Nondeterminism

So far, the next state was uniquely defined, given the previous state and the input symbol. This is called deterministic computation. In a nondeterministic automaton several choices may exist for the next state at any point of computation.

Deterministic finite automaton: DFA.

Nondeterministic finite automaton: NFA.

Example 1. Nondeterministic automaton that accepts all strings containing 001:

- Two outgoing edges 0 for $q_1$.
- No outgoing edge 1 for $q_2$.

Computation in NFA

- If there are several choices, the automaton runs several computations in parallel.
• If there is no particular symbol on output edges, the automaton discontinues the computation.

• If there is an outgoing edge labeled “$\varepsilon$”, the automaton runs several computations in parallel, starting with the current state and also the states pointed by the $\varepsilon$-arrow.

• NFA accepts the input string if and only if one of the parallel computations has accepted.

Deterministic

Nondeterministic

Example 2. Consider how automaton from Example 1 works on input 100011. Computations form a tree.
Example 3. NFA that accepts all strings containing either 001 or 101 (or both).

Definition. A nondeterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- $Q$ is a finite set of states;
- $\Sigma$ is a finite alphabet;
• $\delta: Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ is a transition function;

• $q_0 \in Q$ is a start state;

• $F \subseteq Q$ is a set of accept states.

$\mathcal{P}(Q)$ is a set of all subsets of $Q$ (called a power set of $Q$). $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$.

**Example 4.** Let us apply the definition for the automaton from Example 1.

• Set of states $Q = \{q_1, q_2, q_3, q_4\}$.

• Alphabet $\Sigma = \{0, 1\}$.

• $\delta$ is described in the table:

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_1$</td>
<td>${q_1, q_2}$</td>
<td>${q_1}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_2$</td>
<td>${q_3}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$q_3$</td>
<td>$\emptyset$</td>
<td>${q_4}$</td>
<td>${q_2}$</td>
</tr>
<tr>
<td>$q_4$</td>
<td>${q_4}$</td>
<td>${q_4}$</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

• Start state is $q_1$.

• Set of accept states $F = \{q_4\}$.

**Definition.** Let $M = (Q, \Sigma, \delta, q_1, F)$ be an NFA and $w$ be a string over alphabet $\Sigma$. Then we say that $M$ accepts the string $w$, if we can write $w = a_1a_2\ldots a_m$, where $a_i \in \Sigma_\varepsilon$ (some $a_i = \varepsilon$, i.e. “empty”), and there is a sequence of states $r_0, r_1, \ldots, r_m$, satisfying the following:

1. $r_0 = q_0$;

2. for each $i = 0, 1, \ldots, m-1$ it holds $r_{i+1} \in \delta(r_i, a_{i+1})$;

3. $r_m \in F$.

**Definition.** Two automata are equivalent if they recognise the same language.

Trivially, every DFA is an NFA. The opposite state can be formulated.

**Theorem.** Every NFA has an equivalent DFA.

**Proof idea.** Given NFA, we construct DFA that recognises the same language. If NFA has $k$ states then the corresponding DFA will have $2^k$ states, representing all the subsets of states of NFA.
Proof. Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA that recognises a language $L$. We want to construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$, that recognises the same language $L$.

1) First consider a simple special case, where $N$ does not have $\varepsilon$-arrows.

- $Q' = \mathcal{P}(Q)$, a set of all subsets of $Q$.
- For $R \in Q'$ and $a \in \Sigma$, let
  \[ \delta'(R, a) = \bigcup_{r \in R} \delta(r, a) = \{ q \in Q \mid q \in \delta(r, a) \text{ for some } r \in R \}. \]

In other words, if $R$ is a state of $M$, it is also a set of states of $N$. When $M$ reads a symbol $a$ in state $R$, it can go to any of the states $\delta(r, a)$ for all $r \in R$.

- Start state: $q'_0 = \{q_0\}$. $M$ starts in the single state $q_0$.
- Accept states:
  \[ F' = \{ R \in Q' \mid R \text{ contains an accept state of } N \}. \]

2) Now consider $\varepsilon$ arrows. Denote (so called $\varepsilon$-closure):

\[ E(R) = \{ q \mid q \text{ can be reached from } R \text{ by travelling from } R \text{ by using } \varepsilon \text{ arrows only} \}. \]

(For instance, in Example 1 $E(\{q_1\}) = \{q_1\}$, $E(\{q_3\}) = \{q_2, q_3\}$). Replace (*) by

\[ \delta'(R, a) = \bigcup_{r \in R} E(\delta(r, a)) = \{ q \in Q \mid q \in E(\delta(r, a)) \text{ for some } r \in R \}. \]

We also need to replace the start state $\{q_0\}$ by $E(\{q_0\})$. This completes the construction.

The construction of $M$ works correctly. At every step in the computation of $M$ on an input, it enters a state that corresponds to a subset of states that $N$ could be in at that point.

Practise session

1. There is an NFA:
More formally, \( N = (\{q_1, q_2, q_3\}, \{a, b\}, \delta, q_1, \{q_1\}) \), where transition function is in the table

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>( \varepsilon )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q_1 )</td>
<td>( \emptyset )</td>
<td>{q_2}</td>
<td>{q_3}</td>
</tr>
<tr>
<td>( q_2 )</td>
<td>{q_2, q_3}</td>
<td>{q_3}</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>( q_3 )</td>
<td>{q_1}</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
</tbody>
</table>

Construct an equivalent DFA.

