15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

UNDECIDABLE PROBLEMS

THURSDAY Feb 13

Definition: **A Turing Machine is a 7-tuple**
$T = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

Q is a finite set of states

Σ is the input alphabet, where $\Box \notin \Sigma$

Γ is the tape alphabet, where $\Box \in \Gamma$ and $\Sigma \subseteq \Gamma$

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$

$q_0 \in Q$ is the start state

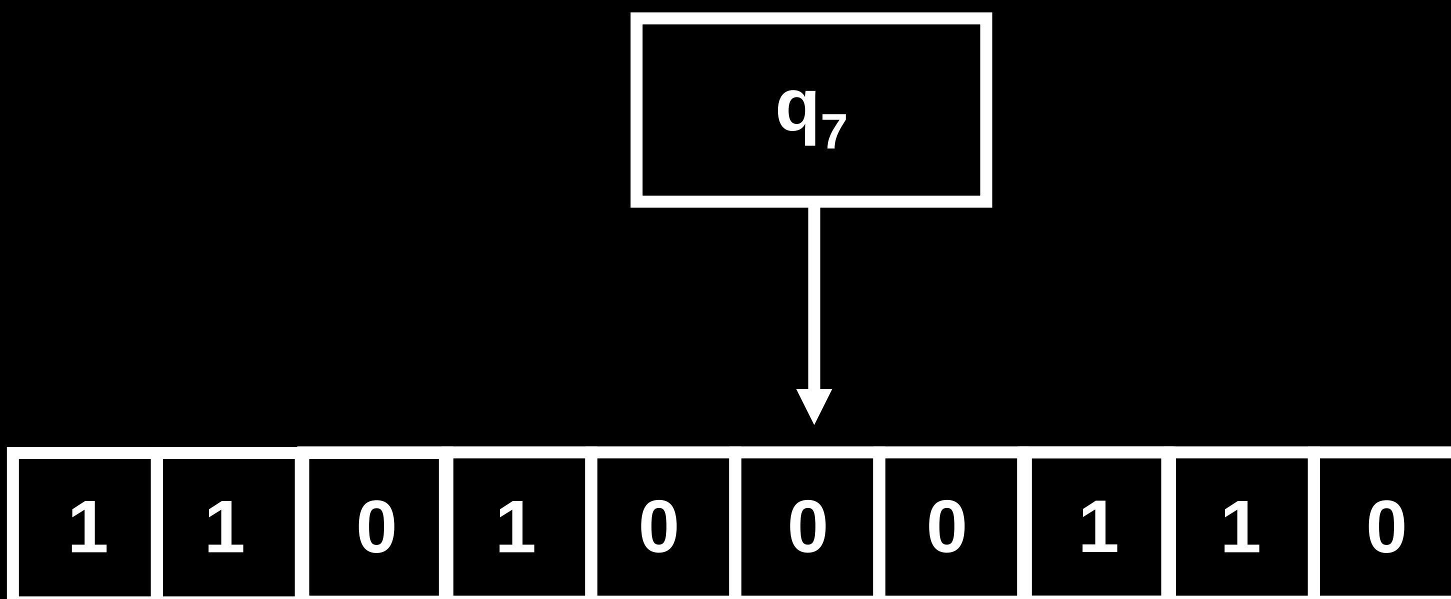
$q_{accept} \in Q$ is the accept state

$q_{reject} \in Q$ is the reject state, and $q_{reject} \neq q_{accept}$

CONFIGURATIONS

$$11010q_7 00110$$

corresponds to:



A Turing Machine M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that

1. C_1 is a *start* configuration of M on input w, ie C_1 is q_0w
2. each C_i *yields* C_{i+1}, ie M can legally go from C_i to C_{i+1} in a single step

ua q_i bv *yields* u q_j acv if $\delta(q_i, b) = (q_j, c, L)$

ua qi bv *yields* uac q_j v if $\delta(q_i, b) = (q_j, c, R)$

A Turing Machine M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that

1. C_1 is a *start* configuration of M on input w, ie C_1 is q_0w
2. each C_i *yields* C_{i+1}, ie M can legally go from C_i to C_{i+1} in a single step
3. C_k is an *accepting* configuration, ie the state of the configuration is q_{accept}

