Automata Theory and Formal Languages
Class 12

Marcin Luckner, PhD
mluckner@mini.pw.edu.pl

Version 1.0
16 stycznia 2022

Deterministic finite automata

$$M = (Q, \Sigma, \delta, q_0, F)$$

Deterministic finite automaton has the following components

- Q a finite set of states
- Σ a finite input alphabet
- q_0 the start state $q_0 \in Q$
- F the set of final states $F \subset Q$
- δ transition function $\delta : Q \times \Sigma \to Q$

Nondeterministic finite automata

$$M = (Q, \Sigma, \delta, q_0, F)$$

Nondeterministic finite automaton has identical components as DFA except the transition function

δ transition function $\delta : Q \times \Sigma \to 2^Q$

Finite automation with ϵ–movements

$$M = (Q, \Sigma, \delta, q_0, F)$$

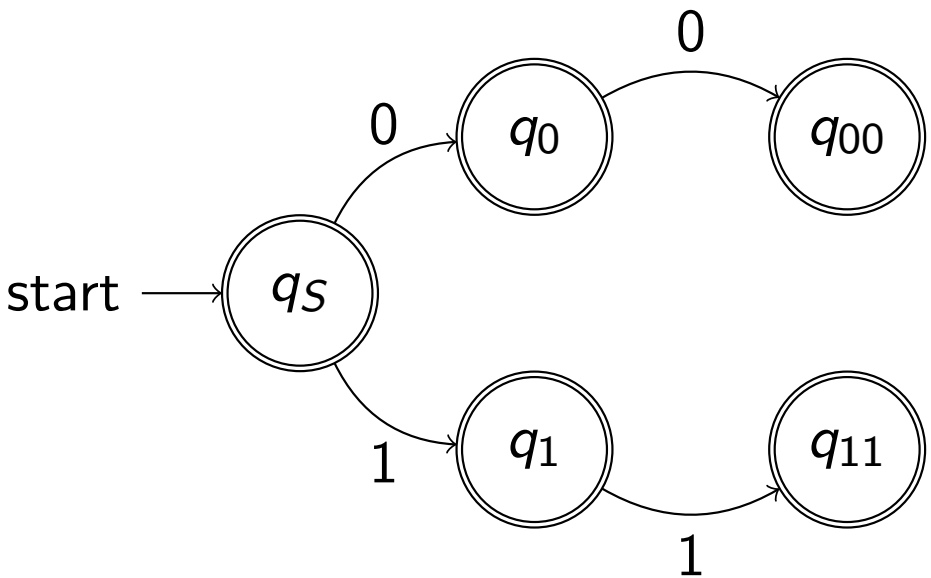
Finite automaton with ϵ–movements has identical components as NAS except the transition function

δ the transition function $\delta : Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q$

