Problem 1 \text{ DECIDABLE ?}

\{ (M, w) | M is a TM that on input w, tries to move its head past the left end of the tape \}

Problem 2 \text{ DECIDABLE ?}

\{ (M, w) | M is a TM that on input w, moves its head left at least once, at some point \}
Problem 1 UNDECIDABLE

\{ (M, w) | M is a TM that on input w, tries to move its head past the left end of the tape \}

Proof: Assume, for a contradiction, that TM T decides the language

We use T to decide \( A_{\text{TM}} \)

On input \((M,w)\), make a new TM \( N \) that on input \( w \) marks the leftmost tape cell and then simulates \( M(w) \) (as tho the leftmost cell was not there). If \( M \) tries to move to the marked cell, \( N \) moves the head back to the right. If \( M \) accepts, \( N \) tries to move its head past the left end of the tape.

Run T on input \((N,w)\)
Problem 2 DECIDABLE

\{(M, w) \mid M \text{ is a TM that on input } w, \text{ moves its head left at least once, at some point}\}

On input \((M, w)\), run the machine for \(|Q_M| + |w| + 1\) steps:

- **Accept** If M’s head moved left at all
- **Reject** Otherwise

(Why does this work??)
RICE’S THEOREM, THE RECURSION THEOREM, AND THE FIXED-POINT THEOREM

THURSDAY FEB 27
\[ \text{FIN}_{\text{TM}} = \{ M \mid M \text{ is a TM and } L(M) \text{ is finite} \} \]

Is \( \text{FIN}_{\text{TM}} \) Decidable?
\[ \text{FIN}_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is finite} \} \]

Is \( \text{FIN}_{TM} \) Decidable?

Note Properties of this language:

- \( \text{FIN}_{TM} \) is a language of Turing Machines
- If \( M_1 \equiv M_2 \) (ie \( L(M_1) = L(M_2) \)), then either both \( M_1 \) and \( M_2 \) are in \( \text{FIN}_{TM} \) or both are not.
- There are TMs \( M_1 \) and \( M_2 \), such that \( M_1 \in \text{FIN}_{TM} \) and \( M_2 \notin \text{FIN}_{TM} \)
RICE’S THEOREM

Let $L$ be a language over Turing machines. Assume that $L$ satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 \equiv M_2$ then
   $M_1 \in L \iff M_2 \in L$

2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

EXTREMELY POWERFUL!
RICE’S THEOREM

Let $L$ be a language over Turing machines. Assume that $L$ satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 \equiv M_2$ then 
   \[ M_1 \in L \iff M_2 \in L \]

2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \notin L$

Then $L$ is undecidable

\[ F\text{IN}_{\text{TM}} = \{ M \mid M \text{ is a TM and } L(M) \text{ is finite} \} \]
RICE’S THEOREM

Let $L$ be a language over Turing machines. Assume that $L$ satisfies the following properties:

1. For TMs $M_1$ and $M_2$, if $M_1 \equiv M_2$ then $M_1 \in L \iff M_2 \in L$

2. There are TMs $M_1$ and $M_2$, such that $M_1 \in L$ and $M_2 \not\in L$

Then $L$ is undecidable

$$E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \}$$

$$REG_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \}$$
Then L is undecidable

Proof: Will show:

\( \mathbb{A}_{TM} \) is mapping reducible to L
Proof: Show $L$ is undecidable

Show: $A_{TM}$ is mapping reducible to $L$
Proof: Show $L$ is undecidable

Show: $A_{TM}$ is mapping reducible to $L$
RICE’S THEOREM

Proof:

Define $M_\emptyset$ to be a TM that never halts.

Assume, WLOG, that $M_\emptyset \notin L$ Why?

Let $M_1 \in L$ (such $M_1$ exists, by assumption)

Show $A_{TM}$ is mapping reducible to $L$:
RICE’S THEOREM

Proof:

Define $M_\emptyset$ to be a TM that never halts

Assume, WLOG, that $M_\emptyset \not\in L$ Why?

