PROPERTIES of REGULAR LANGUAGES

Chap. 4

Learning Objectives

- State the closure properties applicable to regular languages.
- Prove that regular languages are closed under the operations of 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 Properties

- Theorem 4.1: If L_1 and L_2 are regular languages,

  so are the languages that result from the following operations:
  - $L_1 \cup L_2$
  - $L_1 \cap L_2$
  - $L_1 \cdot L_2$
  - $\overline{L_1} = (L_1)^C$
  - $L_1{}^*$
  - i.e., the family of regular languages is *closed* under *union, intersection, concatenation, complementation,* and *star-closure.*

Proof of the Closure Properties

- Since L_1 and L_2 are regular languages, there exist regular expressions r_1 and r_2 to describe L_1 and L_2, respectively:

  i.e. $L(r_1) = L_1$ and $L(r_2) = L_2$.
- The union of L_1 and L_2 can be denoted by the regular expression $r_1 + r_2$: i.e. $L(r_1 + r_2) = L_1 \cup L_2$
- The concatenation of L_1 and L_2 can be denoted by the regular expression r_1r_2 : i.e. $L(r_1r_2) = L_1 \cdot L_2$
- The star-closure of L_1 can be denoted by the regular expression $r_1{}^*$: $L(r_1{}^*) = L_1{}^*$

---- See the construction of NFA to accept $L(r)$ in Chap. 3

- Therefore, the union, concatenation, and star-closure of arbitrary regular languages are also regular.

Proof of the Closure Properties (cont.)

- Claim: *Closure under complementation of regular* language.

Proof) For arbitrary *regular language* L_1, assume there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L_1: $L(M) = L_1$

$\Rightarrow$ Then, $\exists M'$ s.t. $L(M') = \overline{L_1}$?

Construct a DFA M' that accepts the complement of L_1 ($\overline{L_1}$, L_1^C) as follows:

- M' has the same states, alphabet, transition function, and start state as M.
- The *final states* in M become *non-final states* in M', while the *non-final states* in M become *final states* in M.

  i.e. $M' = (Q, \Sigma, \delta, q_0, Q - F)$

- Since *M'* accepts precisely the strings that M rejects and *M'* rejects precisely the strings that M accepts, *M'* accepts the complement of L_1.
- Therefore, the complement of regular language L_1 ($\overline{L_1}$, L^C) is regular.

Q.E.D.

Proof of the Closure Properties (cont.)

Claim: The *intersection* of two regular languages L_1 & L_2 is also regular.

Proof) Two basic approaches exist:

1. Given a DFA M_1 and DFA M_2 that accepts L_1 and L_2, respectively, construct a new DFA *M'* with states and transition function resulting from a combination of the states and transition functions from M_1 and M_2 :

   i.e. $L(M_1) = L_1$ and $L(M_2) = L_2 \Rightarrow L(M') = L_1 \cap L_2$?

2. Use DeMorgan's Law to show that the intersection of L_1 & L_2 can be obtained by applying union and complementation:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

Since the closure of regular languages under the union and the closure of complementation, the intersection of two regular languages L_1 and L_2 must also produce a regular language.

Q.E.D.

Proof of the Closure Properties (cont.)

Claim: The *intersection* of two regular languages L_1 & L_2 is also regular.

Proof) 1. Given DFAs $M_1=(Q, \Sigma, \delta_1, q_0, F_1)$ and $M_2=(P, \Sigma, \delta_2, p_0, F_2)$

s.t. $L(M_1)=L_1$ and $L(M_2)= L_2$,

Construct a new DFA M' with states and transition functions

from M_1 and M_2, s.t. $L(M') = L_1 \cap L_2$:

$M' =(Q', \Sigma, \delta', q_0', F')$ where

$Q'= Q\times P = \{ (q_i, p_j) \mid \forall q_i \in Q, \forall p_j \in P \}$

$q_0' = (q_0, p_0)$,

$F'= F_1\times F_2 = \{ (q_f, p_f) \mid \forall q_f \in F_1, \forall p_f \in F_2\}$

$\delta' : (Q\times P) \times \Sigma \rightarrow (Q \times P)$,

s.t. $\delta'((q_i, p_j), a) = (q_k, p_l), \forall q_i, q_k \in Q, \forall p_j, p_l \in P$

whenever $\delta_1(q_i, a) = q_k$ and $\delta_2(p_j, a) = p_l$.