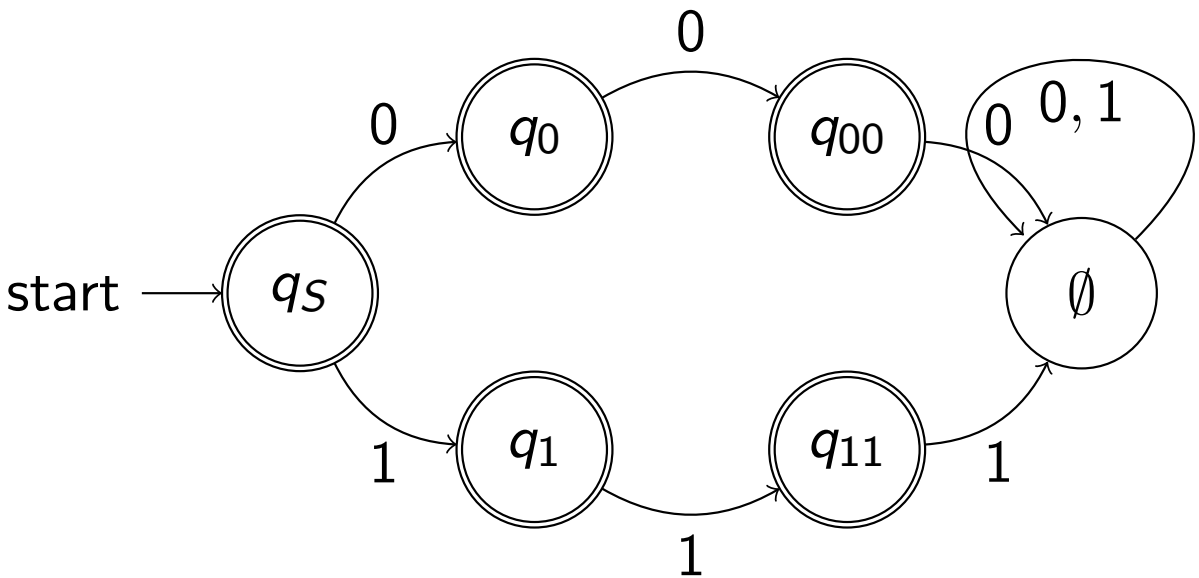
Task

Design deterministic finite automata to recognise the language over alphabet $\Sigma = \{0, 1\}$ of words without a sequence of three identical letters.

