Linear Bounded Automata
LBAs

Linear Bounded Automata (LBAs)

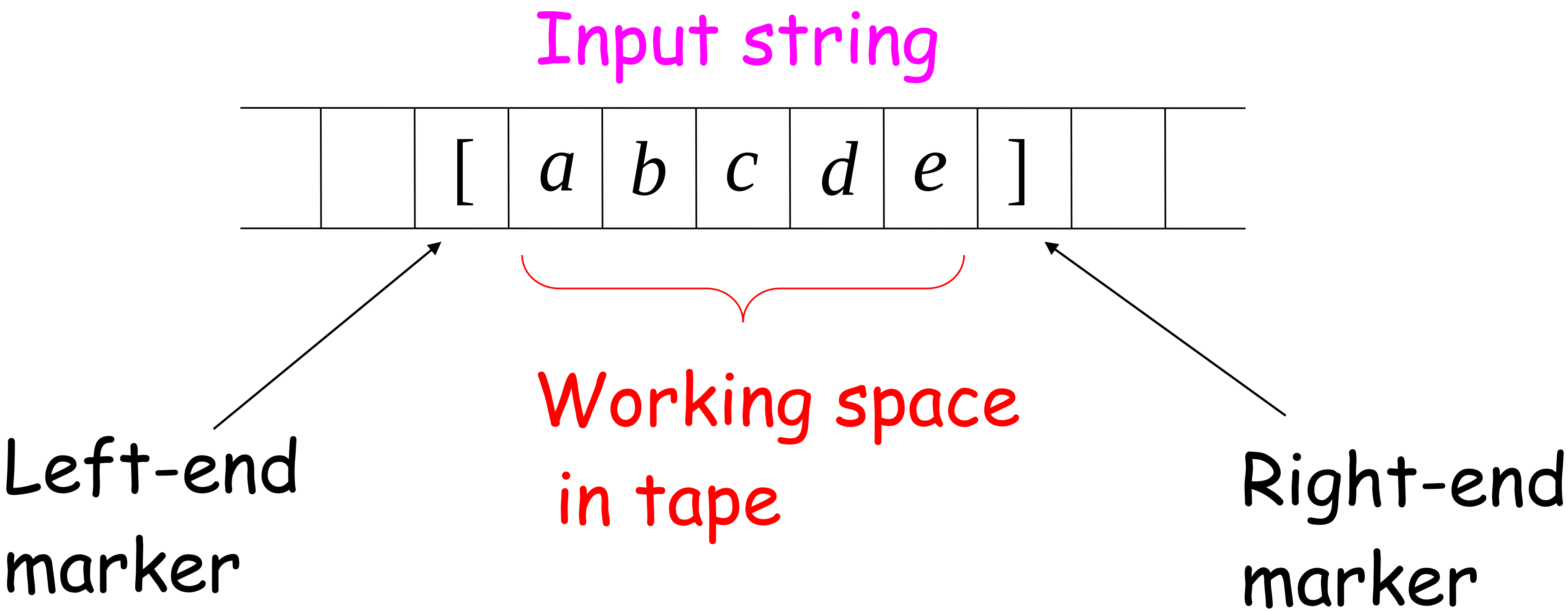
are the same as Turing Machines

with one difference:

The input string tape space

is the only tape space allowed to use

Linear Bounded Automaton (LBA)



All computation is done between end markers

We define LBA's as NonDeterministic

Open Problem:

NonDeterministic LBA's have same power with Deterministic LBA's ?

Example languages accepted by LBAs:

$$L = \{a^n b^n c^n\}$$

$$L = \{a^{n!}\}$$

LBA's have more power than NPDA's

LBA's have also less power
than Turing Machines

The Chomsky Hierarchy

Unrestricted Grammars:

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

Example unrestricted grammar:

$$S \to aBc$$

$$aB \to cA$$

$$Ac \to d$$

aBx

Theorem:

A language L is recursively enumerable if and only if L is generated by an unrestricted grammar

Context-Sensitive Grammars:

Productions

$$u \to v$$

String of variables and terminals

String of variables and terminals

and: $|u| \leq |v|$

The language $\{a^n b^n c^n\}$

is context-sensitive:

$$S \to abc \mid aAbc$$

$$Ab \to bA$$

$$Ac \to Bbcc$$

$$bB \to Bb$$

$$aB \to aa \mid aaA$$

Derive $\underline{a^3 b^3 c^3}$

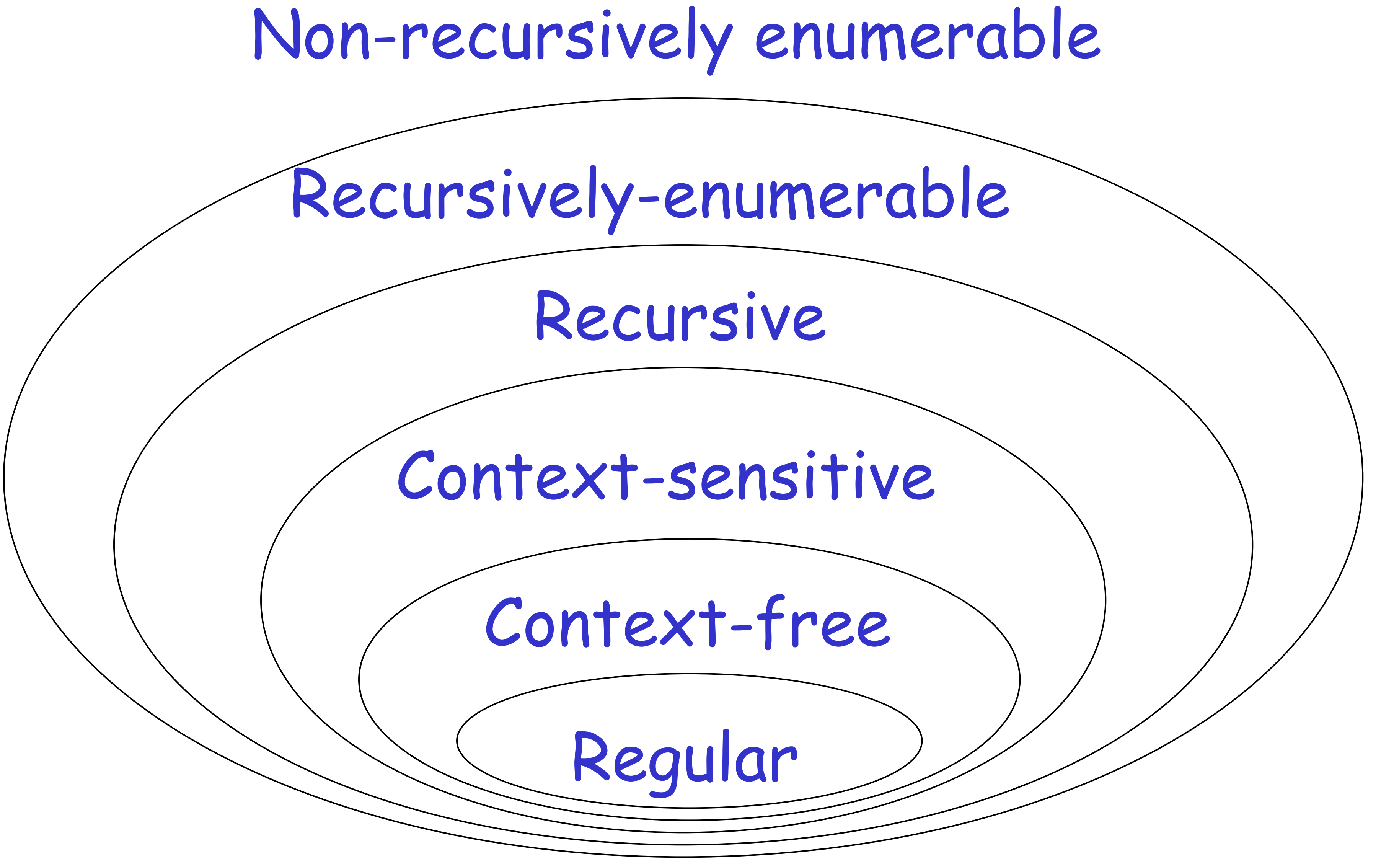
Theorem:

A language L is context sensistive
if and only if
L is accepted by a Linear-Bounded automaton

Observation:

There is a language which is context-sensitive
but not recursive

The Chomsky Hierarchy



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

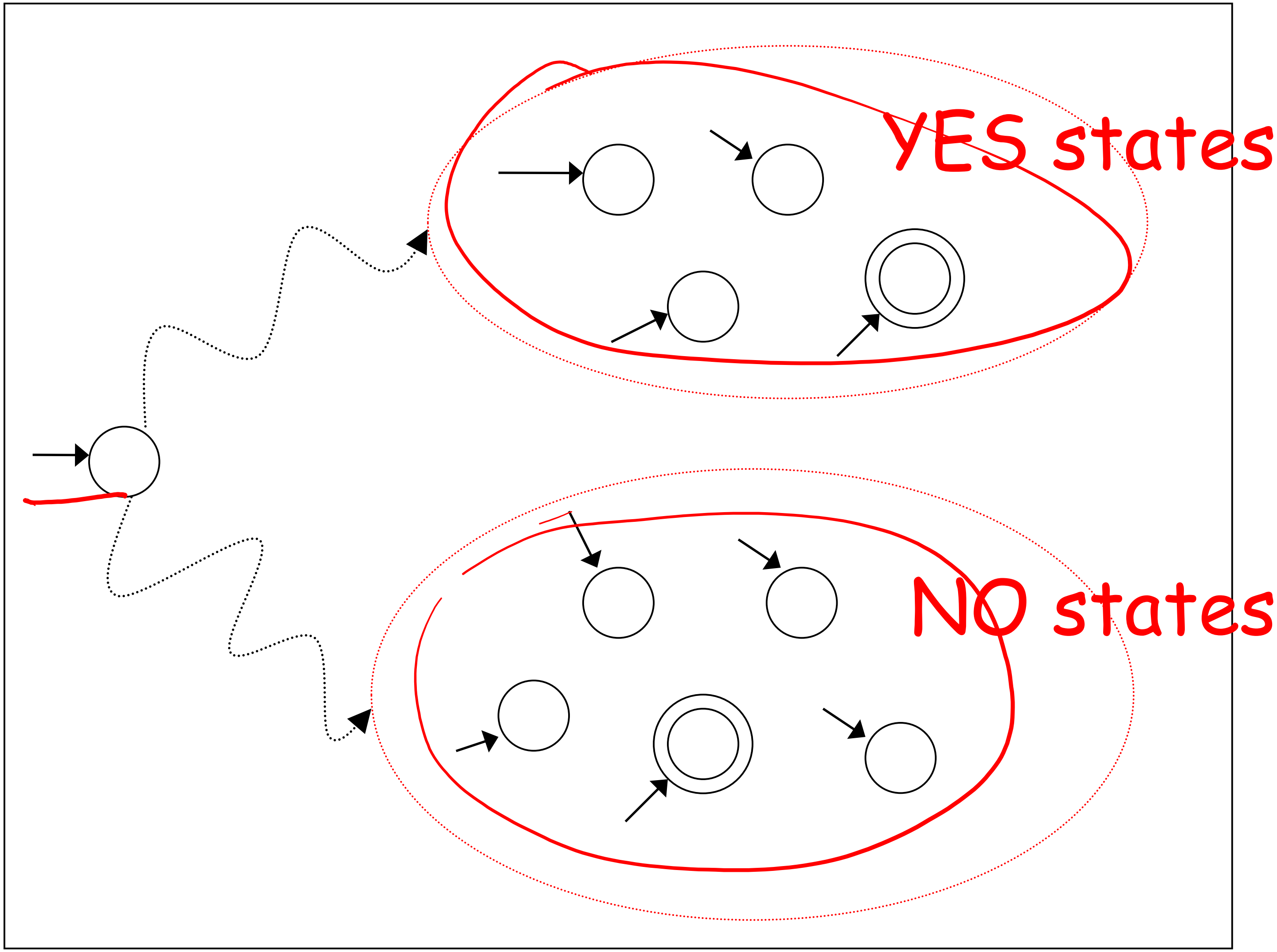
The machine that decides (solves) a problem:

- 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?

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

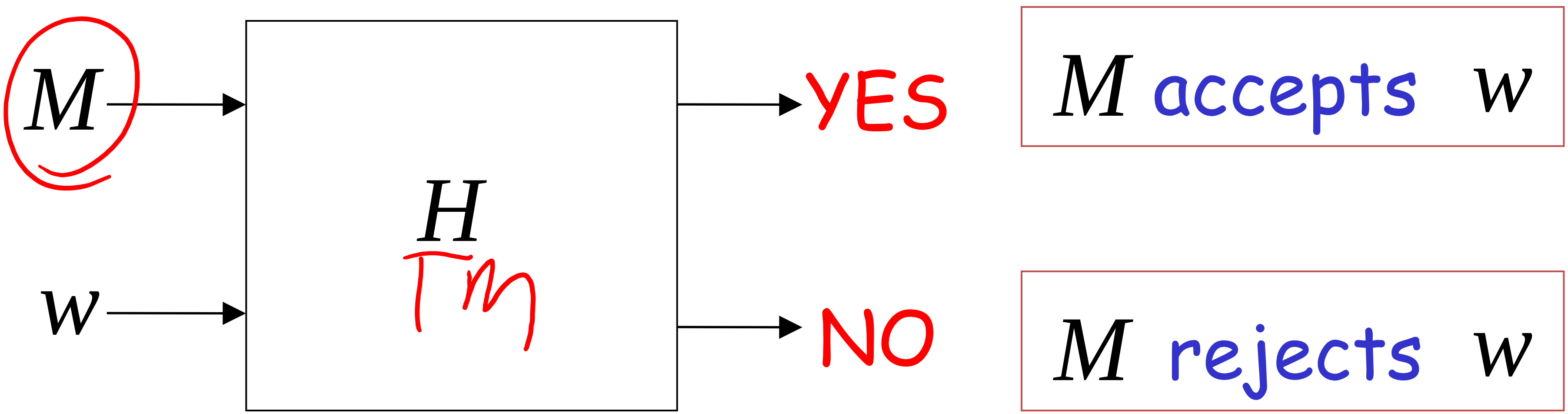
Theorem:

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



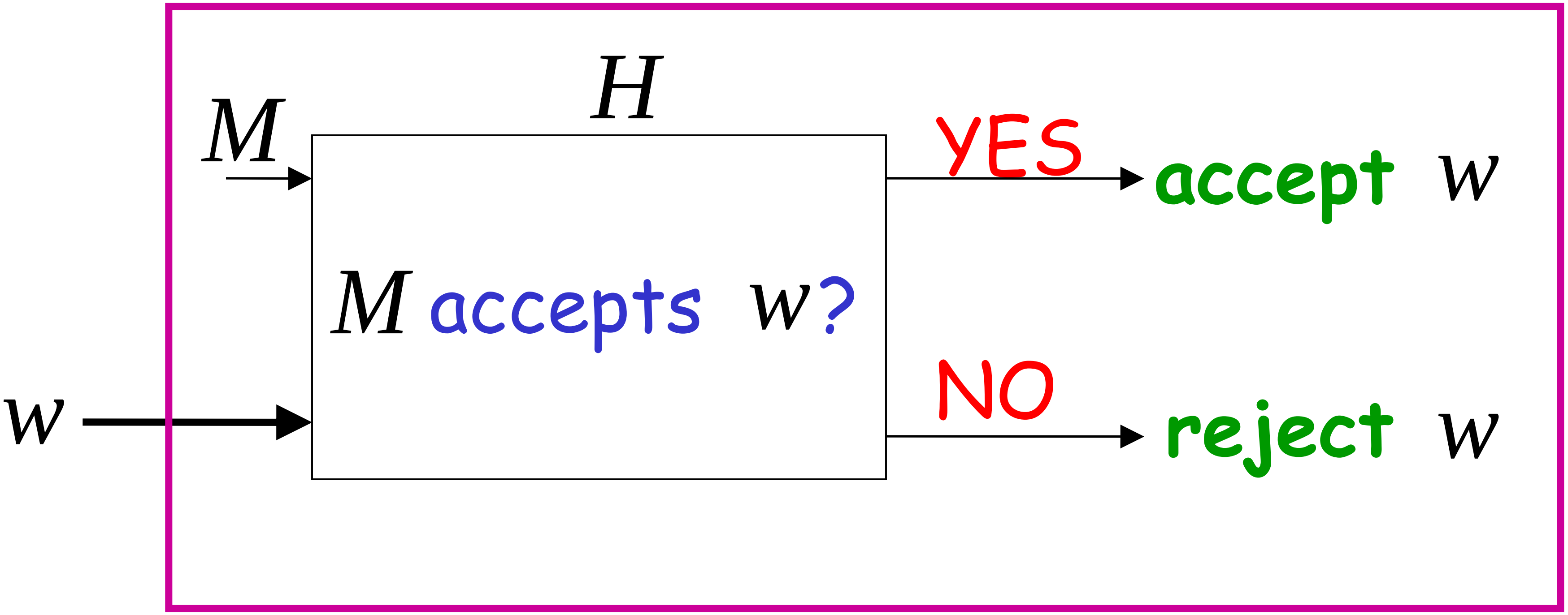
Let L be a recursively enumerable language

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?

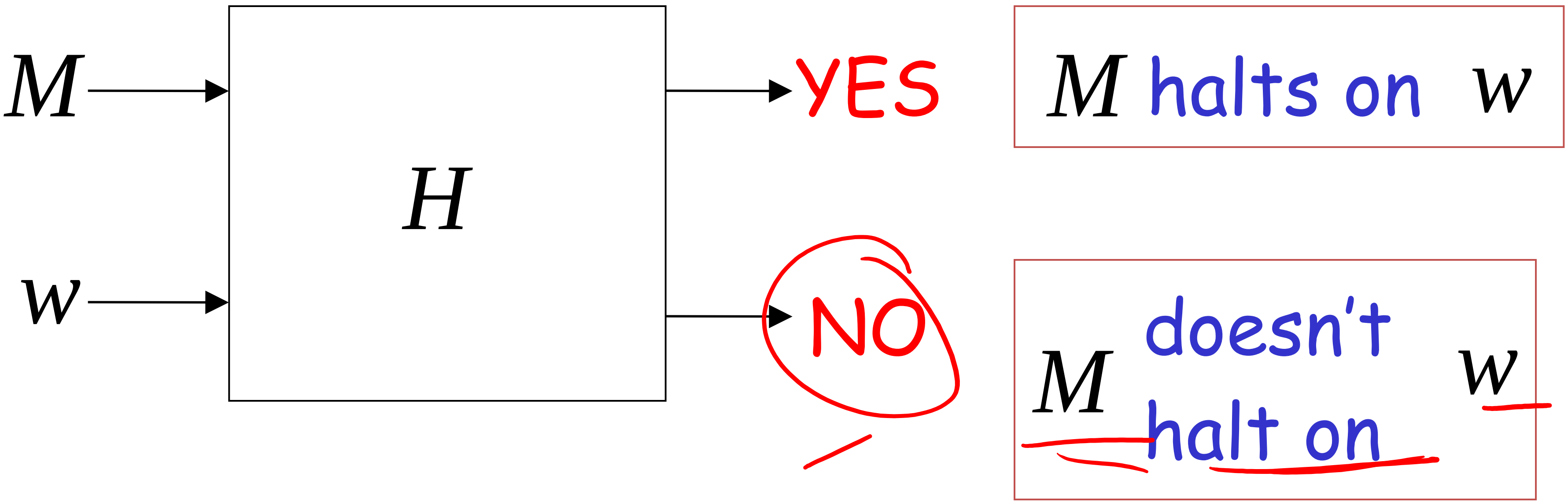
Theorem:

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

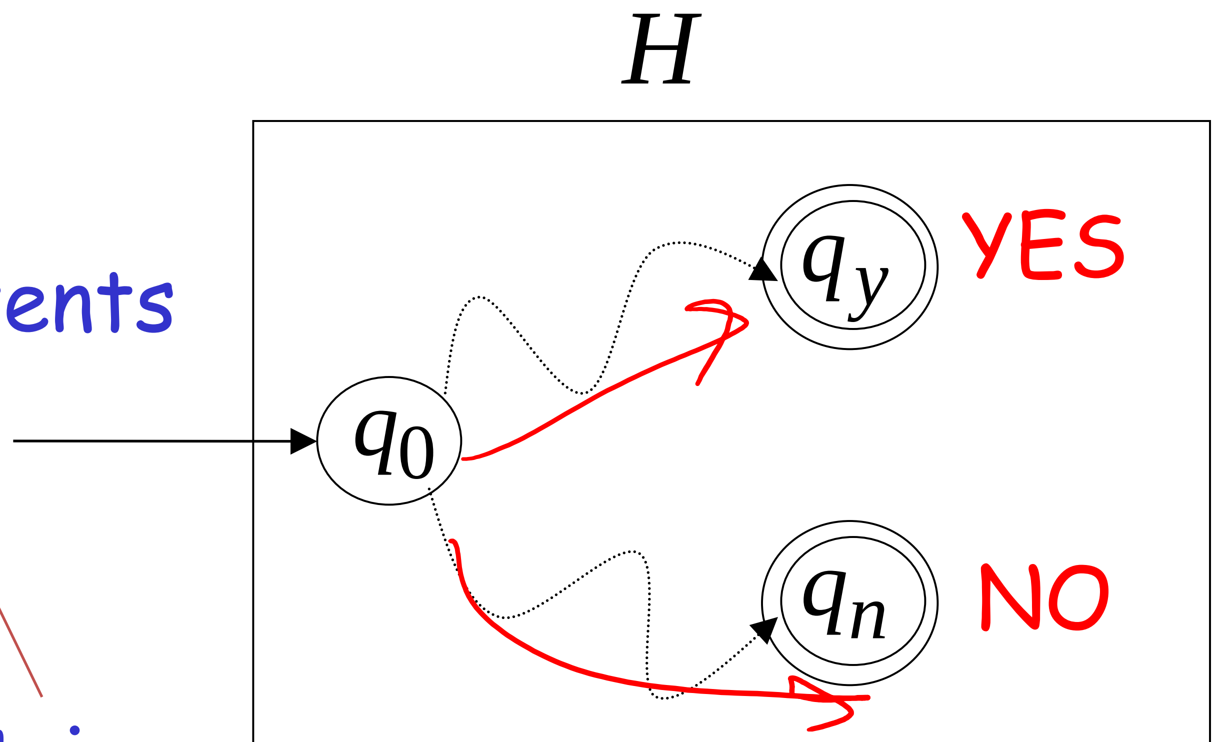
Thus, there exists Turing Machine H that solves the halting problem



Construction of H

Input:
initial tape contents

$w_M \; w$



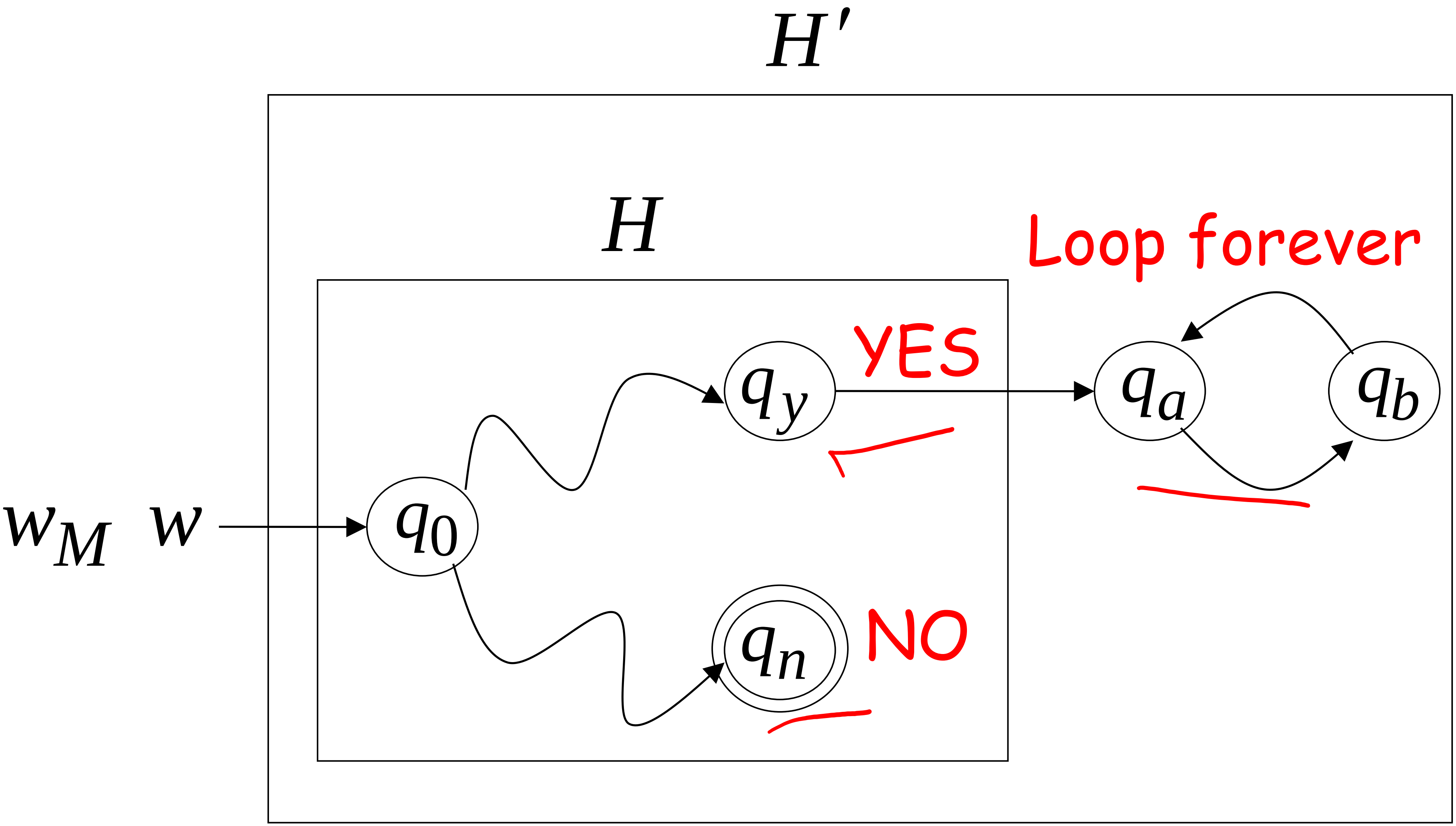
Encoding of M

String w

Construct machine H':

If H returns YES then loop forever

If H returns NO then halt



H'
H
Loop forever
YES
q_y
q_a
q_b
w_M w
q_0
q_n
NO

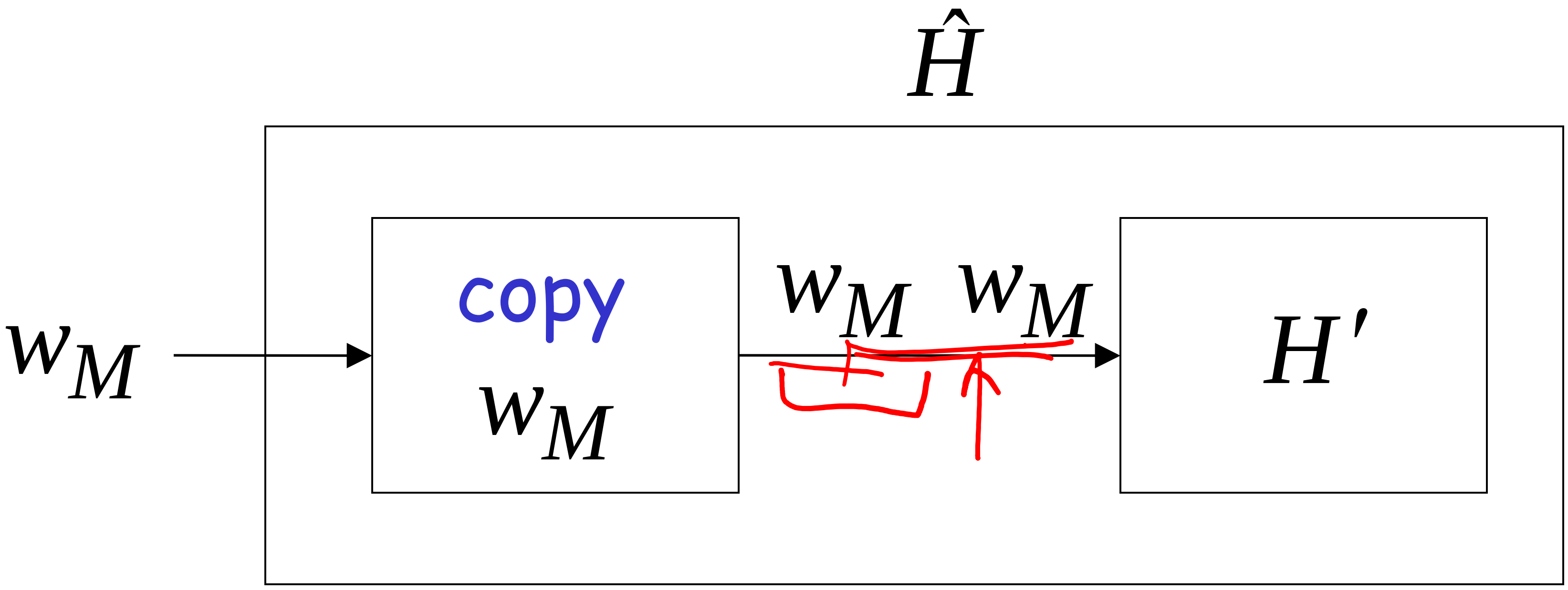
Construct machine $\hat{H}$:

Input: w_M (machine M)

If M halts on input w_M

<u>Then</u> loop forever

<u>Else</u> halt



$\hat{H}$
w_M
copy
w_M
w_M w_M
H'

Run machine $\hat{H}$ with input itself:

Input: $w_{\hat{H}}$ (machine $\hat{H}$)

If $\hat{H}$ halts on input $w_{\hat{H}}$

<u>**Then**</u> loop forever

<u>**Else**</u> halt

$\hat{H}$ on input $w_{\hat{H}}$:

If $\hat{H}$ halts then loops forever

If $\hat{H}$ doesn't halt then it halts

NONSENSE !!!!!

