Ensimag
3rd year – option ISI
February 2019

Exam

Modeling and Analysis of Concurrent Systems: Models and Languages for Model Checking

Duration: 2 hours
All documents allowed

Warning : You are encouraged to be as careful as possible in writing. You will be judged more on the QUALITY than the QUANTITY of your answers.

The score given for each question is purely informative.

Partie I Modeling in LNT (10 points)

We consider a fragment of the BPMN (Business Process Modeling Notation) standard for describing workflows (WF). We use the term "WF" instead of "BPMN process" to avoid confusion with LNT processes. A WF is a directed graph consisting of nodes (denoting activities) connected by edges (denoting sequence flows). The nodes considered here (see Figure 1) are start and finish events, tasks, and gateways. Start and finish events are used to initialize and terminate WFs, respectively. A task represents an atomic activity, and has exactly one incoming and one outgoing flow. Gateways are used to split and merge the execution flow in a WF.

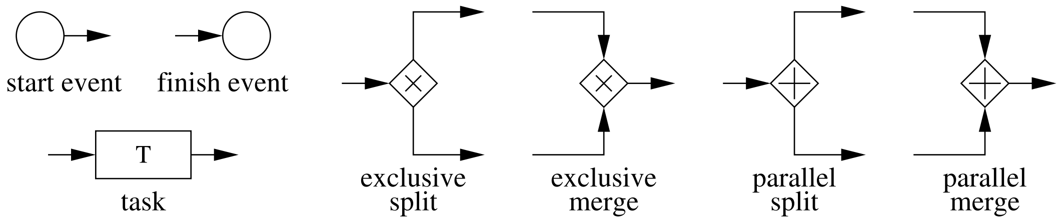


Figure 1: BPMN nodes (events, tasks, and gateways)

The execution of a WF can be defined using "tokens" as follows. Initially, there is exactly one token at the start event. A token can move along a number of sequence flows. A token can also enter and leave a task by following the flow associated to that task. When a token arrives at an exclusive (resp. parallel) split gateway, the gateway is executed, the token is consumed, and one token is generated for a single (resp. for every) outgoing flow of the gateway. When a token arrives at one entry (resp. each entry) of an exclusive (resp. parallel) merge gateway, the gateway is executed, the token is consumed (resp. all tokens are consumed), and one token is generated for the outgoing flow of the gateway.

We model a WF in LNT by representing each node as an LNT process and the sequence flows as gates. Each LNT process representing a task or a gateway executes cyclically (it can be executed several times during the WF lifetime) and synchronizes with its neighbors on the gates denoting the entry and outgoing flows of the node.

For example, the two LNT processes below represent the exclusive split (resp. merge) gateway with two outgoing (resp. incoming) flows.

```
process SPLIT_EXC_2 [INPUT, OUTPUT1, OUTPUT2:none] is
    loop
        INPUT;
        select OUTPUT1 [] OUTPUT2 end select
    end loop
end process

process MERGE_EXC_2 [INPUT1, INPUT2, OUTPUT:none] is
    loop
        select INPUT1 [] INPUT2 end select;
        OUTPUT
    end loop
end process
```

Consider the WF shown on Figure 2, which details the steps required to prepare a business trip within an organization[1]. The process starts by asking for an authorization to organize a trip. Three continuations are possible: the process abruptly terminates, the authorization is to be requested again (e.g., with additional information), or the trip is accepted and then the rest of the process is triggered. In the latter case, the process continues by reserving flight tickets and carrying out mission paperwork. Once the flight tickets are issued, accommodation reservation and other additional activities (e.g., insurance, vaccines) are performed in parallel. The visa process is initiated only when all reservations (i.e., flights and hotels) are ready and when the paperwork is completed. When all the prerequisites for the trip are satisfied, the mission details are stored in a specific database.

