Theory of Formal Languages and Automata

Lecture 5

Mahdi Dolati

Sharif University of Technology

Fall 2025

February 24, 2025

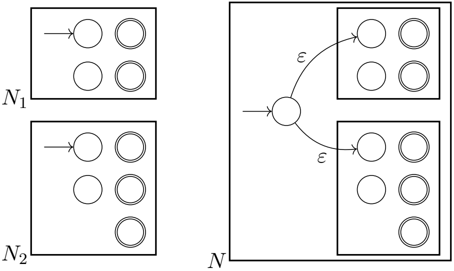
Closure

Theorem

The class of regular languages is closed under the union operation.

Proof idea:

- Consider two regular languages A_1 and A_2 with NFAs N_1 and N_2, respectively.
- Construct machine N with a new start state that has two ε arrows to start states of N_1 and N_2.



Closure

Proof.

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$:

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$.
2. Start state: q_0.
3. $F = F_1 \cup F_2$.
4. $q \in Q$ and $a \in \Sigma_\varepsilon$:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \\ \delta_2(q,a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases} \tag{1}$$

□

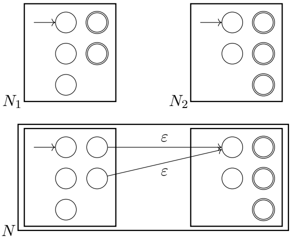
Closure

Theorem

The class of regular languages is closed under the concatenation operation.

Proof idea:

- Consider two regular languages A_1 and A_2 with NFAs N_1 and N_2, respectively.
- Construct machine N, where its start is the start of N_1. Connect accept states of N_1 to start of N_2 with ε arrows. Only accept states of N_2 are accept states in N.



Proof.

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2.
Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$:

1. $Q = Q_1 \cup Q_2$.
2. Start state: q_1.
3. Accept states: F_2.
4. $q \in Q$ and $a \in \Sigma_\varepsilon$:

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases} \tag{2}$$

$\square$

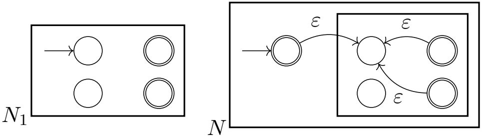
Closure

Theorem

The class of regular languages is closed under the star operation.

Proof idea:

- Consider a regular languages A_1 with NFA N_1.
- Construct machine N that accepts when its input consists of several pieces acceptable by N_1. Return from accept states with ε arrows to the start state of N_1. Also, add a new start state that is also an accept state to accept the empty string.



Proof.

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*:

1. $Q = \{q_0\} \cup Q_1$.
2. Start state: q_0.
3. Accept states: $\{q_0\} \cup F_1$.
4. $q \in Q$ and $a \in \Sigma_\varepsilon$:

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases} \tag{3}$$

□

Closure

- More operators for languages:
  - Prefix:

$$Prefix(A) = \{w \mid w \text{ is a prefix of a string in } A\}, \tag{4}$$

  - Suffix:

$$Suffix(A) = \{w \mid w \text{ is a suffix of a string in } A\}, \tag{5}$$

  - Right quotient of a language A by a string $u \in \Sigma^*$: Every string that becomes a member of A if concatenated by u:

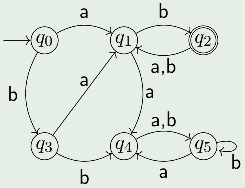
$$Au^{-1} = \{w \mid w \in \Sigma^* \text{ and } wu \in A\}, \tag{6}$$

  - Left quotient of a language A by a string u:

$$u^{-1}A = \{w \mid w \in \Sigma^* \text{ and } uw \in A\}, \tag{7}$$

Closure

Example



Strings in L(M):

- ab
- bab
- abab
- abbb
- babab
- babbb

Strings in Prefix(L(M)):

- ε
- b
- a
- ab
- ba
- bab

Theorem

The class of regular languages is closed under Prefix.

