Department of Computer & Information Sciences
Pakistan Institute of Engineering and Applied Sciences

Linear Bounded Automata
LBAs

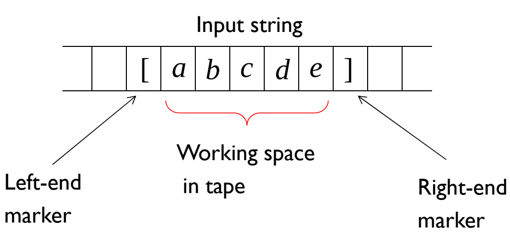
Based on slides by Costas Busch [http://csc.lsu.edu/~busch] Umar Faiz
http://www.pieas.edu.pk/umarfaiz/cis317

Linear bounded Automata

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

Linear Bounded Automaton (LBA)

We define LBA's as NonDeterministic

Open Problem:
NonDeterministic LBA's have same power with Deterministic LBA's ?

Linear Bounded Automaton (LBA)

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

Context-Sensitive Grammars

Unrestricted Grammars:

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

Context-Sensitive Grammars

Example unrestricted grammar:

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

Context-Sensitive Grammars

Theorem:

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

Context-Sensitive Grammars

Productions

$$u \rightarrow v$$

String of variables and terminals

String of variables and terminals

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

Context-Sensitive Grammars

The language $\{a^n b^n c^n\}$ is context-sensitive:

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

Context-Sensitive Grammars

Theorem:

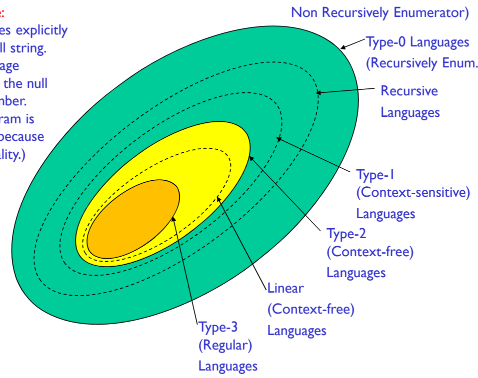
A language L is context sensitive if and only if is accepted by a Linear-Bounded automaton L

Observation:

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

The Chomsky Hierarchy

Technical Note:
Type-1 languages explicitly exclude the null string. All other language families permit the null string as a member. (The Venn diagram is in slight error because of this technicality.)



Decidability

Decidability

Consider problems with answer YES or NO

Examples:

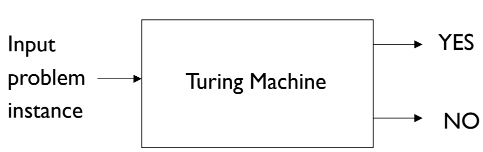
- Does Machine M have three states ?
- Is string w a binary number?
- Does DFA M accept any input?

Decidability

- A problem is decidable if some Turing machine decides (solves) the problem
- Examples:
  - Does Machine M have three states ?
  - Is string w a binary number?
  - Does DFA M accept any input?

Decidability

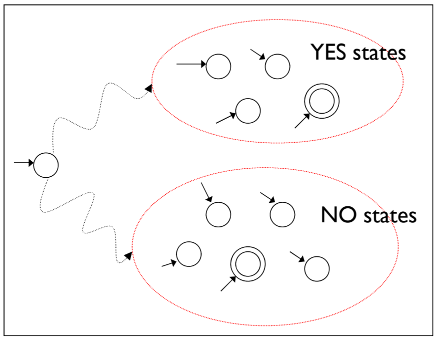
The Turing machine that decides (solves) a problem answers YES or NO for each instance of the problem



Decidability

- 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

Decidability

- Difference between Recursive Languages and Decidable problems
- For decidable problems:
  - The YES states may not be final states

Decidability

- Some problems are undecidable which means there is no Turing Machine that solves all instances of the problem

- Undecidable problems:
  - The membership problem
  - The Halting Problem

The Membership Problem

Input:

Turing Machine M

String w

Question:

Does M accept w ?

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

The Membership Problem

Theorem:

The membership problem is undecidable

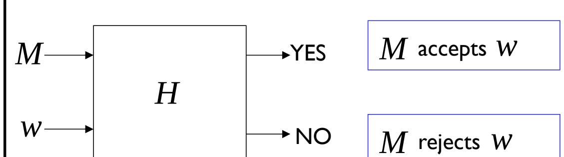
- (there are M and w for which we cannot decide whether $w \in L(M)$)

Proof:

Assume for contradiction that the membership problem is decidable

The Membership Problem

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



The Membership Problem

Let L be a recursively enumerable language

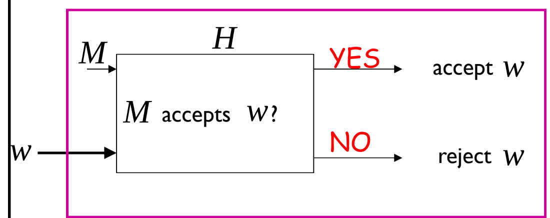
Let M be the Turing Machine that accepts L

We will prove that L is also recursive:

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

The Membership Problem

Turing Machine that accepts L and halts on any input



The Membership Problem

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!!!!

