CS 373 Notes

10 June 2012

Contents

1 General **1**
- 1.1 Chomsky Heirarchy . . . 1
- 1.2 Starting off . . . 1
- 1.3 Strings . . . 1
- 1.4 Language . . . 2

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

3 Context Free Langagues **6**
- 3.1 Formal Definition: . . . 6
- 3.2 Chomsky Normal form (CNF) . . . 6
- 3.3 Deterministic Push Down Automotas . . . 7
- 3.4 Non-Deterministic Push Down Automotas (PDA) . . . 7
- 3.5 Relating PDA to CFL . . . 7
- 3.6 Pumping Lemma for CFL's . . . 7
- 3.7 Closure Properties of CFL's . . . 8
  - 3.7.1 Theorems for Closure . . . 8
- 3.8 CYK algorithm . . . 8
  - 3.8.1 Dynamic Programming (sequential optimization) . . . 8
  - 3.8.2 Algorithm . . . 8

4 Context Sensitive Languages **9**
- 4.1 Formal Definition: Non-Contracting Grammars . . . 9
- 4.2 Normal Form Definition . . . 9
- 4.3 Linear Bounded Automaton . . . 9
  - 4.3.1 Informal Definition . . . 9

5 Turing Machines **10**
5.1 Formal Def . . . 10
5.2 Configurations . . . 10
5.2.1 Halting Configs . . . 10
5.3 Turning Recognizable . . . 10
5.4 Turing Decidable . . . 10
5.4.1 Co-Recognizablity . . . 11
5.5 Turing Machine Variants: . . . 11
5.6 Universial Turing Machine . . . 11
5.7 Undecidability . . . 11
5.7.1 Halting Problem . . . 11
5.8 Reduction . . . 12
5.8.1 Map Reductions . . . 12
5.8.2 Rice's Theorem: . . . 12

6 Kolmogorov Complexity **12**

7 Complexity Theory: **12**
7.1 Definition: . . . 12
7.2 Complexity Classes . . . 13
7.2.1 P vs NP . . . 13

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 Chomsky Heirarchy

Type	Grammar	Rules	Machine
0	Unrestricted	$\alpha \rightarrow \beta$	Turning Machines (recognizable)
1	Context-Sensitive	$\alpha \rightarrow \beta$, $\lvert\beta\rvert \geq \lvert\alpha\rvert$	non-det LBA
2	Context-Free	A $\rightarrow \alpha$	non-det PDA
$3^\star$	Regular	A $\rightarrow a, A \rightarrow aB$	DFA/NFA

- S $\rightarrow \epsilon$ also allowed, but S then cannot appear on the right side of rules

1.2 Starting off

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

1.3 Strings

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

1.4 Language

1. Language(L) $=$ set of strings
2. $\Sigma^n = \{w : |w| = n\}$
3. $\Sigma^0 = \{\epsilon\}$
4. $\Sigma^\star = \cup_{i=0}^{n} \Sigma^i$, 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 $\in \Sigma^\star$ is called regular iff there exists a DFA ,M, s.t. $L(M) = A$
   (b) A DFA is a 5 tuple $M = (Q, \Sigma, \delta, q_0, F)$
      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: $A_1^\star = A_3$

2.2 Non-Deterministic Finite Automotas (NFA)

1. Formal Definition

   (a) $M = (Q, \Sigma, \delta, q_0, F)$

      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) $R = a \in \Sigma$

   (b) $R = \sigma$

   (c) $R = \emptyset$

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

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

   (f) $R = R^\star$

2. Order of operations

   (a) star

   (b) concatination

   (c) union

3. Identities

   (a) $a\emptyset = \emptyset$

   (b) $a\epsilon = a$

   (c) $\emptyset^\star = \{\epsilon\}$

2.4 Generalized NFA (GNFA)

1. Definition

    (a) Q = set of all states

    (b) $Q^0 = Q - q_{start}, q_{accept}$

    (c) <u>The</u> start state has out edges to every $q \in Q - q_{start}$, and no in edges

    (d) <u>The</u> 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 $A \mapsto f(A), A \subseteq \Sigma^\star$

- A is 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

- $\delta : Q \times \Sigma \to Q$
- $\bar{\delta}(q, w) q \in Q, w \in \Sigma^\star$
- $\bar{\delta}(q, w) \equiv$ interative call on delta for all w_i in w
- If $\exists w \in \Sigma^*$ s.t. $[\delta(\bar{p}, w) \in F$ 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. $\bar{A}_1 = \Sigma^\star - A_1$
4. $A_1 \cap A_2$
5. Symmetric Diff
6. $A_1 A_2$
7. $A_1^\star$
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

