15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

UNDECIDABILITY II: REDUCTIONS

TUESDAY Feb 18

A_{TM} = { (M,w) | M 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)) = { Accept if M accepts w; Rejects or loops otherwise }

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) =

- Reject if D_H accepts D_H
  (i.e. if H(D_H , D_H) = Accept)
- Accept if D rejects D_H
  (i.e. if H(D_H , D_H) = Reject)
- loops if D_H loops on D_H
  (i.e. if H(D_H , D_H) loops)

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} “<“ L

A_{TM} = { (M,w) | M is a TM that accepts string w }

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

E_{TM} = { M | M is a TM and L(M) = ∅ } (*)

REG_{TM} = { M | M is a TM and L(M) is regular} (*)

EQ_{TM} = {(M, N) | M, N are TMs and L(M) = L(N)} (*)

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* } (*)

ALL UNDECIDABLE

(*) Use Reductions to Prove

Which are SEMI-DECIDABLE?

THE HALTING PROBLEM

$HALT_{TM}$ = { (M,w) | M 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}

THE HALTING PROBLEM

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

Theorem: $HALT_{TM}$ is undecidable

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

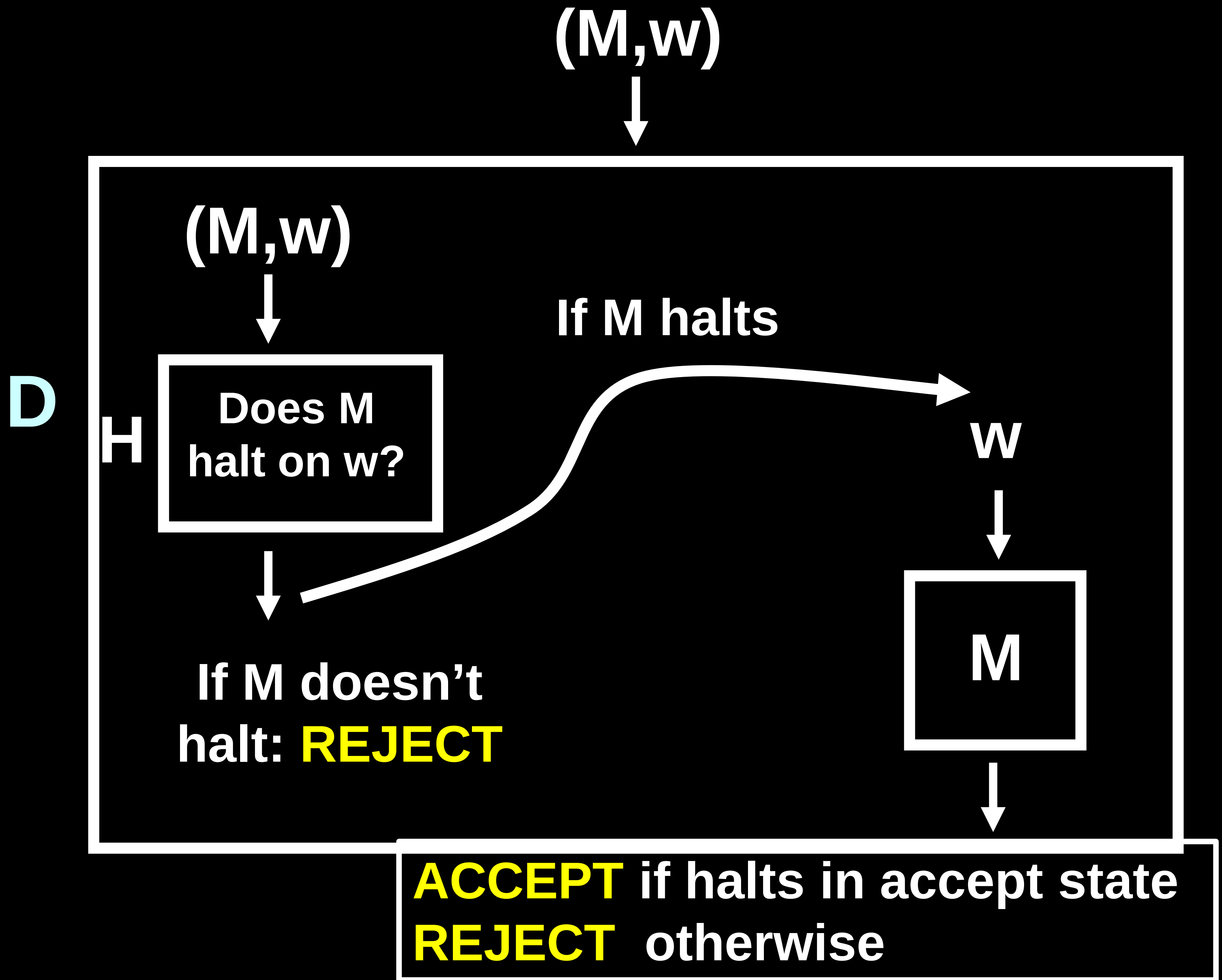
We use H to construct a TM D that decides A_{TM}

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



(M,w)
D
(M,w)
H
Does M
halt on w?
If M halts
w
M
If M doesn't
halt: REJECT
ACCEPT if halts in accept state
REJECT otherwise

