Lecture #4
Languages

Terminology

- Alphabet : a finite set of symbols (ASCII characters)
- String : finite sequence of symbols on an alphabet
  - Sentence and word are also used in terms of string
  - ε is the emptystring
  - $|s|$ is the length of string s.
- Language: sets of strings over some fixed alphabet
  - $\varnothing$ the empty set is a language.
  - $\{\varepsilon\}$ the set containing empty string is a language
  - The set of all possible identifiers is a language.
- Operators on Strings:
  - *Concatenation*: xy represents the concatenation of strings x and y. $s\,\varepsilon = s$ $\quad\varepsilon\, s = s$
  - $s^n = s\,s\,s\,..\,s$ (n times) $\quad s^0 = \varepsilon$

Operations on Languages

- Concatenation: $L_1L_2 = \{ s_1s_2 \mid s_1 \in L_1 \text{ and } s_2 \in L_2 \}$
- Union: $L_1 \cup L_2 = \{ s \mid s \in L_1 \text{ or } s \in L_2 \}$
- Exponentiation: $L^0 = \{\varepsilon\}$ $\quad L^1 = L$ $\quad L^2 = LL$
- Kleene Closure: $L^* =$
- Positive Closure: $L^+ =$

Examples:

- $L_1 = \{a,b,c,d\}$ $\quad L_2 = \{1,2\}$
- $L_1L_2 = \{a1,a2,b1,b2,c1,c2,d1,d2\}$
- $L_1 \cup L_2 = \{a,b,c,d,1,2\}$
- L_1^3 = all strings with length three (using a,b,c,d)
- L^*_1 = all strings using letters a,b,c,d and empty string
- L^+_1 = doesn't include the empty string

Regular Expressions

- We use regular expressions to describe tokens of a programming language.
- A regular expression is built up of simpler regular expressions (using defining rules)
- Each regular expression denotes a language.
- A language denoted by a regular expression is called as a regular set.

For Regular Expressions over alphabet Σ

Regular Expression	Language it denotes
ε	$\{\varepsilon\}$
$a \in \Sigma$	$\{a\}$
$(r_1) \mid (r_2)$	$L(r_1) \cup L(r_2)$
$(r_1)(r_2)$	$L(r_1)\, L(r_2)$ $(r)^*$ $(L(r))^*$
(r)	$L(r)$

- $(r)^+ = (r)(r)^*$
- $(r)? = (r) \mid \varepsilon$
- We may remove parentheses by using precedence rules.
  - $*$ highest
  - concatenation next
  - $\mid$ lowest
- $ab^*|c$ means $(a(b)^*)|(c)$

Examples:

- $\Sigma = \{0,1\}$
- $0|1 = \{0,1\}$
- $(0|1)(0|1) = \{00,01,10,11\}$
- $0^* = \{\varepsilon, 0,00,000,0000,\ldots\}$
- $(0|1)^*$ = All strings with 0 and 1, including the empty string

Finite Automata

- A *recognizer* for a language is a program that takes a string x, and answers "yes" if x is a sentence of that language, and "no" otherwise.
- We call the recognizer of the tokens as a *finite automaton.*
- A finite automaton can be: *deterministic (DFA)* or *non-deterministic (NFA)*
- This means that we may use a deterministic or non-deterministic automaton as a lexical analyzer.

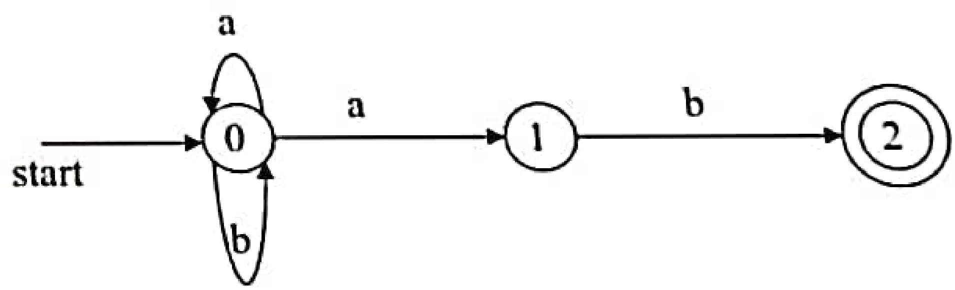
- Both deterministic and non-deterministic finite automaton recognize regular sets.
- Which one?
  - deterministic – faster recognizer, but it may take more space
  - non-deterministic – slower, but it may take less space
  - Deterministic automatons are widely used lexical analyzers.
- First, we define regular expressions for tokens; Then we convert them into a DFA to get a lexical analyzer for our tokens.

Non-Deterministic Finite Automaton (NFA)

- A non-deterministic finite automaton (NFA) is a mathematical model that consists of:
  - S - a set of states
  - Σ - a set of input symbols (alphabet)
  - move - a transition function move to map state-symbol pairs to sets of states.
  - s_0 - a start (initial) state
  - F- a set of accepting states (final states)
- ε- transitions are allowed in NFAs. In other words, we can move from one state to another

  one
- without consuming any symbol.
- A NFA accepts a string x, if and only if there is a path from the starting state to one of accepting states such that edge labels along this path spell out x.

Example:



Transition Graph

0 is the start state s0
{2} is the set of final states F

Σ = {a,b}

S = {0,1,2}

Transition Function:

	a	b
0	{0,1}	{0}
1	{}	{2}
2	{}	{}

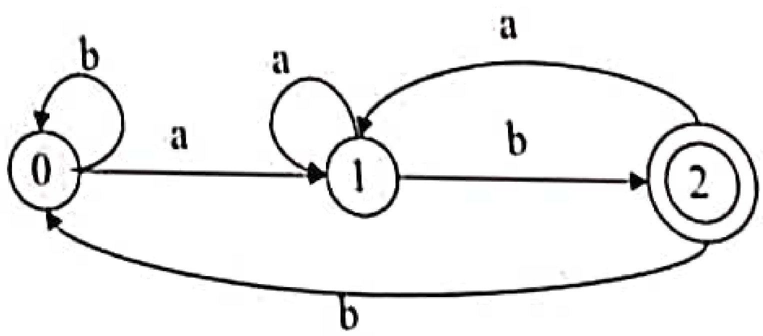
The language recognized by this NFA is (a|b)*ab

Deterministic Finite Automaton (DFA)

- A Deterministic Finite Automaton (DFA) is a special form of a NFA.
- No state has ε- transition
- For each symbol a and state s, there is at most one labeled edge a leaving s. i.e. transition
- function is from pair of state-symbol to state (not set of states)

Example:

The DFA to recognize the language (a|b)* ab is as follows.



0 is the start state s0
{2} is the set of final states F

Σ = {a,b}

S = {0,1,2}

Transition Function:

	a	B
0	1	0
1	1	2
2	1	0

Note that the entries in this function are single value and ...