CS 4410

Automata, Computability, and Formal Language

Dr. Xuejun Liang

Chapter 4
Properties of Regular Languages

1. Closure Properties of Regular Languages
   - Closure under Simple Set Operations
   - Closure under Other Operations
2. Elementary Questions about Regular Languages
3. Identifying Nonregular Languages
   - Using the Pigeonhole Principle
   - A pumping Lemma

Learning Objectives

At the conclusion of the chapter, the student will be able to:

- State the closure properties applicable to regular languages
- Prove that regular languages are closed under union, concatenation, star-closure, complementation, and intersection
- Prove that regular languages are closed under reversal
- Describe a membership algorithm for regular languages
- Describe an algorithm to determine if a regular language is empty, finite, or infinite
- Describe an algorithm to determine if two regular languages are equal
- Apply the pumping lemma to show that a language is not regular

Closure under Simple Set Operations

Theorem 4.1: If L, L_1 and L_2 are regular languages, then so are $L_1 \cup L_2$, $L_1 \cap L_2$, $L_1 L_2$, $\overline{L}$, and L^*. We say that the family of regular language is closed under union, intersection, concatenation, complementation, and star-closure.

Let r, r_1, r_2 be regular expressions such that

$$L = L(r), L_1 = L(r_1), and L_2 = L(r_2)$$

We have $L^* = L(r^*), L_1 \cup L_2 = L(r_1 + r_2), L_1 L_2 = L(r_1 r_2)$

Let M = (Q, Σ, δ, q_0, F) be a DFA such that $L = L(M)$

We have $\overline{L} = L(\overline{M})$, where $\overline{M} = (Q, \Sigma, \delta, q_0, Q-F)$

Finally, we have $L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$

Closure under Simple Set Operations

Example 4.1: Show that if L_1 and L_2 are regular, so is L_1-L_2.

We have $L_1 - L_2 = L_1 \cap \overline{L_2}$

Theorem 4.2: The family of regular languages is closed under reversal.

Let M = $(Q, \Sigma, \delta, q_0, \{q_f\})$ be an NFA such that $L = L(M)$

We have $L^R = L(M^R)$, where $M^R = (Q, \Sigma, \delta^R, q_f, \{q_0\})$

Closure under Other Operations

Definition 4.1: Suppose Σ and Γ are alphabets. Then a function $h: \Sigma^* \rightarrow \Gamma^*$ is called a homomorphism, if

$$h(a_1a_2\ldots a_n) = h(a_1)h(a_2)\ldots h(a_n) \qquad \text{(or } h(uv)=h(u)h(v)\text{)}$$

If L is a language on Σ, then its image is defined as

$$h(L)=\{h(w) : w \in L\}$$

Example 4.2: $\Sigma=\{a, b\}$ and $\Gamma=\{a, b, c\}$. A homomorphism h is defined as h(a)=ab and h(b)=bbc. L={aa, aba}. h(L)=?

Closure under Other Operations

Example 4.3: Σ={a, b} and Γ={b, c, d}. A homomorphism h is defined as h(a) = dbcc and h(b) = bdc.

Let r = (a+b*)(aa)* and

h(r) = (h(a)+h(b)*)(h(a)h(a))*
= (dbcc+(bdc)*)(dbccdbcc)*

Then, we have L(h(r)) = h(L(r))

Theorem 4.3: Let h be a homomorphism, if L is a regular language, then its image h(L) is also regular.

