

Regular Expression: Regular Expression have an important role in computer science applications. In application involving text, users may want to search for strings that satisfy certain pattern. It's a set of symbols, thus if alphabet = {a, b}, then aab, a, baba, bbbbbb, and baaaaa would all be strings of symbols of alphabet. It can be used to define languages and it's like a "pattern"; strings that match the pattern are in the language, strings that do not match the pattern are not in the language. The construction of regular expressions is defined recursively, starting with primitive regular expressions, which can be composed using typical operators to form more complex regular expressions. We include an empty string denoted by Λ which has no symbols in it. For Examples

1^*	is the set of strings $\{\Lambda, 1, 11, 111, 1111, 11111, \dots \text{etc.}\}$
$(1100)^*$	is the set of strings $\{\Lambda, 1100, 11001100, 110011001100, \dots \text{etc.}\}$
$(00+11)^*$	is the set of strings $\{\Lambda, 00, 11, 0000, 0011, 1100, 1111, 000000, 000011, 001100, \text{etc.}\}$
$(0+1)^*$	is all possible strings of zeros and ones, often written as Σ^* where $\Sigma = \{0, 1\}$
$(0+1)^*(00+11)$	is all strings of zeros and ones that end with either 00 or 11.
$(w)^+$	is a shorthand for $(w)(w)^*$ w is any string or expression and the superscript plus, +

1- Concatenation:

if $L1 = \{x, xxx\}$ and $L2 = \{xx\}$ So $(L1.L2)$ means L1 concatenated L2 and it is equal = $\{xxx, xxxxx\}$

Example 1:

$L1 = \{a, b\}$, $L2 = \{c, d\}$

$L1.L2 = \{ac, ad, bc, bd\}$

Example 2:

$\Sigma = \{x\}$.

$L1 = \{\text{set of all odd words over } \Sigma \text{ with odd length}\}$.

$L2 = \{\text{set of all even words over } \Sigma \text{ with odd length}\}$.

$L1 = \{x, xxx, xxxxx, xxxxxxx, \dots\}$.

$L2 = \{\Lambda, xx, xxxx, xxxxxx, \dots\}$.

$L1.L2 = \{x, xxx, xxxxx, xxxxxxx, \dots\}$.

Note: redundant terms are not allowed.

Example 3:

$L1 = \{a, aaa\}$. $L2 = \{aa\}$.

$L1.L2 = \{aaa, aaaaa\}$.

Definition of a Regular Expression

-A regular expression may be the null string

-A regular expression may be an element of the input alphabet

-A regular expression may be the union of two regular expressions

-A regular expression may be the concatenation of two regular expressions

-A regular expression may be the Kleene closure (star) of a regular expression

-A regular expression may be a regular expression in parenthesis

$R = \Lambda$

$R = a$

$R = R1 + R2$

$R = R1R2$

$R = R1^*$

$R = (R1)$

To explain the definitions above

- Epsilon (ϵ) is the empty set denoting zero length string

- Lambda (Λ) is the null string

- 0, 1, a, b, c, are symbols in Alphabet (Σ)

- x is a regular expression, $(xx)(xxx)$ is concatenation, $(xx) \cup (xxx)$ is union and $(xx)^*$ is the Kleene (Kleene Star) closure

- $(\Lambda)(x) = (x)(\Lambda) = x$
- $(\epsilon)(x) = (x)(\epsilon) = \epsilon$
- $(\Lambda) + (x) = (x) + (\Lambda) = x$
- $x + x = x$
- $(\Lambda)^* = (\Lambda)(\Lambda) = \Lambda$
- $(x)^* + (\Lambda) = (x)^* = x^*$
- $(x + \Lambda)^* = x^*$
- $x^* (a+b) + (a+b) = x^* (a+b)$
- $x^* y + y = x^* y$
- $(x + \Lambda)x^* = x^* (x + \Lambda) = x^*$
- $(x + \Lambda)(x + \Lambda)^* (x + \Lambda) = x^*$

Example 4:

$A = \{a,b\}$ // the alphabet is composed of a and b

$A^* = \{\Lambda, a,b,aa,ab,ba,bb,aaa,aab,\dots\}$

Definitions

The symbol $*$ is called the closure (Kleene star).

