2. FINITE AUTOMATA

2.1 Finite Automata

Automata theory is the study of abstract computational devices (abstract state machine). An automaton is an abstract model of a digital computer. As such, every automaton includes some essential features. It has a mechanism for reading input. It will be assumed that the input is a string over a given alphabet. The input mechanism can read theinput string from left to right, one symbol at a time and it can be detected the end of the string. The automaton can produce the output of some form and has a control unit, which can be in any one of a finite number of internal states, and which can change state in some defined manner based on transition functions.

The finite automata (FA) is characterized by the finite number of states and it is known the following types of the FA:

- Deterministic finite automata (DFA).
- Nondeterministic finite automata (NFA).
- ε -Nondeterministic finite automata (ε –NFA).

2.1.1 Deterministic Finite Automata

A deterministic finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where:

- Q is a finite set of states.
- Σ is an input alphabet.
- δ: $Q \times \Sigma \rightarrow Q$ is a transition function.
- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is a set of accepting states (or final states).

-

Example:

Construct an DFA over alphabet {0, 1} that has the sequence 01 somethere in the string.

Answer:

-Analytical representation

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

$Q=\{q_0, q_1, q_2\}$; $\Sigma=\{0, 1\}$; $q_0=\{q_0\}$; $F=\{q_2\}$;

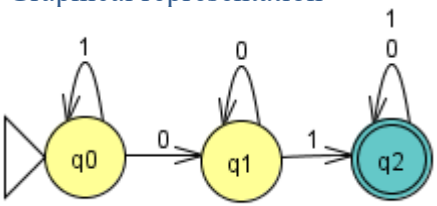
$\delta(q_0,0)=\{q_1\}$; $\delta(q_0,1)=\{q_0\}$; $\delta(q_1,0)=\{q_1\}$; $\delta(q_1,1)=\{q_2\}$;

$\delta(q_2,0)=\{q_2\}$; $\delta(q_2,1)=\{q_2\}$.

- Tabel representation

	0	1
$\rightarrow q_0$	q_1	q_0
q_1	q_1	q_2
$*q_2$	q_2	q_2

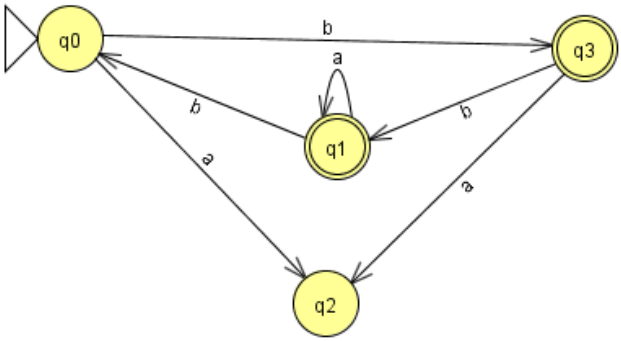
- Graphical representation



The language of a DFA = $(Q, \Sigma, \delta, q_0, F)$ is the set of all strings over w that, starting from q_0 and following the transitions as the string is read left to right, will reach some accepting state (final state). If L is $L(DFA)$ for some DFA, then it is saied that L is regular language.

Practical Tasks

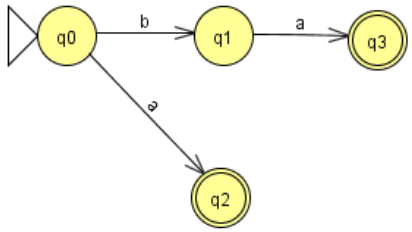
1. For the problem “The Wolf, the Goat and the Cabbage”. present the solution path using finite automata.
2. Define the FA that accepts the identifiers from Java.
3. Design a DFA or NFA (with or without e-transitions) that accepts all strings over the alphabet {$, c, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, .} that correspond to valid currency amounts. A valid string is either a dollar sign followed by a number which has no leading 0's, and may have a decimal point in which case it must be followed by exactly two decimal digits, OR a one or two-digit amount followed by the cent sign c. The single exception to this rule is that strings which begin with "$0." and are followed by exactly two digits are also acceptable. Thus, $432.63, $1, $0.29, 47c, 2c are all accepted, but $021, $4.3, $8.63c, $0.0 are not accepted.
4. It is given an alphabet {0, 1}, design a DFA which recognizes words with an even number of 0’s and odd number of 1’s.
5. Construct DFA which accepts the following language:
$$L=\{w \mid w\in \text{ contains substring } 0101\}$$
6. Let M be the following DFA:



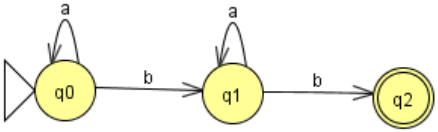
(i) Write down four string accepted by M and the sequence of configurations that shows this.
(ii) Write down four strings not accepted by M.