Let r is regular expression such that $L = L(r)$

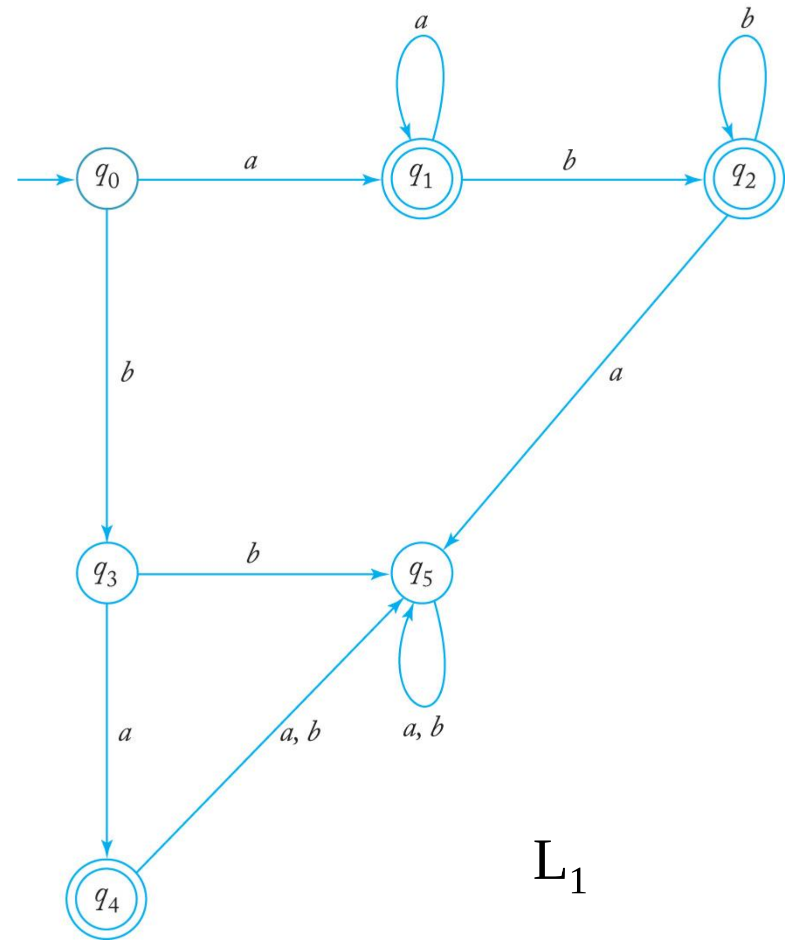
$$h(L) = h\big(L(r)\big) = L(h(r))$$

Closure under Other Operations

Definition 4.2:
Let L_1 and L_2 be languages on the same alphabet. Then the **right quotient** of L_1 with L_2 is defined as $L_1/L_2 =$

$$\{x : xy \in L_1 \text{ for some } y \in L_2\}$$

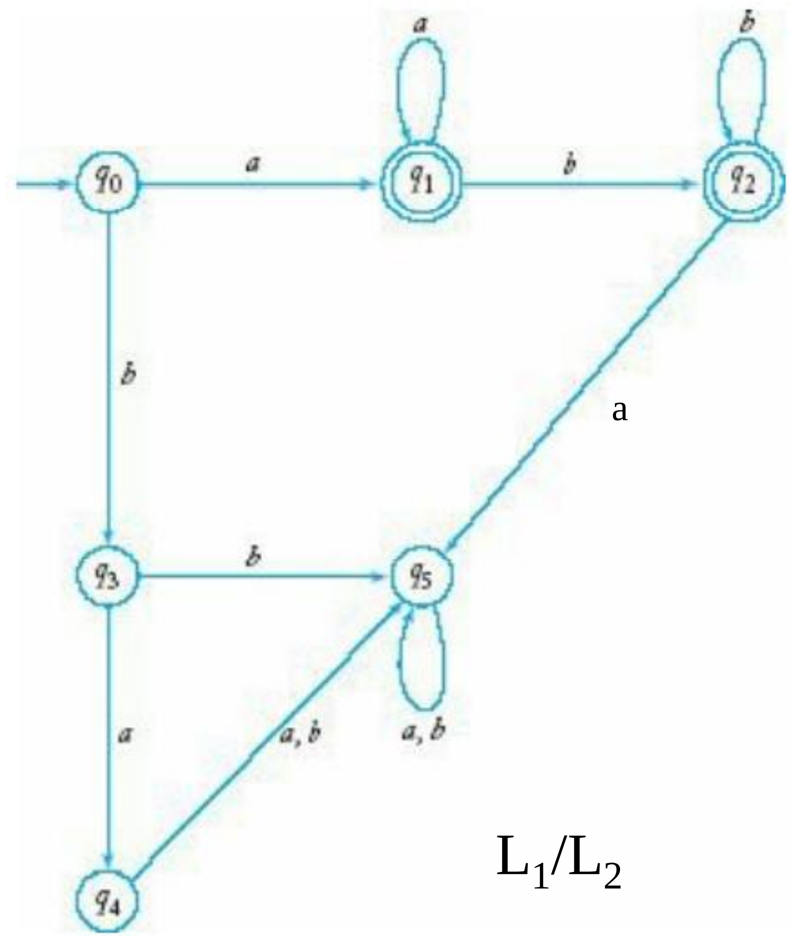
Example 4.4:
Let $L_1=\{a^n b^m : n\geq 1, m\geq 0\}\cup\{ba\}$ and $L_2=\{b^m : m\geq 1\}$.
Then $L_1/L_2 = \{a^n b^m : n\geq 1, m\geq 0\}$



Closure under Other Operations

Definition 4.2:
Let L_1 and L_2 be languages on the same alphabet. Then the right quotient of L_1 with L_2 is defined as L_1/L_2 =
$\{x : xy \in L_1 \text{ for some } y \in L_2\}$

Example 4.4:
Let $L_1=\{a^n b^m : n \geq 1, m \geq 0\} \cup \{ba\}$
and $L_2=\{b^m : m \geq 1\}$.
Then L1/L2= $=\{a^n b^m : n \geq 1, m \geq 0\}$

