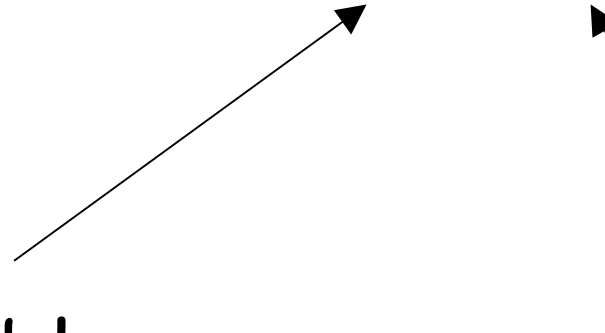
The Chomsky Hierarchy

Unrestricted Grammars:

Rules have form $u \rightarrow v$



String of variables and terminals

String of variables and terminals

Example grammar:

$$S \rightarrow aBc$$

$$aB \rightarrow cA$$

$$Ac \rightarrow d$$

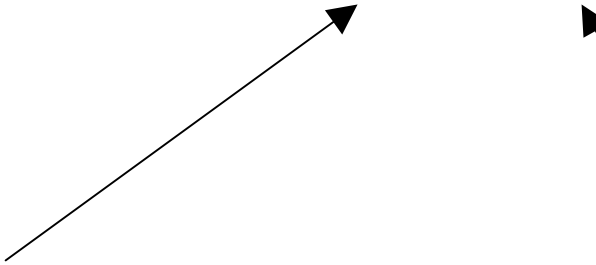
A language L is generated by an unrestricted grammar

if and only if

L is recursively enumerable

Context-Sensitive Grammars:

Rules have form $u \rightarrow v$



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:

$$S \to abc \mid aAbc$$

$$Ab \to bA$$

$$Ac \to Bbcc$$

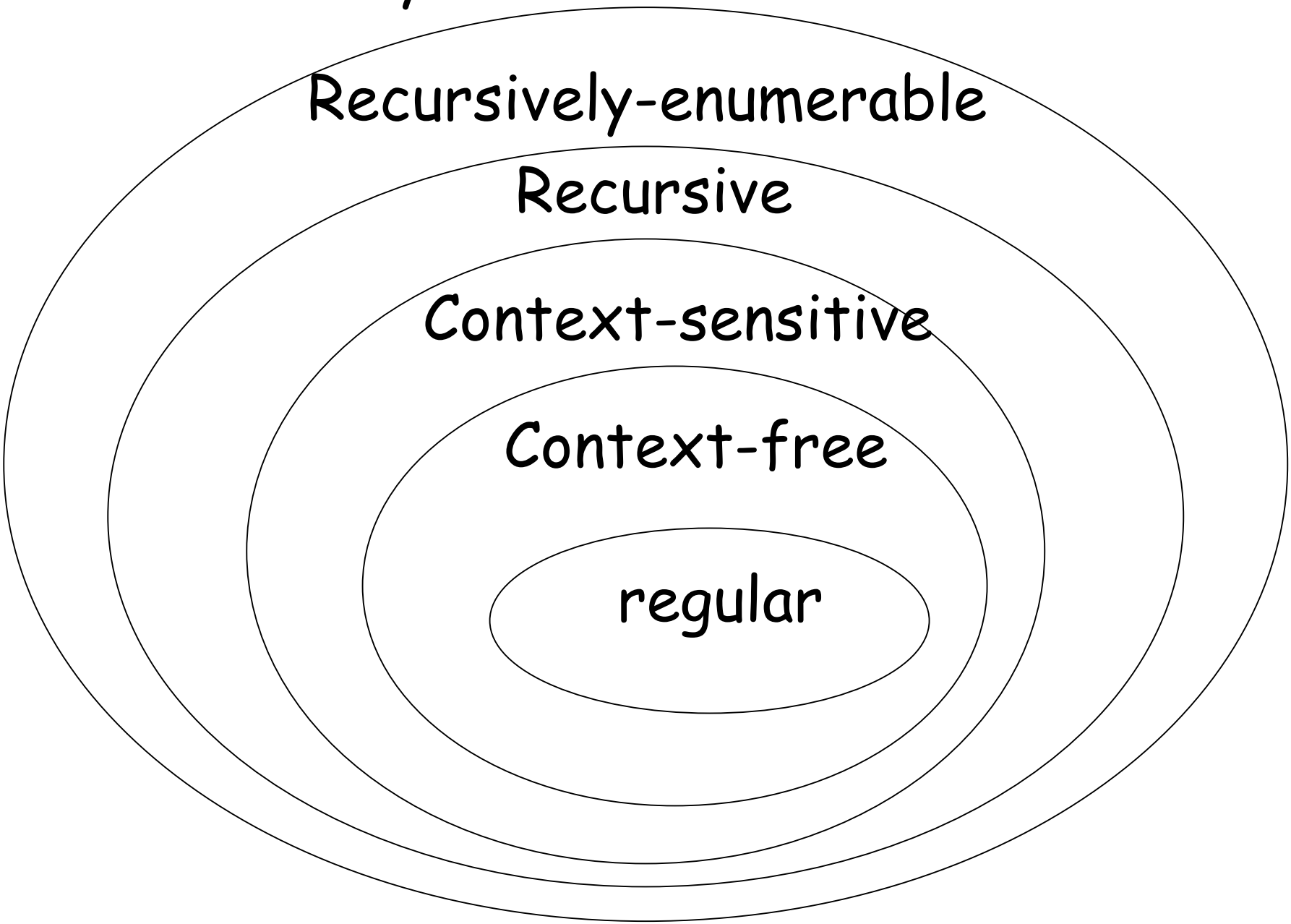
$$bB \to Bb$$

$$aB \to aa \mid aaA$$

A language L is context sensistive if and only if

L is accepted by a Linear-Bounded automaton

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


Non-recursively enumerable
Recursively-enumerable
Recursive
Context-sensitive
Context-free
regular

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 solves the problem

Some decidable problems:

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

The machine that decides the 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

Difference between

Recursive Languages and
Decidable problems

For decidable problems:

The YES halting states may not be final states

There are some problems
which are undecidable:

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

A famous undecidable problem:

The halting problem

The Halting Problem

Inputs:

- Turing Machine M
- String w

Question: Does M halt on w ?

