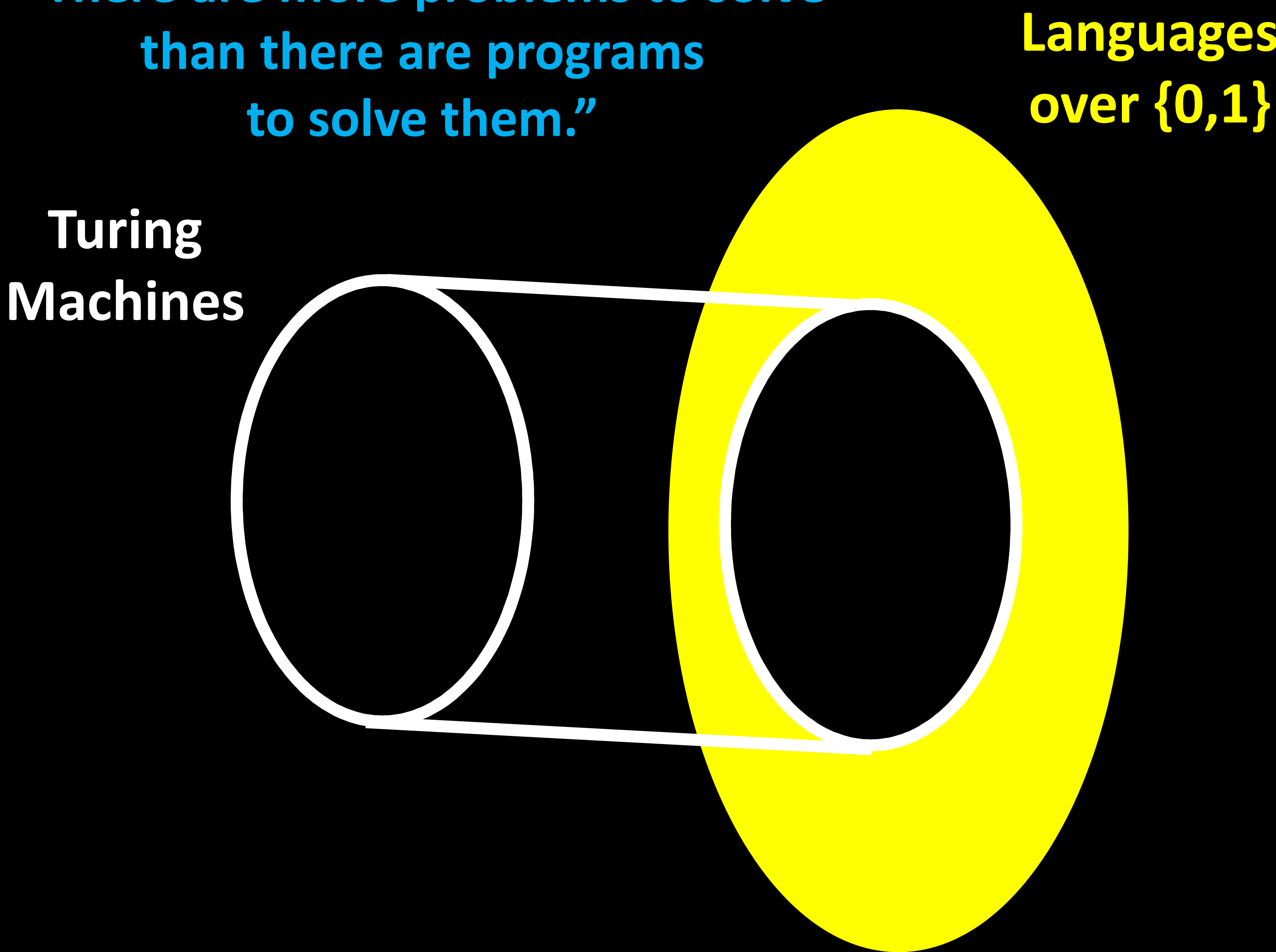
CS 154

Lecture 9:
Diagonalization, Undecidability, Unrecognizability

"There are more problems to solve than there are programs to solve them."



