Today's learning goals

• Define reductions from one problem to another.
• Use reductions to prove undecidability.
• Describe the difference between diagonalization and reduction.
So far

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Recognizable (and not decidable)</th>
<th>Not recognizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
<td>??</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$\text{HALT}_{TM}$</td>
<td></td>
</tr>
<tr>
<td>$\text{EQ}_{DFA}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{ALL}_{DFA}$ (HW)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Give algorithm!

Diagonalization OR reduction
Reduction?

A problem $P_1$ **reduces to** a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

In other words: using a solution for $P_2$ as a subroutine gives a solution for $P_1$.

In our example: we used a solution for $\text{HALT}_\text{TM}$ to get a solution for $A_{\text{TM}}$. This means that $A_{\text{TM}}$ **reduces to** $\text{HALT}_\text{TM}$. 
Reduction?

A problem $P_1$ reduces to a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

If $P_1$ reduces to $P_2$ and

A. $P_1$ is decidable, then $P_2$ is also decidable.
B. $P_2$ is decidable, then $P_1$ is also decidable.
C. Both of the above.
D. None of the above.
E. I don't know.
Reduction?

A problem $P_1$ **reduces to** a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

If $P_1$ reduces to $P_2$ and

A. $P_1$ is undecidable, then $P_2$ is also undecidable.
B. $P_2$ is undecidable, then $P_1$ is also undecidable.
C. Both of the above.
D. None of the above.
E. I don't know.
Reduction?

A problem $P_1$ reduces to a problem $P_2$ if any solution for $P_2$ can be used to solve $P_1$.

New strategy: to prove that a problem is undecidable, prove that a problem we know to be undecidable reduces to it.
Reminder: $\text{HALT}_\text{TM}$ is undecidable \textit{(Theorem 5.1)}

Proof (using reductions): We will show that $A_{\text{TM}}$ reduces to $\text{HALT}_\text{TM}$, and therefore (since $A_{\text{TM}}$ is undecidable), $\text{HALT}_\text{TM}$ must be undecidable.
Reminder: $\text{HALT}_{TM}$ is undecidable

**Proof** *(using reductions)*: We will show that $A_{TM}$ reduces to $\text{HALT}_{TM}$, and therefore (since $A_{TM}$ is undecidable), $\text{HALT}_{TM}$ must be undecidable.

How do we show that $A_{TM}$ reduces to $\text{HALT}_{TM}$?

A. Define an algorithm for $A_{TM}$ that uses a subroutine which checks for membership in $\text{HALT}_{TM}$.
B. Define an algorithm for $\text{HALT}_{TM}$ that uses a subroutine which checks for membership in $A_{TM}$.
C. None of the above.
D. I don't know.
Reminder: $\text{HALT}_{TM}$ is undecidable

Proof (using reductions): We will show that $A_{TM}$ reduces to $\text{HALT}_{TM}$, and therefore (since $A_{TM}$ is undecidable), $\text{HALT}_{TM}$ must be undecidable.

Assume that $M_{\text{HALT}}$ is a machine that decides $\text{HALT}_{TM}$.

Goal: Define decider for $A_{TM}$ using $M_{\text{HALT}}$ as subroutine.
Reminder: HALT<sub>TM</sub> is undecidable

Proof (using reductions): We will show that A<sub>TM</sub> reduces to HALT<sub>TM</sub>, and therefore (since A<sub>TM</sub> is undecidable), HALT<sub>TM</sub> must be undecidable.

Assume that M<sub>HALT</sub> is a machine that decides HALT<sub>TM</sub>.

Goal: Define decider for A<sub>TM</sub> using M<sub>HALT</sub> as subroutine.

What's the input to an algorithm that decides A<sub>TM</sub>?

A. w  
B. <M>  
C. <M,w>  
D. <M, <M> >  
E. I don't know.
Reminder: $\text{HALT}_{\text{TM}}$ is undecidable

Proof (using reductions): We will show that $A_{\text{TM}}$ reduces to $\text{HALT}_{\text{TM}}$, and therefore (since $A_{\text{TM}}$ is undecidable), $\text{HALT}_{\text{TM}}$ must be undecidable.

Assume that $M_{\text{HALT}}$ is a machine that decides $\text{HALT}_{\text{TM}}$.

Goal: Define decider for $A_{\text{TM}}$ using $M_{\text{HALT}}$ as subroutine.

