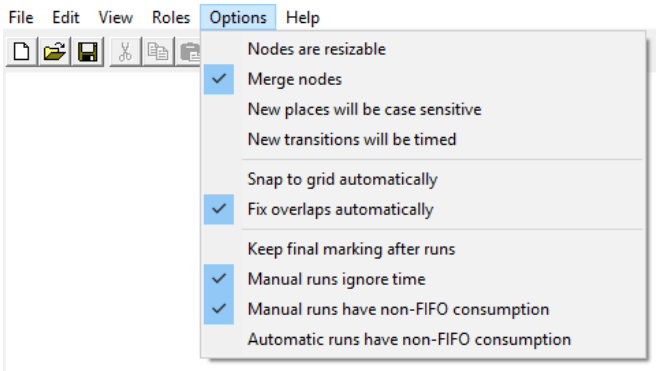
Component-based Systems Modeling and Analysis

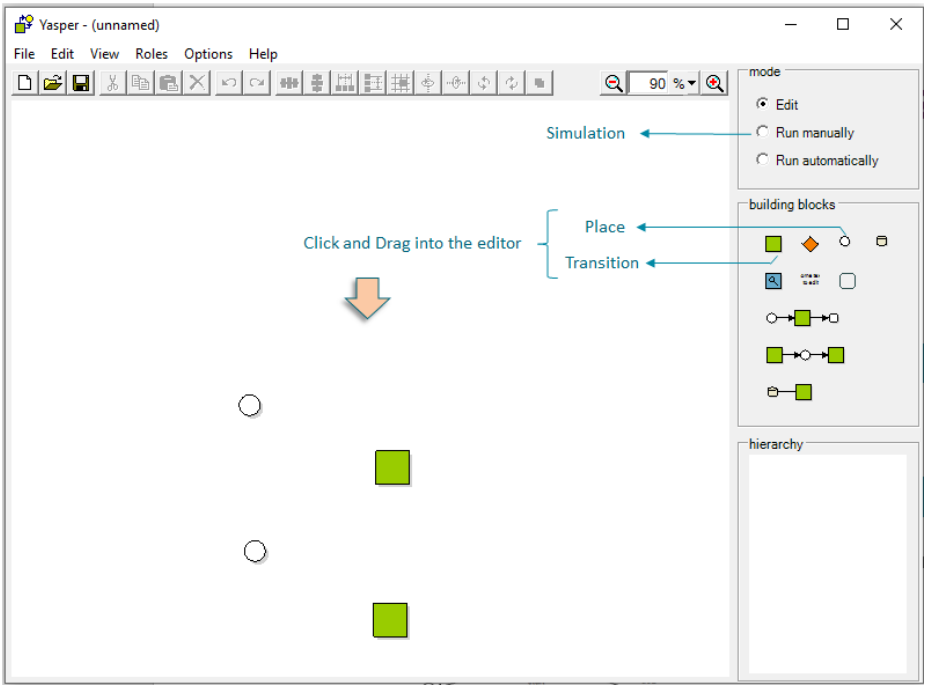
Module 1: Assignments

1 Familiarizing yourself with Yasper

1) Start Yasper and check configuration as shown below

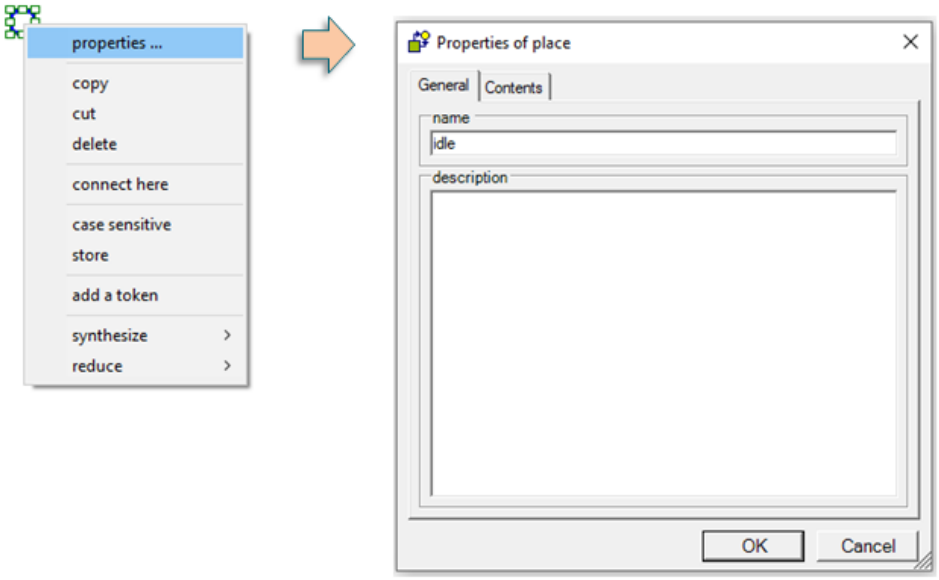


2) Explore the tool layout and try adding places and transitions to the editor.