Let $M_1 \in L$ (such $M_1$ exists, by assumption)

Show $A_{TM}$ is mapping reducible to $L$:

Map $(M, w) \rightarrow M_w$ where

$M_w(s) = \text{accepts if both } M(w) \text{ and } M_1(s) \text{ accept loops otherwise}$

What is the language of $M_w$?
$A_{TM}$ is mapping reducible to $L$

\[ \Sigma^* \xrightarrow{f} A_{TM} \xrightarrow{f} \Sigma^* \]

\[(M,w) \xrightarrow{f} \{M_w \equiv M_1 \} \]

\[(M,w) \xrightarrow{f} \{M_w \equiv M_\emptyset \} \]

QED
Problem

Let $S = \{ M \mid M$ is a TM with the property:
for all $w$, $M(w)$ accepts implies $M(w^R)$ accepts $\}$. $S$ is undecidable.
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

$HALT_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \}$

$E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \}$

$REG_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \}$

$EQ_{TM} = \{ (M,N) \mid M, N \text{ are TMs and } L(M) = L(N) \}$

$ALL_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \}$

**ALL UNDECIDABLE**

*Where is Rice’s Theorm Applicable?*

*Which are SEMI-DECIDABLE or not?*
The rest of the content of today’s lecture has been a major source of headaches and misunderstandings.
“The recursion theorem is just like tennis. Unless you're exposed to it at age five, you'll never become world class.”

-Juris Hartmanis (Turing Award 1993)

(Note: Juris didn’t see the recursion theorem until he was in his 20’s....)
Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. Then there is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$
Theorem: Let \( T \) be a Turing machine that computes a function \( t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^* \).

Then there is a Turing machine \( R \) that computes a function \( r : \Sigma^* \rightarrow \Sigma^* \), where for every string \( w \),

\[
r(w) = t(<R>, w)
\]

\[
(a,b) \rightarrow T \rightarrow t(a,b)
\]

\[
w \rightarrow R \rightarrow t(<R>,w)
\]
Recursion Theorem says:
A Turing machine can obtain its own description (code), and compute with it.

We can use the operation:
"Obtain your own description" in pseudocode!

Given a computable $t$, we can get a computable $r$ such that $r(w) = t(<R>, w)$ where $<R>$ is a description of $r$. 
Recursion Theorem says:
A Turing machine can obtain its own description (code), and compute with it.

We can use the operation:
"Obtain your own description"
in pseudocode!

Given a computable $t$, we can get a computable $r$ such that $r(w) = t(<R>,w)$ where $<R>$ is a description of $r$.

INSIGHT: $T$ (or $t$) is really $R$ (or $r$).
Theorem: \( A_{TM} \) is undecidable

Proof (using the Recursion Theorem):

Assume \( H \) decides \( A_{TM} \) (Informal Proof)

Construct machine \( R \) such that on input \( w \):

1. Obtains its own description \( <R> \)

2. Runs \( H \) on \( (<R>, w) \) and flips the output

Running \( R \) on input \( w \) always does the opposite of what \( H \) says it should!
Theorem: $A_{TM}$ is undecidable

Proof (using the Recursion Theorem):

Assume $H$ decides $A_{TM}$ (Formal Proof)

Let $T_H(x, w) = \begin{cases} 
\text{Reject if } H(x, w) \text{ accepts} \\
\text{Accept if } H(x, w) \text{ rejects} 
\end{cases}$

(Here $x$ is viewed as a code for a TM)

By the Recursion Theorem, there is a TM $R$ such that:

$R(w) = T_H(<R>, w) = \begin{cases} 
\text{Reject if } H(<R>, w) \text{ accepts} \\
\text{Accept if } H(<R>, w) \text{ rejects} 
\end{cases}$

Contradiction!
Theorem: \( \text{MIN}_{\text{TM}} \) is not RE.

Proof (using the Recursion Theorem):

\[
\text{MIN}_{\text{TM}} = \{<M>| M \text{ is a minimal TM, wrt } |<M>|\}
\]
Theorem: \( \text{MIN}_TM \) is not RE.

Proof (using the Recursion Theorem):

Assume \( E \) enumerates \( \text{MIN}_TM \) (Informal Proof)

Construct machine \( R \) such that on input \( w \):

1. Obtains its own description \(<R>\)
2. Runs \( E \) until a machine \( D \) appears with a longer description than of \( R \)
3. Simulate \( D \) on \( w \)

Contradiction. Why?
Theorem: \( \text{MIN}_{\text{TM}} \) is not \( \text{RE} \).

