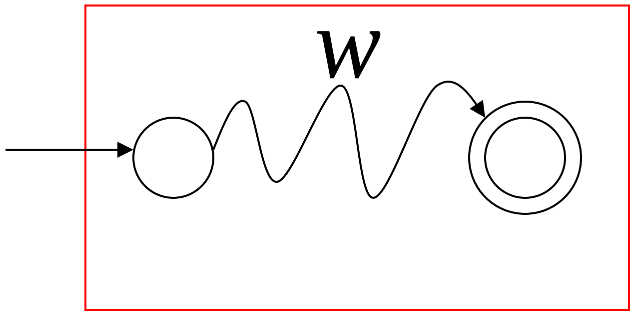
Decidable Problems
of
Regular Languages

Membership Question

Question: Given regular language L
and string w
how can we check if $w \in L$?

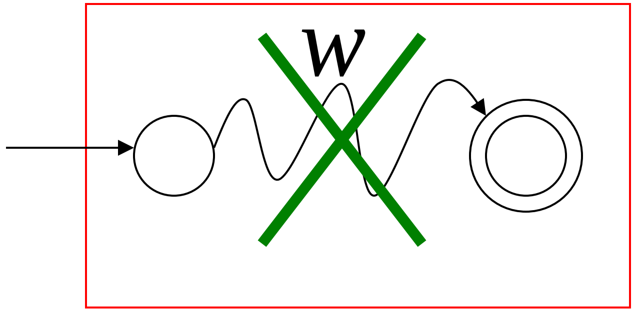
Answer: Take the DFA that accepts L
and check if w is accepted

DFA



$w \in L$

DFA



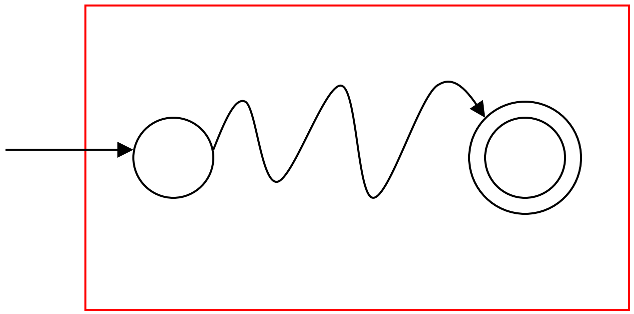
$w \notin L$

Question: Given regular language L
how can we check
if L is empty: $(L = \emptyset)$?

Answer: Take the DFA that accepts L

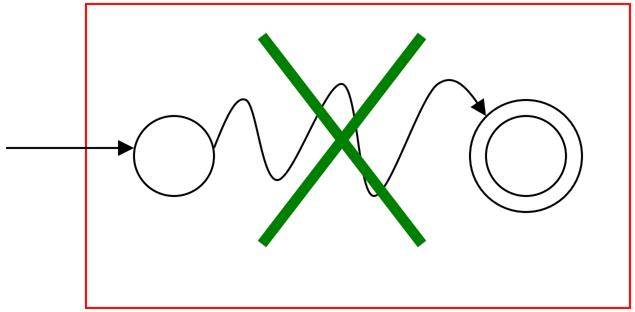
Check if there is any path from
the initial state to an accepting state

DFA



$$L \neq \emptyset$$

DFA

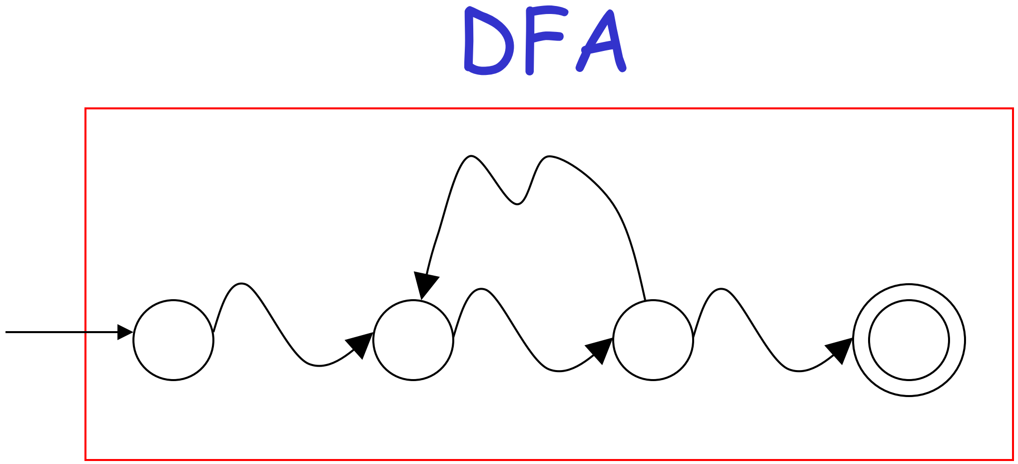


$$L = \emptyset$$

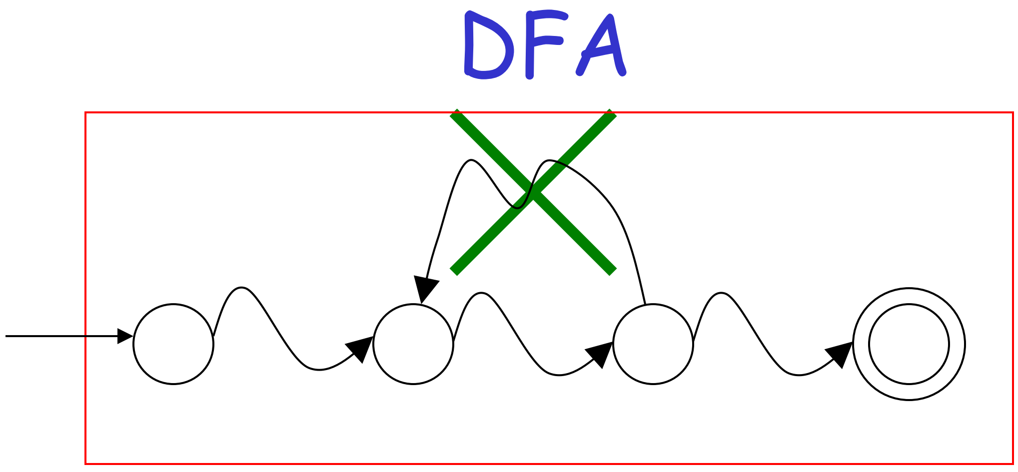
Question: Given regular language L
how can we check
if L is finite?

Answer: Take the DFA that accepts L

Check if there is a walk with cycle
from the initial state to a final state



L is infinite



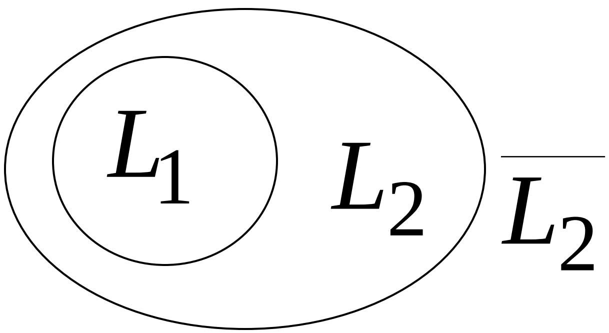
L is finite

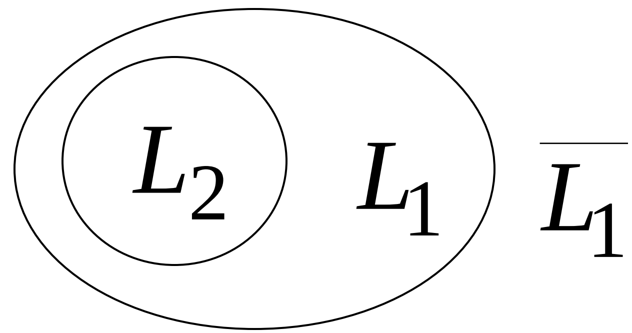
Question: Given regular languages L_1 and L_2 how can we check if $L_1 = L_2$?

