Lesson 1: Finite state automata

Theme: Deterministic and non-deterministic finite state automata.

1 Deterministic finite state automata

A deterministic finite state automaton (DFA) is a system \( A = \langle \Sigma, Q, q_0, F, \delta \rangle \), where each component is as follows.

- \( Q \) is a finite set of states.
- \( q_0 \in Q \) is the initial state.
- \( F \subseteq Q \) is the set of accepting states.
- \( \delta : Q \times \Sigma \to Q \) is the transition function.

In this case, we will say that “\( A \) is a DFA over alphabet \( \Sigma \),” or that “the alphabet of \( A \) is \( \Sigma \).”

Remark 1.1 A DFA \( A = \langle \Sigma, Q, q_0, F, \delta \rangle \) can be visualised as a directed graph as follows.

- The vertices are elements of \( Q \).
- There is an edge from \( p \) to \( p' \) labeled with \( a \), if \( \delta(p, a) = p' \).

On input word \( w = a_1 \cdots a_n \), the run of \( A \) on \( w \) is the sequence:

\[
p_0 a_1 p_1 a_2 p_2 \cdots a_n p_n,
\]

where \( p_0 = q_0 \) and \( \delta(p_i, a_{i+1}) = p_{i+1} \), for each \( i = 0, \ldots, n - 1 \).

Sometimes we are interested in a run that does not start from the initial state. In that case, we can define the run of \( A \) on \( w \) starting from state \( q \) as the sequence defined as above, but with condition \( p_0 = q \). That is,

\[
p_0 a_1 p_1 a_2 p_2 \cdots a_n p_n,
\]

where \( p_0 = q \) and \( \delta(p_i, a_{i+1}) = p_{i+1} \), for each \( i = 0, \ldots, n - 1 \).

A run is called an accepting run, if \( p_0 = q_0 \) and \( q_n \in F \). We say that \( A \) accepts \( w \), if there is an accepting run of \( A \) on \( w \). The language of all words accepted by \( A \) is denoted by \( L(A) \).

A language \( L \) is called a regular language, if there is a DFA \( A \) such that \( L(A) = L \).

Remark 1.2 Let \( A = \langle \Sigma, Q, q_0, F, \delta \rangle \) be a DFA.

- The empty string \( \varepsilon \) is accepted by \( A \) if and only if \( q_0 \in F \).
- For every word \( w \), there is exactly one run of \( A \) on \( w \).

Theorem 1.3 Regular languages are closed under boolean operations, i.e., intersection, union, and complementation. More formally, it can be stated as follows.

- For every DFA \( A \) over alphabet \( \Sigma \), there is a DFA \( A' \) over the same alphabet \( \Sigma \) such that \( L(A') = \Sigma^* - L(A) \).
- For every two DFA \( A_1 \) and \( A_2 \), there is a DFA \( A' \) such that \( L(A') = L(A_1) \cap L(A_2) \).
- For every two DFA \( A_1 \) and \( A_2 \), there is a DFA \( A' \) such that \( L(A') = L(A_1) \cup L(A_2) \).
2 Non-deterministic finite state automata

A non-deterministic finite state automaton (NFA) is a system \( A = (\Sigma, Q, q_0, F, \delta) \), where each component is as follows.

- \( \Sigma \) is the alphabet.
- \( Q \) is a finite set of states.
- \( q_0 \in Q \) is the initial state.
- \( F \subseteq Q \) is the set of accepting states.
- \( \delta \subseteq Q \times \Sigma \times Q \) is the transition relation.

As before, we will say that \( A \) is an NFA over alphabet \( \Sigma \), or that \( \text{the alphabet of } A \text{ is } \Sigma \).

On input word \( w = a_1 \cdots a_n \), a run of \( A \) on \( w \) is a sequence:

\[ q_0 a_1 q_1 a_2 q_2 \cdots a_n q_n, \]

where \((q_i, a_{i+1}, q_{i+1}) \in \delta\), for each \( i = 0, \ldots, n - 1 \) \( ^{\text{It is called accepting run, if } q_n \in F.} \) We say that \( A \) accepts \( w \), if there is an accepting run of \( A \) on \( w \). The language of all words accepted by \( A \) is denoted by \( L(A) \). A language \( L \) is an NFA language, if there is an NFA \( A \) such that \( L = L(A) \), in which, we say that the language \( L \) is accepted by \( A \), or \( A \) accepts the language \( L \).

Remark 1.4 NFA languages are closed under intersection and union. More formally, it can be stated as follows.

- For every two NFA \( A_1 \) and \( A_2 \), there is an NFA \( A' \) such that \( L(A') = L(A_1) \cap L(A_2) \).
- For every two NFA \( A_1 \) and \( A_2 \), there is an NFA \( A' \) such that \( L(A') = L(A_1) \cup L(A_2) \).

Question: Why can we not conclude that NFA languages are closed under complementation directly from the definition of NFA? \( \blacksquare \)

Theorem 1.5 For every NFA \( A \), there is a DFA \( A' \) such that \( L(A) = L(A') \).

In view of Theorem 1.5, we can say that a language is regular if and only if it is accepted by an NFA.

Corollary 1.6 NFA languages are closed under complement. That is, for every NFA \( A \) over alphabet \( \Sigma \), there is a DFA \( A' \) over the same alphabet \( \Sigma \) such that \( L(A') = \Sigma^* - L(A) \).

Theorem 1.7 Regular languages are closed under concatenation and Kleene star. More formally, it can be stated as follows.

- If \( L_1 \) and \( L_2 \) are regular languages, so is \( L_1L_2 \).
- If \( L \) is a regular language, so is \( L^* \).

\(^*\text{As in the case of DFA, we can define a run of } A \text{ on } w \text{ starting from state } q \text{ as above, but starts from state } q.\)
Appendix: Concatenation and Kleene star

For two words $u$ and $v$, $u \cdot v$ denotes the word obtained by concatenating $v$ at the end of $u$. ($u \cdot v$ reads: $u$ concatenates with $v$.) By default, $u \cdot \epsilon = \epsilon \cdot u = u$. We will usually omit $\cdot$ and simply write $uv$ instead of $u \cdot v$.

In the following, let $L_1, L_2$ and $L$ be languages. We define the following operators.

\[
\begin{align*}
L_1 \cdot L_2 & := \{ uv \mid u \in L_1 \text{ and } v \in L_2 \} \quad \text{(Concatenation)} \\
L^n & := \{ u_1 \cdots u_n \mid \text{each } u_i \in L \} \\
L^* & := \bigcup_{n \geq 0} L^n \quad \text{(Kleene star)}
\end{align*}
\]

As before, we usually write $L_1L_2$ to denote $L_1 \cdot L_2$, and $L_1L_2$ reads as $L_1$ concatenates with $L_2$.

Note that by default, for any set $X \subseteq \Sigma^*$, $X^0 = \{ \epsilon \}$. Thus, $\emptyset^* = \{ \epsilon \}$. 