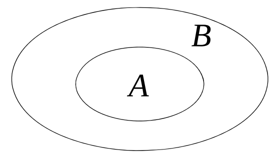
Reducibility

class 20

1

Problem A is reduced to problem B

If we can solve problem B then we can solve problem A



2

Problem A is reduced to problem B

If B is decidable then A is decidable

If A is undecidable then B is undecidable

3

Example: the halting problem

is reduced to

the state-entry problem

4

The state-entry problem

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

Question: Does M enter state q on input w ?

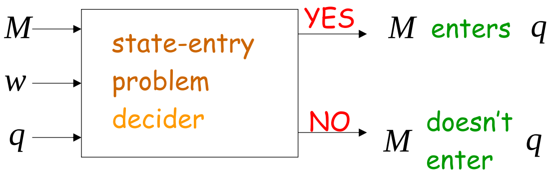
5

Theorem:
The state-entry problem is undecidable

Proof: Reduce the halting problem to the state-entry problem

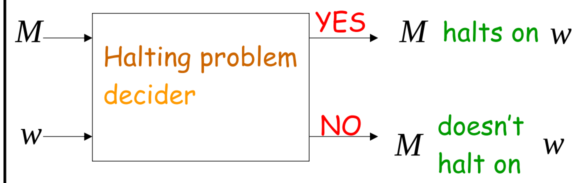
6

Suppose we have a Decider
for the state-entry algorithm:



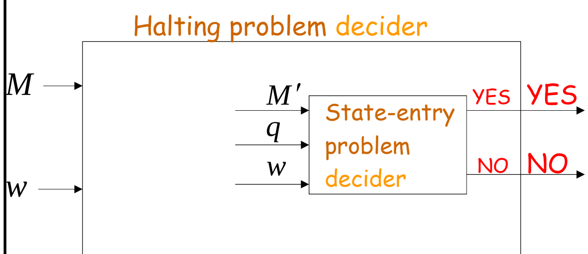
7

We want to build a decider
for the halting problem:



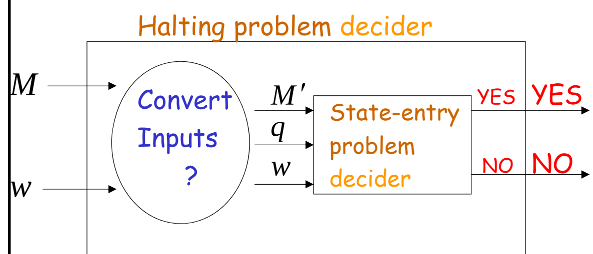
8

We want to reduce the halting problem to
the state-entry problem:



9

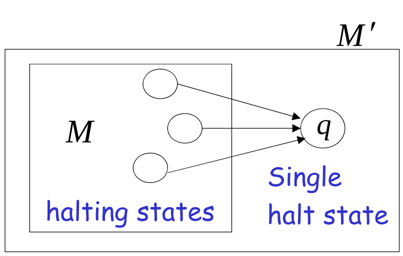
We need to convert one problem instance
to the other problem instance



10

Convert M to M' :

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



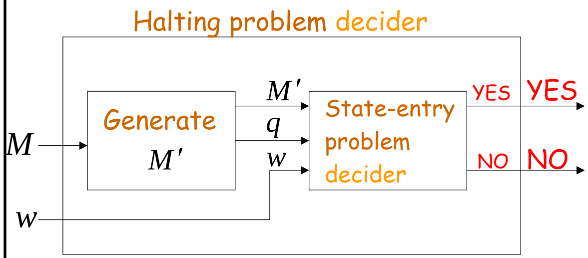
11

M halts on input w

if and only if

M' halts on state q on input w

12



13

We reduced the halting problem to the state-entry problem

Since the halting problem is undecidable, the state-entry problem is undecidable

END OF PROOF

14

Another example:

the halting problem

is reduced to

the blank-tape halting problem

15

The blank-tape halting problem

Input: Turing Machine M

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

16

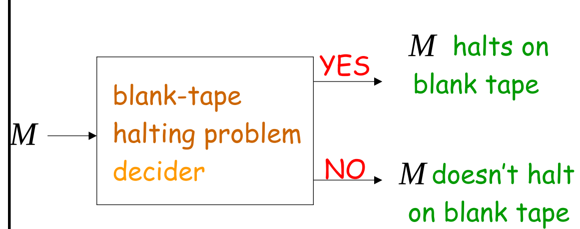
Theorem:

