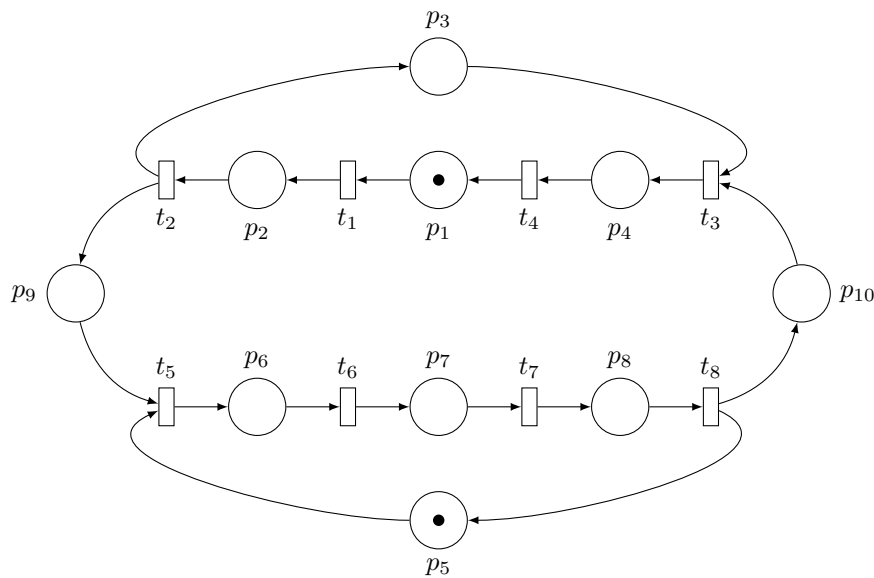


Discrete Event Systems

Solution to Exercise Sheet 11

1 Structural Properties of Petri Nets and Token Game

Given is the following Petri net N_1 :



- a) What are the input (Pre) and output ($Post$) sets of transitions t_5 and t_8 ? Of place p_3 ?
- b) Starting from the initial marking, which transitions are enabled after t_1 and t_2 have fired?
- c) What is the total number of tokens in N_1 before and after t_2 fired?
- d) Play the token game for N_1 and construct the reachability graph.

Hint You may denote the states in such a way that the index indicates the places that hold a token in this state, for example $\vec{s}_0 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0) \triangleq s_{1,5}$.

a) The Pre and Post sets of a transition are defined as follows:

- Pre set: $\bullet t := \{p \mid (p, t) \in F\}$
- Post set: $t \bullet := \{p \mid (t, p) \in F\}$,

where F is the flow set, i.e., the set of place/transition and transition/place arcs. The Pre and Post sets of a place are defined analogously.

For the Petri net N_1 we obtain the following sets:

$$\begin{aligned} \bullet t_5 &= \{p_5, p_9\}, & t_5 \bullet &= \{p_6\} \\ \bullet t_8 &= \{p_8\}, & t_8 \bullet &= \{p_{10}, p_5\} \\ \bullet p_3 &= \{t_2\}, & p_3 \bullet &= \{t_3\} \end{aligned}$$

- b) A transition is enabled if all places in its Pre set contain enough tokens. In the case of N_1 , which has only unweighted edges, one token per place suffices. When t_2 fires, it consumes one token out of each place in the Pre set of t_2 and produces one token on each place in the Post set of t_2 . Hence, the firing of t_2 produces one token on place p_3 and p_9 each, while the token in p_2 is consumed.

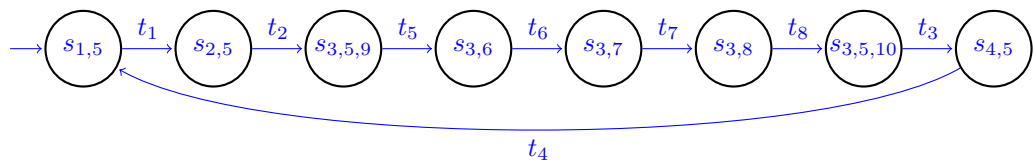
As a result, t_5 is enabled because both p_9 and p_5 hold one token. However, t_3 is not enabled because p_3 contains a token but p_{10} does not.

