

Decidability

Consider problems with answer YES or NO

Examples:

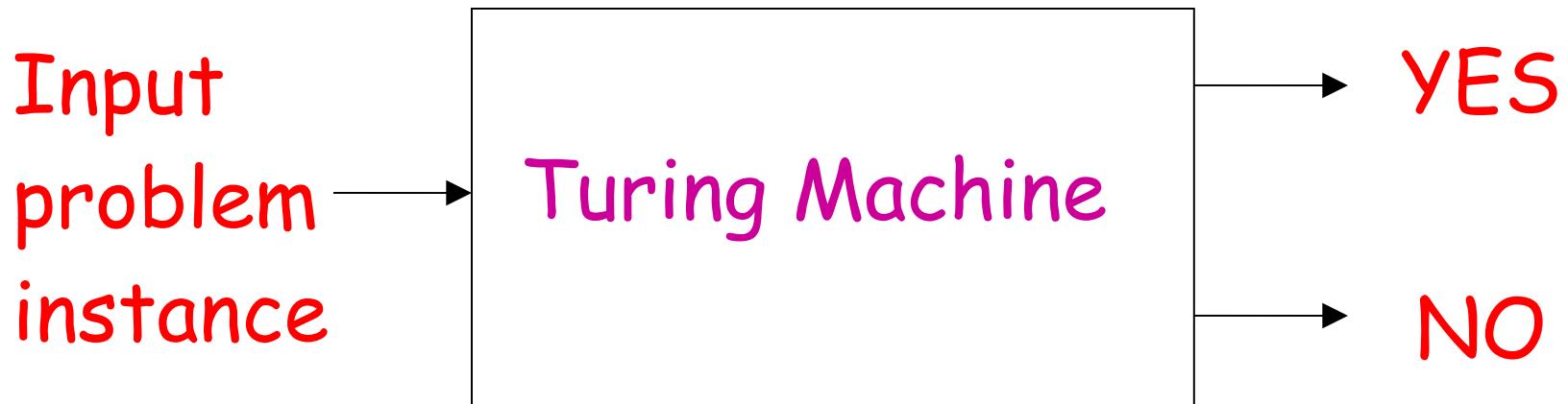
- Does Machine  $M$  have three states ?
- Is string  $w$  a binary number?
- Does DFA  $M$  accept any input?

A problem is decidable if some Turing Machine decides (solves) the problem

Decidable problems:

- Does Machine  $M$  have three states ?
- Is string  $w$  a binary number?
- Does DFA  $M$  accept any input?

The Turing machine that decides (solves)  
a problem answers **YES** or **NO**  
for each instance of the problem