The blank-tape halting problem is undecidable

Proof: Reduce the halting problem to the blank-tape halting problem

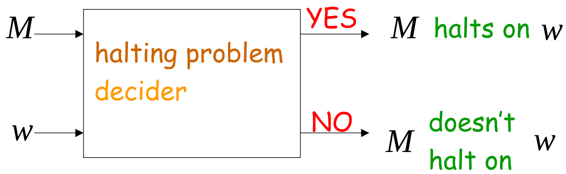
17

Suppose we have a decider for the blank-tape halting problem:

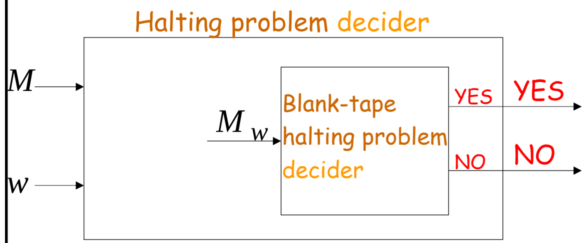


18

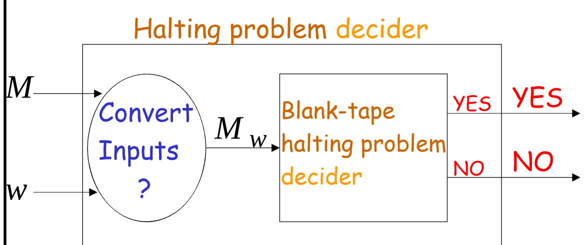
We want to build a decider
for the halting problem:



We want to reduce the halting problem to
the blank-tape halting problem:

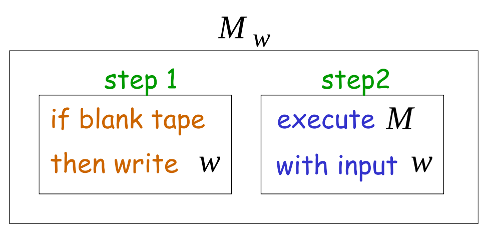


We need to convert one problem instance
to the other problem instance



Construct a new machine M_w

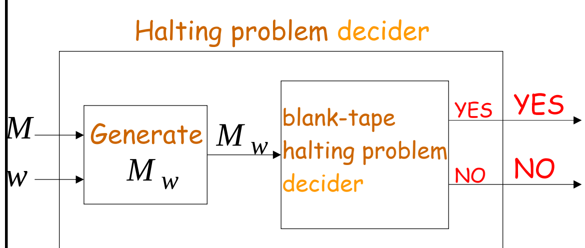
- When started on blank tape, writes w
- Then continues execution like M



M halts on input string w

if and only if

M_w halts when started with blank tape



We reduced the halting problem to the blank-tape halting problem

Since the halting problem is undecidable, the blank-tape halting problem is undecidable

END OF PROOF

Summary of Undecidable Problems

Halting Problem:

Does machine M halt on input w ?

Membership problem:

Does machine M accept string w ?

Blank-tape halting problem:

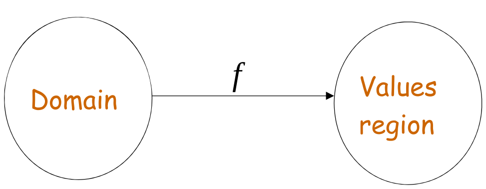
Does machine M halt when starting on blank tape?