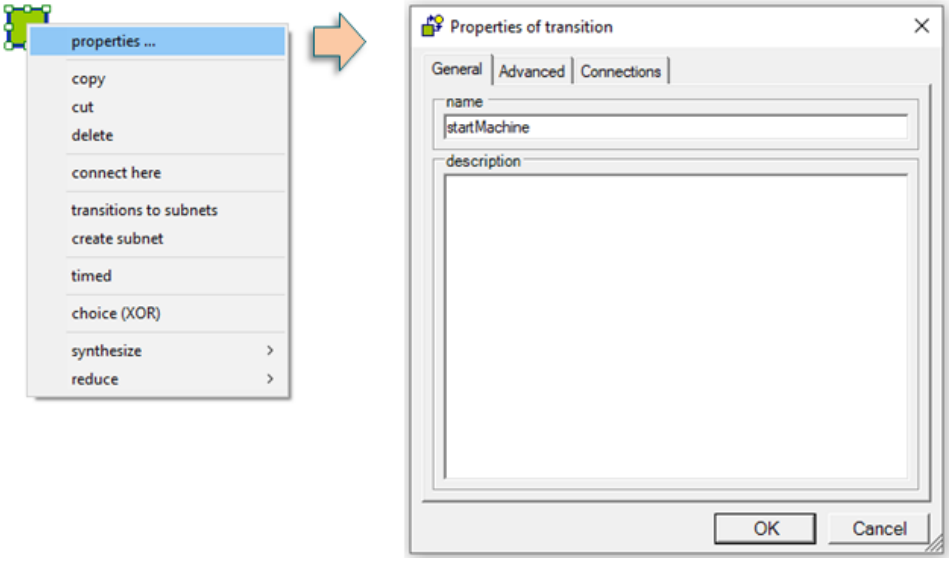


3) Give names to these places and transitions

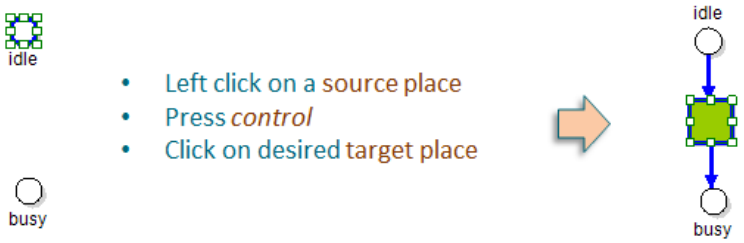
Right click on a place



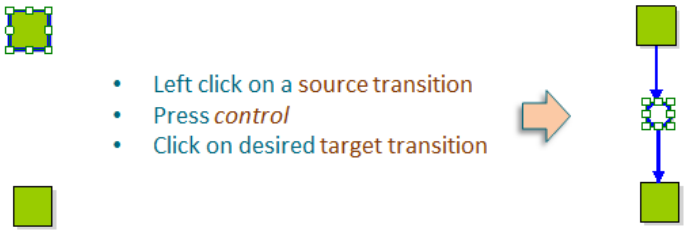
Right click on a transition



4) Now connect two places

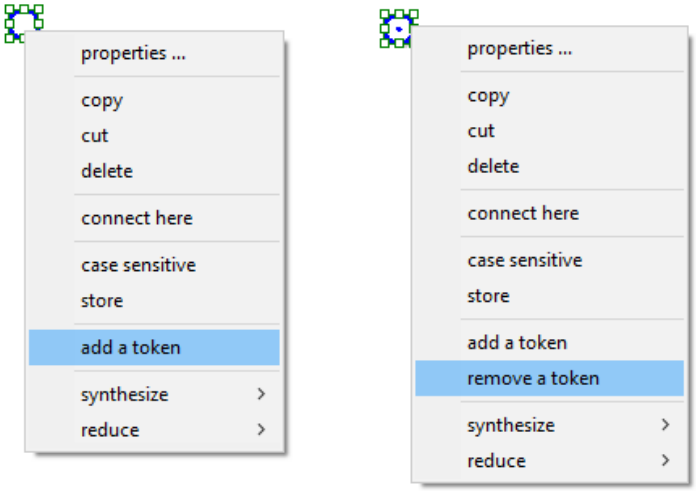


5) Now connect two transitions

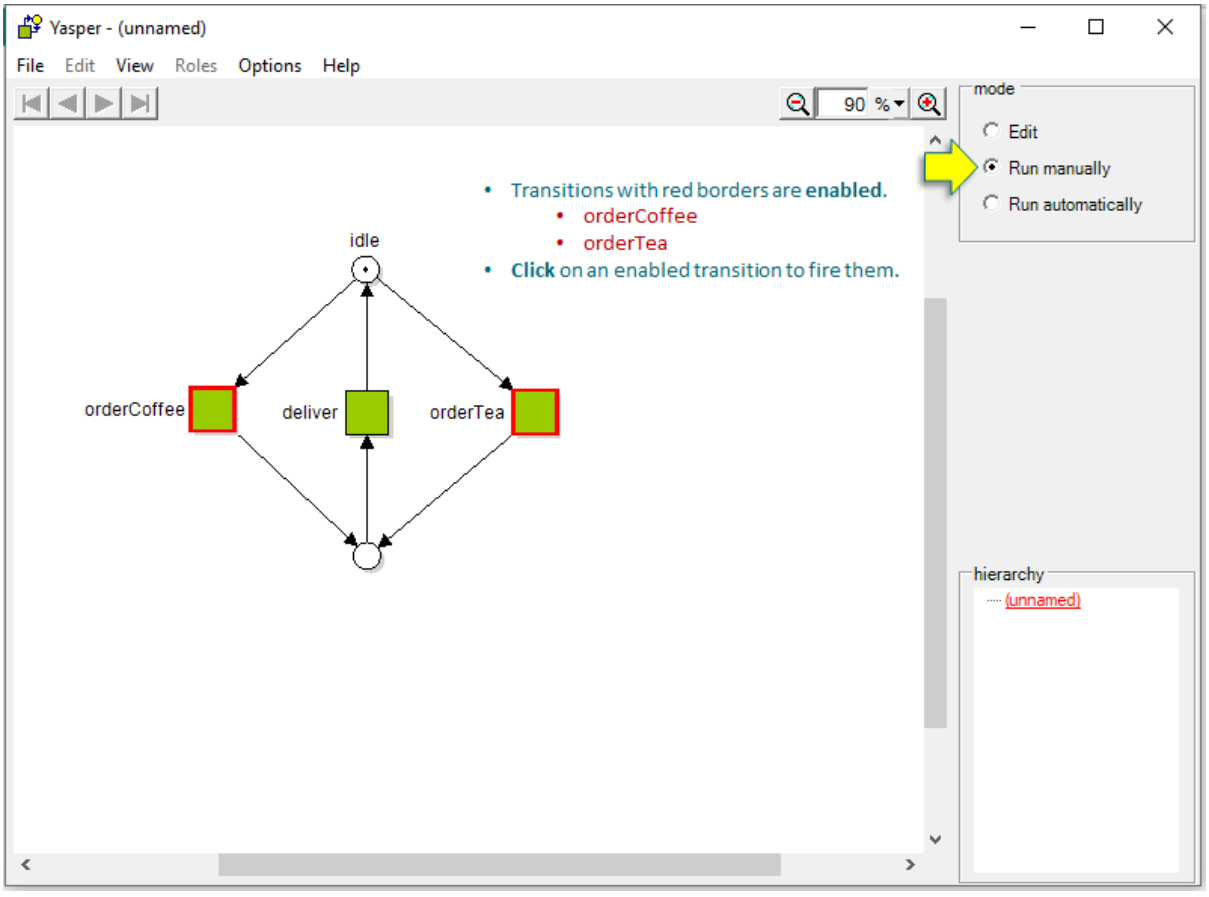


6) Use the same strategy to connect a pair of place and transition.

7) Now try adding and removing tokens from a place.