Therefore, we have contradiction

The halting problem is undecidable

END OF PROOF

Another proof of the same theorem:

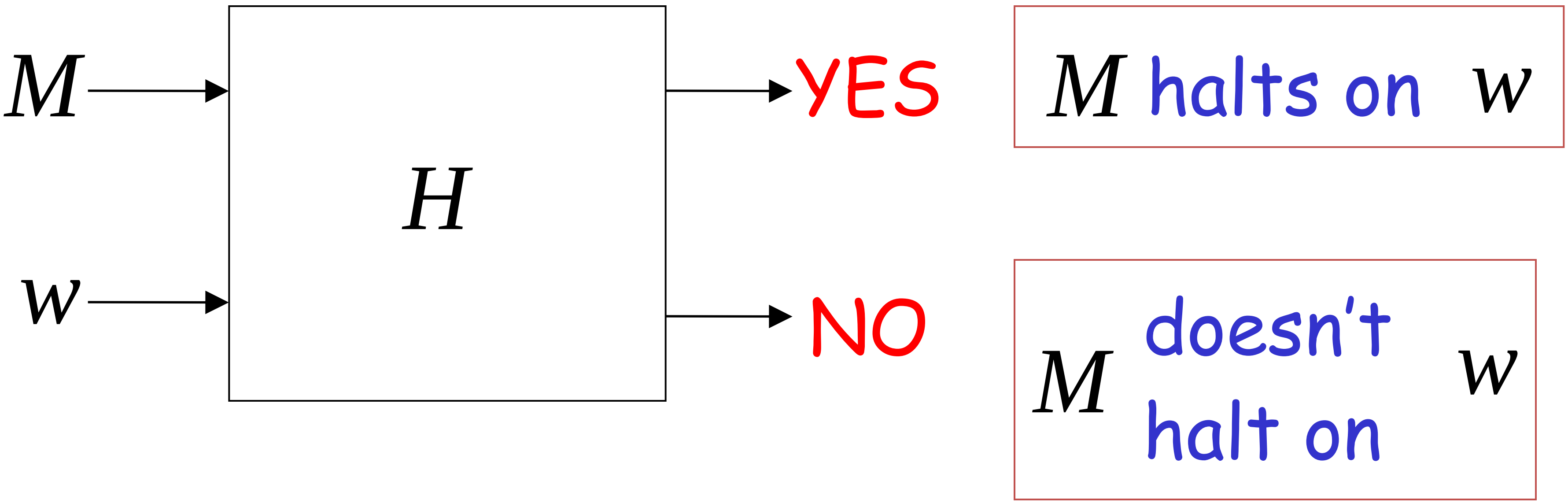
If the halting problem was decidable then every recursively enumerable language would be recursive

Theorem:

The halting problem is undecidable

Proof: Assume for contradiction that
the halting problem is decidable

There exists Turing Machine H that solves the halting problem



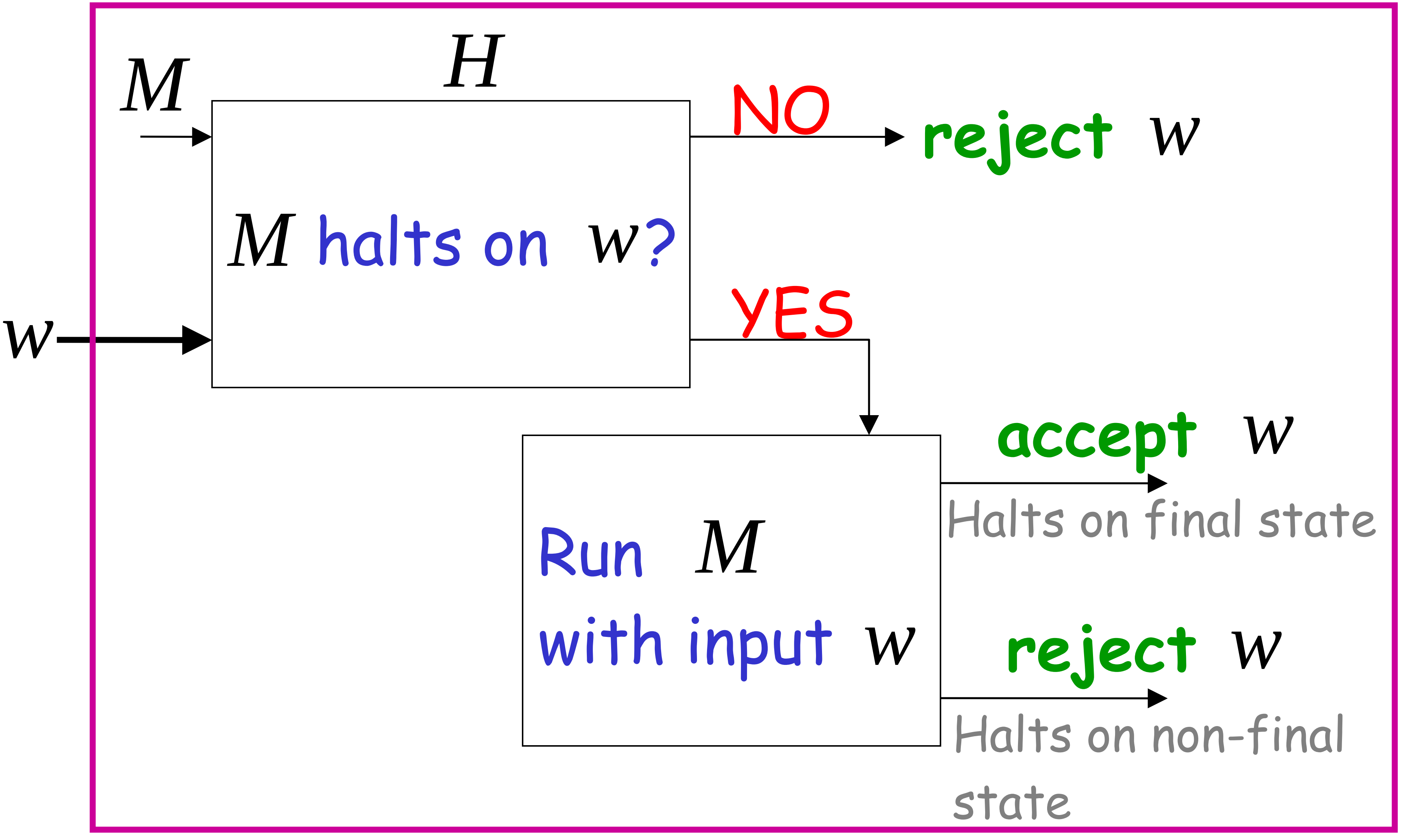
et L be a recursively enumerable language

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

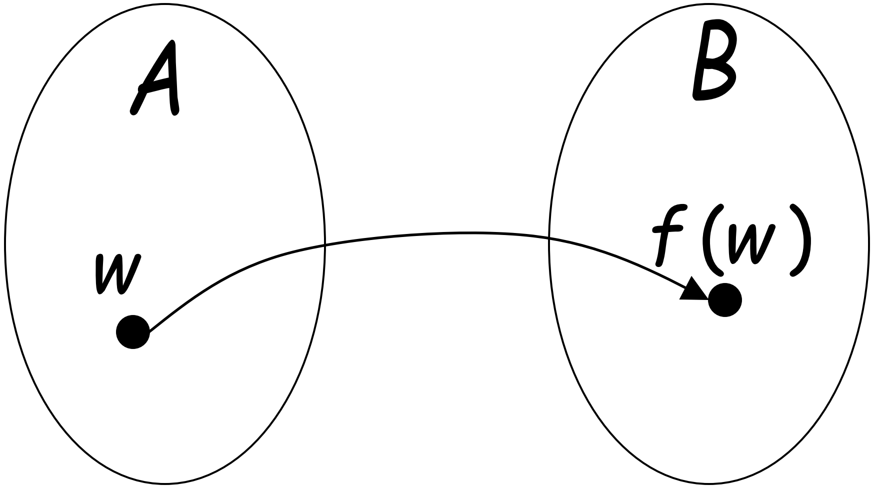
Problem X is reduced to problem Y

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:

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

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

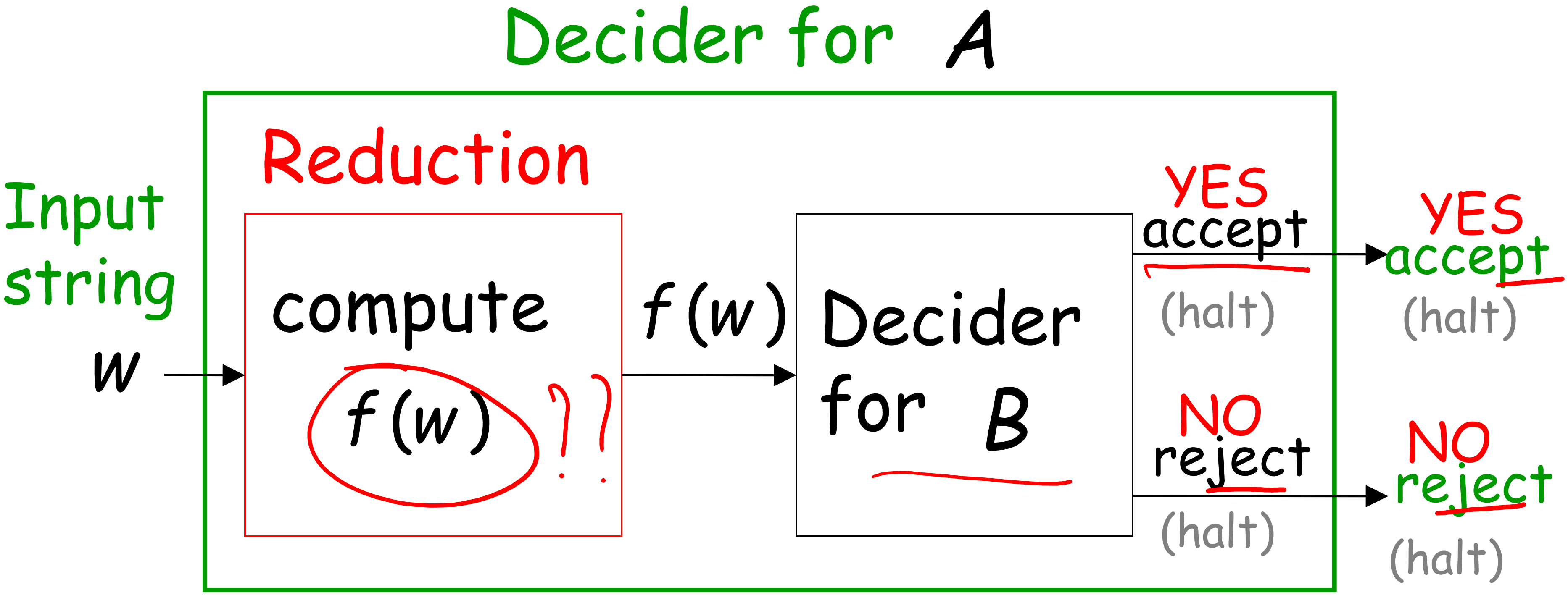
b: Language B is decidable

Then: A is decidable

Proof:

Basic idea:

Build the decider for A using the decider for B



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

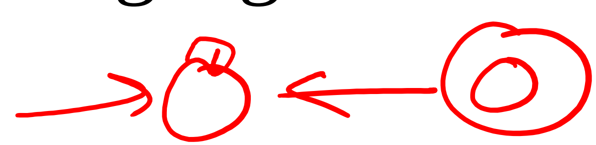
END OF PROOF

Example:

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

is reduced to:

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



We only need to construct:

$$\langle M_1, M_2 \rangle \longrightarrow \boxed{\text{Turing Machine for reduction } f} \longrightarrow f(\langle M_1, M_2 \rangle) = \langle M \rangle \text{ DFA}$$

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

Let L_1 be the language of DFA M_1

Let L_2 be the language of DFA M_2

$$\langle M_1, M_2 \rangle \longrightarrow \boxed{\text{Turing Machine for reduction } f} \longrightarrow f(\langle M_1, M_2 \rangle) = \langle M \rangle \text{ DFA}$$

construct DFA M

by combining M_1 and M_2 so that:

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

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

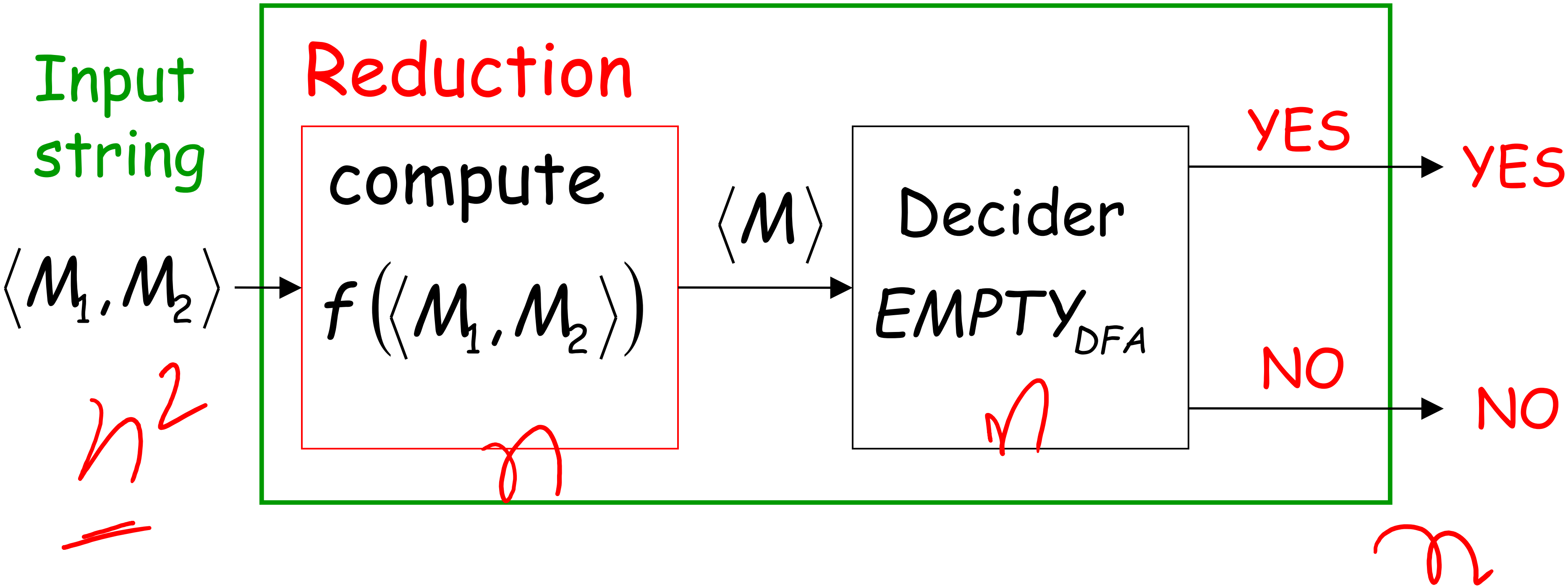
$$\Updownarrow$$

$$L_1 = L_2 \quad \Leftrightarrow \quad L(M) = \emptyset$$

$$\Updownarrow$$

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

Decider for $EQUAL_{DFA}$



Theorem (version 1):

If: a: Language A is reduced to B

b: Language A is undecidable

Then: B is undecidable

(this is the negation of the previous theorem)

Proof: Suppose B is decidable

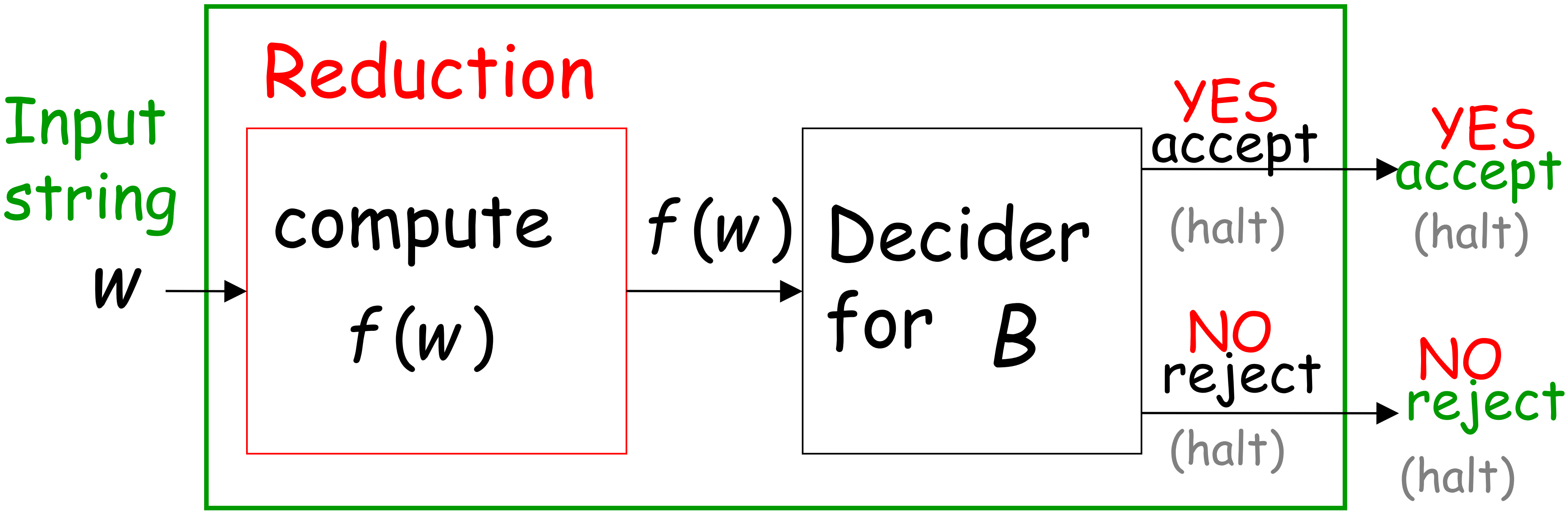
Using the decider for B

build the decider for A

Contradiction!

If B is decidable then we can build:

Decider for A



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

CONTRADICTION!

END OF PROOF

Observation:

In order to prove
that some language B is undecidable

we only need to reduce a
known undecidable language A to B

State-entry problem

Input:
- Turing Machine M
- State q
- String w

Question: Does M enter state q while processing input string w ?

Corresponding language:

$$STATE_{TM} = \{\langle M, w, q \rangle : M \text{ is a Turing machine that enters state } q \text{ on input string } w\}$$

Theorem: $STATE_{TM}$ is undecidable

(state-entry problem is unsolvable)

Proof: Reduce

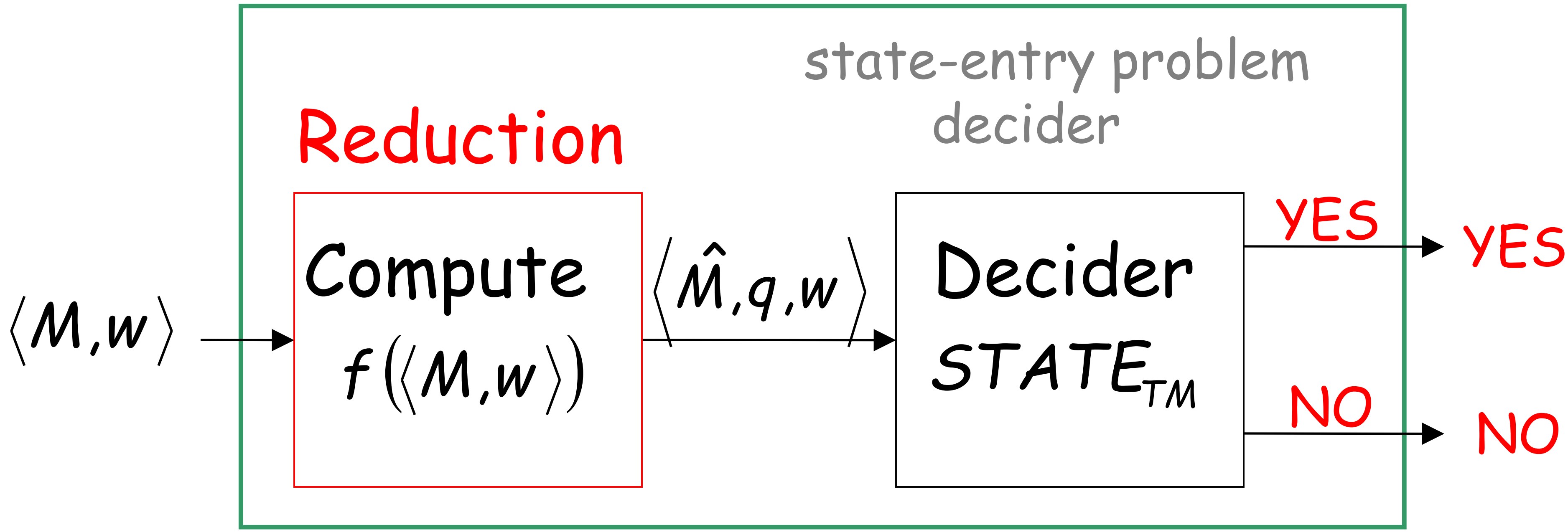
$HALT_{TM}$ (halting problem)

to

$STATE_{TM}$ (state-entry problem)

Halting Problem Decider

Decider for $HALT_{TM}$



Given the reduction,
if $STATE_{TM}$ is decidable,
then $HALT_{TM}$ is decidable

A contradiction!
since $HALT_{TM}$
is undecidable

We only need to build the reduction:

Reduction

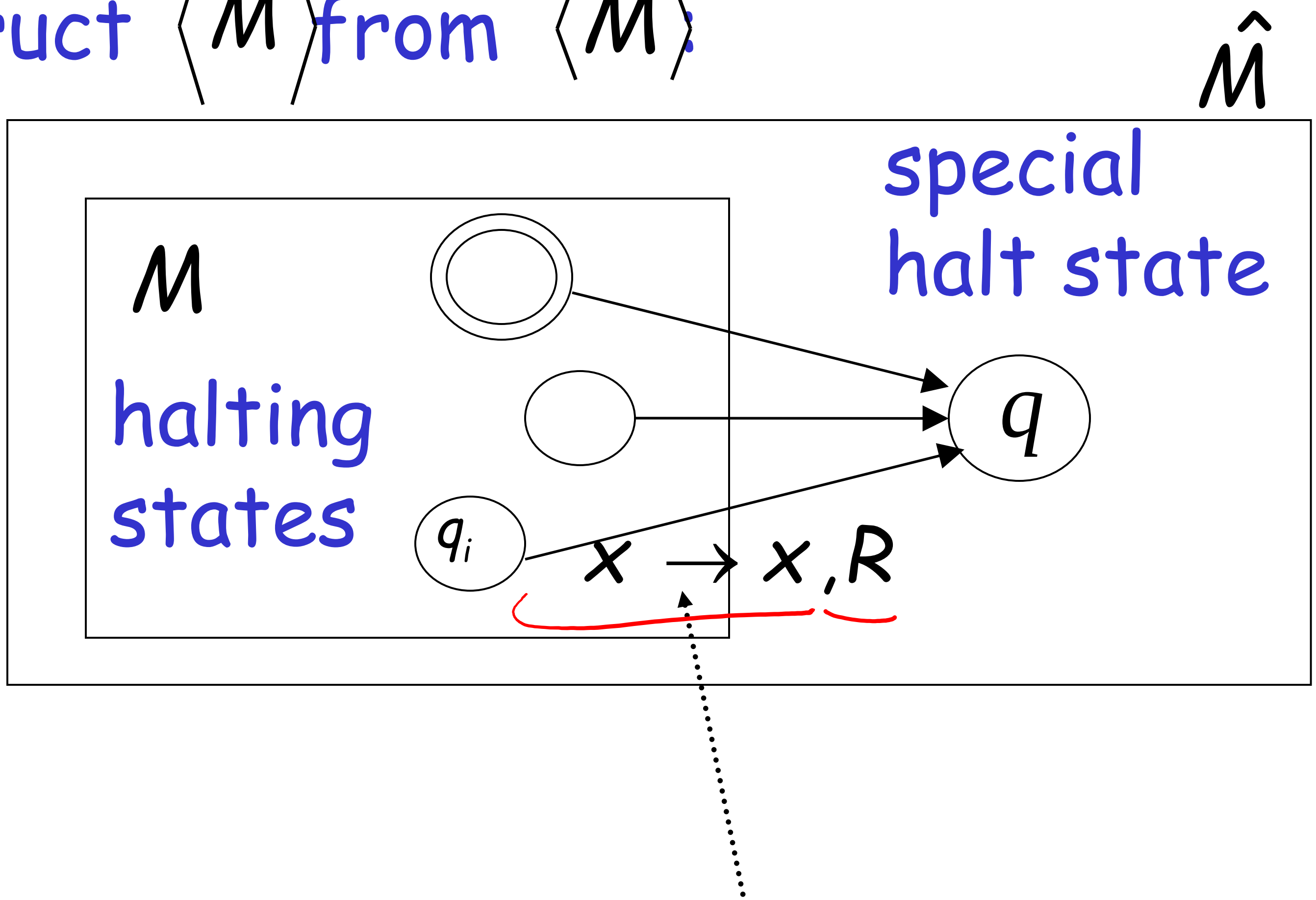
$$\langle M,w\rangle \longrightarrow \boxed{\text{Compute } f(\langle M,w\rangle)} \longrightarrow \langle \hat{M},q,w\rangle$$

$$= f(\langle M,w\rangle)$$

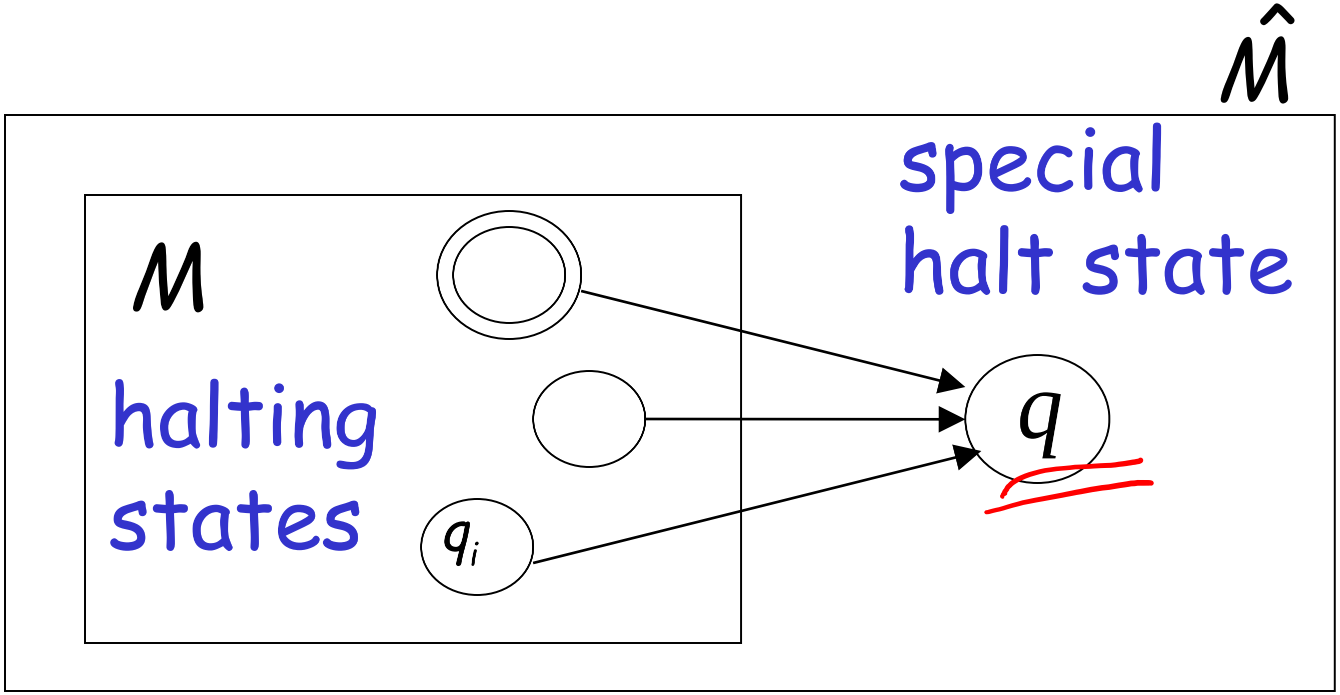
So that:

$$\langle M,w\rangle \in HALT_{TM} \iff \langle \hat{M},w,q\rangle \in STATE_{TM}$$

Construct $\langle \hat{M} \rangle$ from $\langle M \rangle$:



A transition for every unused tape symbol x of q_i



M halts $\Longleftrightarrow$ $\hat{M}$ halts on state q

Therefore: M halts on input w

$$\Updownarrow$$

$\hat{M}$ halts on state q on input w

Equivalently:

$$\langle M, w \rangle \in HALT_{TM} \iff \langle \hat{M}, w, q \rangle \in STATE_{TM}$$

END OF PROOF

Blank-tape halting problem

Input: Turing Machine M

Question: Does M halt when started with a blank tape?

Corresponding language:

$$BLANK_{TM} = \{\langle M \rangle : M \text{ is a Turing machine that halts when started on blank tape}\}$$

Theorem: $BLANK_{TM}$ is undecidable

(blank-tape halting problem is unsolvable)

Proof: Reduce

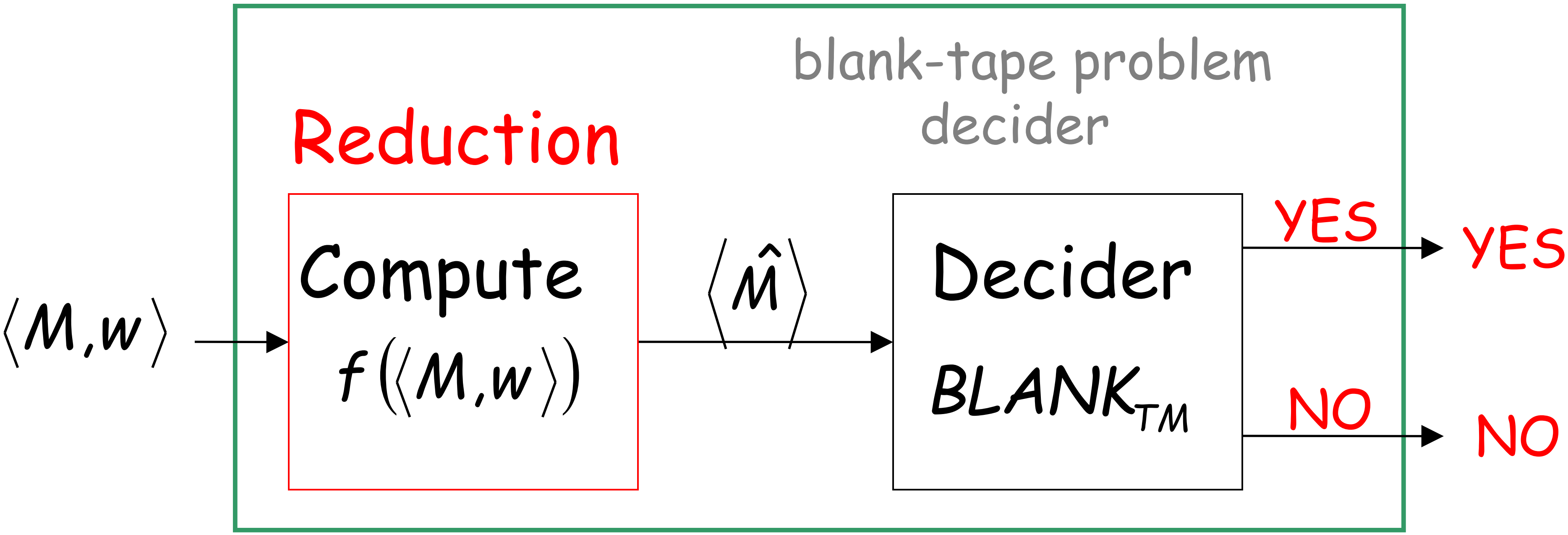
$HALT_{TM}$ (halting problem)

to

$BLANK_{TM}$ (blank-tape problem)

Halting Problem Decider

Decider for $HALT_{TM}$



Given the reduction,
If $BLANK_{TM}$ is decidable,
then $HALT_{TM}$ is decidable

A contradiction!
since $HALT_{TM}$
is undecidable

We only need to build the reduction:

Reduction

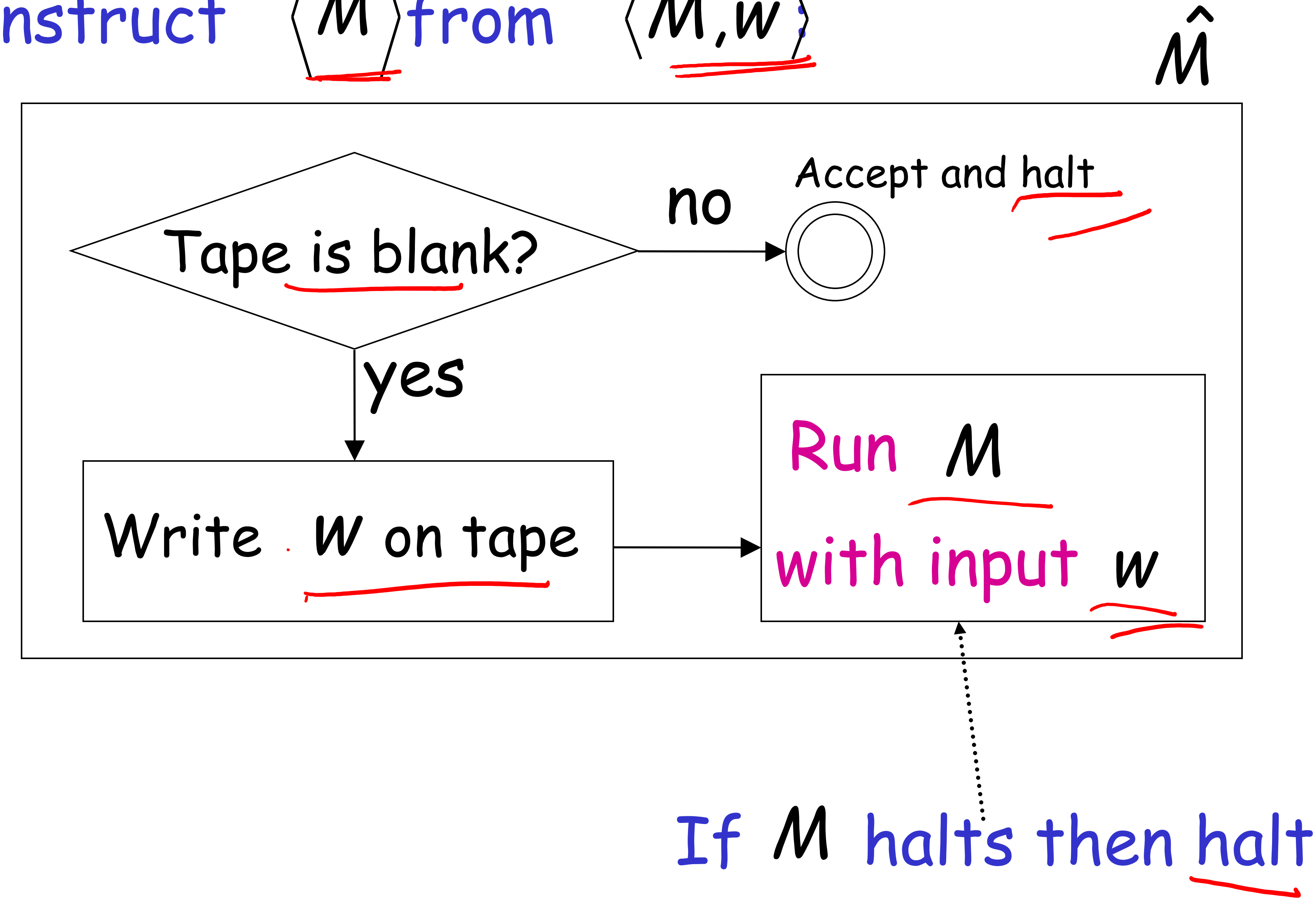
$$\langle M,w \rangle \longrightarrow \boxed{\text{Compute } f(\langle M,w \rangle)} \longrightarrow \langle \hat{M} \rangle$$

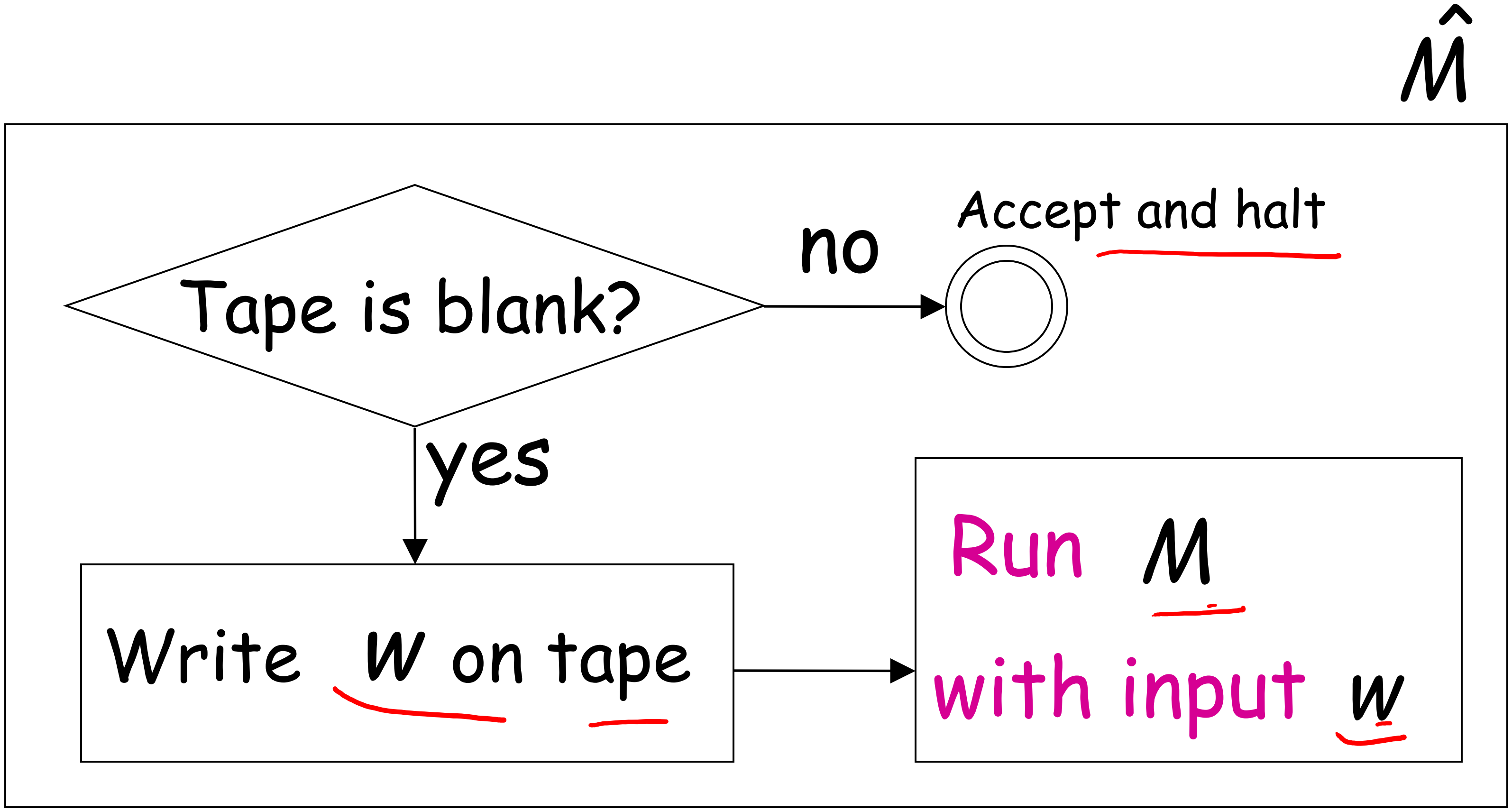
$$= f(\langle M,w \rangle)$$

So that:

$$\langle M,w \rangle \in HALT_{TM} \iff \langle \hat{M} \rangle \in BLANK_{TM}$$

Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:





M halts on input w

$\Updownarrow$

$\hat{M}$ halts when started on blank tape

M halts on input w

$$\Updownarrow$$

$\hat{M}$ halts when started on blank tape

Equivalently:

$$\langle M, w \rangle \in HALT_{TM} \quad \Longleftrightarrow \quad \langle \hat{M} \rangle \in BLANK_{TM}$$

END OF PROOF

Theorem (version 2):

If: a: Language A is reduced to $\overline{B}$

b: Language A is undecidable

Then: B is undecidable

Proof: Suppose B is decidable

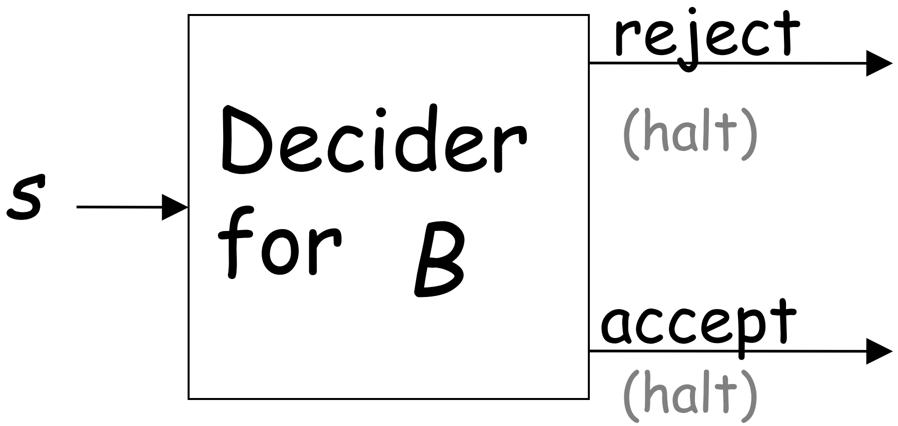
Then $\overline{B}$ is decidable

Using the decider for $\overline{B}$

build the decider for A

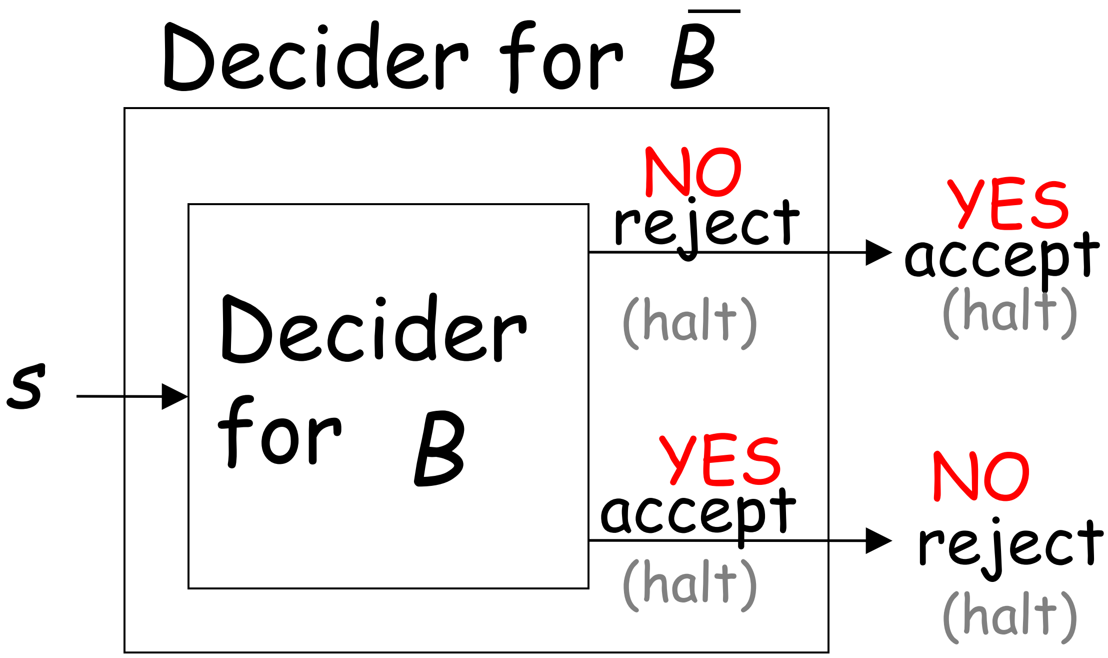
Contradiction!

Suppose B is decidable

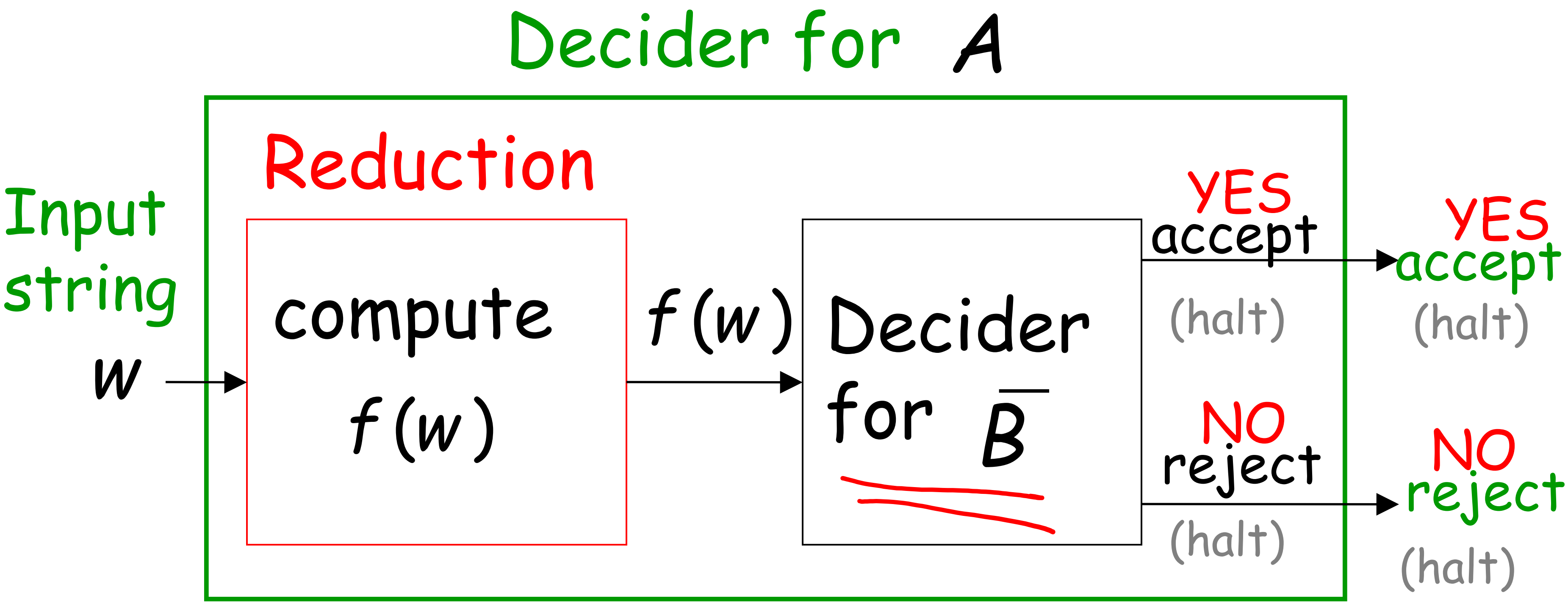


Suppose B is decidable

Then $\overline{B}$ is decidable



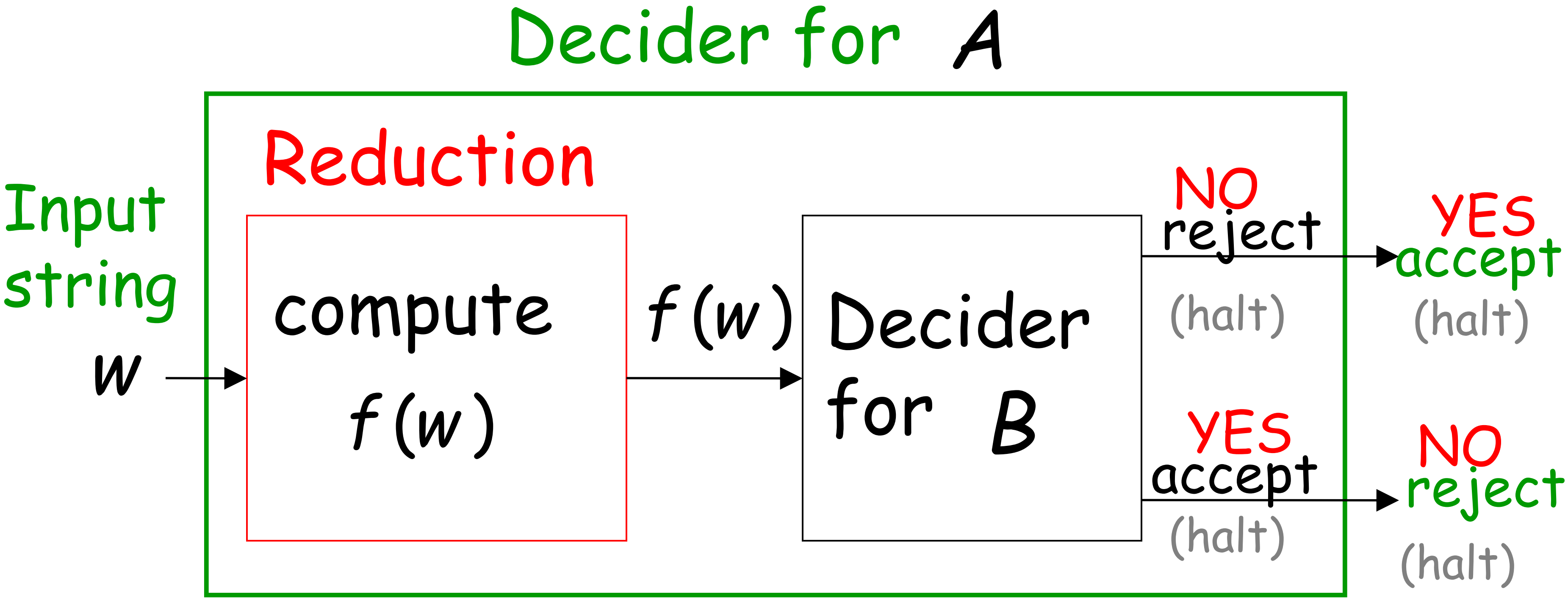
If $\overline{B}$ is decidable then we can build:



$$w \in A \Leftrightarrow f(w) \in \overline{B}$$

CONTRADICTION!

