Chapter 6

Properties of Regular Languages

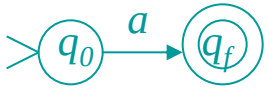
Regular Sets and Languages

- Claim(1). The family of languages accepted by FSAs consists of precisely the *regular sets* over a given alphabet.

- Every *regular set* is accepted by some *NFA-λ*,

  a) ϕ: 

  b) λ: 

  c) $\forall a \in \Sigma$: 

- Defn. 2.3.1 (on regular sets): Let Σ be an alphabet. The regular sets over Σ are defined recursively as follows:

  i) Basis: ϕ, $\{ \lambda \}$, and $\{ a \}$, $\forall a \in \Sigma$, are regular sets over Σ

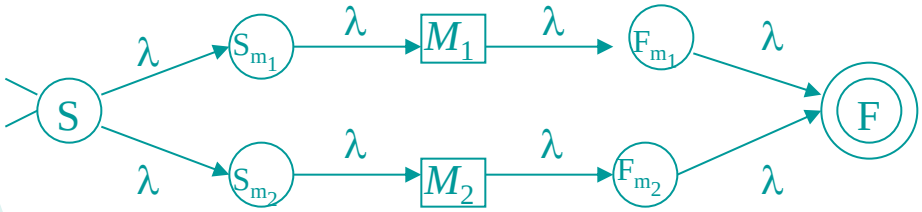
  ii) Recursion: Let X and Y be regular sets over Σ. The sets $X \cup Y$, XY, and X^* are regular sets over Σ.

  iii) Closure: Every regular set is obtained from (i) by a finite number of application of (ii)

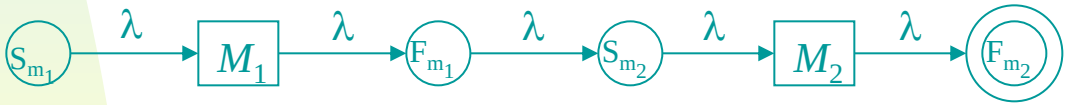
Regular Sets and Languages

- Let M_1 and M_2 be two FSAs, and let S_{m_1}, F_{m_1}, S_{m_2} , and F_{m_1}, be the new *start* and *accepting* states of M_1 and M_2, respectively:

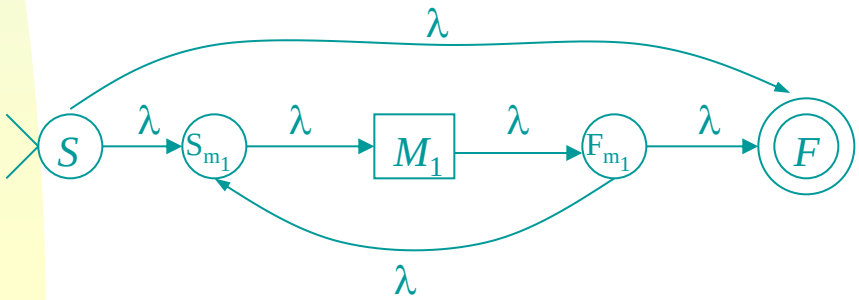
a) $L(M_1) \cup L(M_2)$: a string can be processed by M_1 and M_2 in parallel



b) $L(M_1) \bullet L(M_2)$: a string is processed by the composite machine sequentially



c) $L(M_1)^*$



3.3 Regular Grammars

- A grammar is defined as a quadruple (V, Σ, P, S), where
  - V is a finite set of *nonterminals*, or *variables*
  - Σ is a finite set of *terminals*
  - $S \in V$, is a *start symbol*, and
  - P is a finite set of *rewrite/production rules* of the form $\alpha \rightarrow \beta$, where $\alpha \in (V \cup \Sigma)^*$, $\beta \in (V \cup \Sigma)^*$

- Defn 3.3.1. A regular grammar is a grammar in which each rule has one of the following forms, where $A, B \in V$ and $a \in \Sigma$

