KTU NOTES

The learning companion.

KTU STUDY MATERIALS | SYLLABUS | LIVE NOTIFICATIONS | SOLVED QUESTION PAPERS

CONTEXT- SENSITIVE GRAMMAR

- A grammar G = (V, T, P, S) is said to be context sensitive , if all productions are of the form $\alpha \rightarrow \beta$ where $|\alpha| \leq |\beta|$ and β is not suppose to be empty.
- All rules in P are of the form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$
- The strings α_1 and α_2 may be empty, but β must be non-empty
- Context-sensitive grammars are more powerful than context-free grammars because there are some languages that can be described by CSG but not by context-free grammars
- The languages generated by these grammars are recognized by a linear bounded automaton.

$\alpha \rightarrow \beta$

where $\alpha, \beta \in (V \cup T)^+$ and $|\alpha| \leq |\beta|$

❖Context-sensitive Language

➢The language that can be defined by context-sensitive grammar is called CSL.

❖Properties of CSL

- Union, intersection and concatenation of two context-sensitive languages is context-sensitive.
- Complement of a context-sensitive language is context-sensitive

Example

$AB \rightarrow AbBc$

$A \rightarrow bcA$

$B \rightarrow b$

Consider the following CSG. What is the language generated by this grammar?

S → abc/aAbc

Ab → bA

Ac → Bbcc

bB → Bb

aB → aa/aaA

S → aAbc
→ abAc
→ abBbcc
→ aBbbcc
→ aaAbbcc
→ aabAbcc
→ aabbAcc
→ aabbBbccc
→ aabBbbccc
→ aaBbbbccc
→ aaabbbccc

The language generated by this grammar is $\{a^n b^n c^n \mid n \geq 1\}$

1. The language {anbncn, n > 1} is context-sensitive but not context free.
A grammar for this language is given by:
S → aSBC | aBC,
CB → HB,
HB → HC,
HC → BC,
aB → ab,
bB → bb,
bC → bc,
cC → cc

aabbcc

2.

UNRESTRICTED GRAMMAR

- A grammar G = (V, T, P, S) is called unrestricted , if all the productions are of the form $\alpha \rightarrow \beta$, where α is a string of terminals and nonterminals with at least one non-terminal and α cannot be null. β is a string of terminals and non-terminals.
- They generate the languages that are recognized by a **Turing machine**.