Alternatively:



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

CONTRADICTION!

END OF PROOF

Observation:

In order to prove
that some language B is undecidable
we only need to reduce some
known undecidable language A
to B (theorem version 1)
or to $\overline{B}$ (theorem version 2)

Undecidable Problems for Turing Recognizable languages

Let L be a Turing-acceptable language

- L is empty?
- L is regular?
- L has size 2?

All these are undecidable problems

Let L be a Turing-acceptable language

- L is empty?
- L is regular?
- L has size 2?

Empty language problem

Input: Turing Machine M

Question: Is $L(M)$ empty? $\quad L(M) = \emptyset ?$

Corresponding language:

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

Theorem: $EMPTY_{TM}$ is undecidable

(empty-language problem is unsolvable)

Proof: Reduce

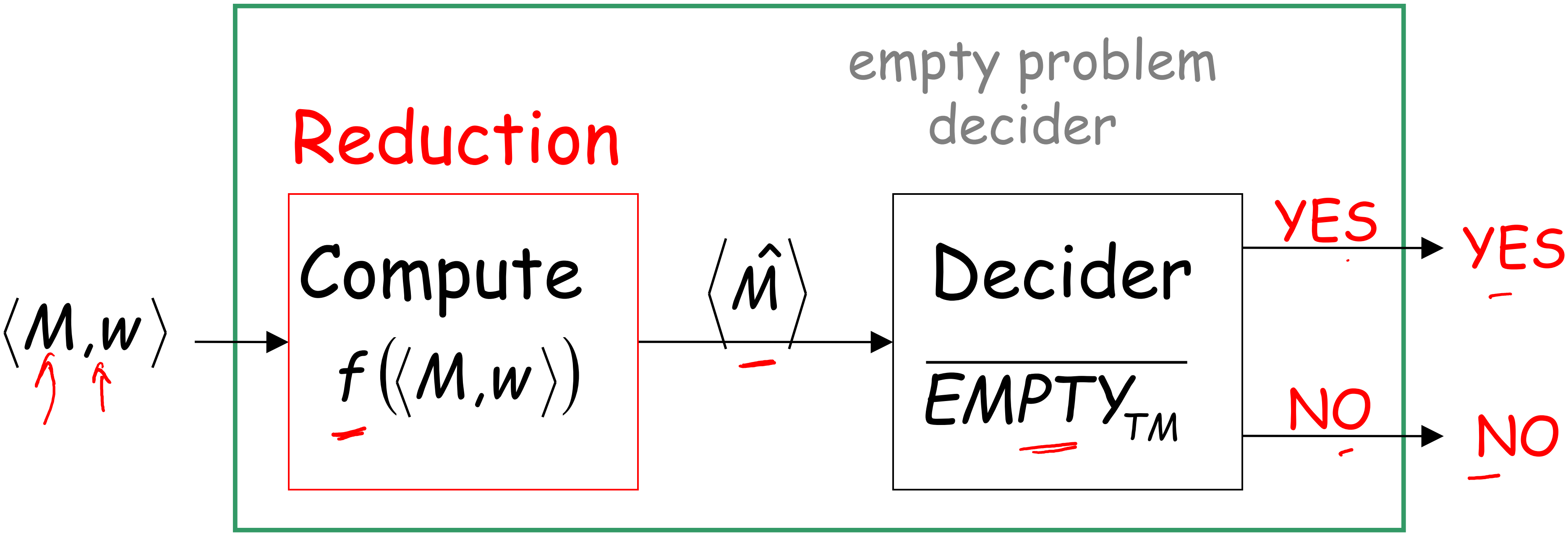
$\underline{A_{TM}}$ (membership problem)

to

$\underline{\overline{EMPTY_{TM}}}$ (empty language problem)

membership problem decider

Decider for A_{TM}



Given the reduction,
if $\overline{EMPTY_{TM}}$ is decidable,
then A_{TM} is decidable

A contradiction!
since A_{TM}
is undecidable

We only need to build the reduction:

Reduction

$$\langle M,w \rangle \longrightarrow \boxed{\text{Compute } f(\langle M,w \rangle)} \longrightarrow \langle \hat{M} \rangle$$

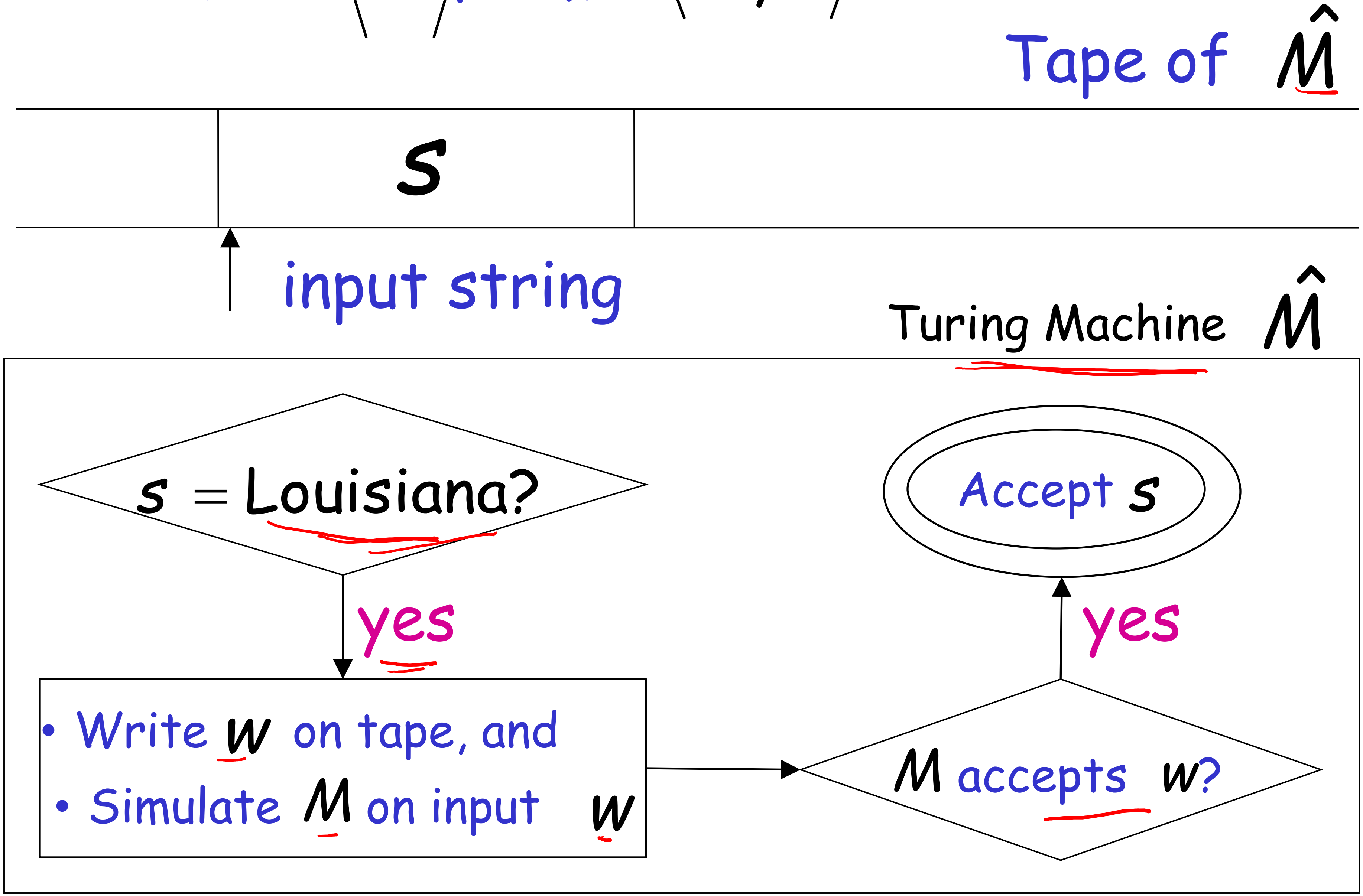
??

$$= f(\langle M,w \rangle)$$

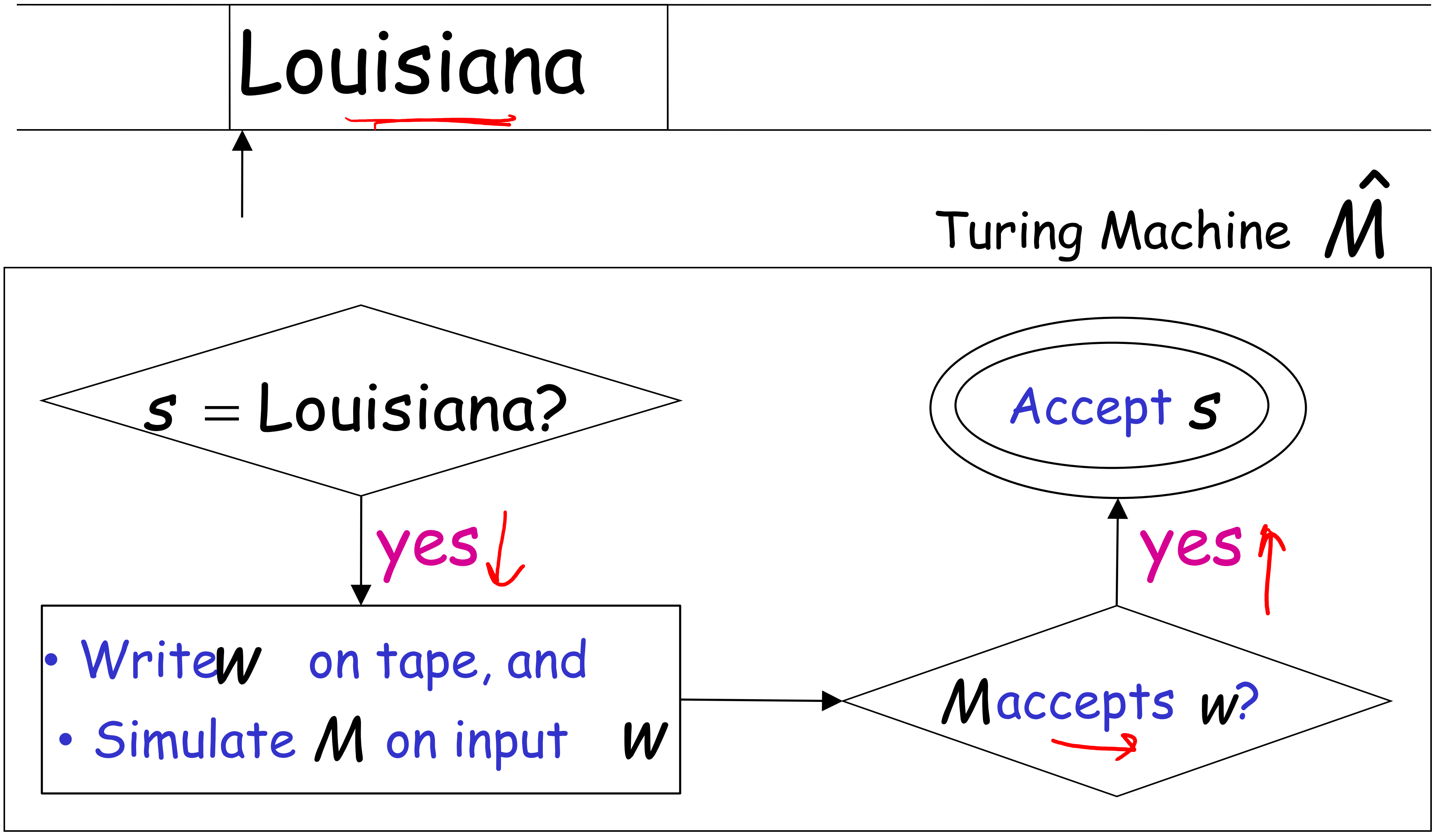
So that:

$$\langle M,w \rangle \in A_{TM} \iff \langle \hat{M} \rangle \in \overline{EMPTY_{TM}}$$

Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:



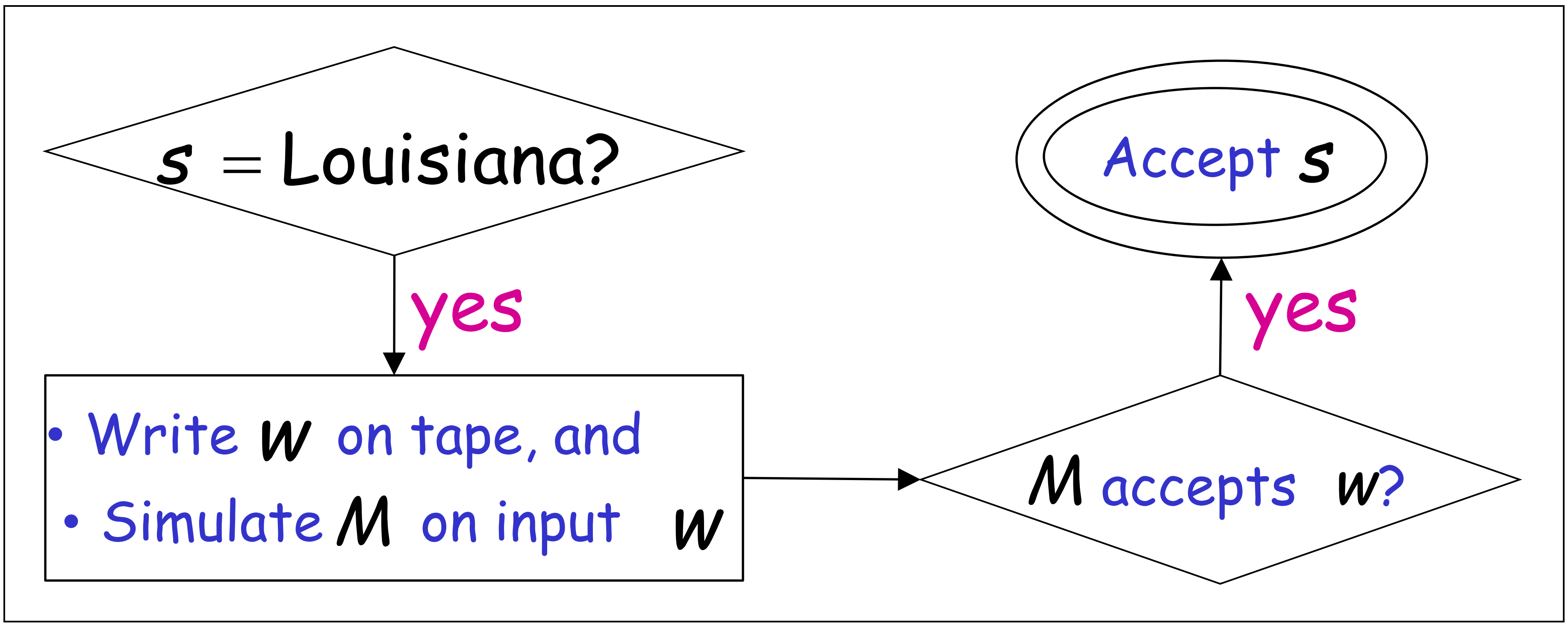
The only possible accepted string s



M accepts w $\Longrightarrow$ $L(\hat{M}) = \{\text{Louisiana}\} \neq \emptyset$

M does not accept w $\Longrightarrow$ $L(\hat{M}) = \emptyset$

Turing Machine $\hat{M}$



Therefore:

M accepts w $\Longleftrightarrow$ $L(\hat{M}) \neq \emptyset$

Equivalently:

$$\langle M, w \rangle \in A_{TM} \Longleftrightarrow \langle \hat{M} \rangle \in \overline{EMPTY_{TM}}$$

END OF PROOF

Let L be a Turing-acceptable language

- L is empty?
- L is regular?
- L has size 2?

Regular language problem

Input: Turing Machine M

Question: Is $L(M)$ a regular language?

Corresponding language:

$$REGULAR_{TM} = \{\langle M \rangle : M \text{ is a Turing machine that accepts a regular language}\}$$

Theorem: $REGULAR_{TM}$ is undecidable

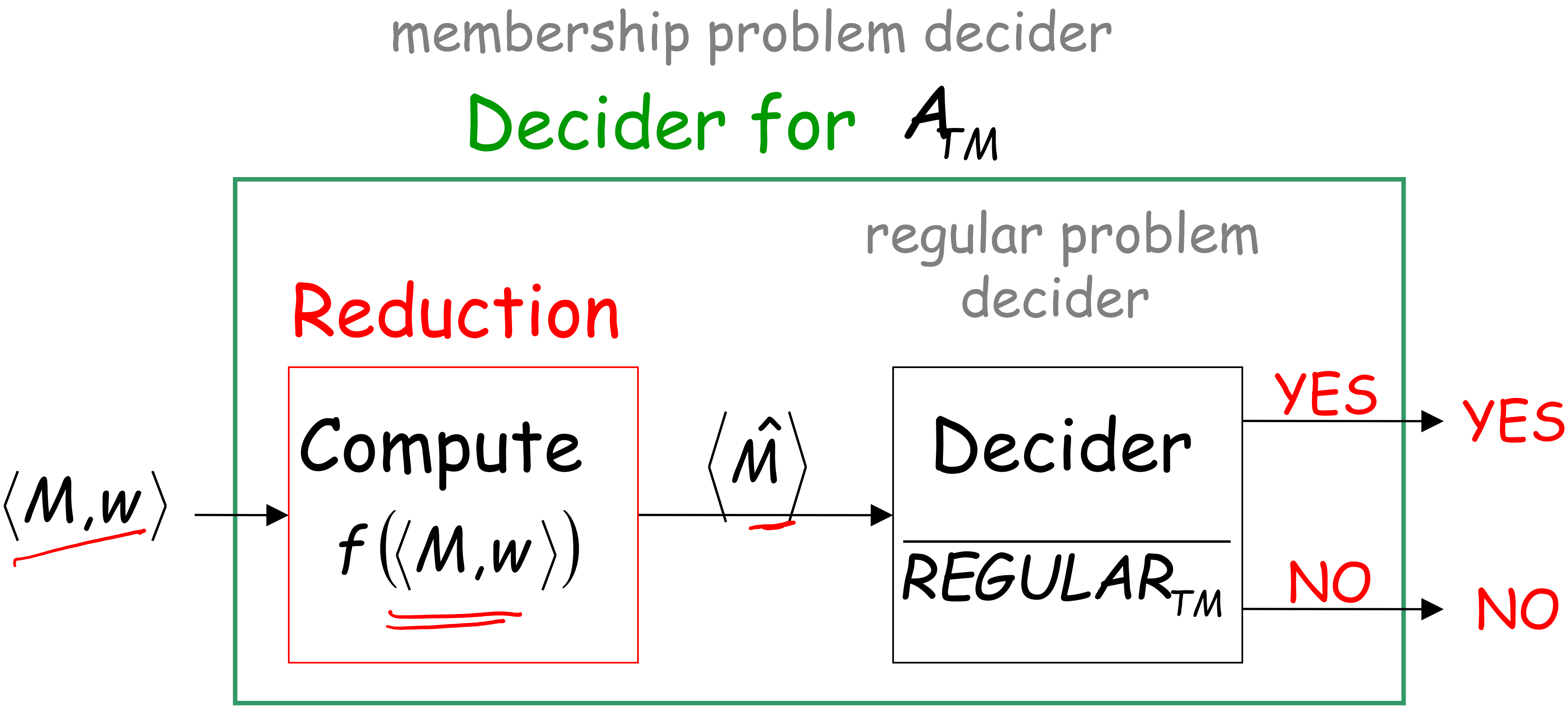
(regular language problem is unsolvable)

Proof: Reduce

$\rightarrow A_{TM}$ (membership problem)

to

$\rightarrow \overline{REGULAR_{TM}}$ (regular language problem)



Given the reduction,
If $\overline{REGULAR_{TM}}$ is decidable,
then A_{TM} is decidable

A contradiction!
since A_{TM}
is undecidable

We only need to build the reduction:

Reduction

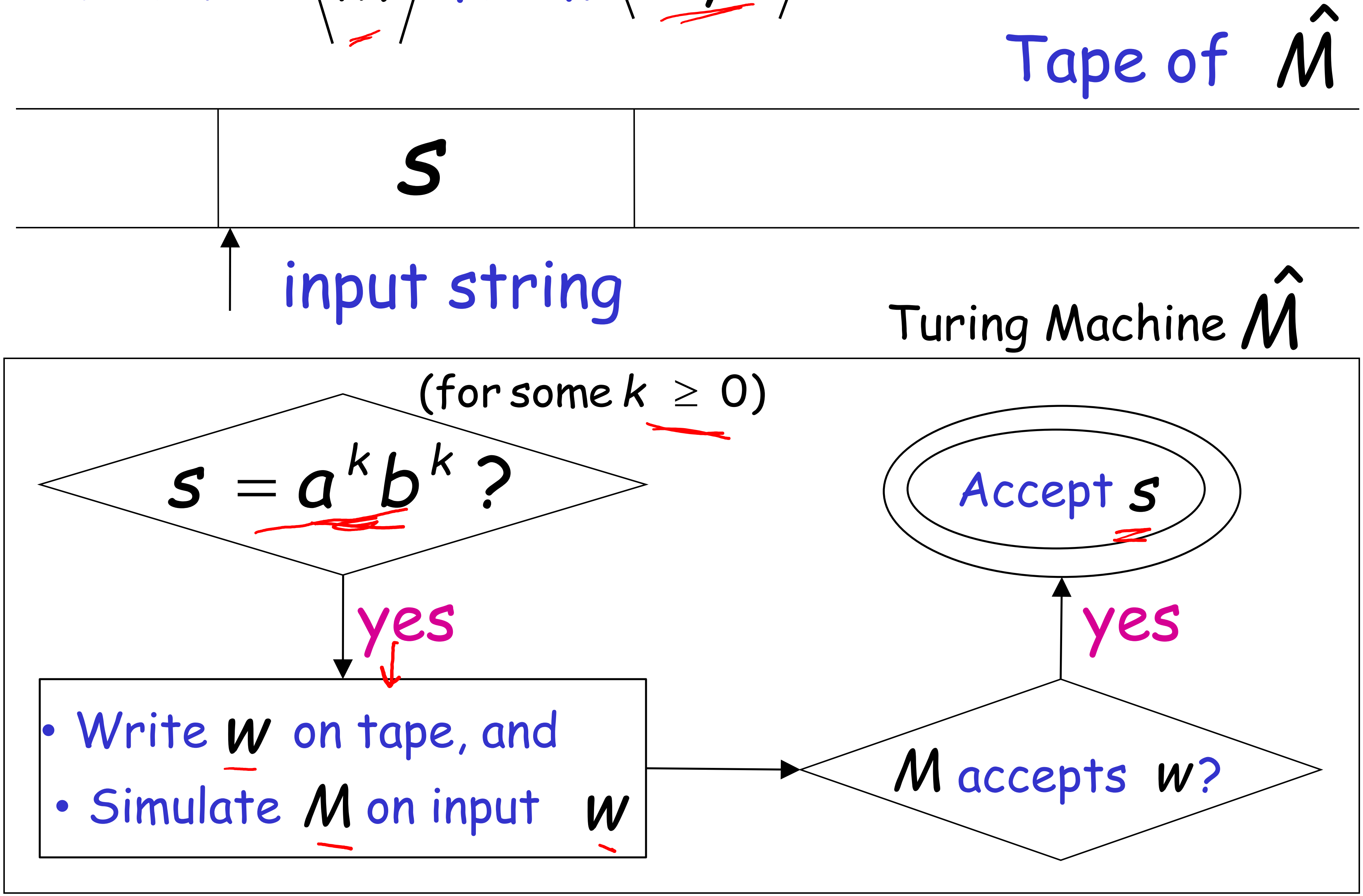
$$\langle M,w\rangle \longrightarrow \boxed{\begin{array}{c}\text{Compute}\\ f(\langle M,w\rangle)\end{array}} \longrightarrow \langle \hat{M}\rangle$$

$$= f(\langle M,w\rangle)$$

So that:

$$\langle M,w\rangle \in AT_{TM} \iff \langle \hat{M}\rangle \in \overline{REGULAR_{TM}}$$

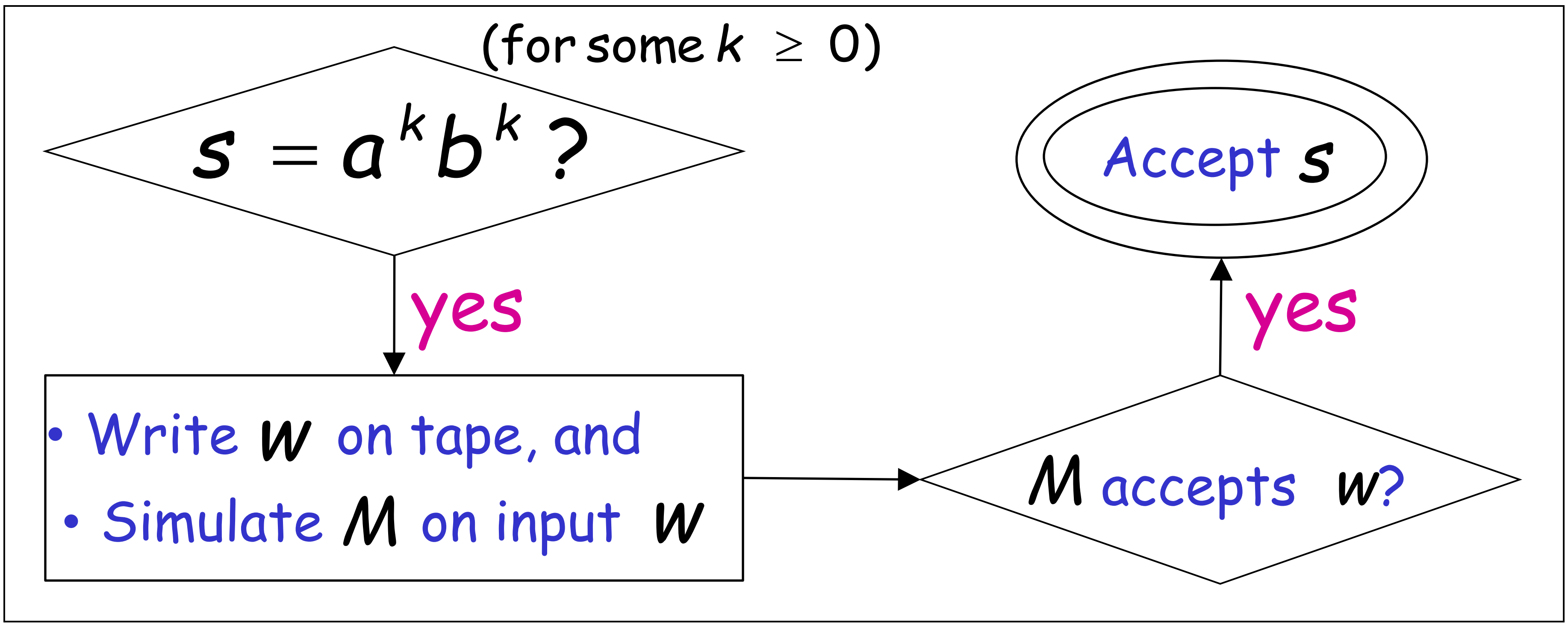
Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:



M accepts w $\Rightarrow$ $L(\hat{M}) = \{a^n b^n : n \geq 0\}$ not regular

M does not accept w $\Rightarrow$ $L(\hat{M}) = \emptyset$ regular

Turing Machine $\hat{M}$



Therefore:

M accepts w $\Longleftrightarrow$ $L(\hat{M})$ is not regular

Equivalently:

$$\langle M,w \rangle \in A_{TM} \Longleftrightarrow \langle \hat{M} \rangle \in \overline{REGULAR_{TM}}$$

END OF PROOF

Let L be a Turing-acceptable language

- L is empty?
- L is regular?
- L has size 2?

