

The Chomsky Hierarchy

Unrestricted Grammars:

Productions

$$u \rightarrow v$$

String of variables
and terminals

String of variables
and terminals

Example unrestricted grammar:

$$S \rightarrow aBc$$

$$aB \rightarrow cA$$

$$Ac \rightarrow d$$

Theorem:

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

Theorem 11.6, Linz, 6th, page 293. Any language generated by an unrestricted grammar is recursively enumerable.

Theorem 11.7, Linz, 6th, page 297. Any language generated by an unrestricted grammar is recursively enumerable.

It is no surprise the unrestricted grammars are recursive enumerable.

A grammar generates strings by a well-defined algorithmic process, so the derivations can be done on a Turing machine.

(An appeal to Church's thesis.)


To show the converse, we describe how any Turing machine can be mimicked by an unrestricted grammar.

The idea is relatively simple, but the complete construction is arduous.

If $q_0 w \vdash^* x q_f w$, then $q_0 w \Rightarrow^* x q_f y$

Context-Sensitive Grammars:

Productions

$$u \rightarrow v$$


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:

$$S \rightarrow abc \mid aAbc$$

$$Ab \rightarrow bA$$

$$Ac \rightarrow Bbcc$$

$$bB \rightarrow Bb$$

$$aB \rightarrow aa \mid aaA$$

Theorem:

A language L is context sensitive
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

Non-recursively enumerable

Recursively-enumerable

Recursive

Context-sensitive

Context-free

Regular

Decidability

A property P of strings is said to be *decidable* if the set of all strings having property P is a recursive set; that is, if there is a total Turing machine that accepts input strings that have property P and rejects those that do not.

P is decidable $\iff \{x \mid P(x)\}$ is recursive.

A is recursive \iff “ $x \in A$ ” is decidable,

Kozen

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
Solves (decides) 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 solves a problem
answers **YES** or **NO** for each instance