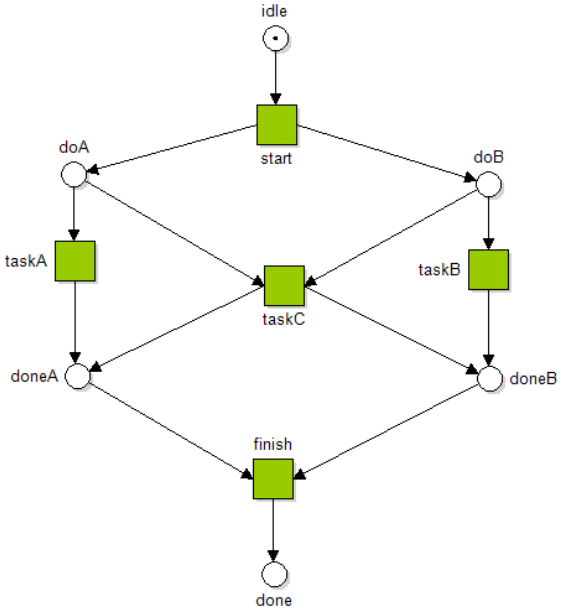


8) Now try simulation. Select option: Run manually and click on enabled transitions to fire them



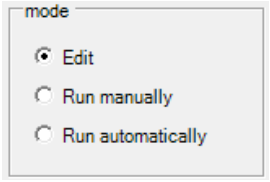
2 Understanding Petri Net Syntax

1) In Models folder you will find a folder named Module 1 Exercises. In this folder, you will find a file named *BasicPetriNet.pnml*. Open it with Yasper.



2) Study the model and try to understand how it behaves. Identify the presence of non-determinism and concurrency.

1) Change mode in Yasper to Run Manually and play the token game. Enabled transitions are marked in Red. Clicking on them fires the transition. Validate your understanding of the behavior.

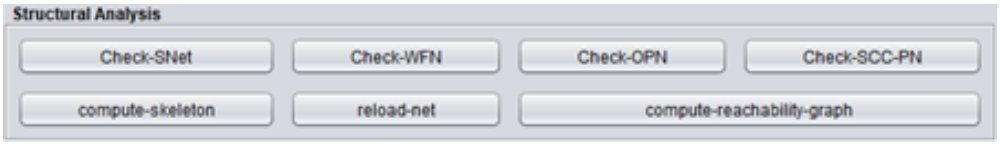


2) Change the mode in Yasper back to Edit and add a transition with name *repeat* to make the net cyclic, i.e. it consumes from place *done* and produces into place *idle*. We refer to such transitions as *closure*. Run the model manually and verify that it is possible to get back to the *idle* place.

