MTAT.03.083 – Systems Modelling

Homework 3 (HW3) – Sequence Diagrams and Code Generation (15 points)

Due on SATURDAY 31.10.2015 at 8 am
complete the homework in groups of 2 students.

Grading criteria:

- 1.5 pts. Correct use of objects in the sequence diagram(s)
- 1.5 pts. Correct use of messages in the sequence diagram(s)
- 1.5 pts. Correct use of other components in a sequence diagram (e.g., loops, optional flows)
- 2 pts. Consistency between sequence diagram(s) and application model (the operations in the application model should correspond to operation invocations in the sequence diagram(s))
- 6 pts. Consistency between Code and Documentation
- 2.5 pts. The application passes a small test

Modelling and Implementing a Conformance Checker

Assumptions: We assume that the Petri net has only 1 start place and only 1 end place. Every event in the log has exactly 1 corresponding transition in the Petri net (they share the same name). In the Petri net there cannot be duplicated transitions (two transitions with the same name). On the other hand, you can have different events with the same name.

Goal: Create one or more sequence diagrams to represent the behavior of a conformance checker. Use the domain model from HW1 as a starting point. Starting from the domain model and the sequence diagram(s) create an application model. The application model has to be used as a starting point to generate code with magic draw. Complete the generated code to implement a fully working conformance checker. The application should take as inputs a Petri net (**a pnml file http://www.pnml.org/**) and an event log (**a xes file http://www.xes-standard.org/**) and producing as outputs different values of metrics representing the level of conformance of the input event log with respect to the input Petri net. I will provide suggestions on where to find parsers for pnml and xes files. However, you can use different solutions or even implement the parsers by yourself. **Important: note that if you use an existing parser the objects generated from it must be used only as an oracle to populate your own classes (after this initialization you forget about it).** The conformance checker can be a command line application or can have a very minimalistic java GUI. **Important: you have to implement 3 separate functionalities to get fitness, simple behavioral appropriateness and simple structural appropriateness.**

You have to submit:

- The domain model
- The sequence diagram(s)
- The application model

- The eclipse java project containing your implemented application (you should provide everything is needed to run it)

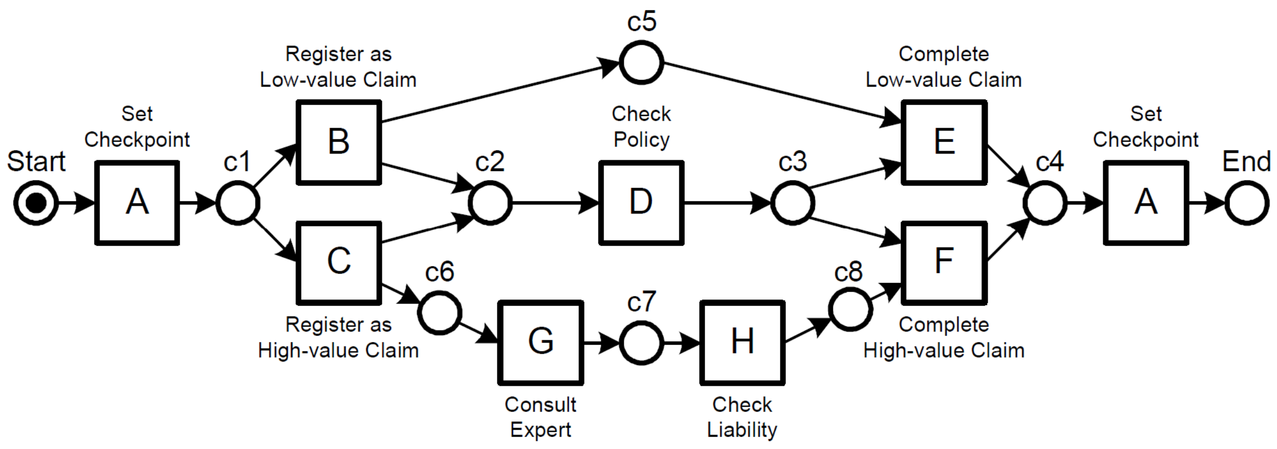
Task: A Petri net consists of places, transitions, arcs, and tokens. An arc is a directed connection between a place and a transition, or between a transition and a place. Arcs between two places or two transitions are not allowed. In this exercise, we consider that there is at most one arc going from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



M1

Let us consider the case where the starting marking is the one shown in the figure (i.e. one token in place *Start*, no other tokens anywhere else), while the final marking is the one where there is one token in place *End* and no other token anywhere else. Given these initial and final markings, the trace ABDEA is conformant with respect to this net. The same can be said of trace ACDGHFA. However, trace ABEA is not conformant. Trace ABD is not conformant as well – all events in this trace can be consumed but the final marking is not the one that contains one token in place *End*.

An event log is a set of cases (also called process instances). Each case corresponds to a trace. Traces are unique and each trace identifies a different sequence of events. However, different cases can correspond to the same trace, i.e., cases are identified by case IDs and different case IDs can correspond to the same sequence of events. A case also contains a key-value map of case attributes. An event has an event name, a timestamp, and a key-value map of event attributes.