Example

$S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

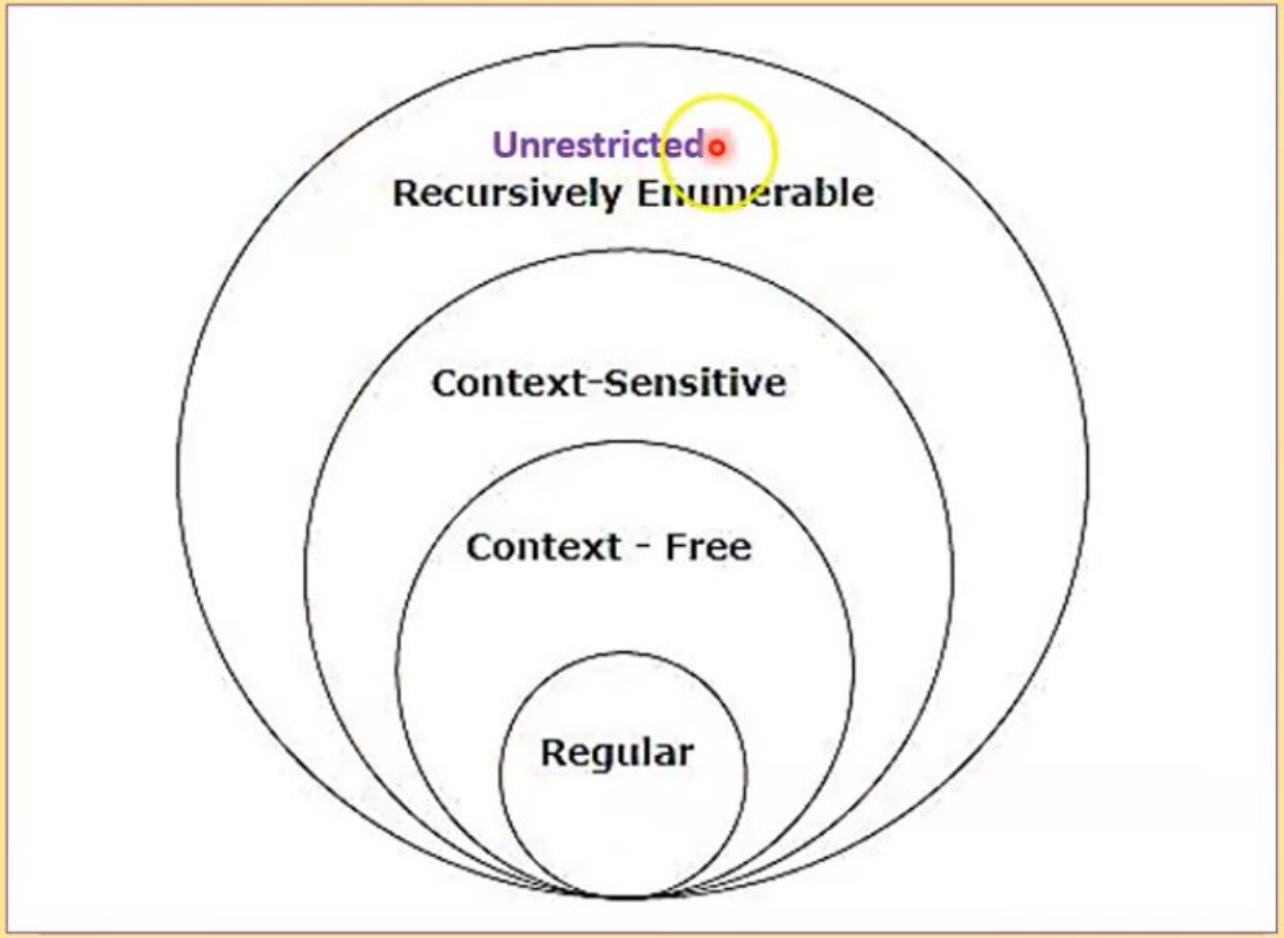
$aD \rightarrow Db$

CHOMSKY HIERARCHY

- According to **Noam Chomsky**, there are four types of grammars – Type 0, Type 1, Type 2, and Type 3. The following table shows how they differ from each other

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown automaton
Type 3	Regular grammar	Regular language	Finite state automaton

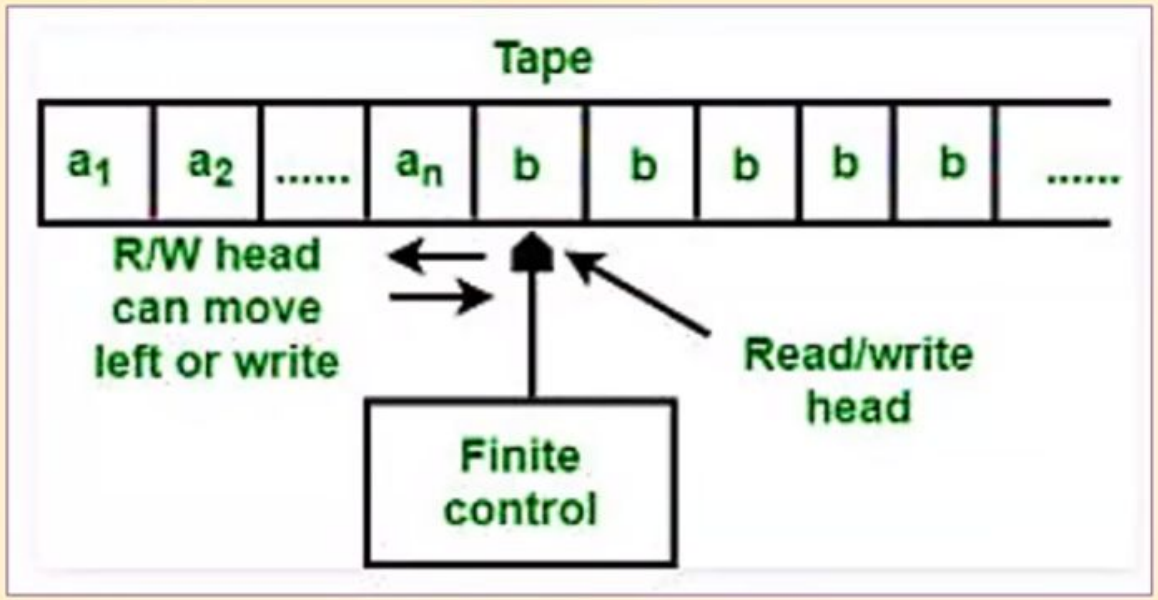
The following illustration shows the scope of each type of grammar



TURING MACHINE (TM)

- Turing machine was invented in 1936 by Alan Turing.
- It is an accepting device which accepts Recursive Enumerable Language generated by type 0 grammar.
- A Turing Machine consists of a tape of infinite length on which read and writes operation can be performed.
- The tape consists of infinite cells on which each cell either contains input symbol or a special symbol called blank.
- It also consists of a head pointer which points to cell currently being read and it can move in both directions.