THEOREM

The halting problem is undecidable

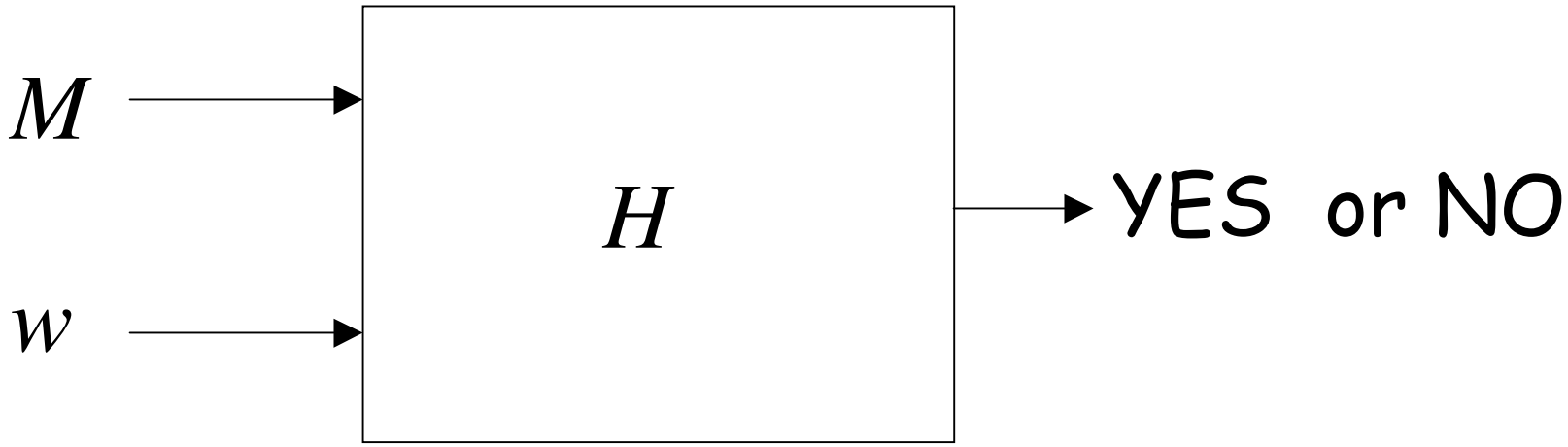
THEOREM

The halting problem is undecidable

PROOF

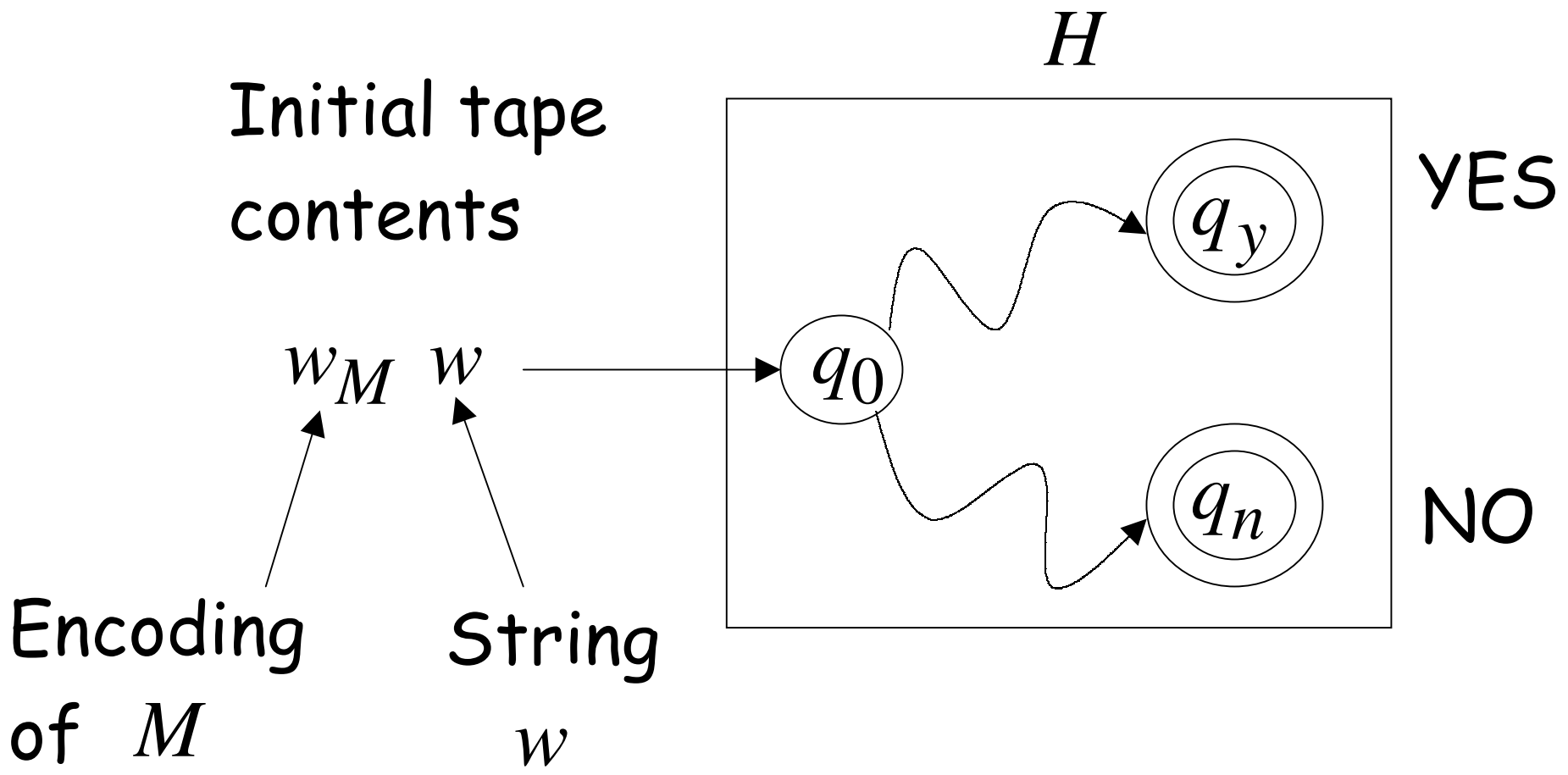
Assume for contradiction that
the halting problem is decidable