2.9 Right Invariant Equivilance

Def: An equivilance relation is called right invariant or concatenation invariant iff $x \sim y \implies \forall w \in \Sigma^\star xw \sim yw$

2.10 Myhill-Nerod Theorem

- Claim: The following statments are equivalent

  1. A is a regular language
  2. $\exists$ a right invariant equivilance relation that has a finite index, and $A =$ union of some of the equivilance classes
  3. $\overset{A}{\sim}$ is of finite index

- Proof: $1 \to 2$

  1. Let M be any DFA, $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $A = L(M)$
  2. Let $\overset{A}{\sim}$ be define as in Right Invariant Equivilance
  3. The number of equivilance classes is $\leq |Q|$
  4. A is then the union of classes that correspond to F $\square$

- Proof: $2 \to 3$

  1. Show that the partition of $\Sigma^\star$ produced by a right invariant is a refinement of the parition induced by $\overset{A}{\sim}$
  2. Since $\sim$ is right invariant, $\forall z \in \Sigma^\star, xz \sim yz$ which implies that $xz \in A$ iff $yz \in A$ which by definition implies $x \overset{A}{\sim} y$

- Proof: $3 \to 1$

  1. Construct a DFA using $\overset{A}{\sim}$
  2. Let $Q \equiv$ set of equivilance classes of $\overset{A}{\sim}$
  3. Let $[x] \in \Sigma^\star, [x] \in Q$, denote the equivilance class that x belongs to
  4. Let $\delta([x], a) \equiv [xa]$, by def of $\overset{A}{\sim}$
  5. Let $q_0 \equiv [\epsilon]$ and $F \equiv [x] : x \in A$

3 Context Free Langagues

3.1 Formal Definition:

1. (V, Σ, R, S)

   (a) $V =$ Finite set of variables or "non-terminals"

   (b) $\Sigma =$ finite set of terminals

      i. $\Sigma \cap V = \emptyset$

      ii. Convention: Variables are uppercase, symbols are lowercase

   (c) R = finite set of rules 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 start variable

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

3. Notation:

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

    (b) Terminal: a,b,c,...0,1,ϵ

    (c) $U \overset{\star}{\Rightarrow} V$ 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

Theorem: If A is a CFL, then $\exists p \geq 0, p \in \mathbb{Z} s.t. \forall s \in A : |S| \geq p \implies \exists$ a paritition $s = uvxyz$ that satisfy the follow conditions

1. $\forall i \geq 0, uv^ixy^iz \in A$
2. $|vy| > 0$
3. $|vxy| \leq p$

Proof:

1. Let G be a CFG s.t. $A = L(G)$
2. Let b be the max length of the right side of a rule $\in R$ and assume $b \geq 2$
   (a) If $b < 2$ the language must be finite thus the pumping lemma is trivially true
3. Consider the derivation tree if the tree height is h, then the length of the generated string,s, $\leq b^h$
4. Let p = $\mathrm{b}^{|\mathrm{V}|+1}$, where V = set of variables
5. Observe that for any $s \in A$ that $|S| \geq p \implies h \geq |v| + 1$
6. Choose the 'smallest' derivation tree by height for s
7. The longest path has length $|v| + 1$ and vists $|v| + 2$ variables
8. Note that $|v| <$ variables visted, thus by the **Pidgen Hole Principle** there must be at least 1 variable repeated
9. Thus There is a cycle in the production process strings which can then be repeated an indefiniate amount of times in the form $uv^ixy^iz\Box$

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 $\mathrm{A}_1 \cup \mathrm{A}_2$ is a CFL

  Proof:

  1. Let $G_i = (V_i, \Sigma_i, R_i, S_i)$ for $i = 1, 2$ and $A_i = L(G_i)$
  2. Without loss of generality, assume $V_1 \cap V_2 = \emptyset \wedge S_3 \notin V_1 \cup V_2$
  3. Construct $G_3 = (V_1 \cup V_2 \cup \{S_3\}, \Sigma_1 \cup \Sigma_2, R_3, S_3)$ with $R_3 = R_1 \cup R_2 \cup \{S_3 \to S_1 | S_2\}.\Box$

- Theorem: If A_1 and A_2 are CFL's then $\mathrm{A}_1 \cdot A_2$ is a CFL

  Proof:

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

- Theorem: If A_1 and A_2 are CFL's then $\mathrm{A}_1{}^{\star}$ is a CFL

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

3.8 CYK algorithm

3.8.1 Dynamic Programming (sequential optimization)

- Richard Bellman 1950
- sequential decision making
- extensive form games
- optimal control theory
- Dijkstra's Algorithm

3.8.2 Algorithm

Is $G \stackrel{\star}{\Rightarrow}$ w true or false?