  (i) $A \rightarrow a$

  (ii) $A \rightarrow aB$

  (iii) $A \rightarrow \lambda$

  - A *language* is *regular* if it can be generated by a *regular grammar*
  - Regular grammars generate precisely the languages defined by regular expressions
  - Regular expressions are used to abbreviate the descriptions of regular sets

3.3 Regular Grammars

- Example. Given G = (V, Σ, P, S), where

$$P = \{\ S \to xX$$
$$X \to yY$$
$$Y \to xX$$
$$Y \to \lambda\ \}$$
$$\Leftrightarrow\ (xy)^+$$

- Example. Let G = (V, Σ, P, S), where

$$P = \{\ S \to aA$$
$$A \to aA \mid bB$$
$$B \to aB \mid bC$$
$$C \to aC \mid \lambda\ \}$$
$$\Leftrightarrow\ a^+ba^*ba^*$$

- Other examples: Examples 3.29, 3.2.10, 3.2.11, 3.2.12

6.2 Expression Graphs

- Claim(2). Every *language* accepted by a FSA is *regular* by constructing a RE
- Defn. 6.2.1 An expression graph is a labeled digraph in which arcs are labeled by *regular expressions*.
  - Paths in expression graphs generate *regular expressions*
- Algorithm 6.2.2 Construction of a *regular expression* from a FSA
  - Produce an arbitrary expression graph by repeatedly **removing nodes** from the state diagram.
  - Produce the language of the FSA by the **union** of the *sets of strings* whose processing successfully terminates in one of the accepting states
  - Case 1.2.1

(i)

(ii)

6.2 Expression Graphs

Algorithm 6.2.2
Construction of a Regular Expression from a Finite Automaton

input: state diagram G of a finite automaton with one accepting state

Let q_0 be the start state and q_t the accepting state of G.

1. **repeat**
   - 1.1. choose a node q_i that is neither q_0 nor q_t
   - 1.2. delete the node q_i from G according to the following procedure:
     - 1.2.1 **for** every j, k not equal to i (this includes $j = k$) **do**
       - i) **if** $w_{j,i} \neq \emptyset$, $w_{i,k} \neq \emptyset$ and $w_{i,i} = \emptyset$, **then** add an arc from node j to node k labeled $w_{j,i}w_{i,k}$
       - ii) **if** $w_{j,i} \neq \emptyset$, $w_{i,k} \neq \emptyset$ and $w_{i,i} \neq \emptyset$, **then** add an arc from node q_j to node q_k labeled $w_{j,i}(w_{i,i})^*w_{i,k}$
       - iii) **if** nodes q_j and q_k have arcs labeled $w_1, w_2, \ldots, w_s$ connecting them, **then** replace the arcs by a single arc labeled $w_1 \cup w_2 \cup \cdots \cup w_s$
     - 1.2.2 remove the node q_i and all arcs incident to it in G

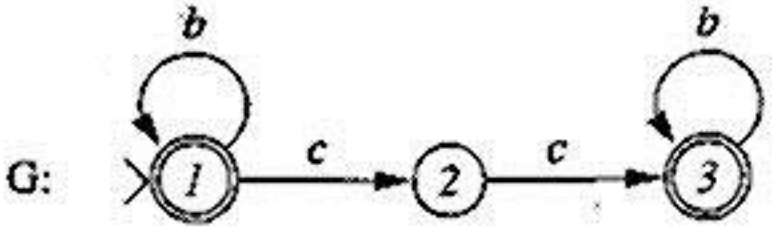
   until the only nodes in G are q_0 and q_t
2. determine the expression accepted by G

Note. If there are multiple final states in the given FSA *M*, then for each accepting state *F*, we produce an expression for the strings accepted by *F*. The language accepted by *M* is the *union* of the regular expressions.