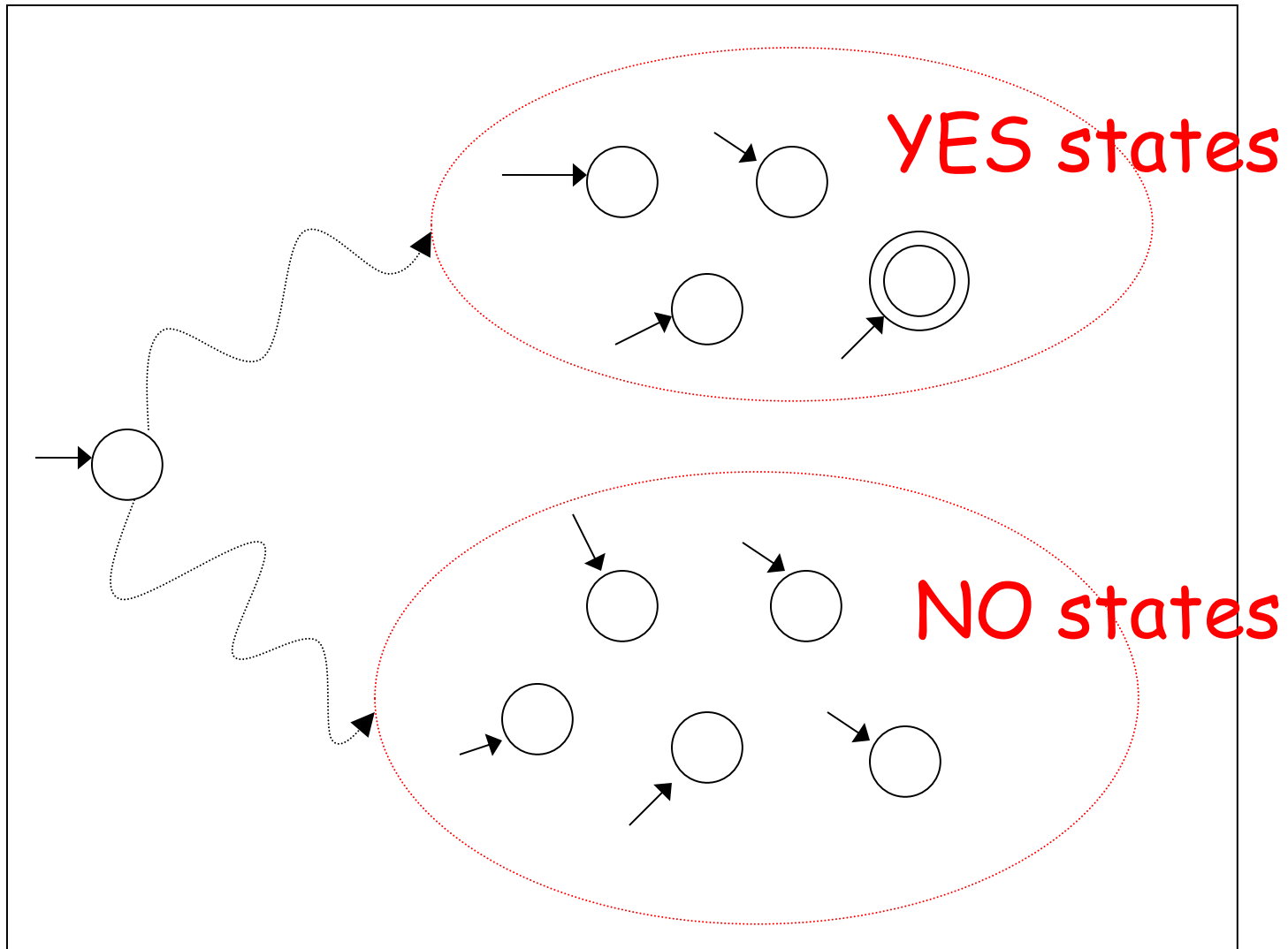


# The machine that decides (solves) a problem:

- If the answer is YES  
then halts in a yes state
- If the answer is NO  
then halts in a no state

These states may not be final states

# Turing Machine that decides a problem



YES and NO states are halting states

# Difference between Recursive Languages and Decidable problems

For decidable problems:

The YES states may not be final states

Some problems are undecidable:

which means:

there is no Turing Machine that  
solves all instances of the problem

A simple undecidable problem:

The membership problem

# The Membership Problem

Input: • Turing Machine  $M$   
• String  $w$

Question: Does  $M$  accept  $w$ ?