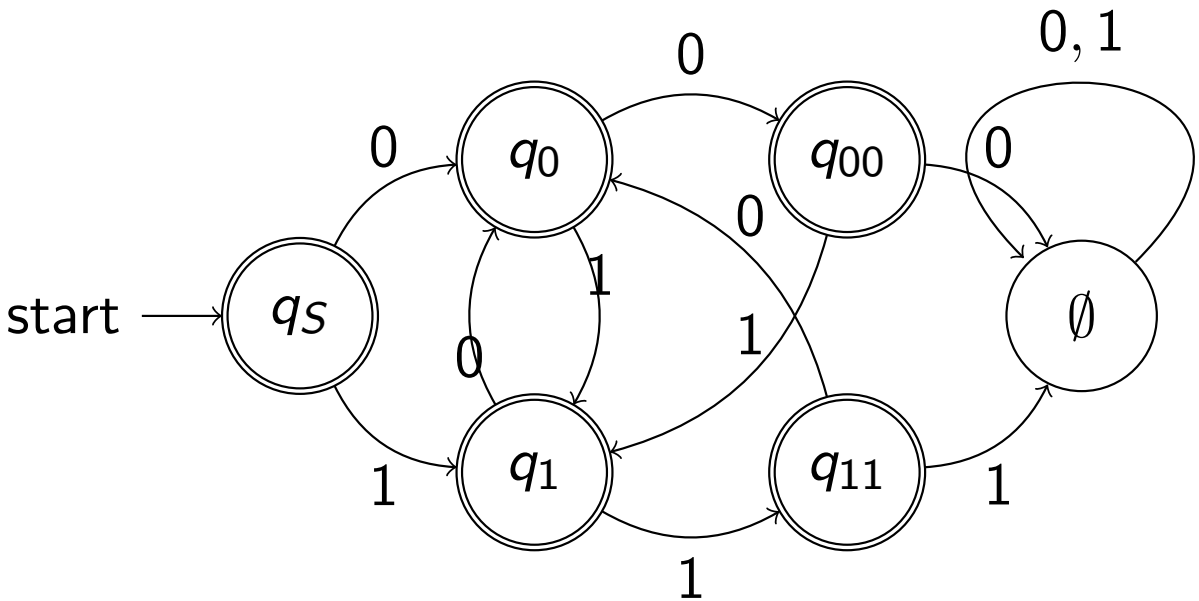
DFA Creation I



DFA Creation II

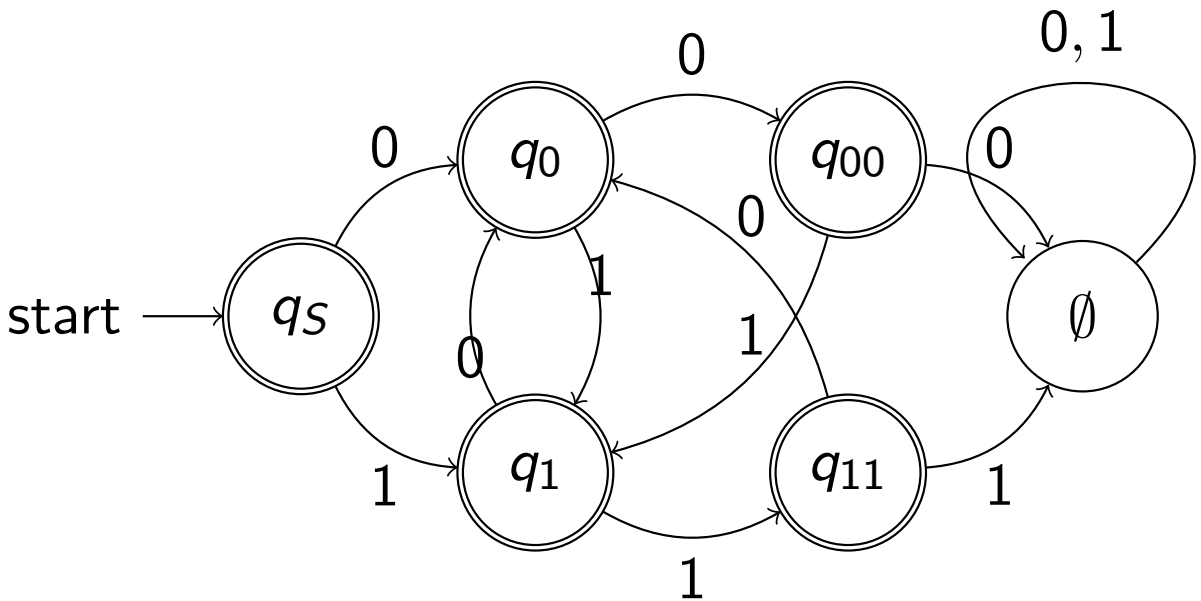


DFA Creation III



Transition table

	0	1
$\overrightarrow{q_S}$	q_0	q_1
$\overrightarrow{q_0}$	q_{00}	q_1
$\overrightarrow{q_1}$	q_0	q_{11}
$\overrightarrow{q_{00}}$	$\emptyset$	q_1
$\overrightarrow{q_{11}}$	q_0	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$



Equivalence of FA and regular expressions

1. If $L = L(A)$ for some finite automaton A then exists regular expressions r such that $L = L(r)$.
2. If $L = L(r)$ for some regular expression r then exists finite automaton A such that $L = L(A)$.

Construction of regular expression from FA I

- Let us enumerate states of DFA A as $1, 2, \ldots, n$.
- Let us define $R_{ij}^{(k)}$ as a path from state i to state j without visiting other states higher than k.
- We can construct the regular expression equivalent of automaton A using the following induction.

Construction of regular expression from FA II

- Basis (k=0):
  - The path cannot visit any other states than i and j.
  - If $i \neq j$
    - If there is no connection between i and j $R_{ij}^{(0)} = \emptyset$
    - If there is a connection between i and j labelled by a $R_{ij}^{(0)} = a$
    - If there is a connection between i and j labelled by $a_1, a_2, \ldots a_k$ $R_{ij}^{(0)} = a_1 + a_2 + \cdots + a_k$
  - If $i = j$ the expressions are extended by ϵ because the path can be traveled without using labelled connections
    - If there is no connection between i and j $R_{ij}^{(0)} = \epsilon$
    - If there is a connection between i and j labelled by a $R_{ij}^{(0)} = \epsilon + a$
    - If there is a connection between i and j labelled by $a_1, a_2, \ldots a_k$ $R_{ij}^{(0)} = \epsilon + a_1 + a_2 + \cdots + a_k$

Construction of regular expression from FA III

- Induction for $R_{ij}^{(k)}$:
  - The path might not visit state k. In such a case, it is described as $R_{ij}^{(k-1)}$.
  - The path might visit state k several times. In such a case, it can be described as a concatenation of three paths $R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$.
  - Both cases can be summarised as:

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \left(R_{kk}^{(k-1)}\right)^* R_{kj}^{(k-1)}$$

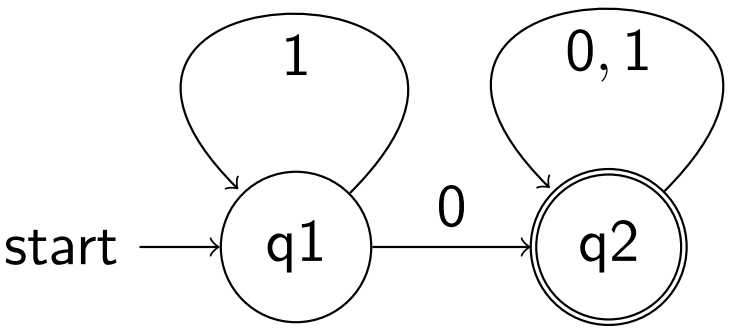
Construction of regular expression from FA IV

- Let's assume that DFA A with n states has the initial state 1 and the final states $f_1, f_2, \ldots, f_m$.
- The regular expression equivalent to the A is an alternative

$$R = R_{1f_1}^{(n)} + R_{1f_2}^{(n)} + \cdots + R_{1f_m}^{(n)}$$

Construction of regular expression example

The language over alphabet $\Sigma = \{0, 1\}$ that accepts words with at least one zero.