7. Which languages are accepted by the following automata:

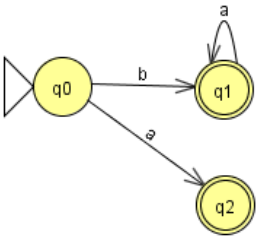
a)



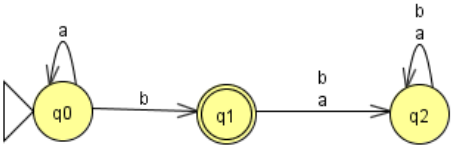
b)



c)



d)



2.1.2 Nondeterministic Finite Automata with ε transitions

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

- Q is a finite set of states.
- Σ is an input alphabet.
- δ: $Q \times (\Sigma \cup \{ \varepsilon \}) \rightarrow$ subsets of Q is a transition function.

- $q_0 \in Q$ is the initial state.
- $F \subseteq Q$ is a set of accepting states (or final states).

Differences from DFA:

- transition function δ can go into several states;
- it is allowed the ε -transitions.

-

Example:

Construct an NFA over alphabet {0, 1} that accepts those strings that contain the pattern 011 somewhere.

Answer:

-Analytical representation

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

$Q=\{q_0, q_1, q_2, q_3\}$; $\Sigma=\{0, 1\}$; $q_0=\{q_0\}$; $F=\{q_3\}$;

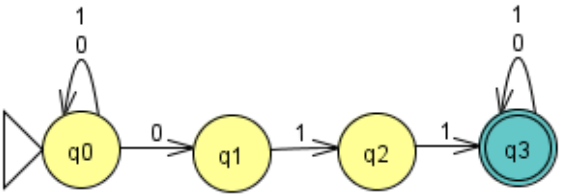
$\delta(q_0,0)=\{q_0\}$; $\delta(q_0,1)=\{q_0\}$; $\delta(q_0,0)=\{q_1\}$; $\delta(q_1,1)=\{q_2\}$;

$\delta(q_2,1)=\{q_3\}$; $\delta(q_3,0)=\{q_3\}$; $\delta(q_3,1)=\{q_3\}$.

- Tabel representation

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\varnothing$	$\{q_3\}$
$*q_3$	$\{q_3\}$	$\{q_3\}$

- Graphical representation



Notice that the only difference between an NFA and a DFA is in the type of value that δ returns: a set of states in case of an NFA and a single state in case of a DFA.

The NFA accepts string $x \in \Sigma^*$ if there is some path that, starting from q_0, leads to an accepting state as the string is read left to right. The language of an NFA is the set of all strings that the NFA accepts.

Example:

Construct an NFA-ε over alphabet {0, 1, 2…9, +, -, .} that accepts decimal numbers (example: -3.75, .462, 0.2)

Answer:

-Analytical representation

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

$Q=\{q_0, q_1, q_2, q_3, q_4, q_5\}$; $\Sigma=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, .\}$;

$q_0=\{q_0\}$; $F=\{q_5\}$;

$\delta(q_0, \varepsilon)=\{q_1\}$; $\delta(q_0, +)=\{q_1\}$; $\delta(q_0, -)=\{q_1\}$;

$\delta(q_1,0)=\{q_1\}$; $\delta(q_1,1)=\{q_1\}$; $\delta(q_1,2)=\{q_1\}$; $\delta(q_1,3)=\{q_1\}$;

$\delta(q_1,4)=\{q_1\}$; $\delta(q_1,5)=\{q_1\}$; $\delta(q_1,6)=\{q_1\}$; $\delta(q_1,7)=\{q_1\}$;

$\delta(q_1,8)=\{q_1\}$; $\delta(q_1,9)=\{q_1\}$; $\delta(q_1, .)=\{q_2\}$;

$\delta(q_1,0)=\{q_4\}$; $\delta(q_1,1)=\{q_4\}$; $\delta(q_1,2)=\{q_4\}$; $\delta(q_1,3)=\{q_4\}$;

$\delta(q_1,4)=\{q_4\}$; $\delta(q_1,5)=\{q_4\}$; $\delta(q_1,6)=\{q_4\}$; $\delta(q_1,7)=\{q_4\}$;

