1 Labelled Graphs

In the lecture we presented an algorithm that checks the reachability of a state in a directed graph. Given a deterministic LTS $L$ and a word $\omega := w_1w_2\ldots w_n$, modify the algorithm from the lecture such that it checks whether $L$ accepts the word $\omega$.

2 Structural Properties of Petri Nets and Token Game

Given is the following petri net $N_1$:

![Petri Net Diagram]

a) What are the pre and post sets of transitions $t_5$ and $t_8$ and of place $p_3$?

b) Which transitions are enabled after $t_1$ and $t_2$ fired?

c) Determine the 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 $s_0 = (1, 0, 0, 0, 1, 0, 0, 0, 0, 0) =: s_{1,5}$. 
3 Basic Properties of Petri Nets

Given is the following petri net $N_2$:

![Petri Net Diagram]

a) Explain the terms boundedness and deadlock-freeness using this example.

b) For which values of $k \in \mathbb{N}$ is the petri net $N_2$ bounded/unbounded and not deadlock-free?

4 Reachability Analysis for Petri Nets

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

a) 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?

b) Consider the petri net $N_2$ from exercise 3. 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.

Note: Use the condition presented in the lecture that is sufficient for the reachability of a state in a weighted petri net.

5 Mutual Exclusion

Your task is to model a system as a petri net in which two processes want to access a common exclusive resource. 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). More concrete, this means:

1. A process executes its program.

2. In order to enter the critical section, a given mutex variable must be 0.

3. If this is the case, the process sets the mutex to 1 and executes its critical section.

4. When done, it resets the mutex to 0 and enters an uncritical section.

5. Then the procedure starts all over again.