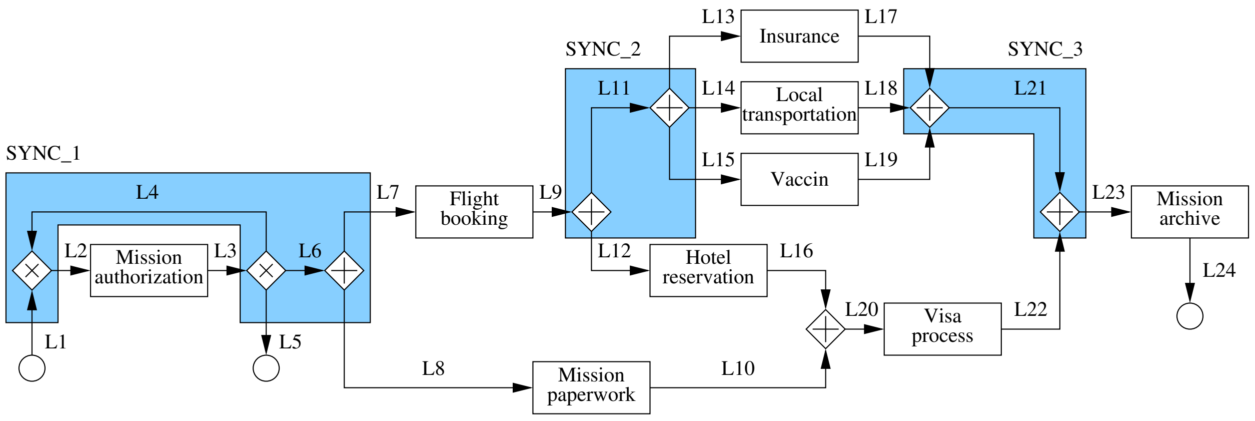


Figure 2: BPMN workflow of a business trip

[1]This is a simplified version of the BPMN workflow given in: F. Durán, C. Rocha, and G. Salaün. Stochastic Analysis of BPMN with Time in Rewriting Logic. Science of Computer Programming 168:1-17, 2018.

Question I.1 Parallel split and merge gateways

A parallel split gateway with two output gates has the following cyclic behaviour: it accepts a synchronization on its input gate; then it performs synchronizations on its two output gates (in any order); and finally it resumes its cyclic behavior. A parallel merge gateway with two inputs has a cyclic behaviour symmetric w.r.t. a parallel split gateway.

Complete the definition of the two processes below modeling these two types of gateways.

```
process SPLIT_PAR_2 [INPUT, OUTPUT1, OUTPUT2:none] is
    -- portion to complete (about 4-6 lines)
end process

process MERGE_PAR_2 [INPUT1, INPUT2, OUTPUT:none] is
    -- portion to complete (about 4-6 lines)
end process
```

Question I.2 Composition of gateways

We consider the composition of the three gateways present in the grey area SYNC_1 in Figure 2. Assuming that the exclusive split gateway with 3 outputs is already available as an LNT process SPLIT_EXC_3 (defined similarly to SPLIT_EXC_2), complete the definition of process SYNC_1 below. The gates used for synchronizing the gateways inside the grey area SYNC_1 are not visible to the external world.

```
process SYNC_1 [L1, L2, L3, L5, L7, L8:none] is
    -- portion to complete (about 9 lines)
end process
```

Question I.3 Synchronous product of two gateways

We consider here the two gateways connected by gate L6 that are part of the SYNC_1 process. The corresponding Labeled Transition Systems (LTSs) are shown on Figure 3.

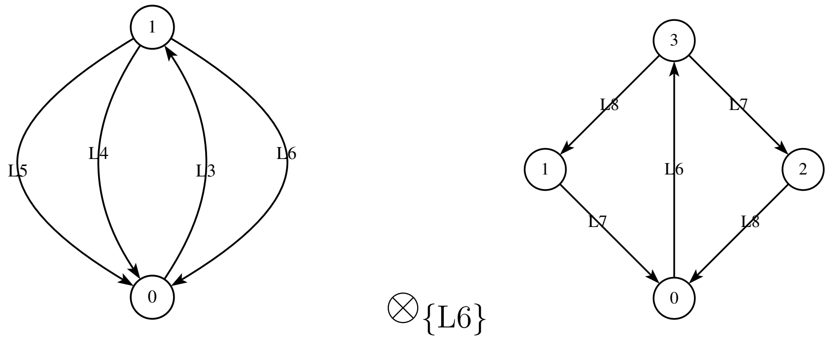


Figure 3: LTSs of exclusive 3-split gateway (left) and parallel 2-split gateway (right)

Draw the LTS of the synchronous product of the two LTSs in Figure 3, synchronized on gate L6.