A Turing Machine M *rejects* input w if there is a sequence of configurations $C_1, \ldots, C_k$ such that

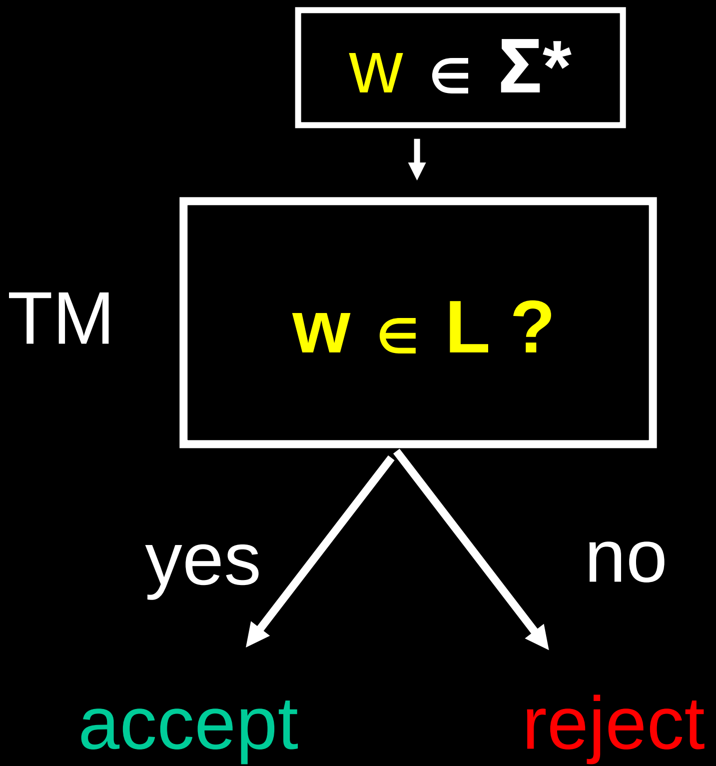
1. C_1 is a *start* configuration of M on input w, ie C_1 is q_0w
2. each C_i *yields* C_{i+1}, ie M can legally go from C_i to C_{i+1} in a single step
3. C_k is a *rejecting* configuration, ie the state of the configuration is q_{reject}

A TM **recognizes** a language if it accepts all and only those strings in the language

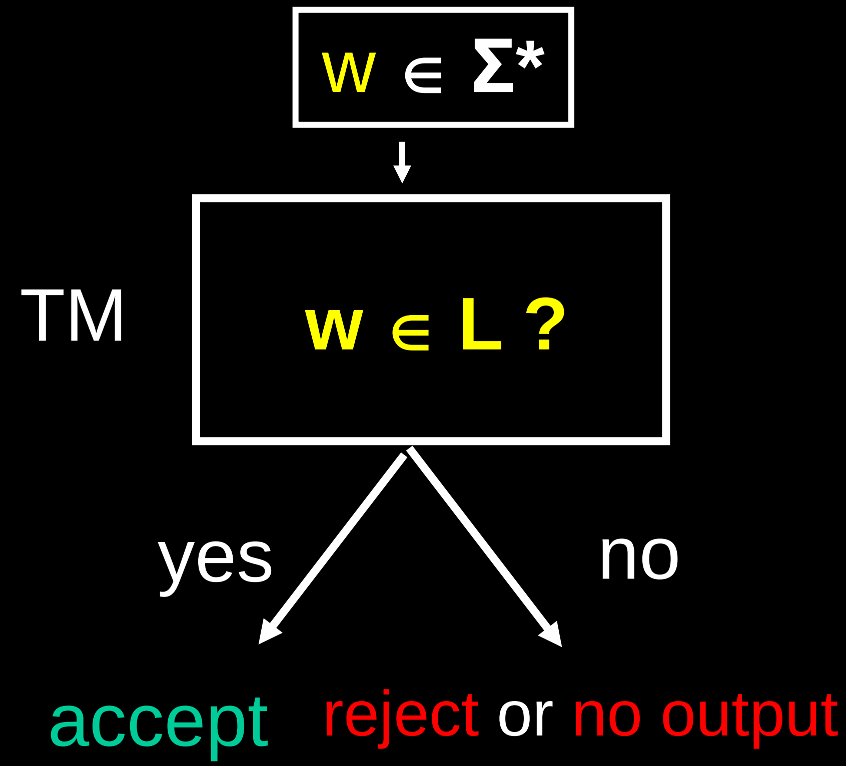
A language is called **Turing-recognizable** or **recursively enumerable,** (or **r.e.** or **semi-decidable**) if some TM recognizes it