Turing Machine



Tape alphabets: $\Sigma = \{0,1,a,b,x,z_o\}$

Blank symbol (␣) is a special symbol which is not in Σ

❖Operations on the Tape

- Read / Scan the symbol which is pointed by the tape head
- Update / Write a symbol which is pointed by the tape head
- Move the tape head one step left
- Move the tape head one step right

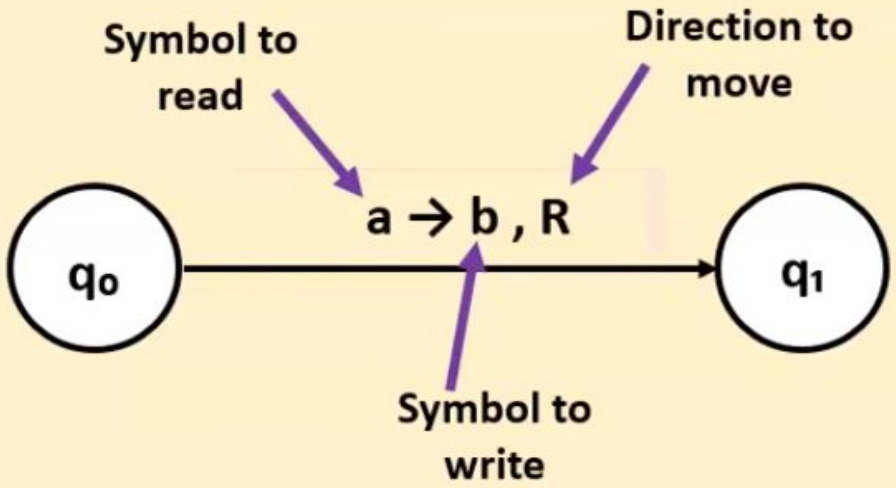
❖Features of the Turing machine:

- It has an external memory which remembers arbitrary long sequence of input.
- It has unlimited memory capability.
- The model has a facility by which the input at left or right on the tape can be read easily.
- The machine can produce a certain output based on its input. Sometimes it may be required that the same input has to be used to generate the output. So in this machine, the distinction between input and output has been removed. Thus a common set of alphabets can be used for the Turing machine.

❖Operation Rule – 1

At each step of computation

- ➢Read the current symbol
- ➢Write (update) the same cell
- ➢Move exactly one cell either LEFT or RIGHT



Transition Diagram

❖Operation Rule – 2

- Turing Machine has a control portion , which controls the Turing Machine and is similar to FSM or PDA. It is deterministic also
- TM has an initial state
- TM has two final states
  - Accept state
  - Reject state

When the computation Halt (stop) ?

- HALT & Accept
- HALT & Reject
- Loop (Keep on computation and fail to stop)

❖Formal definition of Turing machine

A Turing machine can be defined as a collection of 7 components:

- **Q** : the finite set of states
- **∑** : the finite set of input symbols
- **T** : the tape symbol
- **q_o** : the initial state
- **F** : a set of final states (Accept & Reject state)
- **B** : a blank symbol used as a end marker for input
- **δ** : a transition or mapping function.

❖Transition Function (δ)

$$Q \times \Sigma \rightarrow T \times (R/L) \times Q$$

❖Production rule of TM

$$\delta (q_o, a) \rightarrow (q_1 , y, R)$$

- Any computations that can be done on existing digital computer, can also be done by Turing Machine
- If we are having an algorithm of a problem, then that can also be solved by Turing Machine.