E_{TM} = { M | M is a TM and L(M) = ∅ }

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}

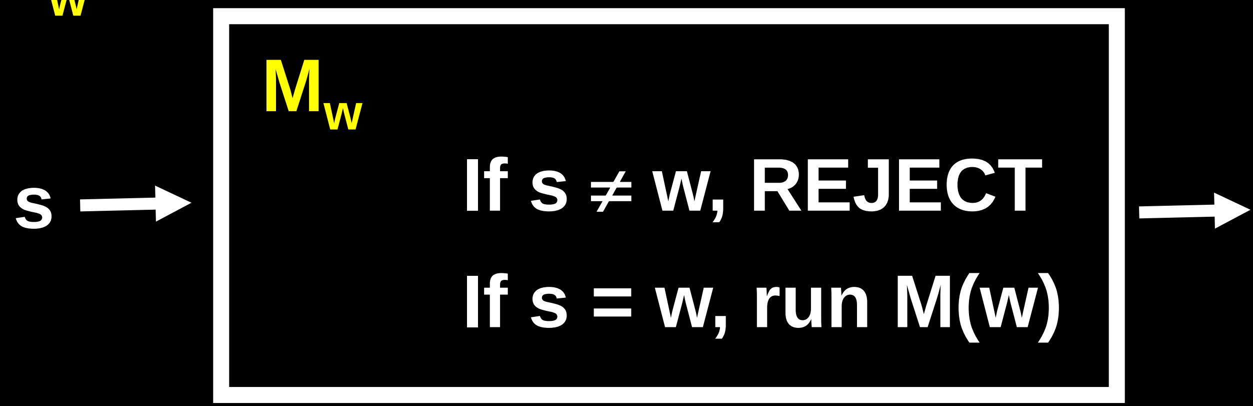
E_{TM} = { M | M is a TM and L(M) = ∅ }

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):