There exists Turing Machine H
that solves the halting problem



YES: M accepts w

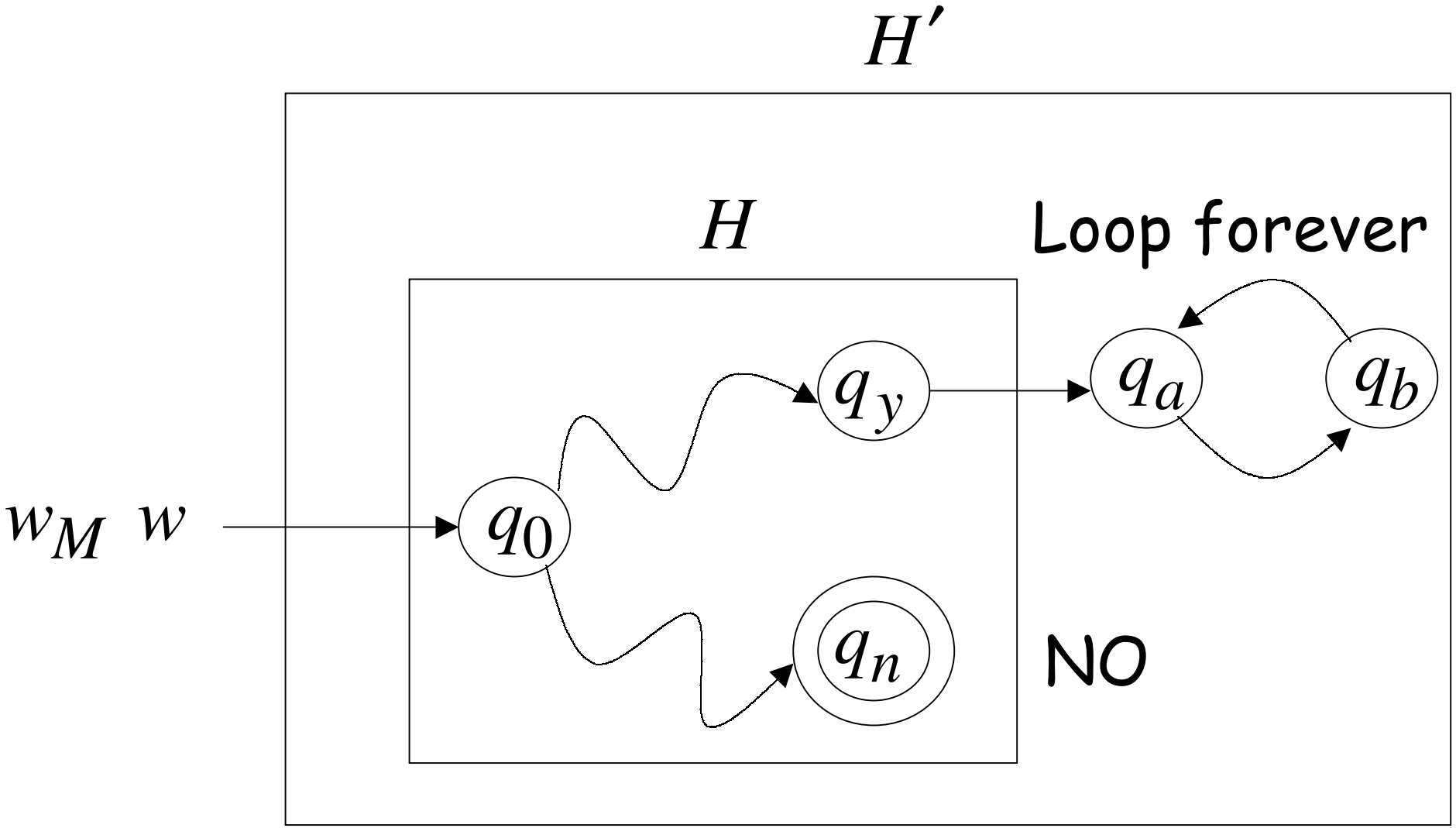
NO: M Doesn't accept w


H
Initial tape contents
w_M w
q_0
q_y
q_n
YES
NO
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
q_y
q_a
q_b
w_M w
q_0
q_n
NO