Answer: Find if $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) = \emptyset$

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

$L_1 \cap \overline{L_2} = \varnothing$ and $\overline{L_1} \cap L_2 = \varnothing$





$L_1 \subseteq L_2$ $L_2 \subseteq L_1$

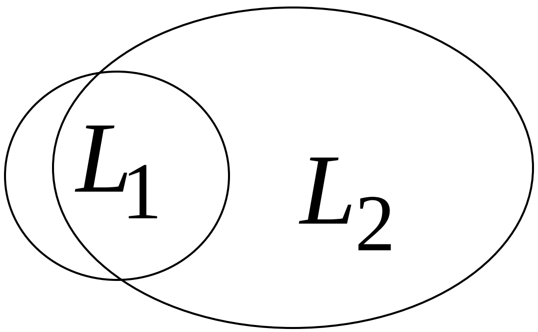
$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

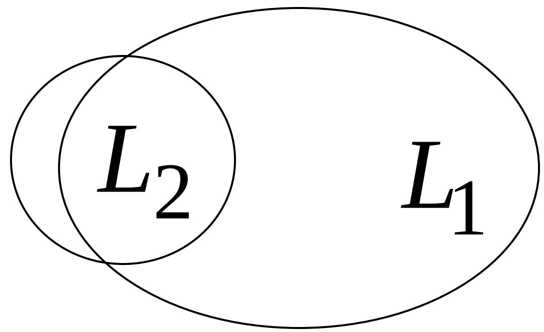
$$L_1 \cap \overline{L_2} \neq \emptyset$$

or

$$\overline{L_1} \cap L_2 \neq \emptyset$$



$$L_1 \not\subset L_2$$



$$L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

Decidable Problems
of
Context-Free Languages

Membership Question:

for context-free grammar G

find if string $w \in L(G)$

Membership Algorithms: Parsers

- Exhaustive search parser
- **CYK** parsing algorithm

Empty Language Question:

for context-free grammar G

find if $L(G) = \varnothing$

Algorithm:

1. Remove useless variables

2. Check if start variable S is useless

Infinite Language Question:

for context-free grammar G

find if $L(G)$ is infinite

Algorithm:

1. Remove useless variables

2. Remove unit and λ productions

3. Create dependency graph for variables

4. If there is a loop in the dependency graph then the language is infinite

Example: $S \to AB$

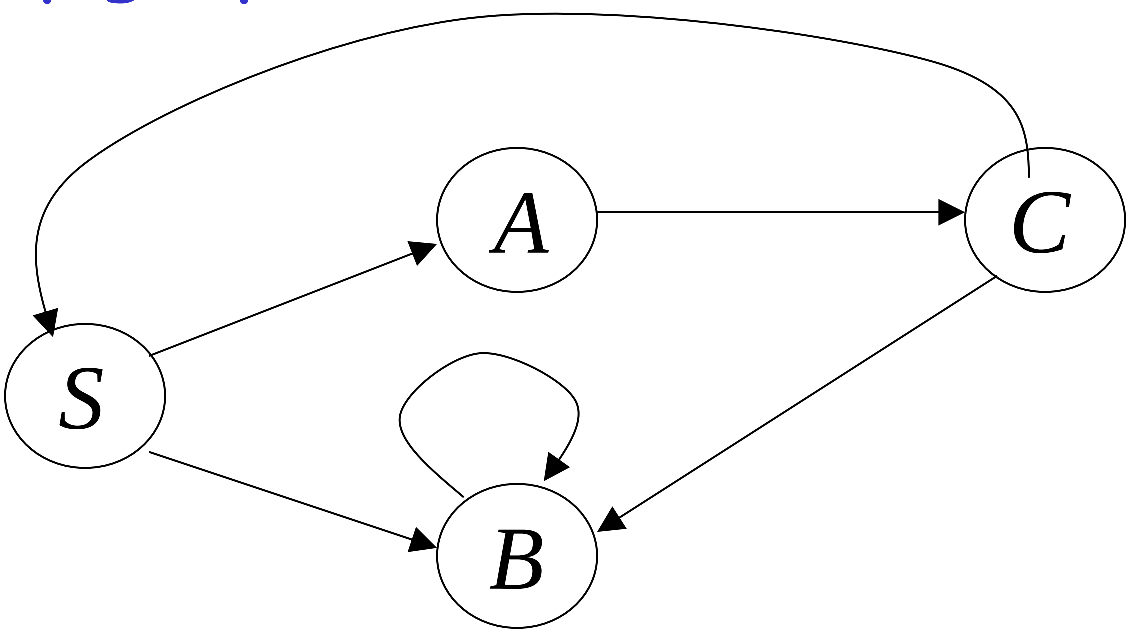
$$A \to aCb \mid a$$

$$B \to bB \mid bb$$

$$C \to cBS$$

Dependency graph

Infinite language



$$S \to AB$$

$$A \to aCb \mid a$$

$$B \to bB \mid bb$$

$$C \to cBS$$

$$S \Rightarrow AB \Rightarrow aCbB \Rightarrow acBSbB \Rightarrow acbbSbbb$$

$$S \overset{*}{\Rightarrow} acbbSbbb \overset{*}{\Rightarrow} (acbb)^2 S (bbb)^2$$

$$\overset{*}{\Rightarrow} (acbb)^i S (bbb)^i$$

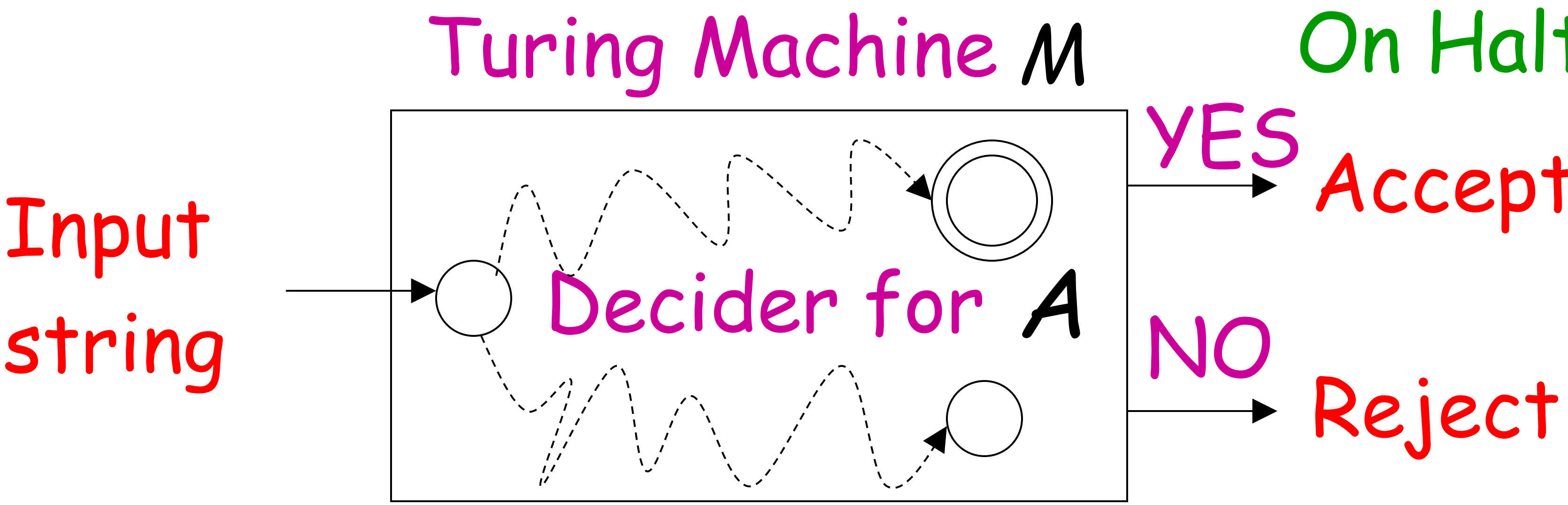
Decidable Languages

Decidable Languages

Recall that:

A language A is decidable,
if there is a Turing machine M (decider)
that accepts the language A and
halts on every input string

Decision On Halt:



A computational problem is decidable
if the corresponding language is decidable

We also say that the problem is solvable

Problem: Does DFA M accept the empty language $L(M) = \emptyset$?

Corresponding Language: (Decidable)

$EMPTY_{DFA} =$

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

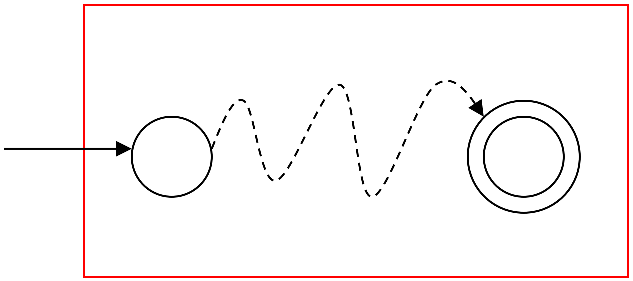
Description of DFA M as a string
(For example, we can represent M as a binary string, as we did for Turing machines)

Decider for $EMPTY_{DFA}$:

On input $\langle M \rangle$:

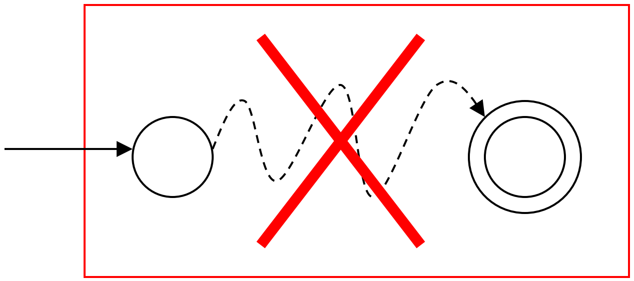
Determine whether there is a path from the initial state to any accepting state

DFA M



$L(M) \neq \emptyset$

DFA M



$L(M) = \emptyset$

Decision: Reject $\langle M \rangle$ Accept $\langle M \rangle$

Problem: Does DFA M accept a finite language?