$\delta(q_1,8)=\{q_4\}$; $\delta(q_1,9)=\{q_4\}$; $\delta(q_4, .)=\{q_3\}$;

$\delta(q_2,0)=\{q_3\}$; $\delta(q_2,1)=\{q_3\}$; $\delta(q_2,2)=\{q_3\}$; $\delta(q_2,3)=\{q_3\}$;

$\delta(q_2,4)=\{q_3\}$; $\delta(q_2,5)=\{q_3\}$; $\delta(q_2,6)=\{q_3\}$; $\delta(q_2,7)=\{q_3\}$;

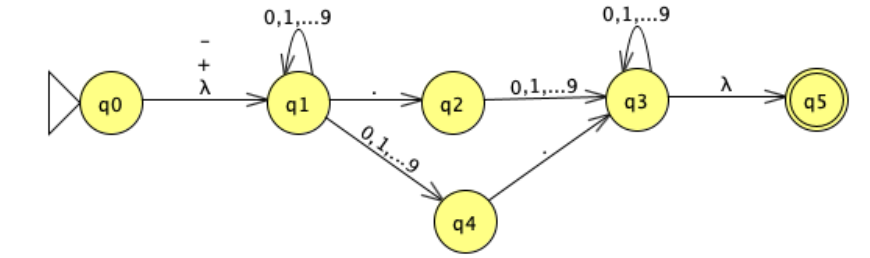
$\delta(q_2,8)=\{q_3\}$; $\delta(q_2,9)=\{q_3\}$;

$\delta(q_3,0)=\{q_3\}$; $\delta(q_3,1)=\{q_3\}$; $\delta(q_3,2)=\{q_3\}$; $\delta(q_3,3)=\{q_3\}$;

$\delta(q_3,4)=\{q_3\}$; $\delta(q_3,5)=\{q_3\}$; $\delta(q_3,6)=\{q_3\}$; $\delta(q_3,7)=\{q_3\}$;

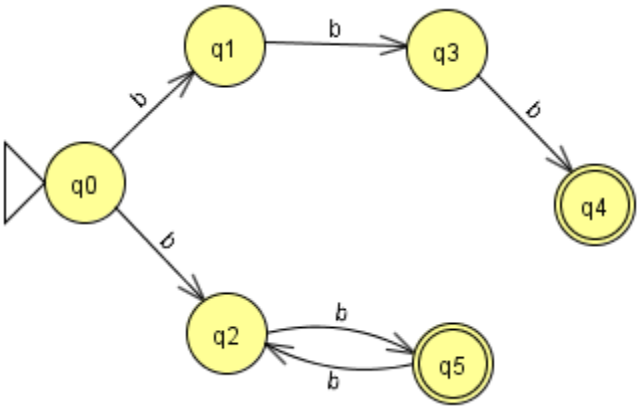
$\delta(q_3,8)=\{q_3\}$; $\delta(q_3,9)=\{q_3\}$; $\delta(q_3, \varepsilon)=\{q_5\}$;

- Graphical representation



Practical Tasks

1. Present the DFA that accepts the language definded by the following NFA:



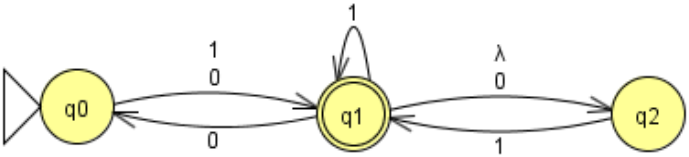
2. Present the NFA with four states that accepts the language: $L=\{ \varepsilon, aba, abcc\}$, $\Sigma=\{a, b, c\}$.

3. Present the NFA with no more than six states for the language: $L=\{abab^n | n>0\} \cup \{aab^n | n\geq 0\}$, $\Sigma=\{a, b\}$.