Proof idea:

- Given a DFA M_1 that recognizes a regular language L,
- We should build another DFA M_2 that recognizes Prefix(L).
- A string w is in L, M reads it and ends up in a state that can reach a final state,
- Thus, we may turn all states in M that can reach a final state into a final state to recognize Prefix(L).

Closure

Proof.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be a DFA that recognizes the regular language L.

Construct $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$:

- $Q_2 = Q_1$, $\delta_2 = \delta_1$ and $q_2 = q_1$.
- $F_2 = \{q \mid q \in Q_1$ where there is a path from q to a state in $F_1\}$.

Consider a string $w \in L(M_2)$ that takes M_2 to state $q \in F_2$. Consider the path from q to a state in F_1, which exists by definition. Build a string x from labels of arrows in that path. Thus, $wx \in L_1$, which implies that $w \in Prefix(L_1)$.

Consider a string $wx \in L(M_1)$ where $w, x \in \Sigma^*$. String w takes M_1 to a state q from which x can reach a state in F_1. Thus, $q \in F_2$ that implies $w \in L(M_2)$. □

Closure

Theorem

The class of regular languages is closed under Suffix.

Closure

Theorem

The class of regular languages is closed under left quotient.

Proof.

Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ be a DFA, $A = L(M_1)$ and $u \in \Sigma^*$. We build a DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ that recognizes:

$$u^{-1}A = \{x \mid x \in \Sigma^* \text{ and } ux \in A\}. \tag{8}$$

Let $Q_2 = Q_1$, $\delta_2 = \delta_1$ and $F_2 = F_1$:

- Let q_2 be the state that M_1 enter after reading u. Any string $x \in \Sigma^*$ that takes M_1 from q_2 to a state in F_1 is in $u^{-1}A$. Choosing q_2 to be the initial state of M_2 ensures all such strings are in the $L(M_2)$.
- If a string x takes M_2 from q_2 to a final state, then ux is in the $L(M_1)$.

□

Regular Grammars

- Another way of describing regular languages,
  - DFAs and NFAs, and
  - Regular Expressions.

Regular Grammars

- A grammar is regular if it is right- or left-linear.

Definition

A grammar $G = (V, T, S, P)$ is right-linear if all rules are of the form:

$$A \to xB,$$
$$A \to x,$$

where, $A, B \in V$, and $x \in T^*$.

Definition

A grammar $G = (V, T, S, P)$ is left-linear if all rules are of the form:

$$A \to Bx,$$
$$A \to x,$$

where, $A, B \in V$, and $x \in T^*$.

Regular Grammars

Example

G_1: A right-linear grammar,

$$S \to abS \mid a$$

A derivation:

$$S \Rightarrow abS \Rightarrow ababS \Rightarrow ababa.$$

Language of G_1 is described by $(ab)^*a$.

Regular Grammars

- A special form:

$$A \to abcB \qquad \begin{aligned} A &\to aB_1 \\ B_1 &\to bB_2 \\ B_2 &\to cB \end{aligned}$$

$$A \to abc \qquad \begin{aligned} A &\to aB_1 \\ B_1 &\to bB_2 \\ B_2 &\to c \end{aligned}$$

Regular Grammars

Example

G_2: A left-linear grammar,

$$
\begin{aligned}
S &\to S_1 ab, \\
S_1 &\to S_1 ab \mid S_2, \\
S_2 &\to a,
\end{aligned}
$$

$$S \Rightarrow S_1 ab \Rightarrow S_1 abab \Rightarrow S_2 abab \Rightarrow aababab.$$

Language of G_2 is described by $aab(ab)^*$.

- Linear grammar:
  - Exactly one variable in the left-hand side of productions,
  - At most one variable in the right-hand side of productions,
  - No restriction on the position of the variable in the right-hand side of productions relative to terminals.
- Regular grammars are linear grammars, but not vice versa.

Example

Grammar G_3:

$$\begin{aligned} S &\rightarrow A, \\ A &\rightarrow aB \mid \varepsilon, \\ B &\rightarrow Ab, \end{aligned}$$

Theorem

If grammar $G = (V, T, S, P)$ is right-linear, then $L(G)$ is a regular language.