Corresponding Language: (Decidable)

$$FINITE_{DFA} =$$

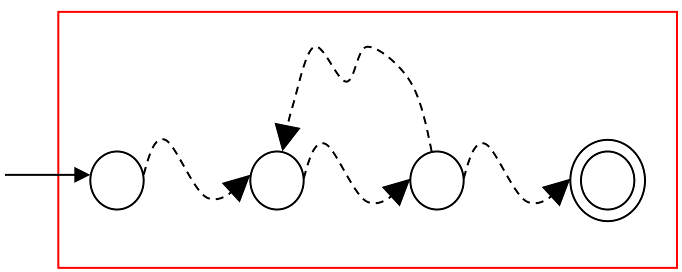
$$\{\langle M \rangle : M \text{ is a DFA that accepts a finite language}\}$$

Decider for $FINITE_{DFA}$:

On input $\langle M \rangle$:

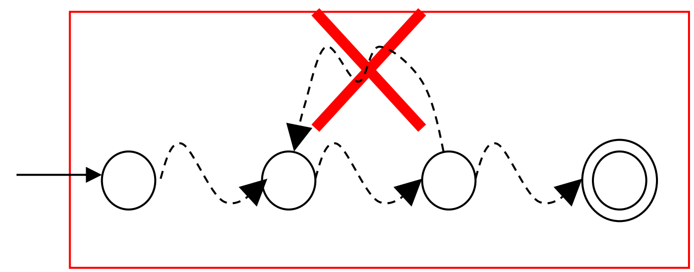
Check if there is a walk with a cycle from the initial state to an accepting state

DFA M



infinite

DFA M



finite

Decision:

Reject $\langle M \rangle$ (NO)

Accept $\langle M \rangle$ (YES)

Problem: Does DFA M accept string w ?

Corresponding Language: (Decidable)

$$A_{DFA} = \{\langle M, w \rangle : M \text{ is a DFA that accepts string } w\}$$

Decider for A_{DFA} :

On input string $\langle M, w \rangle$:

Run DFA M on input string w

If M accepts w

<u>Then</u> accept $\langle M, w \rangle$ (and halt)

<u>Else</u> reject $\langle M, w \rangle$ (and halt)

Problem: Do DFAs M_1 and M_2 accept the same language?

Corresponding Language: (Decidable)

$EQUAL_{DFA} =$

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

Decider for $EQUAL_{DFA}$:

On input $\langle M_1, M_2 \rangle$:

Let L_1 be the language of DFA M_1

Let L_2 be the language of DFA M_2

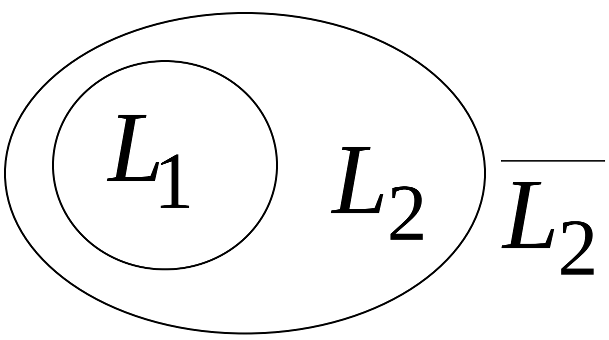
Construct DFA M such that:

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

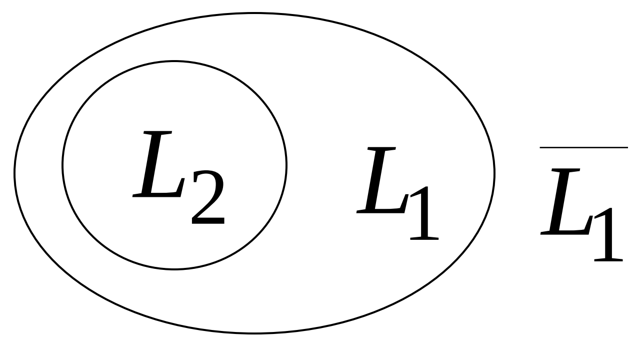
(combination of DFAs)

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

$$L_1 \cap \overline{L_2} = \varnothing \quad \text{and} \quad \overline{L_1} \cap L_2 = \varnothing$$



$$L_1 \subseteq L_2$$



$$L_2 \subseteq L_1$$

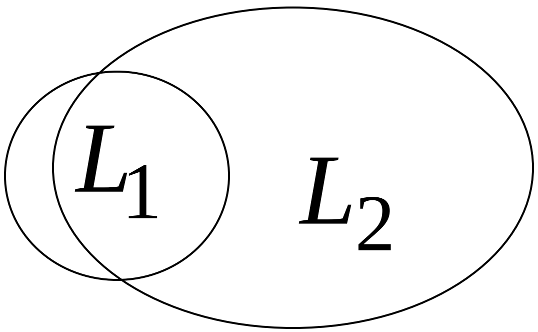
$$L_1 = L_2$$

$$(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2) \neq \emptyset$$

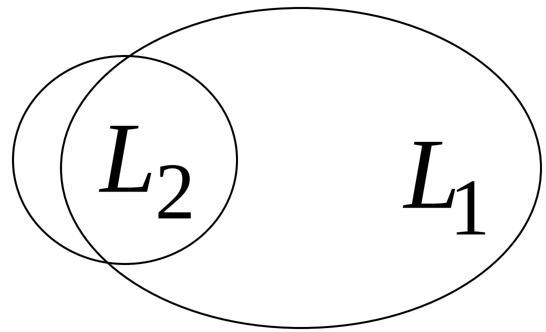
$$L_1 \cap \overline{L_2} \neq \emptyset$$

or

$$\overline{L_1} \cap L_2 \neq \emptyset$$



$$L_1 \not\subset L_2$$



$$L_2 \not\subset L_1$$

$$L_1 \neq L_2$$

Therefore, we only need to determine whether

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

which is a solvable problem for DFAs:

$EMPTY_{DFA}$

Undecidable Problems
(unsolvable problems)

Undecidable Languages

undecidable language = not decidable language

There is no decider:

there is no Turing Machine
which accepts the language
and makes a decision (halts)
for every input string

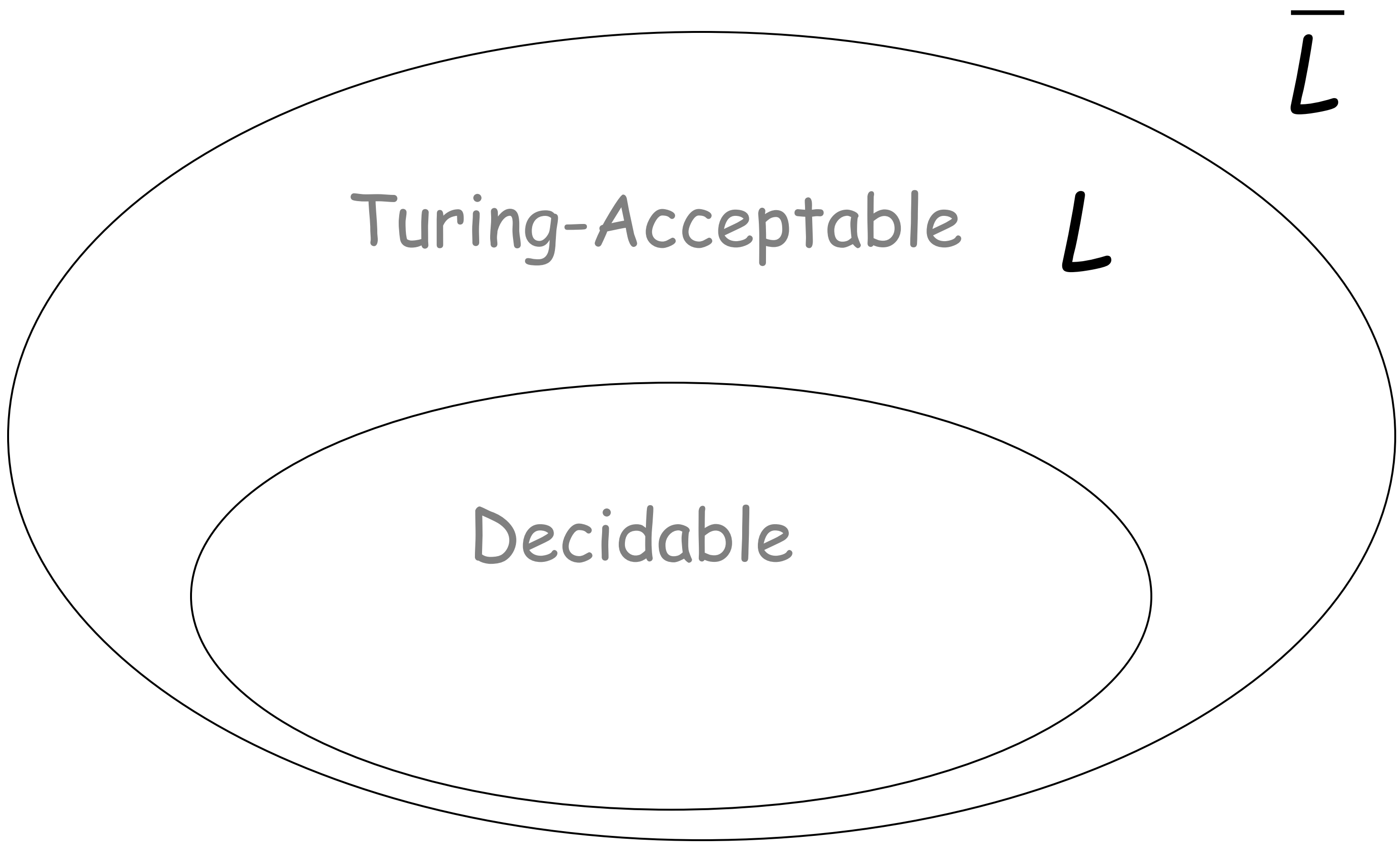
(machine may make decision for some input strings)