4. Find a NFA with four states for:

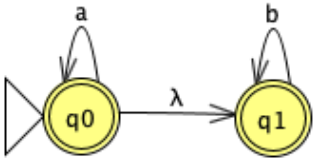
$$L=\{b^n | n\geq 0\} \cup \{b^n a | n\geq 1\}, \Sigma=\{a, b\}.$$

5. Which of the words 00, 01001, 10010, 000, 0000 are accepted by the following NFA:

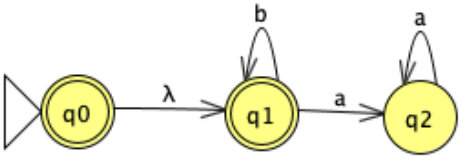


6. Which languages are accepted by the following automaton:

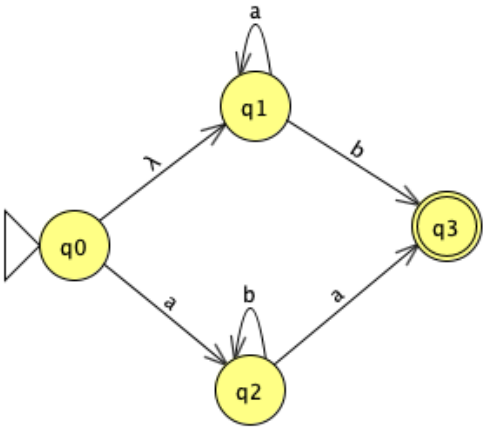
a)



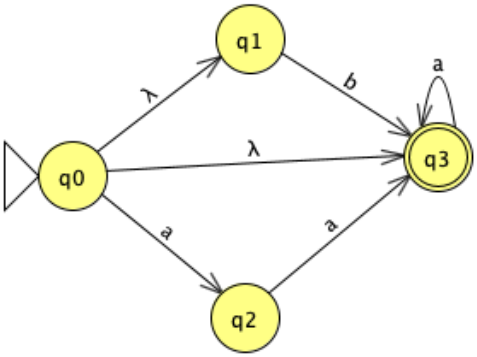
b)



c)



d)



2.1.3 Conversion NFA to DFA

It is given NFA $M=(Q, \Sigma, \delta, q_0, F)$ and it is necessary to be obtained the equivalent DFA $M'=(Q', \Sigma, \delta', q'_0, F')$.

Two automata are equivalent if $L(M)=L(M')$

Algorithm 1:

Step 1: It is initialized the initial state q_0 and this state is adding to the Q':

$$Q'=\varnothing,\ [q_0]'=q_0,\quad Q'=\{[q_0]\}.$$

Step 2: It is determinated the transition function by the following rule:

$$\delta(\{q_0, q_1, q_2, \ldots. q_n\}, a)=\bigcup_{i=1}^{n}\delta(q_1, a)$$

Step 3: For all states $[q_0, q_1, q_2, \ldots. q_n]\in Q'$ it is presented

$$\delta(\{q_0, q_1, q_2, \ldots. q_n\}, a)=\{p_0, p_1, p_2, \ldots. p_n\},$$

$\{p_0, p_1, p_2, \ldots. p_n\}$ are from Σ.

If $\{p_0, p_1, p_2, \ldots. p_n\}\notin Q'$, then it is included to the Q' and it is definded the se set δ.

Step 4: The step 3 it is repeated until occur the changes in Q.

Step 5: It is defined the set of final states

$$F'=\{[q_0, q_1, q_2, \ldots. q_n] \mid q_i,\quad 1\le i\le n,\ q_i\in F\}.$$

Algorithm 2:

Step 1 – It is created the state table for the given NFA.

Step 2 – It is created a blank state table under possible input alphabets for the equivalent DFA.

Step 3 – It is marked the start state of the DFA by $q0$ (same as the NDFA).

Step 4 – It is found out the combination of States $\{q_0, q_1, \ldots, q_n\}$ for each possible input alphabet.

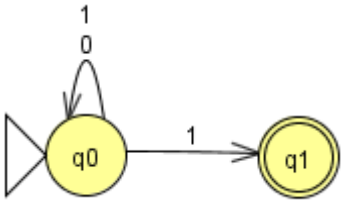
Step 5 − Each time it is generated the new DFA state under the input alphabet columns, it should be applied the step 4 again, otherwise go to step 6.

Step 6 − The states which contain any of the final states of the NFA are the final states of the equivalent DFA.

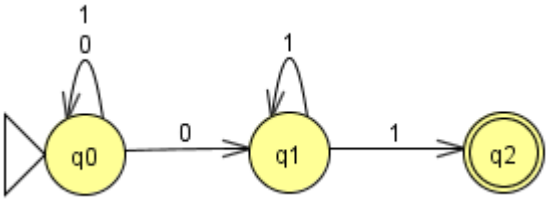
Practical Tasks

Convert the given NFA to DFA:

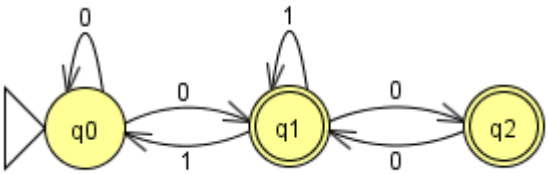
a)



b)



c)



2.1.4 Conversion ε - NFA to DFA

ε – Closure

Given a set of states *S*, the ε closure will give the set of states reachable from each state in *S* using only ε transitions.