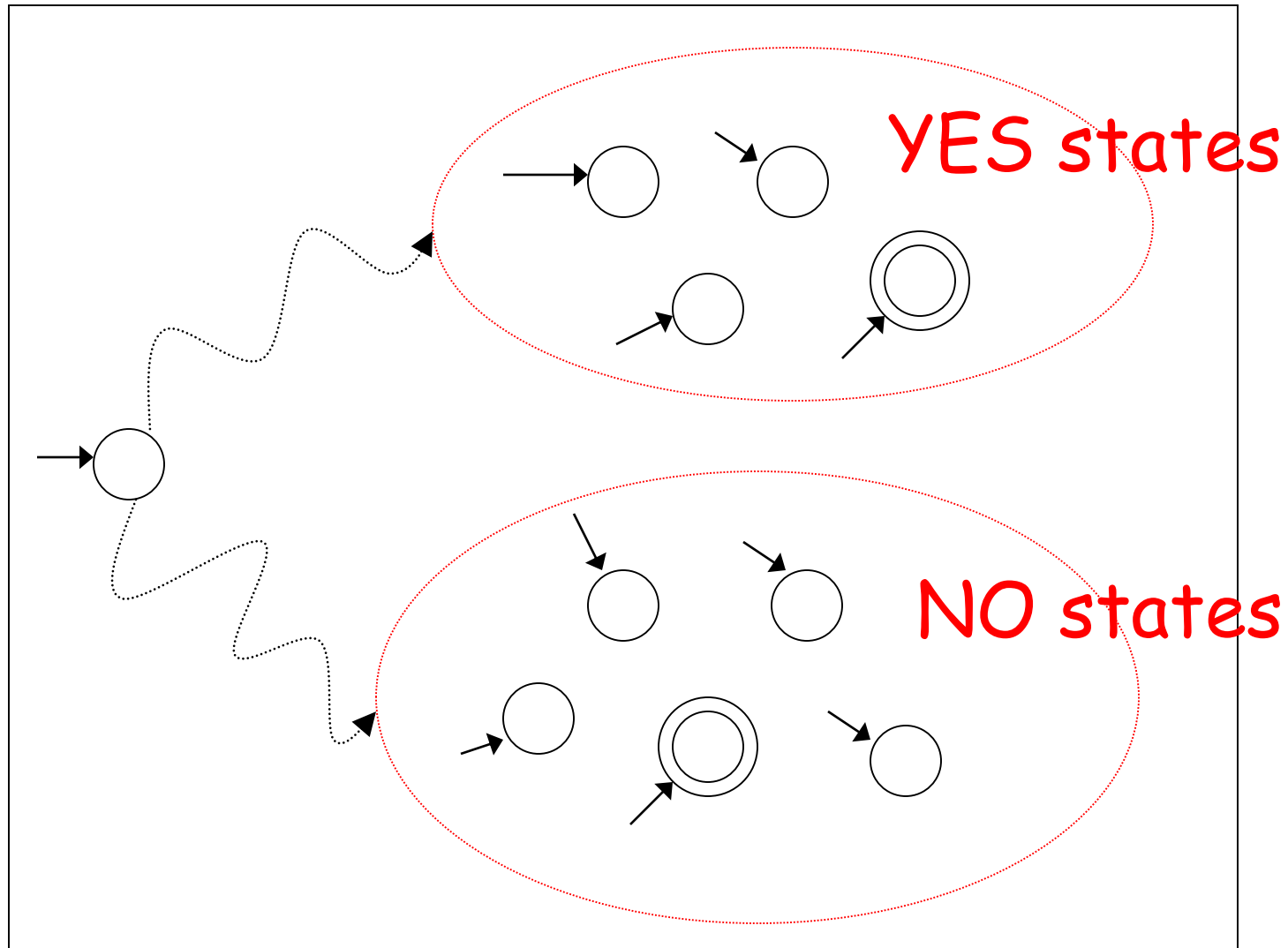


The machine that decides a problem:

- 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

[What does the author mean?]

No harm in assuming YES states are final.

Decidable = Recursive

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 , and
 • String w

Question: Does M accept w ?

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

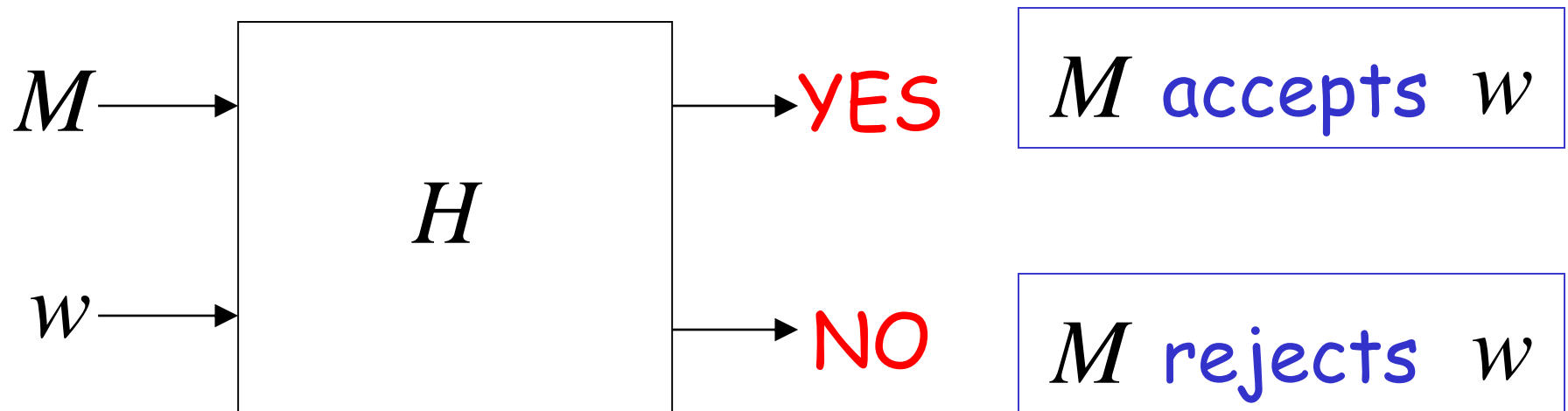
Theorem:

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

Assume there exists a Turing Machine H that decides/solves the membership problem



