Computation Theory

Chapter Three: Regular language & Regular Grammar

zied othman 3/6/22 [Course title]

CHAPTER 3: REGULAR LANGUAGES AND REGULAR GRAMMARS

3.1 Regular Expressions

One way of describing regular languages is via the notation of regular expressions. This notation involves a combination of strings of symbols from some alphabet Σ, parentheses, and the operators +, ., and *.

3.1.1 Formal Definition of a Regular Expression

Let Σ be a given alphabet. Then

1. $\emptyset, \lambda$ and $a \in \Sigma$ are all regular expressions. These are called regular expressions.

2 If r_1 and r_2 are regular expressions, so are $r_1 + r_2, r_1.r_2$, , and (r_1).

3. A string is a regular expression if and only if it can be derived from the regular expressions by a finite number of applications of the rules in (2).

The language-defining symbols we are about to create are called **regular expressions**. The languages that are associated with these regular expressions are called **regular languages**.

3.1.2 Languages Associated with Regular Expressions

The language $L(r)$ denoted by any regular expression r is defined by the following rules.

1. $\emptyset$ is a regular expression denoting the empty set,

2. λ is a regular expression denoting $\{\lambda\}$.

ZIED O. AHMED

3. For every $a \in \Sigma$, a is a regular expression denoting $\{a\}$.

If r_1 and r_2 are regular expressions, then

4. $L(r_1 + r_2) = L(r_1) \cup L(r_2)$,

5. $L(r_1 \cdot r_2) = L(r_1) \cup L(r_2)$;

6 $L((r_1)) = L(r_1)$,

7. $L(r_1^*) = (L(r_1))^*$.

Example consider the language L

where $L=\{\Lambda\ x\ xx\ xxx\ \cdots\}$ by using star notation we may write

L=language(x*).

Since x* is any string of x's (including Λ).

Example if we have the alphabet $\Sigma=\{a,b\}$

And $L=\{a\ ab\ abb\ abbb\ abbbb\ \cdots\}$

Then L=language(ab^*)

Example (ab)*= Λ *or ab or abab or ababab or abababab or* $\cdots$.

Example $L1$=language(xx^*)

The language $L1$ can be defined by any of the expressions:

x, xx, xxx, xxxx, xxxxx, ….

Remember x^* can always be Λ.

Example language(ab^*a)=$\{aa\ aba\ abba\ abbba\ abbbba\ \cdots\}$

Example language(a^*b^*)=$\{\ \Lambda\ a\ b\ aa\ ab\ bb\ aaa\ aab\ abb\ bbb\ \cdots\ \}$ ba and aba are not in this language so $a^*b^* \neq (ab)^*$

ZIED O. AHMED

Example consider the language T defined over the alphabet $\Sigma=\{a,b,c\}$

$T=\{a\ c\ ab\ cb\ abb\ cbb\ abbb\ cbbb\ abbbb\ cbbbb\ \cdots\}$

Then T=language($(a+c)b^*$)

T=language(either a or c then some b's)

Example consider a finite language L that contains all the strings of a's

and b's of length exactly three.

$L=\{aaa\ aab\ aba\ abb\ baa\ bab\ bba\ bbb\}$

L=language$((a+b)(a+b)(a+b))$

L=language$((a+b)^3)$

Note from the alphabet $\Sigma=\{a,b\}$, if we want to refer to the set of all possible strings of a's and b's of any length (including Λ) we could write $(a+b)^*$

Example we can describe all words that begins with a and end with b with the expression $a(a+b)^*b$ which mean a(arbitrary string)b

Example if we have the expression $(a+b)^*a(a+b)^*$ then the word $abbaab$ can be considered to be of this form in three ways: $(\Lambda)a(bbaab)$ or $(abb)a(ab)$ or $(abba)a(b)$

Example $(a+b)^*a(a+b)^*a(a+b)^*$ = (some beginning)(the first important a)(some middle)(the second important a)(some end) Another expressions that denote all the words with at least two a's are:

$b^*ab^*a(a+b)^*,\ (a+b)^*ab^*ab^*,\ b^*a(a+b)^*ab^*$

Then we could write:

language$((a+b)^*a(a+b)^*a(a+b)^*)$

language$(b^*ab^*a(a+b)^*)$

language$((a+b)^*ab^*ab^*)$

language$(b^*a(a+b)^*ab^*)$

all words with at least two a's.