Let it is given FA=(Q, Σ,q_0,δ,F) that is a ε-NFA:

– Let *S* be a subset of *Q*

– The ε closure, denotes ε - CLOSURE(*S*) is defined:

- For each state $p \in S$, $p \in$ ε - CLOSURE(*S*).
- For any $q \in$ ε - CLOSURE(S), every element of δ(q, ε) ∈ ε - CLOSE(*S*).
- No other elements of *Q* are in ε - CLOSURE(*S*).

• Algorithm for buid the ε-Closure:

– To find ε - CLOSURE(S) where S is a subset of *Q*

– Let $T=S$.

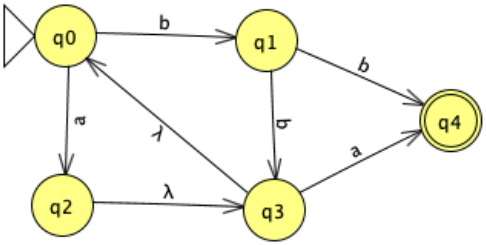
– While (T does not change) do Add all elements of δ(q, ε) where $q \in T$

– ε - CLOSURE(*S*) = *T*.

Example:

It is given ε-NFA represented in the graph form

M= (Q, Σ, δ,q_0, F)



ε – CLOSURE(q_0) ={q_0}

ε – CLOSURE(q_1) ={q_1}

ε – CLOSURE(q_2) ={ q_0, q_2, q_3}

ε – CLOSURE(q_3) ={ q_0, q_3}

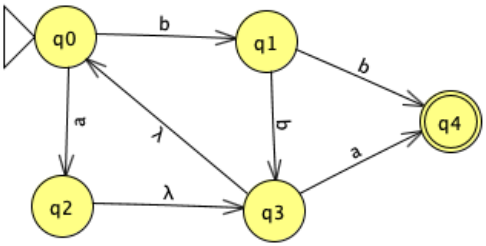
ε – CLOSURE(q_4) ={q_4}

Convertion NFA with ε-move to DFA :

Step 1 : It is obtained the ε – CLOSURE(q_0) for the beginning state of NFA as beginning state of DFA.
Step 2 : It is found the states that can be traversed from the present state for each input symbol (union of transition value and their closures for each states of NFA present in current state of DFA).
Step 3 : If any new state is found take it as current state and repeat step 2.
Step 4 : The Step 2 and Step 3 are repeated until no new state present in DFA transition table.
Step 5 : There are marked the states of DFA which contains final state of NFA as final states of DFA.

Example:
It is given ε-NFA represented in the graph form

M= (Q, Σ, δ,q_0, F)



ε – CLOSURE(q_0) ={q_0}

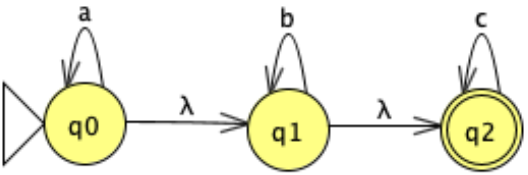
δ'	a	b
→{q_0}	{q_0, q_2, q_3}	{q_1}
{q_0, q_2, q_3}	{q_0, q_2, q_3, q_4}	{q_1}
{q_1}	∅	{q_0, q_3, q_4}
*{q_0, q_2, q_3, q_4}	{q_0, q_2, q_3, q_4}	{q_1}
*{q_0, q_3, q_4}	{q_0, q_2, q_3, q_4}	{q_1}

F={{q_0, q_3, q_4}{q_0, q_2, q_3, q_4}}

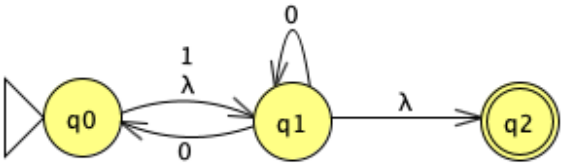
Practical Tasks

Convert the given ε – NFA to DFA:

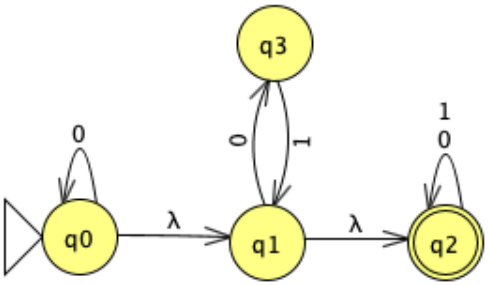
a)



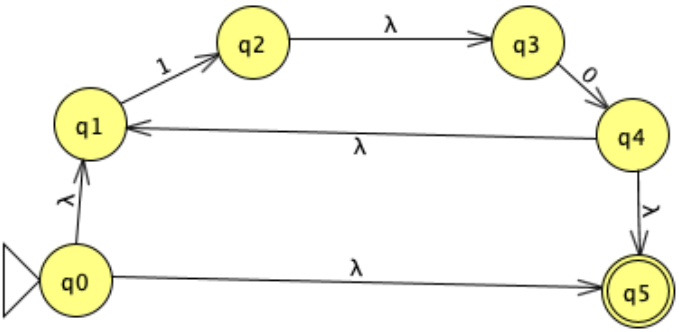
b)



c)



d)



2.1.5 Equivalence of the Finite Automaton with Regular Grammar

Theorem: For the given DFA=(Q, **Σ**, δ, q_0, F) is can be obtained the equivalent regular grammar G=(V_N, V_T, P, S).

Algorithm:

1. The set of non-terminal symbols is equal with set of states:

$$V_N = Q.$$

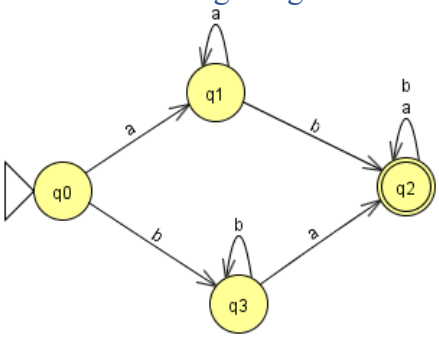
2. The set of terminal symbols is equal with set of states:

$$V_T = \Sigma.$$

3. Strat symbol is equal with q_0, S={q0}
4. For production P:
   a) $P = \varnothing$;
   b) For all values $\delta(q,a)=(q_1,q_2,\ldots q_m)$ we have:
   $P=P\cup\{q\text{->}aq_i \,|\, i=\overline{1,m}\ \}$
c) For all values $\delta(q,a)=(q_1,q_2,\ldots q_m)$, if
$F\cap\{q_1,q_2,\ldots q_m\}\neq\varnothing$ we have $P=P\cup\{q\rightarrow a\}$.

Example:

Convert the given DFA to the regular grammar:



Answer:

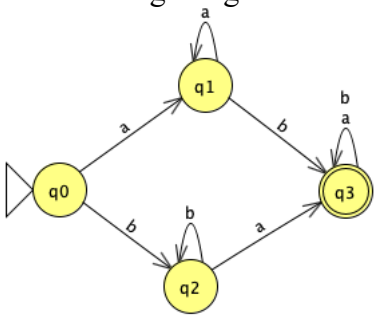
$G=(V_N, V_T, S, P)$, $V_N=\{q_0, q_1, q_2, q_3\}$, $S=\{q_0\}$, $V_T=\{a, b\}$,
$P=\{\ q_0\rightarrow aq_1|bq_3;$

$q_1 \rightarrow aq_1 | bq_2;$
$q_2 \rightarrow aq_2 | bq_2;$
$q_3 \rightarrow aq_2 | bq_3;$
$q_2 \rightarrow \varepsilon \}.$

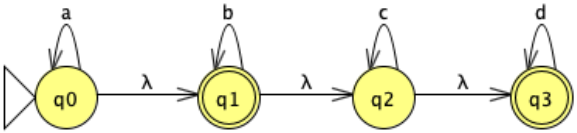
Practical Tasks

Convert the given FA to the regular grammar:

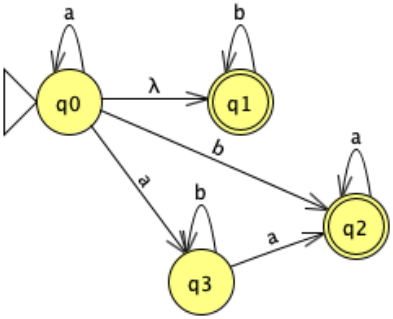
a)



b)



c)



2.1.6 Equivalence of the Regular Grammar with Finite Automata

Theorem: it is given $G=(V_N, V_T, P, S)$, where all productions are in the form

$$A \rightarrow aA$$

$$A \rightarrow a,\ A \in V_N,\ a \in V_T$$

For every regular grammar can be obtained equivalent finite automaton.