For example, in the following three logs, No. of Instances indicates the number of cases corresponding to each trace, and Log Traces are the actual event sequences for each trace. For example: event log L1 contains 4070 cases following the sequence ABDEA.

No. of Instances	Log Traces
4070	ABDEA
245	ACDGHFA
56	ACGDHFA

L1

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

L2

No. of Instances	Log Traces
24	BDE
7	AABHF
15	CHF
6	ADBE
1	ACBGDFAA
8	ABEDA

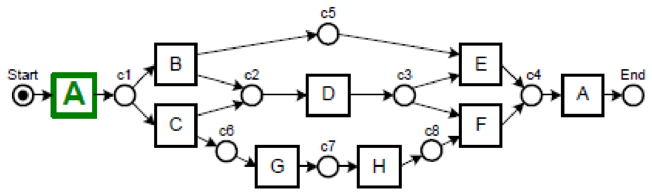
L3

A prerequisite for conformance analysis is that the transitions in the Petri net must be associated with the logged events, which we represent by a label denoting the associated event name. We assume further that all log events that are not associated to any transition in the Petri net are removed before starting the analysis.

The most dominant requirement for conformance is fitness. An event log and a Petri net "fit" if the Petri net can generate each trace in the log. In other words: the Petri net should be able to "parse" every event sequence in the log. Unfortunately, a good fitness does not imply conformance. Indeed, it is easy to construct Petri nets that are able to parse any event log. Therefore, second dimension is introduced: appropriateness. Appropriateness tries to capture the idea of Occam's razor, i.e., "one should not increase, beyond what is necessary, the number of entities required to explain anything". Clearly, this dimension is not as easy to quantify as fitness. We will distinguish between structural appropriateness (if a simple model can explain the log, why choose a complicated one) and behavioral appropriateness (the model should not be too generic and allow for too much behavior). All these metrics can be evaluated through log replay over the Petri net. In the following we give two examples of replay first of a trace that is compliant with model M1 and then of a trace that is not compliant.

Log replay for trace i = 1 of event log L2 in process model M1. The trace can be replayed without any problems, i.e., no tokens are missing (m = 0) or remaining (r = 0):

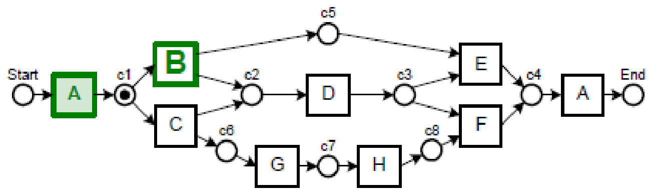
(a)



m = 0
r = 0
c = 0
p = 1

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

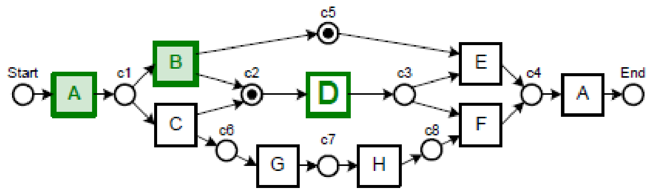
(b)



m = 0
r = 0
c = 1
p = 2

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

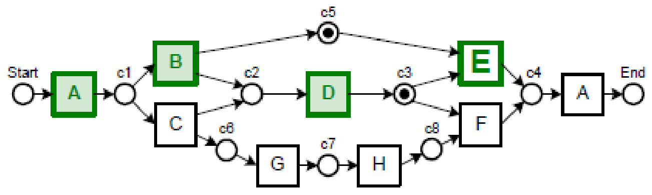
(c)



m = 0
r = 0
c = 2
p = 4

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

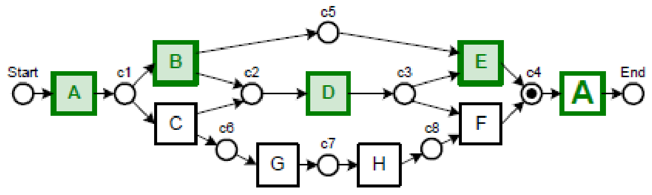
(d)



m = 0
r = 0
c = 3
p = 5

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

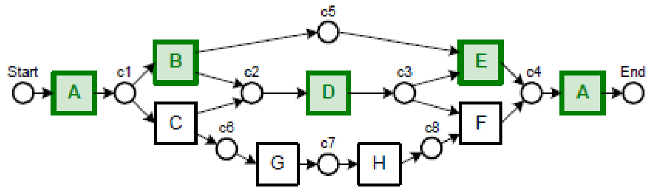
(e)



m = 0
r = 0
c = 5
p = 6

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

(f)

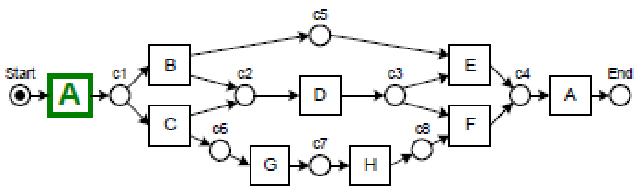


m = 0
r = 0
c = 7
p = 7

No. of Instances	Log Traces
1207	→ ABDEA
145	ACDGHFA
56	ACGDHFA
23	ACHDFA
28	ACDHFA