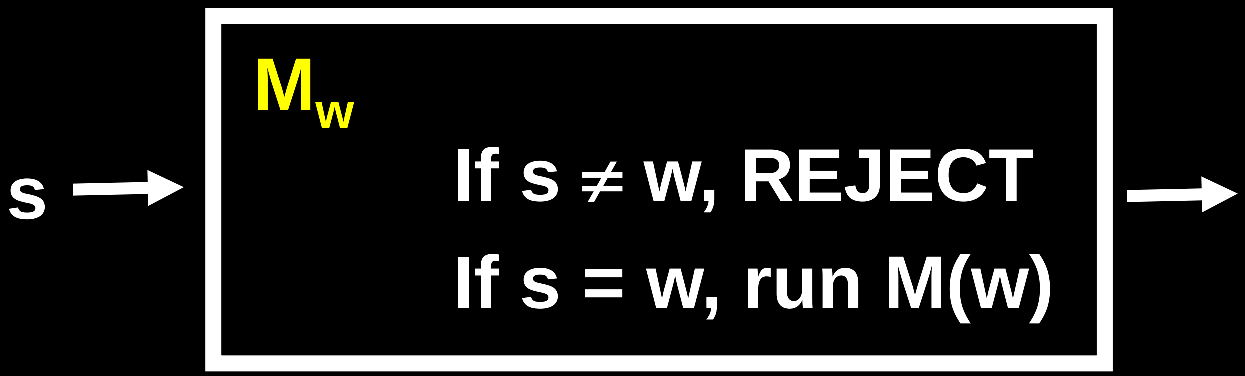
1. Create M_w



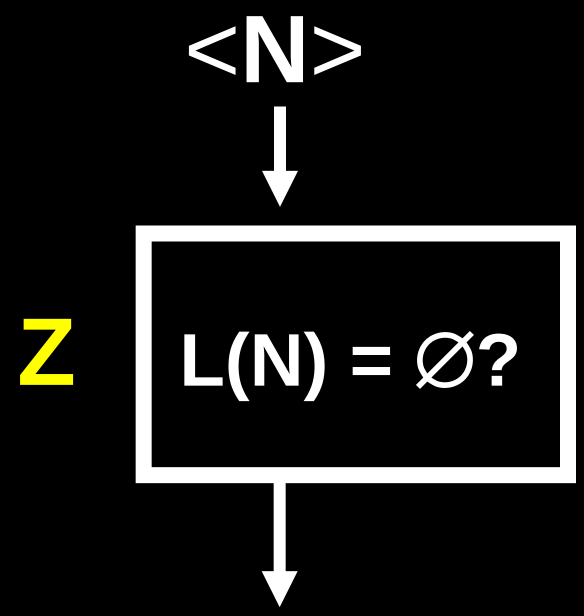
So, L (M_w) = ∅ ⇔ M(w) does not accept

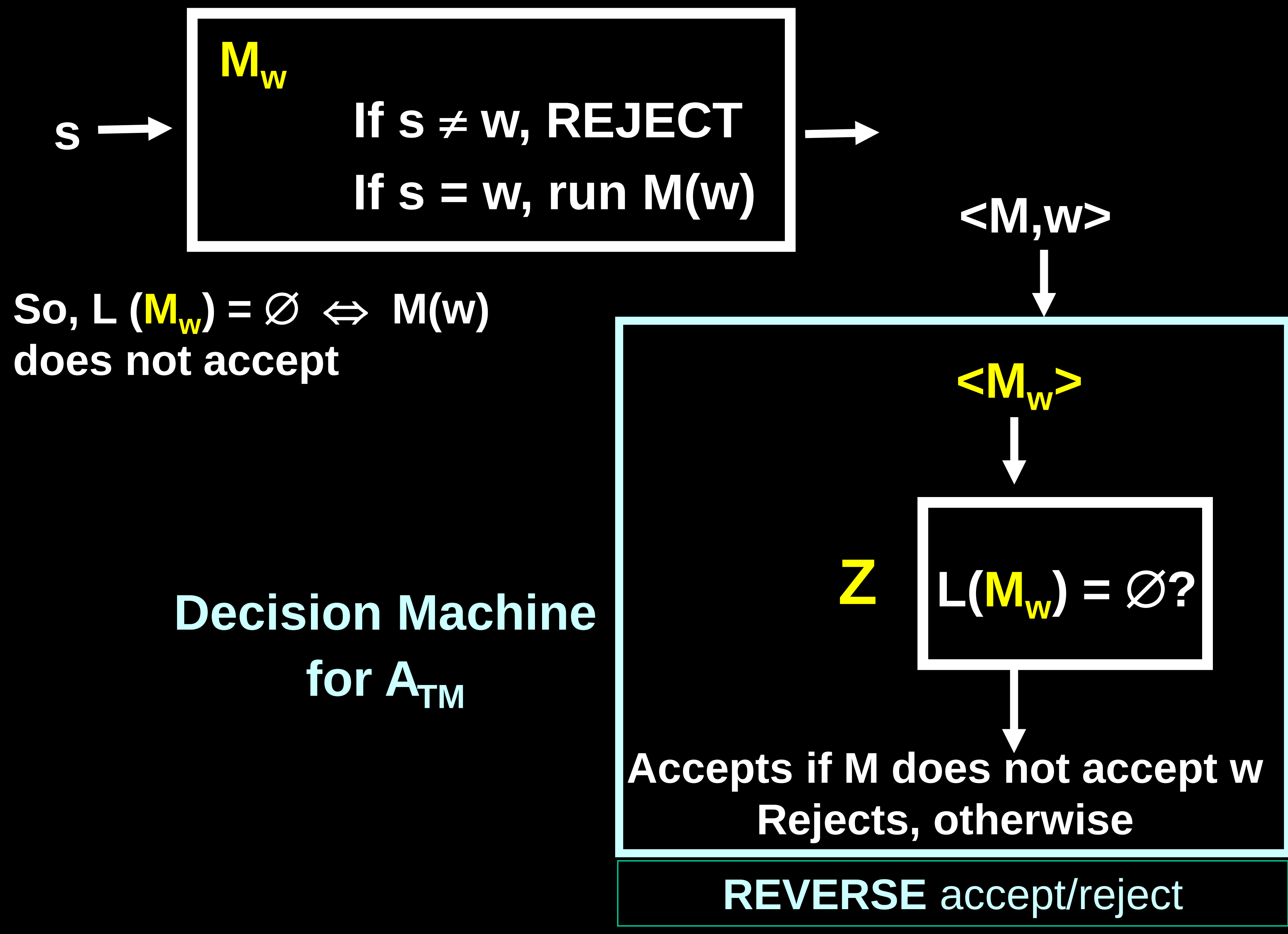
L(M_w) ≠ ∅ ⇔ M(w) accepts

2. Run Z on M_w



So, $L(M_w) = \varnothing \Leftrightarrow$ M(w) does not accept





s
M_w
If s ≠ w, REJECT
If s = w, run M(w)
<M,w>
So, L (M_w) = ∅ ⇔ M(w) does not accept
<M_w>
Z
L(M_w) = ∅?
Decision Machine for A_{TM}
Accepts if M does not accept w
Rejects, otherwise
REVERSE accept/reject

$REGULAR_{TM}$ = { M | M is a TM and L(M) is regular}

Theorem: $REGULAR_{TM}$ is undecidable

Proof: Assume, for a contradiction, that TM R decides $REGULAR_{TM}$

Use **R** as a subroutine to decide **A_{TM}**

$REGULAR_{TM}$ = { M | M is a TM and L(M) is regular}

Theorem: $REGULAR_{TM}$ is undecidable

Proof: Assume, for a contradiction, that TM **R** decides $REGULAR_{TM}$