Algorithm:

1. Input alphabet is equal with set of terminal symbols

$$\Sigma = V_T.$$

2. The set of state is equal with set of non-terminal symbols and one additional state $Q = V_N \cup \{X\}$, where X is a new element, $X \in V_N$.
3. Initial state $q_0=\{S\}$, S- initial state.
4. Final state $F=\{X\}$.
5. It is build the set δ:

For all productions, which have representation A->aB, we obtain:

$$\delta(A, a)=\delta(A,a)\cup\{B\}.$$

For all productions which have representation A-> a, we obtain:

$$\delta(A, a)=\delta(A,a)\cup \{X\}.$$

Example:

Convert the given regular grammar to the DFA:

$G= (V_N, V_T, S, P)$, $V_N = \{S, A, B\}$, $V_T = \{a, b\}$,

$P=\{S \rightarrow aA;$

$A \rightarrow aB|a;$

$B \rightarrow bB|b\}.$

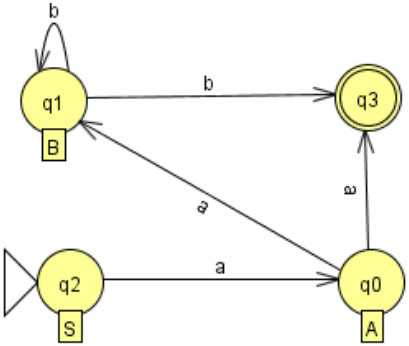
Answer:

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

$Q=\{q_0, q_1, q_2, q_3\}$; $\Sigma=\{a, b\}$; $q_0=\{q_2\}$; $F=\{q_3\}$;

---

q_0={A}; q_1={B}; q_2={S};
$\delta(q_0,a)=\{q_3\}$; $\delta(q_0,a)=\{q_1\}$; $\delta(q_1,b)=\{q_1\}$; $\delta(q_1,b)=\{q_3\}$;
$\delta(q_2,a)=\{q_0\}$.



Practical Tasks

Convert the given regular grammar to the FA

a) $G=(V_N, V_T, S, P)$, $V_N=\{S, A, B\}$, $V_T=\{a, b\}$,
$P=\{S\rightarrow aA; A\rightarrow a|\ \varepsilon\}$.

b) $G=(V_N, V_T, S, P)$, $V_N=\{S, A, B\}$, $V_T=\{a, b\}$,
$P=\{S\rightarrow A; A\rightarrow aA|aB|bB|a; B\rightarrow bB|b\}$.

c) $G=(V_N, V_T, S, P)$, $V_N=\{S, T\}$, $V_T=\{0, 1\}$,
$P=\{S\text{->}1S|0T|\ \varepsilon; T\text{->}1T|0S\}$.

2.1.7 Minimization of the DFA

In automata theory (a branch of theoretical computer science), DFA minimization is the task of transforming a given deterministic finite automaton (DFA) into an equivalent DFA that has a minimum number of states. Here, two DFAs are called equivalent if they recognize the same regular language.

There are two classes of states that can be removed or merged from the original DFA without affecting the language it accepts to minimize it.

- **Unreachable states** are the states that are not reachable from the initial state of the DFA, for any input string.
- **Nondistinguishable** states are those that cannot be distinguished from one another for any input string.

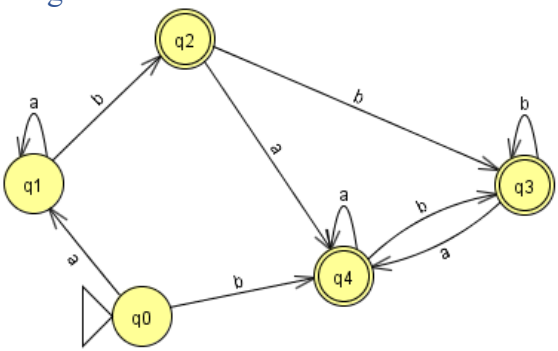
Step 5: All states of one set are merged into one. Number of states in minimized DFA will be equal to number of sets in P_k.

Two states in partition P_k are distinguishable by the following rule:

Two states (q_i, q_j) are distinguishable in partition P_k if for any input symbol a, $\delta(q_i, a)$ and $\delta(q_j, a)$ are in different sets in partition P_{k-1}.

