Embedded Software and Systems 2022
Assignment 1

Deadline: 23:59, November 18, 2022

The submission should be done digitally via Canvas and contain a PDF file and any digital models (e.g. Yakindu or Yasper) of your solution. Late submissions will not be accepted.

1. Statecharts 4 Points

Draw a model of a chess clock in the Statechart formalism. Your Statechart should include the following events:

- ***white moves:*** an external event indicating that player White's button of the clock is pressed and that it is blacks turn.
- ***black moves:*** an external event indicating that player Black's button of the clock is pressed and that it is whites turn.
- ***white turn:*** an internal event indicating that it is player White's turn (emitted once at the beginning of each turn of White)
- ***black turn:*** an internal event indicating that it is player Black's turn (emitted once at the beginning of each turn of Black)
- ***tick:*** an internal event generated once per second.
- ***white wins:*** an internal event generated when White wins (because Black runs out of time)
- ***black wins:*** an internal event generated when Black wins (because White runs out of time)

At any time, exactly one of the two players is active. Player White starts. For each player, the clock maintains two time variables: local time, which is initially set to 10 seconds, and global time, initially set to 30 minutes. As long as there is time left in the active player's local time, one second is deducted from local for every tick; once the local time has expired, one second is deducted from the active player's global time for every tick. If a global time expires, the active player loses the game. When the active player's button is pressed, its local time is reset to 10 seconds, its global time is incremented by 15 seconds, and the other player becomes active. As a further requirement, the clock should also keep track of the number of turns (e.g., the second turn starts, when player White becomes active for the second time): If the number of turns exceeds 40, then 15 minutes are added to the global times of both players.

2. Petri Nets 3 Points

Model the given server interface description of a coffee machine as a State Machine Open Petri (S-OPN) net. Recall that an interface should not model internal behavior and that each transition is either a send or receive, determined by the direction of the arc from the transition to its corresponding interface place.

In the idle state, it is possible to turn the machine on and receive either a positive acknowledgement and the machine goes to the operational state, or a negative acknowledgement and the coffee machine goes back to the 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 sends periodic status updates until the coffee is prepared and the machine returns to idle state.

In state 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 the idle state.

If correctly modelled, the skeleton of this server interface should be safe and weakly terminating, but the combination of this server with a mirrored client is not because of a leg property violation. Resolve this leg property violation by adding a minimum number of (interface) places and transitions. The final net should be safe and weakly terminating (both can be verified with PnAT).

Tip: A mirrored client can be conveniently modeled in Yasper. This done by copying the current net (select all nodes and then press: control + c) and pasting (press: control + v) it next to it (i.e. nodes are not overlapping). While the pasted net is still selected, flip it horizontally to create a mirrored structure (Edit menu -> Flip horizontally, or control+8). After that, fuse matching interface places by dragging and dropping interface places with matching labels on top of each other. Lastly, invert the arcs from interface places to transitions of the client net by right clicking on the arc and choosing invert.

3. Petri Nets 3 Points

The coffee machine below (available in the file *Exercises/SimpleCoffeeMachineWithProblems.pnml* in the Petri Net tutorial) has a race condition, which can cause it to deadlock and refuse to return an inserted coin. Handle the race condition without removing the choice property violation in the On state, i.e. make sure that all possible outcomes of the race are correctly handled.

Solution requirements:

- The inserted coin should always be returned to the client after an exception on the server side by sending a message in an added iReturnCoin interface place.
- The combination of server and client should be safe and weakly terminating.
- Internal behavior should not be included in the interface models.

A minimum number of extra (interface) places and transitions may be added to solve the assignment. Use functionally meaningful labels for any added places or transitions to make the model easier to understand. The server and client are not required to be mirrors.