Size2 language problem

Input: Turing Machine M

Question: Does $L(M)$ have size 2 (two strings)?

$$|L(M)| = 2?$$

Corresponding language:

$$SIZE2_{TM} = \{\langle M \rangle : M \text{ is a Turing machine that accepts exactly two strings}\}$$

Theorem: $SIZE2_{TM}$ is undecidable

(size2 language problem is unsolvable)

Proof: Reduce

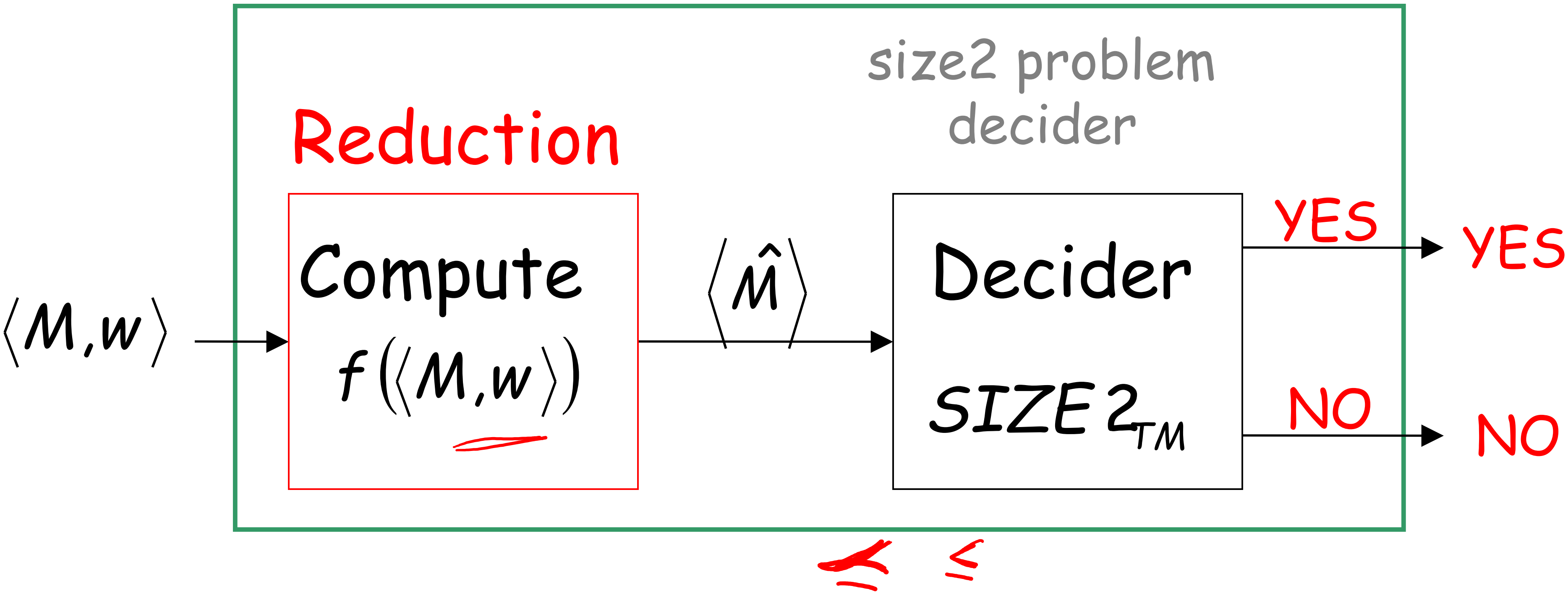
A_{TM} (membership problem)

to

$SIZE2_{TM}$ (size 2 language problem)

membership problem decider

Decider for A_{TM}



Given the reduction,
If $SIZE2_{TM}$ is decidable,
then A_{TM} is decidable

A contradiction!
since A_{TM}
is undecidable

We only need to build the reduction:

Reduction

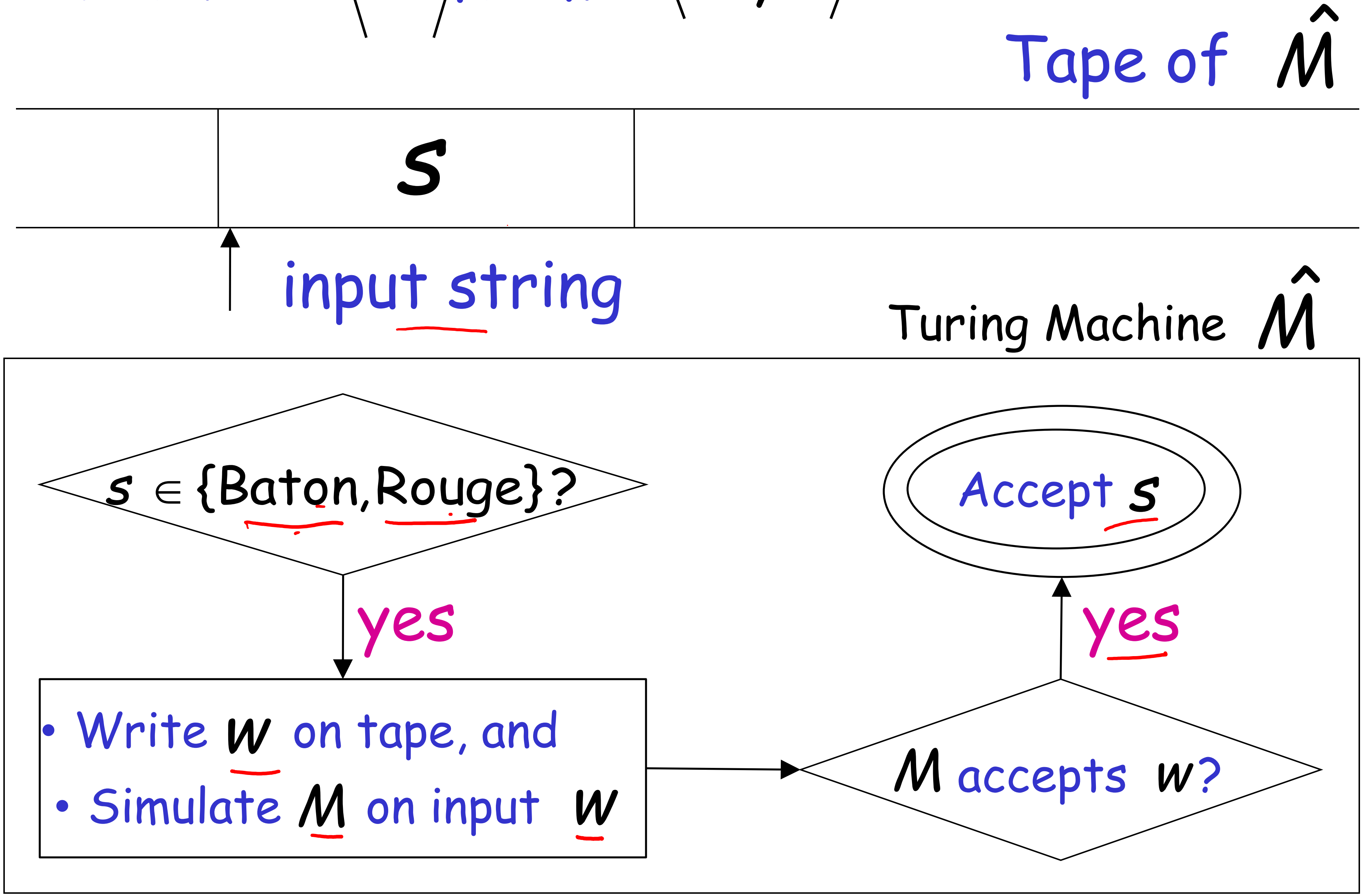
$$\langle M,w\rangle \longrightarrow \boxed{\begin{array}{c}\text{Compute}\\ f(\langle M,w\rangle)\end{array}} \longrightarrow \langle \hat{M}\rangle$$

$$= f(\langle M,w\rangle)$$

So that:

$$\langle M,w\rangle \in A_{TM} \quad \Longleftrightarrow \quad \langle \hat{M}\rangle \in SIZE2_{TM}$$

Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:

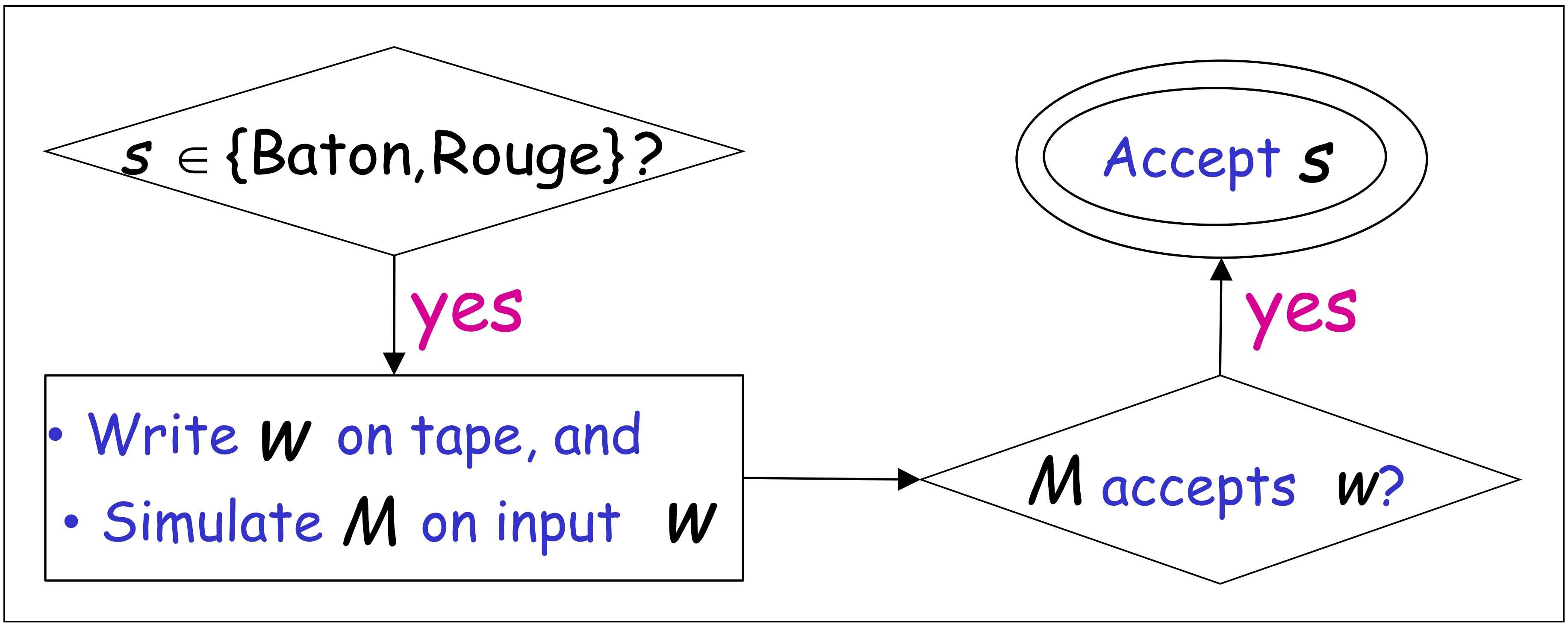


2 strings

M accepts w $\Rightarrow$ $L(\hat{M}) = \{\text{Baton}, \text{Rouge}\}$

M does not accept w $\Rightarrow$ $L(\hat{M}) = \varnothing$ 0 strings

Turing Machine $\hat{M}$



Therefore:

$$M \text{ accepts } w \iff L(\hat{M}) \text{ has size } 2$$

Equivalently:

$$\langle M, w \rangle \in A_{TM} \iff \langle \hat{M} \rangle \in SIZE2_{TM}$$

END OF PROOF

RICE's Theorem

Undecidable problems:

- L is empty?
- L is regular?
- L has size 2?

This can be generalized to all non-trivial properties of Turing-acceptable languages

Non-trivial property:

A property P possessed by some Turing-acceptable languages but not all

Example: P_1 : L is empty?

YES $L = \emptyset$ ✓

NO $L = \{Louisiana\}$ ✗

NO $L = \{Baton, Rouge\}$ ✗

More examples of non-trivial properties:

P_2 : L is regular?

YES $L = \emptyset$ ✓

YES $L = \{a^n : n \geq 0\}$ ✓

NO $L = \{a^n b^n : n \geq 0\}$ X

P_3 : L has size 2?

NO $L = \emptyset$ X

NO $L = \{\text{Louisiana}\}$ X

YES $L = \{\text{Baton}, \text{Rouge}\}$ ✓

Trivial property:

A property P possessed by ALL Turing-acceptable languages

Examples: P_4 : L has size at least 0?

True for all languages

P_5 : L is accepted by some Turing machine?

True for all Turing-acceptable languages

We can describe a property P as the set of languages that possess the property

If language L has property P then $\underline{L \in P}$

Example: $\underline{P}$: L is empty?

YES $L_1 = \emptyset$

$P = \{\underline{L_1}\}$

NO $L_2 = \{\text{Louisiana}\}$

NO $L_3 = \{\text{Baton}, \text{Rouge}\}$

Example: Suppose alphabet is $\Sigma = \{a\}$

P : L has size 1?

NO $\emptyset$

YES $\{\lambda\}$ $\{a\}$ $\{aa\}$ $\{aaa\}$ $\cdots$

NO $\{\lambda, a\}$ $\{\lambda, aa\}$ $\{a, aa\}$ $\cdots$

NO $\{\lambda, a, aa\}$ $\{aa, aaa, aaaa\}$ $\cdots$

$$P = \{\{\lambda\}, \{a\}, \{aa\}, \{aaa\}, \{aaaa\}, \ldots\}$$

Non-trivial property problem

Input: Turing Machine M

Question: Does $L(M)$ have the non-trivial property P?

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

Corresponding language:

$$PROPERTY_{TM} = \{\langle M \rangle : M \text{ is a Turing machine such that } L(M) \text{ has the non-trivial property } P \text{, that is, } L(M) \in P\}$$

Rice's Theorem: $PROPERTY_{TM}$ is undecidable

(the non-trivial property problem is unsolvable)

Proof:

Reduce

A_{TM} (membership problem)

to

$PROPERTY_{TM}$ or $\overline{PROPERTY_{TM}}$

We examine two cases:

Case 1: $\emptyset \in P$

Examples: P : $L(M)$ is empty?

P : $L(M)$ is regular?

Case 2: $\emptyset \notin P$

Example: P : $L(M)$ has size 2?

Case 1: $\emptyset \in P$

Since P is non-trivial, there is a Turing-acceptable language X such that: $X \notin P$

Let M_X be the Turing machine that accepts X

Reduce

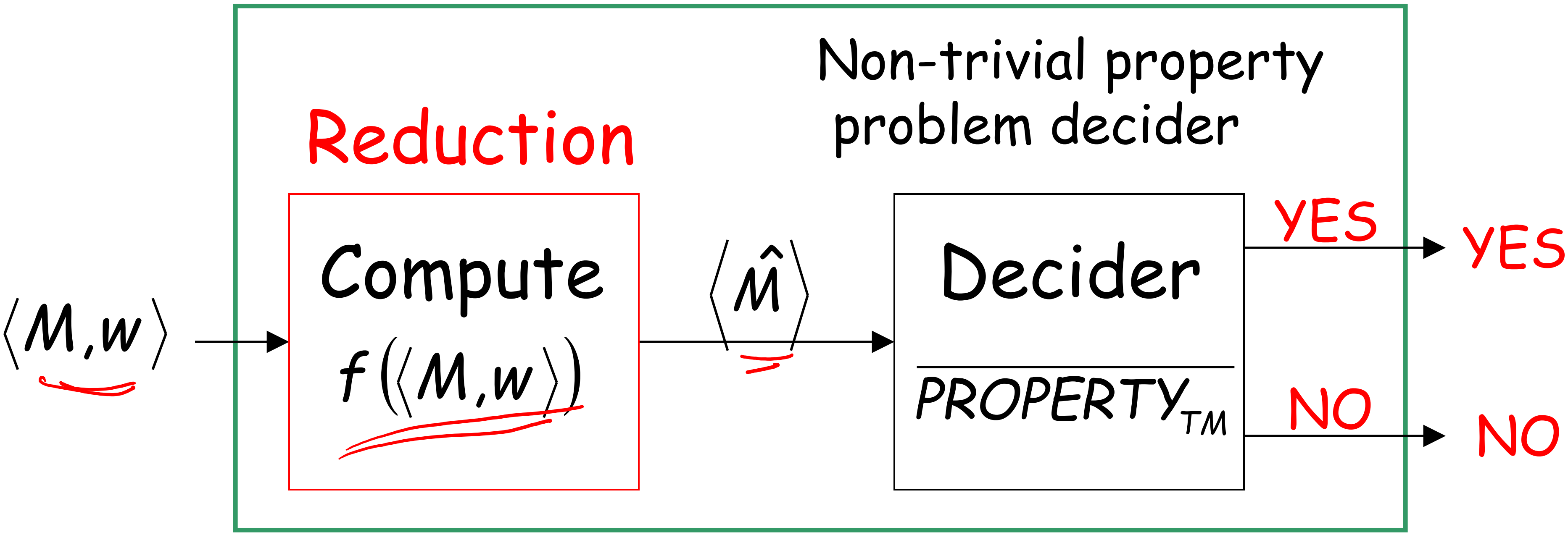
A_{TM} (membership problem)

to

$\overline{PROPERTY_{TM}}$

membership problem decider

Decider for A_{TM}



Given the reduction,
if $\overline{PROPERTY_{TM}}$ is decidable,
then A_{TM} is decidable

A contradiction!
since A_{TM}
is undecidable

We only need to build the reduction:

Reduction

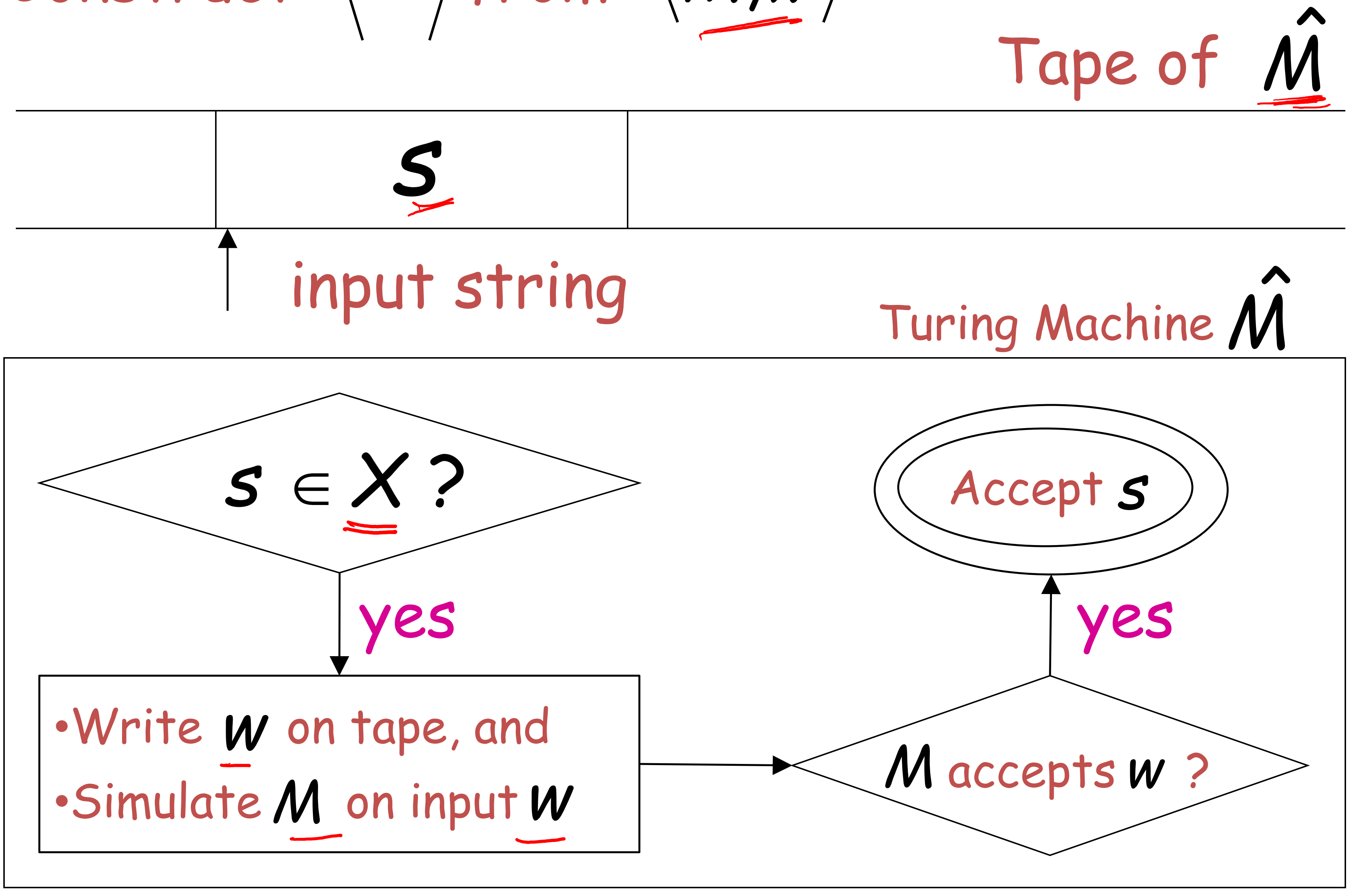
$$\langle M, w\rangle \longrightarrow \boxed{\text{Compute } f(\langle M, w\rangle)} \longrightarrow \langle \hat{M}\rangle$$

$$= f(\langle M, w\rangle)$$

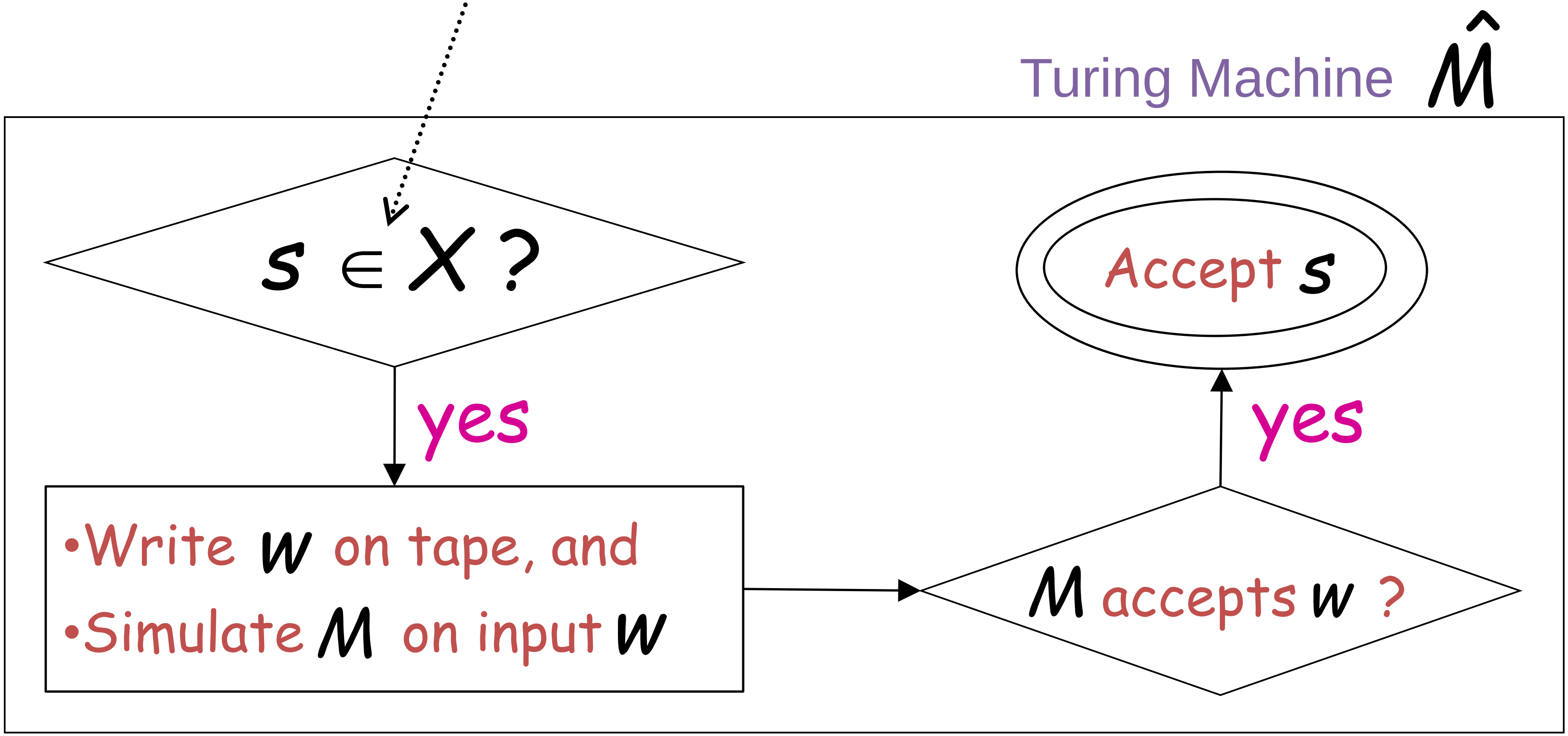
So that:

$$\langle M, w\rangle \in A_{TM} \quad \Longleftrightarrow \quad \langle \hat{M}\rangle \in \overline{PROPERTY_{TM}}$$

Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:



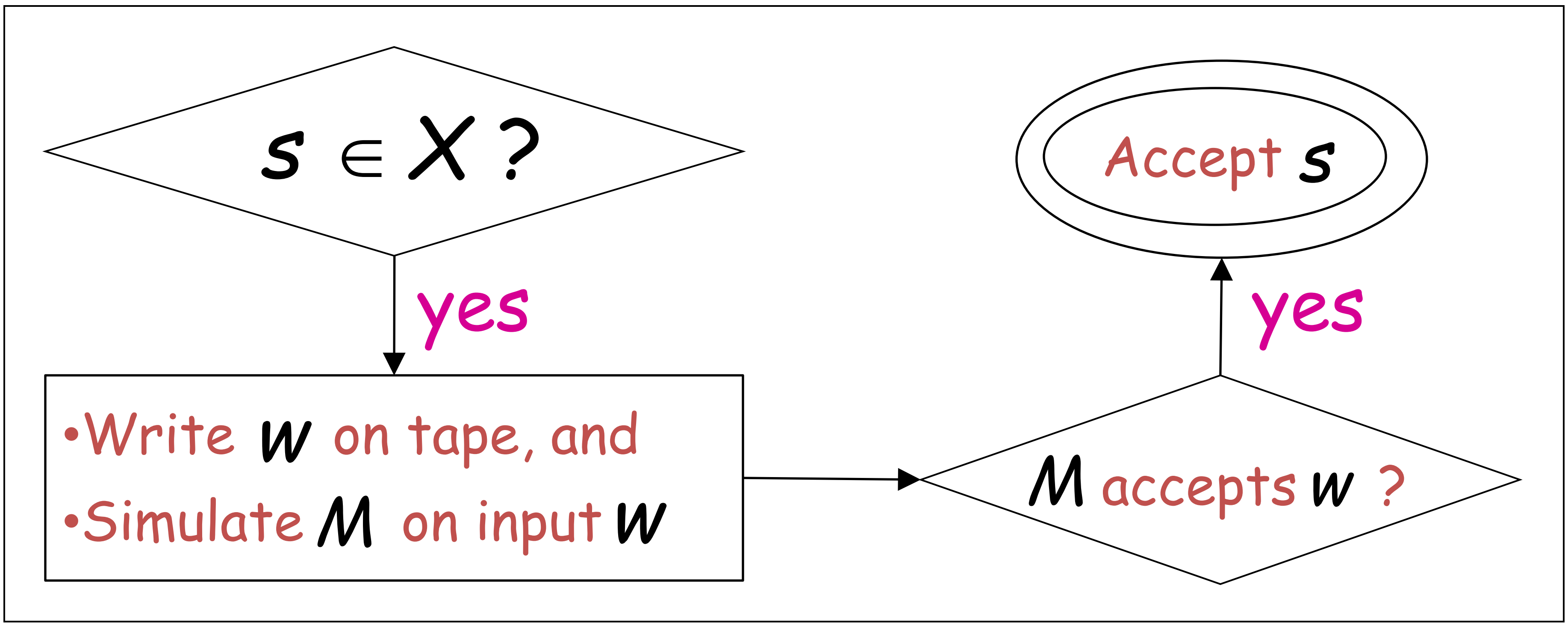
For this we can run machine M_X,
that accepts language X,
with input string s



M accepts w $\Rightarrow$ $L(\hat{M}) = X \notin P$

M does not accept w $\Rightarrow$ $L(\hat{M}) = \emptyset \in P$

Turing Machine $\hat{M}$



Therefore:

M accepts w $\Longleftrightarrow$ $L(\hat{M}) \notin P$

Equivalently:

$\langle M, w \rangle \in A_{TM}$ $\Longleftrightarrow$ $\langle \hat{M} \rangle \in \overline{PROPERTY_{TM}}$

Case 2: $\emptyset \notin P$

Since P is non-trivial, there is a Turing-acceptable language X such that: $X \in P$

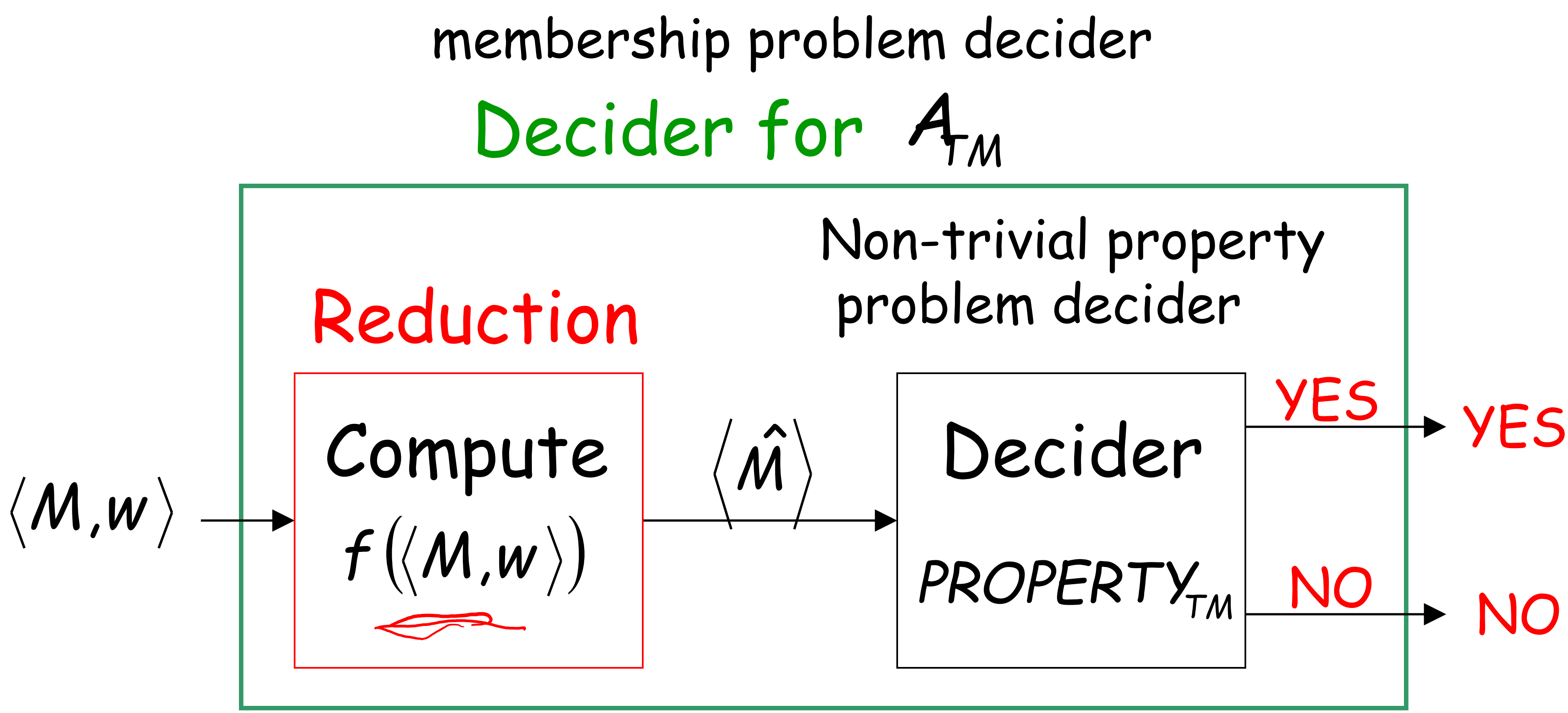
Let M_X be the Turing machine that accepts X

Reduce

A_{TM} (membership problem)

to

$PROPERTY_{TM}$



Given the reduction,
if $PROPERTY_{TM}$ is decidable,
then A_{TM} is decidable

A contradiction!
since $\boxed{A_{TM}}$
is undecidable

We only need to build the reduction:

Reduction

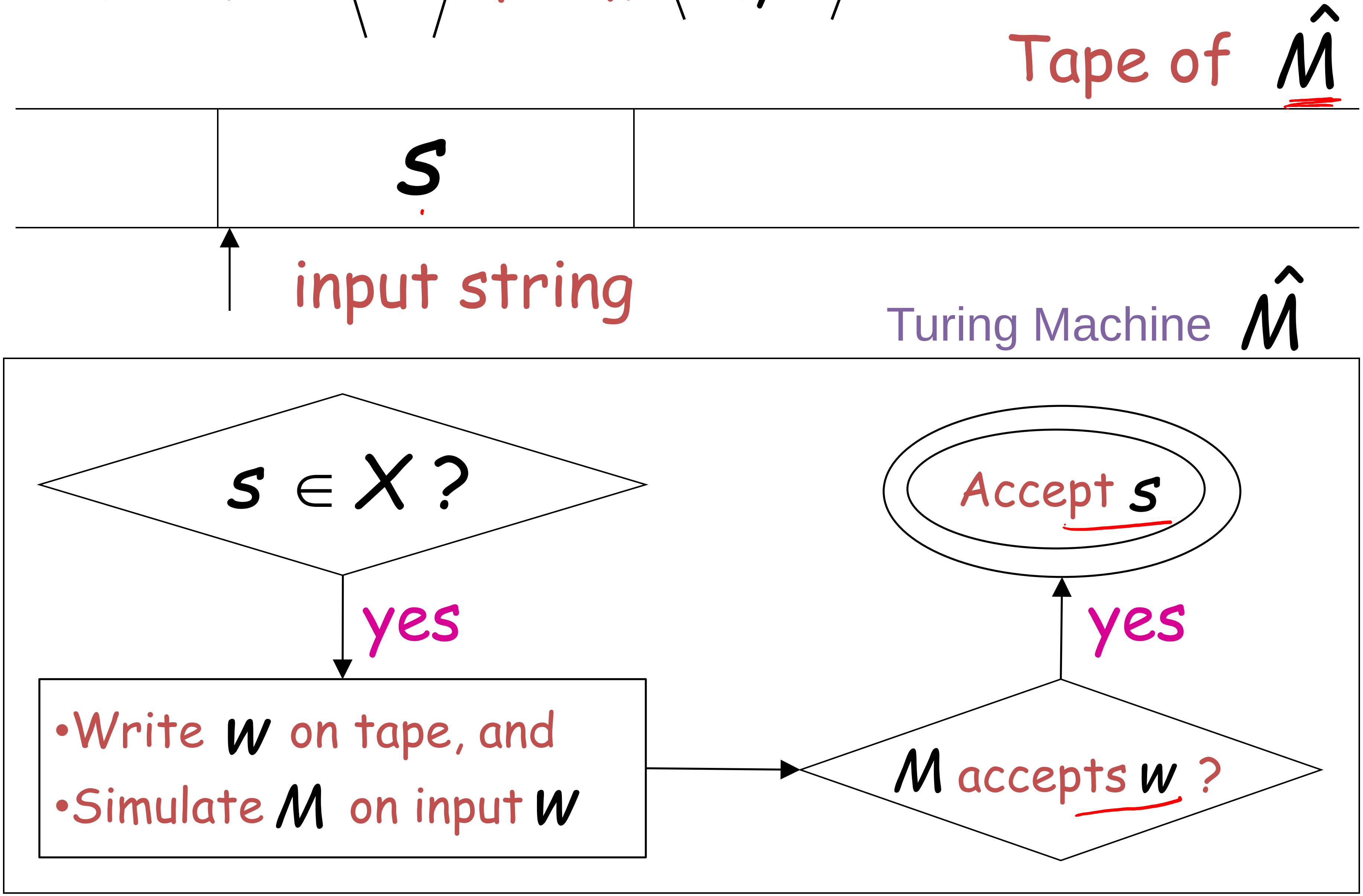
$$\langle M,w\rangle \longrightarrow \boxed{\begin{array}{c}\text{Compute}\\ f(\langle M,w\rangle)\end{array}} \longrightarrow \langle \hat{M}\rangle$$

$$= f(\langle M,w\rangle)$$

So that:

$$\langle M,w\rangle \in AT_{TM} \iff \langle \hat{M}\rangle \in PROPERTY_{TM}$$

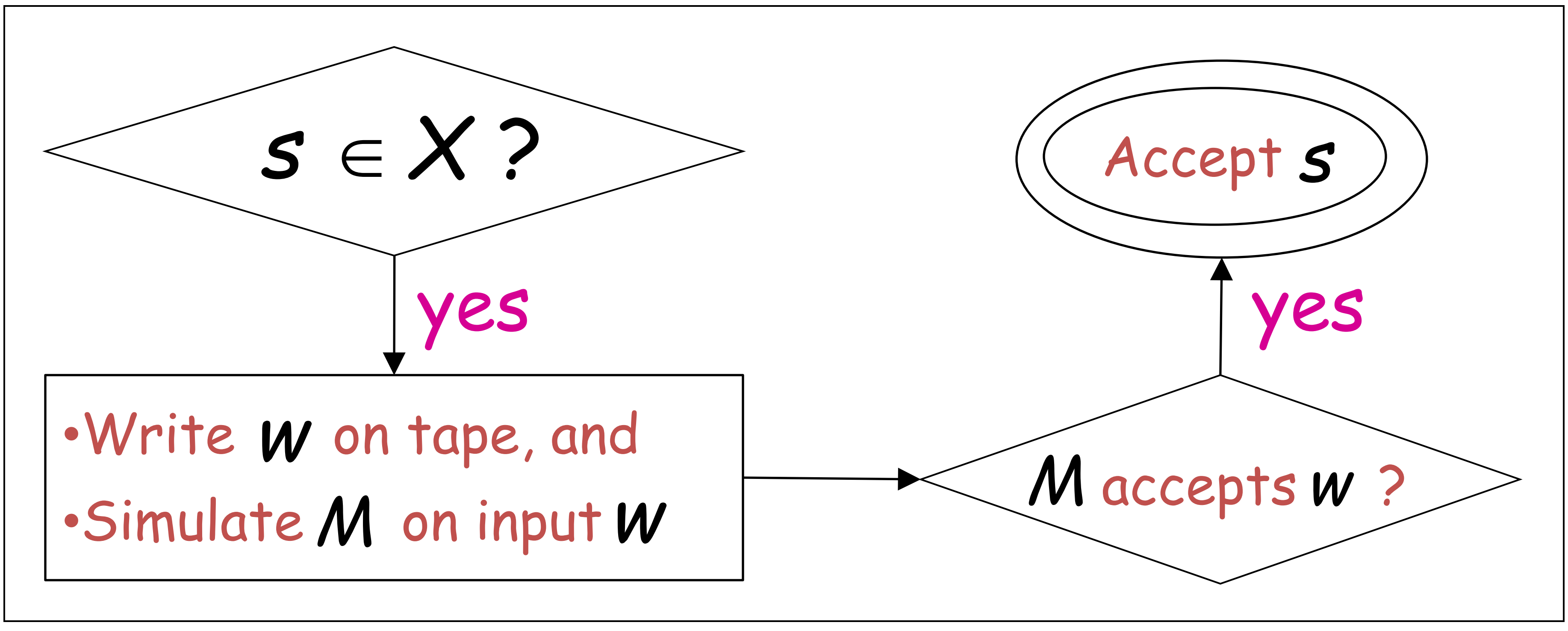
Construct $\langle \hat{M} \rangle$ from $\langle M, w \rangle$:



M accepts w $\Rightarrow$ $L(\hat{M}) = X \quad \in P$

M does not accept w $\Rightarrow$ $L(\hat{M}) = \emptyset \quad \notin P$

Turing Machine $\hat{M}$



Therefore:

M accepts w $\Longleftrightarrow$ $L(\hat{M}) \in P$

Equivalently:

$\langle M, w \rangle \in A_{TM}$ $\Longleftrightarrow$ $\langle \hat{M} \rangle \in PROPERTY_{TM}$

END OF PROOF

The Post Correspondence Problem

PCP

Emil Post

1946

Some undecidable problems for context-free languages:

- Is $L(G_1) \cap L(G_2) = \emptyset$?

  G_1, G_2 are context-free grammars

- Is context-free grammar G ambiguous?

We need a tool to prove that the previous problems for context-free languages are undecidable:

The Post Correspondence Problem

The Post Correspondence Problem

Input: Two sets of n strings

$$A = w_1, w_2, \ldots, w_n$$

$$B = v_1, v_2, \ldots, v_n$$

There is a Post Correspondence Solution if there is a sequence $i, j, \ldots, k$ such that:

PC-solution: $$w_i w_j \cdots w_k = v_i v_j \cdots v_k$$

Indices may be repeated or omitted

Example:

	w_1	w_2	w_3
$A:$	100	11	111

	v_1	v_2	v_3
$B:$	001	111	11

PC-solution: $2,1,3$

i, j, k

$$w_2 w_1 w_3 = v_2 v_1 v_3$$

$w_2 v_1$ $w_2 v_1$

$$\boxed{11100111}$$

Example:

	w_1	w_2	w_3
A:	00	001	1000
	v_1	v_2	v_3
B:	0	11	011

There is no solution

Because total length of strings from B
is smaller than total length of strings from A

The Modified Post Correspondence Problem

MPCP

Inputs: $A = w_1, w_2, \ldots, w_n$

$B = v_1, v_2, \ldots, v_n$

MPC-solution: $1, i, j, \ldots, k$

$$w_1 \underbrace{w_i w_j \cdots w_k}_{} = v_1 \underbrace{v_i v_j \cdots v_k}_{}$$

Example:

	w_1	w_2	w_3
$A:$	11	111	100
	v_1	v_2	v_3
$B:$	111	11	001

MPC-solution: 1,3,2 $\qquad w_1 w_3 w_2 = v_1 v_3 v_2$

11100111

We will show:

1. The MPC problem is undecidable

   (by reducing the membership to MPC)

2. The PC problem is undecidable

   (by reducing MPC to PC)

Theorem: The MPC problem is undecidable

Proof: We will reduce the membership problem to the MPC problem

Membership problem

Input: Turing machine M
string w

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

Undecidable

Membership problem

Input: unrestricted grammar G

string w

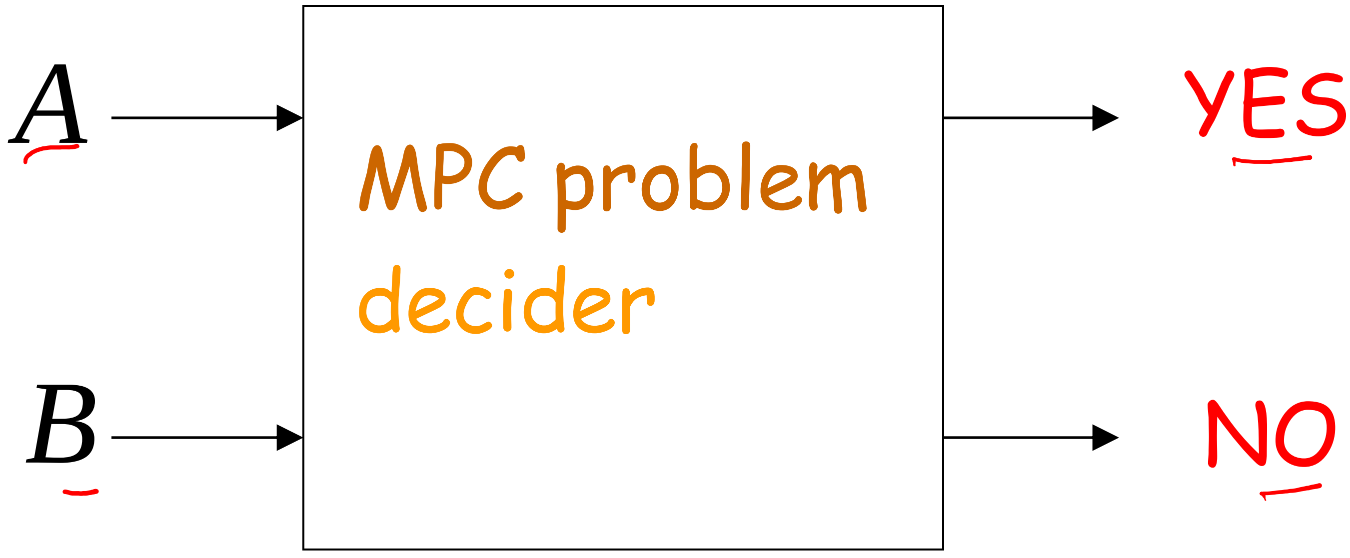
Question: $w \in L(G)?$

Undecidable

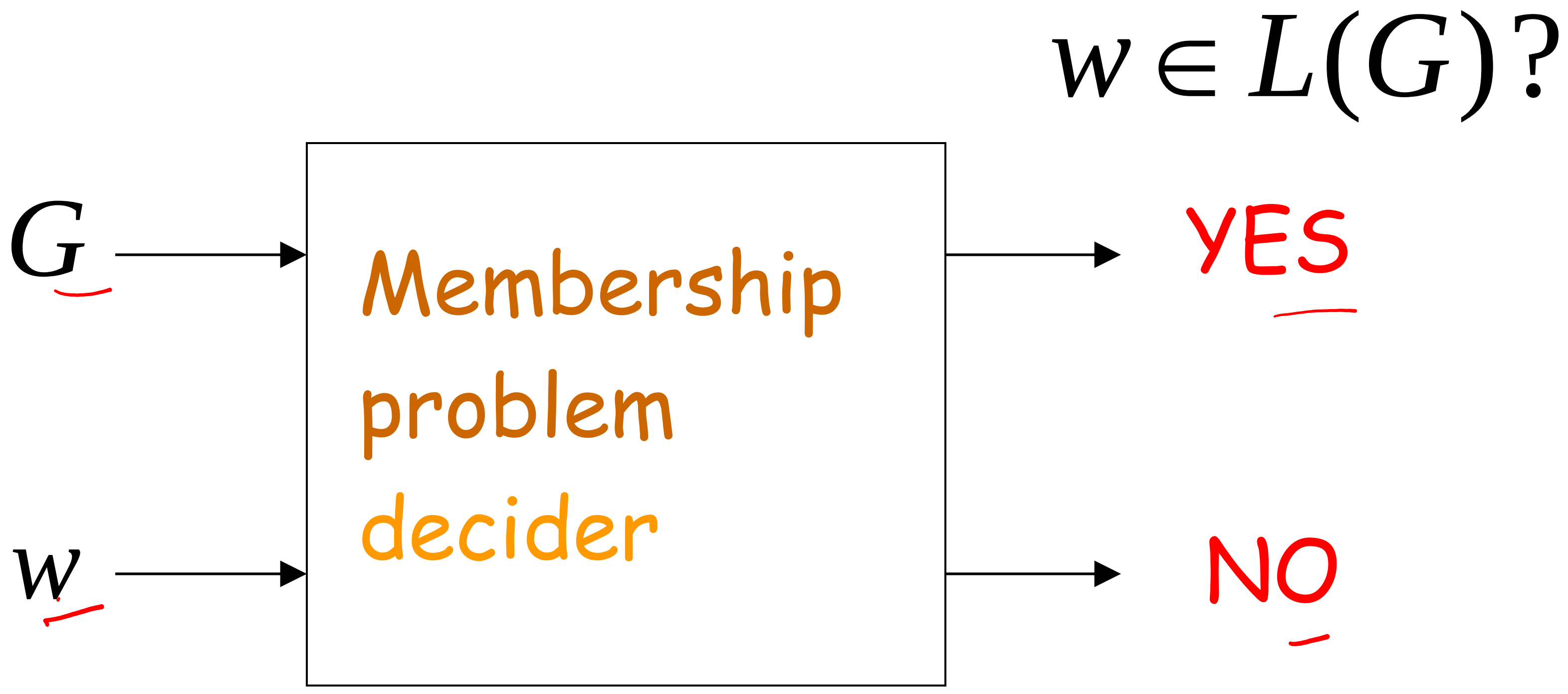
Suppose we have a decider for the MPC problem

String Sequences

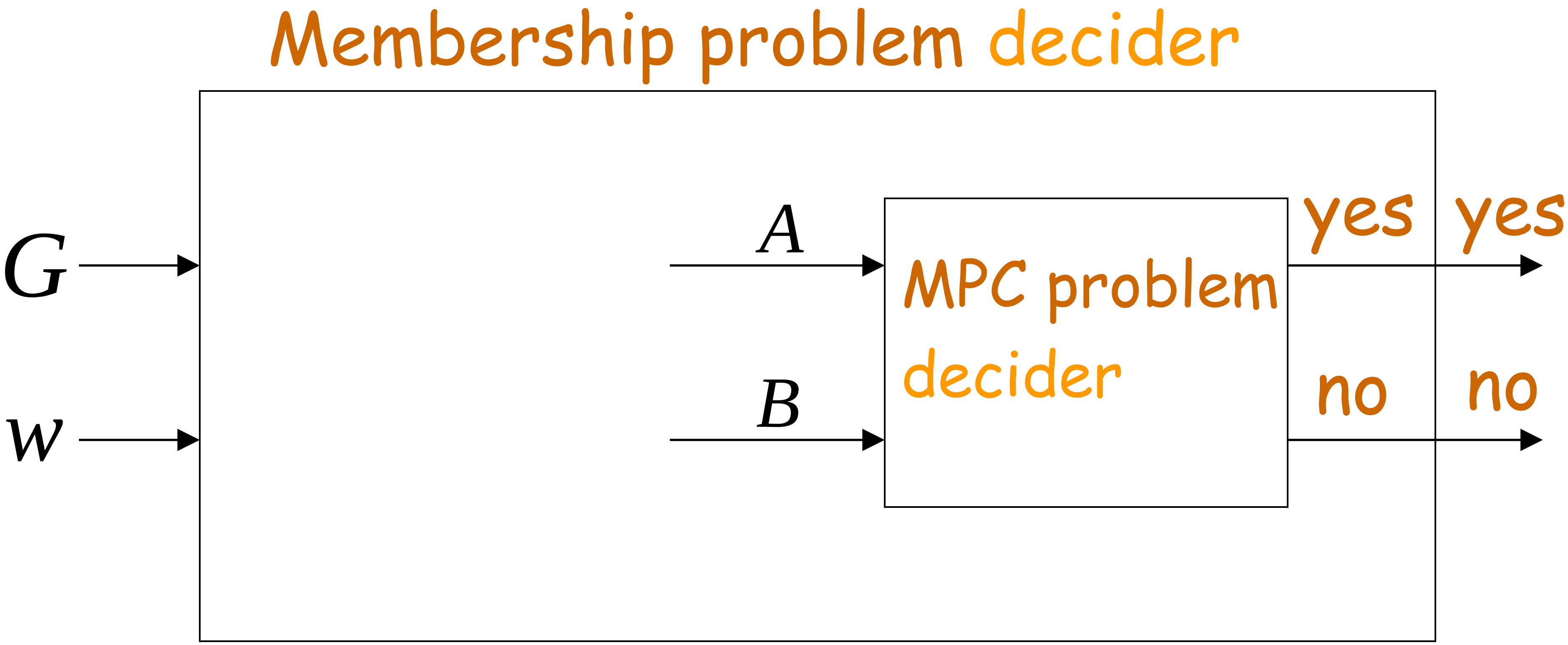
MPC solution?