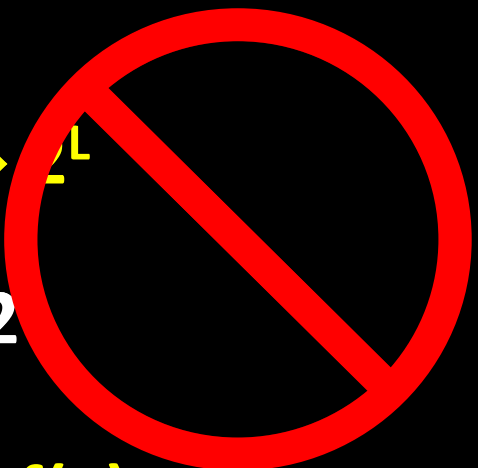
$f : A \rightarrow B$ is onto $\Leftrightarrow$ $(\forall\, b \in B)(\exists\, a \in A)[f(a) = b]$

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: 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!

$f : A \to B$ is *not* onto $\Leftrightarrow$ $(\exists b \in B)(\forall a \in A)[f(a) \neq b]$

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 $f : L \to 2^L$ 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 $x \notin f(x)$ [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 L to 2^L
can "cover" all the elements in 2^L

No matter what the set L is,
the power set 2^L *always* has
strictly larger cardinality than 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: {Turing Machines} $\rightarrow$ {Languages}

{Turing Machines}

$\subseteq$

{0,1}*

=

Set **M**

{Languages over {0,1}}

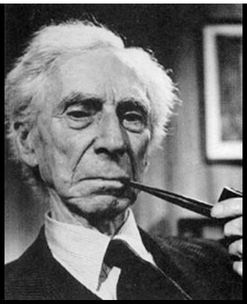
$\updownarrow$

{Sets of strings of 0s and 1s}

=

Set of all subsets of M: 2^M

Therefore, there is *no* onto function from {Turing Machines} $\subseteq$ M to {Languages}. Contradiction!



Russell's Paradox in Set Theory

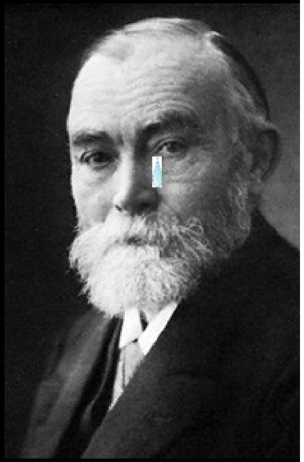
In the early 1900's, logicians were trying to define consistent foundations for mathematics.

Suppose X = "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.

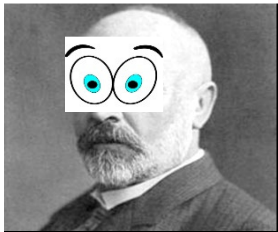
Define $F = \{ S \in X \mid S \notin S \}$



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 Z^+ to the real numbers in (0, 1)

{0,1}* Power set of {0,1}*

Proof: Suppose f is such a function:

$$
\begin{array}{ccc}
1 & \longrightarrow & 0.\mathbf{2}8347279\ldots \\
2 & \longrightarrow & 0.8\mathbf{8}388384\ldots \\
3 & \longrightarrow & 0.77\mathbf{6}35284\ldots \\
4 & \longrightarrow & 0.111\mathbf{1}1111\ldots \\
5 & \longrightarrow & 0.1234\mathbf{5}678\ldots \\
\vdots & & \vdots
\end{array}
$$