Log replay for trace i = 4 of event log L2 in process model M1. The replay of this trace requires the artificial creation of one token (m = 1) and one token is left behind (r = 1):

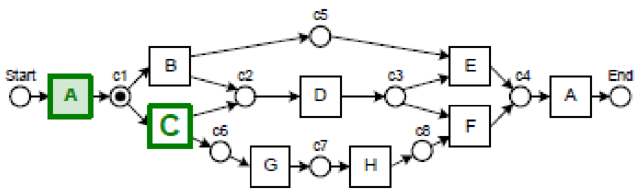
(a)



$m = 0$
$r = 0$
$c = 0$
$p = 1$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

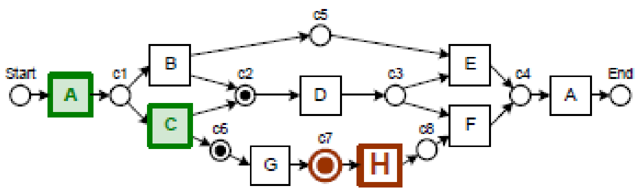
(b)



$m = 0$
$r = 0$
$c = 1$
$p = 2$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

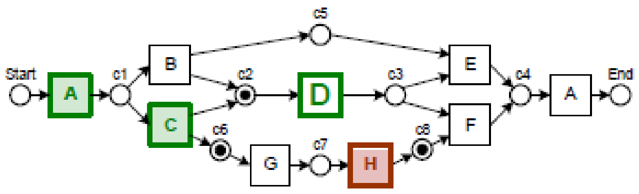
(c)



$m = 1$
$r = 0$
$c = 2$
$p = 4$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

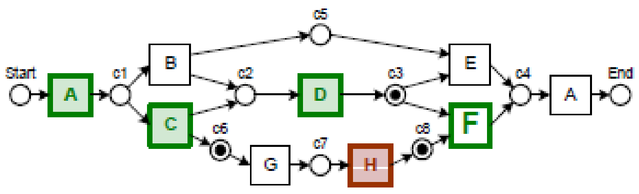
(d)



$m = 1$
$r = 0$
$c = 3$
$p = 5$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

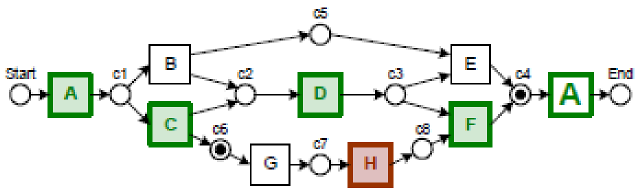
(e)



$m = 1$
$r = 0$
$c = 4$
$p = 6$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

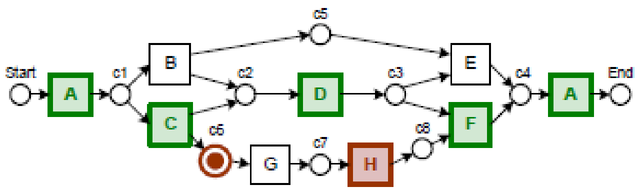
(f)



$m = 1$
$r = 0$
$c = 6$
$p = 7$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

(g)



$m = 1$
$r = 1$
$c = 8$
$p = 8$

No. of Instances	Log Traces
1207	ABDEA
145	ACDGHFA
56	ACGDHFA
23	→ ACHDFA
28	ACDHFA

Starting from the replay of all the traces of a log with respect to a Petri net the following conformance checking metrics can be evaluated.

Fitness *Let k be the number of* different *traces from the aggregated log. For each log trace i $(1 \leq i \leq k)$, n_i is the number of process instances combined into the current trace, m_i is the number of missing tokens, r_i is the number of remaining tokens, c_i is the number of consumed tokens, and p_i is the number of produced tokens during log replay of the current trace. The token-based fitness metric f is defined as follows:*

$$f = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right)$$

Simple Behavioral Appropriateness *Let k be the number of* different *traces from the aggregated log. For each log trace i $(1 \leq i \leq k)$, n_i is the number of process instances combined into the current trace, and x_i is the mean number of enabled transitions during log replay of the current trace. Furthermore, T_V is the set of visible tasks in the Petri net model. The simple behavioral appropriateness metric a_B is defined as follows:*

$$a_B = \frac{\sum_{i=1}^{k} n_i(|T_V| - x_i)}{(|T_V| - 1) \cdot \sum_{i=1}^{k} n_i}$$

Simple Structural Appropriateness *Let L be the set of labels that establish the mapping between tasks in the model and events in the log, and N the set of nodes (i.e., places and transitions) in the Petri net model. The simple structural appropriateness metric a_S is defined as follows:*

$$a_S = \frac{|L| + 2}{|N|}$$

Notes:

1. Use MagicDraw to create the models;
2. I suggest to work based on the following agenda:
    - First week: Sequence diagram(s)/Application model drawing
    - Second/Third week: Code generation/Alignment with the documentation
    - Third/Fourth week: Finalization of the code
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.