Then, show that $w\in L_1 \cap L_2$ iff $w\in L(M')$.

Thus, $L_1 \cap L_2$ is regular.

Example: Closure Property

- Example 4.1:

The family of regular language is closed under *difference*.
i.e. If L_1 and L_2 are regular, then, $L_1 - L_2$ is also regular.

Proof) Define the set difference by its definition,

$$L_1 - L_2 = L_1 \cap \overline{L_2}.$$

Since L_2 is regular, so is $\overline{L_2}$ due to the closure of complement.

Since L_1 and $\overline{L_2}$ are regular, so is $L_1 \cap \overline{L_2}$ from the closure of intersection.

Thus, $L_1 - L_2$ is regular.

Closure under Reversal

- Theorem 4.2:
  The family of regular language is closed under *reversal*.

i.e. if L is a regular language, so is L^R (reversal).

- To prove closure under reversal, we can assume the existence of a NFA M with a *single final state* that accepts L:
  i.e. $L = L(M)$ where $M = (Q, \Sigma, \delta, q_0, q_F)$
- Given the transition graph of M, to construct a NFA M^R that accepts L^R:
  - The start state in M becomes the final state in M^R.
  - The final state in M becomes the start state in M^R.
  - The direction of all transition edges in M is reversed.
  - i.e. $M^R = (Q, \Sigma, \delta^R, q_F, q_0)$
    where $\exists(q_j, a) \rightarrow q_i \in \delta^R, \quad \forall(q_i, a) \rightarrow q_j \in \delta$

Closure under Homomorphism

- Definition 4.1: Suppose Σ and Γ are alphabets.

A function $h: \Sigma \rightarrow \Gamma^*$ is called a *homomorphism*;
i.e. a homomorphism is a substitution/mapping
in which a single letter is replaced with a string.
The domain of the function h is extended to strings;

- If $w=a_1a_2..a_n$, then $h(w) = h(a_1)h(a_2) \ldots h(a_n)$.
- If L is a language on Σ, then its homomorphic image is defined as $h(L) = \{h(w) | w \in L\}$.

- If we have a regular expression r for a language L *(i.e. $L(r) = L$)*, then a regular expression for $h(L)$ can be obtained by simply applying the homomorphism to each Σ symbol of r, i.e. $h(r)$.

Homomorphism (cont.)

- Theorem 4.3: Let h be a homomorphism.
  If L is a regular language, then its homomorphic image $h(L)$ is regular.
  The family of regular language is closed under any *homomorphisms*.

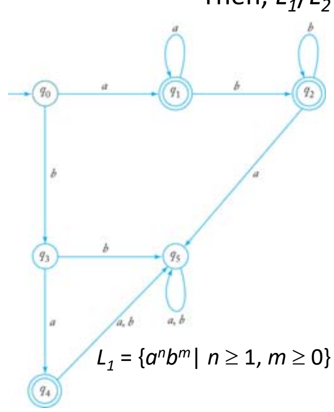
- Example 4.2: $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, c\}$. Define h by
  $h(a) = ab$, $h(b) = bbc$.
  Then, $h(aba) = abbbcab$.
- Example 4.3: $\Sigma = \{a, b\}$ and $\Gamma = \{b, c, d\}$. Define h by
  $h(a) = dbcc$, $h(b) = bdc$.
  If L is the regular language denoted by REX $r = (a+b^*)(aa)^*$,
  then, the REX denoting $h(L)$ is
  $r_1 = (dbcc + (bdc)^*)(dbccdbcc)^* = h(r)$

Right Quotient

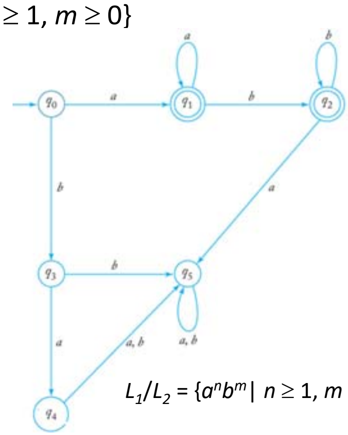
- Definition 4.2: Let L_1 and L_2 be languages on the same alphabet. Then, the *right quotient* of L_1 with L_2 is defined as

  $$L_1/L_2 = \{x \mid xy \in L_1 \text{ for some } y \in L_2\}.$$

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



$L_1 = \{a^n b^m \mid n \geq 1, m \geq 0\} \cup \{ba\}$



$L_1/L_2 = \{a^n b^m \mid n \geq 1, m \geq 0\}$

Right Quotient (cont.)