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

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

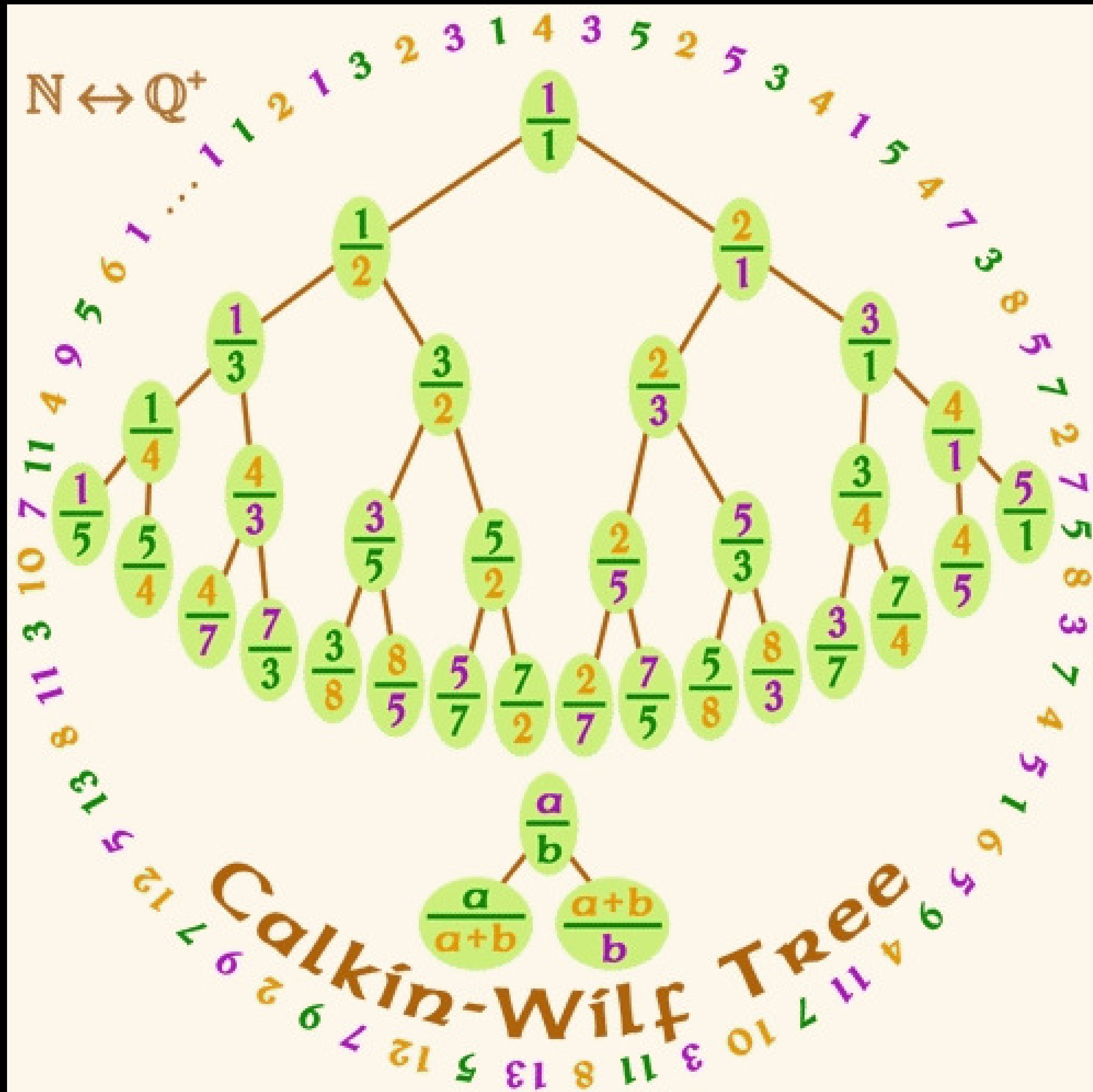
$f(n) \neq r$ for all n (Here, $r = 0.11121\ldots$)

r is never output by f

Let $Z^+ = \{1, 2, 3, 4, \ldots\}$

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

(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)	…


$\mathbb{N} \leftrightarrow \mathbb{Q}^+$
Calkin-Wilf Tree

A Concrete Undecidable Problem: The Acceptance Problem for TMs

A_{TM} = { (**M, w**) | **M** 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) | M 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)

	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		?

