CS 154

Lecture 9:
Diagonalization, Undecidability, Unrecognizability
“There are more problems to solve than there are programs to solve them.”

Languages over \{0,1\}
Let $L$ be any set and $2^L$ be the power set of $L$.

**Theorem:** There is no onto function from $L$ to $2^L$.

**Proof:** Assume, for a contradiction, there is an onto function $f : L \rightarrow 2^L$.

Define $S = \{ x \in L \mid x \notin f(x) \} \in 2^L$.

If $f$ is onto, then there is a $y \in L$ with $f(y) = S$.

Suppose $y \in S$. By definition of $S$, $y \notin f(y) = S$.

Suppose $y \notin S$. By definition of $S$, $y \in f(y) = S$.

Contradiction!
\[ f : A \rightarrow B \text{ is not onto } \iff (\exists b \in B)(\forall a \in A)[f(a) \neq b] \]

Let \( L \) be any set and \( 2^L \) be the power set of \( L \).

**Theorem:** There is no onto function from \( L \) to \( 2^L \).

**Proof:** Let \( f : L \rightarrow 2^L \) be an arbitrary function.

Define \( S = \{ x \in L \mid x \notin f(x) \} \in 2^L \)

For all \( x \in L \),

- If \( x \in S \) then \( x \notin f(x) \) \quad \text{[by definition of \( S \)]}
- If \( x \notin S \) then \( x \in f(x) \)

In either case, we have \( f(x) \neq S \). (Why?)

Therefore \( f \) is not onto!
What does this mean?

No function from \( \mathbb{L} \) to \( 2^\mathbb{L} \) can “cover” all the elements in \( 2^\mathbb{L} \)

No matter what the set \( \mathbb{L} \) is, the power set \( 2^\mathbb{L} \) *always* has strictly larger cardinality than \( \mathbb{L} \)
**Thm:** There are *unrecognizable* languages

**Proof:** If all languages were recognizable, then for all $L$, there’d be a Turing machine $M$ for recognizing $L$. Hence there is an onto $R: \{\text{Turing Machines}\} \rightarrow \{\text{Languages}\}$.

Therefore, there is *no* onto function from $\{\text{Turing Machines}\} \subseteq M$ to $\{\text{Languages}\}$. *Contradiction!*
In the early 1900’s, logicians were trying to define consistent foundations for mathematics.

Suppose $X = \text{“Universe of all possible sets”}$

Frege’s Axiom: Let $f : X \rightarrow \{0,1\}$

Then $\{S \in X \mid f(S) = 1\}$ is a set.

Define $F = \{S \in X \mid S \notin S\}$

Suppose $F \in F$. Then by definition, $F \notin F$.

So $F \notin F$ and by definition $F \in F$.

This logical system is inconsistent!
Theorem: There is no onto function from the positive integers \( \mathbb{Z}^+ \) to the real numbers in \( (0, 1) \).

\[ \{0,1\}^* \quad \text{Power set of } \{0,1\}^* \]

Proof: Suppose \( f \) is such a function:

\[
\begin{align*}
1 & \rightarrow 0.28347279... \\
2 & \rightarrow 0.88388384... \\
3 & \rightarrow 0.77635284... \\
4 & \rightarrow 0.11111111... \\
5 & \rightarrow 0.12345678... \\
\vdots & \\
\end{align*}
\]

Define: \( r \in (0, 1) \):

\[
[\text{n-th digit of } r] = \begin{cases} 
1 & \text{if } [\text{n-th digit of } f(n)] \neq 1 \\
2 & \text{otherwise} 
\end{cases}
\]

\( f(n) \neq r \) for all \( n \) \hspace{1cm} (Here, \( r = 0.11121... \))
Let $Z^+ = \{1, 2, 3, 4, \ldots\}$

There is a bijection between $Z^+$ and $Z^+ \times Z^+$

$\begin{align*}
(1,1) & \quad (1,2) & \quad (1,3) & \quad (1,4) & \quad (1,5) & \ldots \\
(2,1) & \quad (2,2) & \quad (2,3) & \quad (2,4) & \quad (2,5) & \ldots \\
(3,1) & \quad (3,2) & \quad (3,3) & \quad (3,4) & \quad (3,5) & \ldots \\
(4,1) & \quad (4,2) & \quad (4,3) & \quad (4,4) & \quad (4,5) & \ldots \\
(5,1) & \quad (5,2) & \quad (5,3) & \quad (5,4) & \quad (5,5) & \ldots 
\end{align*}$
Calkin-Wilf Tree
A Concrete Undecidable Problem: The Acceptance Problem for TMs

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

**Theorem:** $A_{TM}$ is recognizable but **NOT** decidable

**Corollary:** $\neg A_{TM}$ is not recognizable
$$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$$

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

Suppose $H$ 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 $D$ as follows:

$D(M)$: Run $H$ on $(M,M)$ and output the opposite of $H$

$$D( D ) = \begin{cases} 
\text{Reject} & \text{if } D \text{ accepts } D \\
\text{Accept} & \text{if } D \text{ does not accept } D 
\end{cases}$$
The table of outputs of $H(x,y)$

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>accept</td>
<td>accept</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>
</tr>
<tr>
<td>$M_3$</td>
<td>accept</td>
<td>reject</td>
<td>reject</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>accept</td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td>?</td>
</tr>
</tbody>
</table>
The outputs of $D(x)$

<table>
<thead>
<tr>
<th></th>
<th>$M_1$</th>
<th>$M_2$</th>
<th>$M_3$</th>
<th>$M_4$</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>accept</td>
</tr>
<tr>
<td>$M_2$</td>
<td>reject</td>
<td>reject</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>accept</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>accept</td>
<td>accept</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>

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

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

$A_{TM}$ is undecidable: (constructive proof)

Let $H$ be a machine that recognizes $A_{TM}$

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

Define a new TM $D_H$ as follows:

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

Note: There is no contradiction here!

