CS 373 Notes

15 March 2012

Contents

1 General **1**
- 1.1 Starting off 1
- 1.2 Strings 1
- 1.3 Language 2

2 Regular Languages **2**
- 2.1 Deterministic Finite Automotas 2
- 2.2 Non-Deterministic Finite Automotas (NFA) 3
- 2.3 Regular Expressions 4
- 2.4 Generalized NFA (GNFA) 4
- 2.5 Pumping Lemma for regular languages 5
- 2.6 Subsitutions 5
  - 2.6.1 Substution simple definition 5
  - 2.6.2 Homomorphism 5
  - 2.6.3 Inverse Homorphism 5
- 2.7 DFA Minimization 5
  - 2.7.1 Theory 5
  - 2.7.2 Algorithm 6
- 2.8 Reg Operations (closed under the Reg languages) 6

3 Context Free Grammars **7**
- 3.1 Formal Definition: 7
- 3.2 Chomsky Normal form (CNF) 7
- 3.3 Deterministic Push Down Automotas 8
- 3.4 Non-Deterministic Push Down Automotas (PDA) 8
- 3.5 Relating PDA to CFL 8
- 3.6 Pumping Lemma for CFL's 8
- 3.7 Closure Properties of CFL's 8

3.7.1 Theorems for Closure 8
3.8 CYK algorithm 9

1 General

Sizes	Examples	Countable?
Finite	{a,b}	yes
Countable Infinite	N,Z, Q	yes
Uncountable Infinity	R, Pow(R)	no

name	descriptipn	Machine
regular	LRk	D PDA
context free language	CFG	PDA

1.1 Starting off

1. Alphabet(Σ) = finite non empty set
2. N in this class starts at 0
3. A set X is countably infinite iff $\exists$ a bijection $f : \mathbb{N} \rightarrow X$

1.2 Strings

1. String(w) = sequence of characted in Σ
2. w:{c_i $\in\Sigma$ | 0 <= i <= n}
3. |w| = n = length of the string
4. |w| = 0 $\rightarrow w = \epsilon$

(a) Careful $\sigma \neq \emptyset$

5. Substring subsequence of characters in w
6. Concatination: $w_1 \cdot w_2$
7. Reverse: w^r
8. Palindrum: w = w^r

1.3 Language

1. Language(L) = set of strings
2. Σ^n = {w : |w| = n}
3. Σ^0 = { ϵ }
4. Σ^* =$\cup^n$$_{\mathrm{i=0}}$ Σ^i, Language of all strings

2 Regular Languages

2.1 Deterministic Finite Automotas

1. Finite state machine (M)
2. Takes a string of inputs
3. 2 types of states
   (a) Accept
   (b) Deny
4. There is 1 start state
5. The set of all strings accepted by language of M or L(A)
6. Formal Definition
   (a) a Language A $\in$ Σ^* is called regular iff there exists a DFA ,M, s.t. $L(M) = A$
   (b) DFA is a 5 tuple $ M = (Q,Σ, δ, q_0, F)$
      i. Q is a finte set of states
      ii. Σ is a finite alphabet
      iii. $\delta : Q \times \Sigma \to Q$ is the transition function
      iv. $q_0 \in Q$ is the inital state
      v. $F \subset Q$ is the set of accept states
   (c) L(M) $\equiv$ language of all accepted strings
7. Closure properties/Regular Operations on languages
   (a) A_1 and A_2 are regular

(b) Union: $A_1 \cup A_2 = A_3$

(c) Concatenate: $A_1 A_2 = A_3$

(d) Star: $A_1^* = A_3$

2.2 Non-Deterministic Finite Automotas (NFA)

1. Formal Definition

   (a) M = (Q,Σ,δ,q_0,F)

      i. Q = finite set of states

      ii. Σ is a finate alphabet

      iii. $\delta = Q \times \Sigma_\epsilon \rightarrow Pow(Q)$

         A. $\Sigma_\epsilon = \Sigma \cup \epsilon$

      iv. q_0 = start state

      v. $F \subset Q$

   (b) NFA accepts w If we can write $w = y_1 y_2 ... y_n y_i \in \Sigma_\epsilon$ s.t. there exists a sequence of states path $R = r_0, r_1, ...$