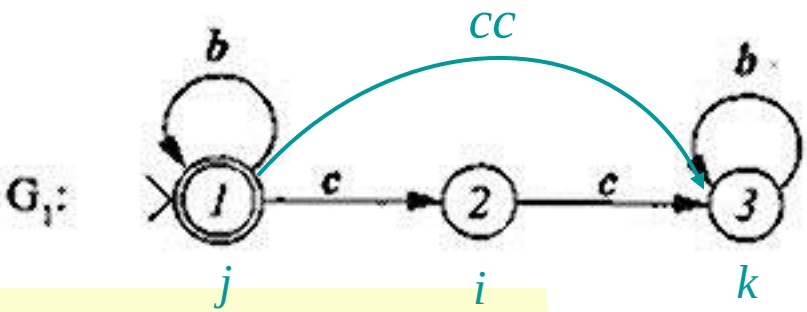
6.2 Expression Graphs

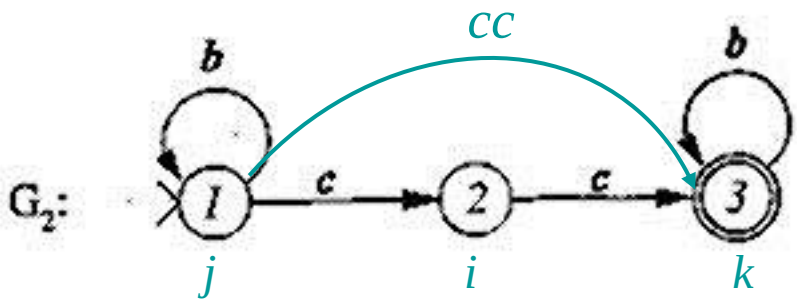
Example

The reduction technique of Algorithm 6.2.2 is used to generate a regular expression for the language of the NFA with state diagram G.



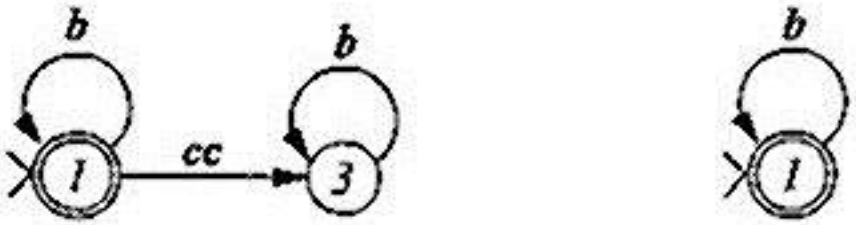
Two expression graphs, each with a single accepting node, are constructed from G.





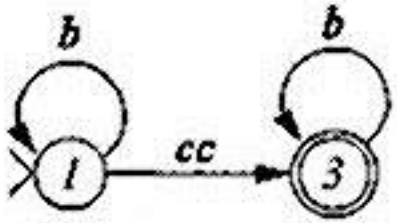
6.2 Expression Graphs

Reducing G_1 consists of deleting nodes 2 and 3 yielding



$j = k$ i

The expression accepted by G_1 is b^*. The removal of node 2 from G_2 produces



There is no such *i* that is neither the *start state* nor a *final state*

with associated expression b^*ccb^* The expression accepted by G, built from the expressions accepted by G_1 and G_2, is $b^* \cup b^*ccb^*$ □

6.3 Regular Grammars and FA

- Given a regular grammar G, there exists a NFA M such that $L(G) = L(M)$

- Theorem. 6.3.1 Let $G = (V, \Sigma, P, S)$ be a *regular* grammar. Define the NFA $M = (Q, \Sigma, \delta, S, F)$ as follows:

a) $Q = \begin{cases} V \cup \{ Z \}, \text{ where } Z \notin V, \text{ if } P \text{ contains a rule } A \to a \\ V \qquad\qquad\qquad\qquad\qquad \text{O.W.} \end{cases}$

b) $\delta(A, a) = B$ whenever $A \to aB \in P$
$\delta(A, a) = Z$ whenever $A \to a \in P$

c) $F = \begin{cases} \{ A \mid A \to \lambda \in P \} \cup \{ Z \}, & \text{if } Z \in Q \\ \{ A \mid A \to \lambda \in P \} & \text{O.W.} \end{cases}$

Then $L(M) = L(G)$

6.3 Regular Grammars and FA

- Example 6.3.1 Given $G = (V, \sum, P, S)$, where $V = \{ S, B \}$, $\sum = \{ a, b \}$, and $P = \{ S \to aS \mid bB \mid a, B \to bB \mid \lambda \}$, the corresponding NFA $M = (Q, \sum, \delta, S, F)$, where

  $Q = \{ S, B, Z \}$,

  $\sum = \{ a, b \}$,

  $\delta(S, a) = S$, $\delta(S, b) = B$, $\delta(S, a) = Z$, $\delta(B, b) = B$,

  $F = \{ B, Z \}$

- Theorem 6.3.2 Given an NFA M, there exists a regular grammar G .э. $L(M) = L(G)$.