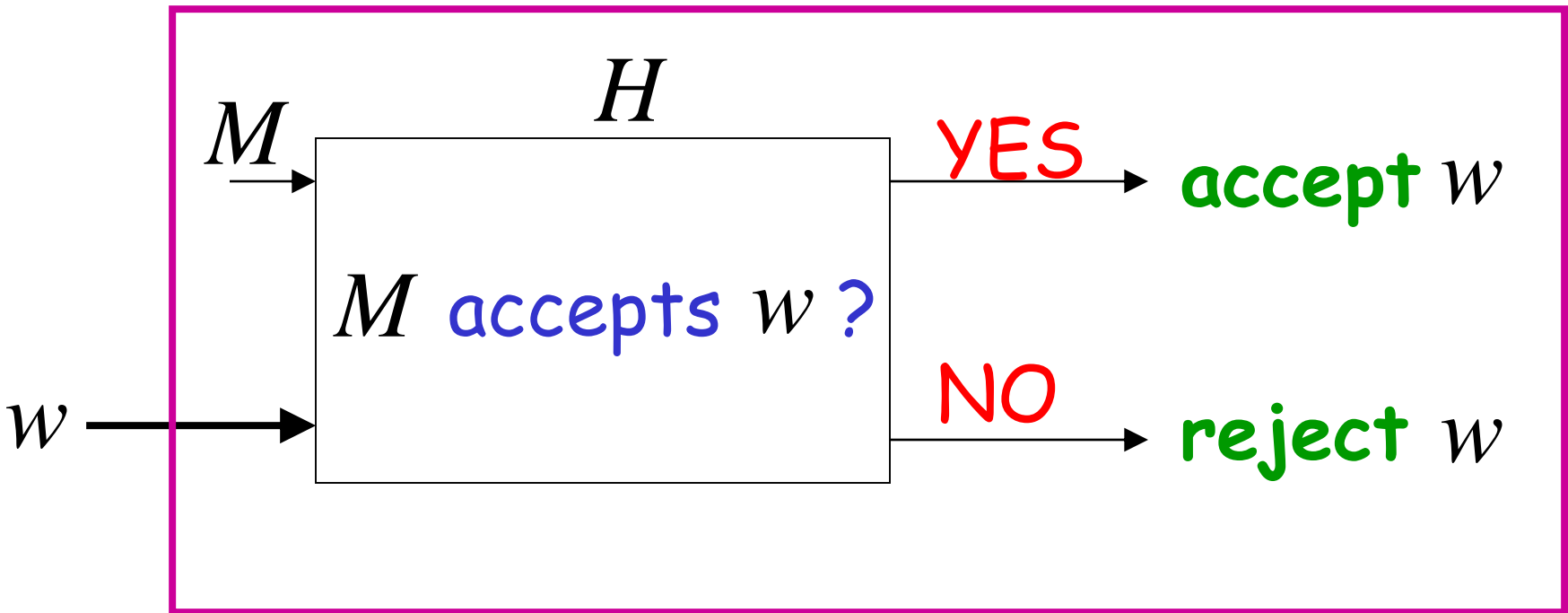
Let L be a recursively enumerable language

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, we have proven that 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

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*,

The Halting Problem

Input: • Turing Machine M , and
 • String w

Question: Does M halt on input w ?

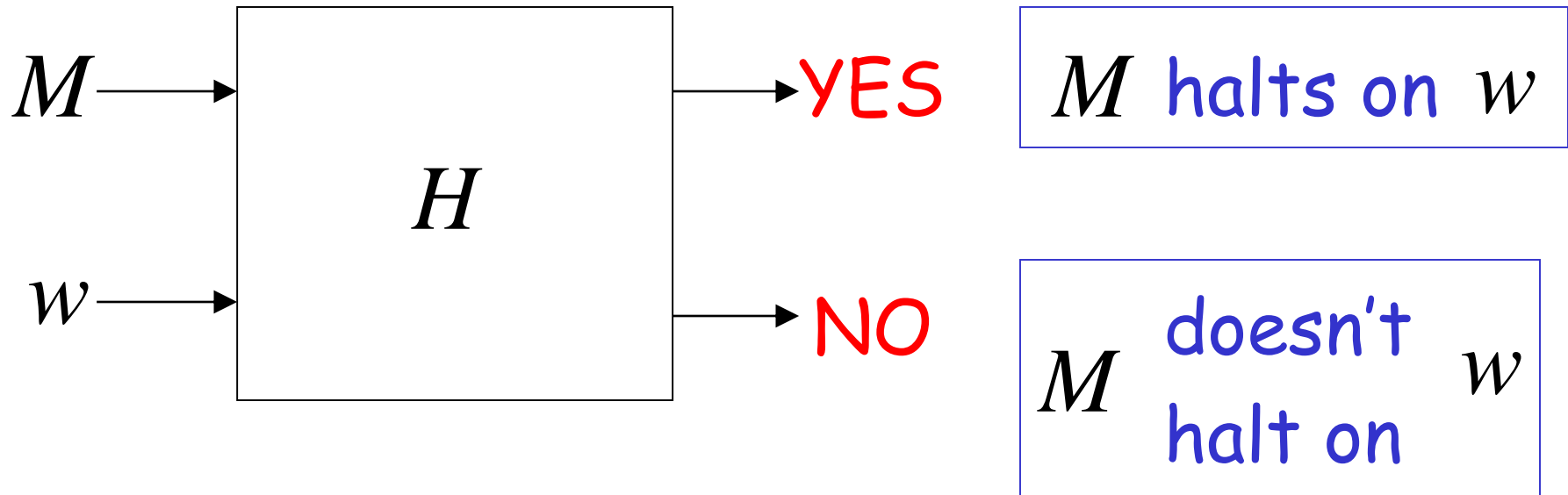
Theorem:

