15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY
UNDECIDABILITY II: REDUCTIONS

TUESDAY Feb 18
$A_{TM} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \}$

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

Assume machine $H$ semi-decides $A_{TM}$ (such exist, why?)

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

Construct a new TM $D_{H}$ as follows: on input $M$, run $H$ on $(M, M)$ and output the “opposite” of $H$ whenever possible.
\[ D_H(M) = \begin{cases} 
\text{Reject if } M \text{ accepts } M \\
\text{(i.e. if } H(M, M) = \text{Accept}) \\
\text{Accept if } M \text{ rejects } M \\
\text{(i.e. if } H(M, M) = \text{Reject}) \\
\text{loops if } M \text{ loops on } M \\
\text{(i.e. if } H(M, M) \text{ loops}) 
\end{cases} \]
\[ D_H ( D_H ) = \begin{cases} 
\text{Reject if } D_H \text{ accepts } D_H \\
\text{(i.e. if } H( D_H , D_H ) = \text{Accept}) \\
\text{Accept if } D_H \text{ rejects } D_H \\
\text{(i.e. if } H( D_H , D_H ) = \text{Reject}) \\
\text{loops if } D_H \text{ loops on } D_H \\
\text{(i.e. if } H( D_H , D_H ) \text{ loops}) 
\end{cases} \]

**Note:** It must be the case that \( D_H \) loops on \( D_H \)

There is no contradiction here!

Thus we **effectively** constructed an instance which does not belong to \( A_{TM} \) (namely, \((D_H, D_H)\)) but \( H \) fails to tell us that.
That is:

Given any semi-decision machine $H$ for $A_{TM}$ (and thus a potential decision machine for $A_{TM}$), we can effectively construct an instance which does not belong to $A_{TM}$ (namely, $(D_H, D_H)$) but $H$ fails to tell us that.

So $H$ cannot be a decision machine for $A_{TM}$.
In most cases, we will show that a language $L$ is undecidable by showing that if it is decidable, then so is $A_{TM}$.

We *reduce* deciding $A_{TM}$ to deciding the language in question:

$$A_{TM} \preceq L$$
\( \mathcal{A}_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \)

\( \mathcal{H}_{TM} = \{ (M,w) \mid M \text{ is a TM that halts on string } w \} \)

\( \mathcal{E}_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \} \)

\( \mathcal{R}_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \} \)

\( \mathcal{E}_{Q_{TM}} = \{( M, N) \mid M, N \text{ are TMs and } L(M) = L(N)\} \)

\( \mathcal{A}_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \} \)

**ALL UNDECIDABLE**

(*) Use Reductions to Prove Which are SEMI-DECIDABLE?
THE HALTING PROBLEM

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

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

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

We use \( H \) to construct a TM \( D \) that decides \( A_{TM} \)
THE HALTING PROBLEM

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

Theorem: HALT_{TM} is undecidable

Proof: Assume, for a contradiction, that TM H decides HALT_{TM}

We use H to construct a TM D that decides A_{TM}

On input (M,w), D runs H on (M,w)

If H rejects then reject

If H accepts, run M on w until it halts:

Accept if M accepts and
Reject if M rejects
If $M$ doesn't halt: REJECT

If $M$ halts:

Does $M$ halt on $w$?

D

H

If $M$ doesn't halt: REJECT

ACCEPT if halts in accept state

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

**Theorem:** $E_{TM}$ is undecidable

**Proof:** Assume, for a contradiction, that TM $Z$ decides $E_{TM}$. Use $Z$ as a subroutine to decide $A_{TM}$.
$$E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \}$$

**Theorem:** $E_{TM}$ is undecidable

**Proof:** Assume, for a contradiction, that TM $Z$ decides $E_{TM}$. Use $Z$ as a subroutine to decide $A_{TM}$.

**Algorithm for deciding $A_{TM}$:** On input $(M,w)$:

1. Create $M_w$.

   $M_w$

   - If $s \neq w$, REJECT
   - If $s = w$, run $M(w)$

2. Run $Z$ on $M_w$.

   So, $L(M_w) = \emptyset \iff M(w) \text{ does not accept}$
   $L(M_w) \neq \emptyset \iff M(w) \text{ accepts}$
If $s \neq w$, REJECT
If $s = w$, run $M(w)$

So, $L(M_w) = \emptyset \iff M(w)$
does not accept

$L(N) = \emptyset$?
Accepts if $M$ does not accept $w$
Rejects, otherwise

If $s \neq w$, REJECT
If $s = w$, run $M(w)$

So, $L(M_w) = \emptyset \iff M(w)$ does not accept

Decision Machine for $A_{TM}$

Accepts if $M$ does not accept $w$
Rejects, otherwise

$Z$
$L(M_w) = \emptyset$?