$G = (V, \Sigma, R, S)$, Put G into Chomsky Normal Form, $w \in \Sigma^\star$

Cocke, Schwartz, Younger, Kasame

```
"""
Preconditions:
let the input be a string S consisting of n characters: a1 ... an.
let the grammar contain r nonterminal symbols R1 ... Rr.
This grammar contains the subset Rs which is the set of start symbols.
let P[n,n,r] be an array of booleans. Initialize all elements of P to false.
"""

for each i = 1 to n:
    for each unit production Rj -> ai:
        P[i][1][j] = true
for each i = 2 to n: # Length of span
    for j in range(1,n-i+2): # Start of span
        for k in range(1,i): # Partition of span
            A = filter(RA -> RB RC, G) # A = list of productions s.t. RA -> RB RC
            for production in A:
                if P[j][k][B] and P[j+k][i-k][C]:
                    P[j][i][A] = True
if any P[1][n]: #x is iterated over the set s, where s are all the indices for Rs)
    print 'S is member of language'
else:
    print 'S is not member of language'
```

4 Context Sensitive Languages

4.1 Formal Definition: Non-Contracting Grammars

$G = (V, \Sigma, R, S)$

1. V is finite set of <u>variables</u>
2. Σ is a finite set of <u>terminals</u> and $\Sigma = \emptyset$
3. $\alpha \rightarrow \beta$
4. $|\alpha| \leq |\beta|$

4.2 Normal Form Definition

3) R is a finite set of rules of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$ in which A is a variable and α, β, γ are strings of terminals and variables.

1. $\alpha, \beta \in (V \cup \Sigma)^{\star}$
2. $\gamma \in (V \cup \Sigma)^{\star} - \epsilon$
3. $S \in V$ is the start variable
4. One additional rule allowed $S \rightarrow \epsilon$ and S is not on the right side of any rule

4.3 Linear Bounded Automaton

4.3.1 Informal Definition

Has no stack but can read/write anywhere on the input string

5 Turing Machines

5.1 Formal Def

A Turing Machine is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ with $Q\Sigma, \Gamma \equiv$ non empty sets

- Q is set of states
- Σ is the input alphabet, which does not contain the blank symbol _
- Γ is the tape alphabet, in which $\_ \in \Gamma$ and $\Sigma \subset \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function
- $q_0 \in Q$ is the start state
- $q_{accept} \in Q$ is the accept state
- $q_{reject} \in Q$ is the reject state
- $q_{accept} \neq q_{reject}$

5.2 Configurations

A configuration of the turing machine $\equiv c_i = (q_i, p_i, t_i)$, where $q_i \in Q, p_i$ is the head pos, and $t_i \in \Gamma^{\star}$ is the tape contents.

Notice that configurations are unique, and from them given the next input symbol one can determine the next configuration. i.e.

$$\delta(q_i, \gamma_i) : (c_i, \gamma_i) \mapsto (q_{i+1}, p_{i+1}, D_{i+1}) \mapsto c_{i+1} \tag{1}$$

for $D_{i+1} \in \{L, R\}$

5.2.1 Halting Configs

If either of the following type of configs is encountered, the turning machine halts and returns 'accept' or 'reject' respectively.

- Accept Config

$$c_{accept} \iff q_i = q_{accept} \tag{2}$$

- Reject Config

$$c_{reject} \iff q_i = q_{reject} \tag{3}$$

5.3 Turning Recognizable

M is <u>Recognizable</u> iff $\forall w \in L, M$ accepts

5.4 Turing Decidable

M, is <u>Decidable</u> iff $\forall w \in L, M$ accepts <u>and</u> $\forall w \notin L, M$ rejects

5.4.1 Co-Recognizablity

1. Define: $Sigma^{\star} - A = \bar{A}$ is recognizable

- Decidablility Theorem:

    1. A language is only Turing Decidable iff it is both recognizable and co-recognizable
    2. If a language is not decidable then its compliment is not recognizable

5.5 Turing Machine Variants:

Note that none of these add any power

- Multi-Tape: $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}^k$
    - Emulate on single tape by striping and recording/marking virtual head position
- Adding Stay: $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ where S doesn't move the head
    - Emulate by moving left and then right while not changing tape contents
- Non-Deterministic:

5.6 Universal Turing Machine

Turing machine that take a turing machine, M, as a string encoding denoted as $\langle M \rangle$