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

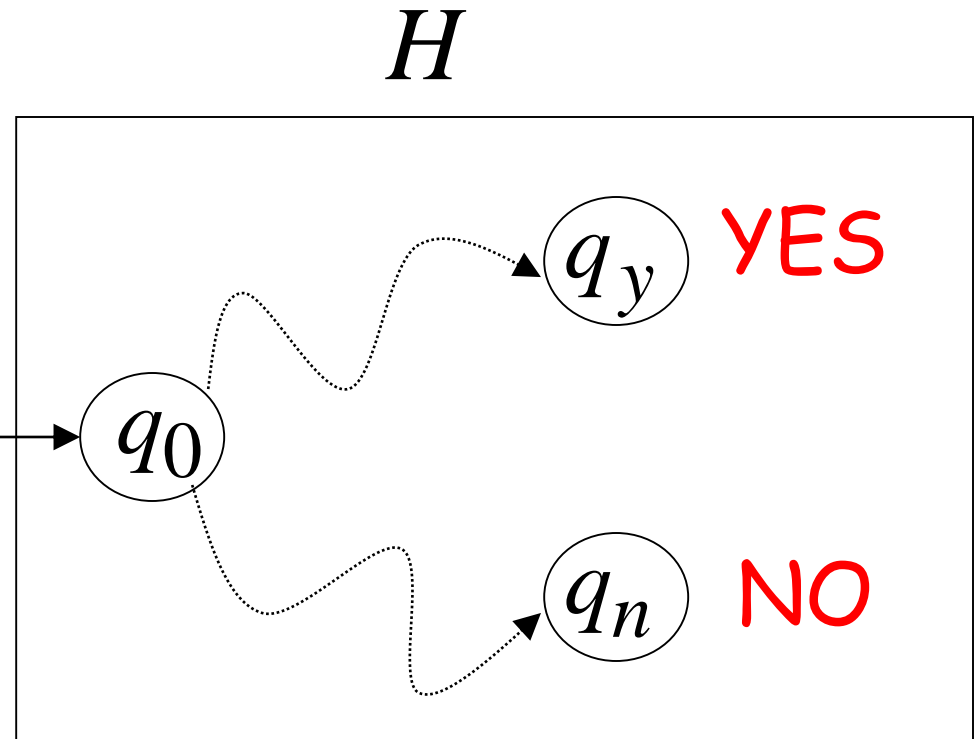


Construction of H

Input:
initial tape contents

Encoding
of M w_M

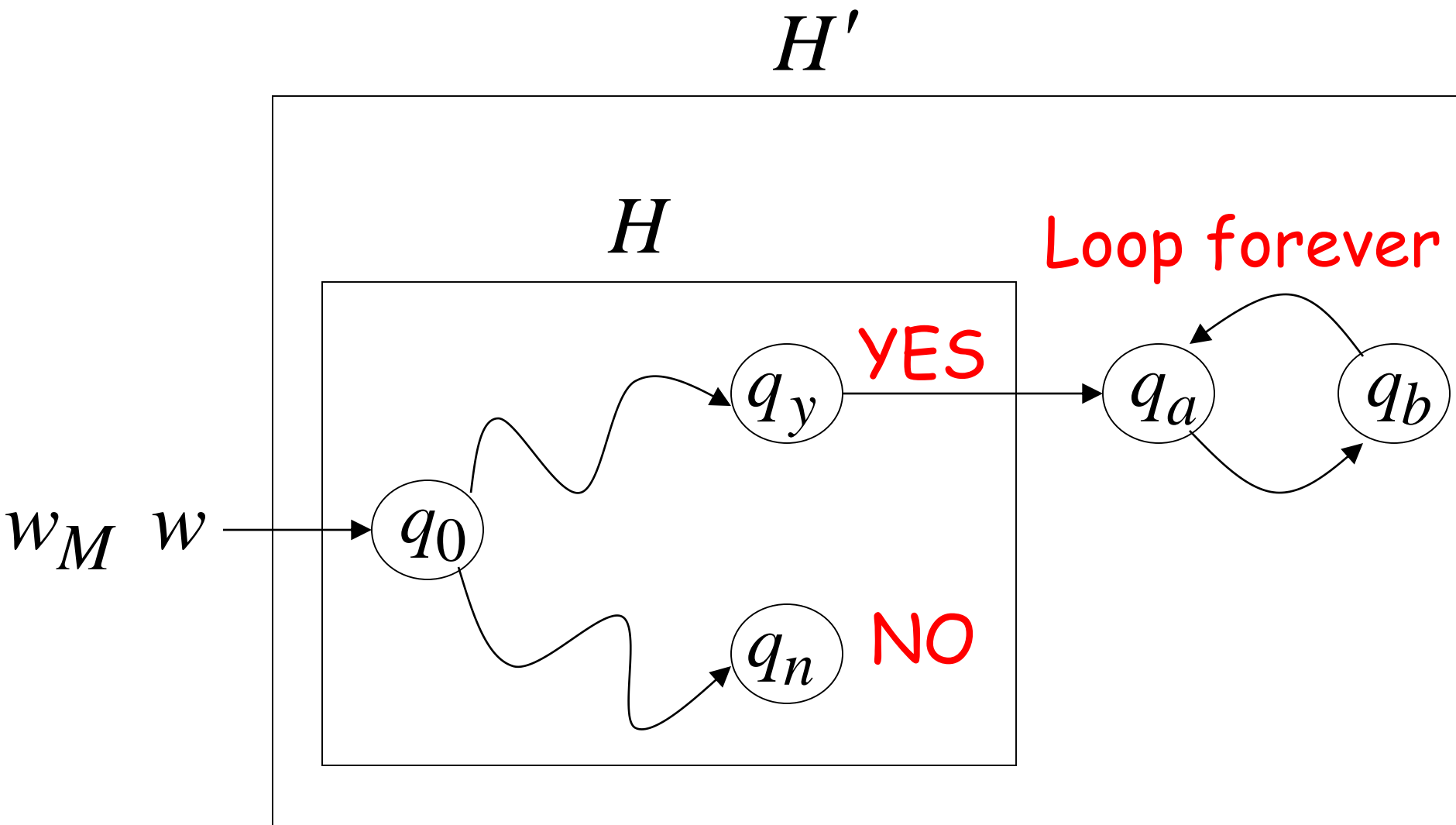
String
 w



Construct machine H' :