Therefore, the membership problem is undecidable

END OF PROOF

The Halting Problem

Input:

Turing Machine M

String w

Question:

Does M halt on input w ?

The Halting Problem

Theorem:

The halting problem is undecidable

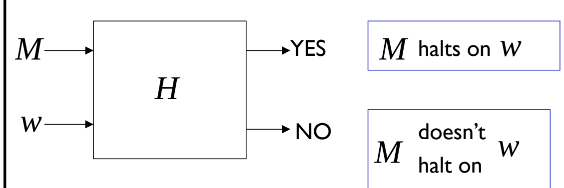
– (there are M and w for which we cannot decide whether M halts on input w)

Proof:

Assume for contradiction that the halting problem is decidable

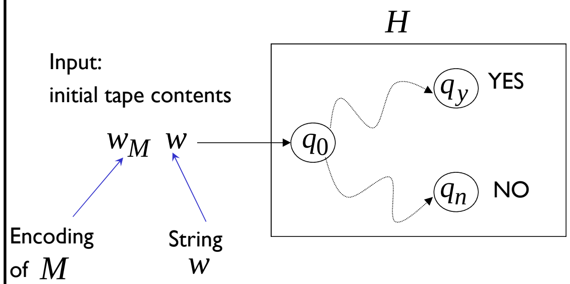
The Halting Problem

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

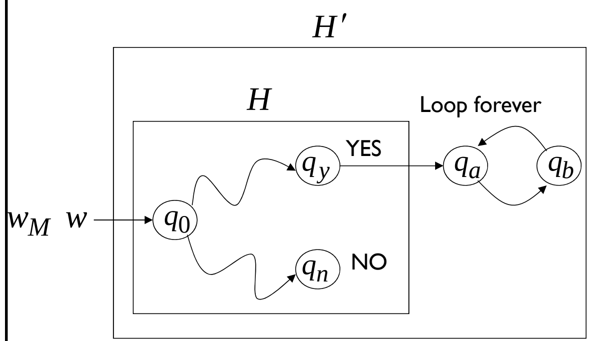


The Halting Problem

Construction of H:



The Halting Problem



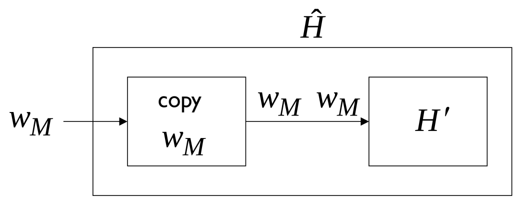
The Halting Problem

Construct machine H' :

If H returns YES then loop forever

If H returns NO then halt

The Halting Problem



The Halting Problem

Construct machine $\hat{H}$:

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

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

Then loop forever

Else halt

The Halting Problem

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

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

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

Which does not make any sense

We have contradiction, therefore, the halting problem is undecidable

END OF PROOF

The Halting Problem

Another proof of the same theorem:

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

The Halting Problem

Theorem:

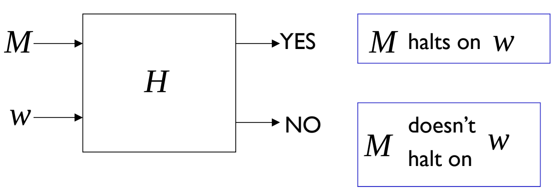
The halting problem is undecidable

Proof:

Assume for contradiction that the halting problem is decidable

The Halting Problem

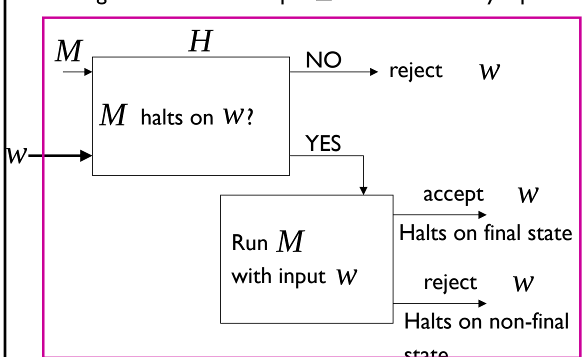
There exists Turing Machine H that solves the halting problem



The Halting 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



The Halting Problem

- 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. Hence we have a contradiction. Therefore, the halting problem is undecidable.