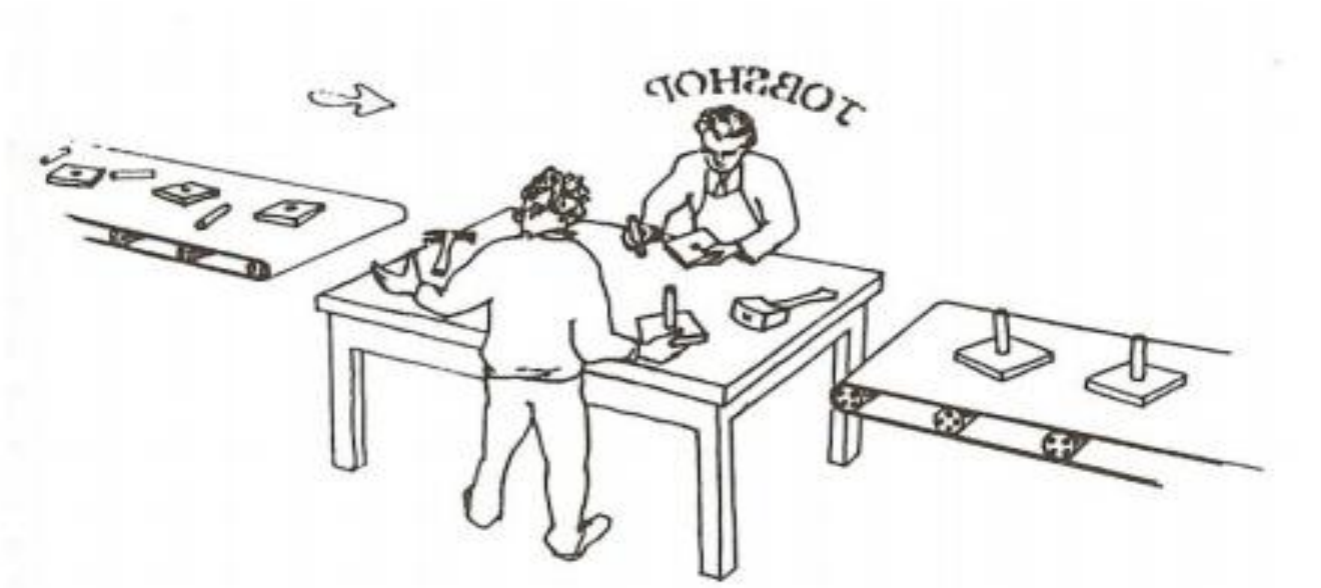
Assignment 1:

- **Model the ATM behavior and try to verify property below:**



- Property: The customer always owns total 100 euro in his Wallet and balance in Bank. Assume Customer has 0 Euros (intially) in his wallet and 100 in bank balance.
  - A[] (Customer.*Initial_Location* and Bank.*Initial_Location*) imply Customer.*cash* + Bank.*cash* == 100
  - *Initial_Location*: is initial location name in your model and
  - *cash* is a local variable name in your model.

Assignment 2

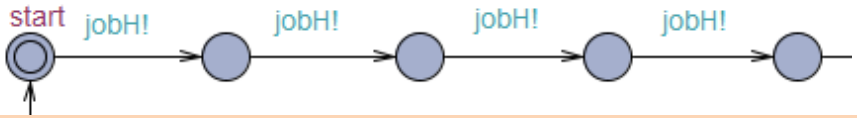


Job shop

We suppose that two people are sharing the use of two tools — a **hammer** and a **mallet** — to manufacture objects from simple components.

Each object is made by driving a peg into a block. We call a pair consisting of a peg and a block a job; the jobs arrive sequentially on a conveyor belt, and completed objects depart on a conveyor belt.

To make the example more specific, we shall assume that the nature of the job influences the jobber's actions in a particular way. We suppose that he may use three predicates **easy, average and hard over jobs,** to determine whether a job is **easy or hard or average**. He will do easy jobs with his hands, hard jobs with both the hammer, and mallet and average jobs with either hammer or mallet.



Consider this sequence of jobs on belt for Tasks

Job shop

A[] not deadlock : Use this property to check deadlock in your model

Task 1: When Hard Job require both Mallet and Hammer with order constraint i.e. first Hammer is require and then Mallet to finish the job.

get_Hammer --> put_Hammer --> get_Mallet --> put_Mallet !

Try to finish all the jobs without deadlock in the model.

See solution "Worker_Case1" in Lab 3 material.

Task 2: When Hard Job require both Mallet and Hammer without any order constraint i.e. any free tool can be used to finish the job.

get_Hammer --> put_Hammer --> get_Mallet --> put_Mallet or

get_Mallet --> put_Mallet --> get_Hammer --> put_Hammer !

Try to finish all the jobs without deadlock in the model.