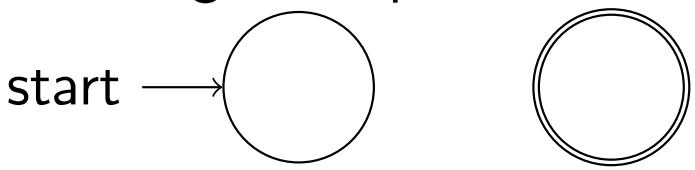


$R_{11}^{(0)}$	$\epsilon + 1$
$R_{12}^{(0)}$	0
$R_{21}^{(0)}$	$\emptyset$
$R_{22}^{(0)}$	$\epsilon + 0 + 1$
$R_{11}^{(1)}$	$\epsilon + 1 + (\epsilon + 1)(\epsilon + 1)^*(\epsilon + 1) = 1^*$
$R_{12}^{(1)}$	$0 + (\epsilon + 1)(\epsilon + 1)^*0 = 1^*0$
$R_{21}^{(1)}$	$\emptyset + \emptyset(\epsilon + 1)^*(\epsilon + 1) = \emptyset$
$R_{22}^{(1)}$	$\epsilon + 0 + 1 + \emptyset(\epsilon + 1)^*00 = \epsilon + 0 + 1$
$R_{12}^{(2)}$	$1^*0 + 1^*0(\epsilon + 0 + 1)^*(\epsilon + 0 + 1)$ $= 1^*0(0 + 1)^*$

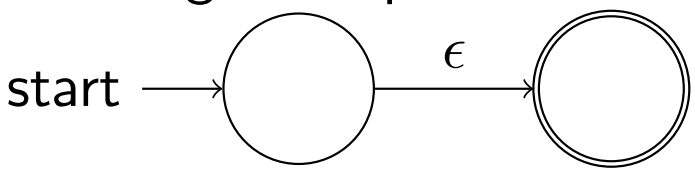
Construction of FA from regular expression I

A regular expression over an alphabet Σ is a construct defined as follows

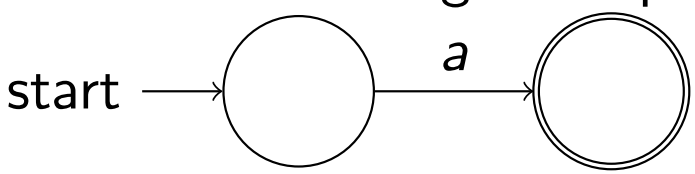
- $\emptyset$ is a regular expression,



- ϵ is a regular expression,



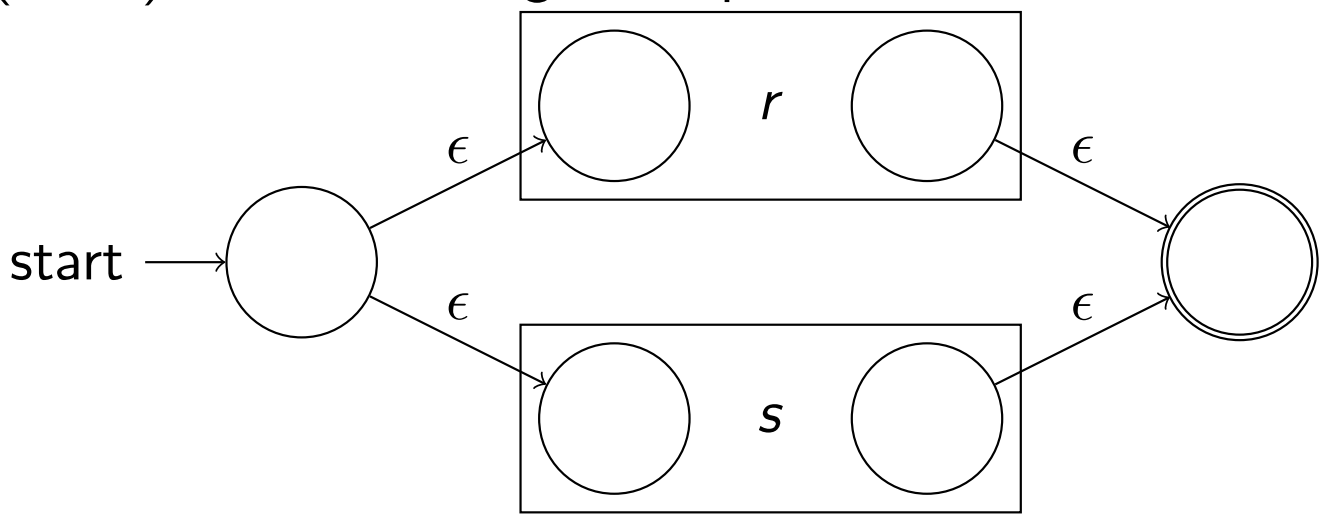
- each $a \in \Sigma$ is a regular expression



