Taibah University
College of Computer Science & Engineering
Course Title: Theory of Computations
Code: CS 301

Chapter 4

Properties of Regular Languages

Dr. Sultan E. Almaghthawi

Based on slides by: Dr. Abdelfatah Farawila

Text Book: Peter Linz, Introduction to Formal Languages and automata 4th edition), 2006.

Closure Properties of Regular Languages

- Consider the following question:

Given two regular languages L1 and L2 , is their **union also regular?** In specific instances, the answer may be obvious, but here we want to address the problem in general.

- Is it true for all regular L1 and L2?

It turns out that the answer is yes, a fact we express by saying that the family of regular languages is closed under union.

➔ This leads us to the study of the closure properties of languages in general

Closure under Simple Set Operations

We begin by looking at the closure of regular languages under the common set operations, such as union and intersection.

Theorem 4.1

If L1 and L2 are regular languages, then so are L1 ∪ L2 , L1 ∩ L2 , L1L2 , $\overline{L_1}$ and L_1^*. We say that the family of regular languages is closed under union, intersection, concatenation, complementation, and star-closure.

Closure Properties

Example 4.1

Show that the family of regular languages is closed under difference. In other words, we want to show that if L1 and L2 are regular, then L1 − L2 is necessarily regular also. The needed set identity is immediately obvious from the definition of a set difference, namely

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

The fact that L2 is regular implies that $\overline{L_2}$ is also regular. Then, because of the closure of regular languages under intersection, we know that $L_1 \cap \overline{L_2}$ is regular, and the argument is complete.

A variety of other closure properties can be derived directly by elementary arguments

Closure Properties

Theorem 4.2

- The family of regular languages is **closed under reversal**.
- **Proof:** The proof of this theorem was suggested as an exercise in Section 2.3. Here are the details:

Suppose that L is a regular language. We then construct an nfa with a single final state for it. By Exercise 7, Section 2.3, this is always possible. In the transition graph for this nfa we make the initial vertex a final vertex, ***the final vertex the initial vertex***, ***and reverse the direction on all the edges***. It is a fairly straightforward matter to show that the **modified nfa accepts w^R** if and only if the original nfa accepts w.

➔ Therefore, the modified nfa accepts L^R , proving closure under reversal.

Closure Properties

Closer Under Other operations

In addition to the standard operations on languages, one can define other operations and investigate closure properties for them. There are many such results; we select only two typical ones. Others are explored in the exercises at the end of this section.

Closure Properties

Definition 4.1

Suppose Σ and Γ are alphabets. Then a function

$$h : \Sigma \to \Gamma^*$$

is called a **homomorphism**. In words, a **homomorphism** is a substitution in which a single letter is replaced with a string. The domain of the function *h* is extended to strings in an obvious fashion; if

$$w = a_1 a_2 \cdots a_n,$$

then

$$h(w) = h(a_1)\, h(a_2) \cdots h(a_n).$$

Homomorphism

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

L(h) = { h(w) : w ∈ Σ }

Example 4.2

Let Σ = {a, b, c} and Γ = {a, b, c,} define h by

h(a) = ab,
h(b) = bbc

Then h (aba) = abbbcab. The homomorphic image of L = {aa, aba} is the language **h (L) = {abab, abbbcab}**.

Homomorphism

Example 4.3

Take $\Sigma = \{a, b\}$ and $\Gamma = \{b, c, d\}$. Define h by

$$h(a) = dbcc,$$
$$h(b) = bdc.$$

If L is the regular language denoted by

$$r = (a + b^*)(aa)^*,$$

then

$$r_1 = (dbcc + (bdc)^*)(dbccdbcc)^*$$

denotes the regular language $h(L)$.

Homomorphism

Theorem 4.3

Let *h* be a homomorphism. **If *L* is a regular language, then its homomorphic image h(L) is also regular**. The family of regular languages is therefore closed under arbitrary homomorphisms.

Proof: Let L be a regular language denoted by some regular expression r. We find *h(r)* by substituting *h(a)* for each symbol a ∈ Σ of r. It can be shown directly by an appeal to the definition of a regular expression that the result is a regular expression. It is equally easy to see that the resulting expression denotes *h(L)*. All we need to do is to show that for every w ∈ L(r), the corresponding h (w) is in L (h (r))

and conversely that every υ in L(h (r)) there is a w in L, such that $\upsilon = h(w)$.

Leaving the details as an exercise, we claim that h (L) is regular

Example

Example 4.4

If

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

and

$$L_2 = \{b^m : m \geq 1\},$$

then

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

The strings in L2 consist of one or more b's.

➔ Therefore, we arrive at the answer by removing one or more b's from those strings in L1 that terminate with at least one b.

Right Quotient of Two Languages

Note that here L1 , L2 , and L1 /L2 are all regular.

This suggests that the right quotient of any two regular languages is also regular

Regular language & Regular Grammar

Theorem 4.6

There exists an algorithm for determining whether a regular language, given in standard representation, is empty, finite, or infinite.

Proof: The answer is apparent if we represent the language as a transition graph of a dfa.

If there is a simple path from the initial vertex to any final vertex, then the language is not empty. To determine whether or not a language is infinite, find all the vertices that are the base of some cycle. **If any of these are on a path from an initial to a final vertex, the language is infinite. Otherwise, it is finite.**

Nonregular Languages

Regular languages can be **infinite**, as most of our examples have demonstrated. The fact that regular languages are associated with automata that have finite memory, however, imposes some limits on the structure of a regular language.