The outputs of D(x)

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

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

D(D) outputs the opposite of H(D,D)=D(D)

A_{TM} = { (M,w) | M 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

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

Theorem: $HALT_{TM}$ is undecidable

Proof: Assume (for a contradiction)
there is a TM H that decides $HALT_{TM}$

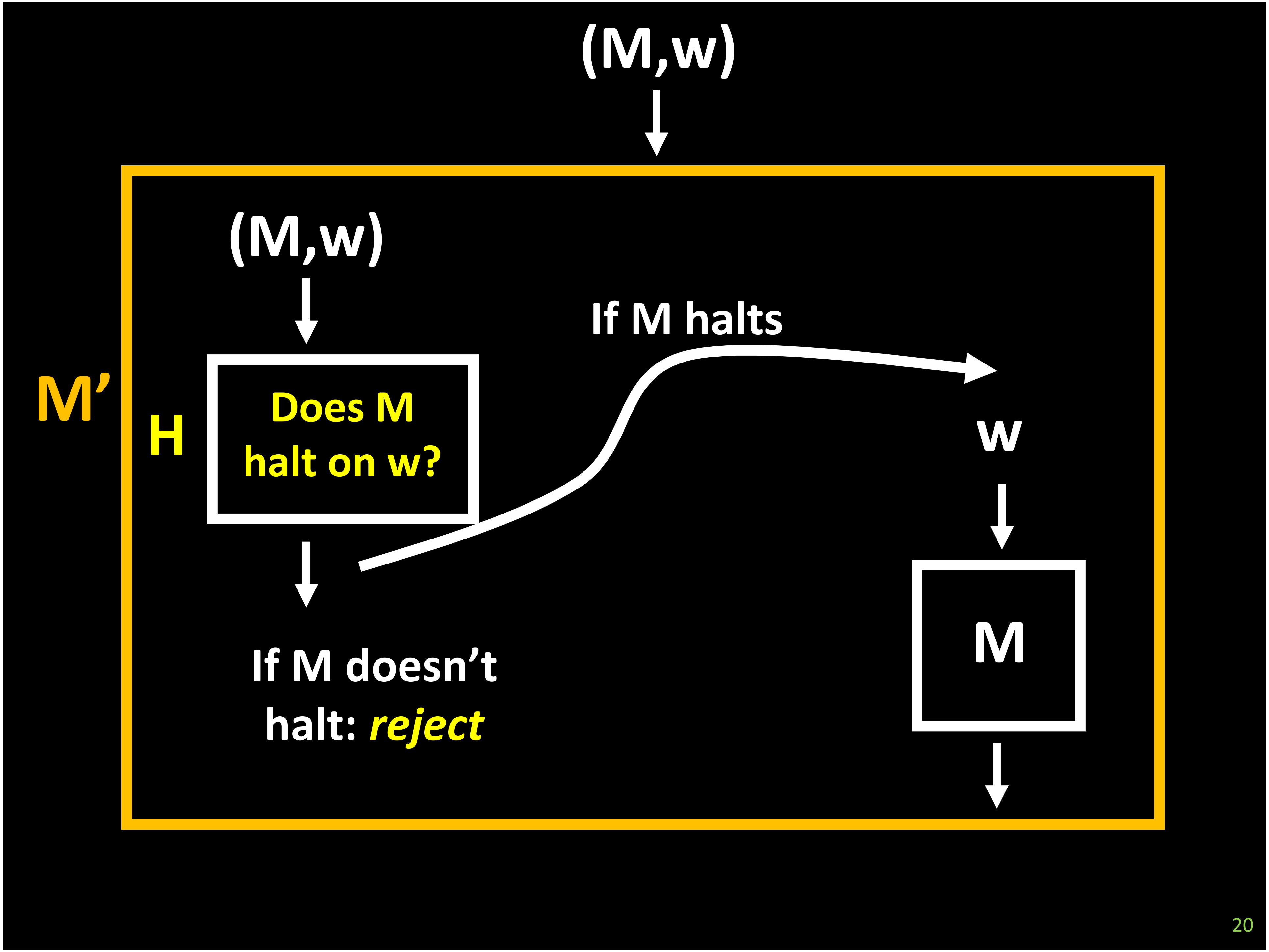
We use H to construct a TM M' that *decides* A_{TM}

