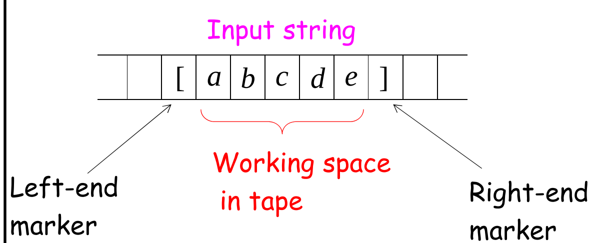
Linear Bounded Automata
LBAs

class 19

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

$u \to v$

String of variables and terminals

String of variables and terminals

7

Example unrestricted grammar:

$$S \to aBc$$
$$aB \to cA$$
$$Ac \to d$$

8

Theorem:

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

9

Context-Sensitive Grammars:

Productions

$u \to v$

String of variables and terminals

String of variables and terminals

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

10

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

is context-sensitive:

$$S \to abc \mid aAbc$$
$$Ab \to bA$$
$$Ac \to Bbcc$$
$$bB \to Bb$$
$$aB \to aa \mid aaA$$

11

Theorem:

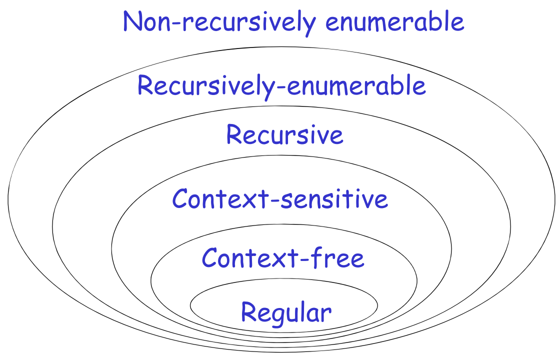
A language L is context sensitive if and only if L is accepted by a Linear-Bounded automaton

Observation:

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

12

The Chomsky Hierarchy



Decidability

Consider problems with answer YES or NO

Examples:

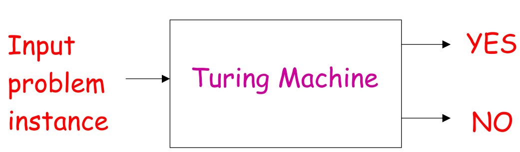
- Does Machine M have three states ?
- Is string w a binary number?
- Does DFA M accept any input?

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

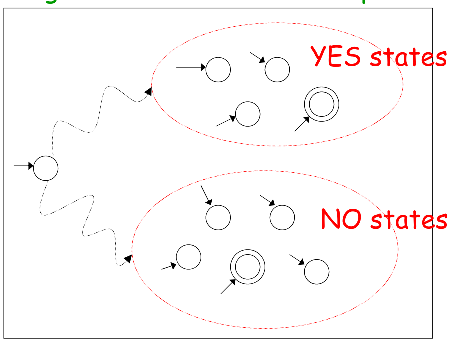


The machine that decides (solves) a problem:

- 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

19

Difference between Recursive Languages and Decidable problems

For decidable problems:

The YES states may not be final states

20

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

21

The Membership Problem

Input:
- Turing Machine M
- String w

Question: Does M accept w ?

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

22

Theorem:

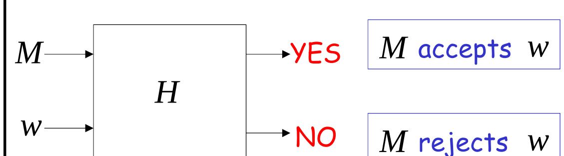
The membership problem is undecidable

(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

23

Thus, there exists a Turing Machine H that solves the membership problem



24

Let L be a recursively enumerable language

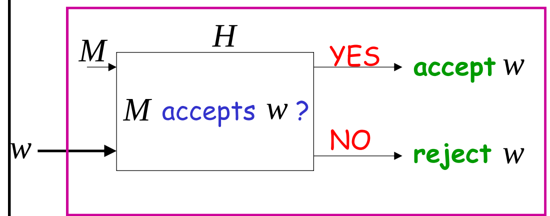
Let M be the Turing Machine that accepts L

We will prove that L is also recursive:

we will describe a Turing machine that accepts L and halts on any input

25

Turing Machine that accepts L and halts on any input



26

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!!!!

