UNIVERSITY OF TARTU

MTAT.03.083 – Systems Modelling

Regular Exam – 6 January 2015

Notes:

- The exam is open-book and open-laptop. Web browsing is allowed.
- You are not allowed to communicate with anyone during the exam in any way (except with the lecturer).
- You may submit your exam on paper, or electronically. In the latter case, include all files of your submission in a zip file and submit it using the "Submit" button in the course Web page.
- If you find that there is not enough information in the text below and you need to make additional assumptions, please write down your assumptions.

Part 1. Application for Conformance Checking with Petri nets

A Petri net consists of places, transitions, arcs, and tokens. An arc is a directed connection between a place and a transition, or between a transition and a place. Arcs between two places or two transitions are not allowed. In this exercise, we consider that there is at most one arc going from a given place to a given transition and at most one arc going from a given transition to a given place.

Every transition is labelled with an event. Here, we assume that a Petri net cannot contain duplicate transitions, i.e., there cannot be two transitions labelled with the same event. A place can optionally have a name. Places may hold zero or more tokens. The state of a Petri net (called a *marking*) is a distribution of tokens over the places of the net, meaning a function that tells us how many tokens are located in each place. The state of a net at the beginning of an execution is called the *initial marking*.

A transition is enabled if each of its input places contains at least one token. An enabled transition can fire. When a transition fires, it consumes a token from each input place and produces a token in each output place.

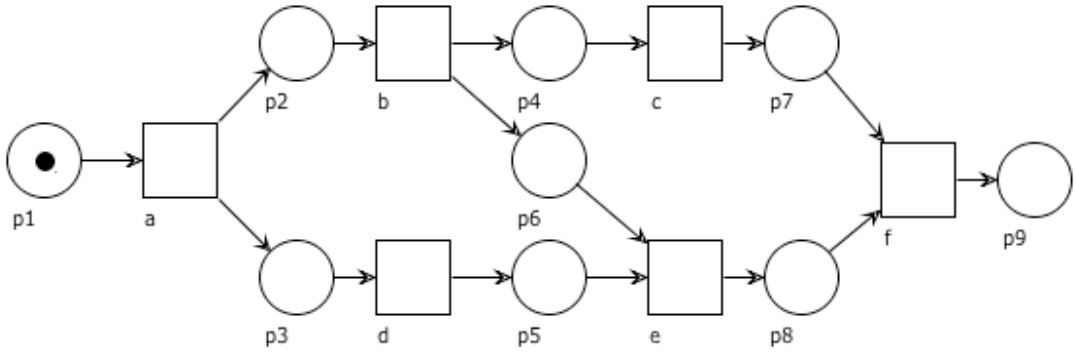
A trace is a finite sequence of events [a, b, d, e, b, e, …] (note that the same event can be repeated multiple times in a trace).

An execution of a Petri net starts from a given initial marking and a given input trace. The execution moves from one marking to another by consuming events from the input trace, starting from the first event in the trace. Given the current marking of the Petri net, an event E can be successfully consumed if there is an enabled transition T labelled with event E. Otherwise, if the current marking does not enable any transition labelled with event E, event E cannot be consumed and the execution of the Petri net is "stuck". When an event E is successfully consumed, the corresponding transition fires and thus the Petri net's execution moves to a new marking.

Given a Petri net P, an initial marking I_m and a final marking F_m, a sequence of events is *conformant* with the triple (P, I_m, F_m) if there is an execution of Petri net P which,

starting from the initial marking I_m, can successfully consume every event in the trace one after another, and after consuming the last event in the trace the execution of the Petri net is in the designated final marking F_m. If after consuming the last event in the trace, the execution is not in the final marking, we say that the trace is *partially conformant*. If at least one event in the trace cannot be consumed, we say that the sequence is *not conformant*.

For example, consider the following Petri net.



Let us consider the case where the starting marking is the one shown in the figure (i.e. one token in place p1, no other tokens anywhere else), while the final marking is the one where there is one token in place p9 and no other token anywhere else. Given these initial and final markings, the trace [a, b, d, e, c, f] is conformant with respect to this net. The same can be said of trace [a, d, b, c, e, f]. However, trace [a, d, e, b, c, f] is not conformant because the execution is "stucked" after event "d" is consumed. Trace [a, b, c] is partially conformant – all events in this trace can be consumed but the final marking is not the one that contains one token in place p9.

The aim of the following tasks is to design an application that takes as input a Petri net P, an initial marking I_m, a final marking F_m, and a trace t, and determines whether trace t is conformant, partially conformant or not conformant with (P, I_m, F_m).

Task 1. [10 points]

Design a domain model (UML class diagram) of Petri nets and their executions as defined above.

Task 2. [10 points]

Write a sequence diagram that describes how the classes in the domain model (and other classes if needed) interact in order to determine if a given event can be successfully consumed by an execution of a Petri net. Assume that the event is produced by an object of an external class called EventFeeder.

Task 3. [10 points]

Write a sequence diagram that describes how the classes in the domain model (and other classes if needed) interact in order to determine if a given input trace is conformant, partially conformant or not conformant with the Petri net. Assume that the trace is produced by an object of an external class called TraceFeeder.