REVERSE accept/reject
$\text{REGULAR}_{\text{TM}} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular}\}$

**Theorem:** $\text{REGULAR}_{\text{TM}}$ is undecidable

**Proof:** Assume, for a contradiction, that TM $R$ decides $\text{REGULAR}_{\text{TM}}$

Use $R$ as a subroutine to decide $A_{\text{TM}}$
REGULAR\textsubscript{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular}\}

**Theorem:** REGULAR\textsubscript{TM} is undecidable

**Proof:** Assume, for a contradiction, that TM R decides REGULAR\textsubscript{TM}

Use R as a subroutine to decide ATM

1. Create \( M'w \)

   \[
   s \rightarrow \begin{cases} 
   \text{If } s = 0^n1^n, \text{ accept} \\
   \text{Else run } M(w) 
   \end{cases}
   \]

   So, \( L(M'w) = \Sigma^* \iff M(w) \text{ accepts} \)

   \( L(M'w) = \{0^n1^n\} \iff M(w) \text{ does not accept} \)

2. Run R on \( M'w \)
Is $L(N)$ regular?

If $s = 0^n1^n$, accept
Else run $M(w)$

$L(M_w') = \Sigma^*$ if $M(w)$ accepts
\{0^n1^n\} otherwise

$L(M_w')$ is regular $\iff$ $M(w)$ accepts
If $s = 0^n1^n$, accept
Else run $M(w)$

$L(M_w') = \Sigma^*$ if $M(w)$ accepts
\{0^n1^n\} otherwise

$L(M_w')$ is regular $\iff$ $M(w)$ accepts

Is $L(M_w')$ regular?

Yes $\iff$ $M$ accepts $w$
f : Σ* → Σ* is a **computable function** if some Turing machine M, on every input w, halts with just f(w) on its tape.

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

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

f is called a **reduction** from A to B.

Think of f as a “**computable coding**” from A to B.
A is mapping reducible to B, \( A \leq_m B \), if there is a computable \( f : \Sigma^* \rightarrow \Sigma^* \) such that \( w \in A \iff f(w) \in B \).

Also, \( \neg A \leq_m \neg B \), why?
Theorem: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable

Proof: Let $M$ decide $B$ and let $f$ be a reduction from $A$ to $B$

We build a machine $N$ that decides $A$ as follows:

On input $w$:

1. Compute $f(w)$
2. Run $M$ on $f(w)$
Theorem: If $A \leq_m B$ and $B$ is (semi) decidable, then $A$ is (semi) decidable

Proof: Let $M$ (semi) decide $B$ and let $f$ be a reduction from $A$ to $B$

We build a machine $N$ that (semi) decides $A$ as follows:

On input $w$:

1. Compute $f(w)$
2. Run $M$ on $f(w)$
All undecidability proofs from today can be seen as constructing an $f$ that reduces $A_{TM}$ to the proper language.

(Sometimes you have to consider the complement of the language.)
All undecidability proofs from today can be seen as constructing an $f$ that reduces $A_{TM}$ to the proper language $A_{TM} \leq_m HALLT_{TM}$ (So also, $\neg A_{TM} \leq_m \neg HALLT_{TM}$):

Map $(M, w) \mapsto (M', w)$
where $M'(w) = M(w)$ if $M(w)$ accepts
loops otherwise

So $(M, w) \in A_{TM} \iff (M', w) \in HALLT_{TM}$
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

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

CLAIM: $A_{TM} \leq_m \neg E_{TM}$

CONSTRUCT $f : \Sigma^* \rightarrow \Sigma^*$

$$f : (M,w) \rightarrow M_w \text{ where } M_w(s) = M(w)$$
\[ A_{TM} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \} \]
\[ E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \} \]

**CLAIM:** \( A_{TM} \leq_m E_{TM} \) \( \neg A_{TM} \leq_m E_{TM} \)

**CONSTRUCT** \( f : \Sigma^* \rightarrow \Sigma^* \)

\[ f: (M, w) \rightarrow M_w \text{ where } M_w(s) = M(w) \]