- Theorem 4.4:

  If L_1 and L_2 are regular languages, then L_1/L_2 is also regular. i.e. the family of regular languages is closed under *right quotient* with a regular language.

Proof) Let $L_1 = L(M)$ for a DFA $M = (Q, \Sigma, \delta, q_0, F)$.

1) Let's construct an DFA $M' = (Q, \Sigma, \delta, q_0, F')$ s.t. $L(M') = L_1/L_2$ as follows.

Repeat for each $q_i \in Q$,

- determine whether there exists $y \in L_2$ s.t. $\delta^*(q_i, y) = q_f \in F$.

  -- It can be done by looking at DFA $M_i = (Q, \Sigma, \delta, \boldsymbol{q_i}, F)$ whose initial state is q_i.
- Now, determine whether there exists $y \in L(M_i)$ that is also in L_2.
- For this, we can use the construction for the intersection of two regular languages in Th[m] 4.1, finding the transition graph for $L_2 \cap L(M_i)$.
- If there is any path between its initial state and any final state, then $L_2 \cap L(M_i) \neq \emptyset$. If so, add q_i to F'.

F' is determined so that M' is constructed.

Right Quotient (cont.)

- Theorem 4.4: If L_1 and L_2 are regular languages, then L_1/L_2 is also regular. i.e. the family of regular languages is closed under *right quotient* with a regular language.

Proof) 2) Prove $L(M') = L_1/L_2$.

$\leftarrow$) Show that for any $x \in L_1/L_2$, $x \in L(M')$, x is accepted by M'.

Let $x \in L_1/L_2$. Then there must be a $y \in L_2$ such that $xy \in L_1$.

This implies that $\delta^*(q_0, xy) \in F$, so that there must be some $q \in Q$ such that

$\delta^*(q_0, x) = q$ and $\delta^*(q, y) \in F$.

Thus, by construction, $q \in F'$ and M' accepts x because $\delta^*(q_0, x) \in F'$.

$\rightarrow$) Show that for any $x \in L(M')$, $x \in L_1/L_2$.

For any $x \in L(M')$, we have $\delta^*(q_0, x) \in F'$.

By construction, it implies that exists a $y \in L_2$ such that $\delta^*(q, y) \in F$.

Thus, xy is in L_1, and x is in L_1/L_2, i.e. $x \in L_1/L_2$.

Therefore, $L(M') = L_1/L_2$ and L_1/L_2 is regular.

Right Quotient (cont.)

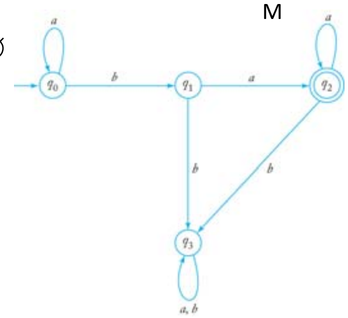
- Example 4.5: Find L_1/L_2 for $L_1 = L(a^*baa^*)$ and $L_2 = L(ab^*)$.

First, find a DFA M that accepts L_1. In M,

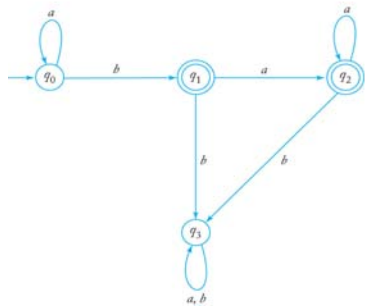
- For q_0, $L(M_0) \cap L_2 = L(a^*baa^*) \cap L(ab^*) = \varnothing$
- For q_1, $L(M_1) \cap L_2 = L(aa^*) \cap L(ab^*) = \{a\}$
- For q_2, $L(M_2) \cap L_2 = L(a^*) \cap L(ab^*) = \{a\}$
- For q_3, $L(M_3) \cap L_2 = \varnothing \cap L(ab^*) = \varnothing$

So, $F' = \{q_1, q_2\}$ in M' s.t. $L(M') = L_1/L_2$.