M'(M,w): 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*


(M,w)
M’
H
(M,w)
Does M
halt on w?
If M halts
w
If M doesn’t
halt: *reject*
M

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

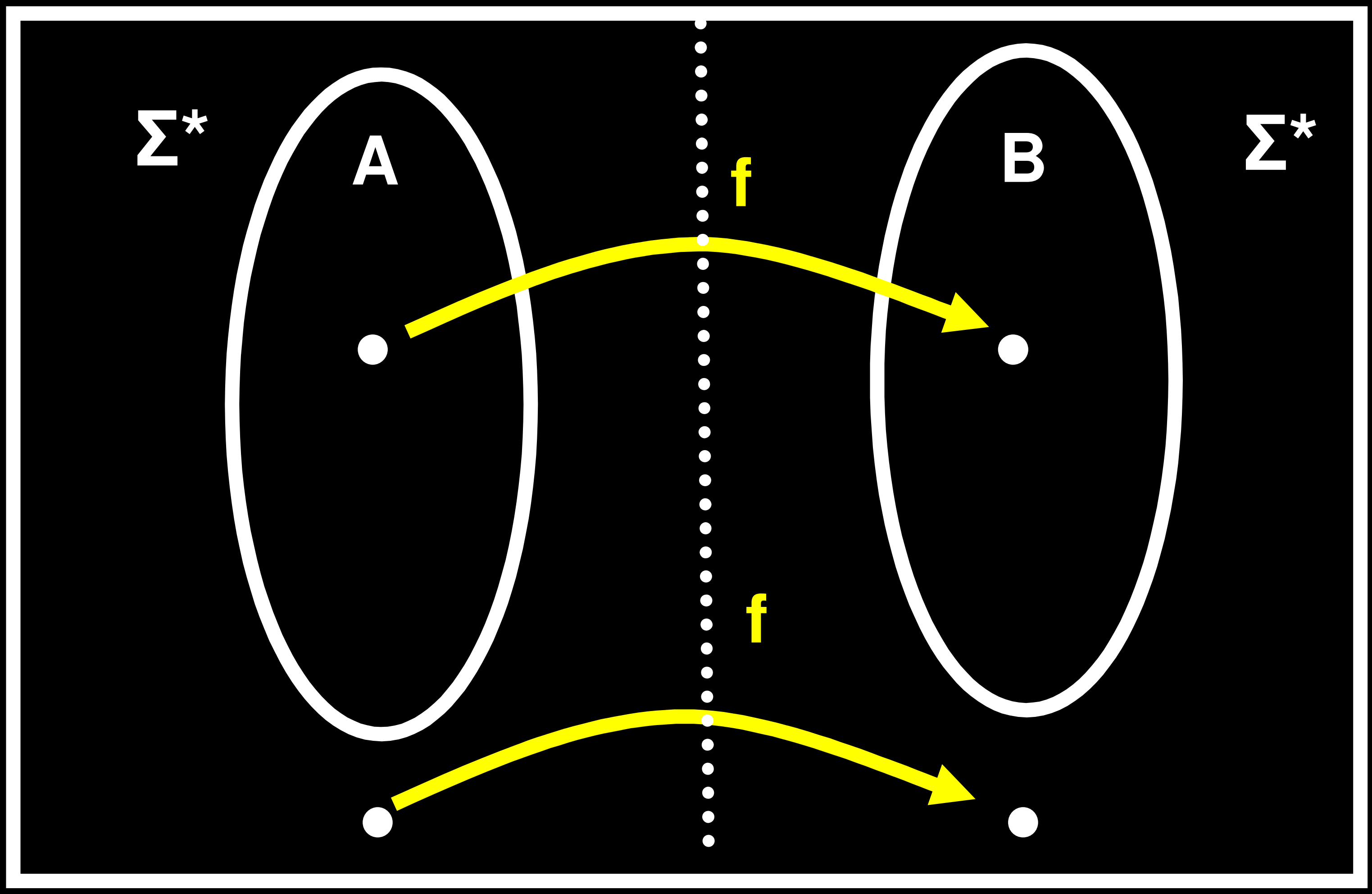
$f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if
there is a Turing machine M that halts with
just f(w) written on its tape, for every input w

