Prof. Dr. Andreas Podelski
Tanja Schindler

Hand in until January 12, 2018
11:59 via the post boxes
Discussion: January 15, 2018

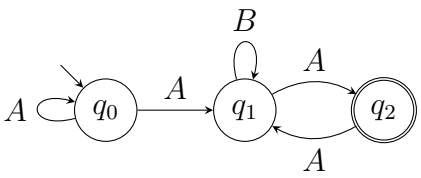
Tutorial for Cyber-Physical Systems - Discrete Models
Exercise Sheet 10

The overall goal of this sheet is to understand Nondeterministic Finite Automata (NFA). These NFAs can be used to verify a certain class of linear-time properties, namely regular safety properties. That is, checking regular safety properties can be reduced to invariant checking with the help of an NFA that accepts the bad prefixes. (Later we will see how ω-regular liveness properties can be verified by using a different class of automata.)
We have already studied algorithms for invariant checking (see Sheet 7). This sheet aims at providing a deeper understanding of how NFAs can be used to expand the domain of application of these algorithms to checking a more general class of properties.

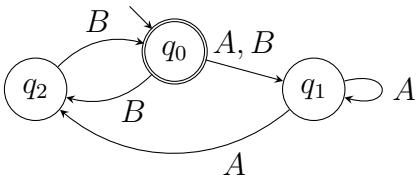
Exercise 1: NFA $\rightarrow$ regular expression

Consider the following NFA over the alphabet $\Sigma = \{A, B\}$, which are given in a graphical representation where accepting states have double circles.

(a)



(b)



Write for each NFA a regular expression such that the language described by the regular expression is the language that is accepted by the NFA.

Motivation: This exercise serves as a preparation for applying NFAs to verify safety properties. It aims at learning how to read NFAs, i.e., learning how to determine the corresponding regular language (NFAs and regular expressions are just two different ways to represent regular languages).

Exercise 2: Regular expression $\rightarrow$ NFA

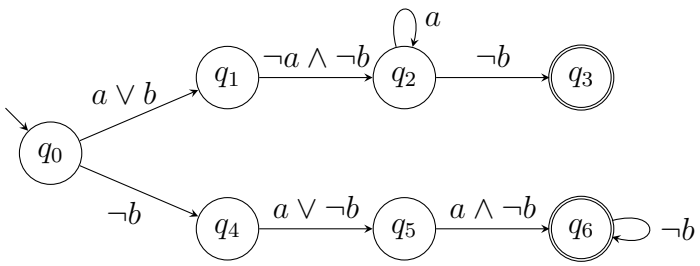
Consider the following regular expressions over the alphabet $\Sigma = \{A, B\}$.

(a) $(AB + B^*)ABA^*A$ (b) $((ABAB)^* + AB)^*AB(B^* + \emptyset A)$

Construct for each regular expression an NFA such that the language accepted by the NFA is the language that is described by the regular expression. You may give your NFA as a five tuple or use the graphical representation.

Motivation: This exercise complements the first exercise. For verification purposes, one needs to be able to construct NFAs that accept a certain language (given, for instance, as a regular expression).

Exercise 3: Symbolic automata
Consider the following symbolic NFA $\mathcal{A}$ over the alphabet 2^{AP} with $AP = \{a, b\}$.



Which of the following words is accepted by $\mathcal{A}$? Give a short explanation.

(a) $w_1 = \{a\}\,\{\}\,\{a\}\,\{b\}$
(b) $w_2 = \{a\}\,\{\}\,\{a\}$
(c) $w_3 = \{b\}\,\{\}\,\{a, b\}\,\{a\}\,\{a\}$
(d) $w_4 = \{a\}\,\{a\}\,\{a\}\,\{\}\,\{a\}$

Motivation: Symbolic notation in terms of a propositional logic formula is often more convenient than enumerating sets of propositions that satisfy the corresponding formula. One goal of this exercise is to learn how to switch between the two representations. Another goal is, as in Exercise 1, to learn how to read (symbolic) NFAs.

Exercise 4: Regular safety properties
Let $AP = \{a, b, c\}$. Consider the following regular safety properties:

(a) P_1: If a becomes valid, afterward b stays valid ad infinitum or until c holds.

(b) P_2: Between two neighbouring occurrences of a, b always holds.

Construct an NFA $\mathcal{A}_i$ for each property P_i such that $L(\mathcal{A}_i) = \text{BadPref}(P_i)$.

Hint: You may use a symbolic NFA with propositional formulae over the set AP as transition labels.

Motivation: The first step towards verifying a regular safety property P is to construct the NFA that accepts the set of bad prefixes of P. The goal of this exercise is to learn to identify the set of bad prefixes of P (a set of finite words) and then "translate" this set into an NFA. Note that it is not possible to construct such an automaton for arbitrary safety properties (why?).