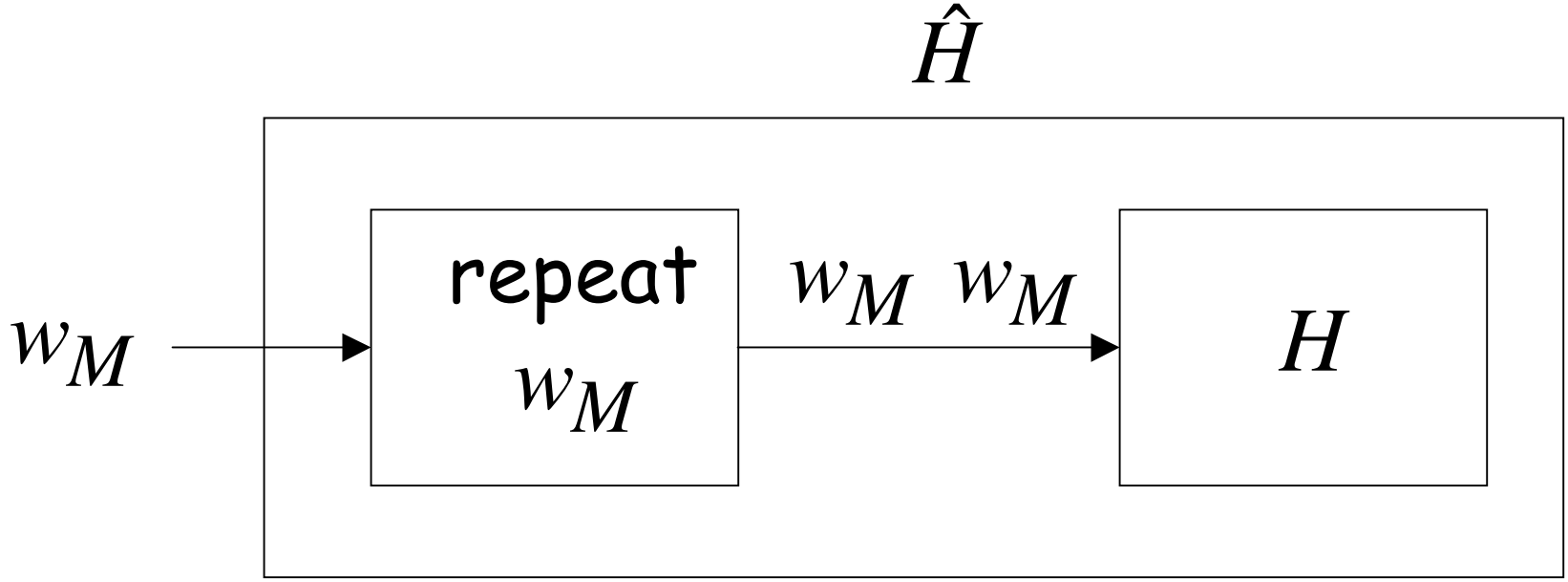
Construct machine $\hat{H}$:

Input: w_M (machine M)

If M halts on input w_M

then loop forever

Else halt


$\hat{H}$
w_M
repeat
w_M
w_M w_M
H

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

Run machine $\hat{H}$:

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 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

Turing Machine H solves the halting problem

For inputs: machine M and string w

H Returns:

YES: if M halts on input w

NO: if M doesn't halt on input w

Let L be a recursively enumerable language

Let M be the Machine that accepts L

We will prove that L is recursive

We will describe a membership algorithm

Run H with inputs: M and w

If H Returns:

NO: M doesn't halt on input w

Then M doesn't accept w

If H Returns:

YES: M halts on input w

Run M with input w

M will halt, and either accept or reject w

Therefore L is recursive

But there are recursively enumerable languages which are not recursive

Contradiction!!!!