❖ Turing Thesis

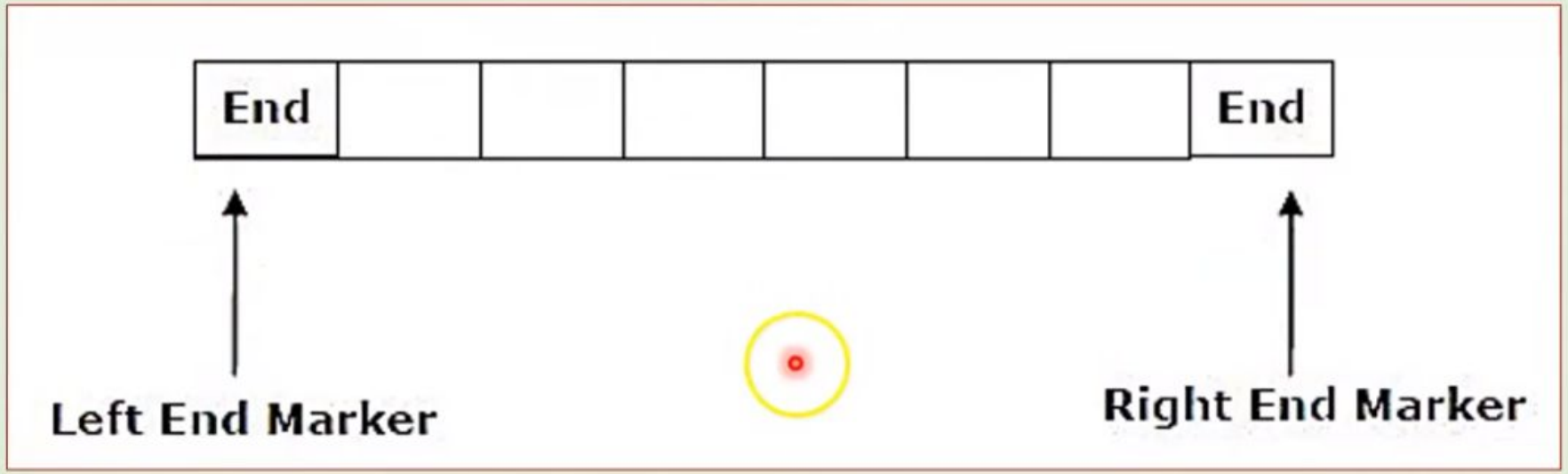
It states that, any computation that can be carried out by mechanical means can be performed by some Turing Machine.

➢ The language that acceptable by the Turing Machine is called Recursively Enumerable Language.

LINEAR BOUNDED AUTOMATA (LBA)

- It behaves as a Turing machine but the storage space of tape is restricted only to the length of the input string.
- It is less powerful than a Turing machine but more powerful than push down automata.
- The computation is restricted to the constant bounded area.
- The input alphabet contains two special symbols which serve as left end markers and right end markers which mean the transitions neither move to the left of the left end marker nor to the right of the right end marker of the tape.

- A deterministic linear bounded automaton is always context-sensitive and the linear bounded automaton with empty language is undecidable.



➢A linear bounded automaton can be defined as an 8-tuple

$(Q, T, \sum, q_o, ML, MR, \delta, F)$ where

Q is a finite set of states

T is the tape alphabet

∑ is the input alphabet

q_o is the initial state

ML is the left end marker

MR is the right end marker where **MR ≠ ML**

δ is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, Constant 'c') where c can be 0 or +1 or -1

F is the set of final states

❖Linear Bounded Automata (LBA) – Applications

- For implementation of genetic programming.
- For constructing syntactic parse trees for semantic analysis of the compiler.

