Recap of Undecidability Proof

- The Question: Are there languages that are not decidable by any Turing machine (TM)?
  - I.e. Are there problems that cannot be solved by any algorithm?
- Consider the language:
  A_{TM} = {<M,w> | M is a TM and M accepts w}
  (Recall that <A,B,…> is just a string encoding the objects A, B, …)
- What can we say about A_{TM}?
  - A_{TM} is Turing-recognizable: Recognizer TM R for A_{TM}:
    On input string <M,w>: Simulate M on w.
    ACCEPT <M,w> if M halts & accepts w;
    REJECT <M,w> if M halts & rejects
    (Loop (& thus reject <M,w>) if M ends up looping).

  R accepts <M,w> iff M accepts w $\Rightarrow$ L(R) = A_{TM}

Is A_{TM} also decidable?

- No, A_{TM} = {<M,w> | M is a TM and M accepts w} is undecidable! 1-slide Proof (by Contradiction):
  1. Assume A_{TM} is decidable $\Rightarrow$ there's a decider H, L(H) = A_{TM}
  2. H on <M,w> = ACC if M accepts w
     REJ if M rejects w (halts in q_{REJ} or loops on w)
  3. Construct new TM D: On input <M>,
     Simulate H on <M,<M>> (here, w = <M>)
     If H accepts, then REJ input <M>
     If H rejects, then ACC input <M>
  4. What happens when D gets <D> as input?
     D rejects <D> if H accepts <D,<D>> if D accepts <D>
     D accepts <D> if H rejects <D,<D>> if D rejects <D>

  Contradiction! D cannot exist $\Rightarrow$ H cannot exist
  Therefore, A_{TM} is not a decidable language.

Undecidability Proof uses Diagonalization

Input string

List of TMs

	<M_1>	<M_2>	<M_3>	…
M_1	ACC	REJ	*loop*	…
M_2	REJ	*loop*	ACC	…
M_3	ACC	ACC	REJ	…
:	:	:	:	

If H exists →

D outputs opposite of diagonal

	<M_1>	<M_2>	<M_3>	…	<D>
M_1	ACC	REJ	**REJ**	…	ACC
M_2	REJ	**REJ**	ACC	…	ACC
M_3	ACC	ACC	REJ	…	REJ
:	:	:	:	:	:
D	REJ	ACC	ACC	…	??

D on <M_i> accepts if and only if M_i on <M_i> rejects.
So, D on <D> will accept if and only if D on <D> rejects!
A contradiction $\Rightarrow$ H cannot exist!

One Last Concept: Reducibility

- How do we show a new problem A is undecidable?
  - Use diagonalization again? Yes, but too tedious.
- Easy Proof: Show that A_{TM} is reducible to the new problem A
  - What does this mean and how do we show this?
- Show that if A was decidable, then you can use the decider for A as a *subroutine* to decide A_{TM}
  - A contradiction, therefore A must also be undecidable

The Halting Problem is Undecidable (Turing, 1936)

- Halting Problem: Does TM M halt on input w?
  - Equivalent language: A_H = { <M,w> | TM M halts on input w}
  - Need to show A_H is undecidable
  - We know A_{TM} = {<M,w> | TM M accepts w} is undecidable
- Show A_{TM} is reducible to A_H (Theorem 5.1 in text)
  - Suppose A_H is decidable $\Rightarrow$ there's a decider M_H for A_H
  - Then, we can construct a decider D_{TM} for A_{TM}:

    On input <M,w>, run M_H on <M,w>.
    - If M_H rejects, then REJ (this takes care of M looping on w)
    - If M_H accepts, then simulate M on w until M halts
    - If M accepts, then ACC input <M,w>; else REJ

    $L(D_{TM}) = A_{TM} \Rightarrow A_{TM}$ is decidable! Contradiction $\Rightarrow A_H$ is undecidable

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 $\overline{A}$ are Turing-recognizable?
  - There exist TMs M1 and M2 that recognize A and $\overline{A}$
  - Can construct a decider for A! On input w:

  1. Simulate M1 and M2 on w one step at a time, alternating between them.
  2. If M1 accepts, then ACC w and halt; if M2 accepts, REJ w and halt.
- A and $\overline{A}$ are both Turing-recognizable iff A is decidable
- Corollary: $\overline{A}_{TM}$ and $\overline{A}_H$ are not Turing-recognizable
  - If they were, then A_{TM} and A_H would be decidable

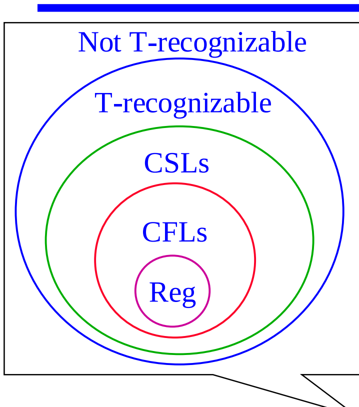
The Chomsky Hierarchy of Languages

Increasing generality →

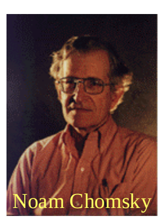
Language	Regular	Context-Free	Decidable	Turing-Recognizable
Computational Models	DFA, NFA, RegExp	PDA, CFG	Deciders – TMs that halt for all inputs	TMs that may loop for strings not in language
Examples	$(0\cup1)^*11$	$\{0^n1^n \mid n \geq 0\}$, Palindromes	$\{0^n1^n0^n \mid n \geq 0\}$, A_{DFA}, A_{CFG}	A_{TM}, A_H