A TM **decides** a language if it accepts all strings in the language and rejects all strings not in the language

A language is called **decidable** or **recursive** if some TM decides it



L is decidable
(recursive)



L is semi-decidable
(recursively enumerable,
Turing-recognizable)

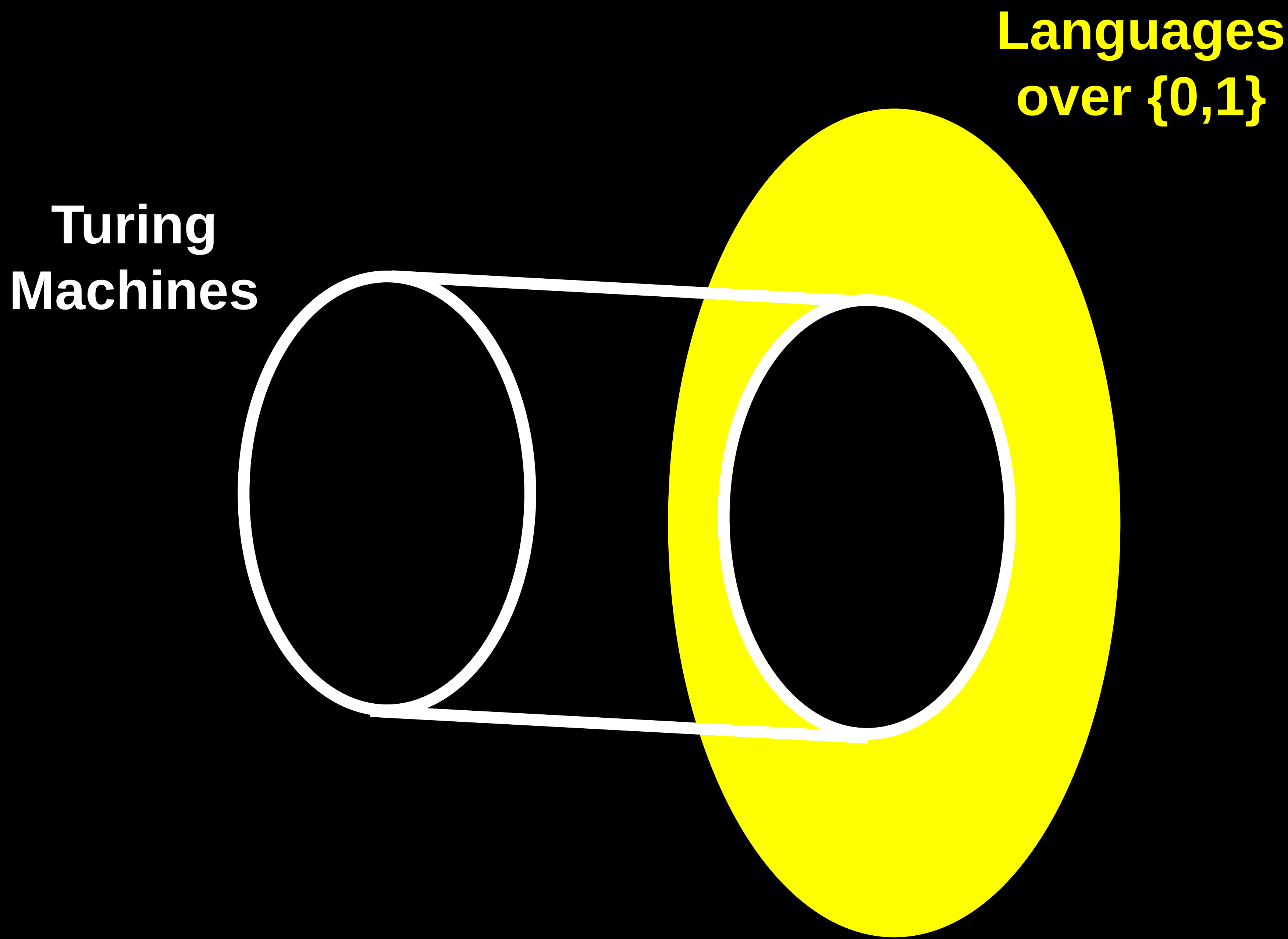
Theorem: L is decidable if both L and ¬L are recursively enumerable