  (i) $V = Q$

  (ii) The transition $\delta(A, a) = B$ yields the rule $A \to aB$ in G

  (iii) For each accepting state C, create the rule $C \to \lambda$ in G

- Example 6.3.4

  P: $S \to bB$, $S \to aA$; $B \to bS$, $B \to aC$, $B \to \lambda$;

  $A \to aS$, $A \to bC$; $C \to bA$, $C \to aB$;

6.4 Regular Languages

- A language over an alphabet Σ is regular if it can be
  - i) specified as a *regular set/expression* over Σ
  - ii) accepted by *DFA / NFA / NFA-λ*
  - iii) generated by a *regular grammar*

- Regular languages are closed under $\cup$, $\bullet$, *, $^-$, and $\cap$ since applying any of these operations to regular languages produces a regular language.

- Theorems 6.4.1, 6.4.2, 6.4.3
  - Let L_1 and L_2 be two regular languages. Then

    $$L_1 \cup L_2, \quad L_1L_2, \quad L_1^*, \quad \overline{L}, \text{ and } \quad L_1 \cap L_2$$

    are regular languages.

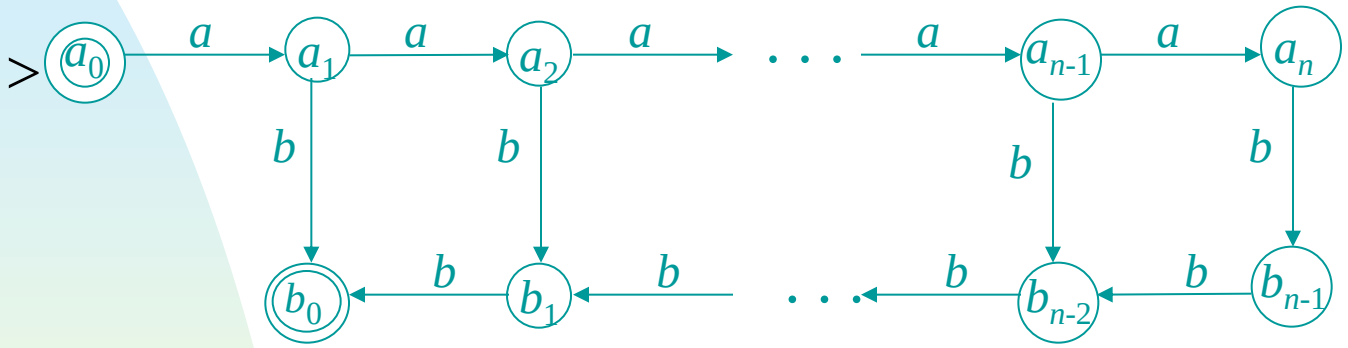
- Example 6.4.1

  $(a \cup b)^*aa(a \cup b)^* \cap \overline{(a \cup b)^*bb(a \cup b)^*}$ is regular

6.5 A Nonregular Language

- The language $\{ a^i b^i \mid 0 \le i \le n \}$ is regular, but the language $\{ a^i b^i \mid i \ge 0 \}$ is not.

  Proof.



  However, this technique cannot be extended to accept the language $\{ a^i b^i \mid i \ge 0 \}$ since an *infinite* number of states are needed.

- Theorem 6.5.1 (P. 204). The language $\{ a^i b^i \mid i \ge 0 \}$ is not *regular* (see a sketched proof in Example 6.6.3).