ϵ (empty set)

Λ (empty string)

$()$ delimiter ,

\cup + union (selection)

concatenation

Given regular expressions x and y , $x + y$ is a regular expression representing the set of all strings in either x or y (set union)

$x = \{a\} \quad y = \{b\} \quad x + y = \{a, b\}$

Example 5

Let $A = \{0,1\}$, $W1 = 00110011$, $W2 = 00000$

$W1W2 = 0011001100000$

$W2W1 = 0000000110011$

$W1 \Lambda = W1 = 00110011$

$\Lambda W2 = W2 = 00000$

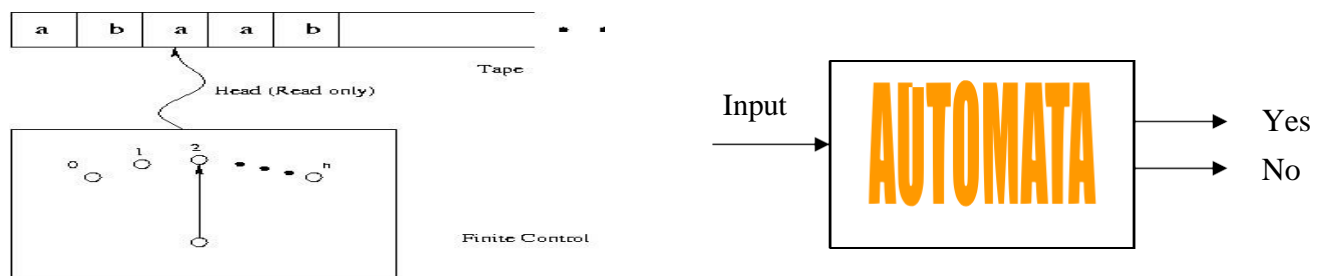
Examples of regular expressions

Describe the language, what is the output (words, strings) of the following RE

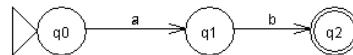
Regular expression	Output (set of strings)
Λ	{ Λ }
Λ^*	{ Λ }
A	{a}
aa	{aa}
a^*	{ Λ , a, aa, aaa, ...}
aa^*	{a, aa, aaa, ...}
a^+	{a, aa, aaa, ...}
$ba^+ = (b aa^*)$	{ba, baa, baaa, ...}
$(ba)^+ = (ba)(ba)^*$	{ba, baba, bababa, ...}
(a b)	{a, b}
$a b^*$	{a, Λ , b, bb, bbb, ...}
$(a b)^*$	{ Λ , a, b, aa, ab, ba, bb, ...}
$aa(ba)^*bb$	{aabb, aababb, aabababb, ...}
(a + a)	{a}
(a + b)	{a, b}
$(a + b)^2$	$(a + b)(a + b) == \{aa, ab, ba, bb\}$
(a + b + c)	{a, b, c}
$(a + b)^*$	{ Λ , a, b, aa, bb, ab, ba, aaa, bbb, aab, bba, ...}
(abc)	{abc}
$(\Lambda + a) bc$	{bc, abc}
ab^*	{a, ab, abb, abbb, ...}
$(ab)^*$	{ Λ , ab, abab, ababab, ...}
$a + b^*$	{a, Λ , b, bb, bbb, ...}
$a(a + b)^*$	{a, aa, ab, aaa, abb, aba, abaa, ...}
$(a + b)^* a (a + b)^*$	{a, aaa, aab, baa, bab, ...}
$(a + \Lambda)^*$	$(a)^* = \{\Lambda, a, aa, aaa, \dots\}$
$x^* (a + b) + (a + b)$	$x^* (a + b)$
$x^* y + y$	$x^* y$
$(x + \Lambda)x^*$	$x^* (x + \Lambda) = x^*$
$(x + \Lambda)(x + \Lambda)^* (x + \Lambda)$	x^*

Finite Automata: is a device consisting of a tape and a control circuit which satisfy the following conditions:

1. The tape starts from left end and extends to the right without an end.
2. The tape is divided into squares in each a symbol.
3. The tape has a read only head.
4. The head moves to the right one square every time it reads a symbol. It never moves to the left. When it sees no symbol, it stops and the automata terminates its operation.
5. There is a control determines the state of the automaton and also controls the movement of the head.



