

Exercise 1 (Time and Space Complexity):

(2 Points)

Consider the comparison of the complexities for DFA and NFA method given in lecture three, slide 9.

- Give an example regular expression, for which the DFA method requires the worst-case space complexity $\mathcal{O}(2^{|\alpha|})$. Argue why your answer is correct.
- Give an example regular expression, for which the NFA method requires the worst-case time complexity $\mathcal{O}(|\alpha| \cdot |w|)$. Defend your answer shortly.

Lösung:

- Every regular expression has a DFA with equivalent language. We say two states of a DFA A say s, s' are different if there exists a string v such that $\delta(s, v)$ is in the final state and $\delta(s', v)$ is not in the final state or vice-versa (notion of equivalent states was somewhat touched in Exercise sheet 1).

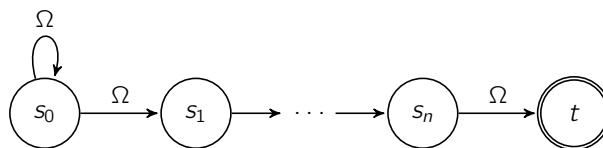
Consider the regular expression ¹

$$r_n = \Sigma^* a \Sigma^n b$$

where alphabet $\Sigma = \{a, b\}$. Let A_n be the corresponding DFA for r_n . Let $w = \alpha a \beta b$ and $w' = \alpha' a \beta' b$ where $|\beta| = |\beta'| = n$. After reading w and w' let the automaton be in the states s, s' , respectively. Let $\beta \neq \beta'$, thus there exist an index i such that $\beta(i) \neq \beta'(i)$ [the i^{th} symbol of β]. W.l.o.g assume $\beta(i) = a$ and $\beta'(i) = b$. The string $v = b^{i+1}$ distinguishes s and s' , as $\delta(s, v)$ is in final state and $\delta(s', v)$ is not (since, $wv \in \mathcal{L}(r_n)$ and $ww' \notin \mathcal{L}(r_n)$).

Thus, for each distinct β there is a distinct state of A_n and there are 2^n different β 's.

- $\alpha_n := \Omega^* \Omega^n$.
 A_n :



The number of states of automaton A_n is $\mathcal{O}(n)$. For all string w , the number of reachable states increase by 1 till it becomes equal to $|Q|$. Thus, the NFA method runs in $\mathcal{O}(|\alpha_n| \cdot |w|)$.

Exercise 2 (Longest First Match Principle):

(4 Points)

- For extended matching two principles have been introduced to resolve nondeterminism during analysis, the *longest match* principle and the *first match* principle. Argue why these principles are reasonable to use. Instead, we could have insisted on an unambiguous definition of the symbol classes, i.e. for regular expressions $\alpha_1, \dots, \alpha_n$ it should hold $\llbracket \alpha_i \rrbracket \cap \llbracket \alpha_j \rrbracket = \emptyset$, for all $1 \leq i < j \leq n$. Why is this not a good idea from a practical point of view? Give examples to support your explanations.

¹This example is due to Samiro

- b)** Let $\alpha_1, \dots, \alpha_n$ be regular expressions over Σ and $w \in \Sigma^*$. In the lecture it was assumed that $\varepsilon \notin \llbracket \alpha_i \rrbracket$ and $\llbracket \alpha_i \rrbracket \neq \emptyset$ for all $i \in \{1, \dots, n\}$. Show that these are reasonable assumptions by proving the following proposition:
- If $\llbracket \alpha_i \rrbracket = \emptyset$ for some $i \in \{1, \dots, n\}$ there exists no *flm*-analysis of w w.r.t. $\alpha_1, \dots, \alpha_n$ that is not a *flm*-analysis of w w.r.t. $\alpha_1, \dots, \alpha_{i-1}, \alpha_{i+1}, \dots, \alpha_n$ as well.
 - If $\varepsilon \in \llbracket \alpha_i \rrbracket$ for some $i \in \{1, \dots, n\}$ then the *flm*-analysis of w w.r.t. $\alpha_1, \dots, \alpha_n$ is not unique (if it exists).

Lösung:

- a)** Example: $\Sigma = \{\leftarrow, \oplus, a, \dots, z\}$

$\alpha_1 = \text{var}$
 $\alpha_2 = \text{while}$
 $\alpha_3 = \text{end}$
 $\alpha_4 = \leftarrow$
 $\alpha_5 = \oplus$
 $\alpha_6 = (a + \dots + z)(a + \dots + z)^*$

$\llbracket \alpha_1 \rrbracket \cap \llbracket \alpha_6 \rrbracket = \{\text{var}\}$ but of course, *var* should be the keyword instead of an identifier.

If we did not assume the *longest match* principle, in addition to *variable* another decomposition of *variable* would be e.g. *var*, *iable*. . .