M



M'



In M',

$L(M') = a^*b + a^*baa^*$

$= a^*b(\lambda + aa^*) = a^*ba^*$ in REX.

Therefore, $L_1/L_2 = L(a^*ba^*)$

Elementary Questions about Regular Languages

- Given a regular language L and an arbitrary string w, is there an algorithm to determine whether w is in L or not ?
  : *membership, $w \in L$?*
- Given a regular language L, is there an algorithm to determine if L is empty, finite, or infinite?
  : *emptiness, (in)finiteness, $L = \varnothing$? $|L| \neq \infty$?*
- Given two regular languages L_1 and L_2, is there an algorithm to determine whether $L_1 = L_2$ or not?
  : *equality*

A Membership Algorithm for Regular Languages

- Theorem 4.5: There exists a membership algorithm for regular languages: $\text{Alg}_{\text{MEM}}(w \in L)$
- Proof Idea)

  To determine if an arbitrary string w is in a regular language *L*, we assume the existence of a standard unambiguous representation of *L*, i.e. L = L(*r*) where r is a REX.

  Given a standard representation of *L*, construct a DFA to accept *L*.

  Simulate the operation of the DFA while processing *w* as the input string.

  If the DFA *halts* in a *final state* after processing *w*, then $w \in L$.

  So, we can determine the membership of w in the regular language L.

Determining whether a Regular Language is Empty, Finite, or Infinite

- Theorem 4.6: There exists an algorithm to determine if a regular language is empty, finite, or infinite: $\text{Alg}_{\text{EMP-FINITE}}(L)$
- Given the transition graph of a DFA that accepts *L*,
  - If there is a simple *path* from the start state to any final state, *L* is *not empty* (since it contains, at least, the corresponding string)
  - If a path from the start state to a final state includes a vertex which is the base of some *cycle*, *L* is infinite; otherwise, L is finite.

Equality Algorithm for Two Regular Languages

- For finite languages, equality could be determined by performing a comparison of the individual strings.
- Theorem 4.7: There exists an algorithm to determine if two regular languages L_1 and L_2 are equal: *$Alg_{EQ}(L_1, L_2)$*

Proof)
- Define the language $L = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$
- By closure, L is regular, so we can construct a DFA M to accept it, and by Theorem 4.6, we can determine whether L is empty.
- $L_1 = L_2$ if and only if $L = \emptyset$.

Identifying Nonregular Languages

- Although regular languages can be infinite, their associated automata have finite memory; therefore,

  incapable of accepting many languages and some limits on the structure of a regular language is imposed.
- A language is regular only if, in processing any string, the information that has to be remembered at any stage is strictly limited.
- Two basic approaches to show that a language is not regular:
  - Use the *Pigeonhole Principle* to construct a *proof by contradiction.*

    -- If we put n objects into m boxes (pigeonholes), and if $n > m$, then at least one box must have more than one item in it.
  - Use a *Pumping Lemma* for regular languages.

Identifying Nonregular Languages

- Example 4.6: Is L = $\{a^n b^n \mid n \geq 0\}$ regular?

Proof by contradiction) Suppose L is regular. Then,

$\exists$ a DFA M = (Q, $\{a,b\}$, δ, q_0, F) s.t. $L(M) = L$.

Let's look at $\delta^*(q_0, a^i)$ *for i=1, 2, 3, ..*

Since there are an unlimited number of *i*'s, but only a finite number of states in *M*, the pigeonhole principle tells us that there must be some state, say *q*, s.t. $\delta^*(q_0, a^n)=q$ *and* $\delta^*(q_0, a^m) = q$ with $n \neq m$.

But, since *M* accepts $a^n b^n$, we must have $\delta^*(q, b^n) = q_f \in F$:

i.e. $\delta^*(q_0, a^n b^n) = \delta^*(\delta^*(q_0, a^n), b^n) = \delta^*(q, b^n) = q_f \in F$.

