Homework 1 (HW1) - Class Modelling (8 points)

Due on 26.09.2016 at 23:59
complete the homework in groups of 2 students.

Submission format: pdf
The submitted file should contain the class model and, if needed, notes specifying assumptions and/or constraints.

Grading criteria:

- 2 pt. Classes correctly identified
- 2 pt. Relations correctly identified
- 2 pt. Multiplicities correct
- 1 pt. Attributes correctly identified
- 0.5 pts. correct use of other elements of a class model
- 0.5 pts. simplicity of the overall model

Note that for each of the above aspects you may lose fractions of points if you provide too complicated solutions.

Transforming a BPMN model into a Petri net

Goal: Create a class model for a model converter, i.e., an application that takes as input a BPMN model and producing as output the translation of the input BPMN model into a Petri net. The class model is a “domain model”. This means in particular that you do not have to specify operations in it and that the classes included in the model should be related to the domain (BPMN and Petri nets in this case) rather than describing system components.

Task: A BPMN model consists of different types of nodes: events, tasks, and gateways. In addition, a BPMN model contains sequence flows, i.e., directed connections between two nodes. Given two nodes there is at most one sequence flow connecting them. Events can be of two types: start events and end events. A start event has only one outgoing sequence flow and no ingoing sequence flows and an end event has only one ingoing sequence flow and no outgoing sequence flows. We assume that in the input BPMN model there is only one start event and only one end event and we assume that every task has exactly one incoming sequence flow and exactly one outgoing sequence flow.

Every task is labelled with a name and has exactly one ingoing sequence flow and one outgoing sequence flow. A gateway can be an AND-split, an AND-join, an XOR-split, or an XOR-join. AND-split and XOR-split have only one ingoing sequence flow and at least two outgoing sequence flows. AND-join and XOR-join have at least two ingoing sequence flows and one outgoing sequence flow. Sequence flows may hold zero or one tokens. The state of a BPMN model (called a marking) is a distribution of tokens over the sequence flows of the model, meaning a function that tells us how many tokens are located in each sequence flow (zero or one). The state of a BPMN model at the beginning of an execution is called the initial marking and consists of one token on the outgoing sequence flow of the start event of the model (meaning that the start event has fired and the process execution can start).
A task is enabled if the ingoing sequence flow contains one token. An enabled task can fire. When a task fires, it consumes a token from the ingoing sequence flow and produces a token in the outgoing sequence flow. An AND-split is enabled if the ingoing sequence flow contains one token. When an AND-split fires, it consumes a token from the ingoing sequence flow and produces a token in all the outgoing sequence flows. An XOR-split is enabled if the ingoing sequence flow contains one token. When an XOR-split fires, it consumes a token from the ingoing sequence flow and produces a token in only one of the outgoing sequence flows. An AND-join is enabled if all the ingoing sequence flows contain one token. When an AND-join fires, it consumes one token from each ingoing sequence flow and produces a token in the outgoing sequence flow. An XOR-join is enabled if one of the ingoing sequence flows contains one token. When an XOR-join fires, it consumes one token from the ingoing sequence flow containing one token and produces a token in the outgoing sequence flow. The state of a BPMN model at the end of an execution is called the **final marking** and consists of one token on the ingoing sequence flow of the end event of the model (meaning that the end event can fire and the process execution can end).

A task can be atomic or compound. A compound task is connected to a sub-process, i.e., a new BPMN model that starts when the compound task fires. The compound task ends (and produces a token in the outgoing sequence flow) when the corresponding sub-process ends. Every sub-process can, in turn, contain compound tasks connected to sub-processes. Hence, the BPMN models we are considering are hierarchical. In addition, we will restrict ourselves to structured BPMN models. This means that for every split there is a unique corresponding join so that the set of sequence flows between a split and its corresponding join form a single-entry-single-exit region of the graph. In Figure 1, the BPMN model on top is unstructured. If we take the AND-split after “Pay by cheque”, we are not able to find any corresponding join gateway so that the set of sequence flows between this split and the join are a single-entry, single-exit region. The second model in this figure is structured.

---

**Figure 1** – Examples of an unstructured and a structured process model
A Petri net consists of places, transitions and arcs. 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. Here, 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. A transition has always at least one ingoing arc and one outgoing arc. A start place has only outgoing arcs and an end place has only ingoing arcs. All the other places have at least one ingoing arc and at least one outgoing arc. The output Petri net deriving from the translation of the input BPMN model always contains only one start place and only one end place.

Every transition is labelled with a name. 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.

The translation from a BPMN model to a Petri net follows the rules specified in Figure 1. The black transition is a special type of transition that does not have a label (because it does not correspond to any action). These transitions are called invisible transitions. In addition, the start event is translated into a start place followed by an invisible task and the end event into an end place preceded by an invisible task. Finally, note that the translation starts from a hierarchical BPMN model and produces a flat Petri net. Examples of an input BPMN model and the output Petri net can be found at: https://www.dropbox.com/s/xd2zbzdbcqm01tx/Process%20documentation%20-%201.0.0.pdf?dl=0 and https://www.dropbox.com/s/n5jz8v1rnwwk8va/PetriTrans.pnml?dl=0

---

**Figure 2 – High-level description of the mapping from BPMN to Petri nets**
Notes:

1. Use MagicDraw to create the class models;
2. This text is supposed to be the description of the entire application and there can be details that are irrelevant for the creation of the class model. For example, you do not have to model the behavior of the application, i.e., you do not have to specify the operations;
3. Submissions: one of the members of the group has to login and submit the assignment using the link “submit” on the course webpage. Please specify in a comment the members of the group. The submission should consist of one single pdf file.