❖Expressive Power of various Automata

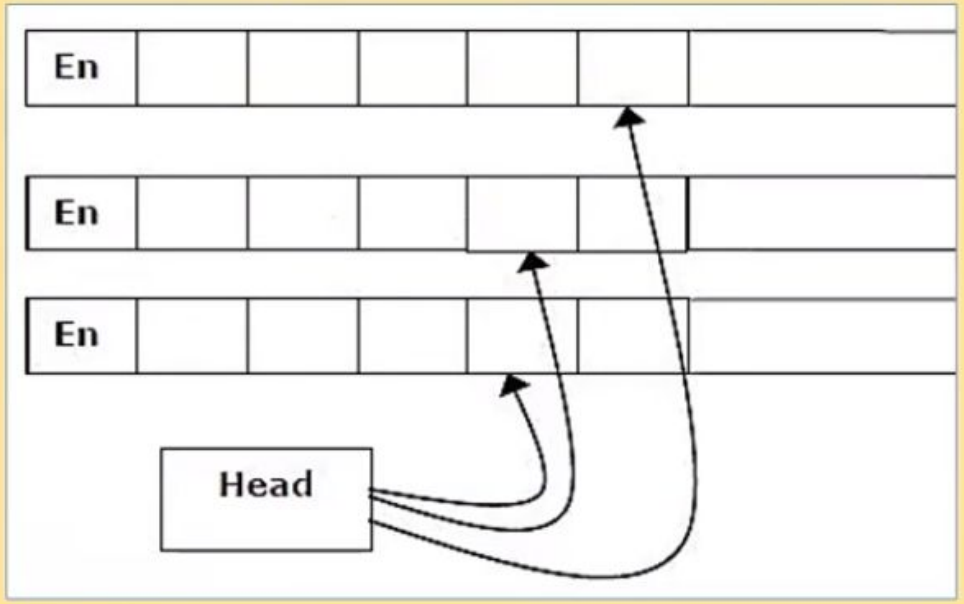
- The Expressive Power of any machine can be determined from the class or set of Languages accepted by that particular type of Machine. Here is the increasing sequence of expressive power of machines :

FA < DPDA < PDA < LBA < TM

MULTI – TAPE TURING MACHINE

- Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head.
- Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank.
- At first, the first tape is occupied by the input and the other tapes are kept blank.
- Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.

A Multi-tape Turing machine can be formally described as a 6-tuple $(Q, T, B, \delta, q_0, F)$ where



Q is a finite set of states

T is the tape alphabet

B is the blank symbol

q_0 is the initial state

F is the set of final states

δ is a relation on states and symbols where

$$\delta: Q \times T^n \rightarrow Q \times T^n \times \{L, R\}^n$$ where n is the number of tapes

➢Every Multi-tape Turing machine has an equivalent single-tape Turing machine.

Example

n = 2

$$\delta(q_0, a, e) = (q_1, x, y, L, R)$$

NON-DETERMINISTIC TURING MACHINE

- Only the difference is in transition function

➢ **δ of Deterministic TM**

$$Q \times \Sigma \rightarrow T \times (R/L) \times Q$$

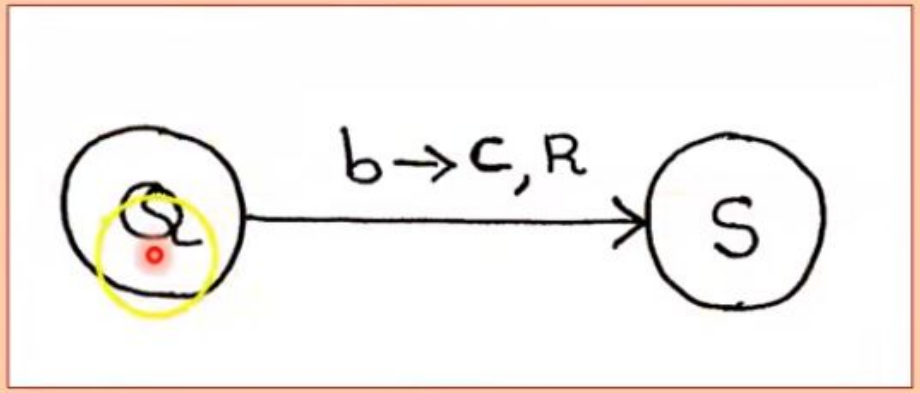
➢ **δ of Deterministic TM**

$$Q \times \Sigma \rightarrow P\{T \times (R/L) \times Q\}$$ where **P** = power set