Note: we say that two regular expressions are equivalent if they describe the same language.

Example if we want all the words with exactly two *a*'s, we could use the expression: $b^*ab^*ab^*$ which describe such words as *aab, baba, bbbabbabbbb,* …

Example the language of all words that have at least one *a* and at least one *b* is:

$(a+b)^*a(a+b)^*b(a+b)^*+(a+b)^*b(a+b)^*a(a+b)^*$

Note: $(a+b)^*b(a+b)^*a(a+b)^* \neq bb^*aa^*$ since the left includes the word *aba*, which the expression on the right side does not.

Note: $(a+b)^* = (a+b)^* + (a+b)^*$

$(a+b)^* = (a+b)^*(a+b)^*$

$(a+b)^* = a(a+b)^* + b(a+b)^* + \Lambda$

$(a+b)^* = (a+b)^*ab(a+b)^* + b^*a^*$

Note: usually when we employ the star operation, we are defining an infinite language. We can represent a finite language by using the plus alone.

EXERCISES

1. Find all strings in $L((a + b)\ b\ (a + ab)^*)$ of length less than four.

2. Find a regular expression for the set $\{a^n b^m : n \geq 3, m \text{ is even}\}$.

3. Find a regular expression for $L = \{ab^n w : n \geq 3,\ w \in \{a, b\}^+\}$.

4. Give regular expressions for the following languages on $\Sigma = \{a, b, c\}$.

(a) all strings containing exactly one *a*,

ZIED O. AHMED

(b) all strings containing no more than three a's,

(c) all strings that contain at least one occurrence of each symbol in Σ,

5. Write regular expressions for the following languages on {0, 1}.

(a) all strings ending in 01,

(b) all strings not ending in 01,

(c) all strings containing an even number of 0's,

(d) all strings having at least two occurrences of the substring 00. (Note that with the usual interpretation of a substring, 000 contains two such occurrences),

(e) all strings with at most two occurrences of the substring 00,

(f) all strings not containing the substring 101.

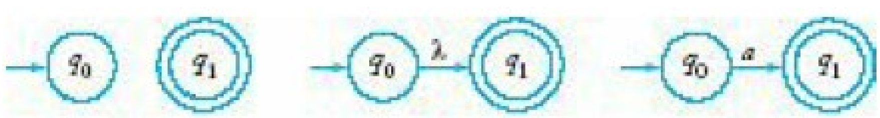
3.2 Connection between Regular Expressions and Regular Languages

for every regular language there is a regular expression, and for every regular expression there is a regular language. We will show this in two parts.

3.2.1 Regular Expressions Denote Regular Languages

Let r be a regular expression. Then there exists some nondeterministic finite automata that accepts $L(r)$. Consequently, $L(r)$ is a regular language.

We begin with automata that accept the languages for the simple regular expressions $\emptyset$, λ, and $a \in \Sigma$. These are shown in Figure (a), (b), and (c), respectively.



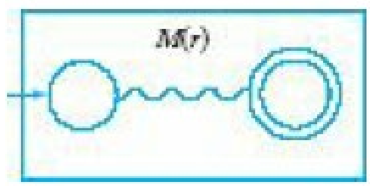
(a) (b) (c)

(a) NFA accepts *Ø*.

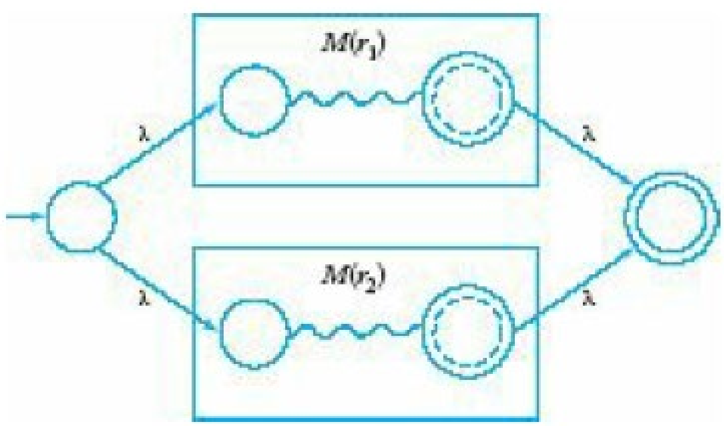
(b) NFA accepts *{ λ }*.

(c) NFA accepts *{a}*.

