

## Are There Languages That Are Not Even Recognizable?

---

- ◆ Recall from last class:  
 $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$   
 $A_H = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on } w \}$
- ◆  $A_{TM}$  and  $A_H$  are undecidable but Turing-recognizable
  - ◇ Are there languages that are not even Turing-recognizable?
- ◆ What happens if a language  $A$  and its complement  $\bar{A}$  are both Turing-recognizable?

## Are There Languages That Are Not Even Recognizable?

---

- ◆  $A_{TM}$  and  $A_H$  are undecidable but Turing-recognizable
  - ◇ Are there languages that are not even Turing-recognizable?
- ◆ **What happens if both  $A$  and  $\bar{A}$  are Turing-recognizable?**
  - ◇ There exist TMs  $M_1$  and  $M_2$  that recognize  $A$  and  $\bar{A}$
  - ◇ **Can construct a decider for  $A$ !** On input  $w$ :
    1. Simulate  $M_1$  and  $M_2$  on  $w$  one step at a time, alternating between them.
    2. If  $M_1$  accepts, then ACC  $w$  and halt; if  $M_2$  accepts, REJ  $w$  and halt.
- ◆ **Thm:  $A$  and  $\bar{A}$  are both Turing-recognizable iff  $A$  is decidable**
- ◆ **Corollary:  $\bar{A}_{TM}$  and  $\bar{A}_H$  are not Turing-recognizable**
  - ◇ If they were, then  $A_{TM}$  and  $A_H$  would be decidable

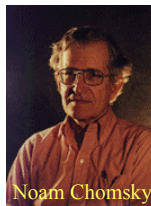
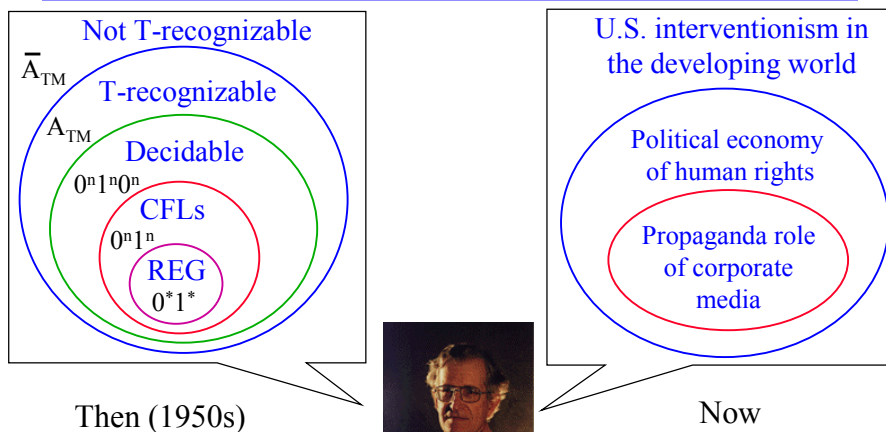
# The Chomsky Hierarchy of Languages

Increasing generality →

Language	Regular	Context-Free	Decidable	Turing-Recognizable
<b>Computational Models</b>	DFA, NFA, RegExp	PDA, CFG	Deciders – TMs that halt for all inputs	TMs that may loop for strings not in language
<b>Examples</b>	$(0 \cup 1)^* 11$	$\{0^n 1^n \mid n \geq 0\}$ , Palindromes	$\{0^n 1^n 0^n \mid n \geq 0\}$ , $A_{DFA}$ , $A_{CFG}$	$A_{TM}$ , $A_H$

(Chomsky also studied context-sensitive languages (CSLs, e.g.  $a^n b^n c^n$ ), a subset of decidable languages recognized by linear-bounded automata (LBA))

# The Chomsky Hierarchy – Then & Now...





Review Slides for the  
Final Exam will appear  
here



**This space for rent**



## Review of Chapters 0-1

---

- ◆ See Midterm Review Slides
  - ⇒ Emphasis on:
    - ◆ Sets, strings, and languages
    - ◆ Operations on strings/languages (concat, \*, union, etc)
    - ◆ Lexicographic ordering of strings
    - ◆ DFAs and NFAs: definitions and how they work
    - ◆ Regular languages and properties
    - ◆ Regular expressions and GNFA's (see lecture slides)
    - ◆ Pumping lemma for regular languages and showing nonregularity