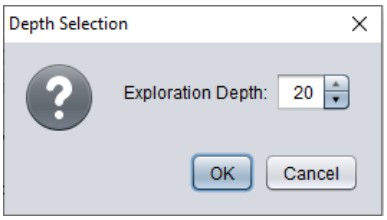
3) Suppose we have a change request! Instead of a single task C, we need to split it up into three tasks (taskC1, taskC2, taskC3), such that taskC2 and taskC3 can be done only after taskC1 has been completed. The taskC2 and taskC3 have no dependencies, so they can be executed concurrently. Run the model and verify that it behaves as intended.

3 Understanding Petri Net Semantics

1) In folder PnAT, double click on *LaunchPetriNetAnalysis.jar* and select the model from the previous step, i.e. *BasicPetriNet.pnml*.

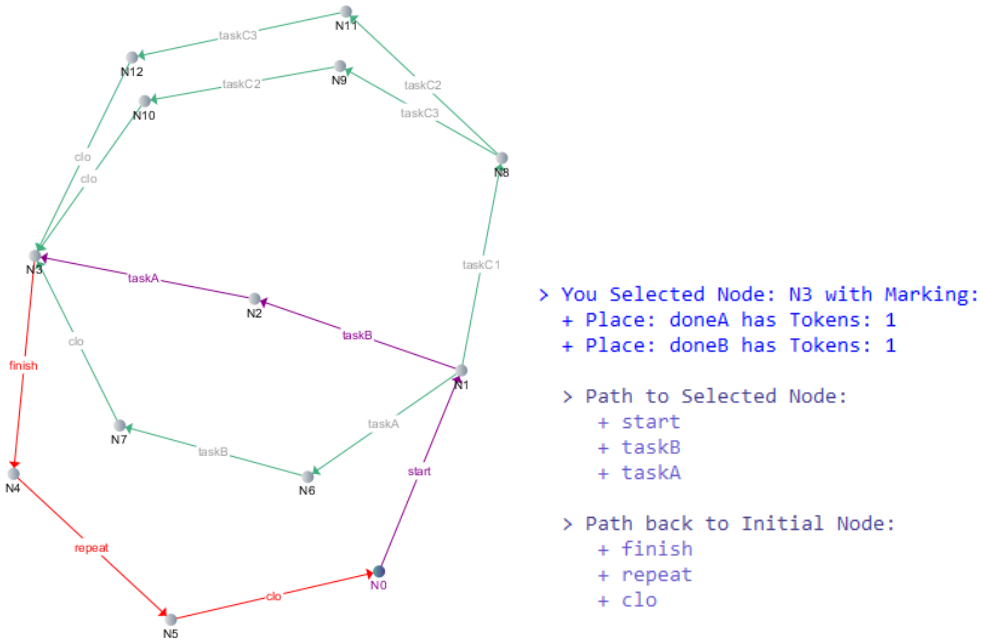


2) You will see many options for analysis. For now, click on compute-reachability-graph. You will be prompted to enter the depth of search, i.e. the maximum number of steps to explore from the initial marking. Enter value 20, and click OK.



3) A file by name *reachabilitygraph.dgs* will have been generated in folder *Models*. In the same folder PnAT, double click on *LaunchPetriNetAnalysis.jar* and select the file *reachabilitygraph.dgs*. The displayed graph is a visualization of the reachability graph.