Example:

Minimize the given DFA:



Answer:

$\boldsymbol{P_1}$**={{**q_0**,** q_1**},{**q_2**,** q_3**,** q_4**}}.**

Step 1. P_0 will have two sets of states. One set will contain q_2, q_3, q_4 which are final states of DFA and another set will contain remaining states. So $\boldsymbol{P_0}$**={{**q_0**,** q_1**},{**q_2**,** q_3**,** q_4**}}.**

Step 2. To calculate P_1, it will be checked whether sets of partition P_0 can be partitioned or not:

---

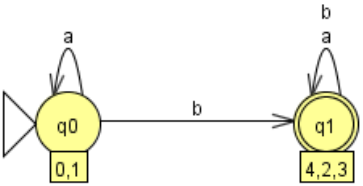
i) For set { q_0, q_1}:
$\delta(q_0, a) = \{q_1\}$ and $\delta(q_1, a) = q_1$;
$\delta(q_0, b) = \{q_4\}$ and $\delta(q_1, b) = q_2$.
Moves of q_0 and q_1 on input symbol a are in q_1 respectively which are in same set in partition P_0. Similarly, moves of q_0 and q_1 on input symbol b are q_4 and q_2 which are in same set in partition P_0. So, q_0 and q_1 are not distinguishable.

ii) For set {q_2, q_3, q_4}:
$\delta(q_2, a) = \{q_4\}$ and $\delta(q_3, a) = q_4$;
$\delta(q_2, b) = \{q_3\}$ and $\delta(q_3, b) = q_3$.
Moves of q_2 and q_3 on input symbol a are in q_4 respectively which is in same set in partition P_0. Similarly, moves of q_2 and q_3 on input symbol b are q_3 which is in same set in partition P_0. So, q_2 and q_3 are not distinguishable.

$\delta(q_2, a) = \{q_4\}$ and $\delta(q_4, a) = q_4$;
$\delta(q_2, b) = \{q_3\}$ and $\delta(q_4, b) = q_3$.
Moves of q_2 and q_4 on input symbol a are in q_4 respectively which is in same set in partition P_0. Similarly, moves of q_2 and q_4 on input symbol b are q_3 which is in same set in partition P_0. So, q_2 and q_4 are not distinguishable.

$\delta(q_3, a) = \{q_4\}$ and $\delta(q_4, a) = q_4$;
$\delta(q_3, b) = \{q_3\}$ and $\delta(q_4, b) = q_3$.
Moves of q_3 and q_4 on input symbol a are in q_4 respectively which is in same set in partition P_0. Similarly, moves of q_3 and q_4 on input symbol b are q_3 which is in same set in partition P_0. So, q_3 and q_4 are not distinguishable.

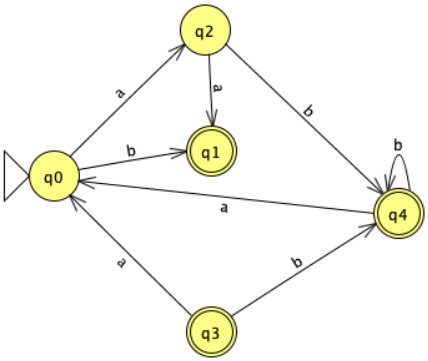
So, $\boldsymbol{P_1}$**=**$\{\{q_0, q_1\},\{q_2, q_3, q_4\}\}$**.**
And we have $P_0 = P_1$, in this case the algorithm is stop.
In this case minimized DFA corresponding to the given DFA is presented below:



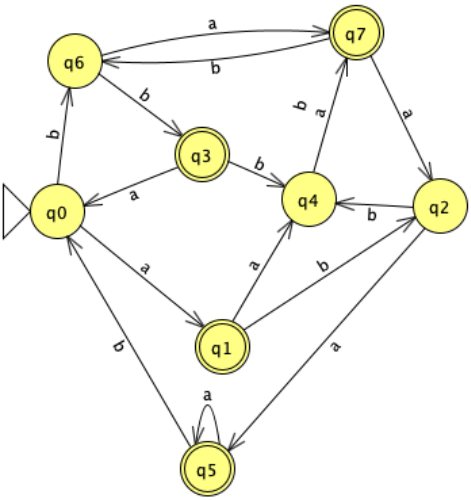
Practical Tasks

Minimize the given DFA:

a)



b)



c)