Regular Grammar

Raju Hazari

Department of Computer Science and Engineering
National Institute of Technology Calicut

September 28, 2020

Regular Grammar

- A language is regular if and only if there exists some deterministic finite automata.
- Another way of describing regular language is by means of certain grammars.
- Grammars are often an alternative way of specifying languages.
- So, regular grammars are associated with regular languages and that for every regular language there is regular grammar.

Regular Grammar

- A grammar $G = (N, T, S, P)$ is said to be **right-linear** if all productions are of the form
$$A \to xB$$
$$A \to x$$
where $A, B \in N$, and $x \in T^*$.

- A grammar is said to be **left-linear** if all the productions are of the form
$$A \to Bx$$
$$A \to x$$

Regular Grammar

- A grammar $G = (N, T, S, P)$ is said to be **right-linear** if all productions are of the form
$$A \to xB$$
$$A \to x$$
where $A, B \in N$, and $x \in T^*$.

- A grammar is said to be **left-linear** if all the productions are of the form
$$A \to Bx$$
$$A \to x$$

- A **regular grammar** is one that is either right-linear or left linear.
  - In a regular grammar, at most one non-terminal appears on the right side of any production.
  - The non-terminal must consistently be either the rightmost or leftmost symbol of the right side of any production.

Regular Grammar

Example 1:

- The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as

  $$S \to abS|a$$

  So, it is a right-linear grammar.

Regular Grammar

Example 1:

- The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as

$$S \rightarrow abS|a$$

So, it is a right-linear grammar.

- The grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 given as

$$\begin{aligned} S &\rightarrow S_1 ab, \\ S_1 &\rightarrow S_1 ab | S_2, \\ S_2 &\rightarrow a \end{aligned}$$

So, it is a left-linear grammar.

Regular Grammar

Example 1:

- The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as
$$S \to abS|a$$
So, it is a right-linear grammar.

- The grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 given as
$$\begin{aligned} S &\to S_1ab, \\ S_1 &\to S_1ab|S_2, \\ S_2 &\to a \end{aligned}$$
So, it is a left-linear grammar.

- Both G_1 and G_2 are regular grammars.

Regular Grammar

Example 1:

- The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as
$$S \rightarrow abS|a$$
So, it is a right-linear grammar.

- The grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 given as
$$\begin{aligned} &S \rightarrow S_1 ab, \\ &S_1 \rightarrow S_1 ab | S_2, \\ &S_2 \rightarrow a \end{aligned}$$
So, it is a left-linear grammar.

- Both G_1 and G_2 are regular grammars.

- The sequence $S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$ is a derivation with G_1. So, the language generated by the grammar $L = \{(ab)^n a | n \geq 0\}$.

Regular Grammar

Example 1:

- The grammar $G_1 = (\{S\}, \{a, b\}, S, P_1)$, with P_1 given as
$$S \rightarrow abS|a$$
So, it is a right-linear grammar.

- The grammar $G_2 = (\{S, S_1, S_2\}, \{a, b\}, S, P_2)$, with P_2 given as
$$\begin{aligned} &S \rightarrow S_1ab, \\ &S_1 \rightarrow S_1ab|S_2, \\ &S_2 \rightarrow a \end{aligned}$$
So, it is a left-linear grammar.

- Both G_1 and G_2 are regular grammars.
- The sequence $S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa$ is a derivation with G_1. So, the language generated by the grammar $L = \{(ab)^n a | n \geq 0\}$.
- The sequence $S \Rightarrow S_1ab \Rightarrow S_1abab \Rightarrow S_1ababab \Rightarrow S_2ababab \Rightarrow aababab$ is a derivation with G_2. So, the language generated by the grammar $L = \{aab(ab)^n | n \geq 0\}$

Regular Grammar

Example 2:

- The grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\rightarrow A, \\ A &\rightarrow aB|a, \\ B &\rightarrow Ab \end{aligned}$$

Regular Grammar

Example 2:

- The grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions

$$S \to A,$$
$$A \to aB|a,$$
$$B \to Ab$$

So, it is not regular.

Regular Grammar

Example 2:

- The grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\to A, \\ A &\to aB|a, \\ B &\to Ab \end{aligned}$$

  So, it is not regular.