Question I.4 Global workflow

We assume that the following LNT processes are already defined:

```
process SYNC_1 [L1, L2, L3, L5, L7, L8:none] is
    -- as defined at Question I.2
end process

process SYNC_2 [L9, L12, L13, L14, L15:none] is
    -- nodes contained in the SYNC_2 grey area in Figure 2
end process

process SYNC_3 [L17, L18, L19, L22, L23:none] is
    -- nodes contained in the SYNC_3 grey area in Figure 2
end process

process TASKS_EVENTS [START, MISSION_AUTHORIZATION, FLIGHT_BOOKING,
                      MISSION_PAPERWORK, HOTEL_RESERVATION, INSURANCE,
                      LOCAL_TRANSPORTATION, VACCINATION, VISA_PROCESS,
                      MISSION_ARCHIVE, FINISH, L1, L2, L3, L5, L7, L8,
                      L9, L10, L12, L13, L14, L15, L16, L17, L18, L19,
                      L20, L22, L23:none]
is
    -- parallel composition of all task and start/finish nodes in Figure 2
end process
```

Complete the definition of the process MAIN below, which represents the whole WF shown on Figure 2, as a parallel composition of the processes above. You can use the shorthand notation "TASKS_EVENTS [...]" for the invocation of the TASKS_EVENTS process. Only the gates present as parameters of the MAIN process are visible to the external world.

```
process MAIN [START, MISSION_AUTHORIZATION, FLIGHT_BOOKING, MISSION_PAPERWORK,
              HOTEL_RESERVATION, INSURANCE, LOCAL_TRANSPORTATION, VACCINATION,
              VISA_PROCESS, MISSION_ARCHIVE, FINISH:none]
is
    -- portion to complete (about 16 lines)
end process
```

Partie II Timed automata (5 points)

We propose to model a timed version of the parallel split gateway (defined in LNT in Question I.1) using a timed automaton. The specification of the parallel split gateway is extended as follows, where we assume that t_1, t_2, and t_3 are time constants. Time constraints are written in italics.

The parallel split gateway with two output gates has the following cyclic behaviour: it accepts a synchronization on its input gate; then it performs synchronizations on its two output gates (in any order), with the following time constraints:

- *the first synchronization on an output gate occurs after a delay of at least t_1 time units after the synchronization on the input gate*;

- *a delay of at least t_2 time units elapses between the first and second synchronizations on the output gates*;
- *the second synchronization on an output gate must happen at most t_3 time units after the synchronization on the input gate*;

finally, the gateway resumes its cyclic behavior.

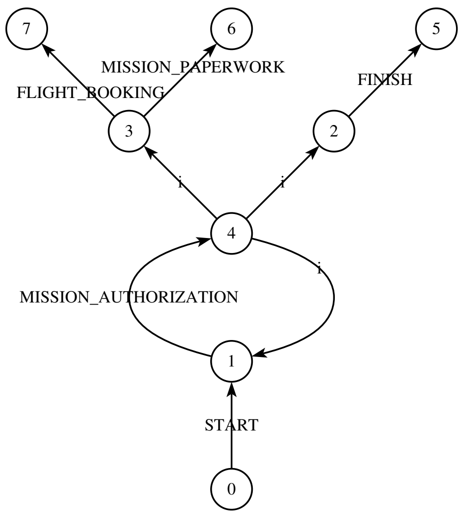
Use the same conventions as seen in class to draw the automaton, i.e, the initial state is identified by a double circle, time invariants are written inside states, and transition labels have the form "$g/a; r$", where g is a guard, a is an action, and r is a (sequence of) resets of the form $x := 0$ where x is a clock.

Hint: you will certainly need more than one clock.

Subsidiary question: which constraints relating t_1, t_2, and t_3 have to be satisfied for this automaton not to lead to a timelock situation?

Partie III Temporal logic (5 points)

Consider the LTS illustrated on the figure below (the initial state has number 0).



For each of the temporal logic formulas below, indicate the numbers of the states satisfying the formula:

1. $[\,\text{true}\,]\,\text{false}$
2. $\langle\,\text{START} \vee \text{FINISH}\,\rangle\,\text{true}$
3. $[\,\text{MISSION\_AUTHORIZATION}\,]\,\text{true}$
4. $\langle\,\text{true}^*.\text{FINISH}\,\rangle\,\text{true}$
5. $[\,\text{true}^*.\text{FLIGHT\_BOOKING}\,]\,\text{false}$
6. $[\,(\neg\text{FINISH})^*\,]\,\langle\,\text{true}^*.\text{FINISH}\,\rangle\,\text{true}$
7. $\mu X.[\,\text{true}\,]\,X$
8. $\mu X.\langle\,\text{true}\,\rangle\,\text{true} \wedge [\,\neg\text{FINISH}\,]\,X$
9. $\nu X.[\,\text{FINISH}\,]\,\text{true} \wedge [\,\text{true}\,]\,X$
10. $\nu X.\langle\,\text{MISSION\_AUTHORIZATION}.\text{true}\,\rangle\,X$

Subsidiary question: which of the above formulas have the same interpretation on any LTS?