5.7 Undecidability

5.7.1 Halting Problem

1. Theorem: A_{TM} is not Turing Decidable
2. Proof by Contradiction:
   (a) Suppose A_{TM} were decidable.
   (b) Let H be a TM that decides A_{TM}
      i. $H(\langle M, w\rangle) =$
         A. accept if $w \in L(M)$
         B. reject if $w \notin L(M)$
   (c) Construct a TM, D, which uses H and give the opposite result
      i. $D \equiv$ on input $\langle M\rangle$, in which M is a T.M
         A. Run H on input $\langle M, \langle M\rangle\rangle$
         B. Return the opposite of what H outputs
   (d) Therefore $D(\langle M\rangle)$:
      i. accept if M rejects $\langle M\rangle$
      ii. reject if M accepts $\langle M\rangle$
   (e) Note that this implies $D(\langle D\rangle)$:
      i. accept if D rejects $\langle D\rangle$
      ii. reject if D accepts $\langle D\rangle$
   (f) Notice that this is a contradiction, thus H and D cannot exist

5.8 Reduction

Using a language that is known to be undecidable prove that another language is not undecidable.

5.8.1 Map Reductions

- Computable Functions

  A function $f : \Sigma^\star \to \Sigma^\star$ is called computable iff $\exists$ a TM, M, s.t. M halts $\forall w \in \Sigma^\star$, and after halting, $f(w)$ appears alone on the tape.

- Mapping reducible

  A language A is called mapping reducible to language B, written $A \leq_M B$ iff:

$$\exists(f : \Sigma^\star \to \Sigma^\star), \forall w \in \Sigma^\star : [w \in A \iff f(w) \in B] \tag{4}$$

  - Theorems

    1. Note the rule of thumb for these theorems is that $\leq_M$ more or less reflects the Chomsky Heirarchy:
    2. **Theorem**: $[A \leq_M B] \implies$ [If B is Turing Decidable, then A is turing Decidable]
    3. **Collary**: $[A \leq_M B] \implies$ [If A is Turing Undecidable, then B is Turing Undecidable]
    4. **Theorem**: $[A \leq_M B] \implies$ [If B is Turing Recognizable then A is Turing Recognizable]
    5. **Collary**: $[A \leq_M B] \implies$ [If A is Turing Unrecognizable then B is Turing Unrecognizable]

5.8.2 Rice's Theorem:

1. If P is a set of TM's with a property that satisfies:
   (a) $\forall$ TM's M and M_2 s.t. $L(M_1) = L(M_2)$, $\langle M_1 \rangle \in P$ iff $\langle M_2 \rangle \in P$
   (b) $\exists$ TM M_1 and M_2 for which $\langle M_1 \rangle \in P$ and $\langle M_2 \rangle \notin P$
2. Then the language of P is <u>not</u> Turing Decidable

6 Kolmogorov Complexity

- How 'Small'(state number) of a turing machine is needed to generate a given string

7 Complexity Theory:

7.1 Definition:

1. **Time Complexity**: How many steps does a Turing machine take to decide?
2. **Space Complexity**: How much space does a turing machine need on the tape to decide?
3. Let $f(n) \equiv$ the max number of steps for a TM to decide w, with $n = |w|$
   (a) $f(n) = O(g(n)) \iff \exists (c, n_0) \in \mathbb{R} \times \mathbb{Z}$ s.t. $\forall n \geq f(n), f(n) \leq cg(n)$
   (b) Which is equivelent to:

$$\forall w \in \Sigma^\star : |w| \geq n_0 \iff f(|w|) \leq cg(|w|) \tag{5}$$

7.2 Complexity Classes

1. Let $t : \mathbb{N} \to \mathbb{N}$ be a function
2. Let $D \equiv$ set of all decidable TMs
3. Let $TIME[t(n)] \equiv A \in D | \exists$ a TM that decides A in $O(t(n))$ steps

7.2.1 P vs NP

- P

$$P \equiv \cup_{k \in \mathbb{N}} TIME(n^k) \tag{6}$$

  1. Notice that $TIME(n^k) \subset TIME(n^{k+1})$
  2. Thus $P =$ all language for which $\exists$ a TM that decides in polynomial time

- NP

$$NP \equiv \cup_{k \in \mathbb{N}} NTIME(n^k) \tag{7}$$

  1. $NTIME(t(n)) \equiv \{A \in D | \exists$ a nondeterministic TM that decides A in $O(t(n))$ steps$\}$
  2. Note that this appears to be equivlent to verification via bruteforce
     (a) **Satisfyability Problem**: A language $A \in NP \iff A$ is polinomially <u>verifiable</u>.

        i. A is verifiable iff $\exists$ a poly-time DTM that takes input w for A and the certificate c and decides if $w \in A$

- Million Dollar Question

  1. Does $P = NP$?

- Hardest NP problems

  1. A Problem, P, is NP-Complete iff:
     (a) $P \in NP$
     (b) $\forall A \in NP, A \leq_P P$
        i. Where $\leq_P$ is a mapping reduction where f is computable in polynomial time