Agenda

• **Last week**
  - Decidability of languages involving DFA’s & CFG’s
  - Non-decidability of the Turing machine acceptance problem $A_{TM}$

• **Today**
  - Revisit the $A_{TM}$ proof
  - Finish Chapter 4
  - Problems

• **Tomorrow**
  - Begin Chapter 5 (pp. 171 - 176)
Announcements

• Homework due next Tuesday (11/2)
  - 4.4, 4.8, 5.1, 5.2, 5.13

• No quiz this week
  - Quiz next Tuesday

• No tutorials this week -- extra office hours instead
  - Tuesday & Wednesday 3:00 – 5:00
  - Or by appointment
Undecidability of $A_{TM}$

**Theorem:** $A_{TM}$ is undecidable

**Proof:** (By contradiction) Assume $A_{TM}$ is decidable and let $H$ be a decider for $A_{TM}$

\[
H(<M,w>) = \begin{cases} 
\text{accept} & \text{if } M \text{ accepts } w \\
\text{reject} & \text{if } M \text{ does not accept } w 
\end{cases}
\]
Proof (cont.)

• Consider the TM $D$ that submits the string $<M>$ as input to the TM $M$

$D = \text{"On input } <M>, \text{ where } M \text{ is a TM:} $

1. Run $H$ on input $<M,<M>>$
2. If $H$ accepts $<M,<M>>$, reject
3. If $H$ rejects $<M,<M>>$, accept"

   ➢ Since $H$ is a decider, it must accept or reject
   ➢ Therefore, $D$ is a decider as well
   ➢ $D$ is a diagonalizing TM
Proof (cont.)

• What happens if D’s input is $<D>$?

$D(<D>) = \begin{cases} 
\text{reject} & \text{if } D \text{ accepts } <D> \\
\text{accept} & \text{if } D \text{ does not accept } <D>
\end{cases}$

• D cannot exist!
• Therefore, $H$ cannot exist – i.e., $A_{TM}$ is undecidable
Recap

• **Assume H decides $A_{TM}$**
  - $H(<M, w>) = \text{accept}$ if TM $M$ accepts $w$, reject otherwise

• **Define D using H**
  - $D(<M>)$ returns opposite of $H(<M, <M>>)$

• **Consider D(<D>)**
  - D accepts <D> if and only if D rejects <D>
Apply this method to $A_{DFA}$

- Assume $H$ decides $A_{DFA}$
  - $H(<A, w>) = \text{accept}$ if DFA $A$ accepts $w$, \text{reject} otherwise

- Define $D$ using $H$
  - $D(<A>)$ returns opposite of $H(<A, <A>>) $

- Consider $D(<D>)$
  - $D$ only accepts DFA’s as input ... $D$ is a TM
  - Method does not apply
Apply this method to $A_{TM-D}$

• Assume $H$ decides $A_{TM-D}$
  - $H(<M,w>) = \text{accept}$ if decider TM $M$ accepts $w$, reject otherwise

• Problem
  - What if $M$ is not a decider and just a regular TM?
  - How can we test our input?

• $A_{TM-D}$ is not decidable
Co-Turing recognizable

Definition: A language $A$ is co-Turing-recognizable if $\overline{A}$ is Turing-recognizable.

Example:

$E_{\text{DFA}} = \{<A> \mid A \text{ is a DFA and } L(A) = \emptyset\}$

$E_{\text{DFA}} = \{S \mid S \text{ does not describe a DFA or } S \text{ describes a DFA with a non-empty language}\}$

- How would we decide this language?
Decidability and recognizability

Theorem: The language $A$ is decidable if and only if it is both Turing-recognizable and co-Turing-recognizable.

Proof: In two parts

- If $A$ is decidable then $A$ and $\bar{A}$ are Turing-recognizable.
  - Follows from definitions

- If $A$ and $\bar{A}$ are Turing-recognizable then $A$ is decidable.
Decider for A

• Assume $M_1$ recognizes $A$ and $M_2$ recognizes $\overline{A}$

• Consider the following TM

$M = \text{“On input } w:\text{”}$

1. Run $M_1$ and $M_2$ on input $w$ in parallel
2. If $M_1$ accepts, accept; if $M_2$ accepts, reject"

• $M$ must halt on $w$ because at least one of the recognizers must halt on $w$

- If $w \in A$, $M_1$ must halt; otherwise, $M_2$ must halt
Corollary

• $A_{TM}$ is not Turing-recognizable

Proof: $A_{TM}$ is Turing-recognizable, but not decidable. By previous theorem, $A_{TM}$ cannot be co-Turing-recognizable.
Proving decidability of $L$

• Create a TM that decides $L$
• Your TM may invoke another TM that decides a language you know is decidable
### Languages we know are decidable

<table>
<thead>
<tr>
<th>Language</th>
<th>Input</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$&lt;D,w&gt;$, $D$ is a DFA, $w$ is a string</td>
</tr>
<tr>
<td>$A_{NFA}$</td>
<td>$&lt;N,w&gt;$, $N$ is an NFA, $w$ is a string</td>
</tr>
<tr>
<td>$A_{REX}$</td>
<td>$&lt;R,w&gt;$, $R$ is an RE, $w$ is a string</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$&lt;D&gt;$, $D$ is a DFA</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>$&lt;C,D&gt;$, $C$ and $D$ are both DFA’s</td>
</tr>
<tr>
<td>$A_{CFG}$</td>
<td>$&lt;G,w&gt;$, $G$ is a CFG, $w$ is a string</td>
</tr>
<tr>
<td>$E_{CFG}$</td>
<td>$&lt;G&gt;$, $G$ is a CFG</td>
</tr>
<tr>
<td>$L(G)$</td>
<td>$G$ is a CFG</td>
</tr>
</tbody>
</table>
Some decidable languages

- $F_{DFA} = \{ \langle A \rangle \mid A$ is a DFA and $L(A)$ is finite $\}$
- PRIME = $\{ n \mid n$ is a prime number $\}$
- CONN = $\{ \langle G \rangle \mid G$ is a connected graph $\}$
- $L_{10DFA} = \{ D \mid D$ is a DFA that accepts every string $w$ with $|w| = 10 \}$
- INT$_{CFG} = \{ \langle G_1, G_2, w \rangle \mid G_1$ and $G_2$ are CFG's and $w$ is accepted by both $\}$
- INTL$_{CFG} = L(G_1 \cap G_2)$, where $G_1$ and $G_2$ are CFG's
Group project 1

\[ F_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) \text{ is finite} \} \]
Group project 2

\[ \text{PRIME} = \{ n \mid n \text{ is a prime number} \} \]
Group project 3

$\text{CONN} = \{<G> \mid G \text{ is a connected graph}\}$
Group project 4

$L_{10}^{DFA} = \{D \mid D \text{ is a DFA that accepts every string } w \text{ with } |w| = 10\}$
INT_{CFG} = \{<G_1, G_2, w> | G_1 and G_2 are CFG's and w is accepted by both\}
Group project 6

\[ \text{INTL}_{\text{CFG}} = L(G_1 \cap G_2), \text{ where } G_1 \text{ and } G_2 \text{ are CFG's} \]