- c) Before t_2 fires there are two tokens in N_1 , one in p_2 and one in p_5 . After the firing, there is one token in places p_3 , p_9 and p_5 , hence 3 tokens in total.
- d) A token traverses the upper cycle until t_2 fires. Then one token remains on p_3 and waits, and another one is produced in p_9 , which enables transition t_5 . When t_5 consumes the tokens on p_9 and p_5 and produces a token on p_6 , this one traverses the lower cycle until t_8 is enabled and fired. One token now remains on p_5 and waits, another is in p_{10} and enables t_3 , because there is another token on p_3 . Then one token traverses the upper cycle again until t_2 is enabled, and so on. This Petri net models two alternating processes.

This Petri net is clearly bounded, thus we can construct its reachability tree. Usually the states of Petri nets are denoted by vectors such that the i -th position in the vector indicates the number of tokens on place p_i of the Petri net, i.e., the marking of the graph. So, for example, the starting state \vec{s}_0 of N_1 , in which the places p_1 and p_5 hold one token each, is denoted by $\vec{s}_0 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0)$.

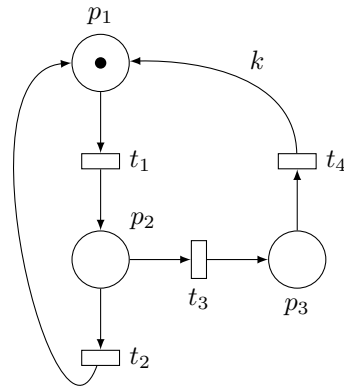
For better legibility we denote the states in such a way that the index contains the places that hold a token in this state, for example $\vec{s}_0 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0) \triangleq s_{1,5}$.

Then the reachability graph can also be written as,



2 Basic Properties of Petri Nets

Given is the following Petri net N_2 :



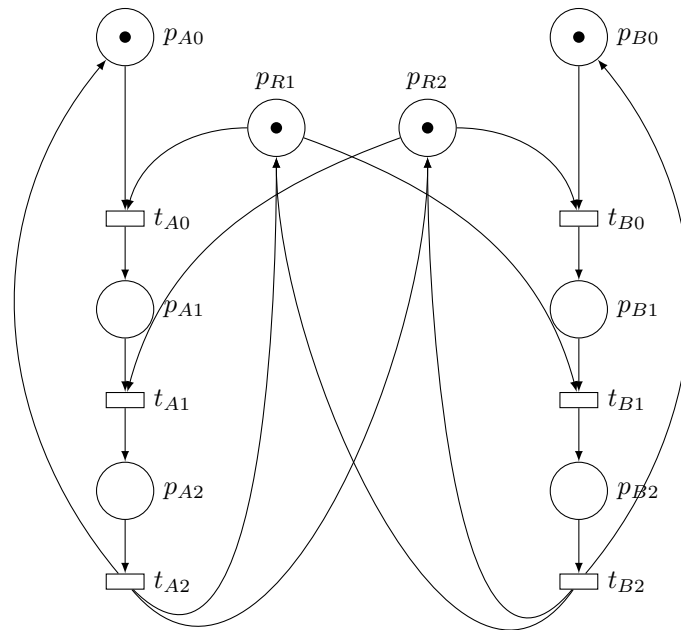
Explain the terms *boundedness* and *deadlock-freeness* using this example, i.e. for which values of $k \in \mathbb{N}$ is the Petri net N_2 bounded/unbounded and not deadlock-free?

A Petri net is k -bounded, if there is no fire sequence that makes the number of tokens in one place grow larger than k . It is obvious that Petri net N_2 is 1-bounded if $k \leq 1$. This holds because in the initial state there is only one token in the net, and in the case $k \leq 1$ no transition increases the number of tokens in N_2 . If $k \geq 2$, the number of tokens in p_1 can grow infinitely large by repeatedly firing t_1 , t_3 and t_4 . So, the Petri net N_2 is unbounded for $k \geq 2$.

A Petri net is deadlock free if no fire sequence leads to a state in which no transition is enabled. If $k = 0$, N_2 is not deadlock-free. The fire sequence t_1, t_3, t_4 causes the only existing token to be consumed and hence, there is no enabled transition any more. For $k \geq 1$, however, no deadlock can occur.

3 Identifying a deadlock

The following Petri net N_3 describes two linear processes ($P_{A0/A1/A2}$ and $P_{B0/B1/B2}$) sharing resources R_1 and R_2 .