Proof idea:

- Derivations have the special form:

$$ab \dots cD,$$

- Using a rule $D \to dE$ yields,

$$ab \dots cD \Rightarrow ab \dots cdE,$$

- States correspond to variables,
- Input correspond to terminal prefix.

Regular Grammars

Proof.

Let $V = \{V_0, V_1, \dots\}$ and $S = V_0$. Also, rules are in form $V_i \to v_j V_k$ and $V_n \to v_l$. Construct a NFA $M = (\{V_i\} \cup \{V_f\}, T_\varepsilon, \delta, V_0, \{V_f\})$:

Rule	Transition
$V_i \to v_j V_k$	$\delta(V_i, v_j)$ contains V_k
$V_n \to v_l$	$\delta(V_n, v_l)$ contains V_f

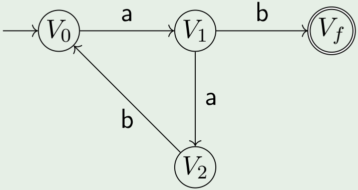
- $w \in L(G)$: Derivation of w is equivalent to a transition from the start state to a final state,
- $w \in L(M)$: The transition from the start state to the final state is equivalent to a derivation in G.

$\square$

Example

Construct a NFA for:

$$\begin{aligned} V_0 &\to aV_1, \\ V_1 &\to abV_0 \mid b. \end{aligned} \quad \xRightarrow{\text{transform}} \quad \begin{aligned} V_0 &\to aV_1, \\ V_1 &\to aV_2 \mid b, \\ V_2 &\to bV_0. \end{aligned}$$



Regular expression: $(aab)^*ab$.

Theorem

If L is a regular language on Σ, then there exists a right-linear grammar G such that $L = L(G)$.

Proof.

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA that accepts L. Assume:

$$Q = \{q_0, q_1, \ldots, q_n\}$$
$$\Sigma = \{a_1, a_2, \ldots, a_m\}.$$

Construct right-linear grammar $G = (V, \Sigma, S, P)$ with:

- $V = Q$,
- $S = q_0$,
- P:

$$\begin{array}{lll} \delta(q_i, a_j) = q_k & \text{adds} & q_i \to a_j q_k \\ q_k \in F & \text{adds} & q_k \to \varepsilon. \end{array}$$

$\square$

Proof Cont.

Consider $w = a_i a_j \dots a_k a_l \in L$. Then, following transitions are possible in M:

$$\begin{aligned}
\delta(q_0, a_i) &= q_p \\
\delta(q_p, a_j) &= q_r \\
&\dots \\
\delta(q_s, a_k) &= q_t \\
\delta(q_t, a_l) &= q_f \in F,
\end{aligned}$$

which entails following derivation is possible in G:

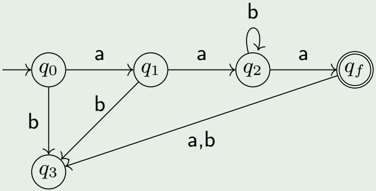
$$\begin{aligned}
q_0 &\Rightarrow a_i q_p \Rightarrow a_i a_j q_r \Rightarrow \cdots \Rightarrow a_i a_j \dots q_s \\
&\Rightarrow a_i a_j \dots a_k q_t \Rightarrow a_i a_j \dots a_k a_l q_f \Rightarrow a_i a_j \dots a_k a_l q_l.
\end{aligned}$$

Conversely, if $w \in L(G)$, by a similar argument, M accepts w, which completes the proof. $\square$

Regular Grammars

Example

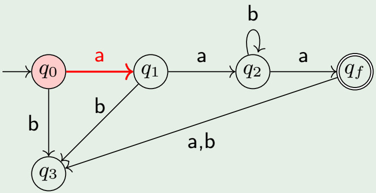
Construct a right-linear grammar for the language of aab^*a.



$$
\begin{aligned}
q_0 &\rightarrow aq_1 \\
q_0 &\rightarrow bq_3 \\
q_1 &\rightarrow aq_2 \\
q_1 &\rightarrow bq_3 \\
q_2 &\rightarrow aq_f \\
q_2 &\rightarrow bq_2 \\
q_f &\rightarrow aq_3 \\
q_f &\rightarrow bq_3 \\
q_f &\rightarrow \varepsilon
\end{aligned}
$$

