Quantitative Evaluation of Embedded Systems (IN4390)
Practical assignments 2021/2022

1 Assignment 2

Learning Objectives. At the end of this assignment you will be able to **(i)** use Petri nets to model a given system, **(ii)** analyze a model to check for deadlocks, etc., and **(iii)** and use reachability and coverability graphs.

Deadline. The due date for mandatory questions is an **Monday 13.12.2021 at 23:59**. The deadline is firm. Submissions that are uploaded after the deadline will not be graded.

Deliverables. Answer the following questions in a single report and upload it on BrightSpace. Please do not forget to add your names, student numbers, team number and the date of submission to the file you upload.

Assessment Instructions. You must answer all mandatory questions. The grade is on a scale from 0-10. You will *pass* this mandatory assignment only if you gain at least 6 points. If answers are incorrect, you will receive feedback from TA's and you will have one more chance to submit the assignment.

2 Mandatory Questions

2.1 Question 1 (4 points)

Consider the Petri net model shown in Fig. 1. Does this Petri net have a deadlock?

- If your answer is **"no"**, prove your claim by drawing the reachability graph and discussing why there is no deadlock.
- If your answer is **"yes"**, show a sequence of transitions that lead to a marking that causes the deadlock. As the second step, explain how to fix the deadlock by adding element(s)[1] to the Petri net. Find a solution that **requires the smallest amount of modification** in terms of the number of elements/changes. Justify your solution.

2.2 Question 2 (4 points)

Consider the Petri net shown in Fig. 2.

1. Draw the coverability tree/graph for this Petri net.
2. Is the following property true about the Petri net? **"Property 1: In any reachable marking m, the number of tokens in p_2 is smaller than or equal to the number of tokens in p_3"** (namely, $\forall m \in R(m_0), m(p_2) \leq m(p_3)$). Justify your answer.
3. Is the following property true in this Petri net? **"Property 2: Marking $m = (0,1,1000)$ can happen infinitely often"**. Justify your answer.

[1] Elements that you can consider are as follows: adding places, transitions, or arcs, changing label of some of the arcs, adding tokens.

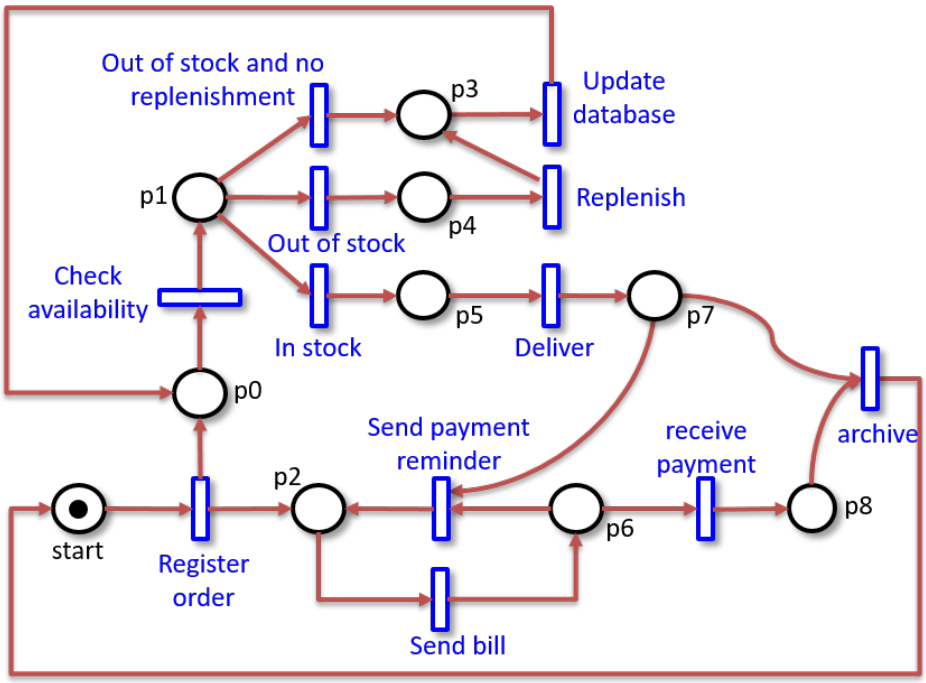

Figure 1: Petri net model for Question 1.

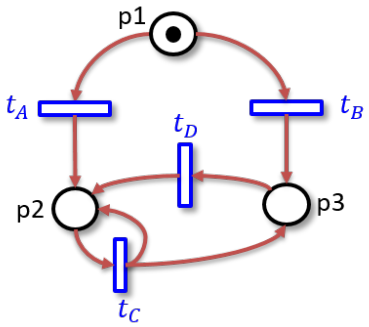

Figure 2: The Petri net model of Question 2.