Therefore, the halting problem is undecidable

END OF PROOF

Reducibility

Problem A is reduced to problem B

means

If B is decidable then A is decidable

Problem A is reduced to problem B

also means

If A is undecidable then B is undecidable

Example:

The halting problem

is reduced to

The state-entry problem

The state-entry problem

Inputs:

- Turing Machine M
- State q
- String w

Question:

Does M enter state q on input w ?

THEOREM

The state-entry problem is undecidable

THEOREM

The state-entry problem is undecidable

PROOF

Reduce the halting problem to

the state-entry problem

Suppose we have an algorithm
for the state-entry problem

We will construct an algorithm
for the halting problem

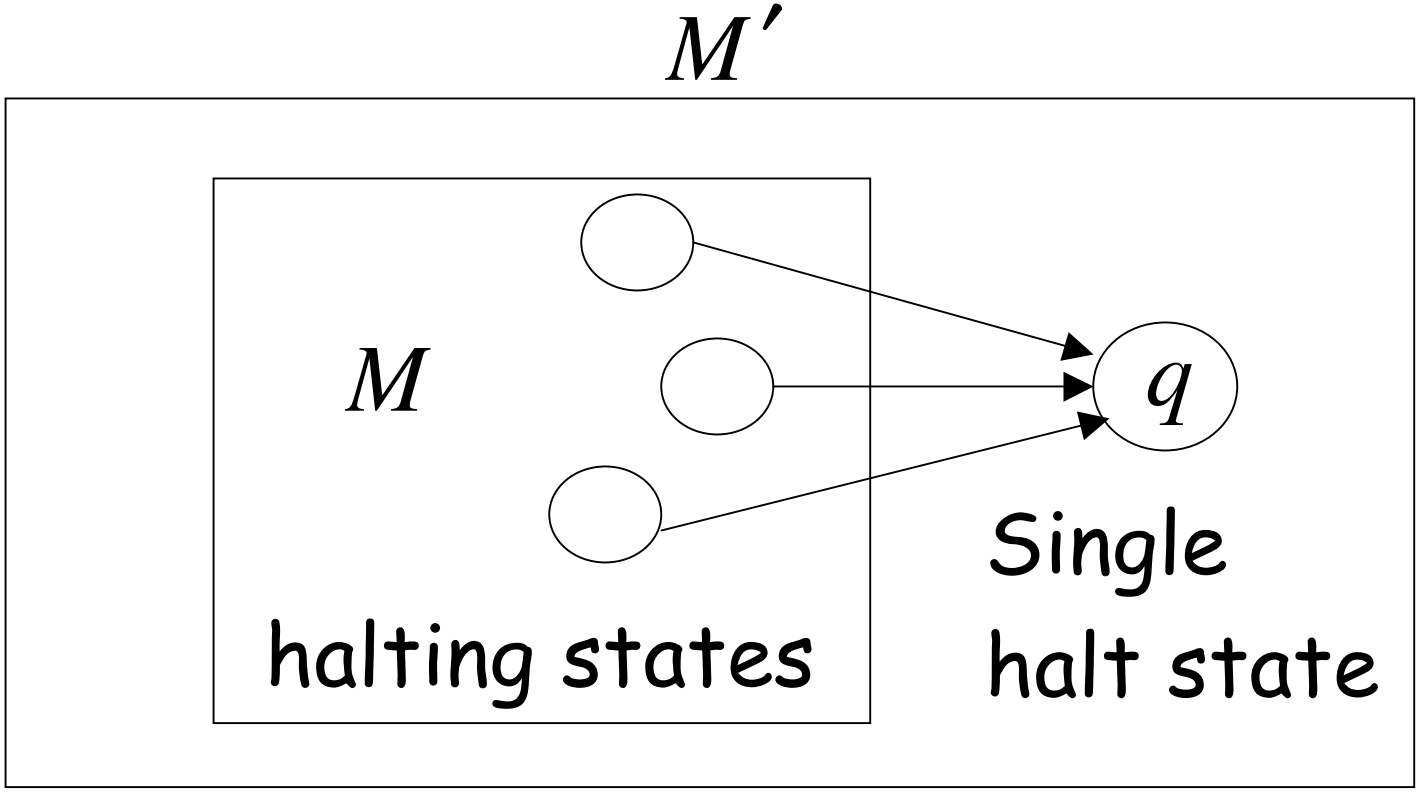
Inputs for the halting problem

- A machine M
- A string w

Algorithm for halting problem must determine if M halts on input w

Modify machine M :