For an undecidable language,
the corresponding problem is
undecidable (unsolvable):

there is no Turing Machine (Algorithm)
that gives an answer (yes or no)
for every input instance

(answer may be given for some input instances)

We have shown before that there are undecidable languages:



L is Turing-Acceptable and undecidable

We will prove that two particular problems are unsolvable:

Membership problem

Halting problem

Membership Problem

Input: •Turing Machine M
•String w

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

Corresponding language:

$$A_{TM} = \{\langle M, w \rangle : M \text{ is a Turing machine that accepts string } w\}$$

Theorem: A_{TM} is undecidable

(The membership problem is unsolvable)

Proof:

Basic idea:

We will assume that A_{TM} is decidable;

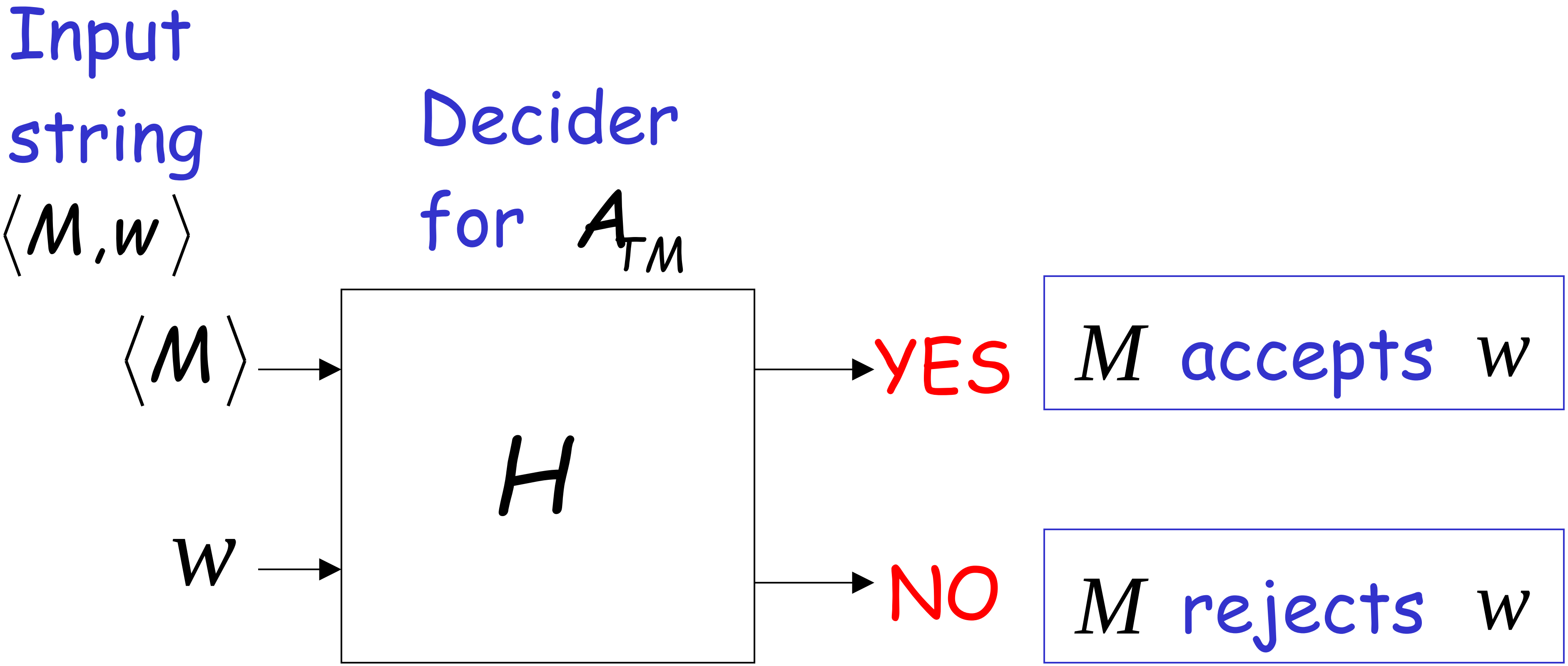
We will then prove that

every decidable language

is Turing-Acceptable

A contradiction!

Suppose that A_{TM} is decidable

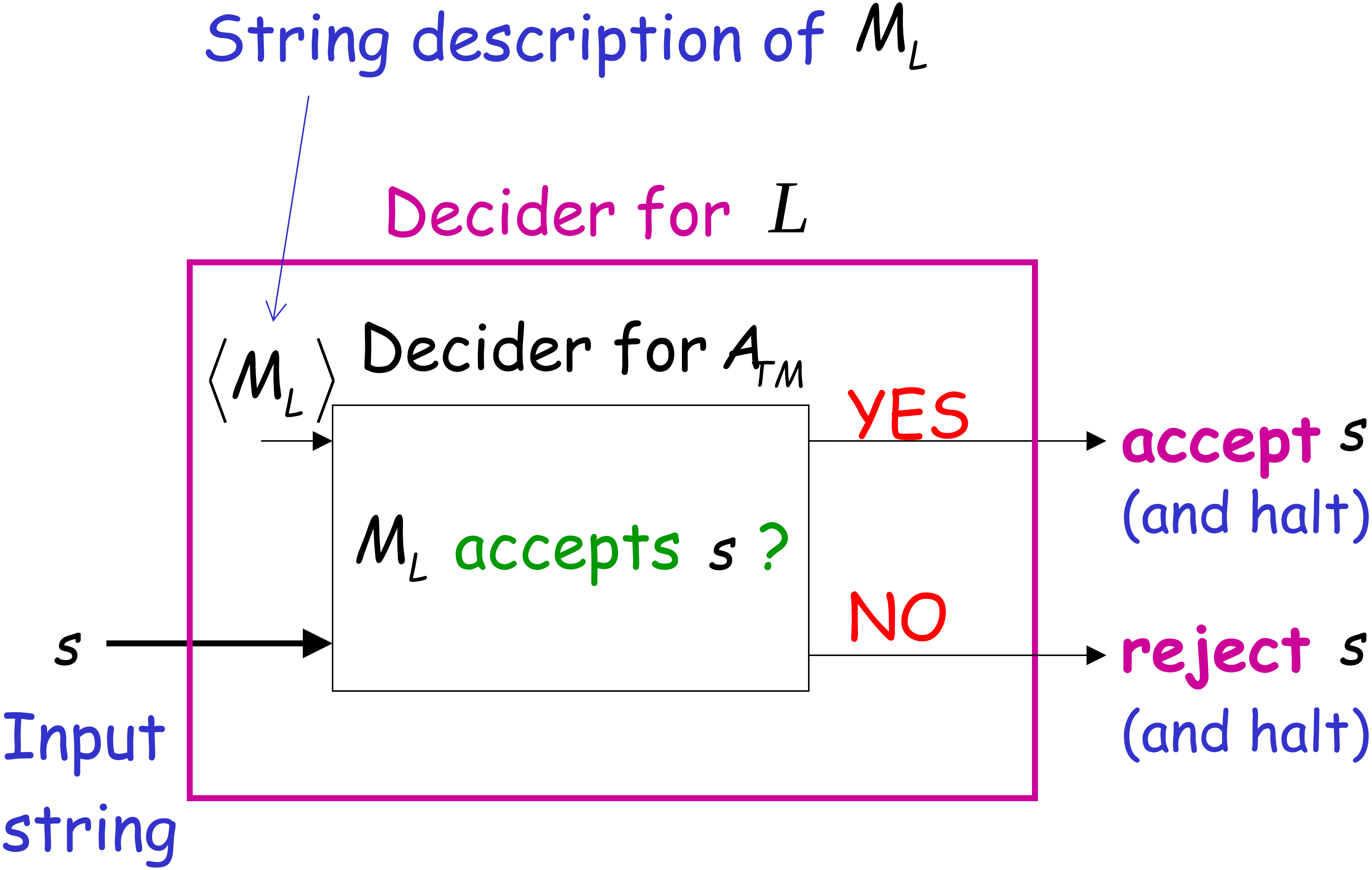


Let L be a Turing recognizable language

Let M_L be the Turing Machine that accepts L

We will prove that L is also decidable:

we will build a decider for L


String description of M_L
Decider for L
$\langle M_L \rangle$
Decider for A_{TM}
M_L accepts s ?
YES
NO
accept s
(and halt)
reject s
(and halt)
s
Input string