2.3 Question 3 (2 points)

Show that two Petri nets may have the same coverability graph/tree while one of them has at least one marking that cannot be generated by the other one. To justify your answer, create two such Petri nets. Explain your answer and arguments clearly.

3 Assignment 2: Optional Parts

Deadline. The due date for optional questions is on **Thursday 23.12.2021 at 23:59**. The deadline is firm. Submissions that are uploaded after the deadline will not be graded.

Deliverables. Upload a single report for all of the optional questions that you would like to answer on BrightSpace. Please do not forget to add your names, student numbers, and the date of submission to the file you upload.

Assessment Instruction. If your grade for an optional question is *below* 50% of the points of that question, you will not receive any point for that question. For example, if a question has 20 points and you scored 9, then you will not receive any point for that question. Consult a TA if you are not sure if your answer will qualify for grading.

3.1 Question 4 (10 points)

Consider the Petri net shown in Fig. 3. Add missing elements (you are allowed to add arcs, tokens, places, and transitions) such that:

- Process 2 can perform t_C only if Process 1 has already performed t_B,
- Process 1 can perform t_2 or t_3 only if Process 2 has performed t_5, and
- any other transition of the two processes can happen at any time with any order.

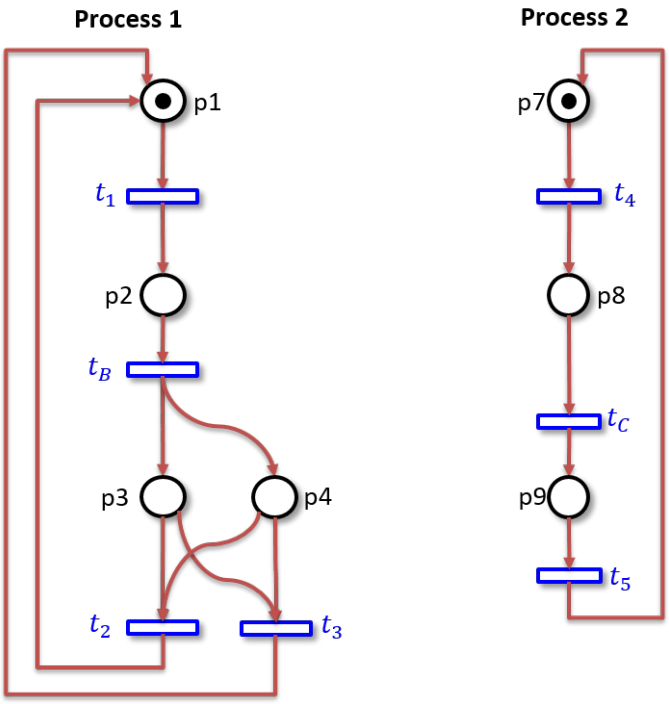

Figure 3: The Petri net model of Question 4.

3.2 Question 5 (50 points)

Description. Model the following system with Petri net. The system includes a base station that provides three types of services for IoT devices that connect to it: (1) object identification, (2) location estimation, and (3) congestion prediction. Each service has several tasks (described below) that must be followed in sequential order (for example, object identification requires Task 1 and Task 2 to be performed one after the other). A task can start only if "enough" processors are available. A task **keeps the processors** until it finishes, and then releases the processors.

1. **Object identification**: **Task 1**: noise removal (needs 2 processor), **Task 2**: object detection (needs 3 processors).
2. **Location estimation**: **Task 1**: scene identification (needs 1 processor), and **Task 2**: location recognition (needs 2 processors)
3. **Congestion prediction**: **Task 1**: reading database (needs 2 processor)

The system has 4 processors that can work concurrently and can serve **at most 2 service-requests at a time** (they can be of any of the three types). The requests that arrive when the system is busy will be pushed to a queue with an infinite capacity.

Deliverables. Present the Petri net model of your system and explain how it works (**10 points**). Your Petri net model must represent the number of processors that are busy and status of each service (i.e., the task being performed) at any time. Then answer the following questions.

Questions. Answer the following questions about the system by analyzing your Petri net:

1. **(20 points)** Does the system have a deadlock? If yes, explain the scenario that reaches to the deadlock, if no, prove there is no deadlock using the reachability graph of the Petri net (please do not use hand-drawings).
2. **(15 points)** Is it possible that one of the requests that has been admitted in the system is never completed (i.e., stays pending forever)? Namely, is there starvation? If your answer is **no**, explain it using the reachability graph. If your answer is **yes**, explain how that may happen.
3. **(5 points)**: find the minimum number of processors that are needed to ensure that any task of any request can be served immediately without waiting for a processor to become available. Justify your answer.