DECISION PROBLEMS FOR REGULAR LANGUAGES

Exercise 1:

Give algorithms to tell whether

a) a given regular language $L$ is universal.
   (i.e. $L = \Sigma^*$).

b) two regular languages have at least one string in common.

STATE MINIMIZATION

Exercise 2

Minimize the following DFA.

![DFA Diagram](image)

Exercise 3

Minimize the following DFA.

![DFA Diagram](image)
SOLUTIONS

1) a) If \( L = \Sigma^* \), then \( \overline{L} = \Sigma^* = \emptyset \)

Hence, we need to check whether \( \overline{L} \) is empty.

Algorithm when \( L \) is given as a DFA \( D_L \):

1) Construct a DFA \( D_{\overline{L}} \) s.t. \( L(D_{\overline{L}}) = \overline{L} \) by swapping final and non-final states of \( D_L \).

2) Check whether \( D_{\overline{L}} \) is empty (by constructing the set of states reachable from the initial state, and checking whether it contains at least one final state).

Algorithm when \( L \) is given as an NFA \( N_L \):

1) Determinize \( N_L \), i.e., construct a DFA \( D_L \) s.t. \( L(D_L) = L(N_L) \) (Note: \( D_L \) might have a number of states that is exponential in the number of states of \( N_L \)).

2) Proceed as in the case of a DFA.

Algorithm when \( L \) is given as a RE \( E_L \):

1) Construct an \( \varepsilon \)-NFA \( N_{\varepsilon_L} \) s.t. \( L(N_{\varepsilon_L}) = L(E_L) \)

2) Eliminate \( \varepsilon \)-transitions from \( N_{\varepsilon_L} \), obtaining an NFA \( N_L \) s.t. \( L(N_L) = L(N_{\varepsilon_L}) \)

3) Proceed as in the case of an NFA.
1) b) To check whether two REs $L_1$ and $L_2$ have at least one string in common, we can check whether $L_1 \cap L_2$ is nonempty.

**Algorithm:**

1) Construct a DFA/NFA/ε-NFA/RE for $L_1 \cap L_2$, starting from DFA/NFA/ε-NFA/REs for $L_1$ and for $L_2$.

2) Check whether $L_1 \cap L_2$ is not-empty.

**Note:** to construct a DFA/NFA/ε-NFA/RE for $L_1 \cap L_2$, we can use De Morgan’s law.

- $L_1 \cap L_2$ is still a RE, since REs are closed under intersection.
2) we start with the set of all states, and in step 0
the set will be separated into two sets of final and non-final
states. (we will use 0,1,...,8 instead of 00,91,...,98).

\[
\{0,1,2,3,4,5,6,7,8\}
\]

step 0: \[
\{0,1,2,3,4,5,8\} \quad \{6,7\}
\]

step 1: \[
\{0,1,2,5,8\} \quad \{3,4\} \quad \{6\} \quad \{7\}
\]

step 2: \[
\{2,5,8\} \quad \{0,1\} \quad \{3,4\} \quad \{6\} \quad \{7\}
\]

step 3: \[
\{2,5,8\} \quad \{0\} \quad \{1\} \quad \{3\} \quad \{4\} \quad \{6\} \quad \{7\}
\]

step 4: \[
\{2,5,8\} \quad \{0\} \quad \{1\} \quad \{3\} \quad \{4\} \quad \{6\} \quad \{7\} \quad \text{<no change>}
\]

\[\{2\} = \{2,5,8\}\]

Since the state \{4\} is not
reachable from the initial
state, it should be eliminated
from the minimized automaton.
Exercise 3) Same as previous exercise we will have:

\[
\{0,1,2,3,4\}
\]

**Step 0:**

\[
\{4\} \quad \{0,1,2,3\}
\]

**Step 1:**

\[
\{4\} \quad \{3\} \quad \{0,1,2\}
\]

**Step 2:**

\[
\{4\} \quad \{3\} \quad \{0,1\} \quad \{2\}
\]

**Step 3:**

\[
\{4\} \quad \{3\} \quad \{0,1\} \quad \{2\}
\]

\[
[0] = \{0,1\}
\]

\[
\text{a, c}
\]

\[
[0]
\]

\[
\text{a, c}
\]

\[
[1]
\]

\[
\text{b}
\]

\[
[2]
\]

\[
\text{a, b, c}
\]

\[
[3]
\]

\[
\text{a, b, c}
\]

\[
[4]
\]

\[
\text{a, b, c}
\]

**Attention:** Although none of final states are reachable from the state [4], it shouldn't be eliminated because we are looking for a DFA, and by eliminating [4], the automation will not be a DFA anymore!

All the states are reachable from the initial state, so we don't need to eliminate any state and the DFA is minimum.