- Add new state q
- From any halting state add transitions to q



M halts if and only if

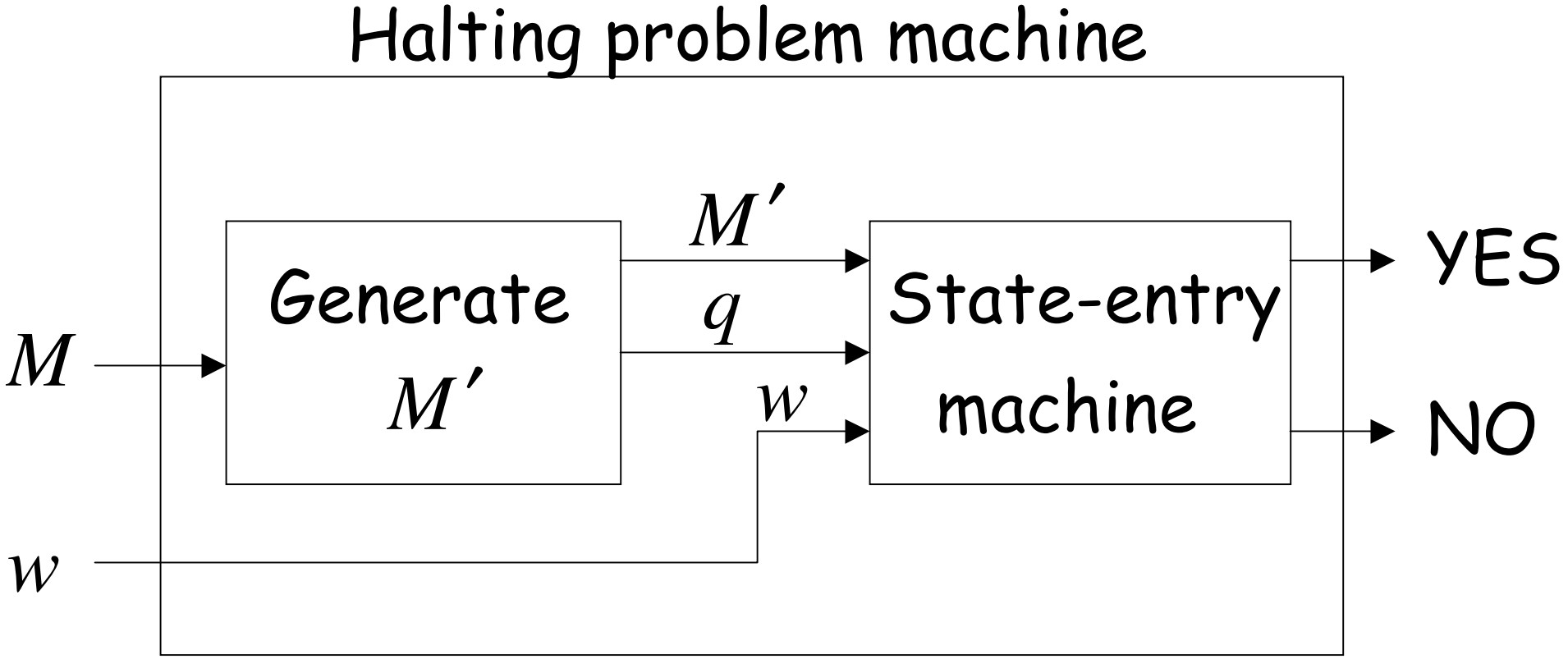
M' halts on state q

Algorithm for halting problem:

Inputs: M and w

1. Construct M' with state q

2. Run algorithm for state-entry problem with inputs: M' , q , w


Halting problem machine
M
w
Generate
M′
M′
q
w
State-entry
machine
YES
NO

Since the halting problem is undecidable,

the state-entry problem is also undecidable

END OF PROOF

Another example:

The halting problem

is reduced to

The blank-tape halting problem

The blank-tape halting problem

Input: Turing Machine M

Question:

Does M halt when started with a blank tape?