Example

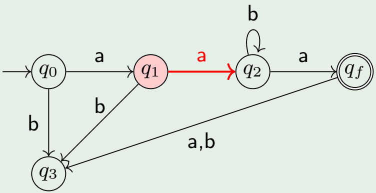
Construct a right-linear grammar for the language of aab^*a.
Examine the acceptance and generation of string aaa:



$q_0 \rightarrow aq_1$
$q_0 \rightarrow bq_3$
$q_1 \rightarrow aq_2$
$q_1 \rightarrow bq_3$
$q_2 \rightarrow aq_f$
$q_2 \rightarrow bq_2$
$q_f \rightarrow aq_3$
$q_f \rightarrow bq_3$
$q_f \rightarrow \varepsilon$

Example

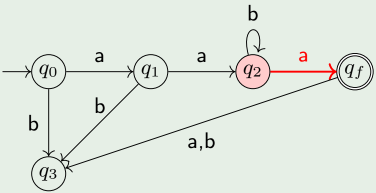
Construct a right-linear grammar for the language of aab^*a.
Examine the acceptance and generation of string aaa:



$$
\begin{aligned}
q_0 &\to aq_1 \\
q_0 &\to bq_3 \\
q_1 &\to aq_2 \\
q_1 &\to bq_3 \\
q_2 &\to aq_f \\
q_2 &\to bq_2 \\
q_f &\to aq_3 \\
q_f &\to bq_3 \\
q_f &\to \varepsilon
\end{aligned}
$$

Example

Construct a right-linear grammar for the language of aab^*a.
Examine the acceptance and generation of string aaa:

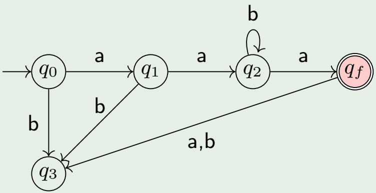


$$q_0 \to aq_1$$
$$q_0 \to bq_3$$
$$q_1 \to aq_2$$
$$q_1 \to bq_3$$
$$q_2 \to aq_f$$
$$q_2 \to bq_2$$
$$q_f \to aq_3$$
$$q_f \to bq_3$$
$$q_f \to \varepsilon$$

Regular Grammars

Example

Construct a right-linear grammar for the language of aab^*a.
Examine the acceptance and generation of string aaa:



$$q_0 \rightarrow aq_1$$
$$q_0 \rightarrow bq_3$$
$$q_1 \rightarrow aq_2$$
$$q_1 \rightarrow bq_3$$
$$q_2 \rightarrow aq_f$$
$$q_2 \rightarrow bq_2$$
$$q_f \rightarrow aq_3$$
$$q_f \rightarrow bq_3$$
$$q_f \rightarrow \varepsilon$$

Regular Grammars

Theorem

A language L is regular if and only if there exists a left-linear grammar G such that $L = L(G)$.

Proof idea:

- We can covert a right-linear grammar $\widehat{G}$ and a left-linear grammar G to each other such that $L(G) = L(\widehat{G})^{\mathcal{R}}$:

$$\begin{array}{lcl} A \rightarrow Bv & \leftrightarrow & A \rightarrow v^{\mathcal{R}}B \\ A \rightarrow v & \leftrightarrow & A \rightarrow v^{\mathcal{R}} \end{array}$$

- We can prove that reverse of a regular language is a regular language.
- Since $\widehat{G}$ is right-linear, $L(\widehat{G})$ is regular. Thus, $L(\widehat{G})^{\mathcal{R}}$ is regular, which means that $L(G)$ is also regular.
- For any regular language L, we can construct a right-linear grammar $\widehat{G}$ such that $L^{\mathcal{R}} = L(\widehat{G})$. Then, we can construct a left-linear grammar G such that $L(G) = L(\widehat{G})^{\mathcal{R}}$, which generates L.