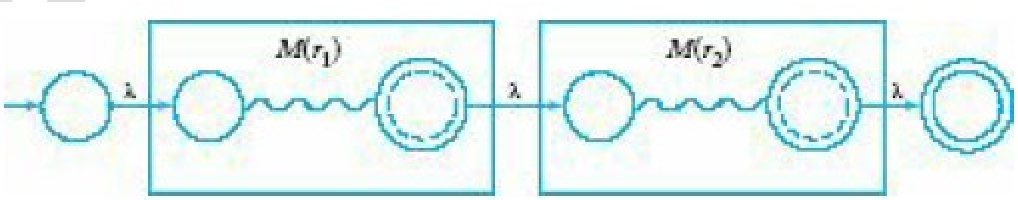
Assume now that we have automata $M\ (r_1)$ and $M\ (r_2)$ that accept languages denoted by regular expressions r_1 and r_2, respectively. We need not explicitly construct these automata, but may represent them schematically, as in Figure below. In this scheme, the graph vertex at the left represents the initial state, the one on the right the final state.



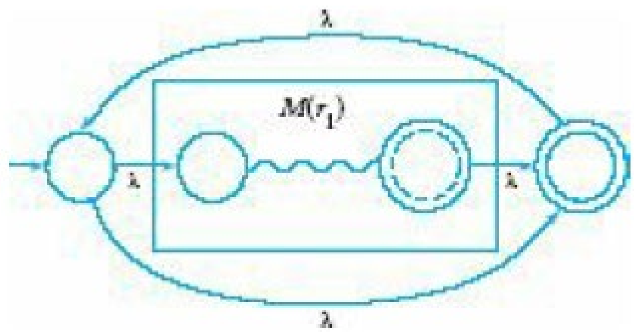
we then construct automata for the regular expressions $r_1 + r_2$, r_1r_2, and r_1*. The constructions are shown in Figures.



Automaton for $L(r_1 + r_2)$.



Automaton for $L(r_1r_2)$.

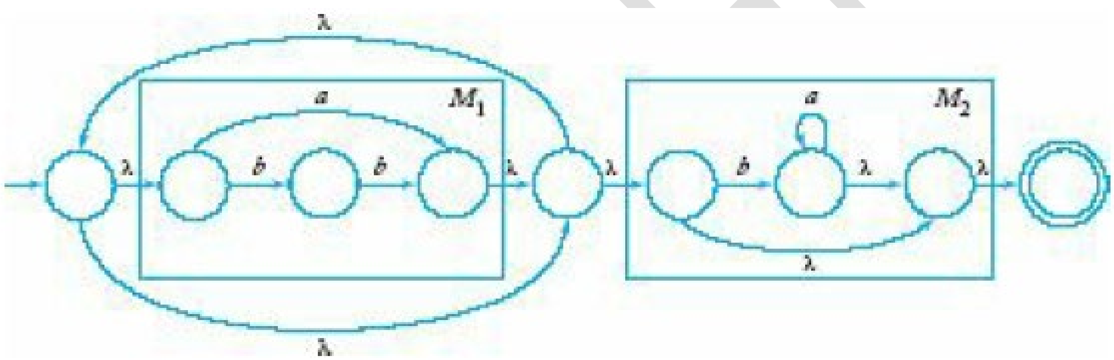


Automaton for $L(r_1*)$.

Example

Find an NFA that accepts *L(r)*, where

$$r=(a + bb)^* (ba^* + \lambda)$$



(a) *M1* accepts *L(a + bb)*.

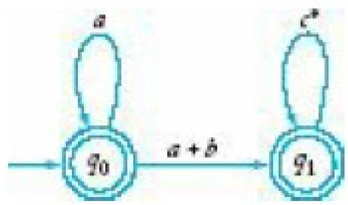
(b) *M2* accepts $L(ba^* + \lambda)$.

3.2.2 Regular Expressions for Regular Languages

A generalized **transition graph** (TG) is a transition graph whose edges are labeled with regular expressions; otherwise, it is the same as the usual transition graph. The label of any walk from the initial state to a final state is the concatenation of several regular expressions, and hence itself a regular expression.