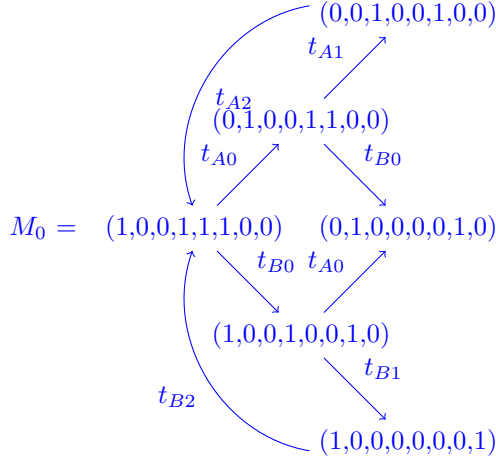


In the following, use $M = (P_{A0}, P_{A1}, P_{A2}, P_{R1}, P_{R2}, P_{B0}, P_{B1}, P_{B2})$ as marking vector and $T = (t_{A0}, t_{A1}, t_{A2}, t_{B0}, t_{B1}, t_{B2})$ as firing vector.

- Construct the reachability graph of net N_3 for the given initial marking. Explicit one firing sequence leading to a blocking marking (i.e., to a deadlock). What is the blocking marking you obtain?
- Express the upstream (W^-) and downstream (W^+) incidence matrices and deduce the incidence matrix A . Using A , compute the marking reached after firing $t_{A0}t_{B0}$ from the initial marking.
- Using the upstream incidence matrix W^- , how can you **prove** that the marking obtained in the previous question represents a deadlock state?
- Suggest a modification to this Petri net which allows the two linear processes P_A and P_B to run "as intended in the first place" (i.e. without blocking).

Hint You don't need more than one place and a few arcs.

- a) There are an infinite number of blocking sequence: any number of cycles $t_{A0}t_{A1}t_{A2}$ and/or $t_{B0}t_{B1}t_{B2}$ terminated by either $t_{A0}t_{B0}$ or $t_{B0}t_{A0}$. It can be read directly from the marking graph below:



- b) From the Petri net structure, we get:

$$W^+ = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}, W^- = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \text{ and}$$

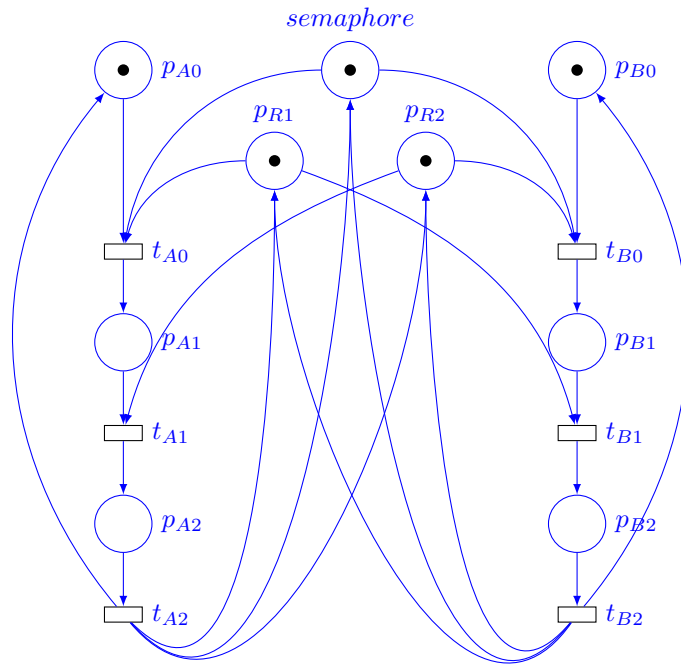
$$A = W^+ - W^- = \begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & -1 & 1 \\ 0 & -1 & 1 & -1 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

Consider the firing sequence $t_{A0}t_{B0}$. It entails:

$$M_{deadlock} = M_0 + A \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \\ -1 \\ -1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

As expected, we find again the blocking marking from the reachability graph of question a).

- c) A deadlock state is a state at which no transition is enabled. Hence, one can use the upstream transition matrix W^- to assess whether or not a marking is blocking. It is the case if and only if the marking vector does not **cover** (i.e., is bigger or equal to) any column of W^- . Otherwise, it implies the transition associated to such column is enabled, and therefore this marking is not blocking.
- d) In order to avoid such deadlock, it suffices to forbid both process to run concurrently. This can be solved easily using a semaphore, as illustrated thereafter:

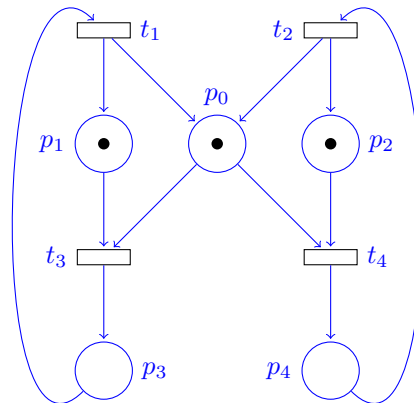


4 From mutual exclusion to starvation

In this exercise, your task is to design a Petri net that models a system in which two processes (A and B) want to access a common exclusive resource, similarly as in Exercise 3. This means that the two processes have to exclude each other mutually from the concurrent access to the resource (e.g. a critical program section). In computer science, this concept is often called a *mutex*. Here is the desired behavior of the system:

1. One process executes its program.
 2. In order to enter the critical section, the mutex value must be 1 (i.e. the mutex is available).
 3. If this is the case, the process sets the mutex to 0 and executes its critical section.
 4. When it is done, it resets the mutex to 1 and enters an uncritical section.
 5. It loops back to start.
- a) Propose a Petri net representing the desired behavior.
Hint Use 5 places and 4 transitions.
 - b) In this setting, it may happen that one process starves the other. That means one process always uses the resource and the other never enters its critical section (e.g. this may happen when using fixed-priority scheduling). Adapt your model such that process A cannot execute more than two times more than B . This may result in having only one process able to start running from the initial marking.
 - c) What is the problem of such approach? What would you suggest as a specification for the scheduling of processes A and B ?
 - d) **Bonus** You can try to express your specification using a Petri Net.

- a) For each process we introduce two places (p_1, p_2, p_3 and p_4) representing the process within the normal program execution (p_1, p_2) as well as in the critical section (p_3, p_4). For each process, we have a token indicating which section of the program is currently executed. Additionally, we introduce a place p_0 representing the mutex variable. If the mutex variable is 0, then we have a token at p_0 . We have to make sure that a process can only enter its critical section if there is a token at the mutex place. The resulting Petri net looks as follows.

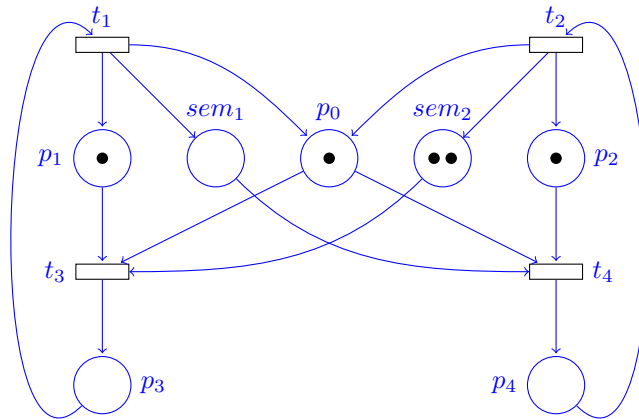


Assume that initially, both processes are in an non-critical section (in the Petri net, this is denoted by a token in place p_1 and p_2 respectively). A process can only enter its critical section (p_3/p_4) if there is a token at p_0 . In this case, the token is consumed when entering the critical section. A new mutex token at p_0 is not created until the process leaves its critical section. Hence, both processes exclude each other mutually from the concurrent access to the critical section.

This is a classical benefit of Petri nets over other DES models. It models very efficiently the sharing of resources, the concurrency of processes, and so on...

- b) In order to avoid starvation of either of the process, one option is to count the number of execution the each of them, or more precisely the difference between them. Assume that at initial state, none has been previously run. According to the specification, we can allow one to the process (say A) to run twice by creating a "counter-resource" with 2 tokens in the initial marking. Running the process A consumes one of these tokens. A new token is produced in this place on completion of process B . Doing that symmetrically (for each process) binds the number of executions of each process together...

Not so clear? Okay, just have a look to the net :-)

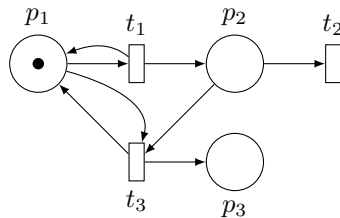


- c) With such mechanism, A will not starve B . However, if for some reason, B does not execute anymore, then A will have to stop as well once it has two executions more than B . This would be a pretty bad design in most cases.