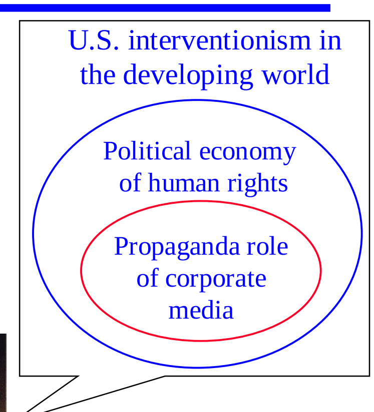
(Chomsky also studied context-sensitive languages (CSLs, e.g. $a^nb^n c^n$) , a subset of decidable languages recognized by linear-bounded automata (LBA))

The Chomsky Hierarchy – Then & Now…



Then (1950s)





Now

Final Review

- Details regarding the Final Exam
  - When: This Friday, Dec. 14, 2001 from 8:30-10:20 a.m.
  - Where: This classroom MGH 231.
  - What will it cover?
    - Chapters 0-4 and Theorem 5.1 (example of reducibility)
    - Emphasis will be on material covered after midterm (Chapter 2 and beyond)
    - You may bring 1 page of notes (8 ½" x 11" sheet!)
    - Approximately 6 questions
  - How do I ace it?
    - Practice, practice, practice!
    - See class website for practice problems

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 GNFAs (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 $A \to w$ 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
  - Σ = input alphabet
  - Γ = stack alphabet
  - q_0 = start state
  - $F \subseteq Q$ = set of accept states
  - Transition function $\delta: Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to Pow(Q \times \Gamma_\varepsilon)$
  - (current state, next input symbol, popped symbol) $\to$ {set of (next state, pushed symbol)}
  - Input/popped/pushed symbol can be ε
- 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 CFL $\Rightarrow \exists$p s.t. $\forall s$ in L s.t. $|s| \geq$ p, $\exists$ u, v, x, y, and z s.t. $s = uvxyz$ and:
    1. $uv^ixy^iz \in L \; \forall \; i \geq 0$, **2.** $|vy| \geq 1$, and **3.** $|vxy| \leq$ p.
- Using the PL to show languages are not CFLs
  - E.g. $\{0^n1^n0^n \mid n \geq 0\}$ and $\{0^n \mid$ n 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
  - Σ = input alphabet not containing blank symbol "_"
  - Γ = tape alphabet containing blank "_", all symbols in Σ, 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 $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- δ(current state, symbol under the head) = (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. $\{0^n1^n0^n \mid n \geq 0\}$, $\{0^n \mid n = m^2$ 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
- <u>Church-Turing Thesis</u>: Any formal definition of “algorithms” or “programs” is equivalent to Turing machines

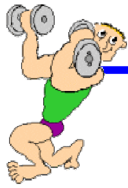
Decidable Problems

- Any problem can be cast as a language membership problem
  - Does DFA D accept input w? Equivalent to:
    Is <D,w> in A_{DFA} = {<D,w> | D is a DFA that accepts input w}?
- Decidable problems concerning languages and machines:
  - A_{DFA}
  - A_{NFA} = {<N,w> | N is a NFA that accepts input w}
  - A_{REX} = {<R,w> | R is a reg. exp. that generates string w}
  - $A_{empty\text{-}DFA}$ = {<D> | D is a DFA and L(D) = ∅}
  - $A_{Equal\text{-}DFA}$ = {<C,D> | C and D are DFAs and L(C) = L(D)}
  - A_{CFG} = {<G,w> | G is a CFG that generates string w}
  - $A_{empty\text{-}CFG}$ = {<G> | G is a CFG and L(G) = ∅}

Undecidability, Reducibility, Unrecognizability

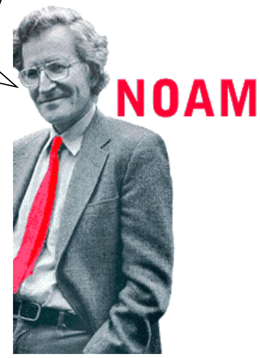
- A_{TM} = {<M,w> | M is a TM and M accepts w} is Turing-recognizable but not decidable (Proof by diagonalization)
- To show a problem A is undecidable, reduce A_{TM} to A
  - Show that if A was decidable, then you can use the decider for A as a *subroutine* to decide A_{TM}
  - E.g. Halting problem = "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 = { <M,w> | TM M halts on input w}
- A is decidable iff A and $\overline{A}$ are both Turing-recognizable
  - Corollary: $\overline{A_{TM}}$ and $\overline{A_H}$ are not Turing-recognizable

Good luck & have a great break!



I believe the Final exam is decidable!

I believe the world's problems are politically decidable.



I believe my next movie will be unrecognizable.