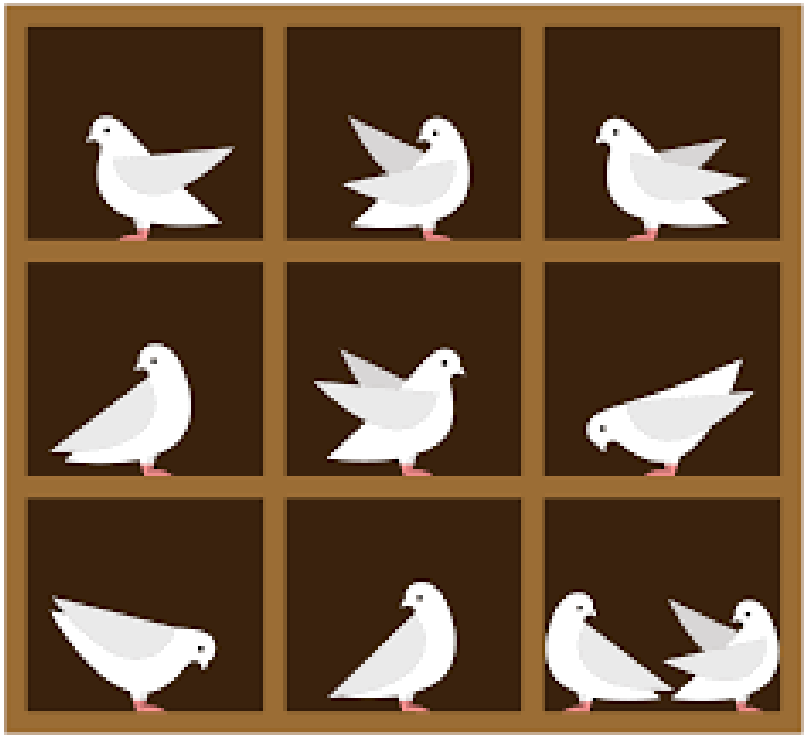
Intuition tells us that a language is regular **only if**, in processing any string, the information that has to be remembered at any stage is strictly limited. This is true, but has to be shown precisely to be used in any meaningful way. There are several ways in which this can be done.

Nonregular Languages

Using the Pigeonhole Principle

The term "***pigeonhole principle***" is used by mathematicians to refer to the following simple observation:

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. This is such an obvious fact that it is surprising how many deep results can be obtained from it.



Nonregular Languages

Example 4.6

Is the language L ={$a^n b^n$: n ≥ 0} regular? The answer is no, as we show using a proof by contradiction.

Suppose L is regular. Then some dfa M = (Q, {a, b},δ, q0 , F) exists for it. Now look at δ* (q0 ,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, such that

$$\delta^*(q_0, a^n) = q$$

And

$$\delta^*(q_0, a^m) = q,$$

with n ≠ m. But since M accepts $a^n b^n$ we must have

$$\delta^*(q, b^n) = q_f \in F$$

Nonregular Languages

Example 4.6 Cont.

From this we can conclude that

$$\begin{aligned}\delta^*(q_0, a^m b^n) &= \delta^*(\delta^*(q_0, a^m), b^n) \\ &= \delta^*(q, b^n) \\ &= q_f.\end{aligned}$$

This contradicts the original assumption that M accepts $a^m b^n$ only if n = m,

and leads us to conclude that ***L*** cannot be regular.

Pumping Lemma

The following result, known as the pumping lemma for regular languages, uses the pigeonhole principle in another form.

The proof is based on the observation that in a transition graph with n vertices, any walk of length n or longer must repeat some vertex, that is, contain a cycle.

Pumping Lemma

Theorem 4.8 Let L be an infinite regular language. Then there exists some positive integer m such that any $w \in L$ with $|w| \geq m$ can be decomposed as

$$w = xyz,$$

with

$$|xy| \leq m,$$

and

$$|y| \geq 1,$$

such that

$$w_i = xy^i z, \tag{4.2}$$

is also in L for all $i = 0, 1, 2, \ldots$.

Pumping Lemma

" Even if we could show (and this is normally quite difficult) that any pumped string must be in the original language, there is nothing in the statement of Theorem 4.8 **that allows us to conclude from this that the language is regular**."

Example 4.7

Use the pumping lemma to show that **L = {$a^n b^n$: n ≥ 0}** is **not regular**. Assume that L is regular, so that the pumping lemma must hold. We do not know the value of **m**, but whatever it is, we can always choose **n = m**. Therefore, the substring **y** must consist entirely of a's.

Suppose **|y| = k.** Then the string obtained by using **i = 0** in Equation (4.2) is

$$W_0 = a^{m-k}b^m$$

and is clearly not in L. This contradicts the pumping lemma and thereby indicates that the assumption that L is regular must be false.

Nonregular languages

Example 4.8

Show that

$$L = ww^R : w \in \sum$$

Is Regular

Whatever m the opponent picks on Step 1, we can always choose a w as shown in Figure 4.5. Because of this choice, and the requirement that $|xy| \leq m$, the opponent is restricted in Step 3 to choosing a y that consists entirely of a's.

In Step 4, we use i = 0. The string obtained in this fashion has fewer a's on the left than on the right and so cannot be of the form ww^R . Therefore, ***L*** is not regular.