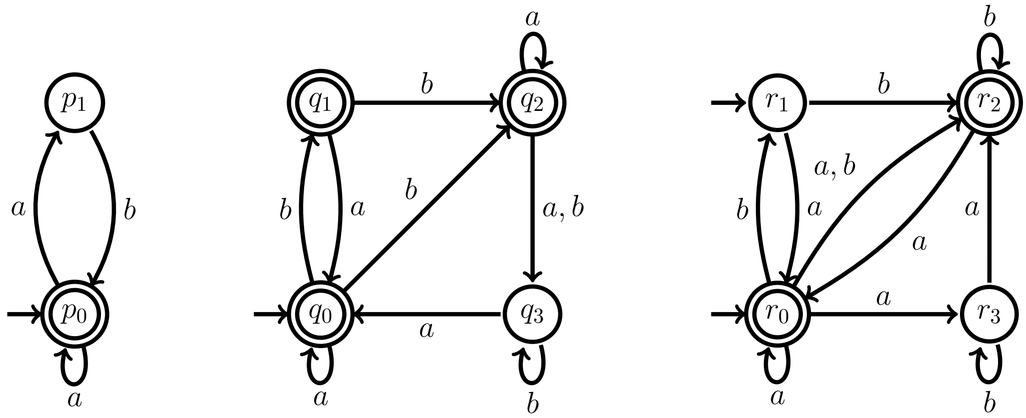
Prof. Javier Esparza
Philipp Czerner
Technical University of Munich
Chair for Foundations of Software Reliability

Automata and Formal Languages

Winter Term 2023/24 – Exercise Sheet 5

Exercise 5.1.

Consider the following NFAs A, B and C:



(a) Use algorithm *UnivNFA* to determine whether $L(B) = \{a, b\}^*$ and $L(C) = \{a, b\}^*$.

(b) For $D \in \{B, C\}$, if $L(D) \neq \{a, b\}^*$, use algorithm *InclNFA* to determine whether $L(A) \subseteq L(D)$.

Exercise 5.2.

Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. For any $S \subseteq Q$, a word $w \in \Sigma^*$ is said to be a *synchronizing word* for S in A if reading w from any state of S leads to a common state, i.e., if there exists $q \in Q$ such that **for every** $\mathbf{p} \in \mathbf{S}$, $p \xrightarrow{w} q$. We now define the *synchronizing word* problem defined as follows:

Given: DFA A and a subset S of the states of A
Decide: If there exists a synchronizing word for S in A

(a) Given states $p, q \in Q$, design a polynomial time algorithm for testing if there is a synchronizing word for $\{p, q\}$ in A.

(b) Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Show that there is a synchronizing word for Q in A if and only if for every $p, q \in Q$, there is a synchronizing word for $\{p, q\}$ in A.

By (a) and (b), we can conclude that there is a polynomial time algorithm for the special case of the synchronizing word problem where the subset S is the set of all states of A. However, for the general case, we have the following result.

(c) [hard] Show that the synchronizing word problem is PSPACE-hard. You may assume that the following problem, called the *DFA intersection problem* is PSPACE-hard:

Given: DFAs $A_1, A_2, ..., A_n$ all over a common alphabet Σ
Decide: If there exists a word w such that $w \in \bigcap_{1 \leq i \leq n} \mathcal{L}(A_i)$

Exercise 5.3.

Let Σ be a finite alphabet and let $L \subseteq \Sigma^*$ be a language accepted by an NFA A. Give an NFA-ε for each of the following languages:

(a) $\sqrt{L} = \{w \in \Sigma^* \mid ww \in L\}$,

(b) [hard] $\mathrm{Cyc}(L) = \{vu \in \Sigma^* \mid uv \in L\}$.

Puzzle exercise 5.4.

Let $M = (Q, \Sigma, \delta, q_0, F)$ denote a DFA. We introduce a new type of finite automaton, which we call *flipping DFAs*. Syntactically, they are the same as DFAs, but they read words differently: after reading a letter $a \in \Sigma$ while in a state q, the automaton moves to state $q' := \delta(q, a)$ and then modifies δ by swapping the incoming transitions of states q and q' (i.e. every transition that previously pointed to q now points to q' and vice versa). It accepts if in a final state.

Prove or disprove that every language accepted by a flipping DFA is regular.

Notes: The puzzle exercises cover advanced material and are not directly relevant to the exam. No model solutions will be provided. You can get feedback on your solution by sending it to me (Zulip or mail), or coming to my office in person (MI 03.11.037).