A Deterministic Finite Automata (DFA) represents a finite state machine that recognizes a Regular Expression (RE). For example, the following Finite Automata (FA):



recognize (accept) string ab

A finite automaton consists of a finite set of states, a set of transitions (moves), one start state, and a set of final states (accepting states). In addition, a DFA has a unique transition for every state combination.

it is a set of states, and its “control” moves from state to state in response to external “inputs”.





A finite automaton (FA), provides the simplest model of a computing device. It has a central processor of finite capacity and it is based on the concept of state.

finite state machine is a 5 tuple $M = (Q, A, T, S, F)$.

where

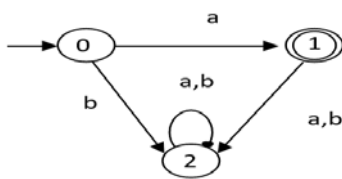
- Q --set of states = {q0, q1, q2,}
- A -- set of input symbols = {a,b, ..., 0, 1, ...}
- T --set of transitions or rules
- S -- an initial state
- F -- the final state -- could be more than one final state

Designing (drawing) Finite Automata (FA)

State with numbers or any name	Start - or small arrow	Final + or double circle	Transition (only one input or symbol on the edge) a,b allowed means (a or b)
			

Example:

$Q = \{ 0, 1, 2 \}$, $A = \{ a, b \}$, $F = \{ 1 \}$, the initial state is 0 and T is shown in the following table.



St	In	In
0	1	2
1	2	2
2	2	2

Transition diagram (TG):

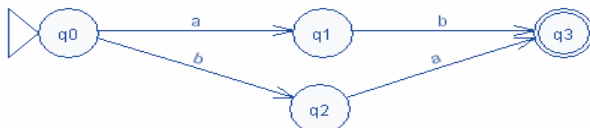
TG has many inputs on the edge ab



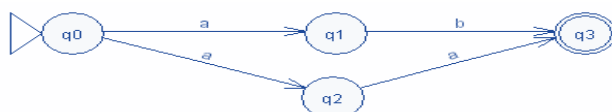
FA has only one input on the edge a

**Deterministic Finite Automata (DFA) and Nondeterministic Finite Automata (NFA)**

DFA: different input from state to different states



NFA: one input from state to different states



University of Basrah







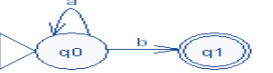



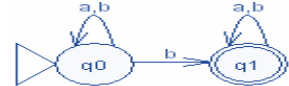


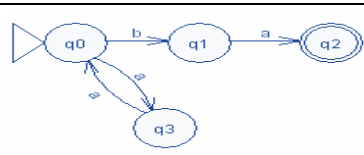
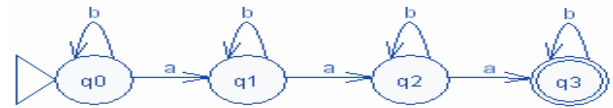
Faculty of Education for Pure Sciences

Computer Department

Language accepted by FA

String is accepted by a FA if and only if the FA starting at the initial state and ends in an accepting state after reading the string.

Examples of languages accepted by FA

FA	RE
	a
	aa
	$a^+ = aa^*$
	a^*
	a+b
	$(a+b)^*$
	a^*b
	$b(a+b)^*$
	$(a+b)^*b$
	$a(a+b)^*b$
	$(a+b)^* b(a+b)^*$
	$ab(a+b)^*$
	a^*babb^*
	$(aa)^*ba$
	contains 3 a's $b^*ab^*ab^*ab^*$