See solution "Worker_Case2" in Lab 3 material

Job shop

A[] not deadlock : Use this property to check deadlock in your model

Task 3: When Hard Job require both Mallet and Hammer at the same time i.e. both tools must be free to finish the job, order constraint (get hammer and mallet) is very important here.

get_Hammer --> get_Mallet --> put_Hammer --> put_Mallet or

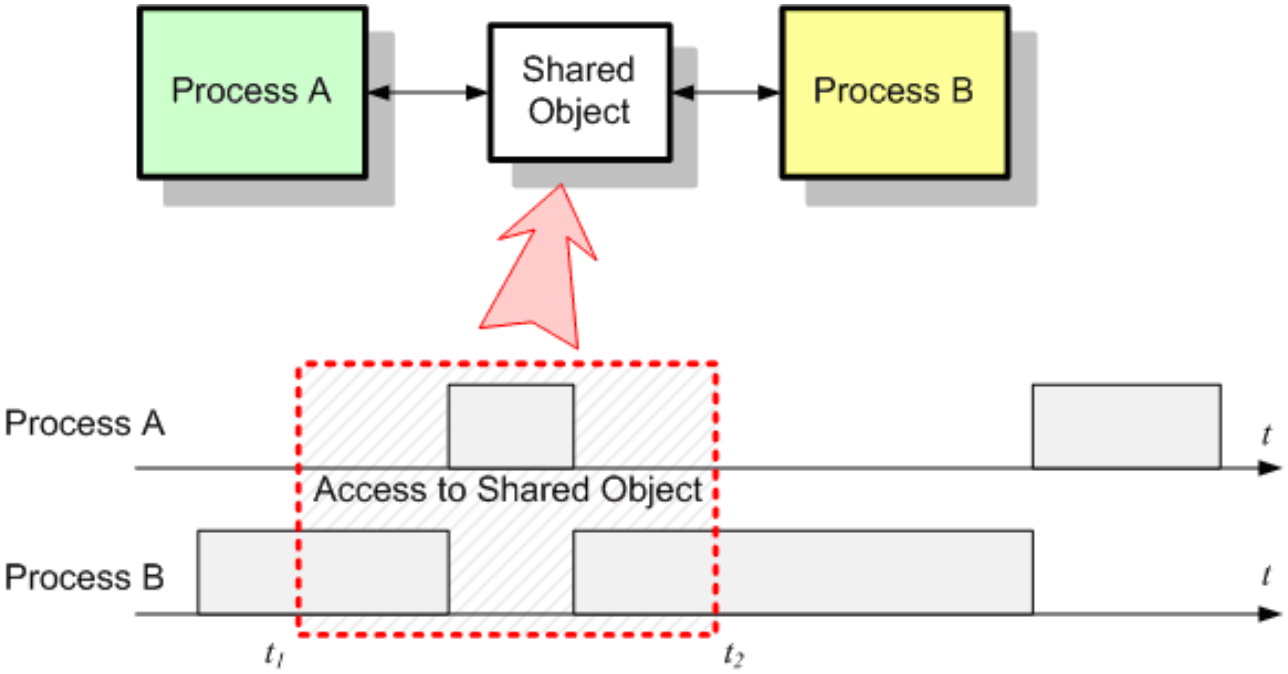
get_Mallet --> get_Hammer --> put_Mallet --> put_Hammer !

Try to create a **deadlock** in the model, where one worker is waiting for other worker to release the tools.

See solution "Worker_Case3" in Lab 3 material

Task 4: Resolve the deadlock in **Task 3** by using mutual exclusion algorithm concepts.

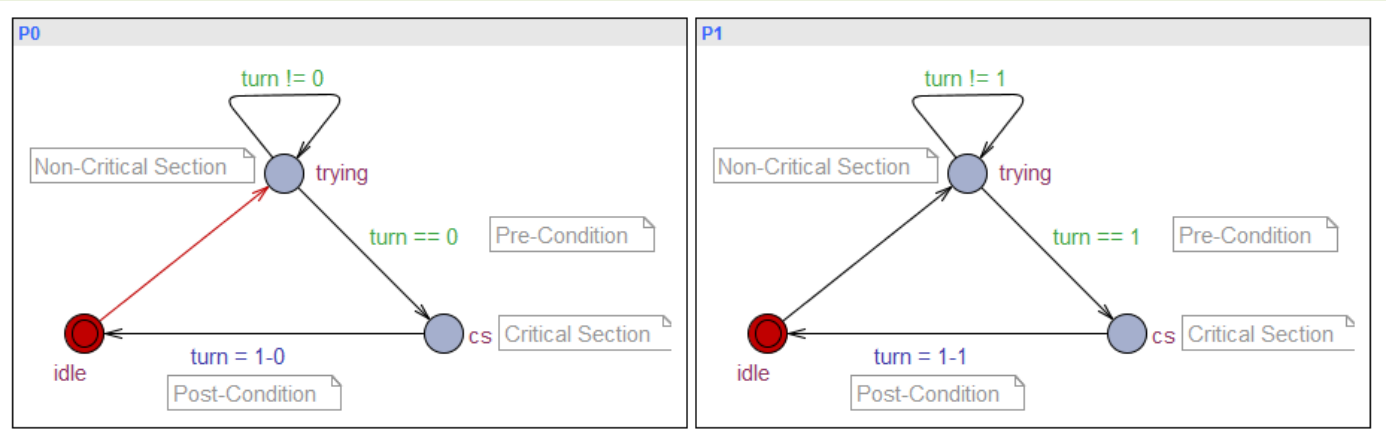
Assignment 3



The Mutual Exclusion Problem

Implementation of Mutual Exclusion Algorithm

- First Attempt (see model below)



- **Assignment:** model other algorithms similarly as First Attempt Algo done.
  - Second Attempt
  - Third Attempt
  - Fourth Attempt
  - Dekker's Algorithm

All Algorithms description is available in lab lecture slides !!

Discussion and Defense

- Defend all the assignment in Lab (22-March-2018).