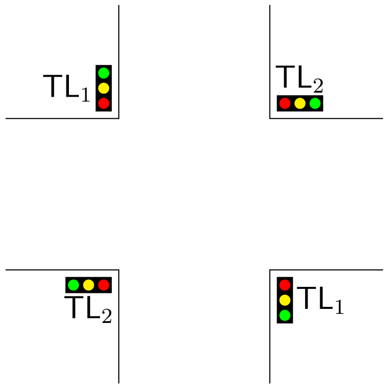
Prof. Dr. Andreas Podelski
Tanja Schindler

Hand in until November 17th, 2017
11:59 via the post boxes
Discussion: November 20th, 2017

Tutorial for Cyber-Physical Systems - Discrete Models
Exercise Sheet 4

Exercise 1: Traffic lights

Consider the crossing of two roads with four traffic lights as depicted on the right. The two traffic lights labelled with TL_1 always show the same color, and likewise the two traffic lights labelled with TL_2 always show the same color. The traffic lights have three modes: `red`, `yellow`, and `green`, and they switch from `green` to `yellow`, from `yellow` to `red`, and from `red` to `green`.



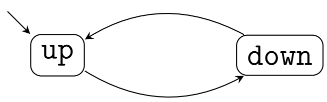
(a) Create two transition systems TS_1 and TS_2 for the traffic lights, one for each direction of a crossing.

Insert suitable actions on which these system can synchronize so that at least one of the lights are in the `red` mode in each state of the transition system $TS_1 \| TS_2$.

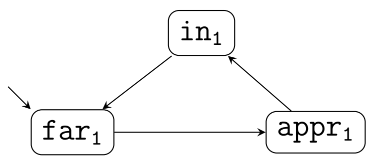
(b) Compute the transition system $TS_1 \| TS_2$. Is the system safe? An informal argument is sufficient.

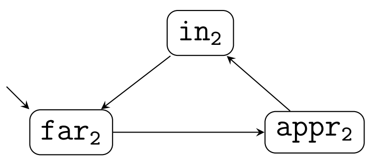
Exercise 2: Railroad crossing controller

In this exercise we build a model for the controller of a railroad crossing. Our railroad crossing has one gate and two train tracks, one track for each direction.
The transition system of the gate has two states and the following graph structure.



The transition systems of the train tracks have three states. In the first state all trains on this track are far away, in the second state one train is approaching, in the third state one train is in the railroad crossing and no other train is approaching on this track. The transition systems of the train tracks have the following graph structure.

Track 1:


Track 2:


Describe a controller (in the form of a transition system) that controls the gate such that whenever a train is in the railroad crossing (state $\texttt{in}_1$ or state $\texttt{in}_2$),the gate is down (state `down`). Your controller may temporarily stop a train in the sense that a train may only move from $\texttt{appr}_i$ to $\texttt{in}_i$ if the controller agrees.
Complete the transition system descriptions of train tracks and gate by adding suitable actions to the graphs given above.
The system should have the property that every train can pass the gate eventually and that the gate is not always down. Hence, e.g., the trivial controller that just stops every train or the trivial controller the keeps the gate down are not valid solutions here.

Exercise 3: Parallel program
We are given three (primitive) processes P_1, P_2, and P_3 with shared integer variable `x`. Process P_i executes ten times the assignment `x++`, which is realized using the three actions `LOAD(x)`, `INC(x)`, and `STORE(x)`. See the following pseudocode:

Algorithm 1: Process P_i

Data: `x` (global)
for $i := 1$ **to** 10 **do**
    `LOAD(x);`
    `INC(x);`
    `STORE(x);`
end

Consider now the following parallel program P:

Algorithm 2: Parallel program P

Data: `x` (global)
$\texttt{x} := 0$;
$P_1 \| P_2 \| P_3$;

(a) Does P have an execution that halts with the terminal value $\texttt{x} = 2$?

(b) Does P have an execution that halts with the terminal value $\texttt{x} = 11$?

Exercise 4: Regular expressions
We use the convention that we omit parentheses in expressions if the meaning is clear.

(a) Consider the alphabet $\Sigma = \{a, b\}$. Describe in natural language what each of the following regular expressions means and find a different regular expression that describes the same language.

(i) $\emptyset a \emptyset$

(ii) $(a^* b^*)^*$

(iii) ε^{**}

(iv) a^{++}

(v) $a(ba)^* + (ab)^*a$

(b) Construct regular expressions for the following languages over the alphabet $\Sigma = \{a, b\}$.

(i) $L_1 = \{w \in \Sigma^* \mid \text{every } a \text{ in } w \text{ is immediately followed by } b\}$

(ii) $L_2 = \{w \in \Sigma^* \mid w \text{ does not contain } bb\}$

(iii) $L_3 = \{w \in \Sigma^* \mid w \text{ contains at least two } a\}$

(iv) $L_4 = \{w \in \Sigma^* \mid w \text{ contains at most two } a\}$

(c) (Trick question) Can you use the definitions in Sect. A.2 to derive what language is denoted by the regular expression $\emptyset^*$?