"On input $<M,w>$ … Want to accept if $w$ in $L(M)$, reject o.w."
Reminder: $\text{HALT}_\text{TM}$ is undecidable

**Proof (using reductions):** We will show that $A_{\text{TM}}$ reduces to $\text{HALT}_\text{TM}$, and therefore (since $A_{\text{TM}}$ is undecidable), $\text{HALT}_\text{TM}$ must be undecidable.

Assume that $M_{\text{HALT}}$ is a machine that decides $\text{HALT}_\text{TM}$.

**Goal:** Define decider for $A_{\text{TM}}$ using $M_{\text{HALT}}$ as subroutine.

"On input $<M,w>$  Want to accept if $w$ in $L(M)$, reject o.w.
1. Run $M_{\text{HALT}}$ on $<M,w>$. If rejects, reject.
2. If accepts, run $M$ on $w$.
3. If accepts, accept; if rejects, reject."
Reminder: \( \text{HALT}_{\text{TM}} \) is undecidable

Proof (using reductions): We will show that \( \text{A}_{\text{TM}} \) reduces to \( \text{HALT}_{\text{TM}} \), and therefore (since \( \text{A}_{\text{TM}} \) is undecidable), \( \text{HALT}_{\text{TM}} \) must be undecidable.

Assume that \( \text{M}_{\text{HALT}} \) is a machine that decides \( \text{HALT}_{\text{TM}} \).

Goal: Define decider for \( \text{A}_{\text{TM}} \) using \( \text{M}_{\text{HALT}} \) as subroutine.

"On input \( \langle M,w \rangle \) [want to accept if \( w \) in \( L(M) \), reject o.w.]
1. Run \( \text{M}_{\text{HALT}} \) on \( \langle M,w \rangle \). If rejects, reject.
2. If accepts, run \( M \) on \( w \).
3. If accepts, accept; if rejects, reject."

Claim: this is a decider for \( \text{M}_{\text{ATM}} \) so \( \text{A}_{\text{TM}} \) reduces to \( \text{HALT}_{\text{TM}} \).
**Claim:** $E_{TM}$ is undecidable.  \( (Theorem \ 5.2) \)

\[
E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is empty} \}
\]
i.e. want to recognize codes of TMs that always reject / loop

- **Proof by reduction?**

To use proof by reduction to prove that $E_{TM}$ is undecidable, we must reduce an undecidable set to $E_{TM}$
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction**
  - **Goal:** show that $A_{TM}$ reduces to $E_{TM}$.
  - i.e. Build an algorithm that uses a decider for $E_{TM}$ as a subroutine and that decides $A_{TM}$
  - **Assume:** have a TM, $R$, that decides $E_{TM}$
  - **Build:** new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$. 
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction**
  - **Goal:** show that $A_{TM}$ reduces to $E_{TM}$.
  - i.e. Build an algorithm **that uses a decider for $E_{TM}$ as a subroutine** and that decides $A_{TM}$.
  - **Assume:** have a TM, $R$, that decides $E_{TM}$.
  - **Build:** new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

What's the input to $R$?

A. $w$
B. $<M>$
C. $<M,w>$
D. $<M, <M> >$
E. I don't know.
Claim: \( E_{\text{TM}} \) is undecidable.

- **Proof by reduction…**
  - Assume: have a TM, \( R \), that decides \( E_{\text{TM}} \)
  - Build: new TM, \( M_{\text{ATM}} \), that decides \( A_{\text{TM}} \)
    - Always halts
    - Accepts iff input \(<M,w>\) and \( w \) is in \( L(M) \).
  - "On input \(<M,w>\>:
    - Run \( R \) on input \(<M>\). If rejects, reject.
    - If accepts, run \( M \) on input \( w \).
      - If accepts, accept; if reject, reject."
Claim: \( E_{TM} \) is undecidable.

- Proof by reduction…
  - Assume: have a TM, \( R \), that decides \( E_{TM} \).
  - Build: new TM, \( M_{ATM} \), that decides \( A_{TM} \).
    - Always halts
    - Accepts iff input \( <M,w> \) and \( w \) is in \( L(M) \).
    - "On input \( <M,w> \):
      - Run \( R \) on input \( <M> \). If rejects, reject.
      - If accepts, run \( M \) on input \( w \).
        - If accepts, accept; if reject, reject."

Does the machine \( M_{ATM} \) always halt?

A. Yes.
B. No, not if \( L(M) \) is empty.
C. No, not if \( L(M) \) is nonempty.
D. No, not if \( M \) is not a decider.
E. I don't know.
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction…**
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
  - "On input $<M,w>$:
    - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$. 

**Fixed version**

Proof by reduction…
- Assume: have a TM, $R$, that decides $E_{TM}$
- Build: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
- "On input $<M,w>$:
  - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$. 

