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:
  \( A_{TM} = \{ <M,w> | M \text{ is a TM and } M \text{ accepts } w \} \)
  (Recall that \(<A,B,...>\) is just a string encoding the objects \(A, B, \ldots\))

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

Is \( A_{TM} \) also decidable?

- No, \( A_{TM} = \{ <M,w> | M \text{ is a TM and } M \text{ accepts } w \} \) 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

<table>
<thead>
<tr>
<th>Input string</th>
<th>M₁</th>
<th>M₂</th>
<th>M₃</th>
<th>M₁</th>
<th>M₂</th>
<th>M₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁</td>
<td>ACC</td>
<td>REJ</td>
<td>loop</td>
<td>ACC</td>
<td>REJ</td>
<td>ACC</td>
</tr>
<tr>
<td>M₂</td>
<td>REJ</td>
<td>loop</td>
<td>ACC</td>
<td>REJ</td>
<td>ACC</td>
<td>ACC</td>
</tr>
<tr>
<td>M₃</td>
<td>ACC</td>
<td>ACC</td>
<td>REJ</td>
<td>ACC</td>
<td>REJ</td>
<td>...</td>
</tr>
<tr>
<td>D</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

If H exists

D outputs opposite of diagonal

D on <Mᵢ> accepts if and only if Mᵢ on <Mᵢ> rejects.
So, D on <D> will accept if and only if D on <D> rejects!
A contradiction ⇒ 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: \( A_H = \{ \langle M, w \rangle \mid \text{TM M halts on input } w \} \)
  ➤ Need to show \( A_H \) is undecidable
  ➤ We know \( A_{TM} = \{ \langle M, w \rangle \mid \text{TM M accepts } w \} \) 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 \( \langle M, w \rangle \), run \( M_H \) on \( \langle M, w \rangle \).
    • 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 \( \langle M, w \rangle \); else REJ
  \[ L(D_{TM}) = A_{TM} \Rightarrow A_{TM} \text{ is decidable! Contradiction } \Rightarrow A_H \text{ is undecidable} \]

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 \( M_1 \) and \( M_2 \) that recognize \( A \) and \( \overline{A} \)
  ➤ 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.

✦ \( A \) 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

<table>
<thead>
<tr>
<th>Language</th>
<th>Regular</th>
<th>Context-Free</th>
<th>Decidable</th>
<th>Turing-Recognizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational</td>
<td>DFA,</td>
<td>PDA,</td>
<td>Deciders –</td>
<td>TMs that may loop</td>
</tr>
<tr>
<td>Models</td>
<td>NFA,</td>
<td>CFG</td>
<td>TMs that</td>
<td>for strings not in</td>
</tr>
<tr>
<td></td>
<td>RegExp</td>
<td></td>
<td>halt for</td>
<td>language</td>
</tr>
<tr>
<td>Examples</td>
<td>$(0\cup 1)^*1$</td>
<td>${0^n1^n \mid n \geq 0}$,</td>
<td>Palindromes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>${0^n1^n 0^n \mid n \geq 0}$,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_{\text{DFA}}$,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_{\text{CFG}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_{\text{TM}}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$A_{\text{H}}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(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…

- **U.S. interventionism in the developing world**
  - Political economy of human rights
  - Propaganda role of corporate media

- **Then (1950s)**
  - Noam Chomsky

- **Now**

R. Rao, CSE 322
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 $uvw$ if $A \rightarrow w$ is a rule in $G$: Written as $uAv \Rightarrow uvw$
  - $u \Rightarrow^* v$ 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 $\delta: Q \times \Sigma \times \Gamma \rightarrow \text{Pow}(Q \times \Gamma)$
    - (current state, next input symbol, popped symbol) → {set of (next state, pushed symbol)}
  - Input/popped/pushed symbol can be $\varepsilon$

- 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. \( uv^ixy^iz \in L \forall i \geq 0, \ 2. |vy| \geq 1, \text{ and } \ 3. |vxy| \leq p. \)

✦ Using the PL to show languages are not CFLs
   ➤ E.g. \( \{0^n1^n0^n \mid n \geq 0\} \) and \( \{0^n \mid n \text{ is a prime number}\} \)

Turing Machines: Definition and Operation

✦ TM \( M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{ACC}}, q_{\text{REJ}}) \)
   ➤ \( Q \) = 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 \( X, Y, \text{ etc.} \)
   ➤ \( q_0 \) = start state
   ➤ \( q_{\text{ACC}} \) = accept and halt state
   ➤ \( q_{\text{REJ}} \) = reject and halt state
   ➤ Transition function \( \delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\} \)

✦ \( \delta(\text{current state, symbol under the head}) \) = (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 \( q_{\text{ACC}} \)
  ➔ For strings not in L, M may either halt in \( q_{\text{REJ}} \) 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 \( q_{\text{ACC}} \)
  ➔ For all strings not in L, M halts in state \( q_{\text{REJ}} \)

✦ 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 \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 <D,w> in ADFA = {<D,w> | D is a DFA that accepts input w}?

✦ Decidable problems concerning languages and machines:
  ➲ ADFA
  ➲ ANFA = {<N,w> | N is a NFA that accepts input w}
  ➲ AREX = {<R,w> | R is a reg. exp. that generates string w}
  ➲ Aempty-DFA = {<D> | D is a DFA and L(D) = ∅}
  ➲ Aequal-DFA = {<C,D> | C and D are DFAs and L(C) = L(D)}
  ➲ ACFG = {<G,w> | G is a CFG that generates string w}
  ➲ Aempty-CFG = {<G> | G is a CFG and L(G) = ∅}

Undecidability, Reducibility, Unrecognizability

✦ ATM = {<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 ATM to A
  ➲ Show that if A was decidable, then you can use the decider for A as a subroutine to decide ATM
  ➲ 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 ATM to AH = {<M,w> | TM M halts on input w}

✦ A is decidable iff A and A are both Turing-recognizable
  ➲ Corollary: A and A are not Turing-recognizable

R. Rao, CSE 322
I believe the Final exam is decidable!

I believe the world’s problems are politically decidable.

I believe my next movie will be unrecognizable.

Good luck & have a great break!