Lecture 10: Undecidability and Reductions
Next Thursday (3/23)

Your Midterm: 2:35-4:00pm,
Walker Memorial (50-340)

Today: no pset! just an optional (not graded)
practice midterm

Solutions to practice midterm will come out during
the weekend. Same with all remaining HW solutions.

When you see the practice midterm...
DON’T PANIC!

Practice midterm will be harder than midterm
Next Thursday (3/23)

Your Midterm: 2:35-4:00pm, Walker Memorial (50-340)

Today: no pset! just an optional (not graded) practice midterm

FAQ: What material is on the midterm?
Everything up to next Tuesday (Lectures 1-11)
But we’ll focus more on earlier material

FAQ: Can I bring notes?
Yes, one single-sided sheet of notes, letter paper
Theorem: $L$ is decidable iff both $L$ and $\neg L$ are recognizable.
A Concrete Undecidable Problem: The Acceptance Problem for TMs

$$A_{TM} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \}$$

Theorem [Turing]:
$$A_{TM}$$ is recognizable but **NOT** decidable
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

\( A_{TM} \) is undecidable: (proof by contradiction)

Assume \( H \) is a machine that decides \( A_{TM} \)

\[
H( (M,w) ) = \begin{cases} 
\text{Accept} & \text{if } M \text{ accepts } w \\
\text{Reject} & \text{if } M \text{ does not accept } w 
\end{cases}
\]

Define a new TM \( D \):

\[
D(M): \text{ Run } H \text{ on } (M,M) \text{ and output the opposite of } H
\]

\[
D( D ) = \begin{cases} 
\text{Reject} & \text{if } D \text{ accepts } D \\
\text{Accept} & \text{if } D \text{ does not accept } D 
\end{cases}
\]

Set \( M = D? \)
### The table of outputs of $H(x,y)$

<table>
<thead>
<tr>
<th></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>?</td>
</tr>
</tbody>
</table>

$M_1, M_2, \ldots$ and $w_1, w_2, \ldots$ are both lex-order lists of all binary strings.
The table of outputs of $H(x,y)$

<table>
<thead>
<tr>
<th></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
</tbody>
</table>

$D(x)$ outputs the opposite of $H(x,x)$
The behavior of $D(x)$ is a *diagonal* on this table.

<table>
<thead>
<tr>
<th></th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
<th>$w_4$</th>
<th>...</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td></td>
</tr>
</tbody>
</table>

$D(x)$ outputs the *opposite* of $H(x,x)$

$D(D)$ outputs the *opposite* of $H(D,D) = D(D)$
A_{TM} = \{(M,w) \mid M \text{ is a TM that accepts string } w \} \\
A_{TM} \text{ is undecidable: (a constructive proof)}

Let \( U \) be a machine that recognizes \( A_{TM} \)

\[
U((M,w)) = \begin{cases} 
  \text{Accept} & \text{if } M \text{ accepts } w \\
  \text{Rejects or loops} & \text{otherwise}
\end{cases}
\]

Define a new TM \( D_U \) as follows:

\( D_U(M) \): Run \( U \) on \((M,M)\) until the simulation halts. Output the opposite answer
\[ D_U( D_U ) = \begin{cases} 
\text{Reject if } D_U \text{ accepts } D_U \\
\text{(i.e. if } H( D_U , D_U ) = \text{Accept}) \\
\text{Accept if } D_U \text{ rejects } D_U \\
\text{(i.e. if } H( D_U , D_U ) = \text{Reject}) \\
\text{Loops if } D_U \text{ loops on } D_U \\
\text{(i.e. if } H( D_U , D_U ) \text{ loops}) 
\end{cases} \]

Note: There is no contradiction here!

\[ D_U \text{ must run forever on } D_U \]

We have an input \((D_U, D_U)\) which is not in \(A_{TM}\) but \(U\) infinitely loops on \((D_U, D_U)\)!
In summary:

Given the code of any machine $U$ that recognizes $A_{\text{TM}}$ (i.e. a Universal Turing Machine) we can effectively construct an input $(D_U, D_U)$, where:

1. $(D_U, D_U) \notin A_{\text{TM}}$
2. $U$ runs forever on the input $(D_U, D_U)$

Therefore $U$ cannot decide $A_{\text{TM}}$

Given any universal Turing machine, we can efficiently construct an input on which the program hangs!
Theorem: \( \overline{A_{TM}} \) is recognizable but NOT decidable

Corollary: \( \overline{\overline{A_{TM}}} \) is not recognizable!

Proof: Suppose \( \overline{\overline{A_{TM}}} \) is recognizable.
Then \( \overline{\overline{A_{TM}}} \) and \( A_{TM} \) are both recognizable.
But that would mean they’re both decidable...
... this is a contradiction!
The Halting Problem [Turing]

\[ \text{HALT}_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \} \]

**Theorem:** \( \text{HALT}_{TM} \) is undecidable

**Proof:** Assume (for a contradiction) there is a TM \( H \) that decides \( \text{HALT}_{TM} \)

Idea: Use \( H \) to construct a TM \( M' \) that decides \( A_{TM} \)

\[ M'(M,w): \text{ Run } H(M,w) \]
- If \( H \) rejects then *reject*
- If \( H \) accepts, run \( M \) on \( w \) until it halts:
  - If \( M \) accepts, then *accept*
  - If \( M \) rejects, then *reject*

**Claim:** If \( H \) exists, then \( M' \) decides \( A_{TM} \) \( \Rightarrow H \) does not exist!
If $M$ doesn't halt:

output reject

If $M$ halts:

Does $M$ halt on $w$?

output answer

$M'$ deciding $A_{TM}$
Can often prove a language $L$ is undecidable by proving: “if $L$ is decidable, then so is $A_{TM}$”

We reduce $A_{TM}$ to the language $L$

$$A_{TM} \leq L$$

Idea: $L$ is “at least as hard as” $A_{TM}$

Given the ability to solve problem $L$, we can solve $A_{TM}$
Reducing One Problem to Another

$f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if there is a Turing machine $M$ that halts with just $f(w)$ written on its tape, for every input $w$.