$D_H$ must loop on $D_H$

We have an instance $(D_H, D_H)$ which is not in $A_{TM}$ but $H$ fails to tell us that!

$H(D_H, D_H)$ runs forever
That is:

Given the code of any machine H that recognizes $A_{TM}$ we can effectively construct an instance $(D_H, D_H)$, where:

1. $(D_H, D_H)$ does not belong to $A_{TM}$
2. H runs forever on the input $(D_H, D_H)$

So H cannot decide $A_{TM}$

Given any program that recognizes the Acceptance Problem, we can efficiently construct an input where the program hangs!
Theorem: $A_{TM}$ is recognizable but NOT decidable

Corollary: $\neg A_{TM}$ is not recognizable!

Proof: Suppose $\neg A_{TM}$ is recognizable. Then $\neg A_{TM}$ and $A_{TM}$ are both recognizable...
But that would mean they’re both decidable!
The Halting Problem

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

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

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

We use \( H \) to construct a TM \( M' \) that decides \( A_{\text{TM}} \)

\[ M'(M,w): \text{ Run } H(M,w) \]

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*
If $M$ doesn’t halt:

- **Reject**

If $M$ halts:

- Does $M$ halt on $w$?

Diagram:

- $(M, w)$
- $M'$
- $H$
- $M$
- $w$
- If $M$ halts
- If $M$ doesn’t halt: **reject**
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 $L$.

$$A_{TM} \leq L$$
Mapping Reductions

$f : \Sigma^* \rightarrow \Sigma^*$ 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 $w$,

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

$f$ is called a mapping reduction (or many-one reduction) from $A$ to $B$. 
Let $f : \Sigma^* \rightarrow \Sigma^*$ 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 $A \leq_m C$
Theorem: If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable

Proof: Let \( M \) decide \( B \).

Let \( f \) be a mapping reduction from \( A \) to \( B \).

To decide \( A \), we build a machine \( M' \):

\[ M'(w): \]

1. Compute \( f(w) \)
2. Run \( M \) on \( f(w) \), output its answer

\( w \in A \iff f(w) \in B \) so \( w \in A \implies M' \) accepts \( w \)

\( w \notin A \implies M' \) rejects \( w \)
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
The proof that the Halting Problem is undecidable can be seen as constructing a mapping reduction from $A_{TM}$ to $HALT_{TM}$

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

$$f(M, w) := \text{Construct } M' \text{ with the specification}$$

"$M'(w) = \text{if } M(w) \text{ accepts then } accept \text{ else loop forever}"$$

Output $(M', w)$

We have $(M, w) \in A_{TM} \iff (M', w) \in HALT_{TM}$
Another way of writing the reduction $f$:

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

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

Construct $M'$ with the specification:

"$M'(w) =$ Simulate $M$ on $w$.
if $M(w)$ accepts then accept
else loop forever"

Output $(M', w)$

We have $z \in A_{TM} \iff (M', w) \in HALT_{TM}$
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: \( \text{HALT}_{\text{TM}} \leq_m \text{A}_{\text{TM}} \)

Proof: Define the computable function:

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

Construct \( M' \) with the specification:

“\( M'(w) = \text{Simulate } M \text{ on } w. \) If \( M(w) \) halts then accept

else \( \text{loop forever} \)”

Output \( (M', w) \)

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

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

Wow, hm, so hard to choose...

I can’t decide!
The Emptiness Problem

\[ \text{EMPTY}_{\text{DFA}} = \{ M \mid M \text{ is a DFA such that } L(M) = \emptyset \} \]

*Given a DFA, does it reject every input?*

**Theorem:** \( \text{EMPTY}_{\text{DFA}} \) is decidable

**Why?**

\[ \text{EMPTY}_{\text{NFA}} = \{ M \mid M \text{ is a NFA such that } L(M) = \emptyset \} \]

\[ \text{EMPTY}_{\text{REX}} = \{ R \mid M \text{ is a regexp such that } L(M) = \emptyset \} \]
The Emptiness Problem for TMs

\[ \text{EMPTY}_{\text{TM}} = \{ M \mid \text{M is a TM such that } L(M) = \emptyset \} \]

*Given a program, does it reject every input?*

**Theorem:** \( \text{EMPTY}_{\text{TM}} \) is *not recognizable*

**Proof:** Show that \( \neg \text{A}_{\text{TM}} \leq_m \text{EMPTY}_{\text{TM}} \)

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

Output a TM \( M' \) with the behavior:

“\( M'(x) := \text{if } (x = w) \text{ then run } M(w), \text{ else reject} \)”

\[
\begin{align*}
z \in \text{A}_{\text{TM}} & \iff L(M') \neq \emptyset \\& \\
& \iff M' \notin \text{EMPTY}_{\text{TM}} \\& \\
& \iff f(z) \notin \text{EMPTY}_{\text{TM}}
\end{align*}
\]