- Although every production is either in right-linear or left-linear form, the grammar itself is neither right-linear nor left-linear, and therefore is not regular.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

The construction is like this-

- $Q = N \cup \{q_f\}$
  - Each non-terminal will correspond to a state in Q, apart from that there is one more final state.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

The construction is like this-

- $Q = N \cup \{q_f\}$
  - Each non-terminal will correspond to a state in Q, apart from that there is one more final state.
- $\Sigma = T$
  - Terminal symbols are the input symbols for the automaton.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

The construction is like this-

- $Q = N \cup \{q_f\}$
  - Each non-terminal will correspond to a state in Q, apart from that there is one more final state.
- $\Sigma = T$
  - Terminal symbols are the input symbols for the automaton.
- $q_0 = S$
  - Initial state corresponds to the start symbol.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

The construction is like this-

- $Q = N \cup \{q_f\}$
  - Each non-terminal will correspond to a state in Q, apart from that there is one more final state.
- $\Sigma = T$
  - Terminal symbols are the input symbols for the automaton.
- $q_0 = S$
  - Initial state corresponds to the start symbol.
- $F = \{q_f\}$
  - F consists of just q_f.

Equivalence between Finite Automaton and Regular Grammar

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

The construction is like this-

- $Q = N \cup \{q_f\}$
  - Each non-terminal will correspond to a state in Q, apart from that there is one more final state.
- $\Sigma = T$
  - Terminal symbols are the input symbols for the automaton.
- $q_0 = S$
  - Initial state corresponds to the start symbol.
- $F = \{q_f\}$
  - F consists of just q_f.
- δ is define as follows:
  - If $A \to aB$ is a rule in P, then $\delta(A, a)$ contains B.
  - If $A \to a$ is in P, then $\delta(A, a)$ contains q_f.

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\rightarrow aS, \\ S &\rightarrow aA, \\ A &\rightarrow bA, \\ A &\rightarrow b \end{aligned}$$

  Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A\}, \{a, b\}, S, P)$ with productions

$$
\begin{aligned}
S &\to aS,\\
S &\to aA,\\
A &\to bA,\\
A &\to b
\end{aligned}
$$

  Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

- The language generated by the grammar:

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\to aS, \\ S &\to aA, \\ A &\to bA, \\ A &\to b \end{aligned}$$

  Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

- The language generated by the grammar: $L = \{a^n b^m | n, m \geq 1\}$

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A\}, \{a, b\}, S, P)$ with productions

$$\begin{aligned} S &\rightarrow aS, \\ S &\rightarrow aA, \\ A &\rightarrow bA, \\ A &\rightarrow b \end{aligned}$$

  Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.
- The language generated by the grammar:$L = \{a^n b^m | n, m \geq 1\}$
- Use the construction, we can construct the NFA

Equivalence between Finite Automaton and Regular Grammar

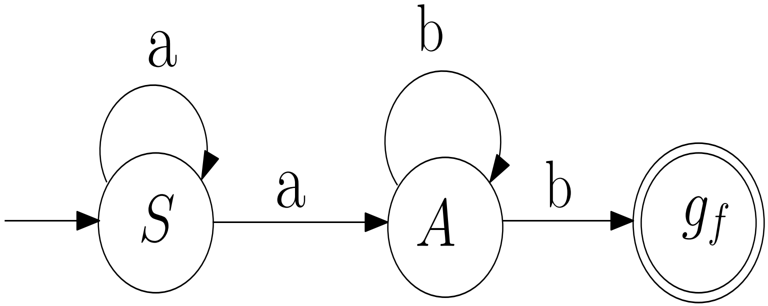
- Consider the grammar $G = (\{S, A\}, \{a, b\}, S, P)$ with productions

$$S \to aS, \\ S \to aA, \\ A \to bA, \\ A \to b$$

  Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

- The language generated by the grammar:$L = \{a^n b^m | n, m \geq 1\}$

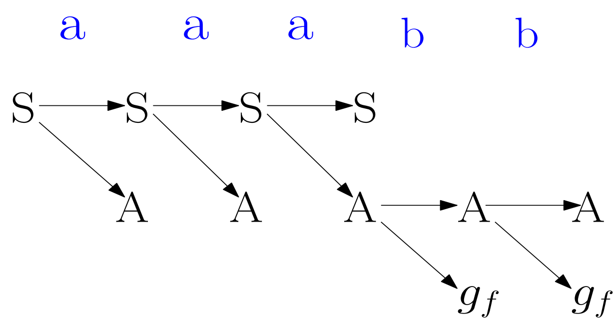
- Use the construction, we can construct the NFA



It is also accept the same language L.