Therefore, L is decidable

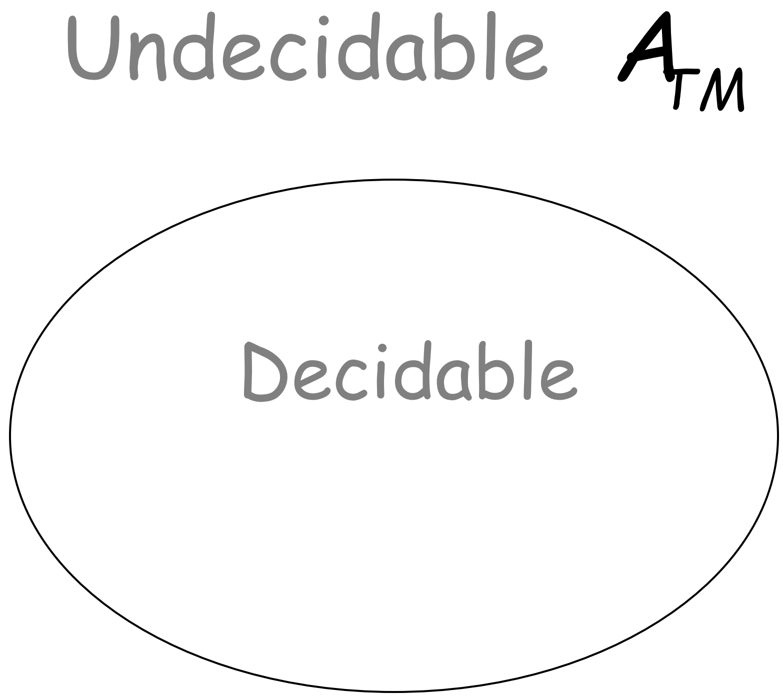
Since L is chosen arbitrarily, every Turing-Acceptable language is decidable

But there is a Turing-Acceptable language which is undecidable

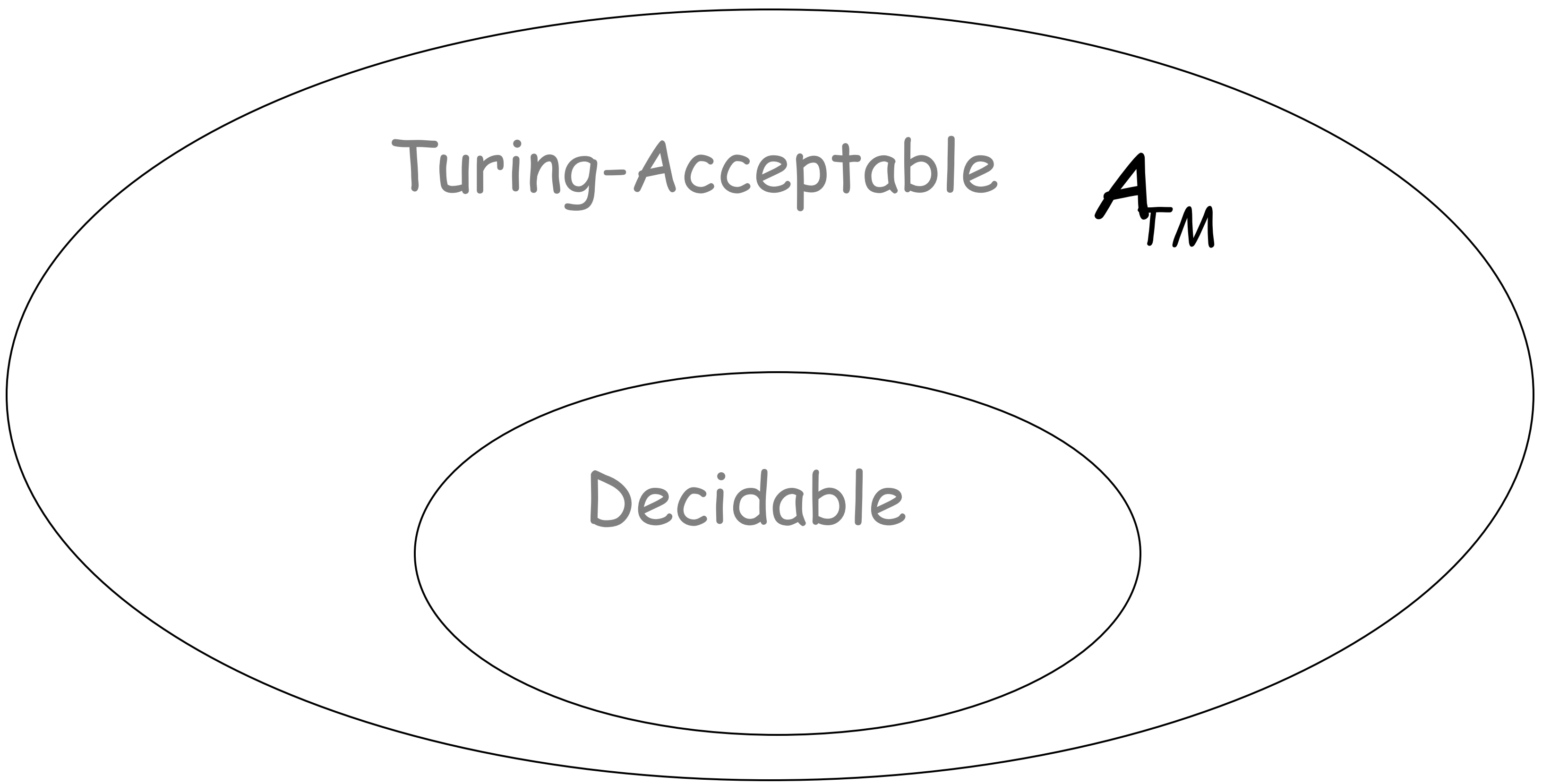
Contradiction!!!!

END OF PROOF

We have shown:



We can actually show:



A_{TM} is Turing-Acceptable

Turing machine that accepts A_{TM} :

$\langle M, w \rangle \longrightarrow$

1. Run M on input w
2. If M accepts w then accept $\langle M, w \rangle$

Halting Problem

Input:
- Turing Machine M
- String w

Question: Does M halt while processing input string w ?

Corresponding language:

$$HALT_{TM} = \{\langle M, w \rangle : M \text{ is a Turing machine that halts on input string } w\}$$

Theorem: $HALT_{TM}$ is undecidable

(The halting problem is unsolvable)

Proof:

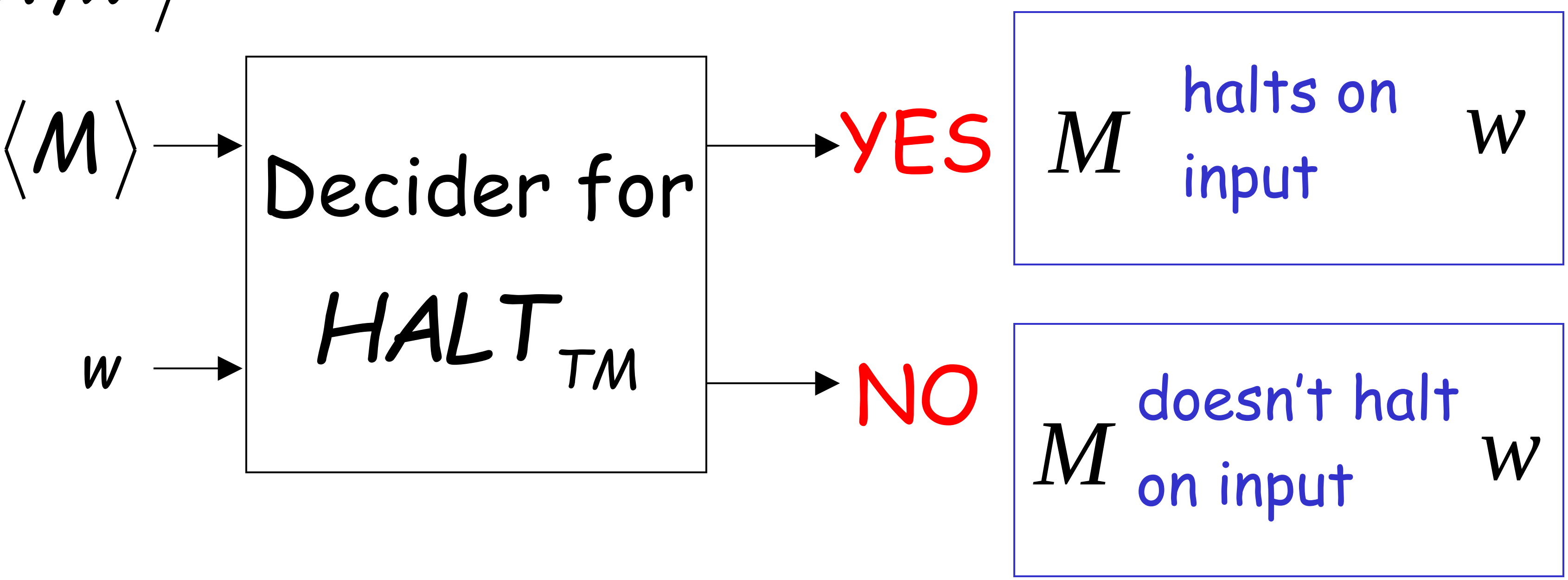
Basic idea:

Suppose that $HALT_{TM}$ is decidable;
we will prove that
every decidable language
is also Turing-Acceptable

A contradiction!