27

Therefore, the membership problem is undecidable

END OF PROOF

28

Another famous undecidable problem:

The halting problem

29

The Halting Problem

Input:
- Turing Machine M
- String w

Question: Does M halt on input w ?

30

Theorem:

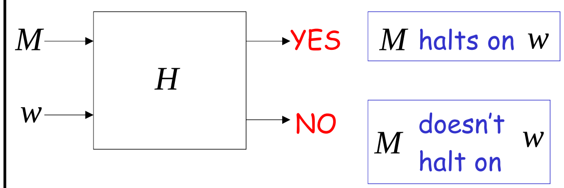
The halting problem is undecidable

(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

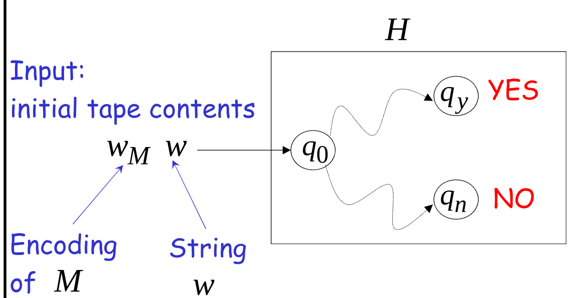
31

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



32

Construction of H



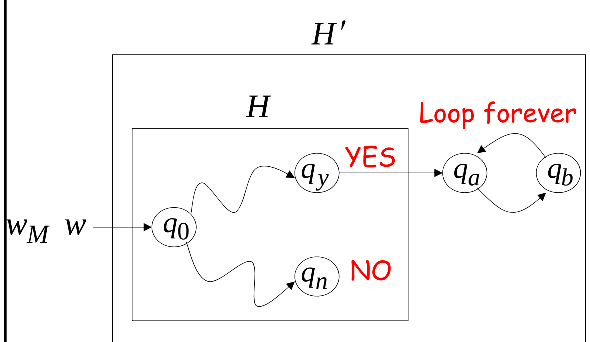
33

Construct machine H' :

If H returns YES then loop forever

If H returns NO then halt

34



35

Construct machine $\hat{H}$:

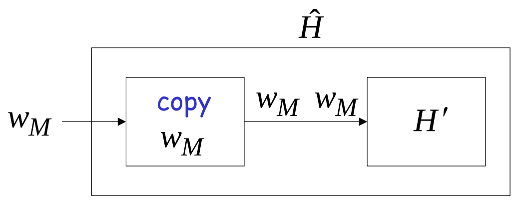
Input: w_M (machine M)

If M halts on input w_M

Then loop forever

Else halt

36



37

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

38

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

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

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

NONSENSE !!!!!

39

Therefore, we have contradiction

The halting problem is undecidable

END OF PROOF

40

Another proof of the same theorem:

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

41

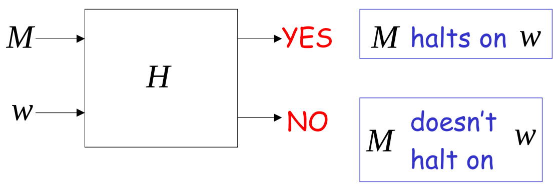
Theorem:

The halting problem is undecidable

Proof: Assume for contradiction that the halting problem is decidable

42

There exists Turing Machine H that solves the halting problem



43

Let L be a recursively enumerable language

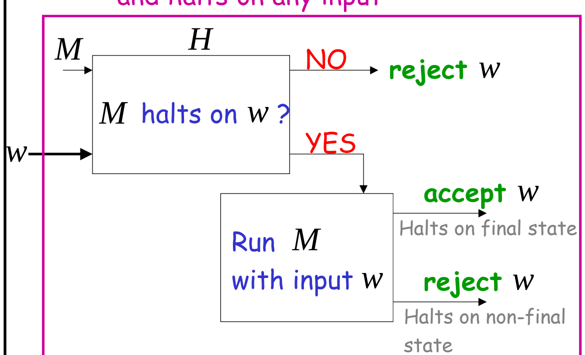
Let M be the Turing Machine that accepts L

We will prove that L is also recursive:

we will describe a Turing machine that accepts L and halts on any input

44

Turing Machine that accepts L and halts on any input



45

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!!!!

46

Therefore, the halting problem is undecidable

END OF PROOF

47