6.5 A Nonregular Language

- Corollary 6.5.2 (Proof of Theorem 6.5.1) Let L be a language over $\sum$. If

  $U_i \in \sum^*$ and $V_i \in \sum^*$, $i \geq 0$, .ɔ. $U_iV_i \in L$ and $U_iV_j \notin L$, $i \neq j$,

  then L is not *regular*, i.e., a language that generates strings with *equal length* of two distinct substrings is not *regular*.

- Example 6.5.1 The set of *palindromes* over { a, b } is not regular.

- Example 6.5.2 Regular grammars are not a sufficiently powerful tool to define programming languages containing *nesting* of *parentheses*.

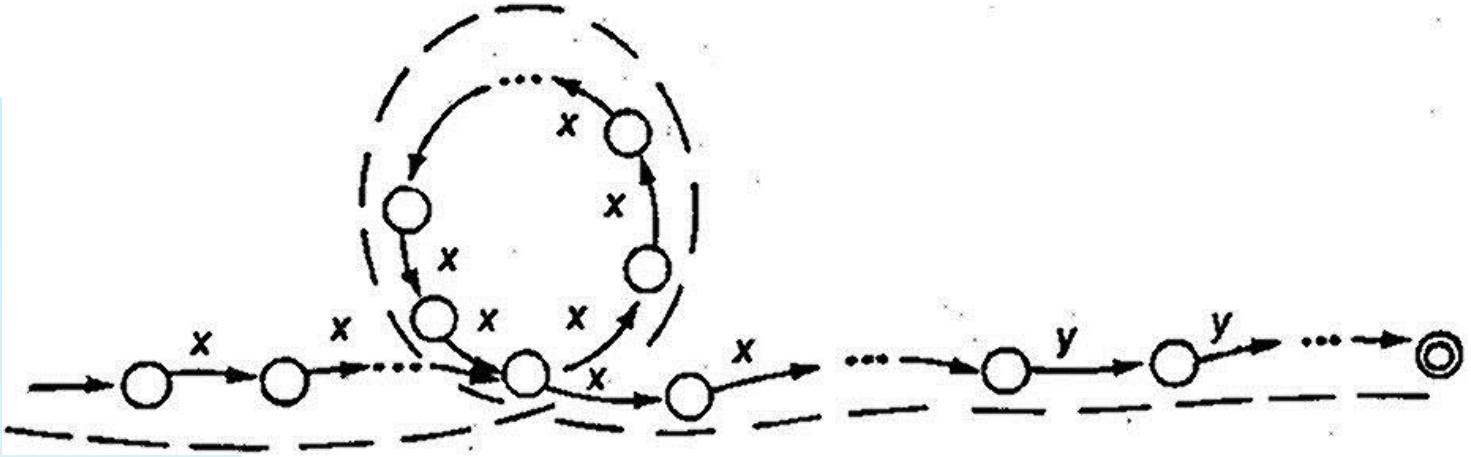
- Example 6.5.3 The language $L = \{ a^ib^j \mid i, j \geq 0 \text{ and } i \neq j \}$ is not regular; otherwise, $\overline{L}$, i.e., $\{ a^ib^i \mid i \geq 0 \}$, is regular.