Suppose that $HALT_{TM}$ is decidable

Input string $\langle M, w \rangle$



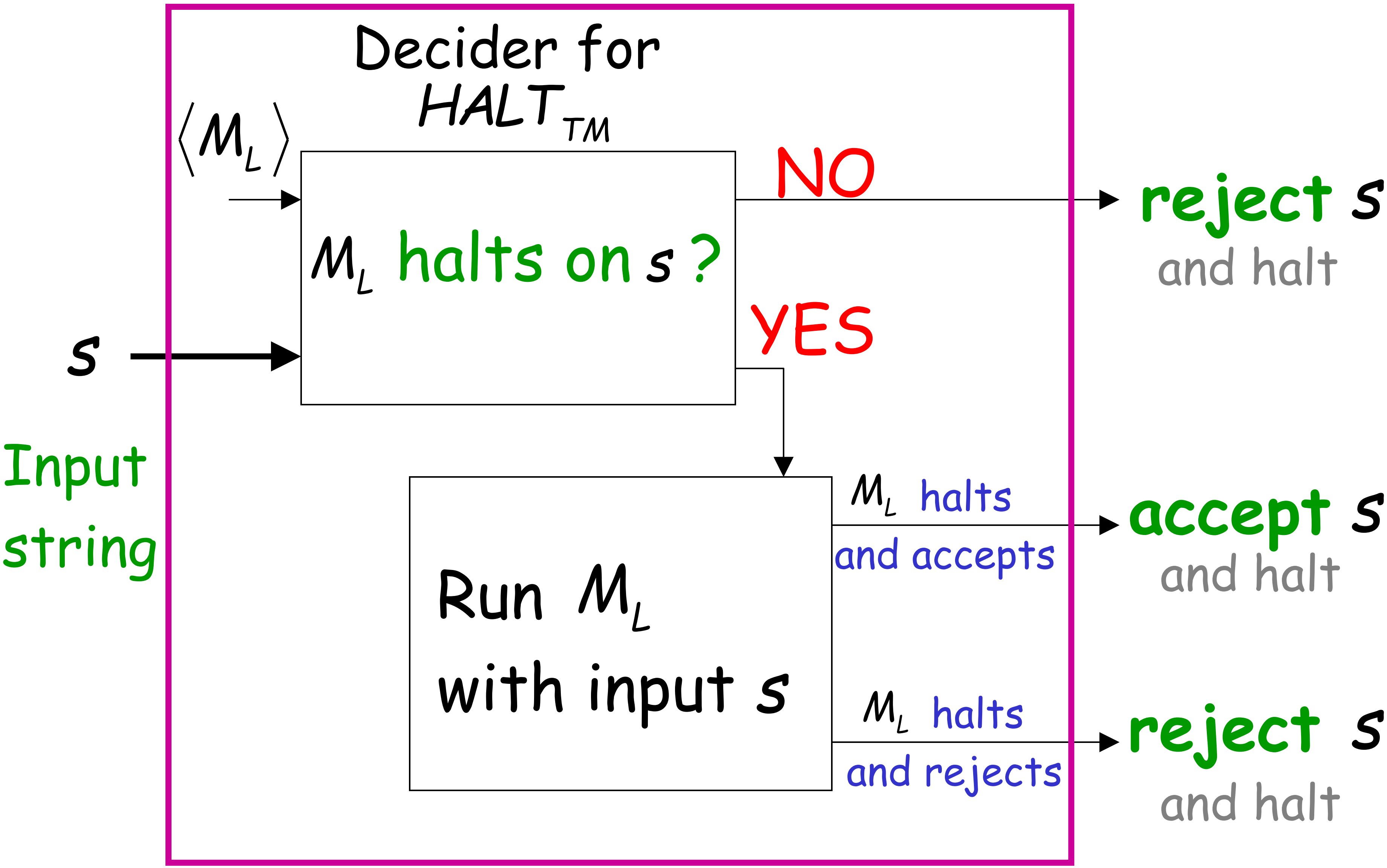
Let L be a Turing-Acceptable language

Let M_L be the Turing Machine that accepts L

We will prove that L is also decidable:

we will build a decider for L

Decider for L



Therefore, L is decidable

Since L is chosen arbitrarily, every Turing-Acceptable language is decidable

But there is a Turing-Acceptable language which is undecidable

Contradiction!!!!

END OF PROOF

An alternative proof

Theorem: $HALT_{TM}$ is undecidable

(The halting problem is unsolvable)

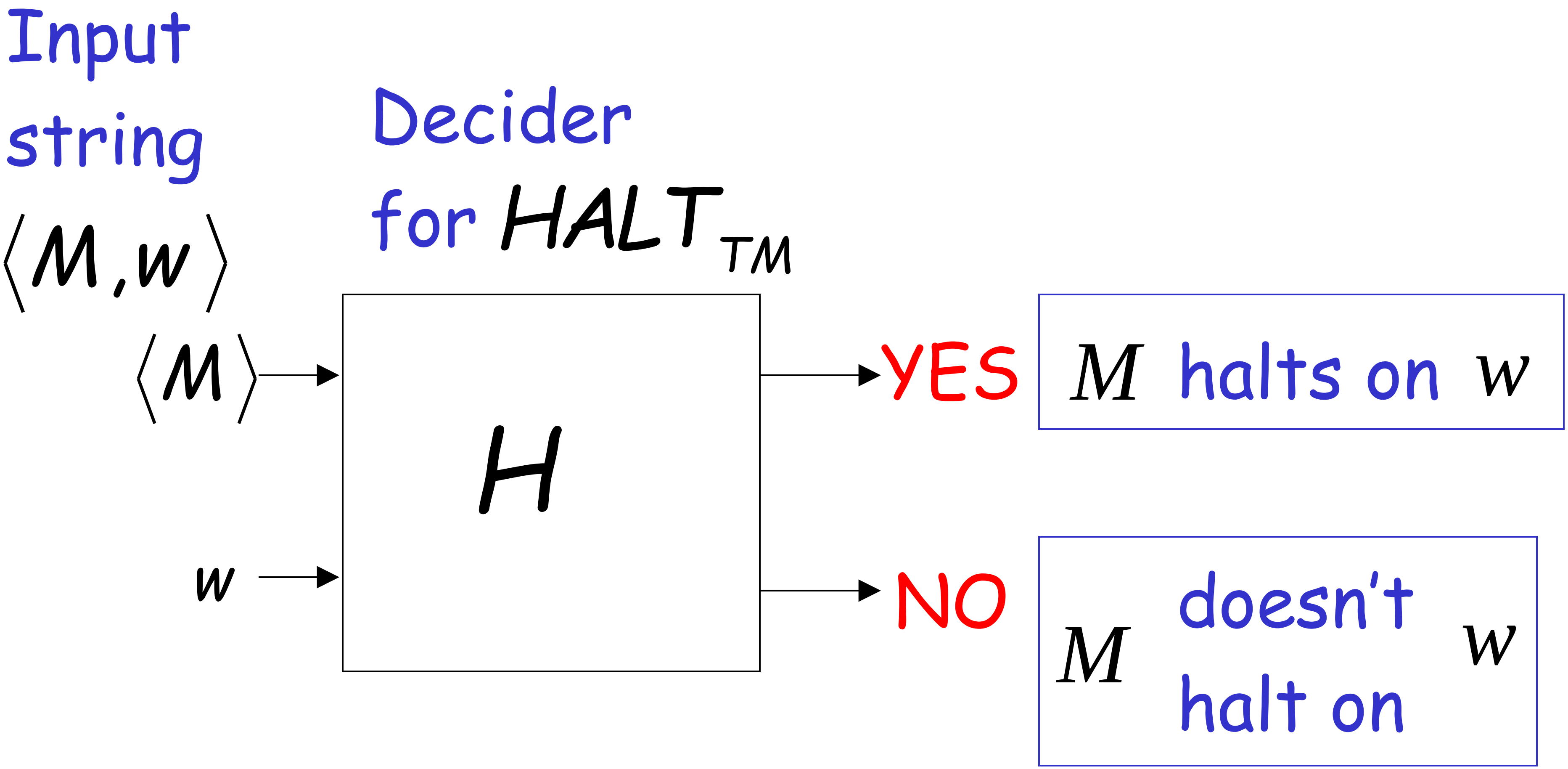
Proof:

Basic idea:

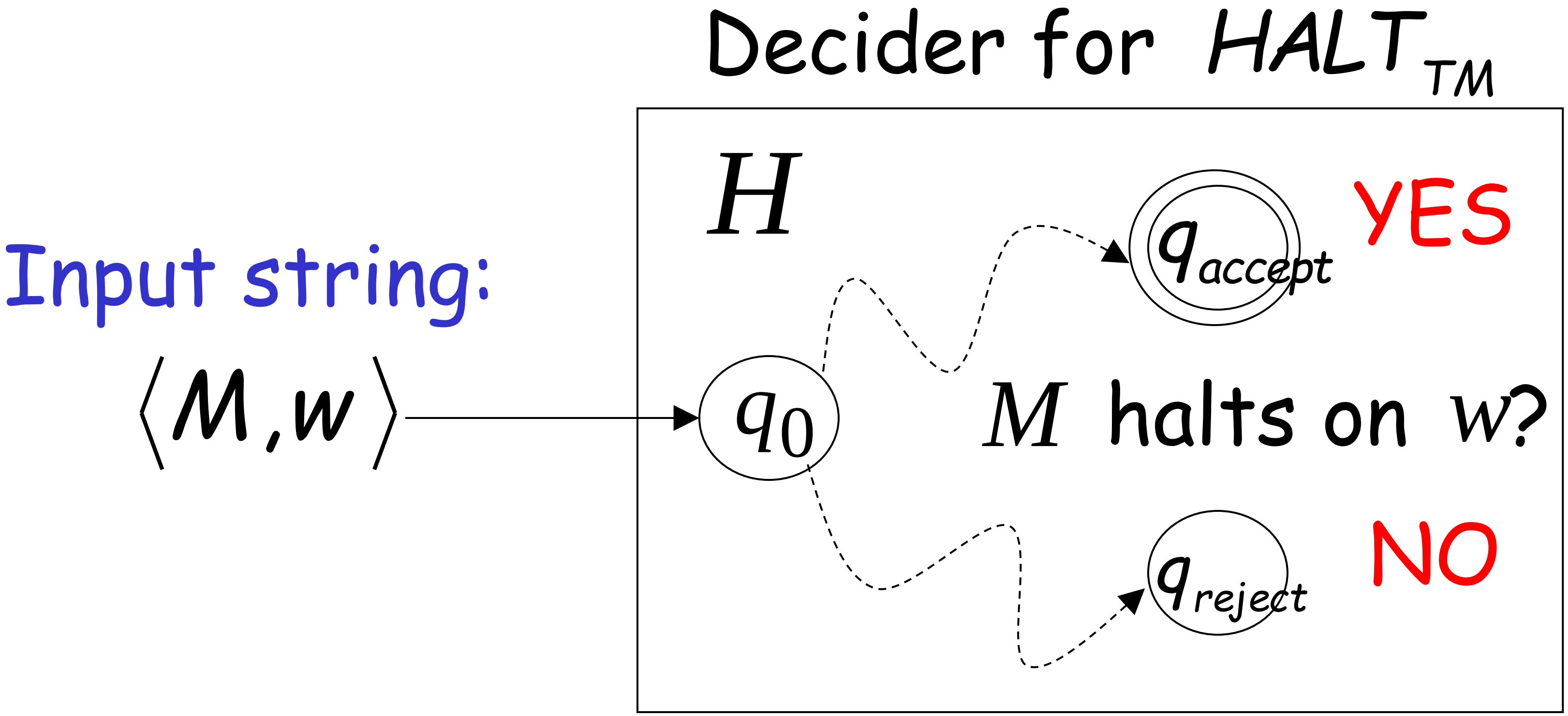
Assume for contradiction that the halting problem is decidable;

we will obtain a contradiction using a diagonilization technique

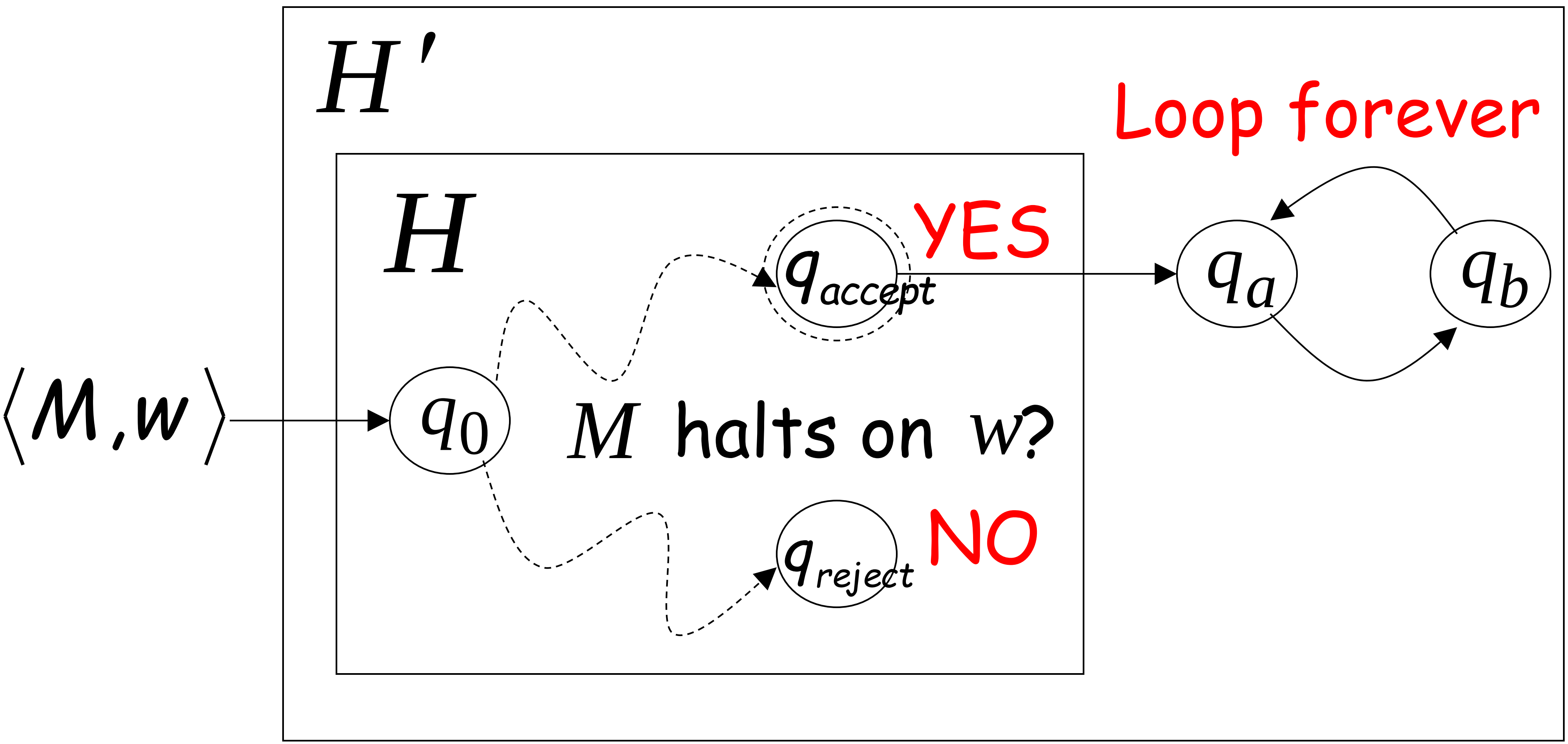
Suppose that $HALT_{TM}$ is decidable



Looking inside H



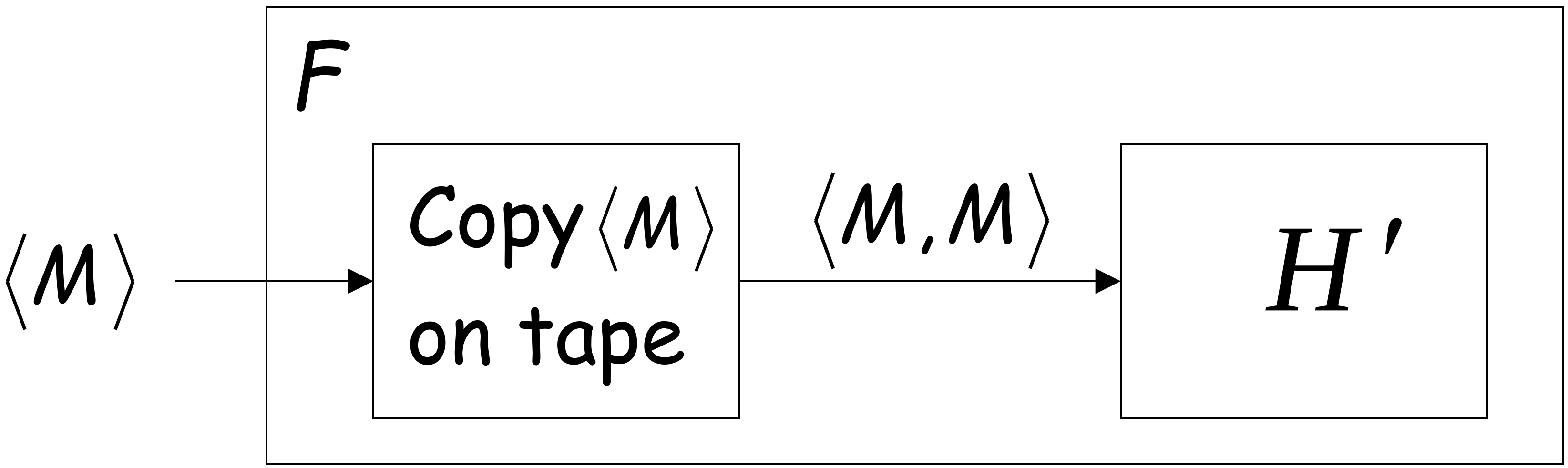
Construct machine H' :