Task 4. [10 points]

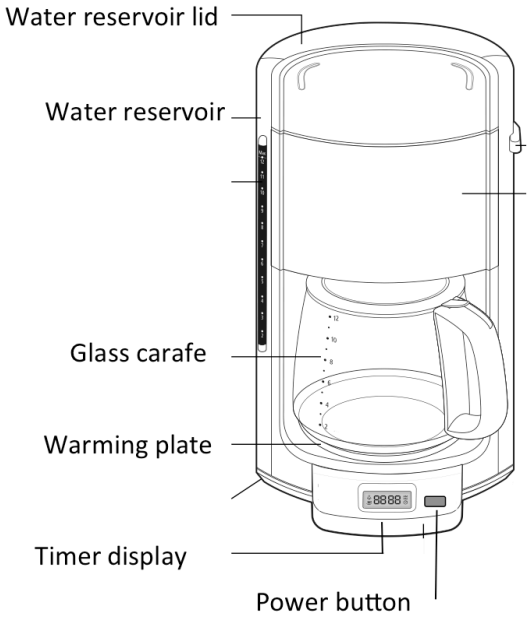
Design an application model for the above scenario. The application model should include methods required to determine if a Petri net execution can consume or not a

given input event, as well as methods required to determine if a trace is conformant, partially conformant or not conformant with the Petri net.

Part 2. Coffee machine

Task 5. [10 points]

Specify the controller of a coffee machine as a statechart diagram.



The key components of the coffee machine are:

1. A single power on/off button,
2. A water reservoir with level sensor (with capacity of up to 2 litters),
3. A warming plate with a weight sensor and a presence sensor (to detect if the carafe is placed on it and the weight inside the carafe),
4. An ergonomic glass carafe with capacity of 1500 ml, and
5. A programmable auto-off timer (for the sake of brevity, we will assume that the function "auto-off timer programming" is specified elsewhere –and not by you. Therefore, we will assume that the auto-off timer is set to 2 hours).

The operation of the coffee machine is as follows. The coffee machine starts operating when the user presses the power button. If the water reservoir is not empty and the carafe is on the warming plate, the coffee machine starts making coffee (i.e. warming up the water in the reservoir). The coffee machine stops making coffee whenever one of the following happens:

1. The carafe contains approx. 1500 ml of coffee, as sensed by the weight sensor (we assume that the weight sensor reports the weight in millilitres based on an approximate conversion from grams to millilitres).
2. the water reservoir is empty (sensed by the level sensor), or
3. the carafe is removed from the warming plate (sensed by the presence sensor).

The warming plate will operate as long as the carafe is on it. However, for safety reasons the warming plate must be turned off when:

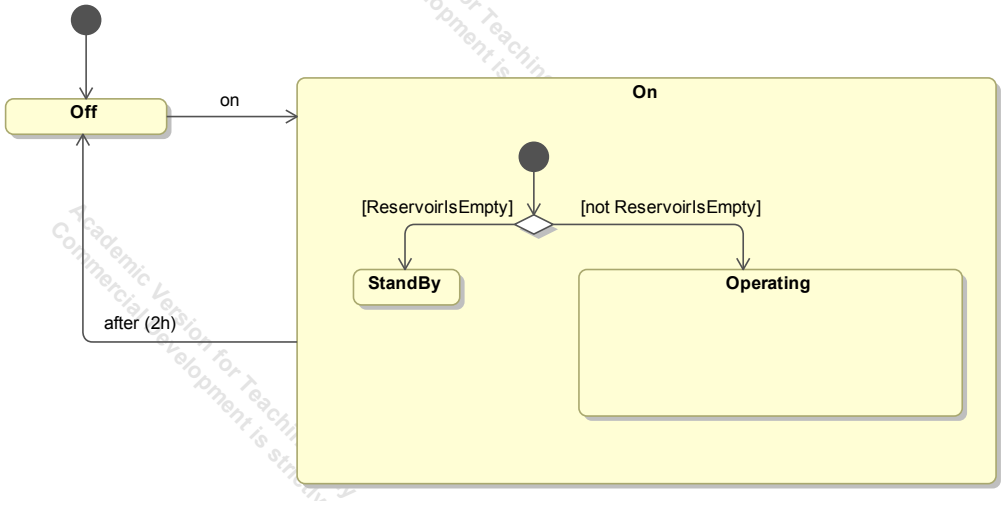
1) the water reservoir is empty (sensed by the level sensor),
2) the carafe contains less than 100 ml (sensed by the weight sensor), or
3) the carafe is removed from the warming plate (sensed by the presence sensor).

When the user returns the carafe to the warming plate, the coffee machine resumes its operation in the state it was before removing the carafe (e.g. either coffee making or warming-only). Since the status of the components of the coffee machine might have been changed while the carafe was out of the warming plate (e.g. the carafe is empty when it is put back into the warming plate), it is important to periodically monitor the state of the components of the coffee machine (e.g. amount of water in the reservoir, amount of water in the carafe, etc.)

The coffee machine is automatically turned off after 2 hours of operation or when the carafe has been removed from the warming plate for more than 5 minutes.

Finally, the user can manually turn the coffee machine off by keeping pressed the power button for at least 2 seconds. (Hint: you can consider that the power button generates two events: buttonPressed and buttonReleased).

Use the following statechart as the starting point for your solution:



For this task, you should submit a statechart diagram (on paper, PDF or MagicDraw format).