From this, we can conclude that

$$\delta^*(q_0, a^m b^n) = \delta^*(\delta^*(q_0, a^m), b^n) = \delta^*(q, b^n) = q_f \in F.$$

Contradiction to the assumption that M accepts, $a^m b^n$ only if $n = m$.

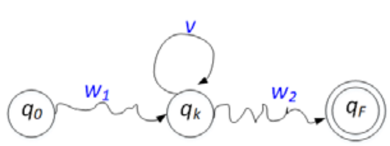
Therefore, *L* can not be regular.

Basis for the Pumping Lemma

- The transition graph for a regular language has certain properties:
  - If the graph has *no cycles*, the language is *finite*.
  - If the graph has a *nonempty cycle*, the language is *infinite*.
  - If the graph has such cycle, the cycle can either be skipped or repeated an arbitrary number of times.

    So, if the cycle has label *v* and if the string $w_1 v w_2$ is in the language, so are the strings $w_1 vv w_2, w_1 vvv w_2, \ldots, w_1 v^n w_2$ etc.
  - If such a cycle exists in a DFA with *m* states, the cycle must be entered by the time *m* symbols have been processed.

- As a basis for the pumping lemma, we observe that given a language *L*, if any string in *L* does not satisfy these properties, *L* is not regular.

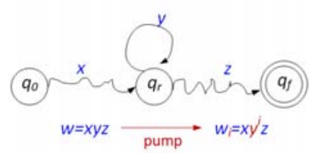


A Pumping Lemma for Regular Languages

- Theorem 4.8: Given an *infinite regular language L*, there exists some positive integer m such that any sufficiently long string $w \in L$ with $|w| \geq m$ can be decomposed as
  - $w = xyz$ with
  - $|y| \geq 1$ and $|xy| \leq m$

    where m is an arbitrary integer, $0 < m \leq |w|$, s.t.
  - an arbitrary number of repetitions of y yields a string in L:

    $w_i = xy^iz \in L,\ i \geq 0.$



- The middle section, y, is said to be "pumped" to generate additional strings in L.
- The Pumping Lemma can be used to show, by *contradiction*, that a certain language is *not regular*.

Pumping Lemma: Proof

If L is regular, there exists a DFA $M=(Q, \Sigma, \delta, q_0, F)$ that accepts it, i.e. $L(M) = L$. Let M have states labeled $q_0, q_1, q_2, \ldots, q_n$.

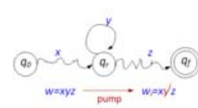
Now, take a string $w \in L$ such that $|w| \geq m = n + 1$.

Since L is assumed to be infinite, this can always be done. Consider the set of states the automaton goes through as it processes w, say

$$q_0,\ q_i,\ q_j,\ \ldots,\ q_f.$$

Since this sequence has exactly $|w| + 1$ entries, at least one state must be repeated, and such a repetition must start no later than the n^{th} move. Thus, the sequence must look like

$$q_0, q_i, q_j, \ldots, q_r, \ldots, q_r, \ldots, q_f,$$



indicating there must be substrings x, y, z of w s.t.

$\delta^*(q_0, x) = q_r,\ \ \delta^*(q_r, y) = q_r,\ \ \delta^*(q_r, z) = q_f$, with $|xy| \leq n+1 = m$, $|y| \geq 1$.

From this, it follows that $\delta^*(q_0, xz) = q_f$,

as well as $\delta^*(q_0, xy^1z) = q_f$, $\delta^*(q_0, xy^2z) = q_f$, $\delta^*(q_0, xy^3z) = q_f$, etc.

Therefore, any $w_i = xy^iz \in L$. Q.E.D.

Example: Proof by Pumping Lemma

- Example 4.7: Show that $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

Proof by contradiction) Assume that L is regular.

Then, Pumping Lemma must hold.