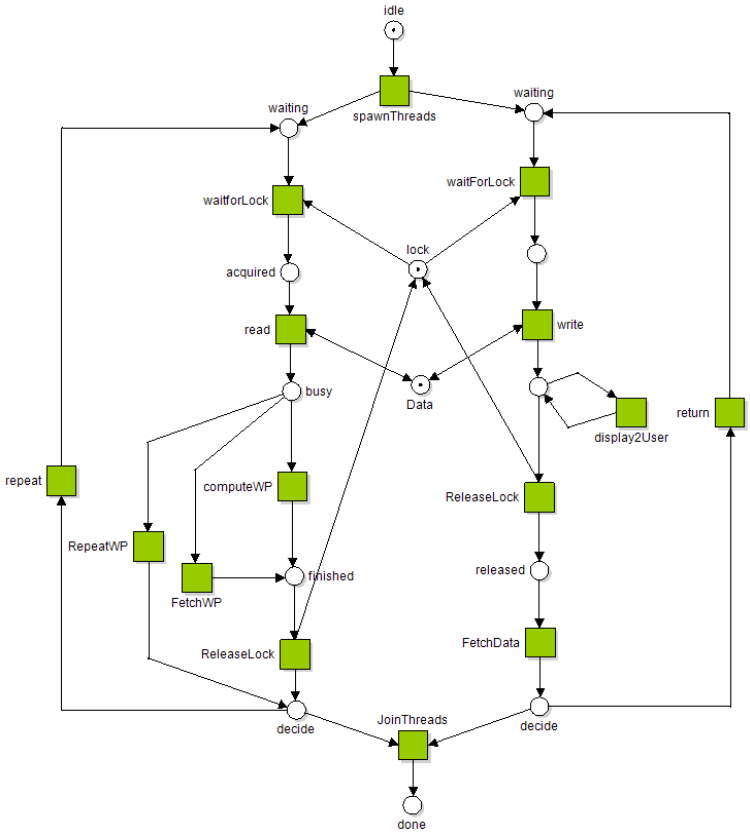
4) Inspect the nodes of the graph by clicking on them



5) Now check for weak termination by clicking on button *check-WT*. Analyze the results. Note that boundedness checks with k-value = 1 (i.e. safe) are also reported.

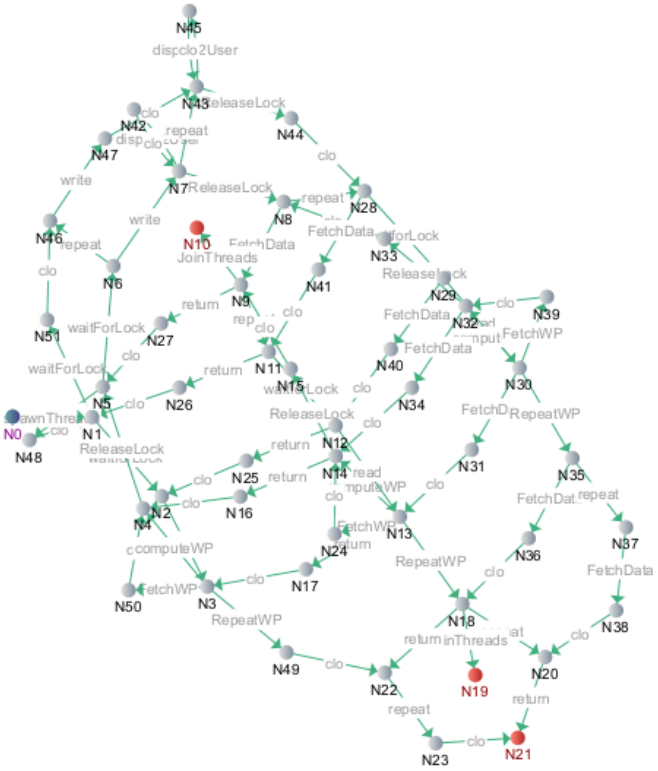
4 Resolving Problems in a Model

1) Open the file *ThreadingAndDeadlock.pnml* in Yasper. The model contains a typical example of threading and synchronization. Look at the model and try to understand what it does. Think about whether this a workflow net and verify your conclusion with PnAT.



2) The model contains a deadlock. If you cannot see the reason, run the model in Yasper and try to find out.

