definitions

Barry

December 12, 2016

Contents

1 Definitions

1.1 DFA:

DFA M = (Q,Σ,δ,q_0,F) consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)

- a transition function (δ: Q x $\Sigma \rightarrow$ Q)
- an initial or start state ($q_0 \in$ Q)

- a set of accepting states (F $\subseteq Q$)
  a string w over the alphabet Σ is accepted by M if a sequence of states, r_0,r_1,...,r_n exists in Q with the following conditions:

  1. $r_0 = q_0$
  2. $r_{i+1} = \delta(r_j, a_{i+1})$, for i = 0,...,n-1
  3. $r_n \in$ F

1.2 NFA:

NFA M = (Q,Σ,δ,q_0,F) consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function (δ: Q x $\Sigma \rightarrow 2^Q$)
- an initial or start state ($q_0 \in$ Q)

- a set of accepting states (F $\subseteq Q$)
  The same rules govern a word w's acceptance by M as with a DFA

1.3 ϵ-NFA:

An ϵ-NFA M = (Q,Sigma,δ,q_0,F) consisting of

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a transition function (δ: Q x ($\Sigma \cup \epsilon$) $\rightarrow 2^Q$)
- an initial or start state ($q_0 \in$ Q)

- a set of accepting states (F $\subseteq Q$)
  The same rules govern a word w's acceptance by M as with a DFA

1.4 Grammar:

A grammar G = (N,Σ,P,S) where

- a finite set of nonterminal symbols that is disjoint with the strings formed by G (N)
- a finite set of terminal symbols that is disjoint from N (Σ)
- a finite set of production rules (P) each of the form:

$$(\Sigma \cup N)^* \; N(\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$$

- a distinguished symbol $S \in N$ that is the start symbol

1.4.1 Right Linear

All nonterminals in the righthand side of productions are at the right ends

1.4.2 Left Linear

All nonterminals in the righthand side of productions are at the left ends

1.4.3 Linear Grammar

All of the productions in a linear grammar are right or left linear (not neccesarily all the same)

1.4.4 Regular Grammar

A left or right linear grammar

1.4.5 context free

a context free grammar imposes the following rules on elements of P

$$A \rightarrow x \text{ where } A \in V \text{ , } x \in (V \text{ U } T)^*$$

1.5 Pumping Lemma for Regular languages:

let L be a regular language. then there exists a constant $n \geq 1$ st. if w is any string in L st. $|w| \geq n$, we can find x,y,z st. w = xyz

1. $|xy| \leq n$
2. $y \neq \epsilon$
3. for all $k \geq 0$, $xy^kz \in L$

1.6 pumping lemma for Context free languages

let L be a context free language. then there exists a constant n $\geq$ 1 st. if z is any string in L st. |z| $\geq$ n, we can find u,v,w,x,y st. z = uvwxy

1. |vwx| $\leq$ n
2. vx $\neq \epsilon$
3. for all i $\geq$ 0, $uv^iwx^iy \in$ L

1.7 PDA

PDA M = (Q,Σ,Γ,δ,Z,F) consisting of:

- a finite set of states (Q)
- a finite set of input symbols called the alphabet (Σ)
- a finite set of stack symbols called the stack alphabet (Γ)
- a transition relation $\delta \subseteq Qx(\Sigma \cup \{\epsilon\})$ x Γ x Q x Γ^*
- an initial or start state ($q_0 \in$ Q)
- Z $\in \Gamma$ is the inital stack symbol

- a set of accepting states (F $\subseteq Q$)
  An element (p,a,A,q,α) $\in \delta$ is a transition of M. It has the intended meaning that M, in state p $\in$ Q, on the input a $\in \Sigma \cup \{\epsilon\}$ and with A $\in \Gamma$ as topmost stack symbol, may read a, change the state to q, pop A, replacing it by pushing $\alpha \in \Gamma^*$.

1.8 Turing Machine

A Turing Machine M = (Q,Γ,b,Σ,δ,q_0,F) consisting of:

- a finite set of states (Q)
- a finite, non-empty set of tape alphabet symbols (Γ)
- a blank symbol, the only symbol allowed to occur on the tape infinitely often at any step (b $\in \Gamma$)
- a set of input symbols ($\Sigma \subseteq \Gamma$ - {b})

  - a partial function δ: Q x $\Gamma \rightarrow$ Q x Γ x {L,R}
  - an initial state (q_0)
- a set of final or accepting states (F $\subseteq Q$)

1.8.1 Turing Machine ID

the ID of a turing machine = αqB where α is the symbols to the left, q is the current state, B is the curent symbol and the everything to the right.

1.9 Recursivly Enumerable Language

a recursivly enumerable language is computable by a turing machine and will halt if there is an answer

1.10 Recursive Language

a recursive language is computable by a turing machine and the machine will eventually halt

1.11 Chomsky hierarchy

each level contains the levels below it

- Recursively Enumerable
  - Turing Machines
- Context Sensitive
  - Linear Bounded Turing Machine
- Context Free
  - Context Free Grammar
  - Pushdown Automata
- Regular
  - Regular Expression
  - DFA
  - NFA
  - ϵ-NFA
  - Regular Grammar