$$w \in L(M)?$$

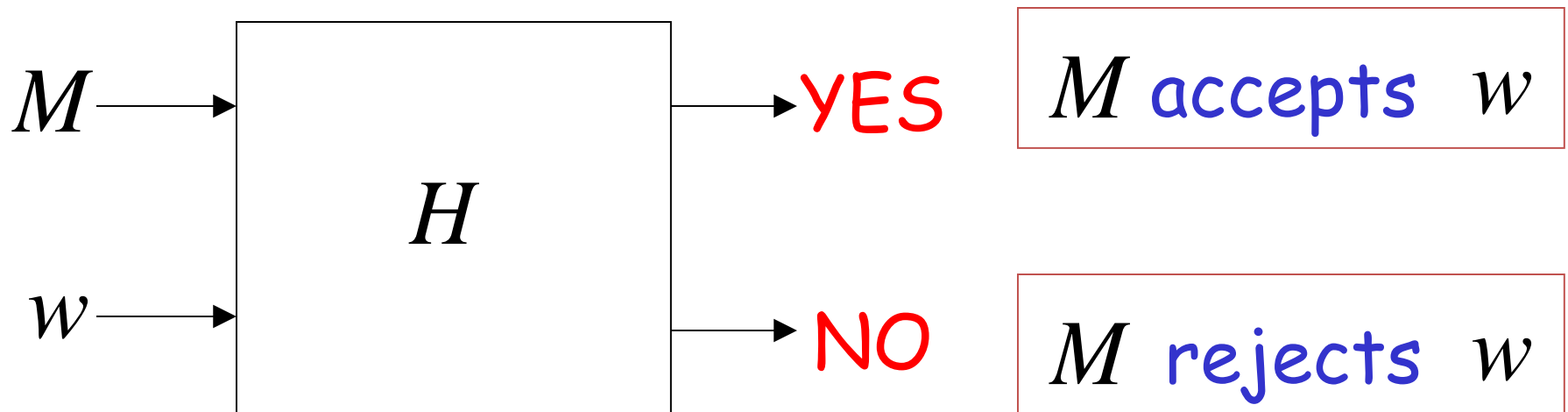
## Theorem:

The membership problem is undecidable

(there are  $M$  and  $w$  for which we cannot  
decide whether  $w \in L(M)$ )

**Proof:** Assume for contradiction that  
the membership problem is decidable

Thus, there exists a Turing Machine  $H$  that solves the membership problem



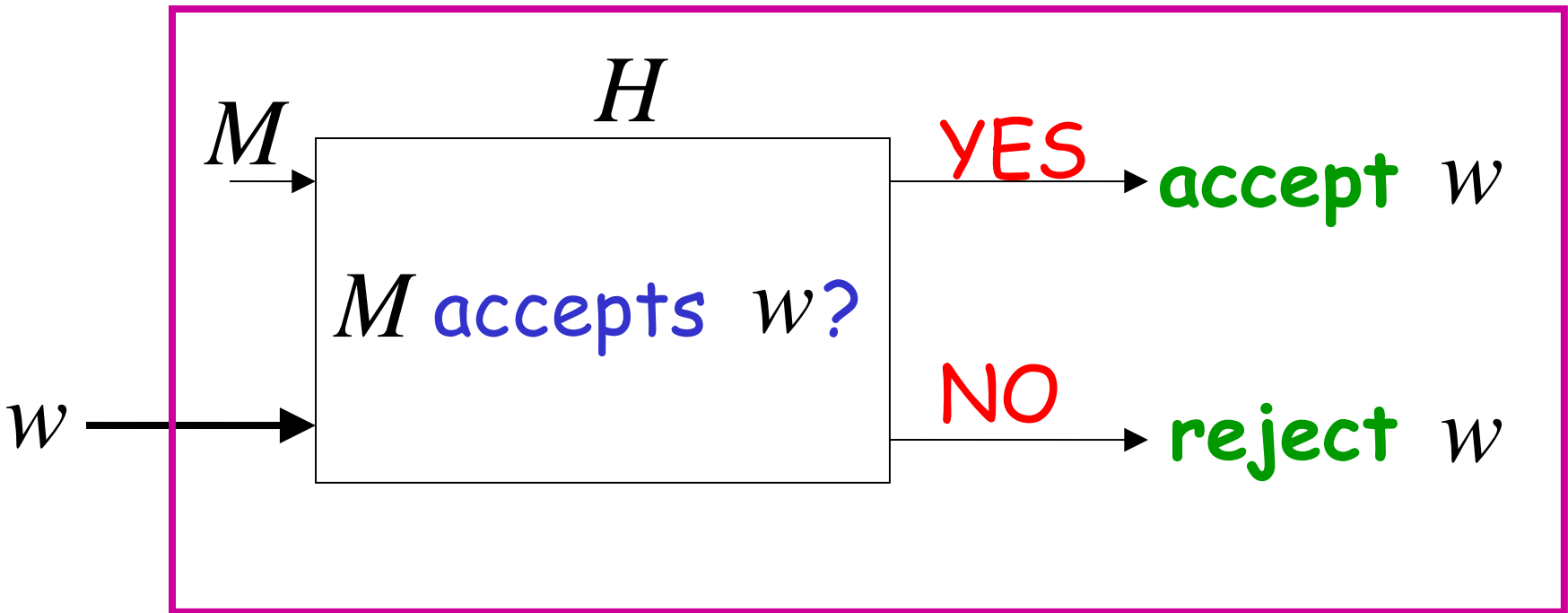
Let  $L$  be a recursively enumerable language

