1 The Winter Train Problem

We can model each train individually and combine the corresponding sub-states using an AND-super-state, see the figure below. Additionally, in order to “synchronize” the trains, a third sub-state is needed (shown in the middle) which implements a mutual exclusion: For instance, if there is no train between Stans and Engelberg and if train 1 is in state c1, T1 can enter the critical section and train 2 has to wait. (Notice that if both trains are in states c1 and c2 respectively, T1 has priority.)

- The trains start at their states m1 and m2. When m1 (m2) is pressed, then train 1 (2) moves to the right in n1 (n2), until it reaches the switch, where it stops in state o1 (o2).

- Now the ”middle”-state can change its state to either y or z, depending on which train got there first. If train 1 (2) arrives first, then the state is changed to y (z) and train 1 (2) can move to state p1 (p2) while moving right.
After arriving at the station Engelberg, the train waits for 100s, then moves to the left and switches to state q1 (q2) – until it hits the switch at b1 (b0), upon which the "middle"-state can change again – and the train continues to its original station, where it stops.

Positions of the trains (train 1 ; train 2):

- m1: Lucerne ; m2: Sarnen
- n1: Between Lucerne and the switch ; n2: Between Sarnen and the switch
- o1: At the left side of the switch ; o2: At the left side of the switch
- p1: Between the switch and Engelberg ; p2: Between the switch and Engelberg
- q1: Between Engelberg and the switch ; q2: Between Engelberg and the switch
- r1: Between the switch and Lucerne ; Between the switch and Sarnen

2 Structural Properties of Petri Nets and Token Game

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

- pre set: \[ t := \{ p \mid (p, t) \in C \} \]
- post set: \[ t• := \{ p \mid (t, p) \in C \}, \]

the pre and post sets of a place are defined analogously.

For the petri net \( N_1 \) we obtain the following sets:

- \( t_5 = \{ p_5, p_9 \} \), \( t_5• = \{ p_6 \} \)
- \( t_8 = \{ p_8 \} \), \( t_8• = \{ p_{10}, p_5 \} \)
- \( p_3 = \{ t_2 \} \), \( p_3• = \{ t_3 \} \)

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 \( (S, S_0, Act, E) \) such that

- \( S \) is the set of reachable states of \( P \) starting from \( \vec{s}_0 \)
- \( S_0 := \{ \vec{s}_0 \} \) is the start state of \( P \)
- \( Act \) is the set of transition labels
- \( E \subseteq S \times Act \times S \) is the set of edges such that \( E = \{ (\vec{s}, t, \delta(\vec{s}, t)) \mid \vec{s} \in S \land t \in T \land \vec{s} \rightarrow t \} \)

2
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:

$$S = \{ (1, 0, 0, 0, 1, 0, 0, 0, 0, 0), (0, 1, 0, 0, 1, 0, 0, 0, 0, 0), (0, 0, 1, 0, 1, 0, 0, 0, 0, 0),$$
$$ (0, 0, 1, 0, 0, 1, 0, 0, 0, 0), (0, 0, 1, 0, 0, 0, 1, 0, 0, 0), (0, 0, 1, 0, 0, 0, 1, 0, 0, 0),$$
$$ (0, 0, 1, 0, 0, 0, 0, 1), (0, 0, 0, 1, 0, 0, 0, 0, 0) \}.$$

$$S_0 = \{ (1, 0, 0, 0, 1, 0, 0, 0, 0, 0) \}.$$

$$\text{Act} = \{ t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10} \}.$$

$$\mathcal{E} = \{ ((1, 0, 0, 0, 1, 0, 0, 0, 0, 0), t_1, (0, 1, 0, 0, 1, 0, 0, 0, 0, 0)),$$
$$ ((0, 1, 0, 0, 1, 0, 0, 0, 0, 0), t_2, (0, 0, 1, 0, 1, 0, 0, 0, 1, 0)),$$
$$ ((0, 0, 1, 0, 1, 0, 0, 0, 1, 0), t_5, (0, 0, 1, 0, 1, 0, 0, 0, 0, 0)),$$
$$ ((0, 0, 1, 0, 0, 0, 0, 0, 0, 0), t_6, (0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0)),$$
$$ ((0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0), t_7, (0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0)),$$
$$ ((0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0), t_8, (0, 0, 1, 0, 1, 0, 0, 0, 0, 1)),$$
$$ ((0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0), t_9, (0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0)),$$
$$ ((0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0), t_{10}, (1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0)) \}.$$

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 $\vec{s}_0 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0) = s_{1.5}$.

Then the reachability graph can also be specified as follows:

3 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.

4 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.