6.6 The Pumping Lemma for Regular Languages

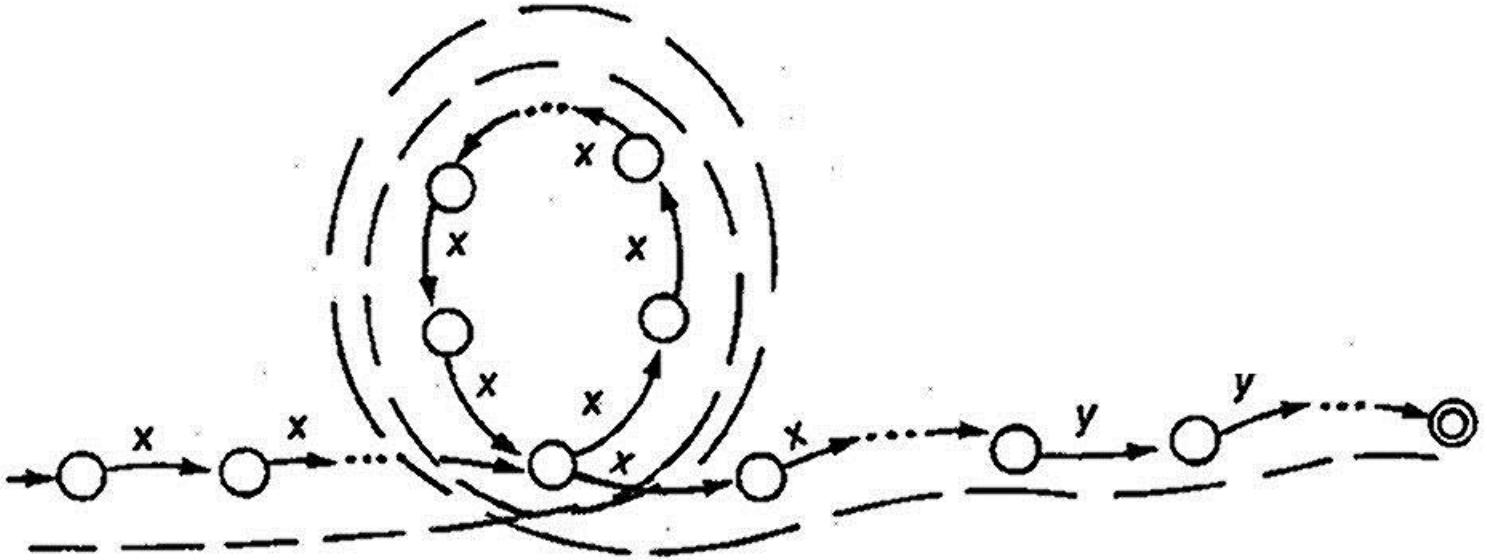
- The pumping lemma can be used to show a language is *nonregular*.
- Pumping a string refers to constructing new strings by *repeating/pumping* substrings in the original string.
- Processing a substring of a string in a DFA *M* correspond to *generating* a path in the state diagram of *M*.
- The proofs of nonregular languages using pumping lemma adopts a simple counting argument known as the pigeonhole principle.
- Lemma 6.6.1 Let *G* be the state diagram of a DFA with *k* states. Any path of length *k* in *G* contains a *cycle*.

  Proof: A *path P* of length *k* contains *k*+1 nodes (states). Since there are only *k* nodes in *G*, one of the nodes, *q*, must occurs in *P* at least twice. The subpath from the first *q* to the second *q* yields a cycle.

The portion of a transition that accepts both $x^k y^k$ and $x^{k+n} y^k$



a. The path traversed when accepting $x^k y^k$



b. The path traversed when accepting $x^{k+n} y^k$, where n is the length of the cycle

6.6 The Pumping Lemma for Regular Languages

- Corollary 6.6.2. Let G be the state diagram of a DFA with k states, and let p be a path of length k or more. The path p can be decomposed into sub paths (some of them can be empty) $\boldsymbol{q}$, $\boldsymbol{r}$, & $\boldsymbol{s}$, where $p = \boldsymbol{qrs}$, $|\boldsymbol{qr}| \leq k$ & $\boldsymbol{r}$ is a *cycle*.

- Theorem 6.6.3 (Pumping Lemma for Regular Languages) Let L be a regular language that is accepted by a DFA M with k states. Let $Z \in L$ with $|Z| \geq k$, and let $Z = uvw$ such that $|uv| \leq k$, $|v| > 0$, and $uv^iw \in L$, $\forall_{i \geq 0}$

- Example 6.6.3. Show that $L = \{a^ib^i \mid i \geq 0\}$ is nonregular.

  Assume that L is regular accepted by a DFA M with k states. Let $Z \in L$ with $|Z| \geq k > 0$ & $Z = a^kb^k$ such that

  $$u = a^i,\ v = a^j, \text{ and } w = a^{k-i-j}b^k$$

  Pumping v twice yields the string

  $$uv^2w = a^i\, a^j\, a^j\, a^{k-i-j}\, b^k = a^k\, a^j\, b^k \notin L, \text{ since } k + j > k.$$