Let  $M$  be the Turing Machine that accepts  $L$

We will prove that  $L$  is also recursive:

we will describe a Turing machine that  
accepts  $L$  and halts on any input

Turing Machine that accepts  $L$   
and halts on any input



Therefore,  $L$  is recursive

Since  $L$  is chosen arbitrarily, every recursively enumerable language is also recursive

But there are recursively enumerable languages which are not recursive

**Contradiction!!!!**

Therefore, the membership problem  
is undecidable

END OF PROOF

Another famous undecidable problem:

The halting problem

# The Halting Problem

Input: • Turing Machine  $M$   
• String  $w$

Question: Does  $M$  halt on input  $w$  ?

## Theorem:

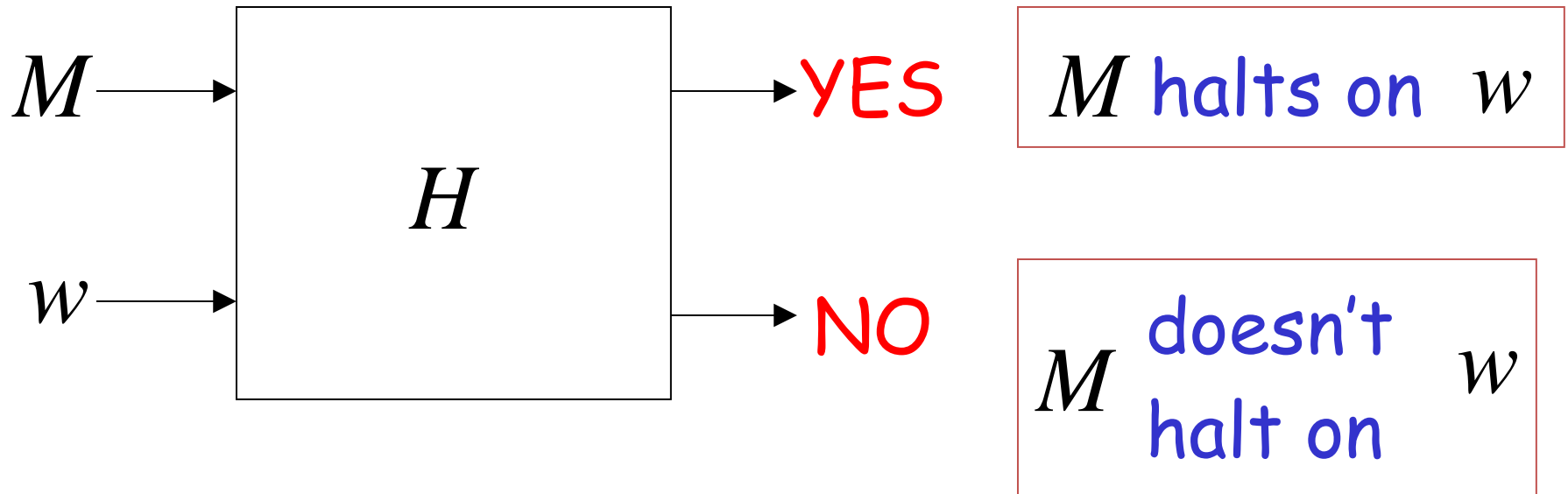
The halting problem is undecidable

(there are  $M$  and  $w$  for which we cannot decide whether  $M$  halts on input  $w$  )

**Proof:** Assume for contradiction that the halting problem is decidable

If the halting problem was decidable then  
every recursively enumerable language  
would be recursive