There are languages over {0,1} that are not decidable

If we believe the Church-Turing Thesis,
this is MAJOR: it means there are things that computers inherently cannot do

We can prove this using a counting argument. We will show there is no onto function from the set of all Turing Machines to the set of all languages over {0,1}. (Works for any Σ) Hence there are languages that have no decider.

Then we will prove something stronger:
There are semi-decidable (r.e.) languages that are NOT decidable


Languages
over {0,1}
Turing
Machines

Let L be any set and 2^L be the power set of L

Theorem: There is no onto map from L to 2^L

Proof: Assume, for a contradiction, that there is an onto map $f : L \rightarrow 2^L$

$$\text{Let } S = \{ x \in L \mid x \notin f(x) \}$$

If $S = f(y)$ then $y \in S$ if and only if $y \notin S$

Let L be any set and 2^L be the power set of L

Theorem: There is no onto map from L to 2^L

Proof: Assume, for a contradiction, that there is an onto map $f : L \rightarrow 2^L$

$$\text{Let } S = \{ x \in L \mid x \notin f(x) \}$$

If $S = f(y)$ then $y \in S$ if and only if $y \notin S$

Can give a more constructive argument!

Theorem: There is no onto function from the positive integers to the real numbers in (0, 1)

Proof: Suppose f is any function mapping the positive integers to the real numbers in (0, 1:

1	$\longrightarrow$	0.28347279…
2	$\longrightarrow$	0.88388384…
3	$\longrightarrow$	0.77635284…
4	$\longrightarrow$	0.11111111…
5	$\longrightarrow$	0.12345678…
:		:

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

$f(n) \neq r$ for all n (Here, r = 11121...) So f is not onto

THE MORAL:

No matter what L is,

2^L *always* has more elements than L

Not all languages over {0,1} are decidable, in fact: not all languages over {0,1} are semi-decidable

{decidable languages over {0,1}}

{semi-decidable languages over {0,1}}

{Turing Machines}

{Strings of 0s and 1s}

Set L

{Languages over {0,1}}

{Sets of strings of 0s and 1s}

Set of all subsets of L: 2^L

Let $Z^+ = \{1,2,3,4\ldots\}$. There exists a bijection between Z^+ and $Z^+ \times Z^+$ (or Q^+)

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	…
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	…
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	…
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	…
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	…

Let $Z^+ = \{1,2,3,4\ldots\}$. There exists a bijection between Z^+ and $Z^+ \times Z^+$ (or Q^+)

(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	…
(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	…
(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	…
(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	…
(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	…

THE ACCEPTANCE PROBLEM

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

Theorem: A_{TM} is semi-decidable (r.e.) but NOT **decidable**

A_{TM} is r.e. :

Define a TM U as follows:

On input (M, w), U runs M on w. If M ever accepts, accept. If M ever rejects, reject.

