Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2014 Prof. R. Wattenhofer / K.-T. Foerster, T. Langner, J. Seidel

Discrete Event Systems

Solution to Exercise Sheet 15

1 Structural Properties of Petri Nets and Token Game

a) The pre and post sets of a transition are defined as follows:

- pre set: $\bullet t := \{p \mid (p,t) \in C\}$
- post set: $t\bullet := \{p \mid (t,p) \in C\}$,

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, the one on p_2 is consumed. After this, 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 on p_2 and p_5 each. Directly afterwards, there are tokens on places p_3, p_9 und p_5.

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 can traverse the lower cycle until t_8 is enabled. One token now remains on p_5 and waits, another one enables t_3, because there is still one token on p_3. Now one token traverses the upper cycle again until t_2 is enabled, and so on.

Hence, this petri net models two processes which always appear alternately.

The reachability graph $RG(P, \vec{s}_0)$ of a petri net P is a quadruple $(\mathbb{S}, \mathbb{S}_0, Act, \mathbb{E})$ such that

- $\mathbb{S}$ is the set of reachable states of P starting from $\vec{s}_0$
- $\mathbb{S}_0 := \{\vec{s}_0\}$ is the start state of P
- Act is the set of transition labels
- $\mathbb{E} \subseteq \mathbb{S} \times Act \times \mathbb{S}$ is the set of edges such that $\mathbb{E} = \{(\vec{s}, t, \delta(\vec{s}, t)) \mid \vec{s} \in \mathbb{S} \land t \in T \land \vec{s} \triangleright t\}$

Usually the states of the petri net 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. 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 $\vec{s}_0 = (1,0,0,0,1,0,0,0,0,0)$. Hence, the reachability graph looks as follows:

$$
\begin{aligned}
\mathbb{S} = \{ \quad & (1,0,0,0,1,0,0,0,0,0), (0,1,0,0,1,0,0,0,0,0), (0,0,1,0,1,0,0,0,1,0), \\
& (0,0,1,0,0,1,0,0,0,0), (0,0,1,0,0,0,1,0,0,0), (0,0,1,0,0,0,0,1,0,0), \\
& (0,0,1,0,1,0,0,0,0,1), (0,0,0,1,1,0,0,0,0,0) \quad \}, \\[1em]
\mathbb{S}_0 = \{ \quad & (1,0,0,0,1,0,0,0,0,0) \quad \}, \\[1em]
Act = \{ \quad & t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8 \quad \}, \\[1em]
\mathbb{E} = \{ \quad & \big((1,0,0,0,1,0,0,0,0,0), t_1, (0,1,0,0,1,0,0,0,0,0)\big), \\
& \big((0,1,0,0,1,0,0,0,0,0), t_2, (0,0,1,0,1,0,0,0,1,0)\big), \\
& \big((0,0,1,0,1,0,0,0,1,0), t_5, (0,0,1,0,0,1,0,0,0,0)\big), \\
& \big((0,0,1,0,0,1,0,0,0,0), t_6, (0,0,1,0,0,0,1,0,0,0)\big), \\
& \big((0,0,1,0,0,0,1,0,0,0), t_7, (0,0,1,0,0,0,0,1,0,0)\big), \\
& \big((0,0,1,0,0,0,0,1,0,0), t_8, (0,0,1,0,1,0,0,0,0,1)\big), \\
& \big((0,0,1,0,1,0,0,0,0,1), t_3, (0,0,0,1,1,0,0,0,0,0)\big), \\
& \big((0,0,0,1,1,0,0,0,0,0), t_4, (1,0,0,0,1,0,0,0,0,0)\big) \quad \}.
\end{aligned}
$$

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 $\vec{s}_0 = (1,0,0,0,1,0,0,0,0,0) = s_{1,5}$.

Then the reachability graph can also be specified as follows:

$$
\begin{tikzcd}
\arrow[r] & s_{1,5} \arrow[r, "t_1"] & s_{2,5} \arrow[r, "t_2"] & s_{3,5,9} \arrow[r, "t_5"] & s_{3,6} \arrow[r, "t_6"] & s_{3,7} \arrow[r, "t_7"] & s_{3,8} \arrow[r, "t_8"] & s_{3,5,10} \arrow[r, "t_3"] & s_{4,5} \arrow[lllllll, bend left, "t_4"]
\end{tikzcd}
$$

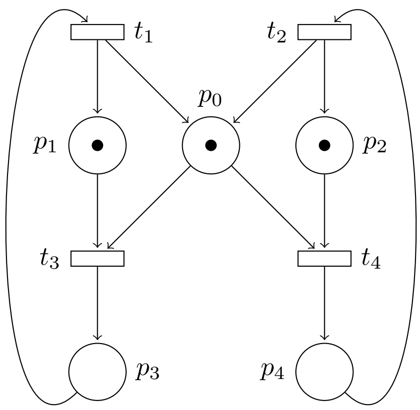
2 Basic Properties of Petri Nets

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 Mutual Exclusion

For each process we introduce two places (p_1, p_2, p_3 und 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 currently is 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 uncritical 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.