There exists Turing Machine  $H$   
that solves the halting problem



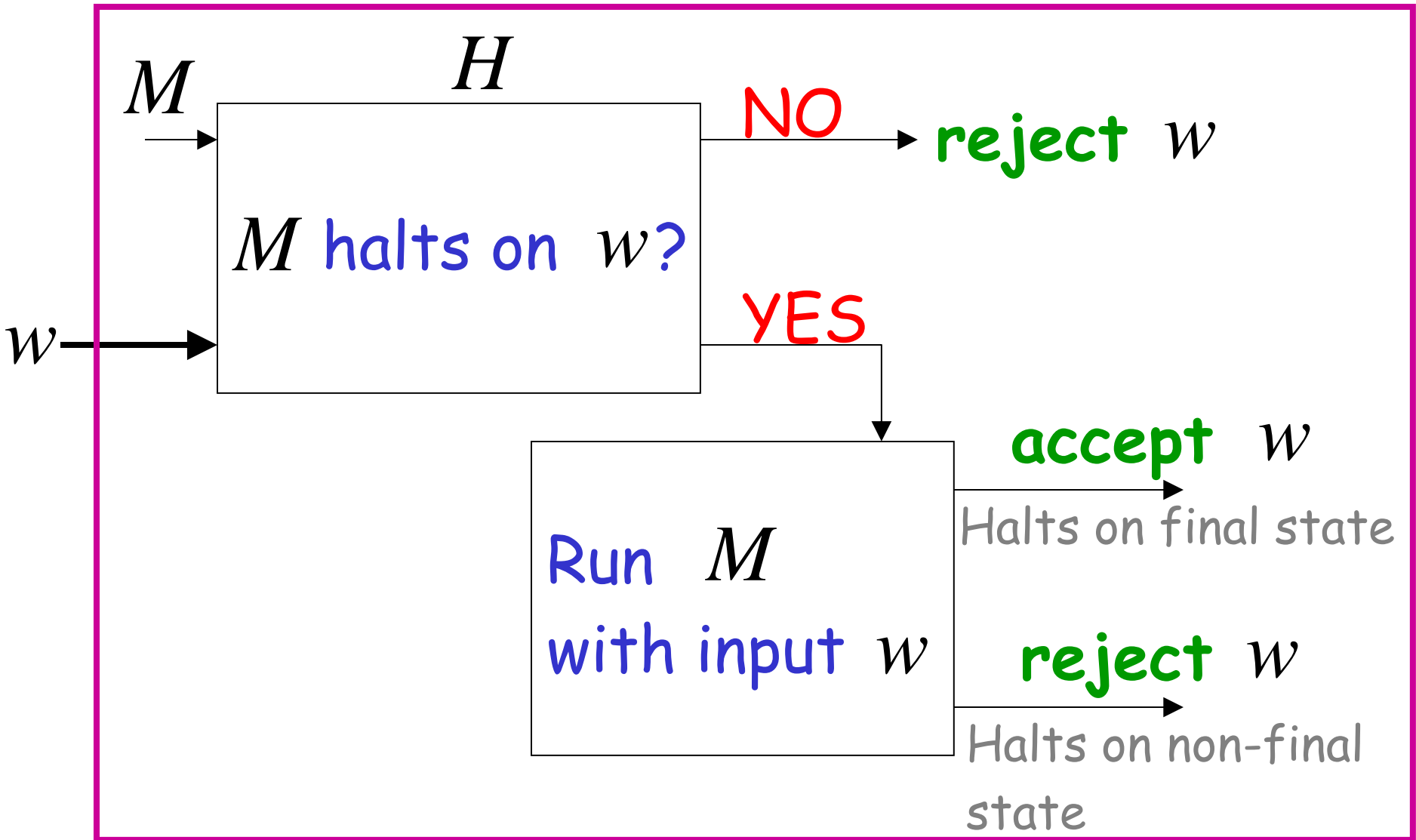
Let  $L$  be a recursively enumerable language