**Solution.** We construct \( M = (Q', \{a, b\}, \delta', q'_0, F') \).

- \( Q' = \{q'_0, q'_1, q'_2, q'_3, q'_{12}, q'_{13}, q'_{23}, q'_{123}\} \)

- Transition function \( \delta' \):

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>( q'_0 )</td>
<td>( q'_0 )</td>
<td>( q'_0 )</td>
</tr>
<tr>
<td>( q'_1 )</td>
<td>( q'_0 )</td>
<td>( q'_2 )</td>
</tr>
<tr>
<td>( q'_2 )</td>
<td>( q'_{23} )</td>
<td>( q'_3 )</td>
</tr>
<tr>
<td>( q'_3 )</td>
<td>( q'_{13} )</td>
<td>( q'_0 )</td>
</tr>
<tr>
<td>( q'_{12} )</td>
<td>( q'_{23} )</td>
<td>( q'_3 )</td>
</tr>
<tr>
<td>( q'_{13} )</td>
<td>( q'_{23} )</td>
<td>( q'_2 )</td>
</tr>
<tr>
<td>( q'_{23} )</td>
<td>( q'_{123} )</td>
<td>( q'_3 )</td>
</tr>
<tr>
<td>( q'_{123} )</td>
<td>( q'_{123} )</td>
<td>( q'_23 )</td>
</tr>
</tbody>
</table>

- The start state \( q'_{13} \) (since in \( N \) we can arrive to \( q_3 \) from \( q_1 \) by using \( \varepsilon \)-arrows).

- \( F = \{q'_1, q'_{12}, q'_{13}, q'_{123}\} \).

The resulting \( M \) is:
Note that we will never arrive to states $q'_1$ and $q'_12$. Therefore, they can be removed. We are left with the following DFA:

\[ L_1 \circ L_2 \] is also a regular language.

**Proof idea.** Let $N_1$ and $N_2$ be two NFAs, recognising languages $L_1$ and $L_2$, respectively. We construct NFA $N$ by connecting all accept states of $N_1$ to $N_2$ start state by $\varepsilon$-arrows:
Solution. Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be NFAs recognising $L_1$ and $L_2$, respectively. We construct NFA $N = (Q, \Sigma, \delta, q_1, F_2)$ that recognises $L_1 \circ L_2$.

- States set $Q = Q_1 \cup Q_2$.
- Start state is $q_1$, i.e. the same as start state of $N_1$.
- Accept states set is $F_2$, i.e. the same as in $N_2$.
- Transition function $\delta$ is as follows:

$$
\delta(q, a) =
\begin{cases}
\delta_1(q, a) & \text{if } q \in Q_1, q \notin F_1, \\
\delta_1(q, a) & \text{if } q \in F_1, a \neq \varepsilon, \\
\delta_1(q, a) \cup \{q_2\} & \text{if } q \in F_1, a = \varepsilon, \\
\delta_2(q, a) & \text{if } q \in Q_2.
\end{cases}
$$

Automaton $N$ accepts input words $w$ if and only if there exists a computation path starting at $q_1$ and finishing in one of the states in $F_2$:

$$
q_1 \xrightarrow{w_1} r_1 \xrightarrow{\varepsilon} q_2 \xrightarrow{w_2} r_2
$$

where $w = w_1w_2$. This is equivalent to: $w_1 \in L_1$, $w_2 \in L_2$, $w = w_1w_2$. And this is equivalent to $w \in L_1 \circ L_2$.

3. Prove that if $L_1$ is a regular language, then $L_1^*$ is a regular language.

Proof idea. Let $N_1$ be an NFA recognising $L_1$. We construct a new automaton $N$ so that each accept state of $N_1$ is connected to its old start state with $\varepsilon$-arrow. We also add the new start state, that is connected to an old start state with $\varepsilon$-arrow:
Solution. Let \( N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) be an NFA that recognises \( L_1 \). Construct an NFA \( N = (Q, \Sigma, \delta, q_0, F) \), that recognises \( L_1^* \).

- State set \( Q = Q_1 \cup \{q_0\} \), i.e. we add new state \( q_0 \).
- New start state \( q_0 \).
- Set of accept states is \( F = F_1 \cup \{q_0\} \).
- Transition function is

\[
\delta(q, a) = \begin{cases} 
\delta_1(q, a) & \text{if } q \in Q_1, q \notin F_1, \\
\delta_1(q, a) & \text{if } q \in F_1, a \neq \varepsilon, \\
\delta_1(q, a) \cup \{q_1\} & \text{if } q \in F_1, a = \varepsilon, \\
\{q_1\} & \text{if } q = q_0, a = \varepsilon, \\
\emptyset & \text{if } q = q_0, a \neq \varepsilon.
\end{cases}
\]

Suppose that \( N \) accepts input \( w \). This is equivalent that either \( w = \varepsilon \) or \( w = w_1 w_2 \cdots w_k \), where:

This is equivalent that \( w = \varepsilon \) or \( w_1 \in L_1, w_2 \in L_1, \ldots, w_k \in L_1 \) and \( w = w_1 w_2 \cdots w_k \). This is equivalent that \( w \in L_1^* \).