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,

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

f **is called a mapping reduction**
(or many-one reduction) from A to B

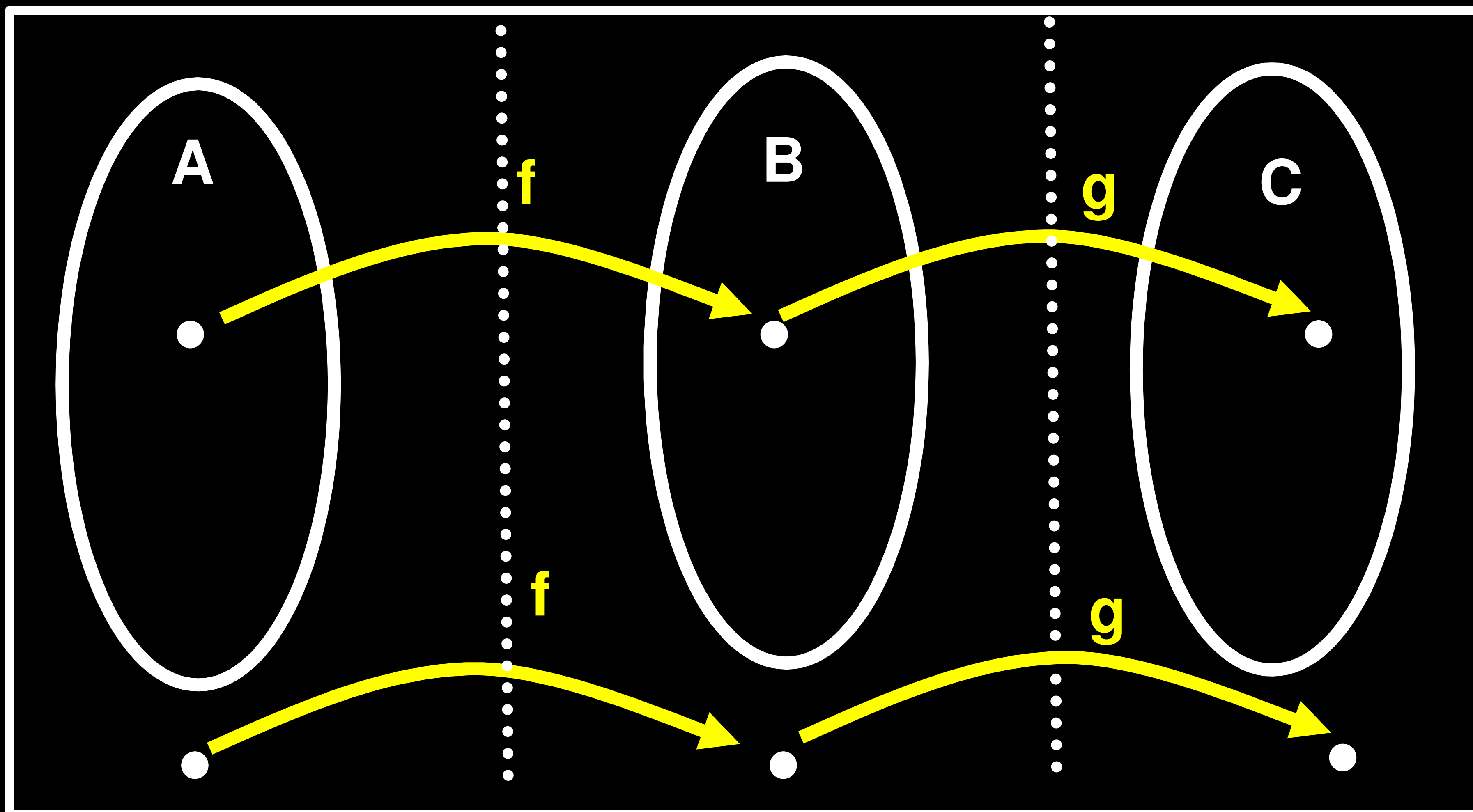
Let $f : \Sigma^* \rightarrow \Sigma^*$ be a computable function such that $w \in A \Leftrightarrow f(w) \in B$