Equivalence between Finite Automaton and Regular Grammar

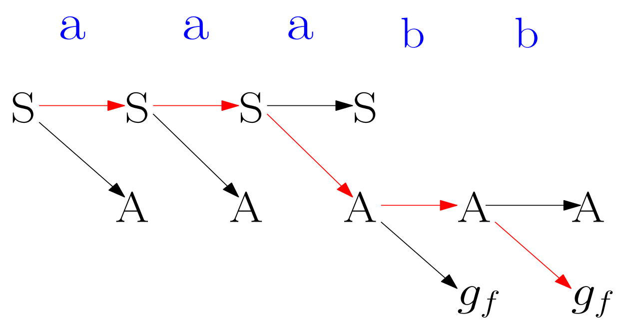
- Let us consider a string $aaabb$. How this string is generated by the grammar G and accept by the NFA M ?
- The sequence $S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaabA \Rightarrow aaabb$
- The state sequence for NFA



- So, the way it generated, it is accepted in the same sequence.

Equivalence between Finite Automaton and Regular Grammar

- Let us consider a string $aaabb$. How this string is generated by the grammar G and accept by the NFA M ?
- The sequence $S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaA \Rightarrow aaabA \Rightarrow aaabb$
- The state sequence for NFA



- So, the way it generated, it is accepted in the same sequence.

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions
$$\begin{aligned} S &\to aA, \\ S &\to bA, \\ A &\to aB, \\ A &\to bB, \\ B &\to aS, \\ B &\to bS, \\ B &\to a, \\ B &\to b \end{aligned}$$
Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions
$$\begin{aligned} &S \to aA, \\ &S \to bA, \\ &A \to aB, \\ &A \to bB, \\ &B \to aS, \\ &B \to bS, \\ &B \to a, \\ &B \to b \end{aligned}$$
Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.
- The language generated by the grammar:

Equivalence between Finite Automaton and Regular Grammar

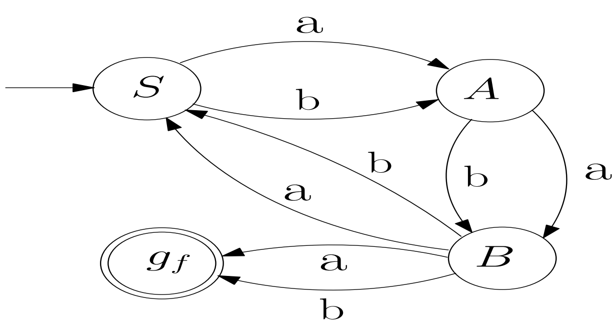
- Consider the grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions
$$S \to aA,$$
$$S \to bA,$$
$$A \to aB,$$
$$A \to bB,$$
$$B \to aS,$$
$$B \to bS,$$
$$B \to a,$$
$$B \to b$$
Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.
- The language generated by the grammar: $L = \{w | w \in \{a, b\}^*$, length of w is multiple of 3$\}$

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions
$$\begin{aligned} S &\to aA, \\ S &\to bA, \\ A &\to aB, \\ A &\to bB, \\ B &\to aS, \\ B &\to bS, \\ B &\to a, \\ B &\to b \end{aligned}$$
Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.
- The language generated by the grammar:$L = \{w | w \in \{a, b\}^*$, length of w is multiple of 3$\}$
- Use the construction, we can construct the NFA

Equivalence between Finite Automaton and Regular Grammar

- Consider the grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions
$$S \to aA,$$
$$S \to bA,$$
$$A \to aB,$$
$$A \to bB,$$
$$B \to aS,$$
$$B \to bS,$$
$$B \to a,$$
$$B \to b$$
Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.
- The language generated by the grammar: $L = \{w | w \in \{a, b\}^*$, length of w is multiple of 3$\}$
- Use the construction, we can construct the NFA



From this, it is clear that the language generated by the grammar and the language accepted by the NFA are same.

Equivalence between Finite Automaton and Regular Grammar

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

Equivalence between Finite Automaton and Regular Grammar

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

The construction is like this-

- $N = Q$
  - For each state, there is a non-terminal.

Equivalence between Finite Automaton and Regular Grammar

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

The construction is like this-

- $N = Q$
  - For each state, there is a non-terminal.
- $T = \Sigma$
  - Set of terminal symbols are the same as the set of input symbols.