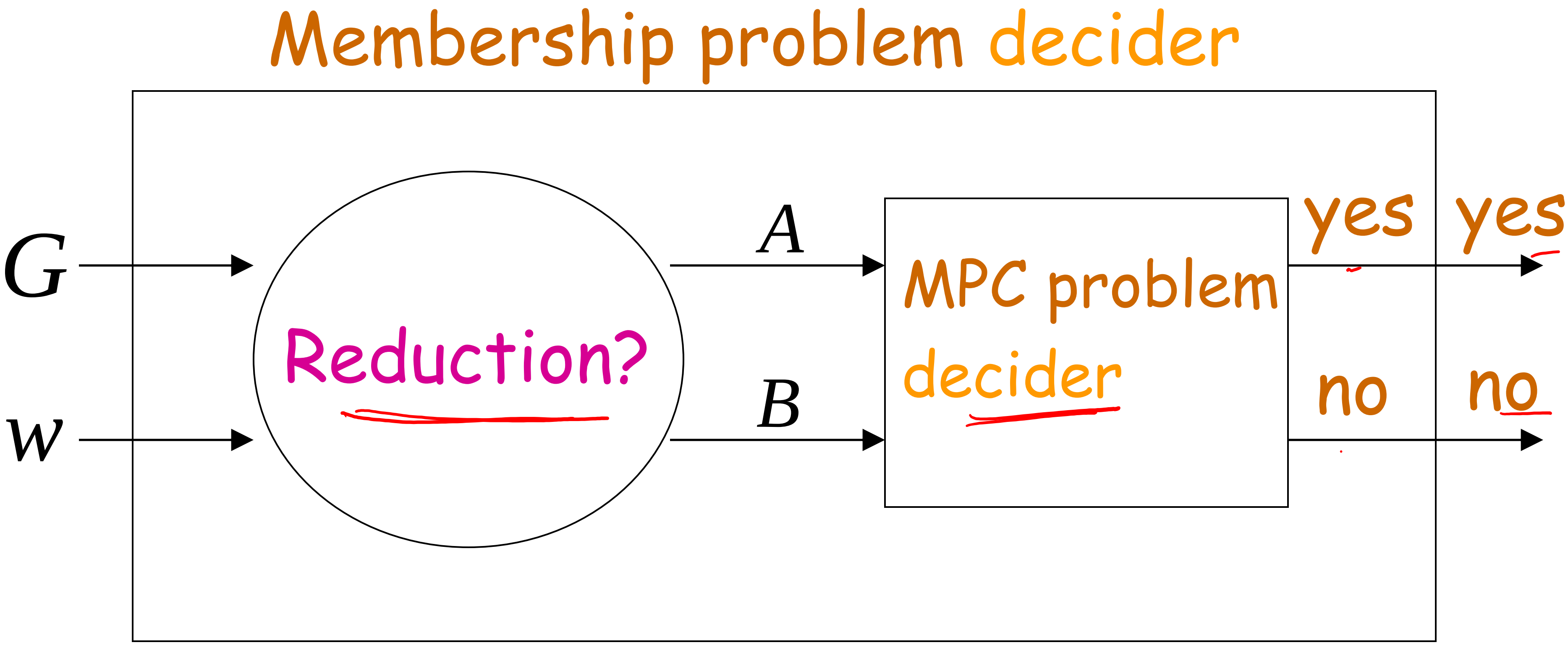
We will build a decider for the membership problem



The reduction of the membership problem to the MPC problem:



We need to convert the input instance of one problem to the other



Reduction:

Convert grammar G and string w

to sets of strings A and B

Such that:

G generates w $\Longleftrightarrow$ There is an MPC solution for A, B

A	B	Grammar G
$FS \Rightarrow$ 1	F 1	S : start variable F : special symbol
a	a	For every symbol a
V	V	For every variable V

A	B	Grammar G
E	$\Rightarrow wE$	string w E : special symbol
y	x	For every production $x \to y$
$\Rightarrow$	$\Rightarrow$	

Example:

Grammar G :

$$S \rightarrow aABb \mid Bbb$$

$$Bb \rightarrow C$$

$$AC \rightarrow aac$$

String $w = aaac$

	A			B	
w_1 :	$FS \Rightarrow$		v_1 :	F	
w_2 :	a		v_2 :	a	
w_3 :	b	a Σ	v_3 :	b	Σ
	c			c	
$\vdots$	A		$\vdots$	A	
	B	V		B	V
	C			C	
w_8 :	S		v_8 :	S	

	A		B
w_9 :	E	v_9 :	$\Rightarrow \overset{w}{aaac}E$
	$aABb$		S
	Bbb		S
$\vdots$		$\vdots$	
	C		Bb
	aac		AC
w_{14} :	$\Rightarrow$	v_{14} :	$\Rightarrow$

Grammar G :

$$S \to aABb \mid Bbb$$

$$Bb \to C$$

$$AC \to aac$$

$aaac \in L(G)$:

$$S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac$$

Derivation: S

$$S \to aABb \mid Bbb$$
$$Bb \to C$$
$$AC \to aac$$

$A :$ w_1

$$\overbrace{F \quad S \Rightarrow}^{w_1}$$
$$\underbrace{F}_{v_1}$$

$B :$ v_1

Derivation:

$$S \Rightarrow aABb$$

$w = aaac$

$$\boxed{\begin{aligned} S &\to aABb \mid Bbb \\ Bb &\to C \\ AC &\to aac \end{aligned}}$$

A : w_1 w_{10}

$$\overbrace{\underbrace{F}_{v_1}\ \ \underbrace{S}_{v_{10}}}^{w_1} \Rightarrow \overbrace{a\ \ A\ \ B\ \ b}^{w_{10}}$$

B : v_1 v_{10}

Derivation:

$$S \Rightarrow aABb \Rightarrow \underline{aAC}$$

$$S \rightarrow aABb \mid Bbb$$

$$Bb \rightarrow C$$

$$AC \rightarrow aac$$

$$A: \quad \overbrace{F\; S \Rightarrow}^{w_1} \overbrace{a\; A\; B\; b}^{w_{10}} \overbrace{\Rightarrow}^{w_{14}} \overbrace{a}^{w_2} \overbrace{A}^{w_5} \overbrace{C}^{w_{12}}$$

$$B: \quad \underbrace{F}_{v_1}\; \underbrace{S}_{v_{10}}\; \underbrace{\Rightarrow}_{v_{14}}\; \underbrace{a}_{v_2}\; \underbrace{A}_{v_5}\; \underbrace{B\; b}_{v_{12}} \Rightarrow a\; A\; C$$

Derivation:

$$S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac$$

$$S \rightarrow aABb \mid Bbb$$
$$Bb \rightarrow C$$
$$AC \rightarrow aac$$

$$A: \overbrace{F\ S\ \Rightarrow}^{w_1}\overbrace{a\ A\ B}^{w_{10}}\overbrace{b\Rightarrow}^{w_{14}}\overbrace{a}^{w_2}\overbrace{A}^{w_5}\overbrace{C}^{w_{12}}\overbrace{\Rightarrow}^{w_{14}}\overbrace{a}^{w_2}\overbrace{a\ a\ c\ E}^{w_{13}}$$

$$B: \underbrace{F}_{v_1}\underbrace{S}_{v_{10}}\underbrace{\Rightarrow}_{v_{14}}\underbrace{a}_{v_2}\underbrace{A}_{v_5}\underbrace{B\ b}_{v_{12}}\underbrace{\Rightarrow}_{v_{14}}\underbrace{a}_{v_2}\underbrace{A\ C}_{v_{13}}\ a\ a\ a\ c\ E$$

Derivation:

$$S \Rightarrow aABb \Rightarrow aAC \Rightarrow aaac$$

w

$$\boxed{\begin{array}{l} S \rightarrow aABb \mid Bbb \\ Bb \rightarrow C \\ AC \rightarrow aac \end{array}}$$

A: $w_1 \quad w_{10} \quad w_{14} \quad w_2 \quad w_5 \quad w_{12} \quad w_{14} \quad w_2 \quad w_{13} \quad w_9$

$$\overbrace{F\ S \Rightarrow}^{w_1} \overbrace{a\ A\ B\ b}^{w_{10}} \overbrace{\Rightarrow}^{w_{14}} \overbrace{a}^{w_2} \overbrace{A}^{w_5} \overbrace{C}^{w_{12}} \overbrace{\Rightarrow}^{w_{14}} \overbrace{a}^{w_2} \overbrace{a\ a\ c}^{w_{13}} \overbrace{E}^{w_9}$$

$$\underbrace{F}_{v_1} \underbrace{S}_{v_{10}} \underbrace{\Rightarrow}_{v_{14}} \underbrace{a}_{v_2} \underbrace{A}_{v_5} \underbrace{B\ b}_{v_{12}} \underbrace{\Rightarrow}_{v_{14}} \underbrace{a}_{v_2} \underbrace{A\ C}_{v_{13}} \underbrace{\Rightarrow a\ a\ a\ c\ E}_{v_9}$$

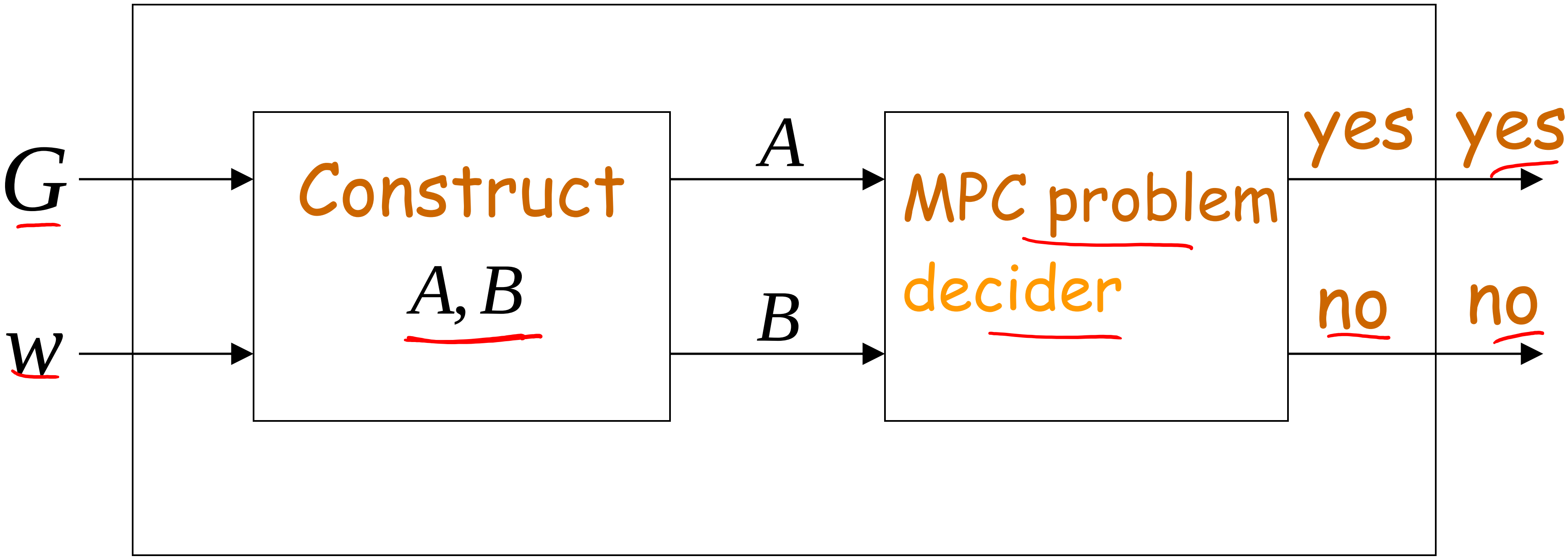
B: $v_1 \quad v_{10} \quad v_{14} \quad v_2 \quad v_5 \quad v_{12} \quad v_{14} \quad v_2 \quad v_{13} \quad v_9$

(A, B) has an MPC-solution

if and only if

$$w \in L(G)$$

Membership problem decider



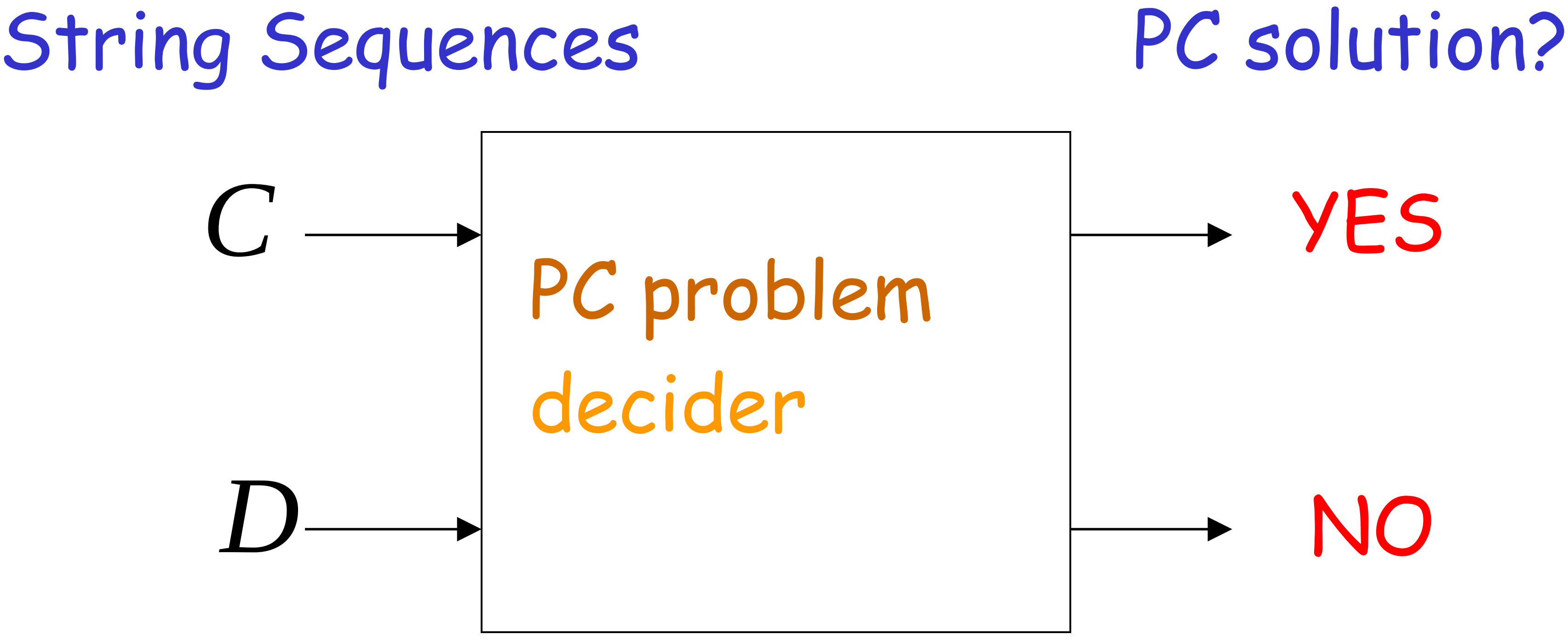
Since the membership problem is undecidable,
The MPC problem is undecidable

END OF PROOF

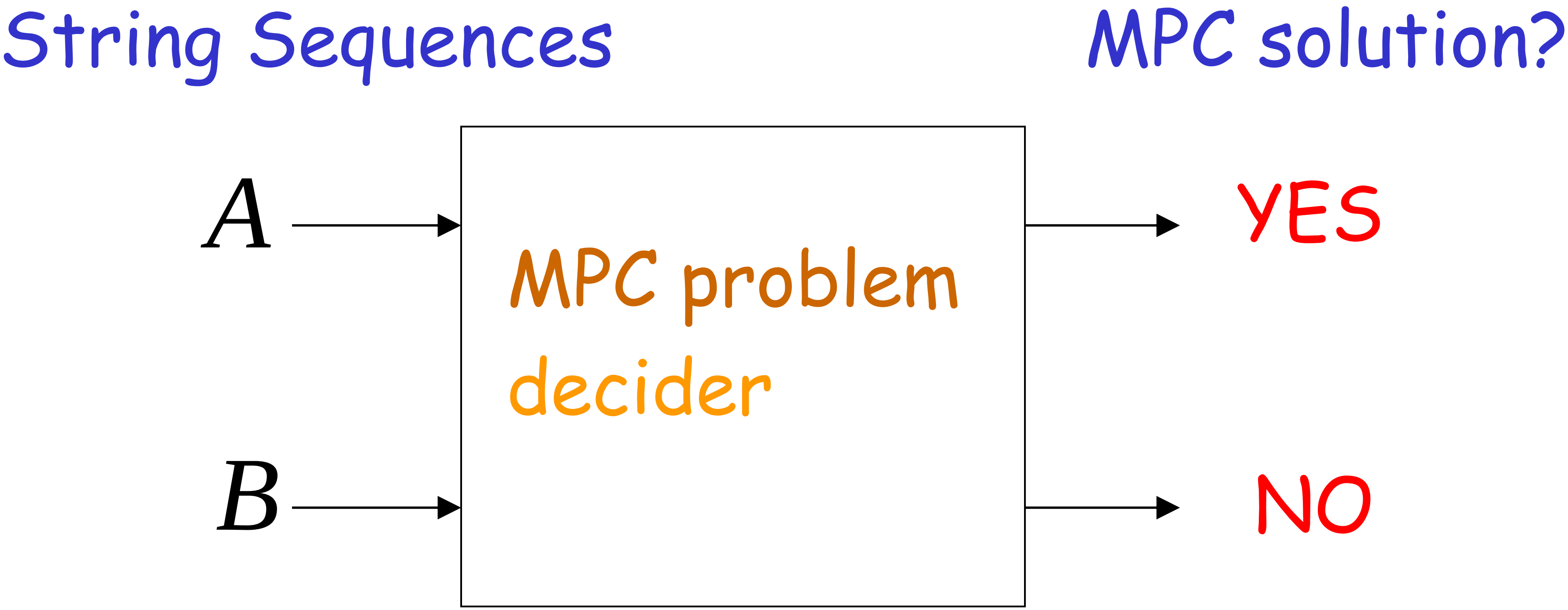
Theorem: The PC problem is undecidable

Proof: We will reduce the MPC problem to the PC problem

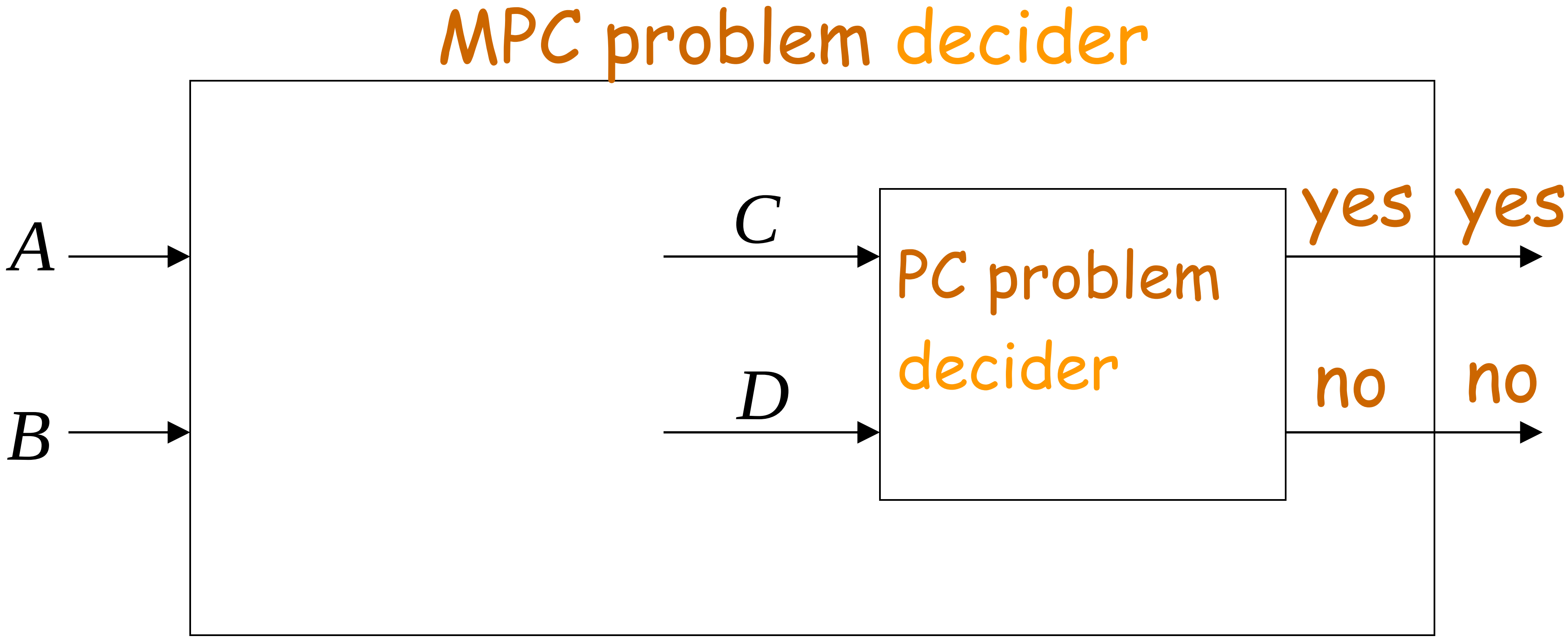
Suppose we have a decider for the PC problem



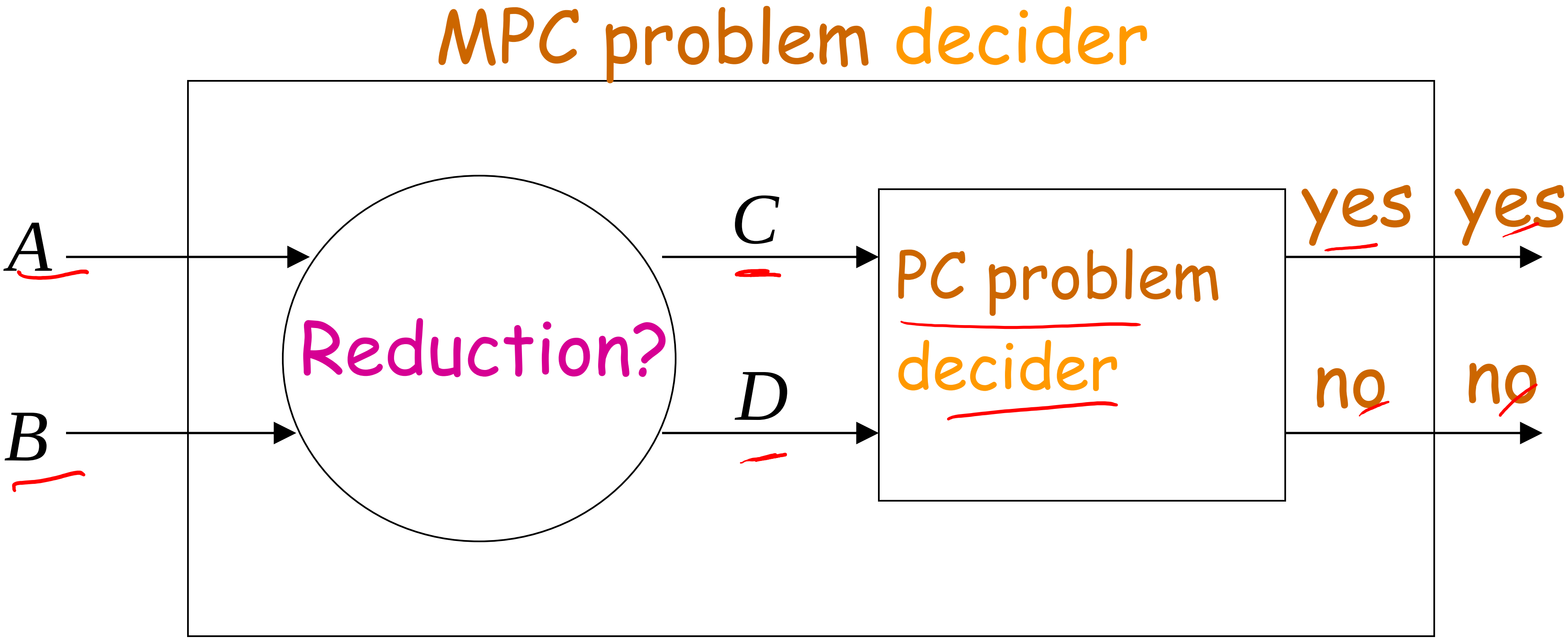
We will build a decider for the MPC problem