## Context-Free Grammars (CFGs)

---

- ◆ CFG  $G = (V, \Sigma, R, S)$ 
  - ⇒ Variables, Terminals, Rules, Start variable
  - ⇒  $uAv$  yields  $uwv$  if  $A \rightarrow w$  is a rule in  $G$ : Written as  $uAv \Rightarrow uwv$
  - ⇒  $u \Rightarrow^* v$  if  $u$  yields  $v$  in 0, 1, or more steps
  - ⇒  $L(G) = \{w \mid S \Rightarrow^* w\}$
  - ⇒ CFGs for regular languages: Convert DFA to a CFG (Create variables for states and rules to simulate transitions)
- ◆ Ambiguity: Grammar  $G$  is ambiguous if  $G$  has two or more parse trees for some string  $w$  in  $L(G)$ 
  - ⇒ See lecture notes/text/homework for examples
- ◆ Closure properties of Context-Free languages
  - ⇒ Closed under  $\cup$ , concat, \* *but not*  $\cap$  or complementation.
  - ⇒ See homework and lecture slides

## Pushdown Automata (PDA)

---

- ◆ PDA  $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$ 
  - ⇒  $Q$  = set of states
  - ⇒  $\Sigma$  = input alphabet
  - ⇒  $\Gamma$  = stack alphabet
  - ⇒  $q_0$  = start state
  - ⇒  $F \subseteq Q$  = set of accept states
  - ⇒ Transition function  $\delta: Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \text{Pow}(Q \times \Gamma_\epsilon)$
  - ⇒ (current state, next input symbol, popped symbol) → {set of (next state, pushed symbol)}
  - ⇒ Input/popped/pushed symbol can be  $\epsilon$
- ◆ Example PDAs for:
  - ⇒  $\{w\#w^R \mid w \in \{0,1\}^*\}$ ,  $\{ww^R \mid w \in \{0,1\}^*\}$ , Palindromes

## Context-Free Languages: Main Results

---

- ◆ CFGs and PDAs are equivalent in computational power
  - ⇒ Generate/recognize the same class of languages (CFLs)
  - 1. If  $L = L(G)$  for some CFG  $G$ , then  $L = L(M)$  for some PDA  $M$ 
    - ◆ Know how to convert a given CFG to a PDA
  - 2. If  $L = L(M)$  for some PDA  $M$ , then  $L = L(G)$  for some CFG  $G$ 
    - ◆ Be familiar with the construction – no need to memorize the induction proof
- ◆ Pumping Lemma for CFLs
  - ⇒ Know the exact statement:  $L \text{ CFL} \Rightarrow \exists p \text{ s.t. } \forall s \text{ in } L \text{ s.t. } |s| \geq p,$   
 $\exists u, v, x, y, \text{ and } z \text{ s.t. } s = uvxyz \text{ and:}$ 
    1.  $uv^i xy^j z \in L \forall i \geq 0$ ,
    2.  $|vy| \geq 1$ , and
    3.  $|vxy| \leq p$ .
- ◆ Using the PL to show languages are not CFLs
  - ⇒ E.g.  $\{0^n 1^n 0^n \mid n \geq 0\}$  and  $\{0^n \mid n \text{ is a prime number}\}$

## Turing Machines: Definition and Operation

---

- ◆  $TM M = (Q, \Sigma, \Gamma, \delta, q_0, q_{ACC}, q_{REJ})$ 
  - ⇒  $Q$  = set of states
  - ⇒  $\Sigma$  = input alphabet not containing blank symbol “\_”
  - ⇒  $\Gamma$  = tape alphabet containing blank “\_”, all symbols in  $\Sigma$ , plus possible temporary variables such as  $X, Y$ , etc.
  - ⇒  $q_0$  = start state
  - ⇒  $q_{ACC}$  = accept and halt state
  - ⇒  $q_{REJ}$  = reject and halt state
  - ⇒ Transition function  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- ◆  $\delta(\text{current state, symbol under the head}) = (\text{next state, symbol to write over current symbol, direction of head movement})$ 
  - ⇒ *Configurations* of a TM, definition of language  $L(M)$  of a TM  $M$