Use **R** as a subroutine to decide A_{TM}

1. Create M'_w

M'_w (input: s)

If $s = 0^n1^n$, accept

Else run M(w)

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

$L(M'_w) = \{0^n1^n\}$ $\Leftrightarrow$ M(w) does not accept

2. Run **R** on M'_w

M_w'

s → [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 ⇔ M(w) accepts

<N>

↓

R [Is L(N) regular?]

↓

$s \rightarrow$

M_w'

If $s = 0^n1^n$, accept

Else run M(w)

$\rightarrow$

L(M_w') = Σ^* if M(w) accepts

$\{0^n1^n\}$ otherwise

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

< M_w'>

$\downarrow$

R Is L(M_w') **regular?**

$\downarrow$

Yes $\Leftrightarrow$ M accepts w

MAPPING REDUCIBILITY

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

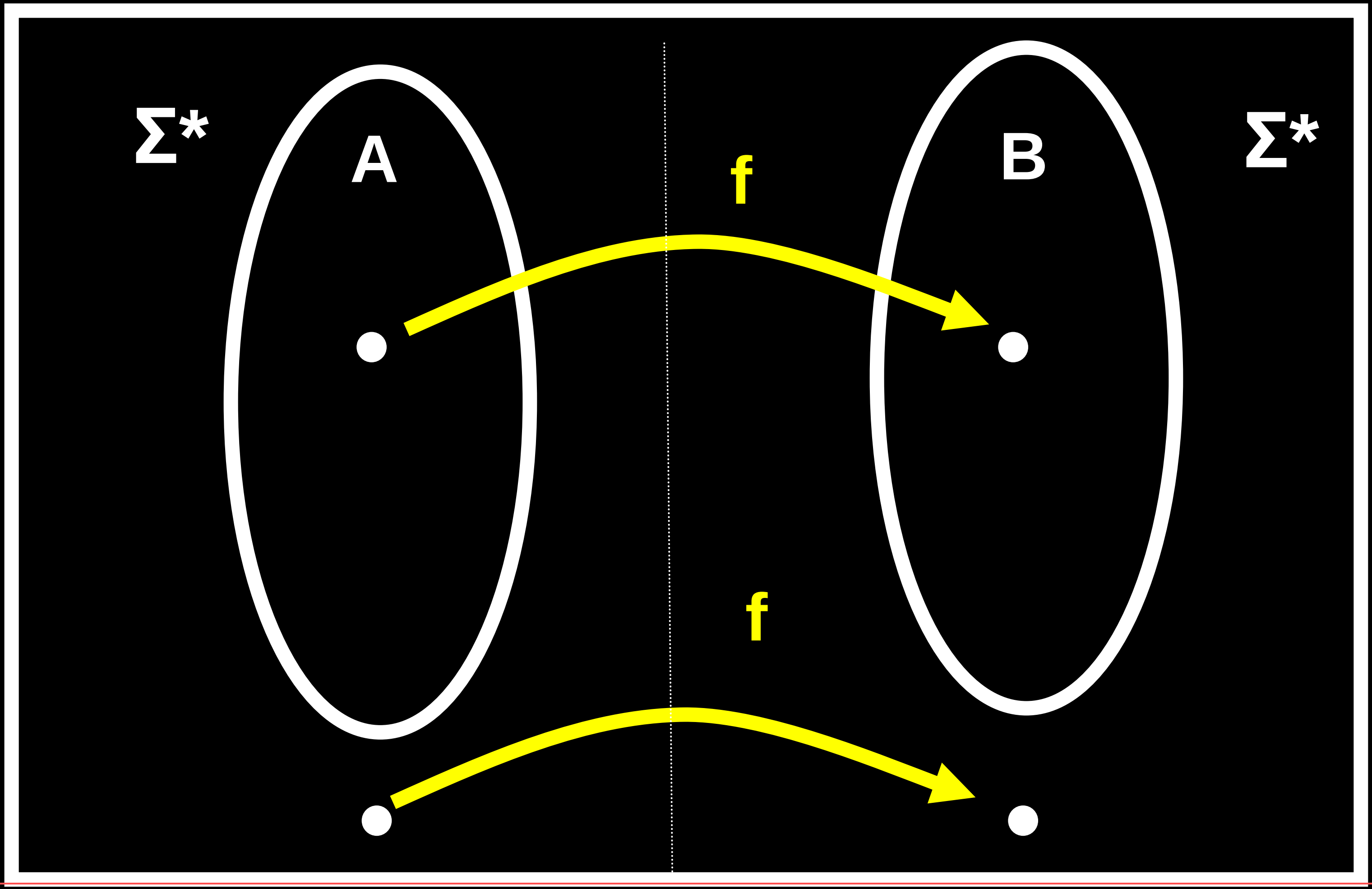
$f : \Sigma^* \rightarrow \Sigma^*$, where for every w,