NB. When we write "input **(M, w)**" we really mean "input   code for (code for **M**, **w**)"

THE ACCEPTANCE PROBLEM

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

Theorem: A_{TM} is semi-decidable (r.e.) but NOT decidable

A_{TM} is r.e. :

Define a TM U as follows:

U is a *universal TM*

On input (M, w), U runs M on w. If M ever accepts, accept. If M ever rejects, reject.

Therefore,

U accepts (M,w) $\Leftrightarrow$ M accepts w $\Leftrightarrow$ (M,w) $\in A_{TM}$

Therefore, U *recognizes* A_{TM}

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

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

Assume machine H decides A_{TM}

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

Construct a new TM D as follows: on input M, run H on (M,M) and output the opposite of H

$$D(\,M\,) = \begin{cases} \text{Reject} & \text{if } M \text{ accepts } M \\ \text{Accept} & \text{if } M \text{ does not accept } M \end{cases}$$

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

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

Assume machine H decides A_{TM}

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

Construct a new TM D as follows: on input M, run H on (M,M) and output the opposite of H

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

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

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

Assume machine H decides A_{TM}

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

Construct a new TM D as follows: on input M, run H on (M,M) and output the opposite of H

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

OUTPUT OF H

	M_1	M_2	M_3	M_4	…
M_1	accept	accept	accept	reject	accept
M_2	reject	accept	reject	reject	reject
M_3	accept	reject	reject	accept	accept
M_4	accept	reject	reject	reject	accept
:					

OUTPUT OF H

	M_1	M_2	M_3	M_4 ...	D
M_1	accept	accept	accept	reject	accept
M_2	reject	accept	reject	reject	reject
M_3	accept	reject	reject	accept	accept
M_4	accept	reject	reject	reject	accept
:					
D	reject	reject	accept	accept	?

Theorem: A_{TM} is r.e. but NOT decidable

Theorem: $\neg A_{TM}$ is not even r.e.!

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

A_{TM} **is undecidable:** A constructive proof:

Let machine H semi-decides A_{TM} (Such $\exists$, why?)

H((M,w)) = { Accept if M accepts w; Reject or No output if M does not accept w }

$$H((M,w)) = \begin{cases} \text{Accept} & \text{if } M \text{ accepts } w \\ \text{Reject or} & \\ \text{No output} & \text{if } M \text{ does not accept } w \end{cases}$$

Construct a new TM D as follows: on input M, run H on (M,M) and output

$$D(M) = \begin{cases} \text{Reject} & \text{if } H(M, M) \text{ Accepts} \\ \text{Accept} & \text{if } H(M\ M) \text{ Rejects} \\ \text{No output} & \text{if } H(M, M \text{ has No output} \end{cases}$$

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

A_{TM} is undecidable: A constructive proof:

Let machine H semi-decides A_{TM} (Such ∃ , why?)

$$H((M,w)) = \begin{cases} \text{Accept} & \text{if } M \text{ accepts } w \\ \text{Reject or} & \\ \text{No output} & \text{if } M \text{ does not accept } w \end{cases}$$

Construct a new TM D as follows: on input M, run H on (M,M) and output

$$D(D) = \begin{cases} \text{Reject} & \text{if } H(D, D) \text{ Accepts} \\ \text{Accept} & \text{if } H(D, D) \text{ Rejects} \\ \text{No output} & \text{if } H(D, D \text{ has No output} \end{cases}$$

H((D,D)) = No output **No Contradictions !**

We have shown:

Given any machine H for semi-deciding A_{TM}, we can *effectively construct* a TM D such that (D,D) $\notin A_{TM}$ but H fails to tell us that.

That is, H *fails* to be a decider on instance (D,D).

In other words,

Given any "good" candidate for deciding the *Acceptance Problem*, we can effectively construct an instance where the candidate fails.