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

$w \in A \Leftrightarrow f(w) \in B$ so $w \in A \Rightarrow$ M' accepts w
$w \notin A \Rightarrow$ 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) := Construct M' with the specification
"M'(w) = if M(w) accepts then *accept*
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 $z \in A_{TM} \Leftrightarrow (M', w) \in HALT_{TM}$

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

Proof: Define the computable function:

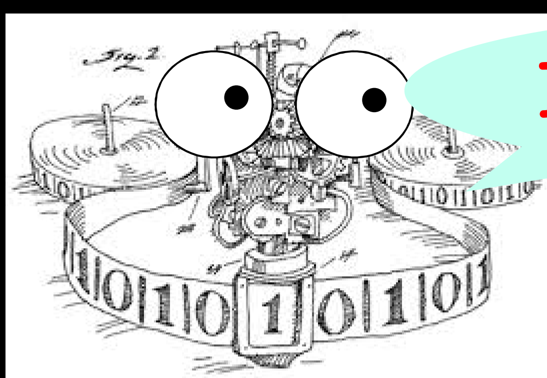
f(z) := Decode z into a pair (M, w)
Construct M' with the specification:
"M'(w) = Simulate M on w.
If M(w) halts then *accept*
else *loop forever*"
Output (M', w)

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

Corollary: $HALT_{TM} \equiv_m A_{TM}$

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

$\text{EMPTY}_{\text{DFA}} = \{ M \mid M \text{ is a DFA such that } L(M) = \varnothing \}$

Given a DFA, does it reject every input?

Theorem: $\text{EMPTY}_{\text{DFA}}$ is decidable

Why?

$\text{EMPTY}_{\text{NFA}} = \{ M \mid M \text{ is a NFA such that } L(M) = \varnothing \}$

$\text{EMPTY}_{\text{REX}} = \{ R \mid M \text{ is a regexp such that } L(M) = \varnothing \}$

The Emptiness Problem for TMs

EMPTY_{TM} = { M | M is a TM such that L(M) = $\varnothing$}

Given a program, does it reject every input?

Theorem: EMPTY_{TM} is *not recognizable*

Proof: Show that $\neg A_{\text{TM}} \leq_m \text{EMPTY}_{\text{TM}}$

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

Output a TM M' with the behavior:

"M'(x) := if (x = w) then run M(w), else *reject*"

$$
\begin{aligned}
z \in A_{\text{TM}} \quad &\Leftrightarrow \quad L(M') \neq \varnothing \\
&\Leftrightarrow \quad M' \notin \text{EMPTY}_{\text{TM}} \\
&\Leftrightarrow \quad f(z) \notin \text{EMPTY}_{\text{TM}}
\end{aligned}
$$