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

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 $f : \Sigma^* \to \Sigma^*$
such that $w \in A \Leftrightarrow f(w) \in B$



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

$A_{TM} \leq_m HALT_{TM}$ (So also, $\neg A_{TM} \leq_m \neg HALT_{TM}$):

Map $(M, w) \rightarrow (M', w)$

where $M'(w) = M(w)$ if $M(w)$ accepts

loops otherwise

So $(M, w) \in A_{TM} \Leftrightarrow (M', w) \in HALT_{TM}$

A_{TM} = { (M,w) | M is a TM that accepts string w }

E_{TM} = { M | M is a TM and L(M) = ∅ }

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

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

CONSTRUCT f : Σ* → Σ*

f: (M,w) → M_w where M_w (s) = M(w)

A_{TM} = { (M,w) | M is a TM that accepts string w }

E_{TM} = { M | M is a TM and L(M) = ∅ }

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

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

CONSTRUCT f : Σ* → Σ*

f: (M,w) → M_w where M_w (s) = M(w)

So, M(w) accepts ⇔ L (M_w) ≠ ∅

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

A_{TM} = { (M,w) | M is a TM that accepts string w }

E_{TM} = { M | M is a TM and L(M) = $\varnothing$ }

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

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

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

f: (M,w) $\rightarrow$ M_w where M_w (s) = M(w)

So, M(w) accepts $\Leftrightarrow$ L (M_w) $\neq \varnothing$

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

So $\neg E_{TM}$ is NOT DECIDABLE, but it is SEMI-DECIDABLE (why?) Is E_{TM} SEMI-DECIDABLE?

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) 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) = accept if $s = 0^n1^n$	
	M(w) otherwise

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) is regular}

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

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

f: (M,w) $\to$ M'_w where M'_w (s) = accept if s = 0^n1^n
M(w) otherwise

So, L (M'_w) = Σ^* if M(w) accepts
{0^n1^n} if not

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

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) is regular}

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

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

f: (M,w) → M'_w where M'_w (s) = accept if $s = 0^n1^n$
M(w) otherwise

So, L (M'_w) = Σ^* if M(w) accepts
$\{0^n1^n\}$ if not

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

Is REG SEMI-DECIDABLE? ($\neg$ REG is not. Why?)

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) 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) → M''_w where M''_w (s) = accept if $s = 0^n1^n$ *and* M(w) accepts

Loop otherwise

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) 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) → M"$_w$ where M"$_w$ (s) = accept if s = 0^n1^n *and* M(w) accepts
Loop otherwise

So, L (M'$_w$) = {0^n1^n} if M(w) accepts
∅ if not

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

A_{TM} = { (M,w) | M is a TM that accepts string w }

REG_{TM} = { M | M is a TM and L(M) is regular}

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

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

f: (M,w) → M"$_w$ where M"$_w$ (s) = accept if $s = 0^n1^n$ *and* M(w) accepts Loop otherwise

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

$\varnothing$ if not

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

So, REG NOT SEMI-DECIDABLE

A_{TM} = { (M,w) | M is a TM that accepts string w }

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

E_{TM} = { M | M is a TM and L(M) = ∅ }

REG_{TM} = { M | M is a TM and L(M) is regular}

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?

A_{TM} = { (M,w) | M is a TM that accepts string w }

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

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

REG_{TM} = { M | M is a TM and L(M) is regular}

EQ_{TM} = {(M, N) | M, N are TMs and L(M) = L(N)}

ALL_{PDA} = { P | P is a PDA and L(P) = Σ^* }

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?

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

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

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

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

$f : M \to (M, M_{\varnothing})$ where $M_{\varnothing}(s)$ = Loops

So, $M \in E_{TM} \Leftrightarrow (M, M_{\varnothing}) \in EQ_{TM}$

E_{TM} = { M | M is a TM and L(M) = ∅ }

EQ_{TM} = {(M, N) | M, N are TMs and L(M) =L(N)}

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

CONSTRUCT f : Σ* → Σ*

f : M → (M, $M_{\varnothing}$) where $M_{\varnothing}$(s) = Loops

So, M ∈ E_{TM} ⇔ (M, $M_{\varnothing}$) ∈ 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) | M is a TM that accepts string w }

EQ_{TM} = {(M, N) | M, N 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) | M is a TM that accepts string w }

EQ_{TM} = {(M, N) | M, N 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) → (M_w, M_A)

Where for each s in **Σ***,

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

A_{TM} = { (M,w) | M is a TM that accepts string w }

EQ_{TM} = {(M, N) | M, N are TMs and L(M) =L(N)}

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

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

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

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

Where for each s in Σ^*,

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

So, (M,w) $\in A_{TM} \Leftrightarrow (M_w, M_A) \in EQ_{TM}$

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

$A_{TM} \leq_m REG_{TM}$

$A_{TM} \leq_m \neg REG_{TM}$

$E_{TM} \leq_m EQ_{TM}$

So, $\neg A_{TM} \leq_m EQ_{TM}$

Also, $A_{TM} \leq_m EQ_{TM}$

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

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.