If H returns YES then loop forever

If H returns NO then halt



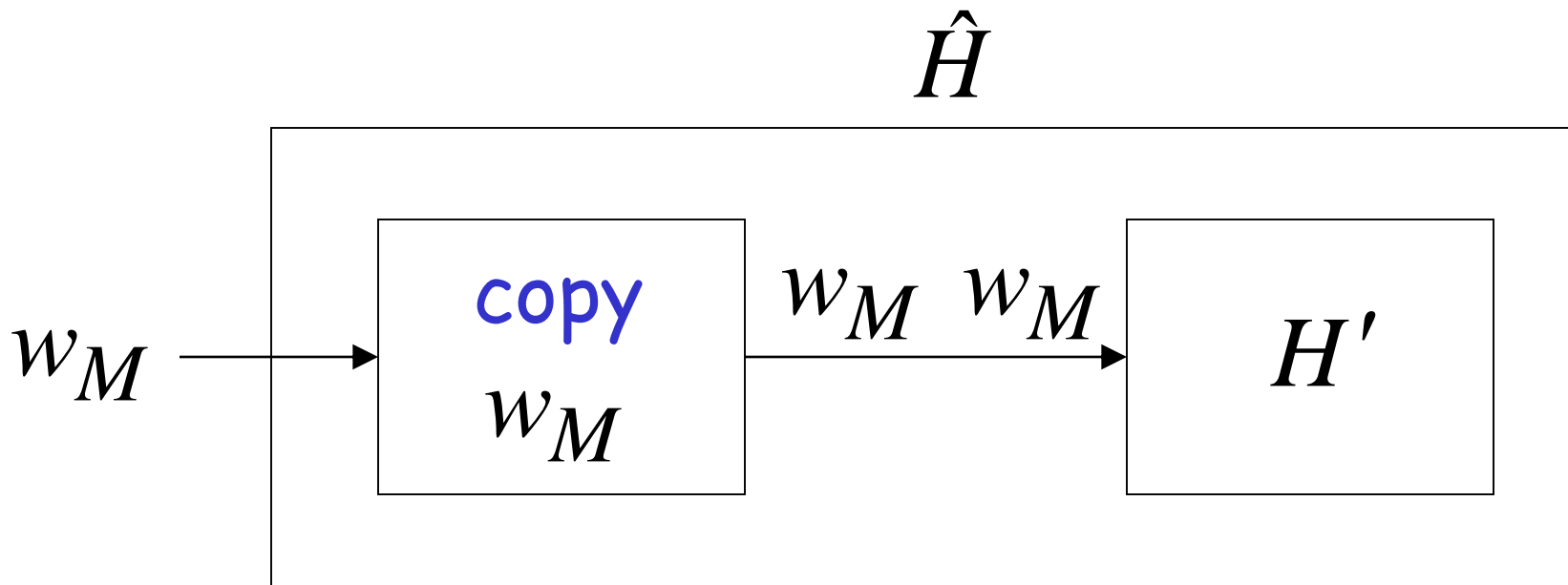
Construct machine \hat{H} :

Input: w_M (machine M)

If M halts on input w_M

Then loop forever

Else halt



Run machine \hat{H} with input itself:

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

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

Then loop forever

Else 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 were decidable then every recursively enumerable language would be recursive.

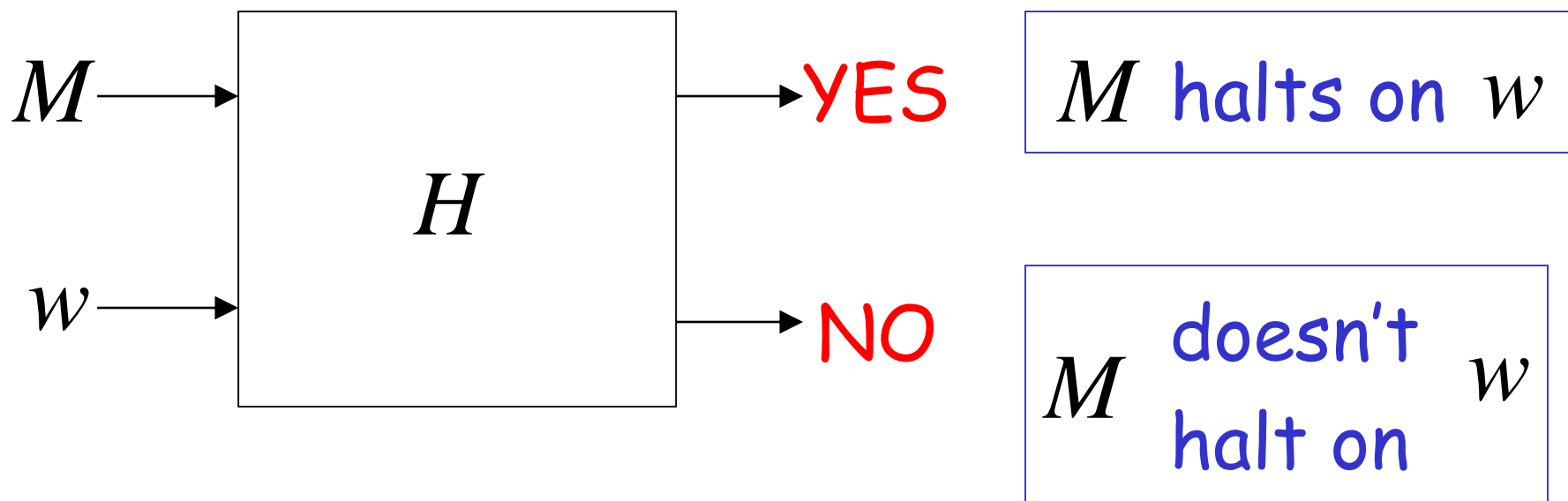
If $L = \{M_w \# w \mid M_w \text{ halts on } w\}$ were recursive, then every r.e. set is recursive.