Proof (using the Recursion Theorem):

Assume \( E \) enumerates \( \text{MIN}_{\text{TM}} \) (Formal Proof)

Let \( T_E(x, w) = D(w) \) where \( <D> \) is first in \( E \)'s enumeration s.t. \( |<D>| > |x| \)

By the *Recursion Theorem*, there is a \( \text{TM} \ R \) such that:

\[ R(w) = T_E(<R>, w) = D(w) \]

where \( <D> \) is first in \( E \)'s enumeration s.t. \( |<D>| > |<R>| \)

Contradiction. Why?
THE FIXED-POINT THEOREM

Theorem: Let $f : \Sigma^* \rightarrow \Sigma^*$ be a computable function. There is a TM $R$ such that $f(<R>)$ describes a TM that is equivalent to $R$. 
THE FIXED-POINT THEOREM

Theorem: Let \( f : \Sigma^* \rightarrow \Sigma^* \) be a computable function. There is a TM \( R \) such that \( f(<R>) \) describes a TM that is equivalent to \( R \).

Proof: Pseudocode for the TM \( R \):

On input \( w \):

1. Obtain the description \( <R> \)
2. Let \( g = f(<R>) \) and interpret \( g \) as a code for a TM \( G \)
3. Accept \( w \) iff \( G(w) \) accepts
Theorem: Let $f : \Sigma^* \to \Sigma^*$ be a computable function. There is a TM $R$ such that $f(<R>)$ describes a TM that is equivalent to $R$.

Proof: Let $T_f(x, w) = G(w)$ where $<G> = f(x)$

(Here $f(x)$ is viewed as a code for a TM)

By the Recursion Theorem, there is a TM $R$ such that:

$R(w) = T_f(<R>, w)$
THE FIXED-POINT THEOREM

Theorem: Let $f : \Sigma^* \rightarrow \Sigma^*$ be a computable function. There is a TM $R$ such that $f(<R>)$ describes a TM that is equivalent to $R$.

Proof: Let $T_f(x, w) = G(w)$ where $<G> = f(x)$
(Here $f(x)$ is viewed as a code for a TM)

By the Recursion Theorem, there is a TM $R$ such that:

$R(w) = T_f(<R>, w) = G(w)$ where $<G> = f(<R>)$

Hence $R \equiv G$ where $<G> = f(<R>)$, ie $<R> \equiv f(<R>)$

So $R$ is a fixed point of $f$!
THE FIXED-POINT THEOREM

**Theorem:** Let \( f : \Sigma^* \rightarrow \Sigma^* \) be a computable function. There is a TM \( R \) such that \( f(<R>) \) describes a TM that is *equivalent* to \( R \).

**Example:**

Suppose a virus flips the first bit of each word \( w \) in \( \Sigma^* \) (or in each TM).

Then there is a TM \( R \) that “remains uninfected”.
Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$.

Then there is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$
THE RECURSION THEOREM

Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \to \Sigma^*$.

Then there is a Turing machine $R$ that computes a function $r : \Sigma^* \to \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$

So first, need to show how to construct a TM that computes its own description (ie code).
Suppose in general we want to design a program that prints its own description. How?

Print this sentence.

Print two copies of the following (the stuff inside quotes), and put the second copy in quotes:
“Print two copies of the following (the stuff inside quotes), and put the second copy in quotes:”
Lemma: There is a computable function $q : \Sigma^* \rightarrow \Sigma^*$, where for any string $w$, $q(w)$ is the \textit{description (code)} of a TM $P_w$ that on any input, prints out $w$ and then accepts
A TM SELF THAT PRINTS <SELF>

B (<M>) = < P_{<M>} M >  where  P_{<M>} M (w') = M (<M>)
A TM SELF THAT PRINTS \( \langle \text{SELF} \rangle \)

\[ \langle B \rangle \rightarrow B \rightarrow w' \rightarrow P_{<B>} \rightarrow B \rightarrow B (\langle B \rangle) \]

\[ B (\langle M \rangle) = \langle P_{<M>} M \rangle \text{ where } P_{<M>} M (w') = M (\langle M \rangle) \]

So, \( B (\langle B \rangle) = \langle P_{<B>} B \rangle \text{ where } P_{<B>} B (w') = B (\langle B \rangle) \)
A TM SELF THAT PRINTS <SELF>

B (<M>) = < P_<M> M > where P_<M> M (w') = M (<M>)