If M halts on input w **Then** Loop Forever

Else Halt

Construct machine F :

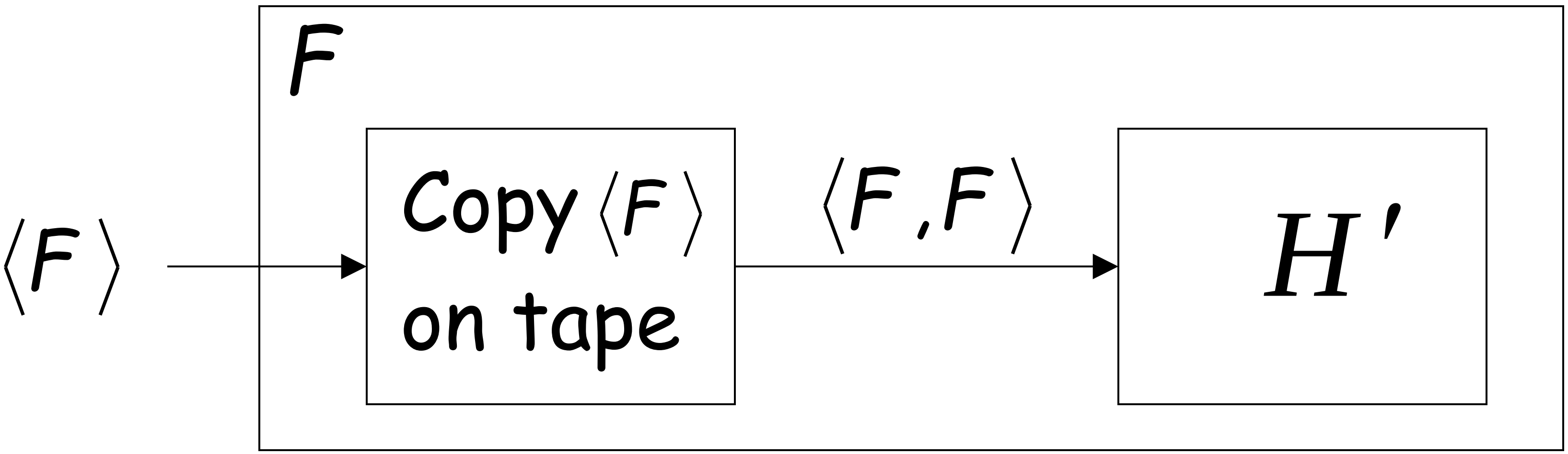


If M halts on input $\langle M \rangle$

Then loop forever

Else halt

Run F with input itself



If F halts on input $\langle F \rangle$

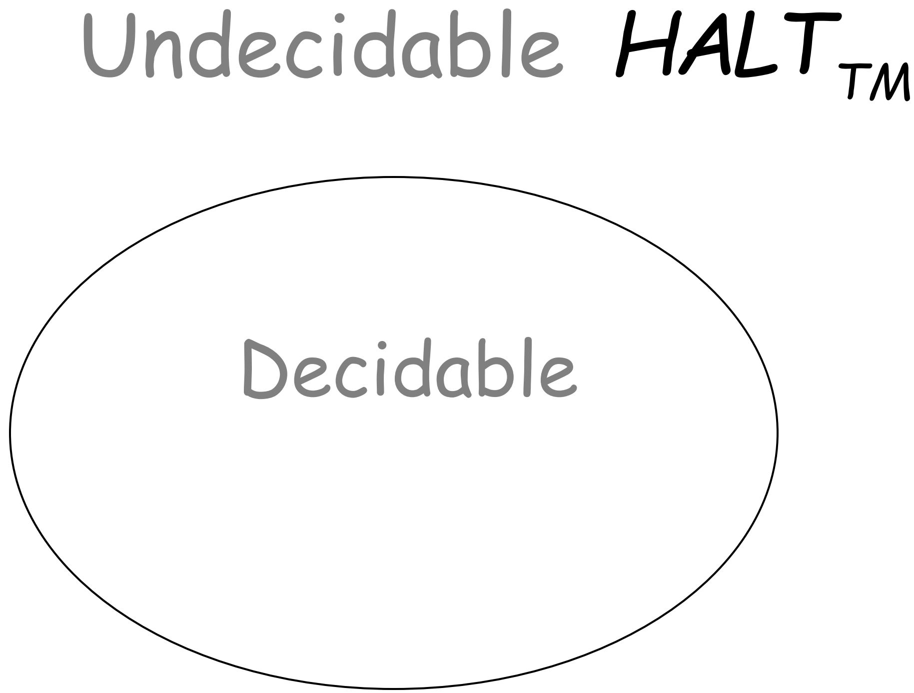
Then F loops forever on input $\langle F \rangle$

Else F halts on input $\langle F \rangle$

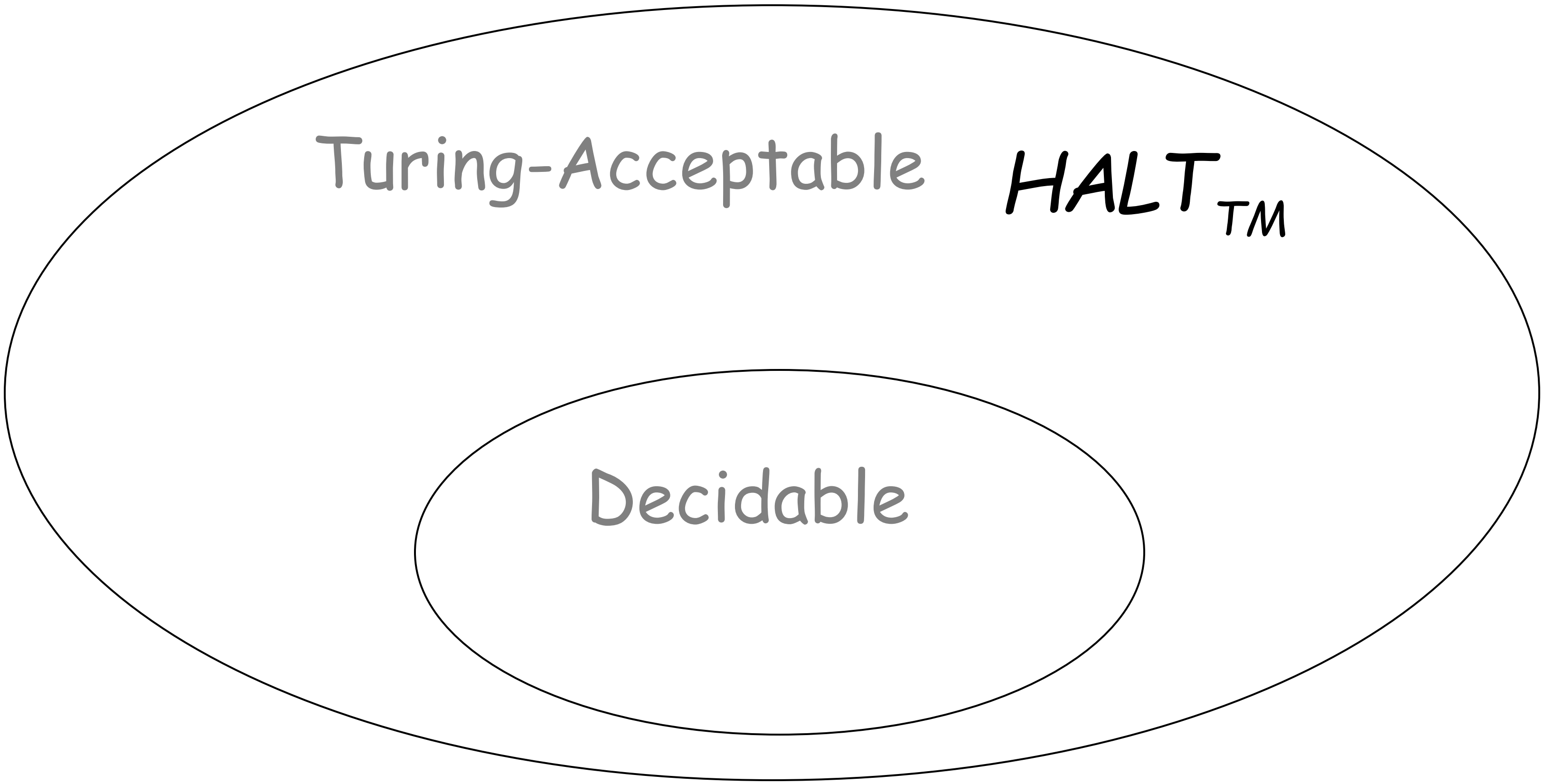
CONTRADICTION!!!

END OF PROOF

We have shown:



We can actually show:



$HALT_{TM}$ is Turing-Acceptable

Turing machine that accepts $HALT_{TM}$:

$\langle M, w \rangle \longrightarrow$

1. Run M on input w
2. If M halts on w then accept $\langle M, w \rangle$