Equivalence between Finite Automaton and Regular Grammar

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

The construction is like this-

- $N = Q$
  - For each state, there is a non-terminal.
- $T = \Sigma$
  - Set of terminal symbols are the same as the set of input symbols.
- $S = q_0$
  - S corresponds to the initial state.

Equivalence between Finite Automaton and Regular Grammar

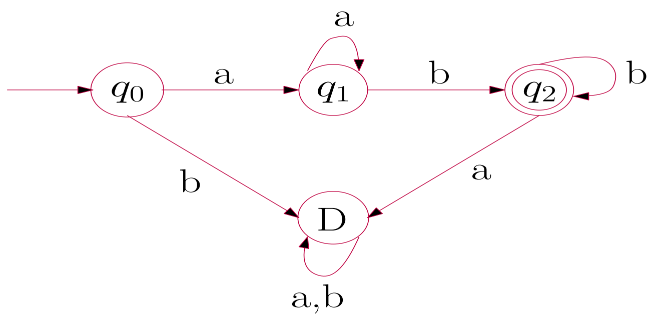
Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

The construction is like this-

- $N = Q$
  - For each state, there is a non-terminal.
- $T = \Sigma$
  - Set of terminal symbols are the same as the set of input symbols.
- $S = q_0$
  - S corresponds to the initial state.
- P is define as follows:
  - If $\delta(A, a) = B$, then $A \to aB \in P$.
  - If $\delta(A, a) = B \wedge B \in F$, then $A \to a$.

Equivalence between Finite Automaton and Regular Grammar

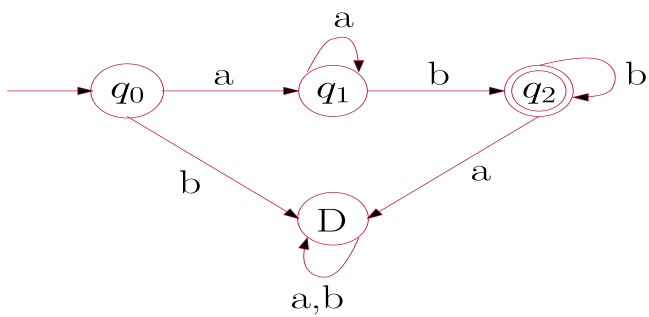
- Consider the DFA



Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$.

Equivalence between Finite Automaton and Regular Grammar

- Consider the DFA

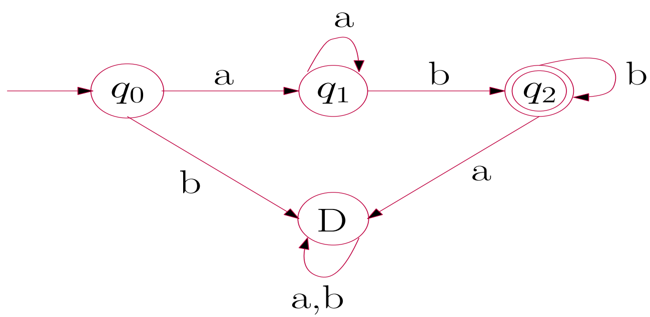


Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$.

- The language accepted by the DFA $L = \{a^n b^m | n, m \geq 1\}$

Equivalence between Finite Automaton and Regular Grammar

- Consider the DFA



Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$.

- The language accepted by the DFA $L = \{a^n b^m | n, m \geq 1\}$
- The corresponding grammar $N = \{q_0, q_1, q_2, D\}$, $T = \{a, b\}$, $S = q_0$ and P:

$$q_0 \to aq_1$$
$$q_0 \to bD$$
$$q_1 \to aq_1$$
$$q_1 \to bq_2$$
$$q_2 \to bq_2$$
$$q_2 \to aD$$
$$D \to aD$$
$$D \to bD$$

These are non-terminal rules. Now, whenever you have q_2, you have terminal rule. So, here $q_1 \to b$ and $q_2 \to b$ are the terminal rules.

Equivalence between Finite Automaton and Regular Grammar

- From D you cannot drive a terminal string. So, all these D is a useless non-terminal, all these rules can be removed. So we will end with only six rules.

$$\begin{aligned}
q_0 &\rightarrow aq_1 \\
q_1 &\rightarrow aq_1 \\
q_1 &\rightarrow bq_2 \\
q_2 &\rightarrow bq_2 \\
q_1 &\rightarrow b \\
q_2 &\rightarrow b
\end{aligned}$$

- You can generate one a by $q_0 \rightarrow aq_1$ and any number of a by $q_1 \rightarrow aq_1$. Similarly, you can generate one b by $q_1 \rightarrow bq_2$ and any number of b by $q_2 \rightarrow bq_2$.