      i. $r_0 = q_0$

      ii. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, 1 ... m - 1$

      iii. $r_m \in F$

2. Useful Lemma: For all NFA, M, there exists an DFA ,N, s.t. L(m) = L(n)

2.3 Regular Expressions

1. Def: R is a regex over a fixed alphabet iff one of the following is true:

   (a) R = a ∈Σ

   (b) R = σ

   (c) $R = \emptyset$

   (d) \$R = $R_1 \cup R_2$,\$ given $R_1 \wedge R_2$ are regex

   (e) \$R = $R_1 R_2$,\$ given $R_1 \wedge R_2$ are regex

   (f) $R = R^*$

2. Order of operations

(a) star
(b) concatination
(c) union

3. Identities

   (a) $a\emptyset = \emptyset$
   (b) $a\sigma = \sigma$
   (c) $\emptyset^* = \sigma$

2.4 Generalized NFA (GNFA)

1. Definition

   (a) Q = set of all states
   (b) $Q^0 = Q - q_start, q_accept$
   (c) The start state has out edges to every $q \in Q - q_start$, and no in edges
   (d) The accept state has inedges from every $q \in Q - q_accept$, and no outedges
   (e) An edege exists from every $q_1 \in Q^0$ to every $q_2 \in Q^0$ even if $q_1 = q_2$
   (f) Every edge is labed with a regex

2. Useful lemma: Any NFA can be written as a GNFA

3. lemma: Given a GNFA, M, with 2 states, the regex between the 2 states describes the language of M

2.5 Pumping Lemma for regular languages

If A is regular, then $\exists p \in \mathbb{N}$ s.t. $\forall s \in A$ for which $|s| >= p, s$ can be written as xyz and satisfy the following conidition:

1. $\forall i >= 0, xy^i z \in A$

2. $|y| > 0$ i.e. $y \neq \epsilon$

3. $|xy| <= p$

P is called the "pumping length

2.6 Subsitutions

2.6.1 Substution simple definition

- A is a reg language and $A \mapsto f(A), A \subseteq E^*$
- A s described w. a regex and R_a is a regex using Γ
- $\forall a \in \Sigma a \mapsto R_a$
- $\epsilon \mapsto \epsilon$ and $\emptyset \mapsto \emptyset$

2.6.2 Homomorphism

- $A \mapsto h(A)$
- $a \mapsto w, w \in \Gamma^*, a \in \Sigma$

2.6.3 Inverse Homorphism

- $h^{-1}(A) = \{w \ \in \Sigma^* | h(w) \in A\}$

2.7 DFA Minimization

2.7.1 Theory

Problem: Given a DFA, M, with L(M) = A, find another DFA, M_{2c}, s.t. L(M) = L(M_2) and $|Q_2|$ is as small as possible

- \$$\delta$: Q $\times$ $\Sigma \to Q$\$
  $\bar{\delta}(q, w) q \in Q, w \in \Sigma^*$
  $\bar{\delta}(q, w) \equiv$ interative call on delta for all w_i in w
  If $\exists w \in \Sigma^*$ s.t. $[\bar{\delta}(p, w) \in F$ and $\delta q, w \notin F]$ or $[\bar{\delta}(p, w) \notin F$ and $\delta q, w \in F]$ then p and q are distrguishable

2.7.2 Algorithm

```
for (p,q) in Q^2:
    if (p in F) and (not q in F):
        A.push((p,q)) # marked list
    else:
        B.push((p,q)) # unmarked list
for (p,q) in B:
```

```
if (delta(p,a),delta(q,a)) in B:
    A.push((p,q))
```

2.8 Reg Operations (closed under the Reg languages)

1. $A_1 \cup A_2$
2. $A_1 - A_2$
3. $\bar{A}_1 = \Sigma^* - A_1$
4. $A_1 \cap A_2$
5. Symmetric Diff
6. $A_1 A_2$
7. A_1^*
8. A^r
9. Reg langagues are clased under subsitution
10. Reg langagues are clased under homomorphism
11. Reg langagues are clased under inverse homorphism
12. Reg langagues are clased under