Claim: $E_{TM}$ is undecidable.
Claim: $E_{TM}$ is undecidable.

- Proof by reduction…
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
  - "On input $<M,w>$:
    - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.

For a given $<M,w>$, what's $L(X)$?
A. $\{w\}$
B. $w$
C. $\{ x | x \neq w \}$
D. $\Sigma^*$
E. The empty set.
**Claim:** $E_{TM}$ is undecidable.

- **Proof by reduction…**
  - **Assume:** have a TM, R, that decides $E_{TM}$
  - **Build:** new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
  - "On input $<M,w>$:
    - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.
    - Run $R$ on $<X>$.
      - If accepts, reject; if rejects; accept."
Claim: $E_{TM}$ is undecidable.

• Proof by reduction…
  • Assume: have a TM, R, that decides $E_{TM}$
  • Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    • Always halts
    • Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
    • "On input $<M,w>$:
      • First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.
      • Run $R$ on $<X>$.
        • If accepts, reject; if rejects; accept."
  • Correctness: Is $M_{ATM}$ a decider for $A_{TM}$?
### So far

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$HALT_{TM}$</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>$E_{TM}$</td>
</tr>
<tr>
<td>$ALL_{DFA}$ (HW)</td>
<td></td>
</tr>
</tbody>
</table>

Give algorithm!  
Diagonalization OR reduction
And another … (Theorem 5.3)

REGULAR\textsubscript{TM} = \{ <M> | M is TM and L(M) is regular \}

Claim: REGULAR\textsubscript{TM} is undecidable.

Proof: WTS A\textsubscript{TM} reduces to REGULAR\textsubscript{TM}

Assume R is TM that decides REGULAR\textsubscript{TM}. We will build a decider for A\textsubscript{TM} that calls R as a subroutine.

What should the decider for A\textsubscript{TM} do?
And another … (Theorem 5.3)

Claim: REGULAR_{TM} is undecidable.
Proof: WTS A_{TM} reduces to REGULAR_{TM}
Assume R is TM that decides REGULAR_{TM}. We will build a decider for A_{TM} that calls R as a subroutine.

"On input <M,w>
1. Build machine X such that L(x) is regular iff w is in L(M).
2. Run R on <X>.
3. If R accepts, accept. If R rejects, reject.
.."
How do we build X?

**Goal:** For fixed parameters M a TM and w a string
- if w in L(M) then L(X) is a regular set
- if w not in L(M) then L(X) is not a regular set

So... we need some example regular and non-regular sets

Regular sets: $\emptyset, L(0^*1^*), \Sigma^*$

Non-regular sets: $\{0^n1^n | n \geq 0\}$
How do we build X?

**Goal:** For fixed parameters $M$ a TM and $w$ a string

if $w$ in $L(M)$ then $L(X) = \Sigma^*$

if $w$ not in $L(M)$ then $L(X) = \{0^n1^n \mid n \geq 0\}$

**Construction:** $X = \text{"On input } x:\$

1. ...

2. ...

"
So far

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$HALT_{TM}$</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>$E_{TM}$</td>
</tr>
<tr>
<td>$ALL_{DFA} (HW)$</td>
<td>$REGULAR_{TM}$</td>
</tr>
<tr>
<td></td>
<td>${ &lt;M&gt;</td>
</tr>
</tbody>
</table>

Give algorithm!

Diagonalization OR reduction
{ <M> | M is a decider }

A new proof that this language is undecidable…

Can we show that $A_{TM}$ reduces to it?

Let $R$ be a decider for $\{ <M> | M \text{ is a decider} \}$. Build a decider for $A_{TM}$ by: "On input $<M, w>$

1. 
2. 
3. "
General approach

To prove that \{ <M> | M is a TM and L(M) has property P \} is undecidable

• Assume \textbf{towards a contradiction} that R is a decider for \{<M> | M is a TM and L(M) has P\}.
• Build decider for \( A_{TM} \) by: "On input <M,w> 
  1. Construct a new TM X such that X has P iff w in L(M) 
  2. Run R on <X>: if accepts, accept; if rejects, reject."

Note: sometimes easier to build X so that X has P iff w not in L(M)
More examples

• \{ <M> \mid M \text{ is a TM and } M \text{ accepts } w \text{ iff } M \text{ accepts } w^R \}\\
• \text{INF}_{TM} = \{ <M> \mid M \text{ is TM and } L(M) \text{ is infinite} \}\\
• \text{EQ}_{TM} = \{ <M_1, M_2> \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}