Last time
• Reductions
• Mapping reductions

Today
• Mapping reductions
• Computation history method
Proving undecidability and unrecognizability

Mapping Reductions
Given languages $A$ and $B$, 

$A \leq_m B$

if there is a computable function $f$, 
such that for all strings $w$, 

$w \in A$ iff $f(w) \in B$. 

Diagram: 

```
A ----> f ----> B
```
Theorem. If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

Proof: Let $M$ be a decider for $B$ and $f$ be a mapping reduction from $A$ to $B$. Construct a decider for $A$:
```
``On input $w$:
1. Compute $f(w)$.
2. Run $M$ on $f(w)$.
3. If it accepts, accept. O.w. reject."
```
Using mapping reductions to prove *undecidability*

**Theorem.** If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

**Corollary.** If $A \leq_m B$ and $A$ is *undecidable*, then $B$ is *undecidable*.

**Example:** If $A_{TM} \leq_m B$, then $B$ is *undecidable*. 
Theorem. If $A \leq_m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.

Proof: Let $M$ be a TM that recognizes $B$ and $f$ be a mapping reduction from $A$ to $B$. Construct a TM that recognizes $A$:
``On input $w$:
1. Compute $f(w)$.
2. Run $M$ on $f(w)$.
3. If it accepts, accept. O.w. reject.""
Using mapping reductions to prove unrecognizability

**Theorem.** If $A \leq_m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.

**Corollary.** If $A \leq_m B$ and $A$ is unrecognizable, then $B$ is unrecognizable.

**Example:** If $A_{TM} \leq_m B$, then $B$ is unrecognizable.
If $\bar{A} \leq_m \bar{B}$, we can conclude that

A. $A \leq_m B$
B. $B \leq_m A$
C. $\bar{A} \leq_m \bar{B}$
D. $\bar{B} \leq_m A$
E. None of the above.
Old proof that $\text{EQ}_{TM}$ is undecidable

$\text{EQ}_{TM} = \{(M_1, M_2) \mid M_1, M_2 \text{ are TMs}, L(M_1) = L(M_2)\}$

Proof: Suppose to the contrary that $\text{EQ}_{TM}$ is decidable, and let $R$ be a TM that decides it. We construct TM $S$ that decides $A_{TM}$.

$S = \text{``On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}"

   - $M' = \text{``On input } x, \text{ ignoring the input.}"
   - $M'' = \text{``Accept.}"

2. Run TM $R$ on input $\langle M', M'' \rangle$.

3. If it accepts, accept. O.w. reject."

$E_{TM}$ is undecidable.
Proof: The following TM computes the reduction:

\[ F = \text{``On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:} \]

1. Construct TMs \( M', M'' \).
   
   \( M' = \text{``On input } x, \)
   
   1. Ignore the input.
   2. Run TM \( M \) on input \( w \).
   3. If it accepts, accept.”

2. Output \( <M', M''> \).”
Conclusions from $A_{TM} \leq_m EQ_{TM}$

1. Since $A_{TM}$ is undecidable, so is $EQ_{TM}$

2. $A_{TM} \leq m EQ_{TM}$
   Since $A_{TM}$ is unrecognizable, so is $EQ_{TM}$
Prove that $\text{EQ}_{\text{TM}}$ is unrecognizable

Proof: We give a mapping reduction $\overline{\text{A}_{\text{TM}}} \leq_m \text{EQ}_{\text{TM}}$

The following TM computes the reduction:

$F = \text{``On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}'}$


   $M' = \text{``On input } x,$
   
   1. Ignore the input.
   2. Run TM $M$ on input $w$.
   3. If it accepts, accept.”

   $M'' = \text{``Reject.”}$

2. Output $\langle M', M'' \rangle$.”
# Problems in language theory

<table>
<thead>
<tr>
<th>$A_{DFA}$</th>
<th>$A_{CFG}$</th>
<th>$A_{TM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>decidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E_{DFA}$</th>
<th>$E_{CFG}$</th>
<th>$E_{TM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>decidable</td>
<td>undecidable</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$EQ_{DFA}$</th>
<th>$EQ_{CFG}$</th>
<th>$EQ_{TM}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>?</td>
<td>undecidable</td>
</tr>
</tbody>
</table>
Proving undecidability for languages that do not involve TM descriptions

Computation history method
A linear bounded automaton (LBA) is a TM variant that has bounded tape, with the number of tape squares equal to the size of the input.

![Diagram of finite state control and tape]

```
1 1 0 1 0 0
```
A configuration of an LBA is a setting of state, head position and tape contents.

Configurations

$q_7$

110$q_7$1000110
How many distinct configurations does an LBA have if it has $q$ states, $g$ symbols in tape alphabet, and the tape of length $n$?

A. $qgn$
B. $q + g + n$
C. $qg^n$
D. $qng^n$
E. None of the above.
Prove that $A_{\text{LBA}}$ is decidable

$$A_{\text{LBA}} = \{ \langle B, w \rangle \mid B \text{ is an LBA that accepts string } w \}$$

**Idea:** Given $\langle B, w \rangle$, simulate $B$ on $w$.
If it halts, we know the answer.
If it loops, we can detect because $B$ repeats a configuration.

S = ``On input $\langle B, w \rangle$, where $B$ is an LBA and $w$ is a string:

1. Simulate $B$ on $w$ for $qng^n$ steps.
2. If it accepts, accept.
3. If it rejects or does not halt, reject.”"
An accepting computation history for a TM M on input w is a sequence of configurations entered by M on input w:

\[ C_0 \# C_1 \# \ldots \# C_\ell \]

starting configuration

accepting configuration
Starting configuration

\[ C_0 = \langle q_0 \rangle w \]
Accepting configuration

\[ C_\ell = \ldots \langle q_{\text{acc}} \rangle \ldots \]
Each $C_{i+1}$ legally follows from $C_i$

\[ C_i = 1000 \langle q_7 \rangle 0110 \]

\[ C_{i+1} = 1000 2 \langle q_5 \rangle 110 \]
LBAs can check computation histories of TMs

Given a TM M and a string w, we can construct an LBA that checks whether its input is the accepting computation history of M on w.
Prove that $E_{\text{LBA}}$ is undecidable

$E_{\text{LBA}} = \{ \langle B \rangle \mid B \text{ is a LBA and } L(B) = \emptyset \}$

Proof: Suppose to the contrary that TM $R$ decides $E_{\text{LBA}}$. We construct TM $S$ that decides $A_{\text{TM}}$.

$S = \text{``On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}$$

1. Construct an LBA $B$ from $M$ and $w$:

   $B = \text{``On input } x, \text{ Accept if } x = C_0 \# \ldots \# C_\ell \text{ is the accepting computation history of } M \text{ on } w:$

   1. $C_0$ is the starting configuration of $M$ on $w$
   2. Each $C_{i+1}$ legally follows from $C_i$
   3. $C_\ell$ is an accepting configuration for $M$ ”

2. Run TM $R$ on input $\langle B \rangle$.

3. If it rejects, accept. O.w. reject.”