3 Context Free Grammars

3.1 Formal Definition:

1. (V, Σ, R, S)
   (a) V = Finite set of <u>variables</u> or "non-terminals"
   (b) Σ = finite set of <u>terminals</u>
      i. $\Sigma \cap V = \emptyset$
      ii. Convention: Variables are uppercase, symbols are lowercase
   (c) R = finite set of <u>rules</u> or "subsitution rules" or "productions" 1.Rules: examples
      i. $A \to aaBc|a$

A. This means the for an A you can replace it with aaBc or a

ii. $A \Rightarrow OA1 \Rightarrow 00A11 \Rightarrow 001A011 \Rightarrow 001011$

(d) S is the <u>start variable</u>

2. $L(G) = \{w \in \Sigma^* | S \Rightarrow^* w\}$

3. Notation:

(a) Variables: A,B,C...

(b) Terminal: a,b,c,...0,1,\$ ϵ \$

(c) $U \Rightarrow^* V$ is defined as $\exists$ sequence $U_1..U_k$, s.t. $U \Rightarrow U_1 \Rightarrow U_2 \Rightarrow ... \Rightarrow U_k \rightarrow V$

3.2 Chomsky Normal form (CNF)

- All rules have the form
  - $A \rightarrow BC$, where B, C cannot be S
  - $A \rightarrow a$
  - if $A \rightarrow \epsilon$ then $A = S$
- Lemma: Any CFG can be written in CNF

3.3 Deterministic Push Down Automotas

- $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
  - $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow Q \times \Gamma_\epsilon$

3.4 Non-Deterministic Push Down Automotas (PDA)

- $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
  - $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow Pow(Q \times \Gamma_\epsilon)$

3.5 Relating PDA to CFL

- A language is context free iff $\exists$ a PDA that recognizes it
  - Lemma: If A is CF, then $\exists$ a PDA,M, s.t. $A = L(M)$
  - Lemma: $\forall$ PDA, M, $\exists$ CFL, G, s.t. $L(G) = L(M)$
    - * Proof Idea: Make a conical PDA (while preserving acceptance) as follows
      1. 1 accept states
      2. Stack is empty when accepting
      3. Every transition either push or pops but not both

3.6 Pumping Lemma for CFL's

3.7 Closure Properties of CFL's

1. $A_1 \cap A_2$
2. $A_1 \cdot A_2$
3. $A_1{}^\star$
4. Closure under substituion

3.7.1 Theorems for Closure

Let $G_i = (V_i, \Sigma_i, R_i, S_i)$ for i=1,2 and $A_i = L(G_i)$
Without loss of generality, assume $V_1 \cap V_2 = \emptyset \wedge S_3$

- Theorem: If A_1 and A_2 are CFL's, then $A_1 \cup A_2$ is a CFL

  Proof:

  Let $G_i = (V_i, \Sigma_i, R_i, S_i)$ for i=1,2 and $A_i = L(G_i)$
  Without loss of generality, assume $V_1 \cap V_2 = \emptyset \wedge S_3 \notin V_1 \cup V_2$

  Construct $G_3 = (V_1 \cup V_2 \cup \{S_3\}, \Sigma_1 \cup \Sigma_2, R_3, S_3)$ with
  $R_3 = R_1 \cup R_2 \cup \{S_3 \rightarrow S_1 | S_2\}$. $\square$

- Theorem: If A_1 and A_2 are CFL's then $A_1 \cdot A_2$ is a CFL

Proof:

$\notin V_1 \cup V_2$ Construct $G_3 = (V_1 \cup V_2 \cup \{S_3\}, \Sigma_1 \cup \Sigma_2, R_1 \cup R_2 \cup \{S_3 \rightarrow S_1$
$S_2\}, S_3)$

- Theorem: If A_1 and A_2 are CFL's then $A_1{}^{\star}$ is a CFL

  Proof: Construct $G_3 = (V_1 \cup \{S_3\}, \Sigma_1, R_1 \cup \{S_2 \rightarrow S_1\ S_2 \mid \epsilon\ \}\ )$

3.8 CYK algorithm