Prof. Javier Esparza
Philipp Czerner

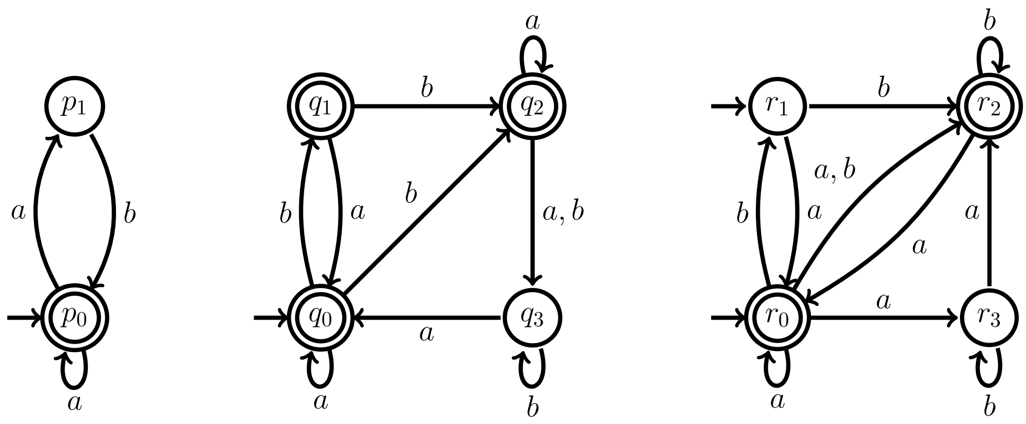
Technical University of Munich
Chair for Foundations of Software Reliability

# 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 $L(A) \subseteq L(D)$.

*Solution.*

(a) The trace of the execution for NFA $B$ is as follows:

| **Iter.** | $\mathcal{Q}$ | $\mathcal{W}$ |
|---|---|---|
| 0 | $\emptyset$ | $\{\{q_0\}\}$ |
| 1 | $\{\{q_0\}\}$ | $\{\{q_1, q_2\}\}$ |
| 2 | $\{\{q_0\}, \{q_1, q_2\}\}$ | $\{\{q_2, q_3\}\}$ |
| 3 | $\{\{q_0\}, \{q_1, q_2\}, \{q_2, q_3\}\}$ | $\{\{q_3\}\}$ |

At the fourth iteration, the algorithm encounters state $\{q_3\}$ which is non final, and hence it returns *false*. Therefore, $L(B) \neq \{a, b\}^*$.

The trace of the execution for NFA $C$ is as follows:

| **Iter.** | $\mathcal{Q}$ | $\mathcal{W}$ |
|---|---|---|
| 0 | $\emptyset$ | $\{\{r_0, r_1\}\}$ |
| 1 | $\{\{r_0, r_1\}\}$ | $\{\{r_0, r_2, r_3\}, \{r_1, r_2\}\}$ |
| 2 | $\{\{r_0, r_1\}, \{r_0, r_2, r_3\}\}$ | $\{\{r_1, r_2\}\}$ |
| 3 | $\{\{r_0, r_1\}, \{r_0, r_2, r_3\}, \{r_1, r_2\}\}$ | $\{\{r_0\}, \{r_2\}\}$ |
| 3 | $\{\{r_0, r_1\}, \{r_0, r_2, r_3\}, \{r_1, r_2\}, \{r_0\}\}$ | $\{\{r_2\}\}$ |
| 3 | $\{\{r_0, r_1\}, \{r_0, r_2, r_3\}, \{r_1, r_2\}, \{r_0\}, \{r_2\}\}$ | $\emptyset$ |

At the fifth iteration, $\mathcal{W}$ becomes empty and hence the algorithm returns *true*. Therefore $L(C) = \{a, b\}^*$.

(b) The trace of the algorithm for $A$ and $B$ is as follows:

| **Iter.** | $\mathcal{Q}$ | $\mathcal{W}$ |
|---|---|---|
| 0 | $\emptyset$ | $\{[p_0, \{q_0\}]\}$ |
| 1 | $\{[p_0, \{q_0\}]\}$ | $\{[p_1, \{q_0\}]\}$ |
| 2 | $\{[p_0, \{q_0\}], [p_1, \{q_0\}]\}$ | $\{[p_0, \{q_1, q_2\}]\}$ |
| 3 | $\{[p_0, \{q_0\}], [p_1, \{q_0\}], [p_0, \{q_1, q_2\}]\}$ | $\emptyset$ |

At the third iteration, $\mathcal{W}$ becomes empty and hence the algorithm returns *true*. Therefore $L(A) \subseteq L(B)$.