A **language $A$ is mapping reducible** to language $B$, written as $A \leq_m B$, if there is a computable $f : \Sigma^* \rightarrow \Sigma^*$ such that for every $w \in \Sigma^*$,

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

$f$ is called a mapping reduction (or many-one reduction) from $A$ to $B$. 
Let $f : \Sigma^* \to \Sigma^*$ be a **computable function** such that $w \in A \iff f(w) \in B$

Say: **“A is mapping reducible to B”**
Write: $A \leq_m B$
Theorem: If $A \leq_m B$ and $B \leq_m C$, then $A \leq_m C$
Some Examples

\[ A_{\text{DFA}} = \{ (D, w) \mid D \text{ encodes a DFA over some } \Sigma, \text{ and } D \text{ accepts } w \in \Sigma^* \} \]

\[ A_{\text{NFA}} = \{ (N, w) \mid N \text{ encodes an NFA, } D \text{ accepts } w \} \]

Theorem: For every regular language \( L' \), \( L' \leq_m A_{\text{DFA}} \)

For every regular language \( L' \), there’s a DFA \( D \) recognizing \( L' \).

So here’s a mapping reduction \( f \) from \( L' \) to \( A_{\text{DFA}} \):

\[ f(w) := \text{Output } (D, w) \]

Then, \( w \in L' \iff D \text{ accepts } w \iff f(w) = (D, w) \in A_{\text{DFA}} \)

So \( f \) is a mapping reduction from \( L' \) to \( A_{\text{DFA}} \)
Some Examples

\[ A_{DFA} = \{ (D, w) \mid D \text{ encodes a DFA over some } \Sigma, \text{ and } D \text{ accepts } w \in \Sigma^* \} \]

\[ A_{NFA} = \{ (N, w) \mid N \text{ encodes an NFA, } D \text{ accepts } w \} \]

Theorem: \[ A_{DFA} \leq_m A_{NFA} \]

Every DFA can be trivially written as an NFA. Here’s a mapping reduction \( f \) from \( A_{DFA} \) to \( A_{NFA} \):

\[ f(D, w) := \text{Construct NFA } N \text{ which is equivalent to } D \]

Output \((N, w)\)

Theorem: \[ A_{NFA} \leq_m A_{DFA} \]

\[ f(N, w) := \text{Use the subset construction to convert NFA } N \text{ into an equivalent DFA } D. \text{ Output } (D, w) \]
Theorem: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

Proof: Suppose TM $M$ decides $B$. Let $f$ be a mapping reduction from $A$ to $B$.

We build a machine $M'$ for deciding $A$

$M'(w)$:
1. Compute $f(w)$
2. Run $M$ on $f(w)$, output its answer

Then: $w \in A \iff f(w) \in B$ [since $f$ reduces $A$ to $B$]
$\iff M$ accepts $f(w)$ [since $M$ decides $B$]
$\iff M'$ accepts $w$ [by def of $M'$]
Theorem: If $A \leq_m B$ and $B$ is recognizable, then $A$ is recognizable.

Proof: Let $M$ recognize $B$. Let $f$ be a mapping reduction from $A$ to $B$.

To recognize $A$, we build a machine $M'$

$M'(w)$:

1. Compute $f(w)$
2. Run $M$ on $f(w)$, output its answer if you ever receive one
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

Theorem: If \( A \leq_m B \) and \( B \) is recognizable, then \( A \) is recognizable

Corollary: If \( A \leq_m B \) and \( A \) is unrecognizable, then \( B \) is unrecognizable
A mapping reduction from $A_{TM}$ to $HALT_{TM}$

Theorem: $A_{TM} \leq_m HALT_{TM}$

$f(z) :=$ Decode $z$ into a pair $(M, w)$

Construct a TM $M'$ with the specification:

“$M'(w) =$ Simulate $M$ on $w$. If the sim accepts then $accept$ else $loop$ forever”

Output $(M', w)$

We have $z \in A_{TM} \iff (M', w) \in HALT_{TM}$

Corollary: $HALT_{TM}$ is undecidable
Theorem: \( A_{TM} \leq_m HALT_{TM} \)

Corollary: \( \neg A_{TM} \leq_m \neg HALT_{TM} \)

Proof?

Corollary: \( \neg HALT_{TM} \) is unrecognizable!

Proof: If \( \neg HALT_{TM} \) were recognizable, then \( \neg A_{TM} \) would be recognizable...
Theorem: $\text{HALT}_{\text{TM}} \leq_m A_{\text{TM}}$

Proof: Define the computable function $f$:

$$f(z) := \text{Decode } z \text{ into a pair } (M, w)$$

Construct a TM $M'$ with the specification:

“$M'(w) = \text{Simulate } M \text{ on } w$. If the simulation halts, then accept”

Output $(M', w)$

Observe $(M, w) \in \text{HALT}_{\text{TM}} \iff (M', w) \in A_{\text{TM}}$
Corollary: $\text{HALT}_{\text{TM}} \equiv_m A_{\text{TM}}$

Yo, T.M.! I can give you the magical power to either solve the halting problem, or the acceptance problem. Which do you want?

Wow, hm, so hard to choose...

I can't decide!