3) Check if you found all the deadlocks using PnAT tools by generating the reachability graph. The visualization of this graph will already indicate if there are deadlock markings by coloring them in Red. Click on these nodes and inspect the graph. Note that a marking with one token in place done is also a deadlock marking but this is not harmful since it is a final marking of a workflow net.



4) Analyze and resolve the deadlock by modifying the net.
5) Add a closure transition and verify that there are still no more deadlocks in the model.

5 Coffee Machine

5.1 Model Interface Protocol of a Coffee Machine

Objective: Understand workflow nets, state machine nets and skeletons of OPN.

1) Model the given description of a coffee machine as a Petri net in Yasper.
   Recall that in an OPN a transition is either a send or receive, determined by the direction of the arc from the transition to its corresponding interface place.
   For now, we will not consider interface places and only look at the protocol as a skeleton net. In the next step, we will associate interface places to transitions.

   In the idle state, it is possible to turn on the machine. As part of turning on, the machine checks for water and milk levels. If both are okay, then there is a positive acknowledgement and the machine goes to the operational state. Otherwise, there is a negative acknowledgement and the coffee machine goes back to idle state.

   In the operational state, it is possible to select the type of coffee. Once the selection is made, the machine goes to the selected state. In the selected state, the machine can accept coins until the amount is sufficient leading to the preparing state. In the preparing state, the machine gives periodic status updates until the coffee is prepared and the machine returns to idle state.

   From both states selected and preparing, it is possible that the machine raises an error and goes to the fault state.

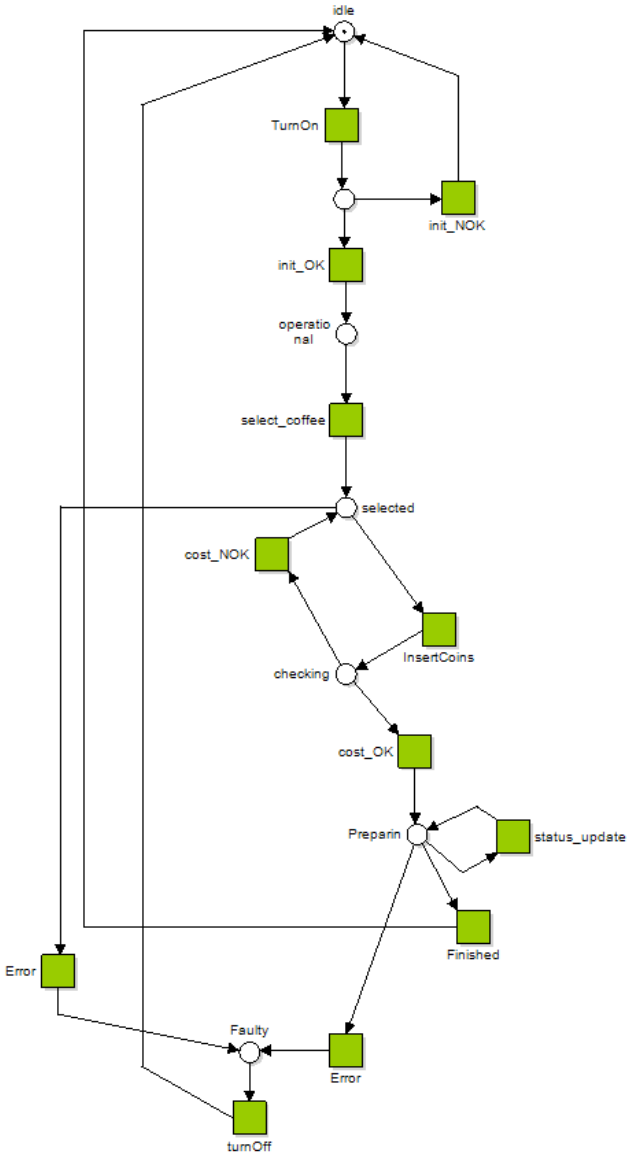
   In the fault state, it is only possible to switch off the coffee machine and return to idle state.