Let  $M$  be the Turing Machine that accepts  $L$

We will prove that  $L$  is also recursive:

we will describe a Turing machine that  
accepts  $L$  and halts on any input

# Turing Machine that accepts $L$ and halts on any input



Therefore  $L$  is recursive

Since  $L$  is chosen arbitrarily, every recursively enumerable language is also recursive

But there are recursively enumerable languages which are not recursive

**Contradiction!!!!**

Therefore, the halting problem is undecidable

END OF PROOF

# Reductions

Problem  $X$  is reduced to problem  $y$

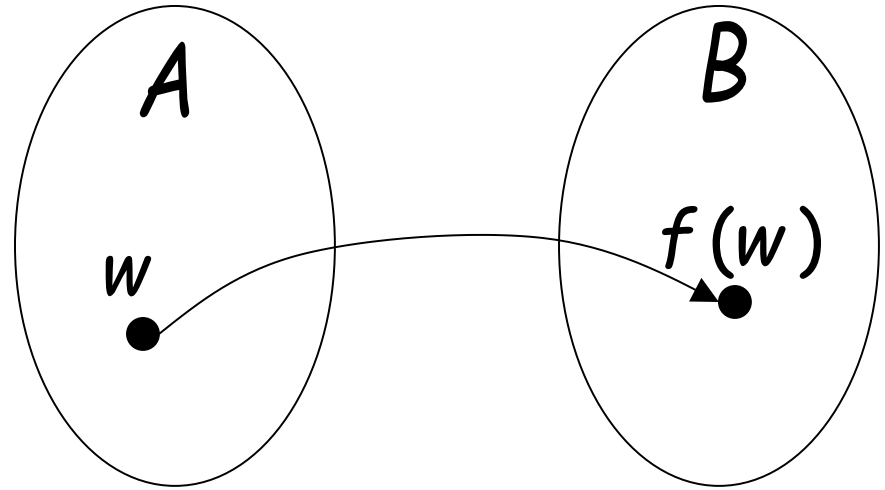


If we can solve problem  $y$

then we can solve problem  $X$

Definition:

Language  $A$   
is reduced to  
language  $B$



There is a computable  
function  $f$  (reduction) such that:

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

Recall:

Computable function  $f$ :

There is a deterministic Turing machine  $M$   
which for any string  $w$  computes  $f(w)$