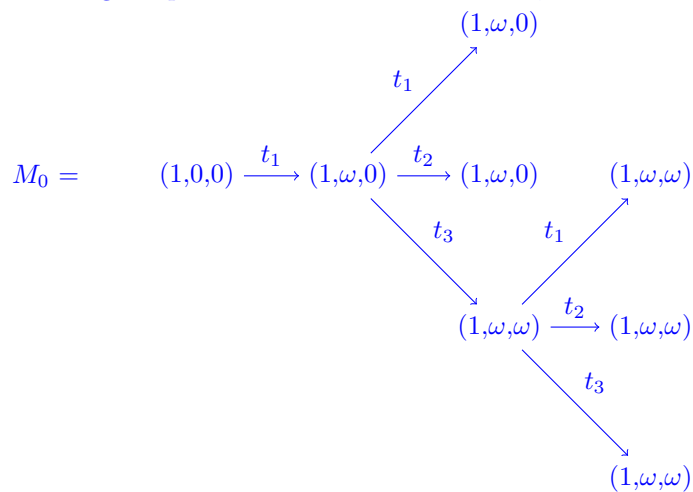
The naive idea would be to say that "if both processes want to access the resource, they get it in turns".

5 Coverability tree and graph

Given is the following Petri net N_5 , compute its coverability tree and coverability graph. Deduce which are the unbounded places of this net given the initial marking.



Following the procedure from the lecture note, we can construct the following coverability tree:



One can merge the equivalent node and obtain the coverability graph:

$$M_0 = (1,0,0) \xrightarrow{t_1} \underbrace{(1,\omega,0)}_{t_2, t_3} \xrightarrow{t_3} \underbrace{(1,\omega,\omega)}_{t_1, t_2, t_3}$$

It follows that for this net with initial marking $(1, 0, 0)$, places p_2 and p_3 are unbounded.

6 Reachability Analysis for Petri Nets

In the lecture we presented an algorithm to perform a reachability analysis on Petri nets.

- Why is it not possible with a reachability algorithm to determine *in general*, whether a given state in a Petri net is reachable or not? To which extend the notion of *coverability* helps?
- Consider the Petri net N_2 from exercise 2. Is the state $s = (p_1 = 101, p_2 = 99, p_3 = 4)$ reachable from the initial state $s_0 = (1, 0, 0)$ if $k = 2$? Prove your answer, either by giving a suitable firing sequence, or a justification for non-reachability.

Hint Start with incidence matrix of N_2 and express the necessary condition (presented in the lecture) for the reachability of the marking s from s_0 . Eventually, discuss whether or not the marking is reachable.

- Petri nets may possess infinite reachability graphs, e.g. N_2 with $k \geq 2$. If a marking is actually reachable in such a Petri net, the reachability check will eventually terminate. But if it is not reachable, the algorithm may not be able to determine reachability with absolute certainty (cf. halting problem).

Constructing a coverability tree or graph is guaranteed to terminate. It can be used to prove that a given marking is not reachable, in the case where the marking you are interested is **not covered by any** marking in the coverability tree/graph. However, this is not a sufficient to prove reachability in the general case: a marking may be covered by the coverability graph, and yet not being reachable.

- We determine the incidence matrix of the Petri net as explained in the lecture.

$$\mathbf{A} = \begin{pmatrix} -1 & 1 & 0 & 2 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

We are interested in whether the state $\vec{s} = (101, 99, 4)$ is reachable from the initial state $\vec{s}_0 = (1, 0, 0)$. If the equation system $\mathbf{A} \cdot \vec{f} = \vec{s} - \vec{s}_0$ has no solution, we know for sure that the state \vec{s} is not reachable from s_0 . "Unfortunately",

$$\begin{pmatrix} -1 & 1 & 0 & 2 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} = \begin{pmatrix} 100 \\ 99 \\ 4 \end{pmatrix}$$

is satisfiable. Using linear algebra, the solutions to this system can be computed (here, $f_1 = Q, f_2 = Q - 306, f_3 = 207, f_4 = 203$, for any $Q \in \mathbb{N}$). If \vec{s} is reachable from \vec{s}_0 , the firing sequence will be of this form. However, there is no guarantee that it is actually feasible for the net! Ultimately, one has to look at the net and propose a suitable firing sequence (although the solution to the previous system of equations gives us the "shape" of the firing sequence we are looking for).

So, to prove that \vec{s} is reachable from \vec{s}_0 , we have to give a firing sequence through which we get from \vec{s}_0 to \vec{s} . Considering the Petri net, we can see that – starting from \vec{s}_0 – the number of tokens in p_1 increases by one after firing the sequence t_1, t_3, t_4 . Repeating this for 203 times yields the state $(204, 0, 0)$. Firing t_1 for 103 more times, followed by firing t_3 for four times finally yields state \vec{s} .