$A_{TM} \leq_m REG_{TM}$

$A_{TM} \leq_m \neg REG_{TM}$

$E_{TM} \leq_m EQ_{TM}$

So, $\neg A_{TM} \leq_m EQ_{TM}$

Also, $A_{TM} \leq_m EQ_{TM}$

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

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.

$A_{TM} \leq_m REG_{TM}$

$A_{TM} \leq_m \neg REG_{TM}$

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.

$E_{TM} \leq_m EQ_{TM}$

So, $\neg A_{TM} \leq_m EQ_{TM}$

Also, $A_{TM} \leq_m EQ_{TM}$

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

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.

$A_{TM} \leq_m REG_{TM}$

$A_{TM} \leq_m \neg REG_{TM}$

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.

$E_{TM} \leq_m EQ_{TM}$

So, $\neg A_{TM} \leq_m EQ_{TM}$

Also, $A_{TM} \leq_m EQ_{TM}$

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_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$

$\neg A_{TM} \leq_m ALL_{PDA}$

A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$

$\neg A_{TM} \leq_m ALL_{PDA}$

CONSTRUCT f : Σ* → Σ*

Idea!

More subtle construction

A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$

$\neg A_{TM} \leq_m ALL_{PDA}$

CONSTRUCT f : Σ* → Σ*

Idea!

More subtle construction

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

So, (M, w) $\notin A_{TM} \Leftrightarrow P_{M,w} \in ALL_{PDA}$

A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$

$\neg A_{TM} \leq_m ALL_{PDA}$

CONSTRUCT f : Σ* → Σ*

Idea!

More subtle construction

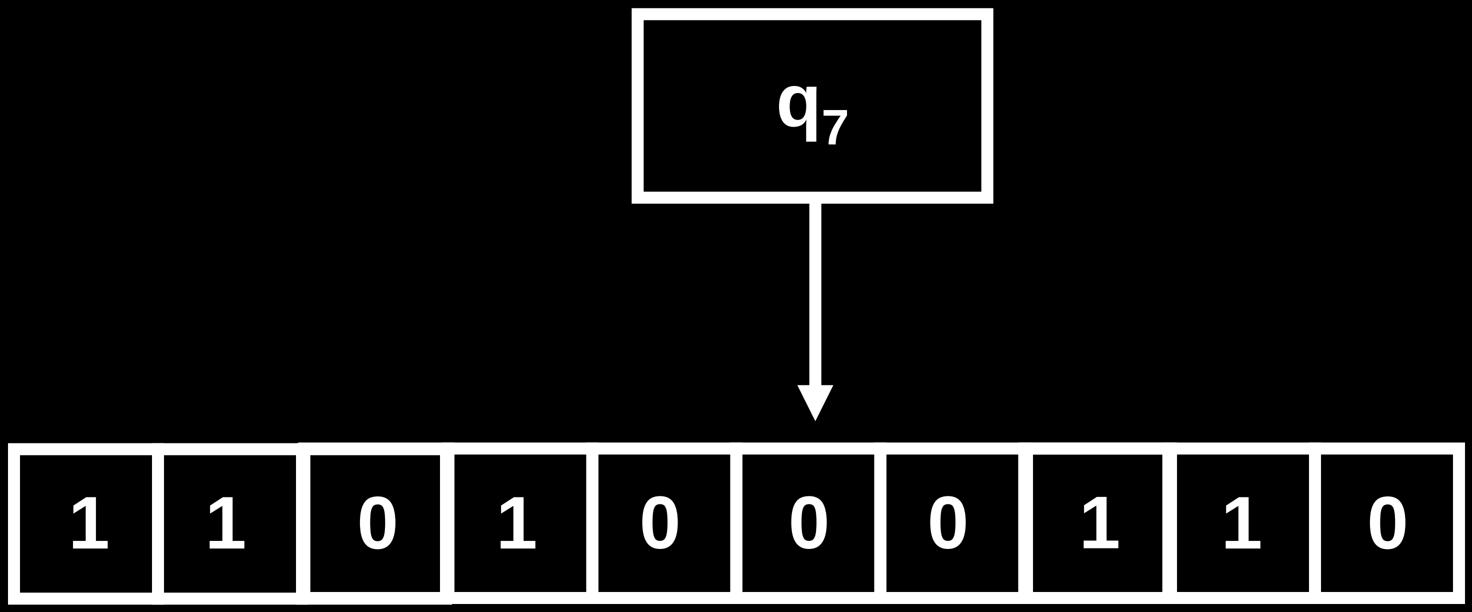
Map (M,w) to a PDA $P_{M,w}$ that recognizes Σ* if and only if M does not accept w