So, \( M(w) \) accepts \( \iff \) \( L(M_w) \neq \emptyset \)

So, \( (M, w) \in A_{TM} \iff M_w \in \neg E_{TM} \)
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

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

CLAIM: $A_{TM} \leq_m \neg E_{TM}$

CONSTRUCT $f : \Sigma^* \rightarrow \Sigma^*$

So, $M(w)$ accepts $\iff L(M_w) \neq \emptyset$

So, $(M, w) \in A_{TM} \iff M_w \in \neg E_{TM}$

So $\neg E_{TM}$ is NOT DECIDABLE, but it is SEMI-DECIDABLE (why?) Is $E_{TM}$ SEMI-DECIDABLE?
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

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

CLAIM: $A_{TM} \leq_m REG_{TM}$ So $REG_{TM}$ is UNDECIDABLE

CONSTRUCT $f : \Sigma^* \rightarrow \Sigma^*$

$f: (M,w) \rightarrow M'_{w}$ where $M'_{w}(s) = \text{accept if } s = 0^n1^n$ $M(w)$ otherwise
Claim: $A_{TM} \leq_m \text{REG}_{TM}$ So \text{REG}_{TM} is \text{UNDECIDABLE}

Construct $f: \Sigma^* \rightarrow \Sigma^*$

$f: (M,w) \rightarrow M'_w$ where $M'_w(s) = \text{accept}$ if $s = 0^n1^n$

So, $L(M'_w) = \Sigma^*$ if $M(w)$ accepts

$\{0^n1^n\}$ if not

So, $(M, w) \in A_{TM} \iff M'_w \in \text{REG}_{TM}$
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

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

**CLAIM:** \( A_{TM} \leq_m REG_{TM} \)  So \( REG_{TM} \) is UNDECIDABLE

**CONSTRUCT** \( f : \Sigma^* \rightarrow \Sigma^* \)

\[
f: (M,w) \rightarrow M'_w \quad \text{where} \quad M'_w(s) = \text{accept if } s = 0^n1^n \]
\[
M(w) \text{ otherwise}
\]

So, \( L(M'_w) = \Sigma^* \) if \( M(w) \) accepts
\[ \{0^n1^n\} \text{ if not} \]

So, \( (M, w) \in A_{TM} \iff M'_w \in REG_{TM} \)

Is \( REG \) SEMI-DECIDABLE? (\( \neg \) \( REG \) is not. Why?)
\[ \mathcal{A}_TM = \{ (M, w) \mid M \text{ is a TM that accepts string } w \} \]

\[ \mathcal{R}_TM = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \} \]

CLAIM: \( \neg \mathcal{A}_TM \leq_m \mathcal{R}_TM \)  So \( \mathcal{R}_TM \) is NOT SEMI-DECIDABLE

CONSTRUCT \( f : \Sigma^* \rightarrow \Sigma^* \)

\[
f: (M, w) \rightarrow M''_w \quad \text{where} \quad M''_w (s) = \text{accept if } s = 0^n1^n \text{ and } M(w) \text{ accepts Loop otherwise}
\]
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

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

CLAIM: $\neg A_{TM} \leq_m REG_{TM}$ So $REG_{TM}$ is NOT SEMI-DECIDABLE

CONSTRUCT $f : \Sigma^* \rightarrow \Sigma^*$

$f: (M,w) \rightarrow M''_w \text{ where } M''_w (s) = \begin{cases} \text{accept} & \text{if } s = 0^n1^n \\ \text{Loop} & \text{otherwise} \end{cases}$

So, $L(M'_w) = \{0^n1^n\} \text{ if } M(w) \text{ accepts}$

$\emptyset \text{ if not}$

So, $(M, w) \not\in A_{TM} \iff M''_w \in REG_{TM}$
$A_{TM} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \}$

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

**CLAIM:** $\neg A_{TM} \leq_m REG_{TM}$  So $REG_{TM}$ is NOT SEMI-DECIDABLE

**CONSTRUCT** $f : \Sigma^* \rightarrow \Sigma^*$

$f : (M, w) \rightarrow M''_w$ where $M''_w (s) = \text{accept if } s = 0^n1^n$ and $M(w)$ accepts