The reduction of the MPC problem to the PC problem:



We need to convert the input instance of one problem to the other



A, B : input to the MPC problem

$$A = w_1, w_2, \ldots, w_n$$

$$B = v_1, v_2, \ldots, v_n$$

Translated to

C, D : input to the PC problem

$$C = w'_1, \ldots, w'_n, w'_{n+1}$$

$$D = v'_1, \ldots, v'_n, v'_{n+1}$$

A $\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ C

$w_i = \sigma_1 \sigma_2 \cdots \sigma_k$ ⟹ $w_i' = \sigma_1 * \sigma_2 * \cdots \sigma_k *$

For each i

replace $w_1' = *w_1'$

$w_{n+1}' = \Diamond$

B $\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad$ D

$v_i = \pi_1 \pi_2 \cdots \pi_k$ ⟹ $v_i' = *\pi_1 * \pi_2 * \cdots * \pi_k$

For each i

$v_{n+1}' = *\Diamond$

PC-solution

$$\overset{C}{w_1' w_i' \cdots w_k' w_{n+1}'} = \overset{D}{v_1' v_i' \cdots w_k' v_{n+1}'}$$

Has to start with
These strings

C PC-solution D

$$w'_1 w'_i \cdots w'_k w'_{n+1} = v'_1 v'_i \cdots w'_k v'_{n+1}$$

A B

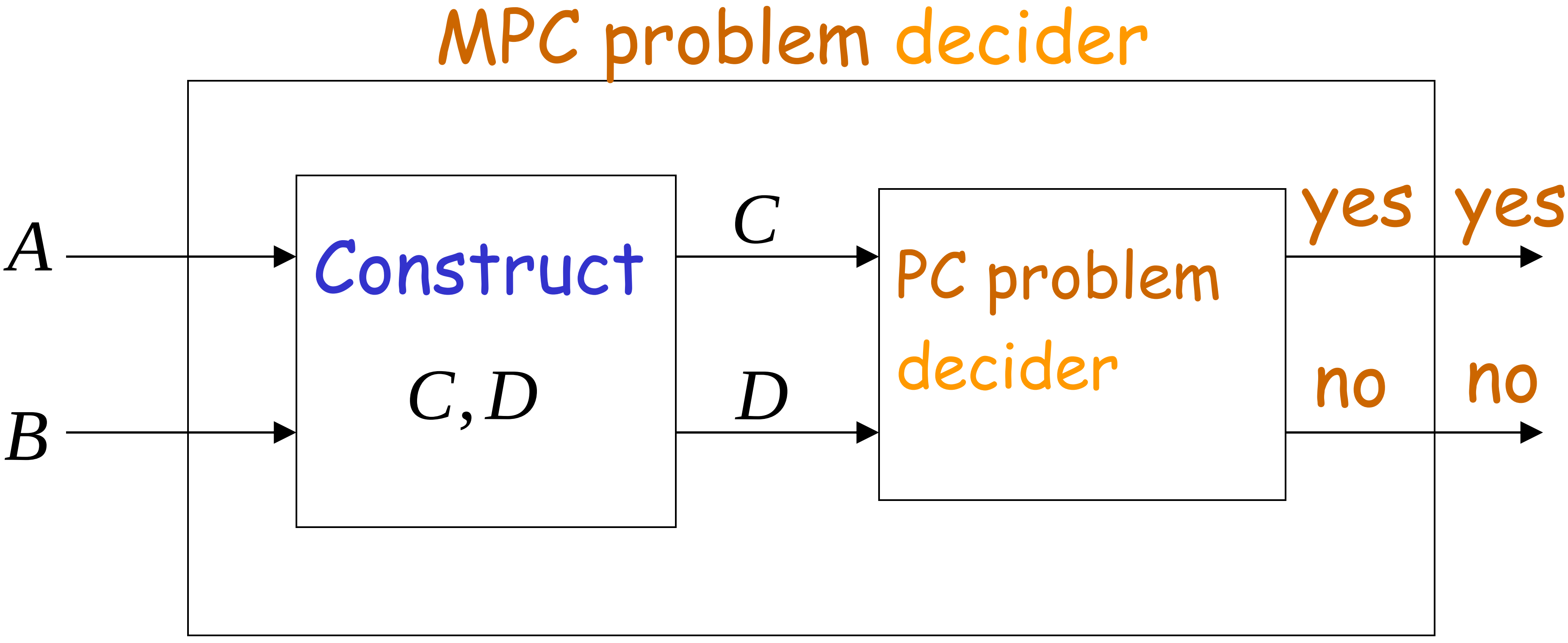
$$w_1 w_i \cdots w_k = v_1 v_i \cdots v_k$$

MPC-solution

C, D has a PC solution

if and only if

A, B has an MPC solution


MPC problem decider
A
B
Construct
C, D
C
D
PC problem
decider
yes
no
yes
no

Since the MPC problem is undecidable,
The PC problem is undecidable

END OF PROOF

Some undecidable problems for context-free languages:

- Is $L(G_1) \cap L(G_2) = \emptyset$?

  G_1, G_2 are context-free grammars

- Is context-free grammar G ambiguous?

We reduce the PC problem to these problems

Theorem: Let G_1, G_2 be context-free grammars. It is undecidable to determine if

$$L(G_1) \cap L(G_2) = \emptyset$$

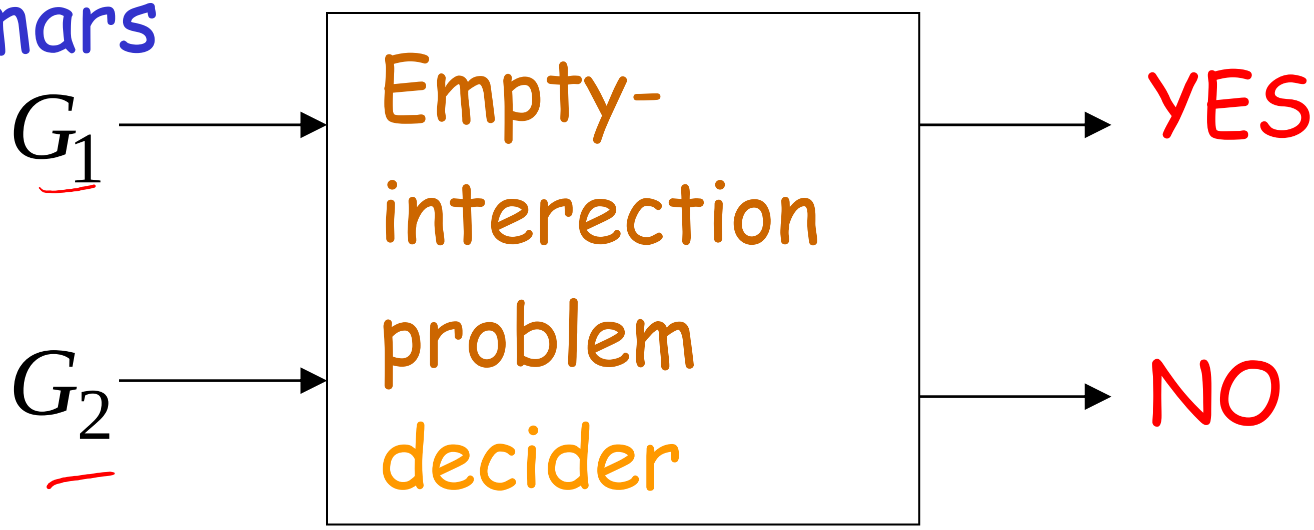
(intersection problem)

Proof: Reduce the PC problem to this problem

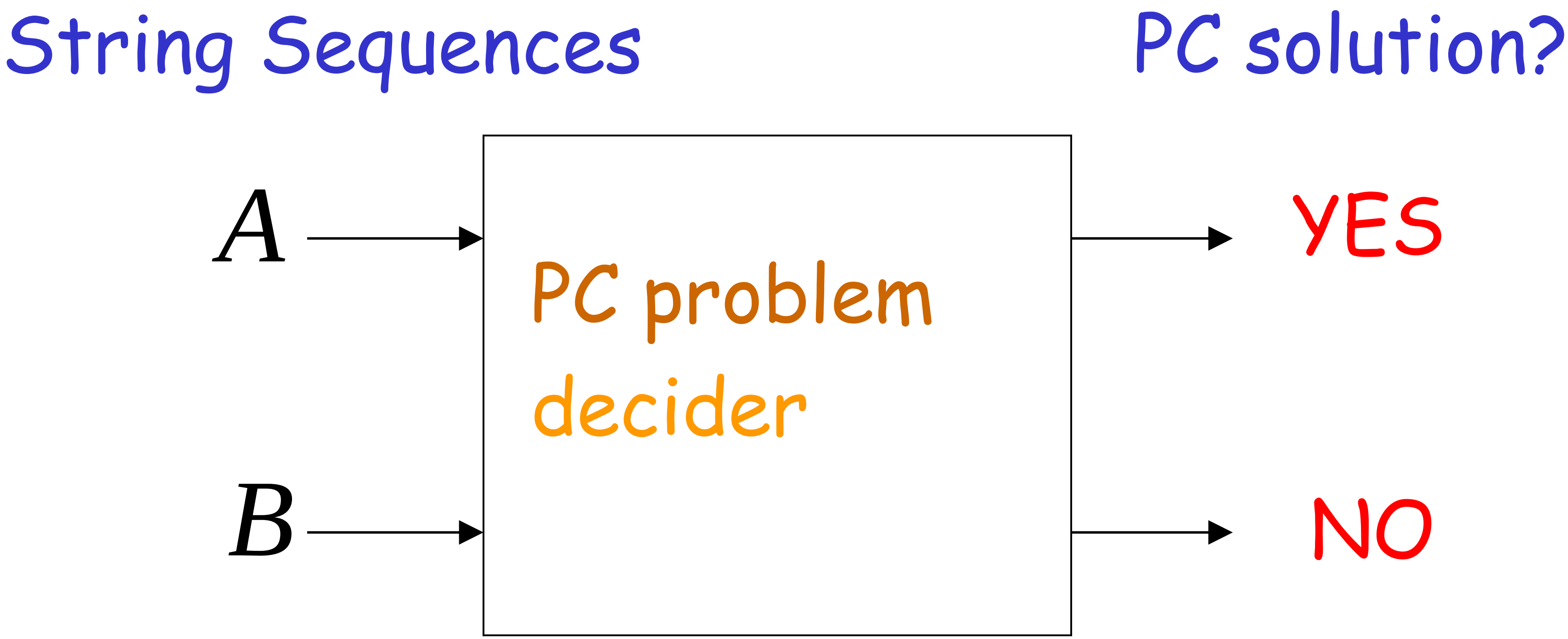
Suppose we have a decider for the intersection problem

Context-free grammars

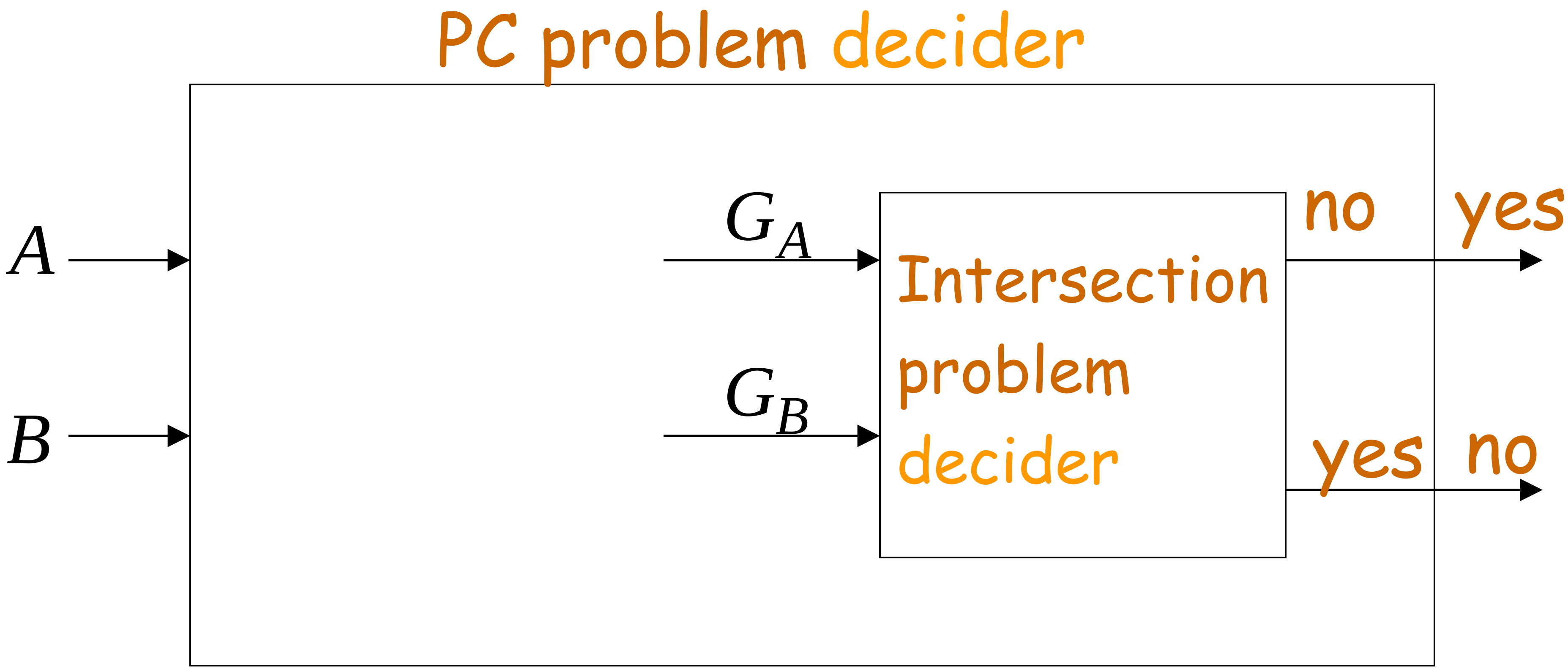
$$L(G_1) \cap L(G_2) = \emptyset ?$$



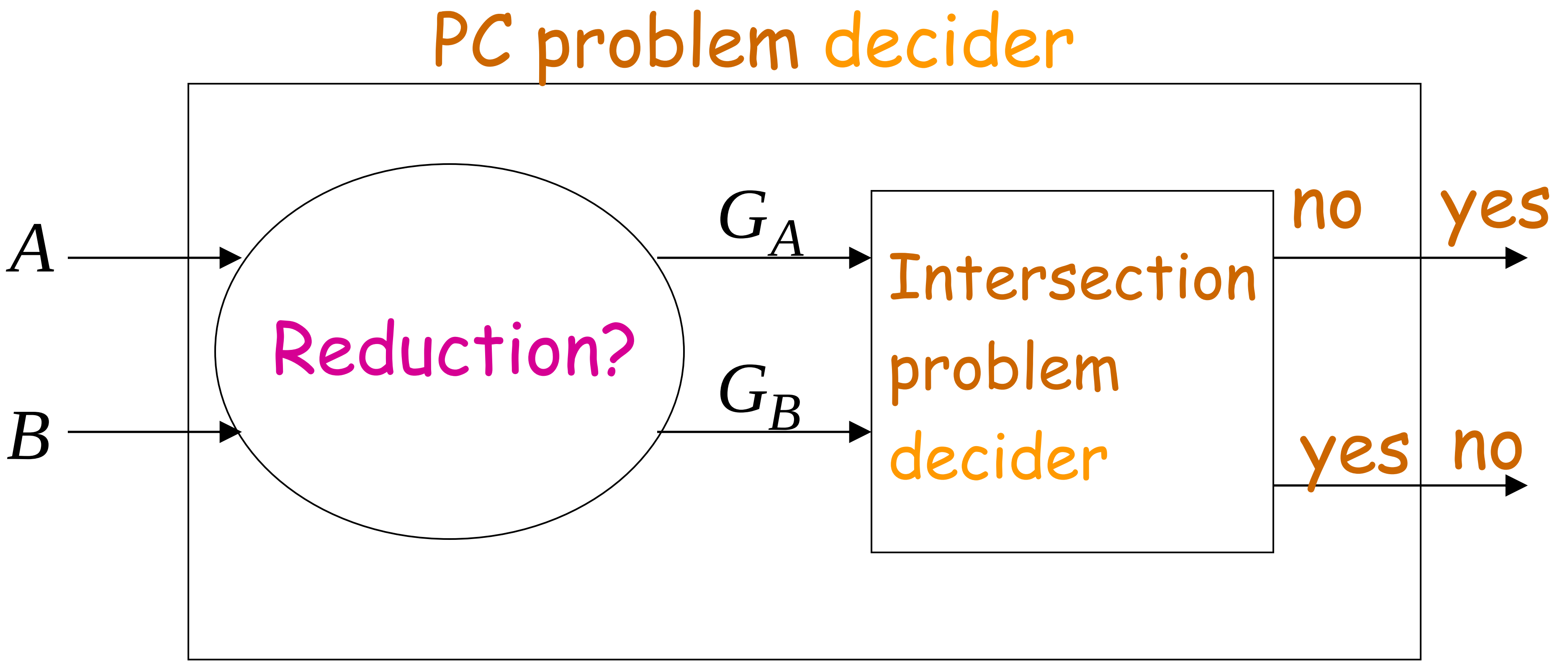
We will build a decider for the PC problem



The reduction of the PC problem to the empty-intersection problem:



We need to convert the input instance of one problem to the other



Introduce new unique symbols: $a_1, a_2, \ldots, a_n$

$$A = w_1, w_2, \ldots, w_n$$

$$L_A = \{s: \quad s = w_i w_j \cdots w_k a_k \cdots a_j a_i\}$$

Context-free grammar G_A: $S_A \rightarrow w_i S_A a_i \mid w_i a_i$

$$B = v_1, v_2, \ldots, v_n$$

$$L_B = \{s: \quad s = v_i v_j \cdots v_k a_k \cdots a_j a_i\}$$

Context-free grammar G_B: $S_B \rightarrow v_i S_B a_i \mid v_i a_i$

(A, B) has a PC solution

if and
only if

$$L(G_A) \cap L(G_B) \neq \emptyset$$

$$L(G_1) \cap L(G_2) \neq \emptyset$$

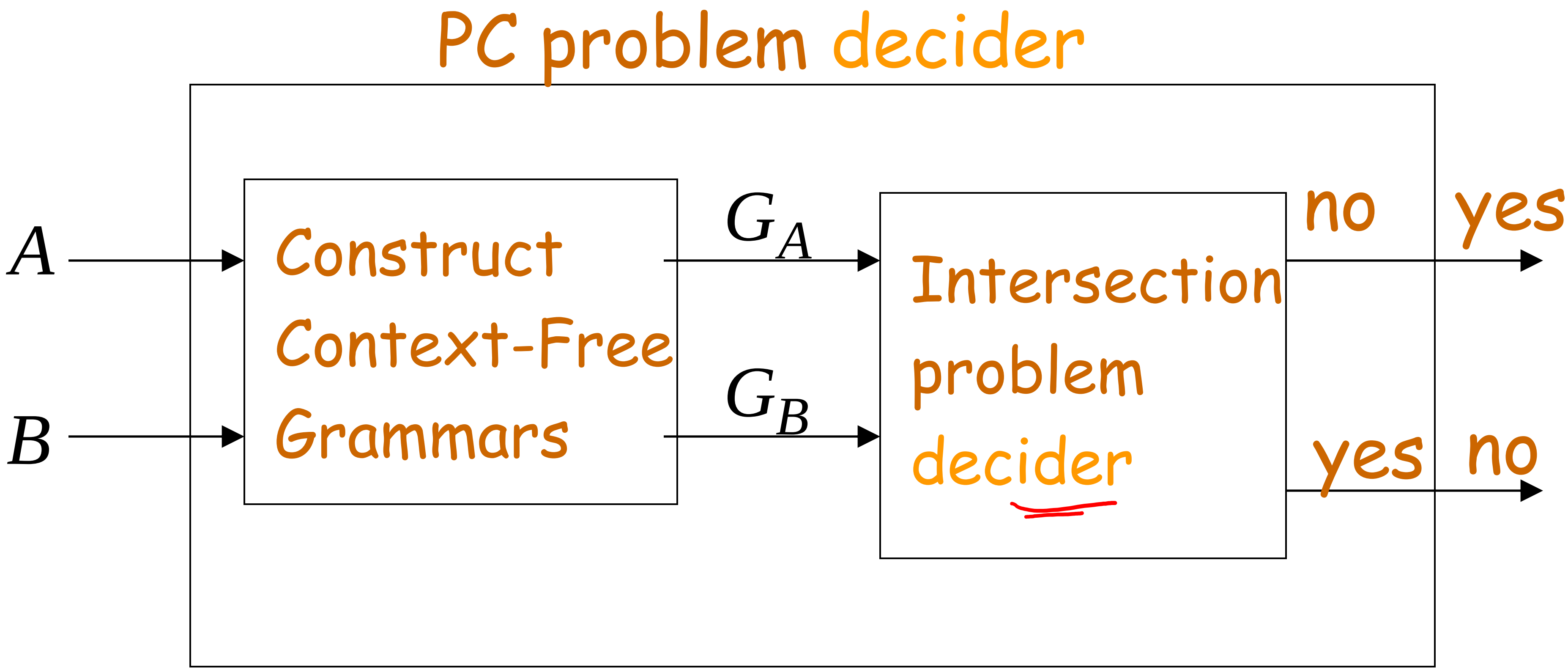
$$s = w_i w_j \cdots w_k a_k \cdots a_j a_i$$

$$s = v_i v_j \cdots v_k a_k \cdots a_j a_i$$

Because $a_1, a_2, \ldots, a_n$ are unique

There is a PC solution:

$$w_i w_j \cdots w_k = v_i v_j \cdots v_k$$



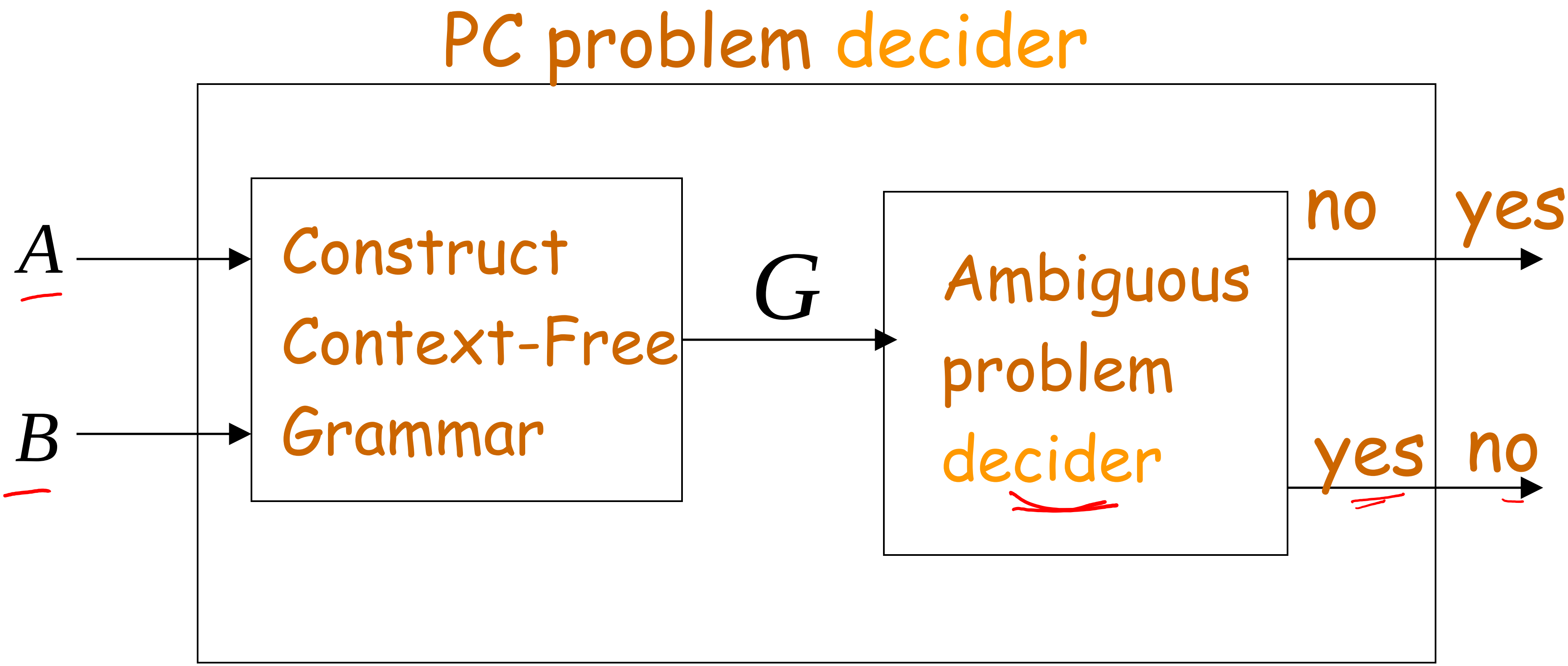
PC problem decider
A
B
Construct
Context-Free
Grammars
G_A
G_B
Intersection
problem
decider
no
yes
yes
no

Since PC is undecidable,
the Intersection problem is undecidable

END OF PROOF

Theorem: For a context-free grammar G,
it is undecidable to determine
if G is ambiguous

Proof: Reduce the PC problem
to this problem



PC problem decider
A
B
Construct
Context-Free
Grammar
G
Ambiguous
problem
decider
no
yes
yes
no

S_A start variable of G_A

S_B start variable of G_B

S start variable of G

$$S \rightarrow S_A \mid S_B$$

(A, B) has a PC solution

if and only if

$$L(G_A) \cap L(G_B) \neq \emptyset$$

if and only if

G is ambiguous

w

$S \to S_A \mid S_B$