We don't know the value of m, but whatever it is, we can (always) choose $m=n$. Thus, in $w=xyz$ where $|xy| \leq m=n$,

the substring y must consist of all a's in w. Suppose $|y| = k \geq 1$.

i.e. $x = a^j$, $y = a^k$, $z= a^{m-j-k}b^m$, for any $m \geq j, k \geq 1$

$$\overbrace{\underbrace{a\ a \ldots\ldots a}_{x = a^j}\ \underbrace{a \ldots a}_{\substack{y = a^k \\ |y| \geq 1}}}^{|xy| \leq m}\ \underbrace{a \ldots a\ \ b\ b \ldots\ldots\ldots\ldots b}_{z = a^{m-j-k}\, b^m}$$

Example: Proof by Pumping Lemma

- Example 4.7 (cont.): Show that $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

$$\overbrace{\underbrace{a\ a \ldots\ldots a}_{x = a^j}\ \underbrace{a \ldots a}_{\substack{y = a^k \\ |y| \geq 1}}}^{|xy| \leq m}\ \underbrace{a \ldots a\ \ b\ b \ldots\ldots\ldots\ldots b}_{z = a^{m-j-k}\, b^m}$$

Then, the new string pumping y with $i = 0$ is $w_0 = a^{m-k}b^m$.
and w_0 is clearly not in L ($w_0 \notin L$).
Contradiction to the Pumping Lemma!
Thus, it contradicts the assumption 'L is regular'.
Therefore, $L = \{a^n b^n \mid n \geq 0\}$ is not regular. Q.E.D.

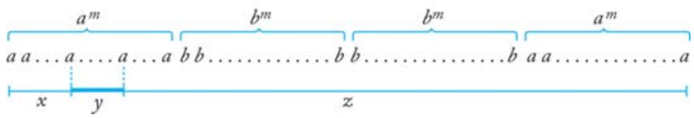
Example: Proof by Pumping Lemma

- Example 4.8: Show that $L = \{ww^R \mid w \in \Sigma^*\}$ is not regular.

Pf) Assume that L is regular. So, Pumping Lemma must hold.

Let's choose $w = a^m b^m b^m a^m \in L$ with $|w| \geq m$. Thus, in $w=xyz$ with $|xy| \leq m$, the substring y must consist of all a's s.t.

$$x=a^j,\ y=a^k,\ z=a^{m-j-k}b^m b^m a^m, \quad j, k > 0$$



Suppose $|y| = k \geq 1$.

Then, the new string pumping y with $i = 0$ is $w_0 = a^{m-k} b^m b^m a^m$

and w_0 is clearly not in L ($w_0 \notin L$). This contradicts the Pumping Lemma.

Thus, the assumption ' L is regular' must be false.

Therefore, $L = \{a^n b^n : n \geq 0\}$ is not regular.

Example: Proof by Pumping Lemma

- Example 4.9: $\Sigma=\{a, b\}$.

  Show $L = \{w \in \Sigma^* \mid n_a(w) < n_b(w)\}$ is not regular.

Proof) Suppose m is given. Choose $w = a^m b^{m+1}$ where $|w| \geq m$.

Since $|xy| \leq m$, y is with all a's, i.e. $y = a^k$, $m \geq k \geq 1$.

So, $x = a^j$, $y = a^k$, $z = a^{m-j-k} b^{m+1}$, for any $m \geq j, k \geq 1$.

Let's pump y with $i = 3$: $w_3 = a^j \mathbf{a^{3k}} a^{m-j-k} b^{m+1} = \mathbf{a^{m+2k}} b^{m+1}$.

Since $k \geq 1$, w_3 is clearly not in L ($w_3 \notin L$).

Contradiction to the Pumping Lemma!

Thus, it contradicts the assumption 'L is regular'.

Therefore, $L = \{w \in \Sigma^* \mid n_a(w) < n_b(w)\}$ is not regular.

Example: Proof by Pumping Lemma

- Example 4.10:
  Show that $L = \{(ab)^n a^k \mid n > k, k \geq 0\}$ is not regular.
- Example 4.11:
  Show that $L = \{a^n \mid n \text{ is a perfect square}\}$ is not regular.
- Example 4.12:
  Show that $L = \{a^n b^k c^{n+k} \mid n, k \geq 0\}$ is not regular.
- Example 4.13:
  Show that $L = \{a^n b^l \mid n \neq l\}$ is not regular.

Pf) See the textbook.