THE classical 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 $HALT_{TM}$

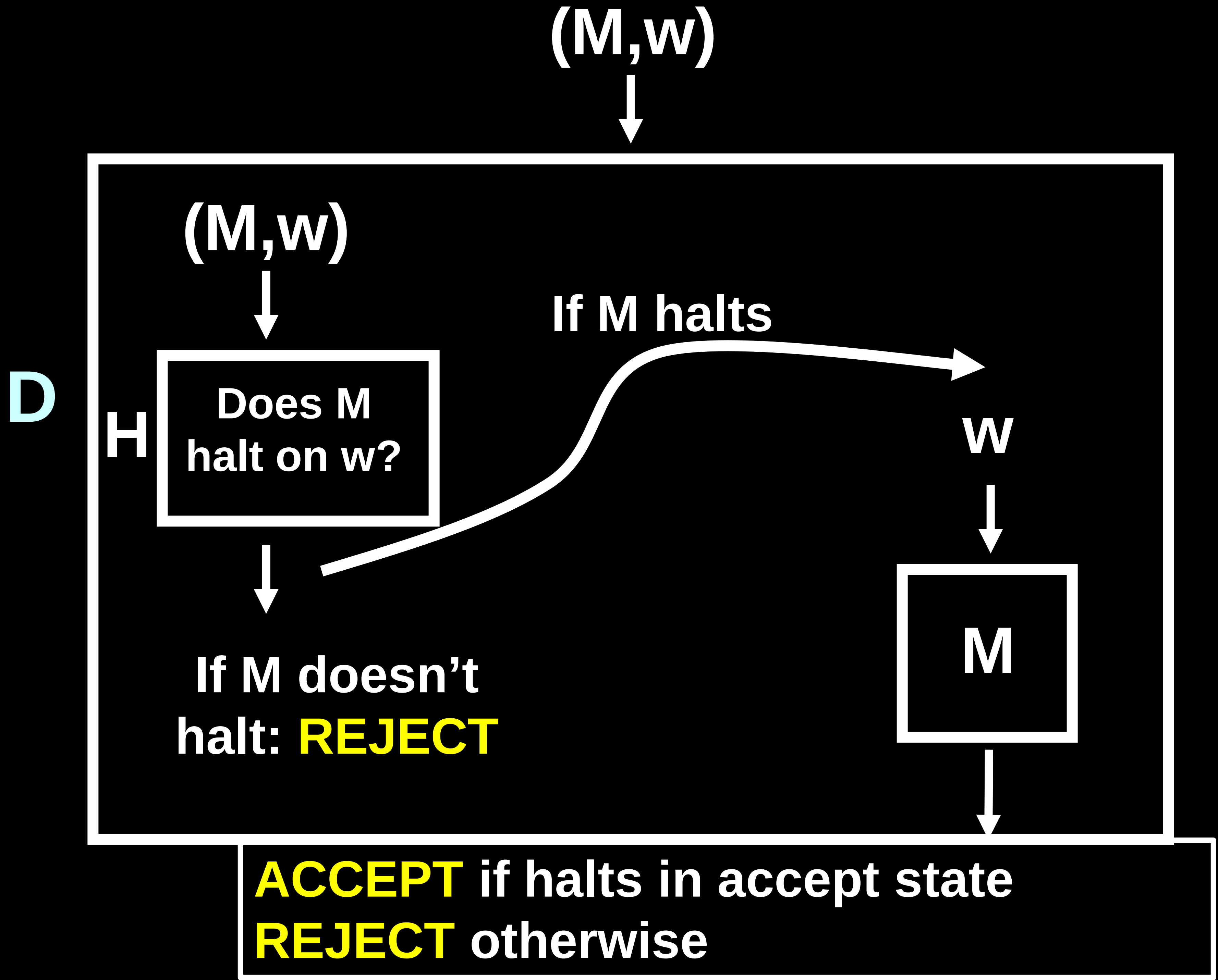
We use H to construct a TM D that decides A_{TM}

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 ie halts in an accept state
Otherwise reject


(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

In many cases, one can 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

$$A_{TM} \leq L$$

We just showed: $A_{TM} \leq Halt_{TM}$
Is $Halt_{TM} \leq A_{TM}$?

WWW.FLAC.WS

Read chapter 4 of the book for next time