Example

Figure represents a transition graph. The language accepted by it is *L* $(a^* + a^* (a + b)\ c^*)$, as should be clear from an inspection of the graph. The edge (q_o, q_o) labeled *a* is a cycle that can generate any number of *a*'s, that is, it represents *L* (a^*). We could have labeled this edge a^* without changing the language accepted by the graph



Exercise

1. Find DFA's that accept the following languages

(a) a^*

(b) $a + b$

(c) $(a + b)^*$

(d) $a^*\ b$

(e) $b(a + b)^*$

(f) $ab(a + b)^*$

(g) $a\ a^*\ b$

(h) $a^*\ b\ a^*$

(i) $a^*\ bab\ b^*$

(j) $aba + bab$

(k) $(aa)^*\ ba$

(l) contains 2 a

(m) contains even number of a

2. Find an NFA that accepts the language *L* $(ab^*aa + bba^*ab)$.
3. Give an NFA that accepts the language $L((a + b)^*\ b(a + bb)^*)$.
4. Find DFA's that accept the following languages.

   (a) *L* $(aa^* + aba^*b^*)$.

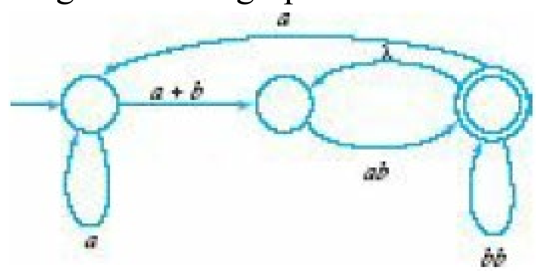
   (b) *L* $(ab\ (a + ab)^*\ (a + aa))$.

   (c) *L* $((abab)^* + (aaa^* + b)^*)$.

   (d) *L* $(((aa^*)^*\ b)^*)$.

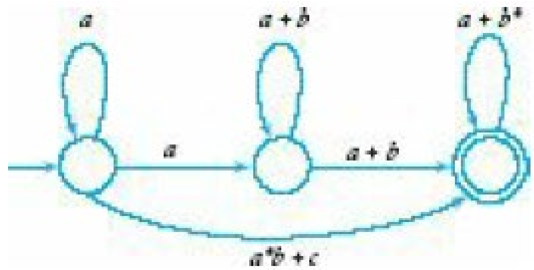
ZIED O. AHMED

5. Consider the following transition graph.



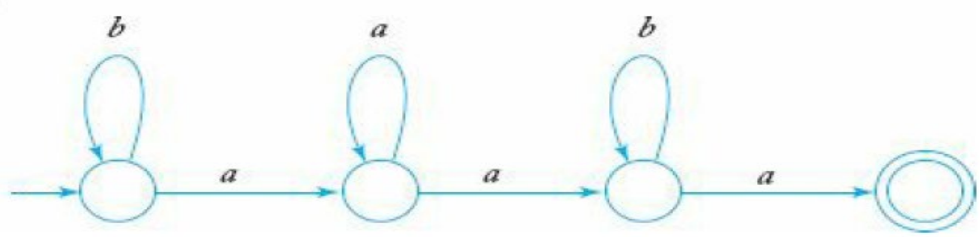
What is the language accepted by this graph?

6. What language is accepted by the following transition graph?

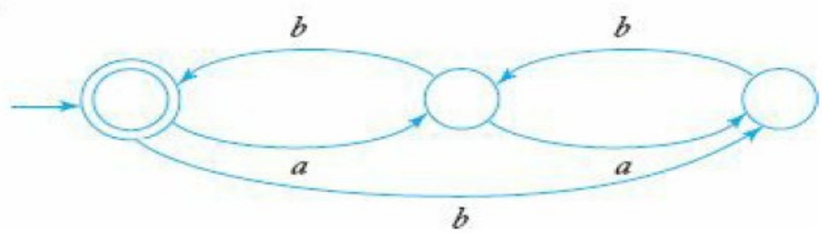


7. Find regular expressions for the languages accepted by the following automata.

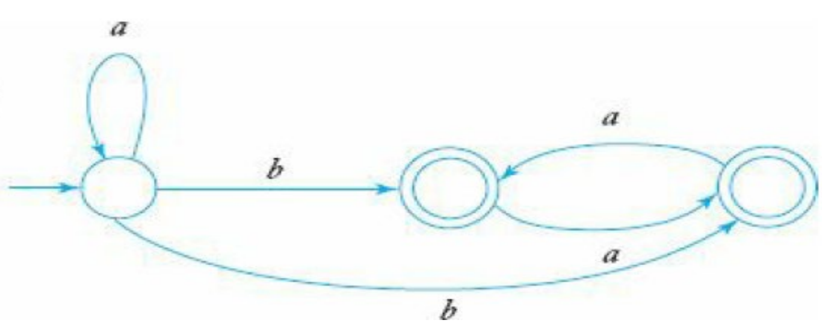
(a)



(b)



(c)



8. Find a regular expression over the alphabet $\{a, b\}$ that contain exactly three a's.
9. Find a regular expression over the alphabet $\{a, b\}$ that end with ab.
10. Find a regular expression over the alphabet $\{a, b\}$ that has length of 3.
11. Find a regular expression over the alphabet $\{a, b\}$ that contain exactly two successive a's.