Loop otherwise

So, $L (M'_w) = \{0^n1^n\}$ if $M(w)$ accepts

$\emptyset$ if not

So, $(M, w) \notin A_{TM} \iff M''_w \in REG_{TM}$

**So,** $REG$ NOT SEMI-DECIDABLE
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

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

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

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

**ALL UNDECIDABLE**

Which are **SEMI-DECIDABLE**?
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

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

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

\[ \text{REG}_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \} \]

\[ \text{EQ}_{TM} = \{ (M, N) \mid M, N \text{ are TMs and } L(M) = L(N) \} \]

\[ \text{ALL}_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \} \]

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?
\[ E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \} \]

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

CLAIM: \( E_{TM} \leq_m EQ_{TM} \) \quad So \( EQ_{TM} \) is UNDECIDABLE

CONSTRUCT \( f : \Sigma^* \rightarrow \Sigma^* \)

\[
f : M \rightarrow (M, M \emptyset) \quad \text{where } M \emptyset (s) = \text{Loops}
\]

So, \( M \in E_{TM} \iff (M, M \emptyset) \in EQ_{TM} \)
\[ E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \} \]

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

CLAIM: \( E_{TM} \leq_m EQ_{TM} \) So \( EQ_{TM} \) is UNDECIDABLE

CONSTRUCT \( f : \Sigma^* \rightarrow \Sigma^* \)

\[
f : M \rightarrow (M, M_\emptyset) \text{ where } M_\emptyset(s) = \text{Loops}
\]

So, \( M \in E_{TM} \iff (M, M_\emptyset) \in EQ_{TM} \)

Is \( EQ_{TM} \) SEMI-DECIDABLE? NO, since,

\[ \neg A_{TM} \leq_m E_{TM} \leq_m EQ_{TM} \]

What about \( \neg EQ_{TM} \)?
\[ A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

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

CLAIM: \( A_{TM} \leq_m EQ_{TM} \)

So \( \neg EQ_{TM} \) is not semi-decidable
\( A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \)

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

CLAIM: \( A_{TM} \leq_m EQ_{TM} \)

So \( \neg EQ_{TM} \) is not semi-decidable

CONSTRUCT \( f : \Sigma^* \rightarrow \Sigma^* \)

\[ f : (M, w) \rightarrow (M_w, M_A) \]

Where for each \( s \) in \( \Sigma^* \),

\( M_w(s) = M(w) \) and \( M_A(s) \) always accepts
$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$

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

**CLAIM:** $A_{TM} \leq_m EQ_{TM}$

So $\neg EQ_{TM}$ is not semi-decidable

**CONSTRUCT** $f : \Sigma^* \rightarrow \Sigma^*$

$f : (M, w) \rightarrow (M_w, M_A)$

Where for each $s$ in $\Sigma^*$,

$M_w(s) = M(w)$ and $M_A(s)$ always accepts

So, $(M, w) \in A_{TM} \iff (M_w, M_A) \in EQ_{TM}$
\[ A_{TM} \leq m \rightarrow E_{TM} \]

\[ A_{TM} \leq m \rightarrow \text{REG}_{TM} \]

\[ A_{TM} \leq m \rightarrow \neg \text{REG}_{TM} \]

\[ E_{TM} \leq m \rightarrow \text{EQ}_{TM} \]

So, \( \neg A_{TM} \leq m \rightarrow \text{EQ}_{TM} \)

Also, \( A_{TM} \leq m \rightarrow \text{EQ}_{TM} \)
Undecidable given a TM to tell if the language it recognizes is empty. It’s not even semi-decidable, altho it is semi-decidable to tell if the language is non-empty.

\[ A_{TM} \leq_m \neg E_{TM} \]

\[ A_{TM} \leq_m \neg E_{TM} \]

\[ A_{TM} \leq_m \neg E_{TM} \]

\[ E_{TM} \leq_m \neg E_{TM} \]

So, \( \neg A_{TM} \leq_m \neg E_{TM} \)

Also, \( A_{TM} \leq_m \neg E_{TM} \)
Undecidable given a TM to tell if the language it recognizes is empty. It’s not even semi-decidable, altho it is semi-decidable to tell if the language is non-empty.

Undecidable given a TM to tell if it is equivalent to a FSM. It’s not even semi-decidable, nor is it semi-decidable to tell if it is not equivalent to a FSM.

So, $\neg A_{TM} \leq_m E_{Q_{TM}}$

Also, $A_{TM} \leq_m E_{Q_{TM}}$
Undecidable given a TM to tell if the language it recognizes is empty. It’s *not even semi-decidable*, *altho* it is semi-decidable to tell if the language is non-empty.