## Theorem:

If: a: Language  $A$  is reduced to  $B$

b: Language  $B$  is decidable

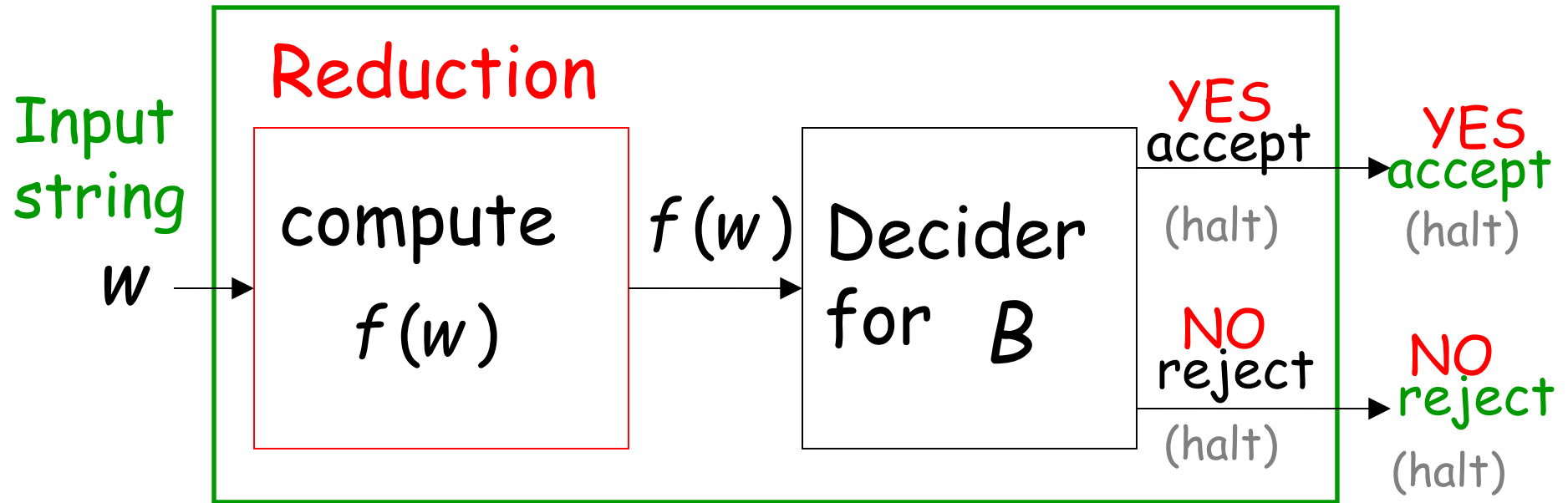
Then:  $A$  is decidable

## Proof:

Basic idea:

Build the decider for  $A$   
using the decider for  $B$

# Decider for $A$



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

END OF PROOF

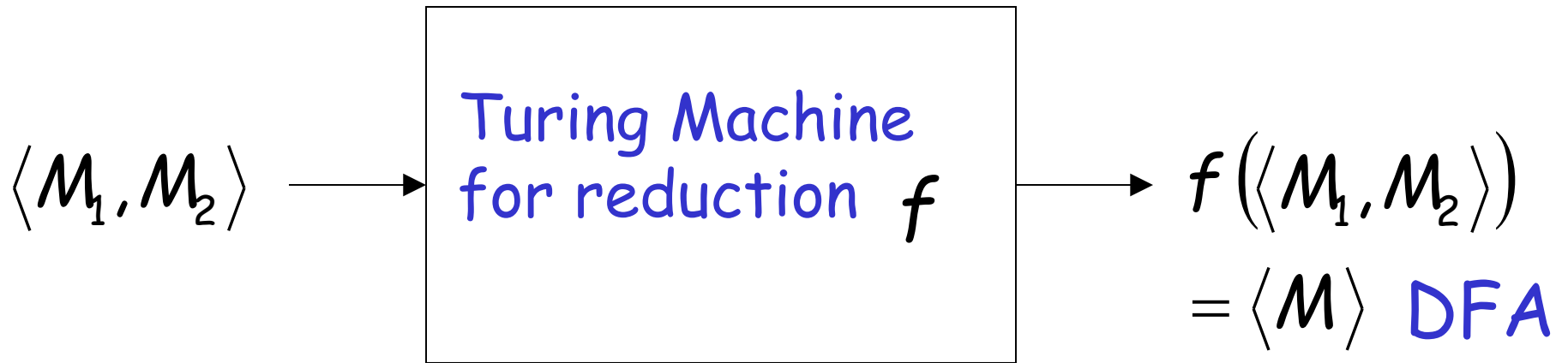
Example:

$$EQUAL_{DFA} = \{\langle M_1, M_2 \rangle : M_1 \text{ and } M_2 \text{ are DFAs} \\ \text{that accept the same languages}\}$$

is reduced to:

$$EMPTY_{DFA} = \{\langle M \rangle : M \text{ is a DFA that accepts} \\ \text{the empty language } \emptyset\}$$

We only need to construct:



$$\langle M_1, M_2 \rangle \in EQUAL_{DFA} \iff \langle M \rangle \in EMPTY_{DFA}$$

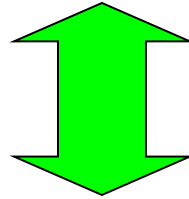
Let  $L_1$  be the language of DFA  $M_1$   
 Let  $L_2$  be the language of DFA  $M_2$



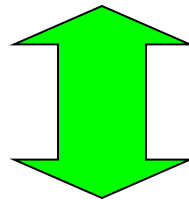
construct DFA  $M$   
 by combining  $M_1$  and  $M_2$  so that:

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

$$L(M) = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$



$$L_1 = L_2 \iff L(M) = \emptyset$$



$$\langle M_1, M_2 \rangle \in EQUAL_{DFA} \iff \langle M \rangle \in EMPTY_{DFA}$$

## Decider for $EQUAL_{DFA}$