Theorem:

The halting problem is undecidable

Proof: Assume for contradiction that the halting problem is decidable

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

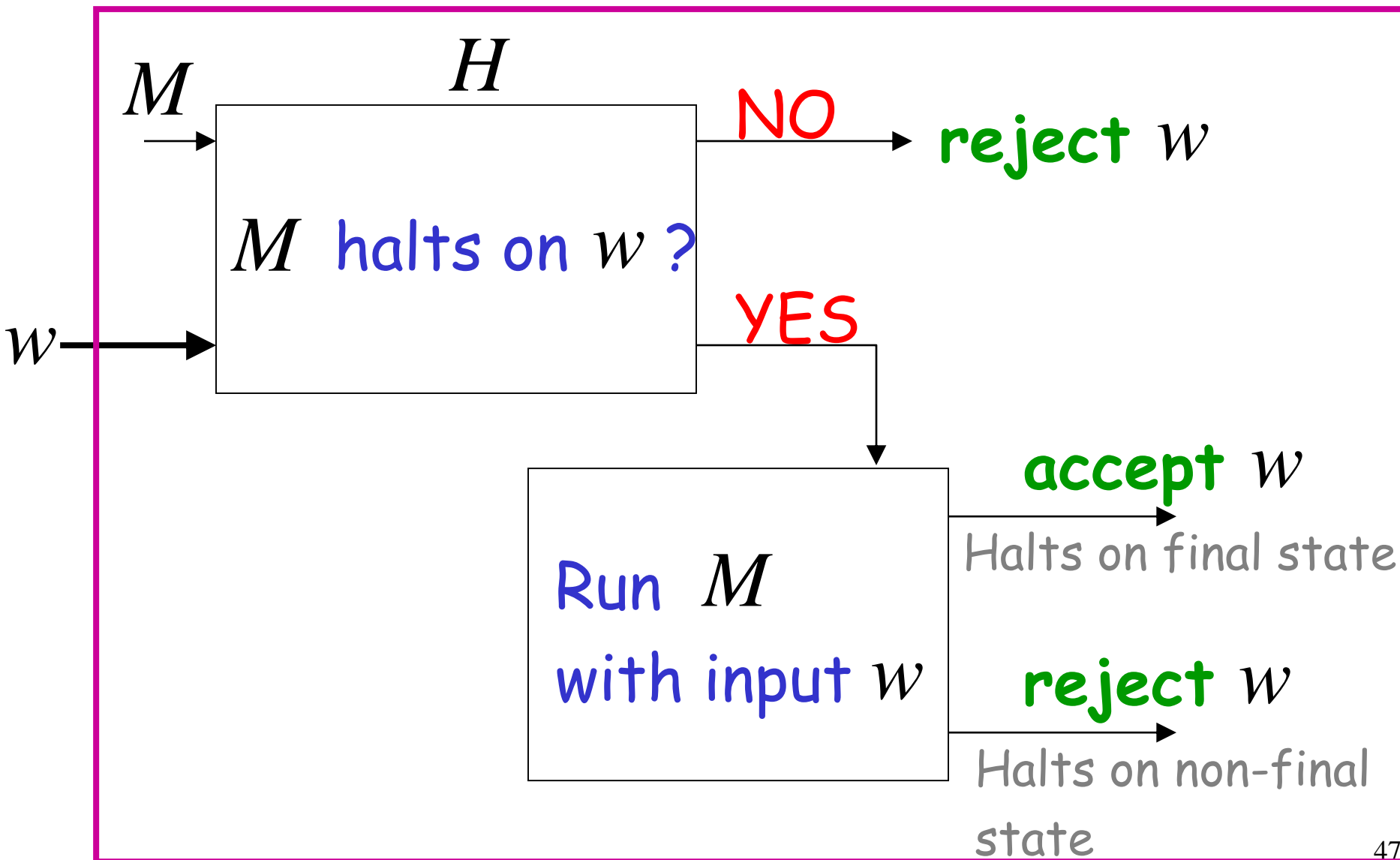


Let L be a recursively enumerable language

Let M be the Turing Machine that accepts L

we will describe a Turing machine that accepts L and halts on any input, proving that L is also recursive.

Turing Machine that accepts L and halts on any input



Therefore L is recursive

Since L is chosen arbitrarily, we have proven that 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