Undecidable given a TM to tell if it is equivalent to a FSM. It’s *not even semi-decidable*, *nor* is it semi-decidable to tell if it is not equivalent to a FSM.

Undecidable given 2 TMs to tell if they are equivalent. It’s *not even semi-decidable*, *nor* is it semi-decidable to tell if they are not.
\[ A_{\text{TM}} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \]

\[ \text{ALL}_{\text{PDA}} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \} \]

CLAIM: \( A_{\text{TM}} \leq_m \neg \text{ALL}_{\text{PDA}} \)

\[ \neg A_{\text{TM}} \leq_m \text{ALL}_{\text{PDA}} \]
\( A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \)

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

**CLAIM:** \( A_{TM} \leq_m \neg \text{ALL}_{PDA} \)

**CONSTRUCT** \( f : \Sigma^* \rightarrow \Sigma^* \)

**Idea!** More subtle construction
\[ A_{\text{TM}} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \} \]

\[ \text{ALL}_{\text{PDA}} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \} \]

Claim: \( A_{\text{TM}} \leq_m \neg \text{ALL}_{\text{PDA}} \)

Construct \( f : \Sigma^* \rightarrow \Sigma^* \)

Idea! More subtle construction

Map \((M, w)\) to a PDA \(P_{M,w}\) that recognizes \(\Sigma^*\) if and only if \(M\) does not accept \(w\)

So, \((M, w) \notin A_{\text{TM}} \iff P_{M,w} \in \text{ALL}_{\text{PDA}}\)
\( A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \} \)

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

CLAIM: \( A_{TM} \leq_m \neg \text{ALL}_{PDA} \)

CONSTRUCT \( f : \Sigma^* \rightarrow \Sigma^* \)

Idea! More subtle construction

Map \((M,w)\) to a PDA \( P_{M,w} \) that recognizes \( \Sigma^* \)
if and only if \( M \) does not accept \( w \)

So, \((M, w) \notin A_{TM} \iff P_{M,w} \in \text{ALL}_{PDA} \)

\( P_{M,w} \) will recognize all (and only those) strings that are NOT accepting computation histories for \( M \) on \( w \)
CONFIGURATIONS

11010q_7001110
An accepting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. $C_1$ is the start configuration,
2. $C_k$ is an accepting configuration,
3. Each $C_i$ follows from $C_{i-1}$
An accepting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. $C_1$ is the start configuration,
2. $C_k$ is an accepting configuration,
3. Each $C_i$ follows from $C_{i-1}$

An rejecting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. $C_1$ is the start configuration,
2. $C_k$ is a rejecting configuration,
3. Each $C_i$ follows from $C_{i-1}$
COMPUTATION HISTORIES

An accepting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. $C_1$ is the start configuration,
2. $C_k$ is an accepting configuration,
3. Each $C_i$ follows from $C_{i-1}$

An rejecting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

1. $C_1$ is the start configuration,
2. $C_k$ is a rejecting configuration,
3. Each $C_i$ follows from $C_{i-1}$

M accepts $w$ if and only if there exists an accepting computation history that starts with $C_1=q_0w$.
$P_{M,w}$ will recognize all strings (read as sequences of configurations) that:

1. Do not start with $C_1$ ($= q_0 w$) or
2. Do not end with an accepting configuration or
3. Where some $C_i$ does not properly yield $C_{i+1}$

Non-deterministic checks for 1, 2, and 3.
$P_{M,w}$ will reject all strings (read as sequences of configurations) that:

1. Start with $C_1 ( = q_0 w)$ and
2. End with an accepting configuration and
3. Where each $C_i$ properly yields $C_{i+1}$