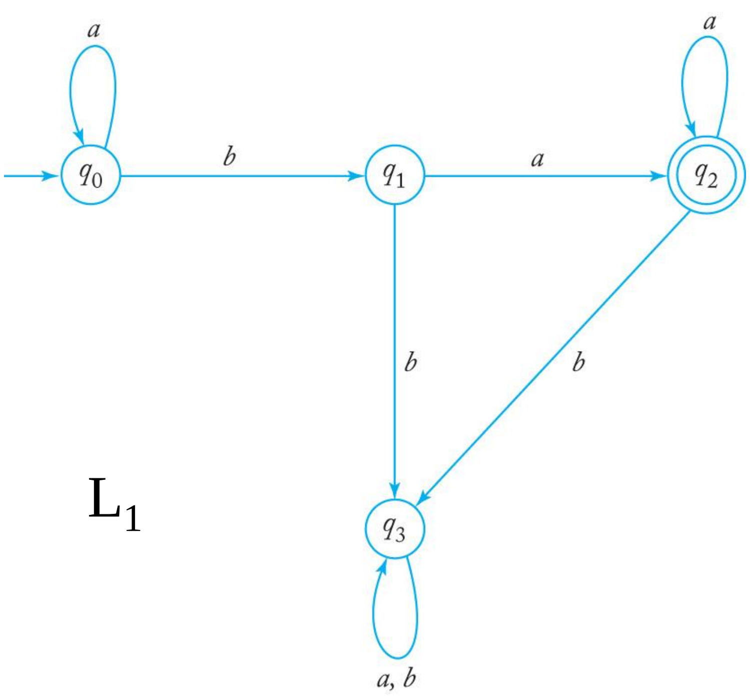


L_1/L_2

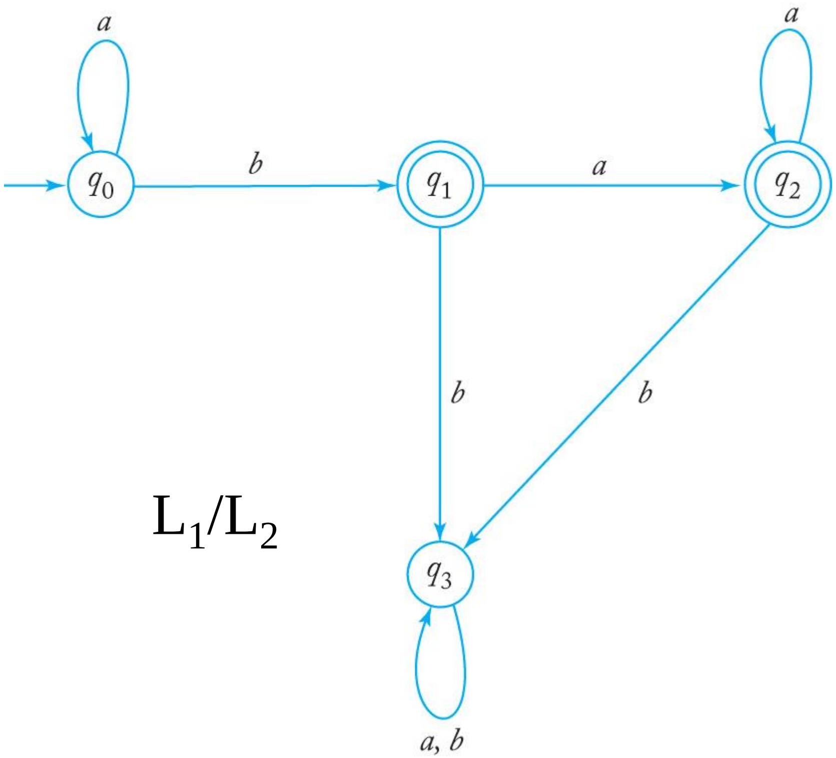
Closure under Other Operations

Theorem 4.4: If L_1 and L_2 are regular, then L_1/L_2 is also regular.

Example 4.5: Let $L_1=L(a^*baa^*)$ and $L_2=L(ab^*)$. Find L_1/L_2.



L_1



L_1/L_2

Elementary Questions

Recall: What is a regular language?

Finite automaton, Regular expression, Regular grammar

Theorem 4.5: Given any regular language L on Σ and any w ∈ Σ*, there exists an algorithm for determining whether or not w is in L.

Theorem 4.6: There exists an algorithm for determining whether a regular language is empty, finite, or infinite.

Theorem 4.7: Given two regular languages L_1 and L_2, there exists an algorithm for determining whether or not $L_1 = L_2$.

$$L_3 = (L_1 \cap \overline{L_2}) \cup ((\overline{L_1} \cap L_2))$$