## Decidable versus Recognizable Languages

---

- ◆ A language is Turing-recognizable if there is a Turing machine  $M$  such that  $L(M) = L$ 
  - ⇒ For all strings in  $L$ ,  $M$  halts in state  $q_{ACC}$
  - ⇒ For strings not in  $L$ ,  $M$  may either halt in  $q_{REJ}$  or loop forever
- ◆ A language is decidable if there is a “decider” Turing machine  $M$  that halts on all inputs such that  $L(M) = L$ 
  - ⇒ For all strings in  $L$ ,  $M$  halts in state  $q_{ACC}$
  - ⇒ For all strings not in  $L$ ,  $M$  halts in state  $q_{REJ}$
- ◆ Showing a language is decidable by construction:
  - ⇒ *Implementation level description of deciders*
  - ⇒ E.g.  $\{0^n 1^n 0^n \mid n \geq 0\}$ ,  $\{0^n \mid n = m^2 \text{ for some integer } m\}$ , see text

## Equivalence of TM Types & Church-Turing Thesis

---

- ◆ Varieties of TMs: Know the definition, operation, and idea behind proof of equivalence with standard TM
  - ⇨ Multi-Tape TMs: TM with  $k$  tapes and  $k$  heads
  - ⇨ Nondeterministic TMs (NTMs)
    - ◆ Decider if all branches halt on all inputs
  - ⇨ Enumerator TM for  $L$ : Prints all strings in  $L$  (in any order, possibly with repetitions) and only the strings in  $L$
- ◆ Can use any of these variants for showing a language is Turing-recognizable or decidable
- ◆ Church-Turing Thesis: Any formal definition of “algorithms” or “programs” is equivalent to Turing machines

## Decidable Problems

---

- ◆ Any problem can be cast as a language membership problem
  - ⇨ Does DFA  $D$  accept input  $w$ ? Equivalent to:  
Is  $\langle D, w \rangle$  in  $A_{\text{DFA}} = \{\langle D, w \rangle \mid D \text{ is a DFA that accepts input } w\}$ ?
- ◆ Decidable problems concerning languages and machines:  
(see textbook for proofs to some of these)
  - ⇨  $A_{\text{DFA}} = \{\langle D, w \rangle \mid D \text{ is a DFA that accepts input } w\}$
  - ⇨  $A_{\text{NFA}} = \{\langle N, w \rangle \mid N \text{ is a NFA that accepts input } w\}$
  - ⇨  $A_{\text{REG}} = \{\langle R, w \rangle \mid R \text{ is a reg. exp. that generates string } w\}$
  - ⇨  $A_{\text{empty-DFA}} = \{\langle D \rangle \mid D \text{ is a DFA and } L(D) = \emptyset\}$
  - ⇨  $A_{\text{Equal-DFA}} = \{\langle C, D \rangle \mid C \text{ and } D \text{ are DFAs and } L(C) = L(D)\}$
  - ⇨  $A_{\text{CFG}} = \{\langle G, w \rangle \mid G \text{ is a CFG that generates string } w\}$
  - ⇨  $A_{\text{empty-CFG}} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$

## Undecidability, Reducibility, Unrecognizability

---

- ◆  $A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$  is Turing-recognizable but not decidable (Proof by diagonalization)
- ◆ To show a new problem  $A$  is undecidable, **reduce**  $A_{TM}$  to  $A$ 
  - ⇒ Show that if  $A$  was decidable, then you can use the decider for  $A$  as a *subroutine* to decide  $A_{TM}$  (leading to a contradiction)
  - ⇒ E.g. Halting problem = “Does a program halt for an input or go into an infinite loop?”
  - ⇒ Can show that the Halting problem is undecidable by reducing  $A_{TM}$  to  $A_H = \{ \langle M, w \rangle \mid \text{TM } M \text{ halts on input } w \}$
- ◆  $A$  is decidable iff  $A$  and  $\bar{A}$  are both Turing-recognizable
  - ⇒ Corollary:  $\bar{A}_{TM}$  and  $\bar{A}_H$  are not Turing-recognizable

## Final Exam

---

- ◆ Details regarding the Final Exam
  - ⇒ When: Monday, June 7, 2004 from 8:30-10:20 a.m.
  - ⇒ Where: Same classroom (MGH 231)
  - ⇒ What will it cover?
    - ◆ Chapters 0-4 and Chapter 5: pages 171-176.
    - ◆ Emphasis will be on material covered after midterm (Chapter 2 and beyond)
    - ◆ You may bring 2 pages of notes (8 ½” x 11” sheets!)
    - ◆ Approximately 6 questions
  - ⇒ How do I ace it?
    - ◆ Practice, practice, practice!
    - ◆ See class website for sample final exam and solutions



I believe the Final exam is decidable!

Stay cool 'n' keep pumpin'!



I believe the world's problems are politically decidable.



I believe my next movie will be unrecognizable.



**NOAM**