Non-deterministic checks for 1, 2, and 3.
\{0^{2^n} | n \geq 0\}

- $q_0 \rightarrow q_1$ with $0 \rightarrow \square$, $R$
- $q_1 \rightarrow q_3$ with $0 \rightarrow x$, $R$
- $q_3 \rightarrow q_2$ with $x \rightarrow x$, $R$
- $q_2 \rightarrow q_0$ with $x \rightarrow x$, $R$
- $q_0 \rightarrow q_{reject}$ with $x \rightarrow x$, $R$
- $q_{reject} \rightarrow q_{accept}$ with $\square \rightarrow \square$, $R$
- $q_{accept} \rightarrow q_4$ with $0 \rightarrow 0$, $R$
- $q_4 \rightarrow q_{accept}$ with $x \rightarrow x$, $R$
- $q_{accept} \rightarrow q_2$ with $\square \rightarrow \square$, $R$
- $q_2 \rightarrow q_{accept}$ with $x \rightarrow x$, $R$
- $q_{accept} \rightarrow q_0$ with $x \rightarrow x$, $R$
- $q_0 \rightarrow q_{accept}$ with $x \rightarrow x$, $R$
- $q_{accept} \rightarrow q_2$ with $\square \rightarrow \square$, $R$
- $q_2 \rightarrow q_{accept}$ with $x \rightarrow x$, $R$
- $q_{accept} \rightarrow q_0$ with $x \rightarrow x$, $R$
- $q_0 \rightarrow q_{accept}$ with $x \rightarrow x$, $R$

Input strings:
- $q_0 \overline{0000}$
- $\square q_1 \overline{000}$
- $\square xq_3 \overline{00}$
- $\square x0q_4 \overline{0}$
- $\square x0xq_3$
- $\square x0q_2 \overline{x}$
- $\square xq_2 \overline{0}x$
- $\square x2 \overline{x0}$
- $q_2 \overline{\square x0}$
$P_{M,w}$ recognizes all strings except “accepting computation histories”:

#$C_1$ # $C_2^R$ # $C_3$ # $C_4^R$ # $C_5$ # $C_6^R$ # .... # $C_k$
If \( i \) is odd, put \( C_i \) on stack and see if \( C_{i+1} \) follows properly:

For example,

If \( 0aqq_i bqv \) and \( \delta (q_i, b) = (q_j, c, R) \),

then \( C_i \) properly yields \( C_{i+1} \iff C_{i+1} = uacq_j v \).
\( P_{M.w} \) recognizes all strings except "accepting computation histories":

\[
\#C_1# \ C_2^R \ #C_3 \ #C_4^R \ #C_5 \ #C_6^R \ #\ldots# \ C_k
\]

If \( i \) is odd, put \( C_i \) on stack and see if \( C_{i+1}^R \) follows properly.

For example,

If \( =\textcolor{red}{u}a\textcolor{red}{q}_i\textcolor{red}{b}v \) and \( \delta (q_i, b) = (q_j, c, L) \),
then \( C_k \) properly yields \( C_{k+1} \iff C_{k+1} = uq_jadv \).
$P_{M.w}$ recognizes all strings except “accepting computation histories”:

$#C_1# \ C_2^R \ #C_3 \ #C_4^R \ #C_5 \ #C_6^R \ #\ldots\ldots# \ C_k$

If $i$ is even, put $C_i^R$ on stack and see if $C_{i+1}$ follows properly.
Odd
EVEN

#q₀0000#000q₁#q₀q₃00#0q₄0x #q₀#q₀x0q₃# ... #
$$A_{TM} = \{ (M,w) \mid M \text{ is a TM that accepts string } w \}$$

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

**CLAIM:** $$A_{TM} \leq_m \neg \text{ALL}_{PDA}$$

**CONSTRUCT** $$f : \Sigma^* \to \Sigma^*$$ where

$$f: (M,w) \to P_{M,w}$$

$$P_{M,w} (s) = \text{accept iff } s \text{ is NOT an accepting computation of } M(w)$$

So, $$(M, w) \notin A_{TM} \iff P_{M,w} \in \text{ALL}_{PDA}$$

So, $$(M, w) \in A_{TM} \iff P_{M,w} \notin \neg \text{ALL}_{PDA}$$

**EXPLAIN THE PROOF TO YOUR NEIGHBOR**
\[ A_{TM} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \} \]
\[ HALT_{TM} = \{ (M, w) \mid M \text{ is a TM that halts on string } w \} \]
\[ E_{TM} = \{ M \mid M \text{ is a TM and } L(M) = \emptyset \} \]
\[ REG_{TM} = \{ M \mid M \text{ is a TM and } L(M) \text{ is regular} \} \]
\[ EQ_{TM} = \{ (M, N) \mid M, N \text{ are TMs and } L(M) = L(N) \} \]
\[ ALL_{PDA} = \{ P \mid P \text{ is a PDA and } L(P) = \Sigma^* \} \]

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?

What about complements?
Read chapter 5.1-5.3 of the book for next time