Prof. Dr. Andreas Podelski
Dominik Klumpp

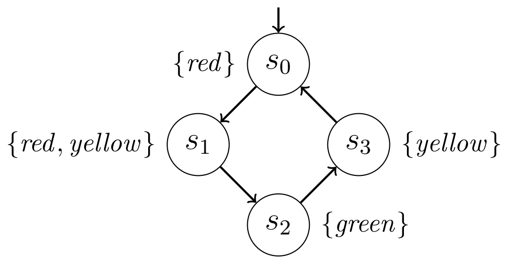
Hand in until February 5th, 2020
15:59 via the post boxes
Discussion: February 10th, 2020

Tutorial for Cyber-Physical Systems - Discrete Models
Exercise Sheet 14

Exercise 1: Traffic Light — 12 Points

In this exercise, we arrive at the goal towards which we have worked the whole semester: For a cyber-physical system (given as a transition system) and desired correctness properties, we are able to determine if the system satisfies these properties.

The following transition system T models the behaviour of a traffic light.



(a) Draw an NFA $\mathcal{A}_T$ over the alphabet $\Sigma = 2^{AP}$ with $AP = \{red, yellow, green\}$ such that $\mathcal{A}_T$ accepts exactly the finite prefixes of $Traces(T)$, i.e., $\mathcal{L}(\mathcal{A}_T) = pref(Traces(T))$.

(b) Consider the following safety properties:

(P_1) *"It is always the case that if the green light is on, then the red light will be off in the next step."*

(P_2) *"It is always the case that if the red light is on, then the green light will be off in the next step."*

Formalize these properties as sets of traces over the set of atomic propositions $AP = \{red, yellow, green\}$, in the style $\{A_0A_1A_2 \ldots \mid \forall i \ldots\}$.

(c) Give automata for the bad prefixes of these properties. I.e., draw NFAs $\mathcal{A}_P$ such that $\mathcal{A}_P$ accepts exactly the bad prefixes of the property P (for $P = P_1$ resp. $P = P_2$).

Draw the automata in symbolic notation, i.e., with propositional formulas as edge labels.

(d) For each of the two properties P, execute the algorithm below to check whether the bad prefix automaton $\mathcal{A}_P$ is disjoint from the automaton $\mathcal{A}_T$, i.e., whether $\mathcal{L}(\mathcal{A}_T) \cap \mathcal{L}(\mathcal{A}_{P_i}) = \{\}$. If they are not disjoint, give a finite prefix that is accepted by both automata.

Algorithm: Convert the bad prefix automaton from symbolic notation to standard notation, i.e., such that each edge is labeled with a set of atomic propositions. Then construct the parallel composition $\mathcal{A}_P \parallel \mathcal{A}_T$. As before, the edge labels are used for synchronization (handshaking). The only difference is that in this case, the edge labels are letters in $\Sigma = 2^{AP}$.

The states of the resulting system are pairs $\langle q_1, q_2 \rangle$ where q_1 is a state of $\mathcal{A}_P$ and q_2 is a state of $\mathcal{A}_T$. If there exists a reachable state $\langle q_1, q_2 \rangle$ such that both q_1 and q_2 are accepting states, then the automata are not disjoint and T violates the property P. If no such state exists, then T satisfies the property P.

Exercise 2: Regular Expressions 3 Points + 1 Bonus Points

Let $AP = \{a, b, c\}$ be a set of atomic propositions. We now consider regular expressions (for sets of finite words) over the alphabet $\Sigma = 2^{AP}$. Answer the questions below.

Reminder: Remember that $\emptyset$ denotes the empty set of atomic propositions (a letter of the alphabet Σ), while $\varnothing$ is a regular expression denoting the empty set of finite words (a set of words over the alphabet Σ).

(a) The regular expression $\emptyset + \{b\} + \{a, b, c\}$ denotes a set of finite words. How many finite words are in this set? What is the length of each of these finite words?

(b) As described in the lecture, we can use a propositional formula Φ to abbreviate the regular expression $(A_1 + A_2 + \ldots + A_n)$, where $\{A_1, A_2, \ldots, A_n\} = \{A \subseteq AP \mid A \models \Phi\}$.

For each of the regular expressions below, list all finite words in the denoted set.

(i) $(a \wedge \neg b) + (a \wedge c)$ (ii) $(a \wedge \neg b) \vee (a \wedge c)$ (iii) $\mathit{false} \, . \, (\neg a)$

(c) We introduced the regular expression $\varnothing$, which denotes the empty set of finite words. If we allow propositional formulas in regular expressions (as above), do we really need the symbol $\varnothing$? Why / why not?

Similarly, do we need the symbol $+$? Why / why not?

Exercise 3*: Properties given by a Transition System 4 Bonus Points

We have learned to distinguish safety and liveness properties. It was discussed in the lecture that the set $\mathit{Traces}(T)$ of a transition system T is itself a property.

(a) For an arbitrary transition system T, is the property $\mathit{Traces}(T)$ always a liveness property? If so, argue why this is the case. If not, give a counterexample.

(b) For an arbitrary transition system T, is the property $\mathit{Traces}(T)$ always a safety property? If so, argue why this is the case. If not, give a counterexample.

Hint: Consider for instance the program `x:=nondet(); while (x > 0){ x--; }` and the set of atomic propositions $AP = \{x \leq 0\}$. Here, the procedure `nondet()` nondeterministically chooses and returns an arbitrary integer.