2) Check that the model is not a workflow net.
3) Check that the model is a state machine net.
4) Check that the model is weakly terminating.

5.2 Model Client and Server Perspectives (Open Petri Nets)

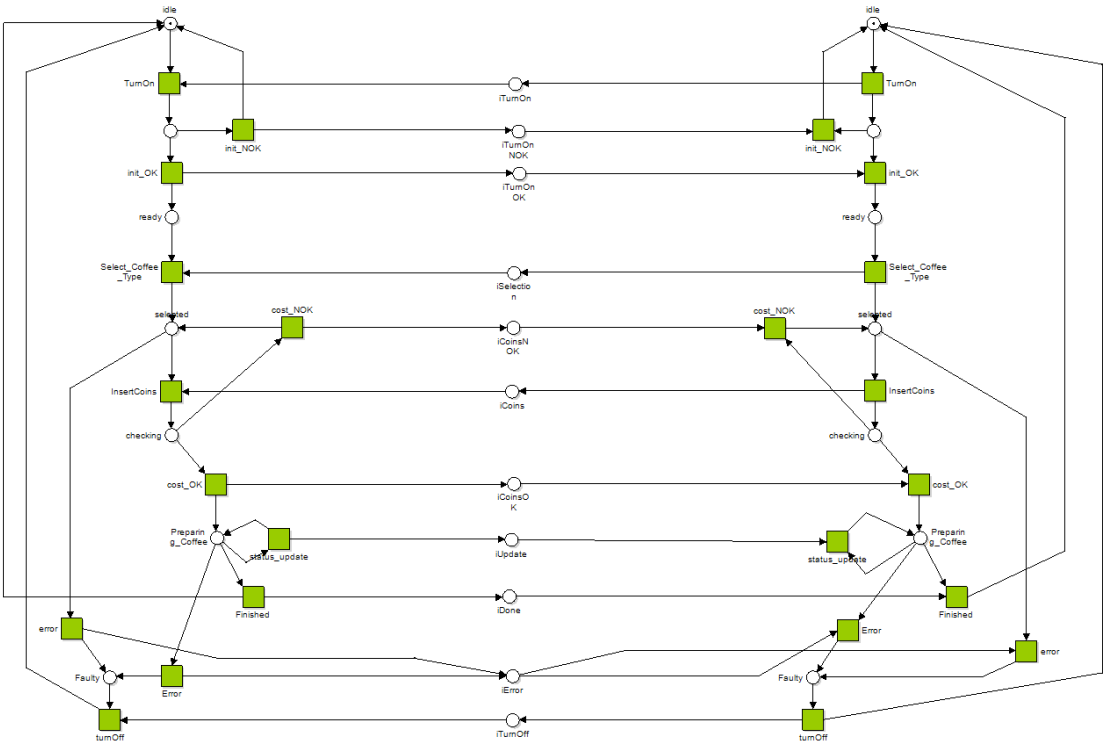
Objective: Understand Open Petri nets

5) If everything went well in the previous step then you must have a model that looks similar to *"ProtocolStateMachine.pnml"*. Check with PnAT that the model is weakly terminating.



6) Make a copy of the skeleton from the previous step. Call one of them a client and other as server.

7) For each unique event, create an interface place with a suitable label. Connect the copies with interface places such that each transition of a client (server) are either producing to (consuming from) an interface place.



8) Check that the model no longer weakly terminates, even though the skeleton was weakly terminating! Resolve the problem(s).

Hint: First resolve the boundedness problems. This results in a much smaller reachability graph. Then find the deadlock and resolve it.

Optional

9) To create a client subnet in Yasper, select all the transition of client, right-click and select create subnet. Repeat the same to create a server subnet.

10) Check using PnAT (i) The model is an OPN, (ii) Client-Server skeletons are S-Nets