THEOREM

The blank-tape halting problem is undecidable

THEOREM

The blank-tape halting problem is undecidable

PROOF

Reduce halting problem to

Blank-tape halting problem

Suppose we have an algorithm
for the blank-tape halting problem

We will construct an algorithm
for the halting problem

Inputs for the halting problem

- A machine M
- A string w

Algorithm for halting problem must determine if M halts on input w

Construct a new machine M_w

- On blank tape writes w
- Then continues execution like M

M halts if and only if

M_w halts when started with blank tape

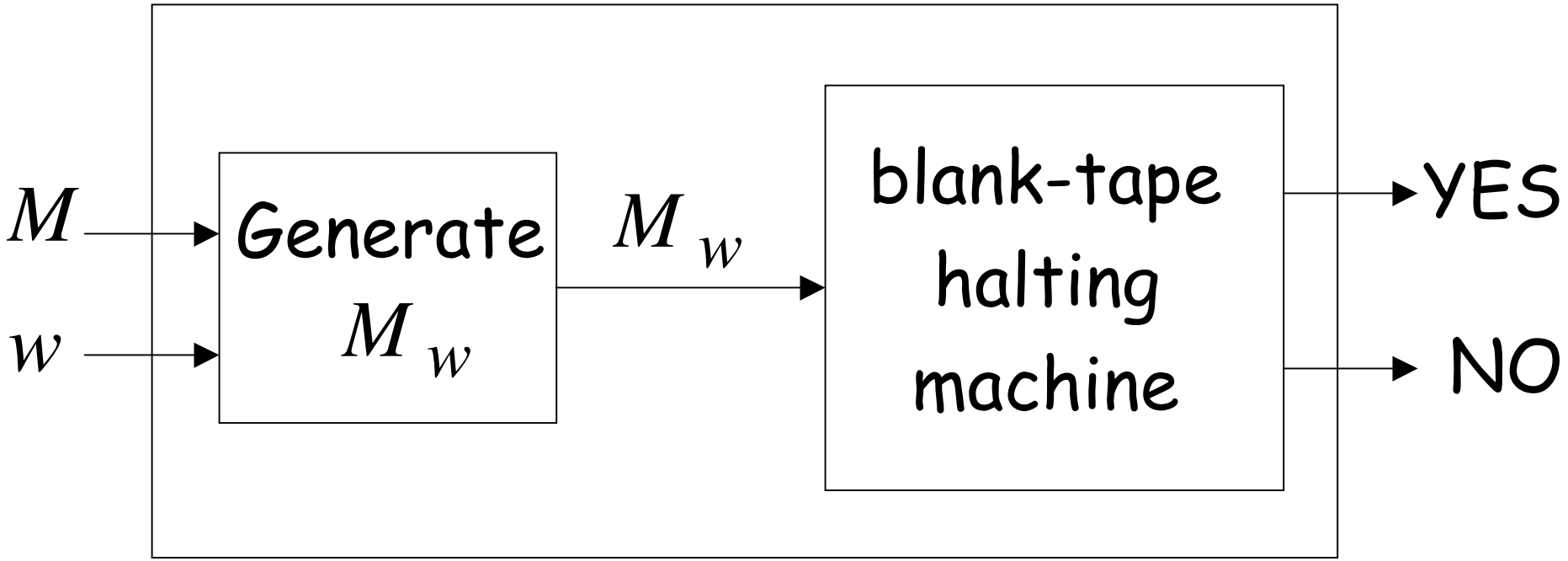
Algorithm for halting problem:

Inputs: M and w

1. Construct M_w

2. Run algorithm for blank-tape halting problem with input M_w

Halting problem machine



Since the halting problem is undecidable,

the blank-tape halting problem is
also undecidable

END OF PROOF