So, $(M, w) \notin A_{TM} \Leftrightarrow P_{M,w} \in ALL_{PDA}$

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

CONFIGURATIONS

11010$q_7$00110



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}**

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}**

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}**

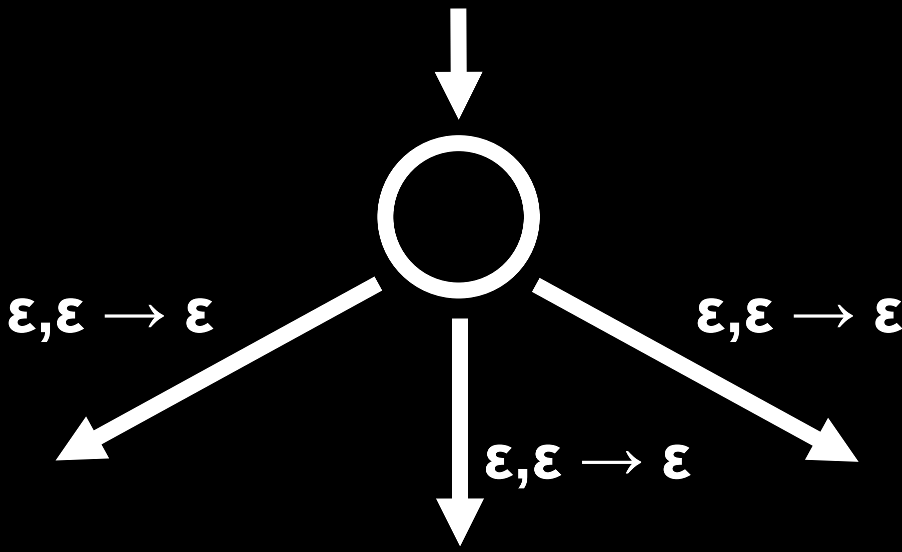
An rejecting computation history is a sequence of configurations $C_1, C_2, \ldots, C_k$, where

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 $C_1 = q_0 w$

$P_{M,w}$ will recognize all strings (read as sequences of configurations) that:

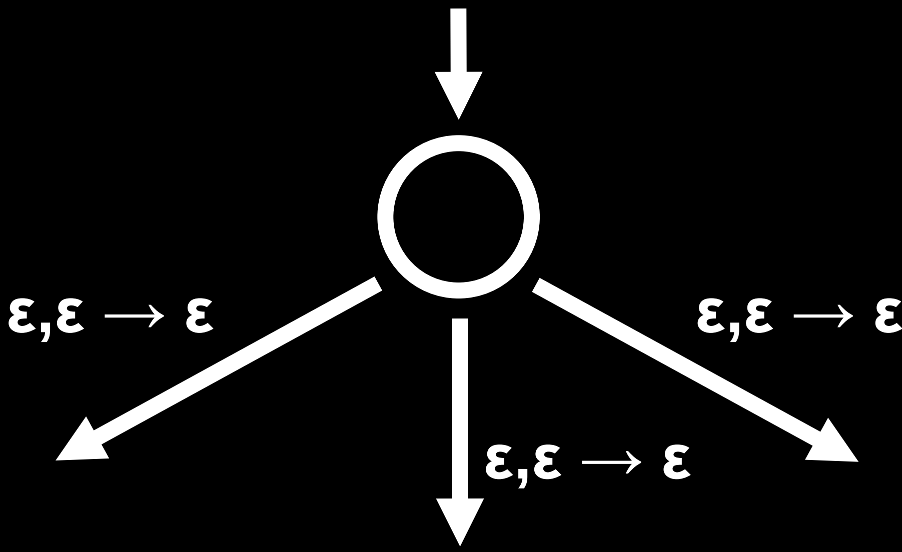
1. Do not start with C_1 (= q_0w) or
2. Do not end with an accepting configuration or
3. Where some C_i does not properly yield C_{i+1}



Non-deterministic checks for 1, 2, and 3.

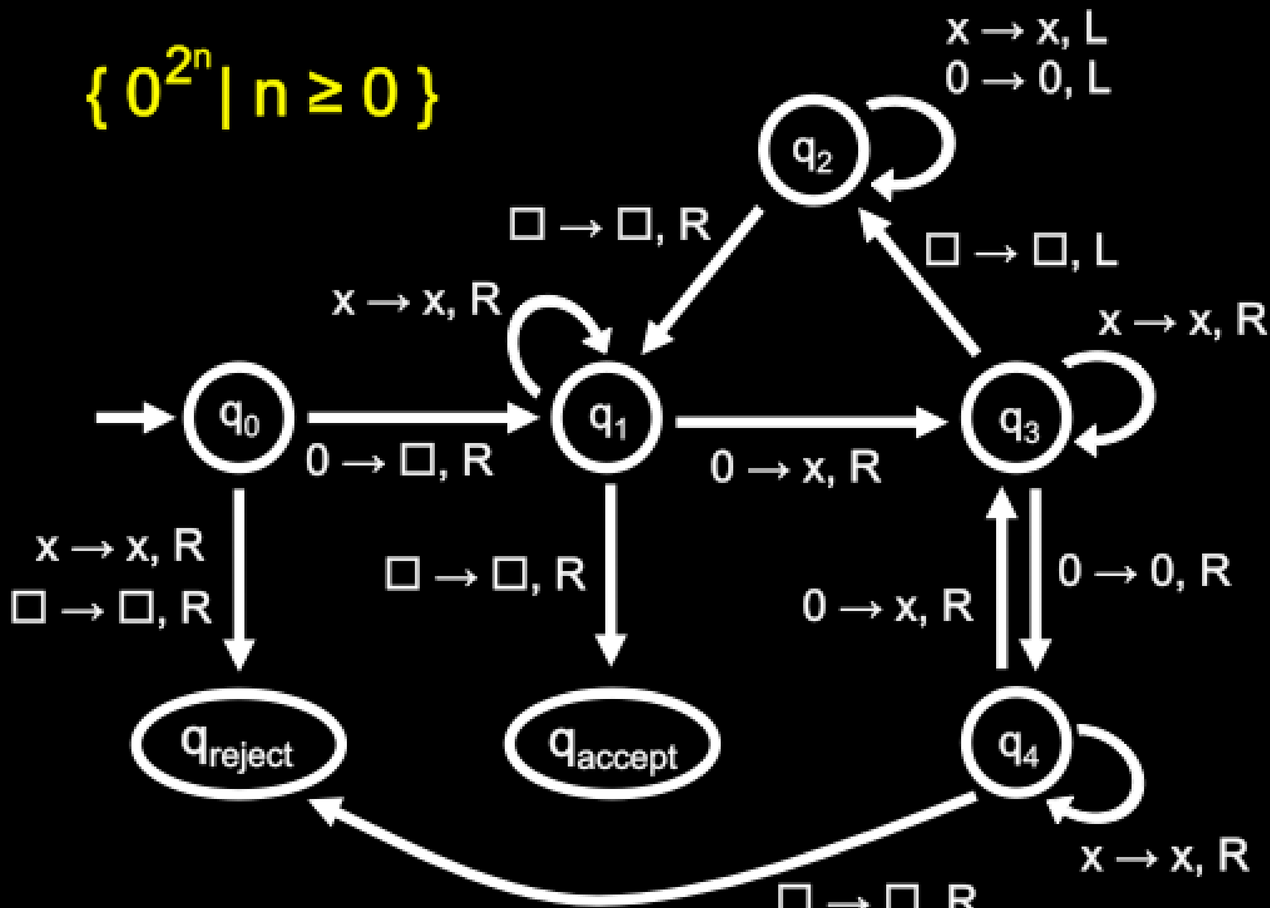
$P_{M,w}$ will reject all strings (read as sequences of configurations) that:

1. Start with C_1 (= q_0w) 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.

$\{ 0^{2^n} \mid n \geq 0 \}$



$q_0$0000

□$q_1$000

□x$q_3$00

□x0$q_4$0

□x0xq_3

□x0q_2x

□x$q_2$0x

□q_2x0x

q_2□x0x

$P_{M,w}$ recognizes all strings except "accepting computation histories" :

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

$P_{M.w}$ 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 $=uaq_ibv$ and $\delta(q_i,b) = (q_j,c,R)$,

then C_i properly yields $C_{i+1} \Leftrightarrow C_{i+1} = uacq_jv$

$P_{M.w}$ 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}^R follows properly.

For example,

If $=uaq_ibv$ and $\delta(q_i,b) = (q_j,c,L)$,

then C_k properly yields $C_{k+1} \Leftrightarrow C_{k+1} = uq_jacv$

$P_{M.w}$ 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 even**, put C_i^R on stack and see if C_{i+1} follows properly.

0	0	0	q_1	□

0
0
0
0
q_0

ODD

$q_0$0000

□$q_1$000

□x$q_3$00

□x0$q_4$0

□x0xq_3

□x0q_2x

□x$q_2$0x

□q_2x0x

⋮

#$q_0$0000#000q_1□#□x$q_3$00#0$q_4$0x □#□x0xq_3# ... #

0	0	0	q_1	$\square$

0
0
0
0
q_0

ODD

$q_0 0000$

$\square q_1 000$

$\square x q_3 00$

$\square x 0 q_4 0$

$\square x 0 x q_3$

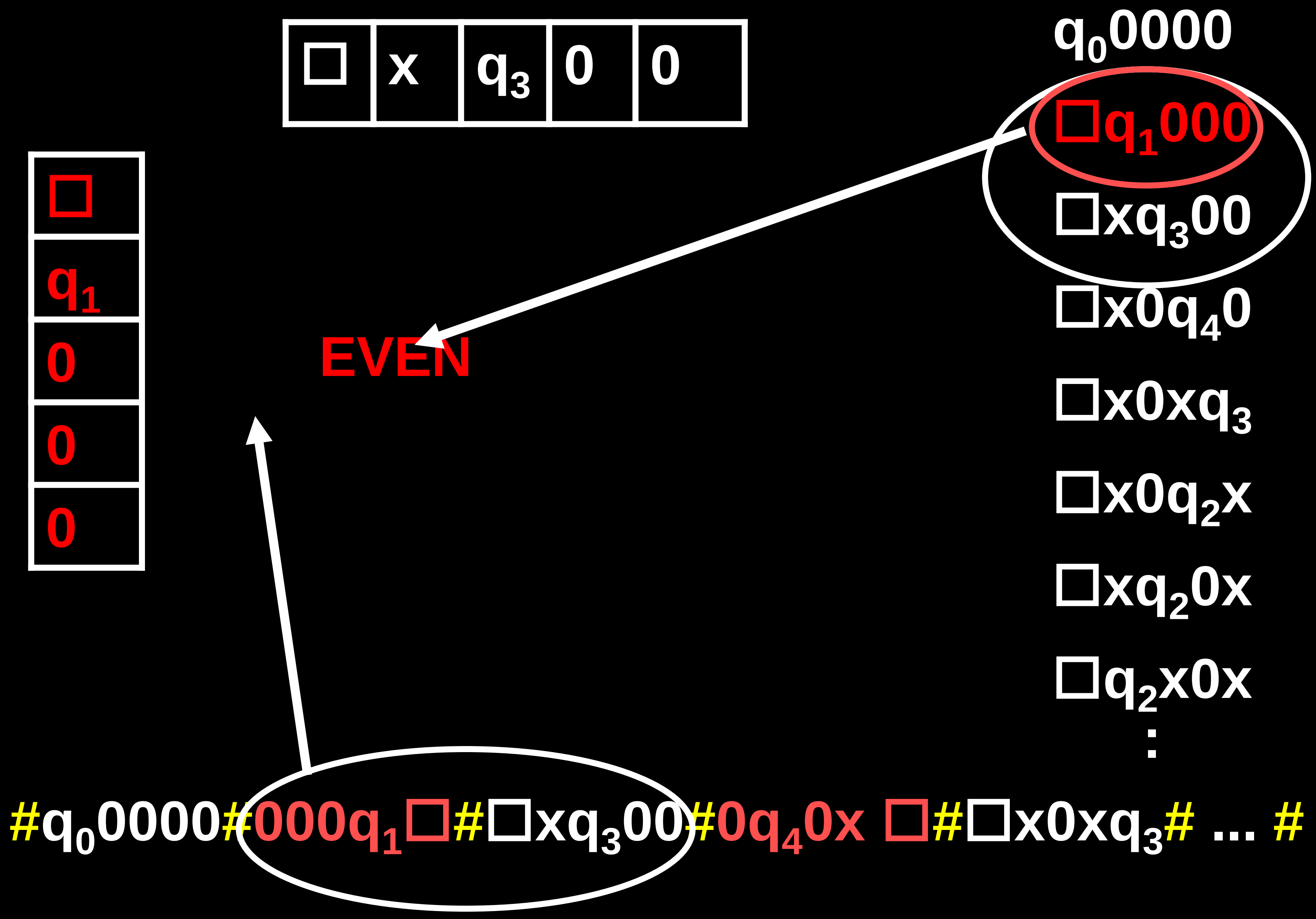
$\square x 0 q_2 x$

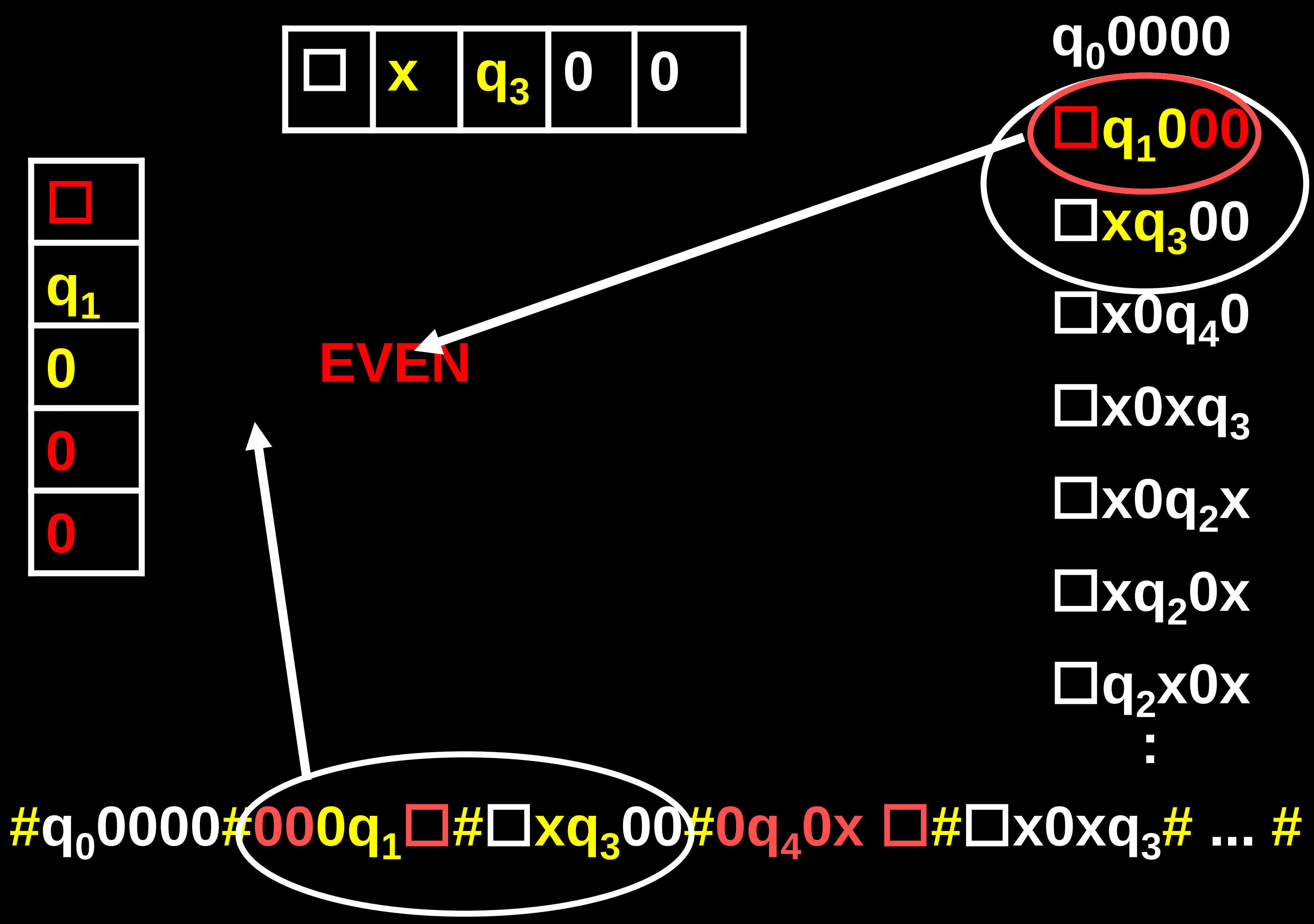
$\square x q_2 0 x$

$\square q_2 x 0 x$

⋮

$\# q_0 0000 \# 000 q_1 \square \# \square x q_3 00 \# 0 q_4 0x\ \square \# \square x 0 x q_3 \# \ldots \#$


□ x q3 0 0
□
q1
0
0
0
EVEN
q00000
□q1000
□xq300
□x0q40
□x0xq3
□x0q2x
□xq20x
□q2x0x
:
#q00000#000q1□#□xq300#0q40x □#□x0xq3# ... #


□ x q3 0 0
q0000
□q1000
□xq300
□x0q40
□x0xq3
□x0q2x
□xq20x
□q2x0x
:
□ q1 0 0 0
EVEN
#q00000#000q1□#□xq300#0q40x □#□x0xq3# ... #

A_{TM} = { (M,w) | M is a TM that accepts string w }

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

CLAIM: $A_{TM} \leq_m \neg ALL_{PDA}$

$\neg A_{TM} \leq_m ALL_{PDA}$

CONSTRUCT f : Σ* → Σ*

f: (M,w) → $P_{M,w}$ where

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

So, (M, w) ∉ A_{TM} ⇔ $P_{M,w}$ ∈ ALL_{PDA}

So, (M, w) ∈ A_{TM} ⇔ $P_{M,w}$ ∈ ¬ ALL_{PDA}

EXPLAIN THE PROOF TO YOUR NEIGHBOR

A_{TM} = { (M,w) | M is a TM that accepts string w }

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

E_{TM} = { M | M is a TM and L(M) = ∅ }

REG_{TM} = { M | M is a TM and L(M) is regular}

EQ_{TM} = {(M, N) | M, N are TMs and L(M) = L(N)}

ALL_{PDA} = { P | P is a PDA and L(P) = Σ* }

ALL UNDECIDABLE

Which are SEMI-DECIDABLE?

What about complements?

WWW.FLAC.WS

Read chapter 5.1-5.3 of the book for next time