State-entry Problem:

Does machine M enter state q on input w ?

Uncomputable Functions

Uncomputable Functions



A function is uncomputable if it cannot be computed for all of its domain

An uncomputable function:

$$f(n) = \begin{cases} \text{maximum number of moves until} \\ \text{any Turing machine with } n \text{ states} \\ \text{halts when started with the blank tape} \end{cases}$$

Theorem: Function $f(n)$ is uncomputable

Proof: Assume for contradiction that $f(n)$ is computable

Then the blank-tape halting problem is decidable

31

Decider for blank-tape halting problem:

Input: machine M

1. Count states of M: m
2. Compute $f(m)$
3. Simulate M for $f(m)$ steps starting with empty tape

If M halts then return YES
otherwise return NO

32

Therefore, the blank-tape halting problem is decidable

However, the blank-tape halting problem is undecidable

Contradiction!!!

33

Therefore, function $f(n)$ in uncomputable

END OF PROOF

34

Undecidable Problems
for
Recursively Enumerable Languages

35

Take a recursively enumerable language L

Decision problems:

- L is empty?
- L is finite?
- L contains two different strings of the same length?

All these problems are undecidable

36

Theorem:

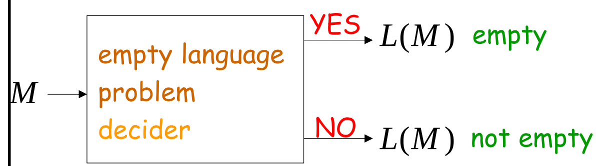
For any recursively enumerable language L it is undecidable to determine whether L is empty

Proof:

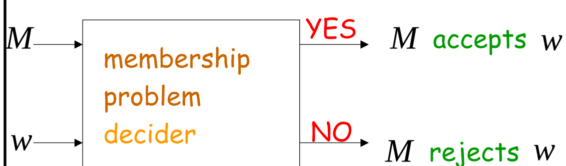
We will reduce the membership problem to this problem

Let M be the TM with $L(M) = L$

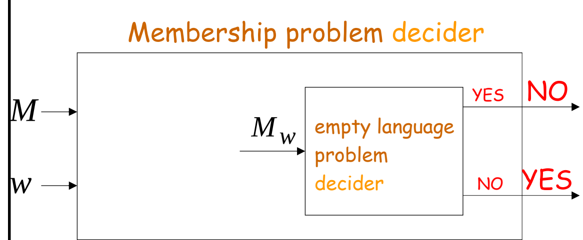
Suppose we have a decider for the empty language problem:



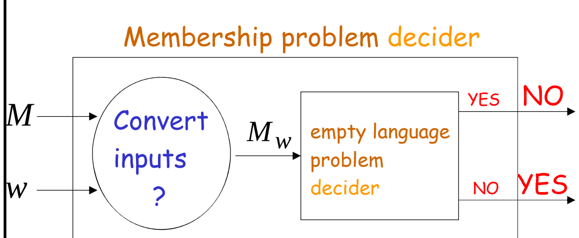
We will build the decider for the membership problem:



We want to reduce the membership problem to the empty language problem:



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



Construct machine M_w:

On arbitrary input string s

M_w executes the same as with M

When M enters a final state, compare s with w

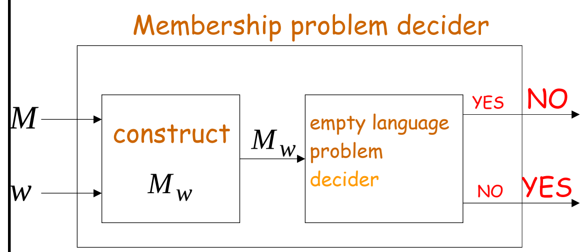
Accept only if $s = w$

$$w \in L$$

if and only if

$L(M_w)$ is not empty

$$L(M_w) = \{w\}$$



END OF PROOF