3.3 Regular Grammars

A third way of describing regular languages is by means of certain **grammars**. Grammars are often an alternative way of specifying languages.

3.3.1 Right- and Left-Linear Grammars

A grammar $G = (V, T, S, P)$ is said to be **right-linear** if all productions are of the form

$$A \to xB,$$

$$A \to x,$$

where $A, B \in V$, and $x \in T^*$. A grammar is said to be **left-linear** if all productions are of the form

$$A \to Bx,$$

or

$$A \to x.$$

A regular grammar is one that is either right-linear or left-linear.

Note that in a regular grammar, at most one variable appears on the right side of any production. Furthermore, that variable must consistently be either the rightmost or leftmost symbol of the right side of any production.

Example

The grammar $G1 = (\{S\}, \{a,b\}, S, P1)$, with $P1$ given as

ZIED O. AHMED

$$S \to abS|a$$

is right-linear. The grammar $G2 = (\{S, S1, S2\}, \{a, b\}, S, P2)$, with productions

$$S \to S1ab,$$

$$S1 \to S1ab|S2,$$

$$S2 \to a,$$

is left-linear. Both G1 and G2 are regular grammars.

Example

The grammar $G = (\{S, A, B\}, \{a, b\}, S, P)$ with productions

$$S \to A$$

$$A \to aB|\lambda,$$

$$B \to Ab,$$

is not regular. Although every production is either in right-linear or left-linear form, the grammar itself is neither right-linear nor left-linear, and therefore is not regular. The grammar is an example of a **linear grammar**.

A linear grammar is a grammar in which at most one variable can occur on the right side of any production, without restriction on the position of this variable. Clearly, a regular grammar is always linear, but not all linear grammars are regular

3.3.2 Right-Linear Grammars Generate Regular Languages

First, we show that a language generated by a right-linear grammar is always regular. To do so, we construct an NFA that mimics the derivations of a right linear grammar. Note that the sentential forms of a right-linear grammar have the special form in which there is exactly one variable and it occurs as the rightmost symbol. Suppose now that we have a step in a derivation

ZIED O. AHMED

$$ab\ldots cD \Rightarrow ab\ldots cdE,$$

arrived at by using a production $D \rightarrow dE$. The corresponding NFA can imitate this step by going from state D to state E when a symbol d is encountered.

Example

Construct a finite automaton that accepts the language generated by the grammar

$$V_0 \rightarrow aV_1,$$
$$V_1 \rightarrow abV_0|b,$$

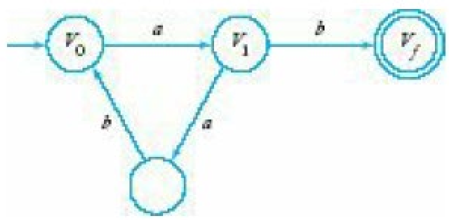
where V_0 is the start variable. We start the transition graph with vertices V_0, V_1, and V_f.

The first production rule creates an edge labeled a between V_0 and V_1.

For the second rule, we need to introduce an additional vertex so that there is a path labeled ab between V_1 and V_0.

Finally, we need to add an edge labeled b between V_1 and V_f, giving the automaton shown in figure.

The language generated by the grammar and accepted by the automaton is the regular language $L((aab)^* ab$.



EXERCISES

1. Construct a DFA that accepts the language generated by the grammar

$$S \rightarrow abA,$$
$$A \rightarrow baB,$$
$$B \rightarrow aA|bb.$$

2. Find a regular grammar that generates the language $L(aa^*(ab+a)^*)$.
3. Construct a left-linear grammar for the language in Exercise 1.
4. Construct right- and left-linear grammars for the language

$$L = \{anbm : n \geq 2, m \geq 3\}.$$

5. find a left-linear grammar for the language accepted by the NFA below.