So, B (<B>) = < P_<B> B > where P_<B> B (w') = B (<B>)

Now, P_<B> B (w') = B(<B>) = <P_<B> B >

So, let SELF = P_<B> B
A TM SELF THAT PRINTS <SELF>

\[ \langle M \rangle \xrightarrow{\text{B}} \langle M \rangle \]

\[ \langle B \rangle \xrightarrow{\text{B}} \langle B \rangle \]

\[ \langle M \rangle \xrightarrow{\text{M}} \langle M \rangle \]

\[ \langle B \rangle \xrightarrow{\text{B}} \langle B \rangle \]
A TM SELF THAT PRINTS <SELF>
A NOTE ON SELF REFERENCE

Suppose in general we want to design a program that prints its own description. How?

Print this sentence.

Print two copies of the following (the stuff inside quotes), and put the second copy in quotes:

“Print two copies of the following (the stuff inside quotes), and put the second copy in quotes:”

= B

= P<B>
Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. There is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$,

$$r(w) = t(<R>, w)$$
Proof:

\[(a,b) \rightarrow T \rightarrow t(a,b)\]
Proof:

\[(a,b) \rightarrow T \rightarrow t(a,b)\]

\[<M> \rightarrow B \rightarrow \rightarrow \rightarrow P \rightarrow M\]
Proof:

(a,b) \rightarrow (a,b) 

\langle M \rangle \rightarrow B 

\langle M \rangle \rightarrow P \langle M \rangle \rightarrow M 

w \rightarrow P \langle BT \rangle \rightarrow B 

w' \rightarrow P \langle BT \rangle \rightarrow BT 

Proof:

\[(a,b) \xrightarrow{} T \xrightarrow{} t(a,b)\]

\[<M> \xrightarrow{} B \xrightarrow{} \xrightarrow{w'} P^{<M>} \xrightarrow{} M \xrightarrow{} R\]

\[w \xrightarrow{} P^{<BT>} \xrightarrow{<BT>} B \xrightarrow{} \xrightarrow{w'} R\]
Proof:

\[(a, b) \rightarrow T \rightarrow t(a, b)\]

\[w' \rightarrow P^{<M>} \rightarrow M\]

\[w\rightarrow P^{<BT>} \rightarrow B \rightarrow R\]

\[<R> = ???\]
Proof:

(a, b) \rightarrow T \rightarrow t(a, b)

\langle M \rangle \rightarrow B \rightarrow w'

\langle R \rangle = ???

w \rightarrow P \langle BT \rangle \rightarrow B \rightarrow P \langle BT \rangle \rightarrow BT
Proof:

(a, b) \rightarrow T \rightarrow t(a, b)

\langle M \rangle \rightarrow B

\langle M \rangle \rightarrow P \langle M \rangle \rightarrow w'

\langle R \rangle (= \langle P_{BT}BT \rangle)

w \rightarrow P \langle BT \rangle \rightarrow B

w' \rightarrow \langle BT \rangle \rightarrow BT
Proof:

$$(a,b) \rightarrow T \rightarrow t(a,b)$$

$$<M> \rightarrow B \rightarrow w'$$

$$<R> = <P_{<BT>BT}>$$

$$w \rightarrow P_{<BT>} \rightarrow B \rightarrow w'$$
Proof:

\[(a, b) \rightarrow T \rightarrow t(a, b)\]

\[\langle M \rangle \rightarrow B \rightarrow w' \rightarrow P_{<M>} \rightarrow M \]

\[\langle R \rangle = \langle P_{<BT>BT}> \]

\[w \rightarrow P_{<BT>} \rightarrow BT \rightarrow B \rightarrow T \]
Proof:

\[(a, b) \rightarrow T \rightarrow t(a, b)\]

\[<M> \rightarrow B \rightarrow w' \rightarrow P_{<M>} \rightarrow M\]

\[R \rightarrow <R> (= P_{<BT> BT>} \rightarrow T \rightarrow t(<R>, w)\]

\[w \rightarrow P_{<BT>} \rightarrow B \rightarrow t(<R>, w)\]
Theorem: Let $T$ be a Turing machine that computes a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$. There is a Turing machine $R$ that computes a function $r : \Sigma^* \rightarrow \Sigma^*$, where for every string $w$, $r(w) = t(<R>, w)$.
Read Chapter 6.1 and 6.3 for next time