If we did not assume the *first match* principle, instead of α_6 we would have to provide regular expression:

$$\alpha'_6 = \neg(\varepsilon + \text{var} + \text{while} + \text{end} + \Sigma^* \oplus \Sigma^* + \Sigma^* \leftarrow \Sigma^*)$$

- b)** If $(w_1, \dots, w_k)$ is a decomposition of w , then

$$w = w_1 \dots w_k \text{ and } \forall j \in \{1, \dots, k\} \exists i_j \in \{1, \dots, n\} : w_j \in \llbracket \alpha_{i_j} \rrbracket \quad (*)$$

If $\ell \in \{1, \dots, n\}$ such that $\llbracket \alpha_\ell \rrbracket = \emptyset$, then $\forall j \in \{1, \dots, k\} : w_j \notin \llbracket \alpha_\ell \rrbracket$, even if $w_j = \varepsilon$. Thus,

$$(*) \Leftrightarrow w = w_1 \dots w_k \text{ and } \forall j \in \{1, \dots, k\} \exists i_j \in \{1, \dots, \ell - 1, \ell + 1, \dots, n\} : w_j \in \llbracket \alpha_{i_j} \rrbracket$$

Let $(w_1, \dots, w_k)$ be a decomposition of w respecting the *flm* principle, i.e. $w = w_1 \dots w_k$, and let α_ε be a regular expression with $\varepsilon \in \llbracket \alpha_\varepsilon \rrbracket$. Then $(w_1, \dots, w_k, w_\varepsilon)$ is a decomposition of w respecting the *flm* principle as well ($w = w_1 \dots w_k \varepsilon$).

Exercise 3 (Backtracking DFA):

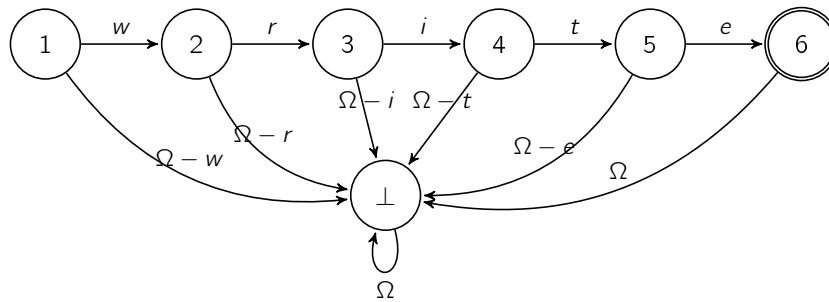
(4 Points)

Let $\alpha_1 = \text{write}$, $\alpha_2 = \Sigma(\Sigma|N)^*$ where $\Sigma := (a| \dots |z|A| \dots |Z)$, $N := (0|1| \dots |9)$.

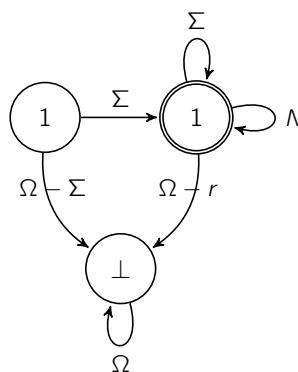
- Construct DFAs $\mathfrak{A}_i$ for α_i such that $\mathcal{L}(\mathfrak{A}_i) = \llbracket \alpha_i \rrbracket$.
- Construct the product automaton $\mathfrak{A} = \mathfrak{A}_1 \otimes \mathfrak{A}_2$.
- Determine the *first match* partitioning of the set of final states in $\mathfrak{A}$.
(The regular expressions are ordered (α_1, α_2) .)
- Determine the set of reachable and productive states in $\mathfrak{A}$.
- Compute the run of the corresponding backtracking DFA for input $\text{writeln}()$. Provide the run by giving the corresponding configurations.

Lösung:

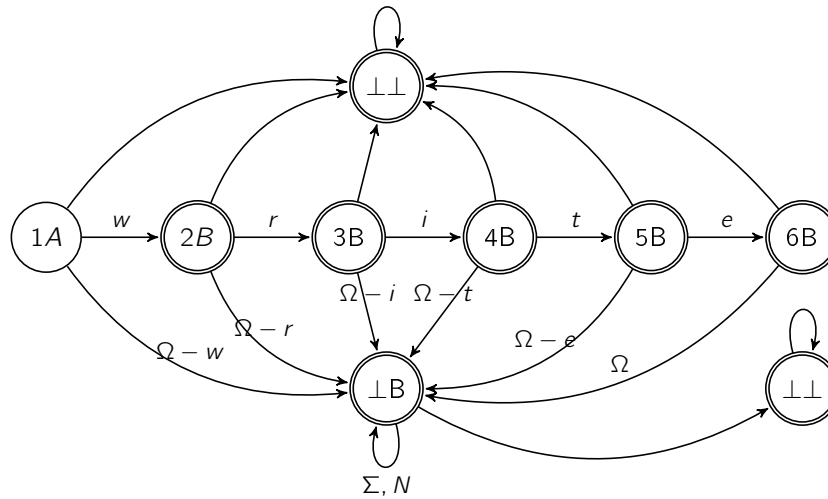
- $\Sigma \cup N \subseteq \Omega$.
 A_1 :



A_2 :



- $A = A_1 \otimes A_2$:



c) $F_1 = \{6B\}$ and $F_2 = \{2B, 3B, 4B, 5B, \perp B\}$.

d) Only non-productive state is $\perp\perp$.

e) $(N, 1A.writeln(), \varepsilon) \vdash (ID, 2B.writeln(), \varepsilon) \vdash (ID, 3B.writeln(), \varepsilon) \vdash (ID, 4B.writeln(), \varepsilon) \vdash (ID, 5B.writeln(), \varepsilon) \vdash (WRITE, 6B.writeln(), \varepsilon) \vdash (ID, \perp B.n(), \varepsilon) \vdash (ID, \perp B(), \varepsilon) \vdash (N, 1A(), ID) \vdash ID.lexerr$