### 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 $\Sigma$
*Decide:* If there exists a word $w$ such that $w \in \bigcap_{1 \leq i \leq n} \mathcal{L}(A_i)$

*Solution.*

(a) By definition, $w$ is a synchronizing word for $\{p, q\}$ in $A$ if and only if there is a state $r$ such that $r = \delta(p, w) = \delta(q, w)$. Consider the following algorithm: For every state $r \in Q$, we construct two DFAs $A_r^p = (Q, \Sigma, \delta, p, r)$ and $A_r^q = (Q, \Sigma, \delta, q, r)$. Notice that $w$ is a synchronizing word for $\{p, q\}$ in $A$ if and only if there exists a state $r$ such that $w \in \mathcal{L}(A_r^p) \cap \mathcal{L}(A_r^q)$. Hence, the polynomial time algorithm to test if there is a synchronizing word for $\{p, q\}$ in $A$ is as follows: For each $r \in Q$, construct the DFAs $A_r^p$ and $A_r^q$ and test if $\mathcal{L}(A_r^p) \cap \mathcal{L}(A_r^q) \neq \emptyset$ by means of the pairing construction and the emptiness check for DFAs. If for at least one state $r$, this test is true, then there is a synchronizing word for $\{p, q\}$ in $A$; otherwise, there is none.

To analyse the running time, note that we are doing at most $|Q|$ pairing constructions and emptiness checks, each of which takes polynomial time. Hence, the overall running time is also a polynomial in the size of the given input.

(b) $(\Rightarrow)$ : Suppose $w$ is a synchronizing word for $Q$ in $A$. Let $p, q \in Q$. By definition of a synchronizing word, $\delta(p, w) = \delta(q, w)$. Hence, $w$ is also a synchronizing word for $\{p, q\}$ in $A$.

$(\Leftarrow)$ : Suppose for every $p, q \in Q$, there is a synchronizing word $w_{p,q}$ for the subset $\{p, q\}$. We now construct a synchronizing word $w_S$ for every subset $S \subseteq Q$, by induction on $|S|$, the size of $S$.

For the base case, note that if $|S| = 1$, then $\epsilon$ is a synchronizing word for $S$. Assume that we have shown that whenever $|S| \leq i$ for some number $i \geq 1$, there is a synchronizing word for $S$. Suppose $S$ is a subset such that $|S| = i+1$. Hence, $|S| \geq 2$ and let $S = \{p_1, p_2, ..., p_{i+1}\}$. By assumption, there is a synchronizing word $w_{p_1,p_2}$ for the subset $\{p_1, p_2\}$. Let $S' = \{\delta(p_1, w_{p_1,p_2}), \delta(p_2, w_{p_1,p_2}), ..., \delta(p_{i+1}, w_{p_1,p_2})\}$. Since $w_{p_1,p_2}$ is a synchronizing word for $\{p_1, p_2\}$, it follows that $|S'| \leq i$. By induction hypothesis, there is a synchronizing word $w_{S'}$ for the subset $S'$. It is then easy to see that the word $w_{p_1,p_2} w_{S'}$ is a synchronizing word for $S$ in $A$. Hence, the induction step is complete.

It then follows that there is a synchronizing word for the set $Q$ in $A$.

(c) We give a polynomial-time reduction from the DFA intersection problem to the synchronizing word problem, which will prove that the latter is PSPACE-hard. Let $A_1, A_2, ..., A_n$ be $n$ DFAs all over a commmon alphabet $\Sigma$ such that each $A_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$. In polynomial time, we have to construct a DFA $B$ and a subset $S$ of the states of $B$ so that

$$S \text{ has a synchronizing word in } B \text{ if and only if } \bigcap_{1 \leq i \leq n} \mathcal{L}(A_i) \neq \emptyset$$

Let us construct $B = (Q_B, \Sigma_B, \delta_B, q_0^B, F_B)$ and $S$ as follows.

- The set $Q_B$ will consist all the states of all the $A_i$'s and in addition, it will have two new states $p$ and $t$. More formally, $Q = \bigcup_{1 \leq i \leq n} Q_i \cup \{p, t\}$ where $p$ and $t$ are two new states.
- The alphabet $\Sigma_B$ will be $\Sigma \cup \{\#\}$ where $\#$ is a fresh letter not present in $\Sigma$.
- The transition function $\delta_B$ will behave in the following way:

- If $q \in Q_i$ for some $i$ and $a \in \Sigma$, then $\delta_B(q, a) = \delta_i(q, a)$. Intuitively, if $q$ is a state of some $A_i$ and $a$ is not #, then the transition function behaves in exactly the same way as $\delta_i$.
- If $q \in F_i$ for some $i$, then $\delta_B(q, \#) = p$. Intuitively, upon reading a # from some accepting state of some $A_i$, we move to $p$.
- If $q \in Q_i \setminus F_i$ for some $i$, then $\delta_B(q, \#) = t$. Intuitively, upon reading a # from some rejecting state of $A_i$, we move to $t$.
- If $q \in \{p, t\}$ and $a \in \Sigma_B$, then $\delta_B(q, a) = q$. Intuitively, the states $p$ and $t$ have a self-loop corresponding to any letter.