- So, the language generated by the grammar is a same as the language accepted by the machine.

Equivalence between Finite Automaton and Regular Grammar

Now we consider ϵ, means $\epsilon \in L$

- For any grammar, to include ϵ in a language do the following:
  - The rule $S \rightarrow \epsilon$ can be applied in the first step only and make sure S does not appear on the right hand side of the any production.

Equivalence between Finite Automaton and Regular Grammar

Now we consider ϵ, means $\epsilon \in L$

- For any grammar, to include ϵ in a language do the following:
  - The rule $S \to \epsilon$ can be applied in the first step only and make sure S does not appear on the right hand side of the any production.

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

Equivalence between Finite Automaton and Regular Grammar

Now we consider ϵ, means $\epsilon \in L$

- For any grammar, to include ϵ in a language do the following:
  - The rule $S \rightarrow \epsilon$ can be applied in the first step only and make sure S does not appear on the right hand side of the any production.

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

- For any string $L - \epsilon$, will be accepted the usual way we constructed earlier. For ϵ, make initial state a final state. So, ϵ will be accepted.

Now we consider ϵ, means $\epsilon \in L$

- For any grammar, to include ϵ in a language do the following:
  - The rule $S \to \epsilon$ can be applied in the first step only and make sure S does not appear on the right hand side of the any production.

Given a regular grammar $G = (N, T, S, P)$. Construct an NFA $M = (Q, \Sigma, \delta, q_0, F)$, such that $L(M) = L(G)$.

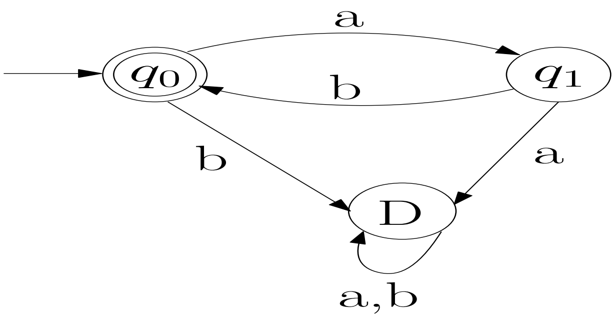
- For any string $L - \epsilon$, will be accepted the usual way we constructed earlier. For ϵ, make initial state a final state. So, ϵ will be accepted.

Given a DFA $M = (Q, \Sigma, \delta, q_0, F)$. Construct a regular grammar $G = (N, T, S, P)$, such that $L(G) = L(M)$

- You have just add the rule $q_0 \to \epsilon$. When you add this rule, you have to make sure that q_0 does not appear on the right hand side of any production. If it is happen, you have to make a slight adjustment.

Equivalence between Finite Automaton and Regular Grammar

Consider the DFA



- $N = \{q_0, q_1, D\},\ T = \{a, b\},\ S = q_0,\ P$:

$$
\begin{aligned}
q_0 &\to aq_1\\
q_0 &\to bD\\
q_1 &\to bq_0\\
q_1 &\to aD\\
D &\to aD\\
D &\to bD\\
q_1 &\to b\\
q_0 &\to \epsilon
\end{aligned}
$$

Equivalence between Finite Automaton and Regular Grammar

- From D we can not drive a terminal string. So, all these D is a useless non-terminal, all these rule can be removed.
- So, we get only four rules

$$
\begin{aligned}
&q_0 \rightarrow aq_1 \\
&q_1 \rightarrow bq_0 \\
&q_1 \rightarrow b \\
&q_0 \rightarrow \epsilon
\end{aligned}
$$

- Now, the start symbol q_0 and $q_0 \rightarrow \epsilon$, but it is occurring on the right hand side. In order to avoid that, add a new start symbol S and $S \rightarrow \epsilon$ and remove $q_0 \rightarrow \epsilon$. Then, whatever is there with q_0, also have with S.

$$
\begin{aligned}
&q_0 \rightarrow aq_1 \\
&q_1 \rightarrow bq_0 \\
&q_1 \rightarrow b \\
&S \rightarrow \epsilon \\
&S \rightarrow aq_1
\end{aligned}
$$