Construction of FA from regular expression II

if r and s are regular expressions then the following structures are also regular expressions

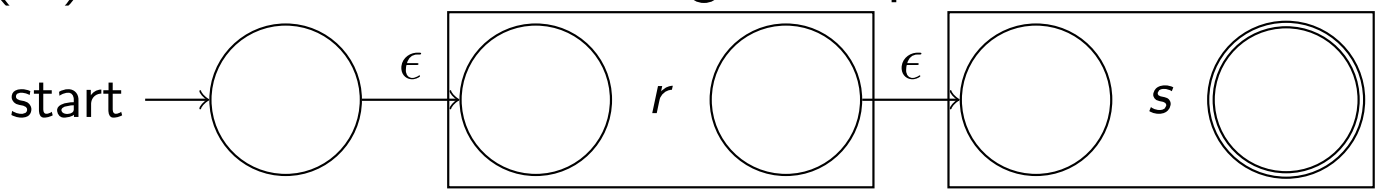
- $(r + s)$ the sum of regular expressions



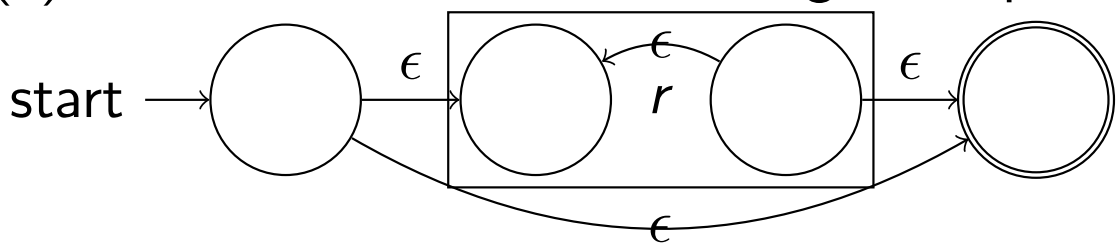
Construction of FA from regular expression III

if r and s are regular expressions then the following structures are also regular expressions

- (rs) the concatenation of regular expressions



- $(r)^*$ the Kleene closure of the regular expression



Quotient of languages

- The right quotient of a language L_1 with a language L_2 is the language L_1/L_2 consisting of such words over the alphabet Σ, which concatenated to words of the dividend give words of the divisor:

$$L_1/L_2 = \{y \in \Sigma^* : (\exists x \in L_2) yx \in L_1\}$$

- The left quotient of a language L_1 with a language L_2 is the language $L_1 \backslash L_2$ consisting of such words over the alphabet Σ words, which concatenated to words of the divisor give words of the dividend:

$$L_1 \backslash L_2 = \{y \in \Sigma^* : (\exists x \in L_2) xy \in L_1\}$$

- The left quotient can be used to create a minimal automation (concerning the number of states).