* We set $q_0^B$ to be $p$ and $F_B$ to be $\{p\}$.
* Finally we set $S$ to be the subset of states given by $\{q_0^1, q_0^2, ..., q_0^n, p\}$.

Suppose $w \in \bigcap_{1 \le i \le n} \mathcal{L}(A_i)$. By construction, it then follows that $w\#$ is a synchronizing word for $S$ in $B$.

Suppose $w$ is a synchronizing word for $S$ in $B$. By definition of $w$ and by construction of $B$, it follows that

$$\delta_B(q_0^1, w) = \delta_B(q_0^2, w) = ... = \delta_B(q_0^n, w) = \delta_B(p, w) = p$$

Notice that to move from the state $q_0^1$ to $p$, it is necessary to read a # at some point. Hence, $w$ must contain an occurrence of #. Split $w$ as $w'\#w''$ so that $w'$ has no occurrences of #. For each $i$, let $q_i = \delta_B(q_0^i, w')$. By construction of $B$, it follows that for each $i$, $q_i \in Q_i$. Suppose for some $i$, $q_i \notin F_i$. It then follows that $\delta_B(q_i, \#w'') = t$, which contradicts the fact that $\delta_B(q_0^i, w\#w'') = p$. Hence, $q_i \in F_i$ for every $i$ and this implies that $w'$ is a word which is accepted by all of the $A_i$'s.

**Exercise 5.3.**

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

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

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

*Solution.* Let $A = (Q, \Sigma, \delta, Q_0, F)$ be an NFA that accepts $L$. Without loss of generality, we can assume that $Q_0 = \{q_0\}$ and $F = \{q_f\}$ for some states $q_0$ and $q_f$.

(a) To begin with we have the following observation:

$w \in \sqrt{L}$ if and only if there exists a state $p \in Q$ such that $p \in \delta(q_0, w)$ and $q_f \in \delta(p, w)$.

With this observation in mind, let us do the following construction: For every state $p \in Q$, construct two NFAs $A_p^1, A_p^2$ defined as $A_p^1 = (Q, \Sigma, \delta, q_0, p)$ and $A_p^2 = (Q, \Sigma, \delta, p, q_f)$. Notice that we can now rephrase the above observation as:

$w \in \sqrt{L}$ if and only if there exists a state $p \in Q$ such that $w \in \mathcal{L}(A_p^1) \cap \mathcal{L}(A_p^2)$.

Let $B$ be any NFA for the language $\cup_{p\in Q} \mathcal{L}(A_p^1) \cap \mathcal{L}(A_p^2)$. By the above observation, it follows that $B$ recognizes $\sqrt{L}$. Note that $B$ can be obtained by pairing operations on the NFAs from the set $\{A_p^i : p \in Q,\ i \in \{1,2\}\}$ and each element in this set can be easily constructed from $A$. It follows then that we can explicitly construct $B$ from $A$.

(b) Once again we begin with an observation:

$w = w_1 w_2 ... w_n \in \text{Cyc}(L)$ if and only if there exists $1 \leq i \leq n$ and $p \in Q$ such that $q_f \in \delta(p, w_1 w_2 ... w_i)$ and $p \in \delta(q_0, w_{i+1} w_{i+2} ... w_n)$.

Indeed, suppose for some word $w$, such an $i$ and $p$ exist. Then, notice that if we set $v = w_1 w_2 ... w_i$ and $u = w_{i+1} ... w_n$, then $uv \in L$ and so $w = vu \in \text{Cyc}(L)$. On the other hand if $w \in \text{Cyc}(L)$, then there is a partition of $w$ into some $v = w_1 w_2 ... w_i$ and $u = w_{i+1} ... w_n$ such that $uv \in L$. Since $uv \in L$, there must be an accepting run of $uv$ in $A$. Let $p$ be the state reached after reading $u$ along this run. It follows then that $p \in \delta(q_0, w_{i+1} ... w_n)$ and $q_f \in \delta(p, w_1 w_2 ... w_i)$.

With this observation, we can do the following: For every state $p \in Q$, construct the two NFAs $A_p^1$ and $A_p^2$ as defined in the subproblem a). Now, notice that

$w \in \text{Cyc}(L)$ if and only if there exists $p \in Q$ such that $w \in \mathcal{L}(A_p^2)\mathcal{L}(A_p^1)$, i.e., $w$ is in the concatenation of the languages of $A_p^2$ and $A_p^1$ for some $p$.

Let $B$ be any NFA for the regular language $\cup_{p\in Q} \mathcal{L}(A_p^2)\mathcal{L}(A_p^1)$. By the above observation, it follows that $B$ recognizes $\text{Cyc}(L)$. Note that given the NFAs $A_p^2$ and $A_p^1$, we can obtain an NFA-$\epsilon$ for their concatenation by simply adding an $\epsilon$ transition from the final state of $A_p^2$ to the initial state of $A_p^1$. By using additional pairing operations, we can explicitly construct $B$ from $A$.

**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 $\delta$ 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).