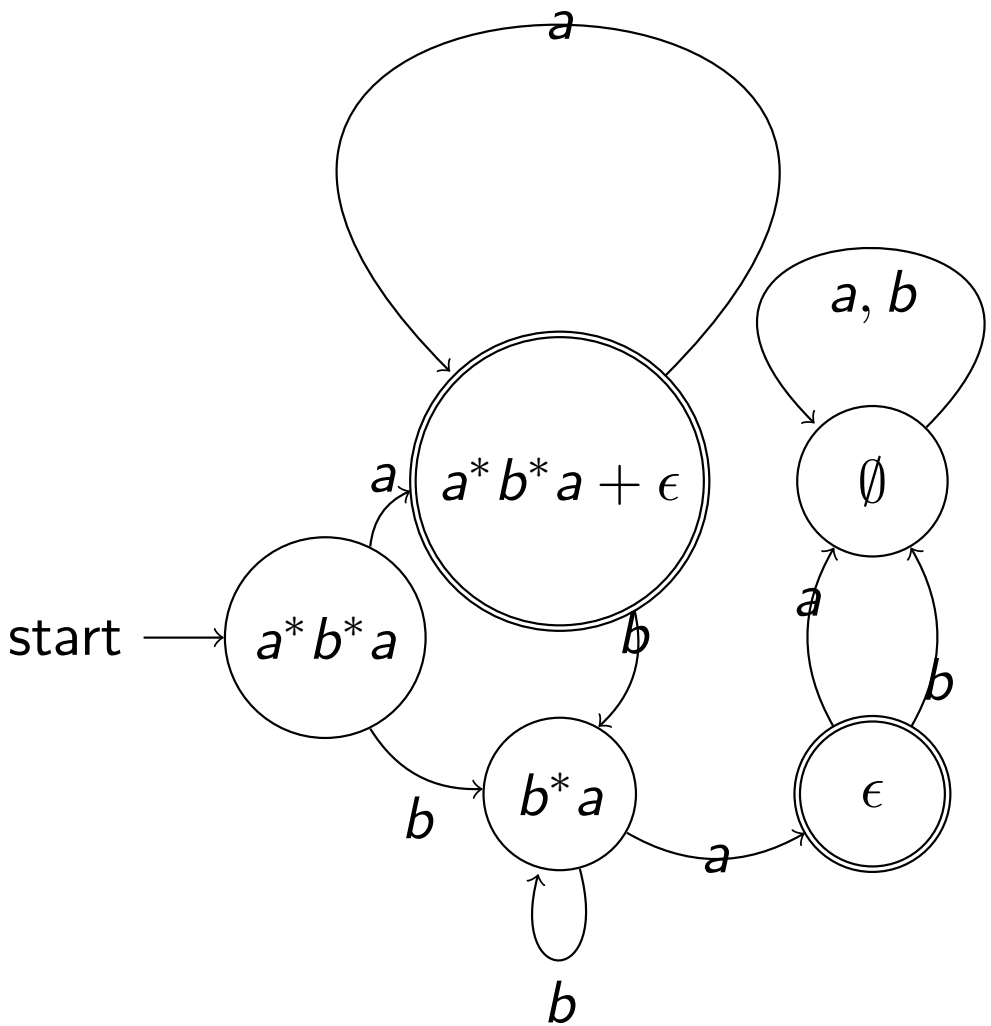
Example

$$a^*b^*a$$

$$
\begin{aligned}
r_0 &= a^*b^*a \\
r_0\backslash a &= a^*b^*a + \epsilon = r_1 \\
r_0\backslash b &= b^*a = r_2 \\
r_1\backslash a &= a^*b^*a + \epsilon = r_1 \\
r_1\backslash b &= b^*a = r_2 \\
r_2\backslash a &= \epsilon = r_3 \\
r_2\backslash b &= b^*a = r_2 \\
r_3\backslash a &= \emptyset \\
r_3\backslash b &= \emptyset \\
\emptyset\backslash a &= \emptyset \\
\emptyset\backslash b &= \emptyset
\end{aligned}
$$

	a	b
$\overrightarrow{\ }q_0$	q_1	q_2
$q_1\overrightarrow{\ }$	q_1	q_2
q_2	q_3	q_2
$q_3\overrightarrow{\ }$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$

Created DFA



- The created automaton has the number of states equal to the number of classes in the Mynhil-Nerode theorem.
  - Each state groups all words from Σ^* that generated starting from the initial state and ending in the state.
  - Each state represents a regular expression that generates a suffix supplementing the words to a word from the language.
- Therefore any state can be removed.

Assignments I

1. Design deterministic finite automata to recognise languages, which are generated by the following regular expressions:
   1.1 $1 + (10)^+(01)^*$,
   1.2 $(\epsilon + 0)(10)^*(1100)(10)^*(\epsilon + 1)$
2. Design deterministic finite automata to recognise the following languages
   2.1 The language over alphabet $\Sigma = \{0, 1\}$ contains words, which have at least two 1's after a pair of adjacent 0's and before the next pair of adjacent 0' or the end of the word. A sequence of three 0's is considered as a single pair,
   2.2 The language over alphabet $\Sigma = \{0, 1\}$ of words without a sequence 010.
3. Design minimal deterministic finite automata for the examples from exercises 1 and 2.