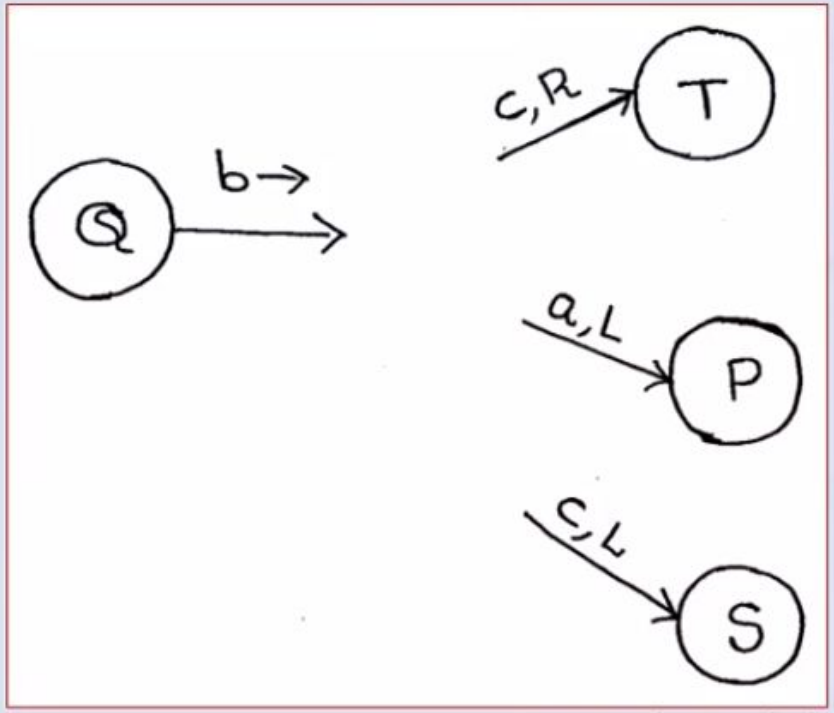
- In a particular state, up on reading a particular input symbol, the Non Deterministic TM will go to more than one state.

Deterministic & Non – Deterministic TM

Deterministic TM



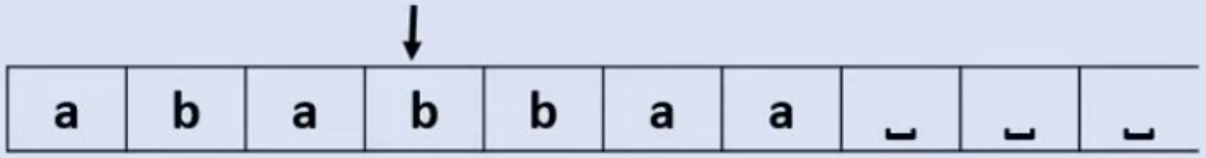
Non-Deterministic TM



❖Configuration of Deterministic and No-Deterministic TM

Configuration :

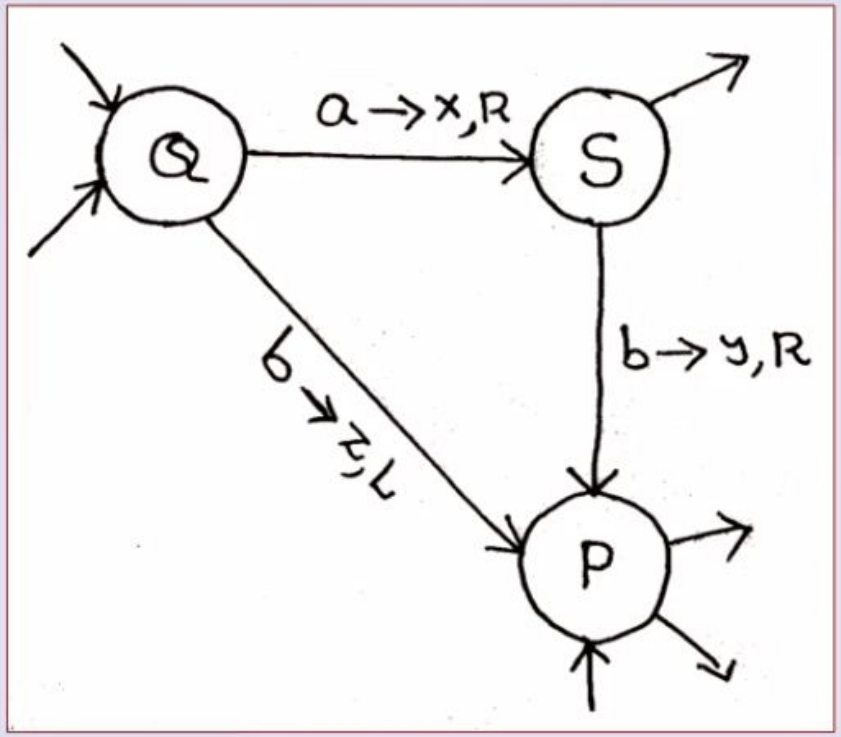
- It is a way to represent the entire state of a TM at a moment during computation
- It is a string which shows
  - The current state
  - The current position of the head
  - The entire tape contents



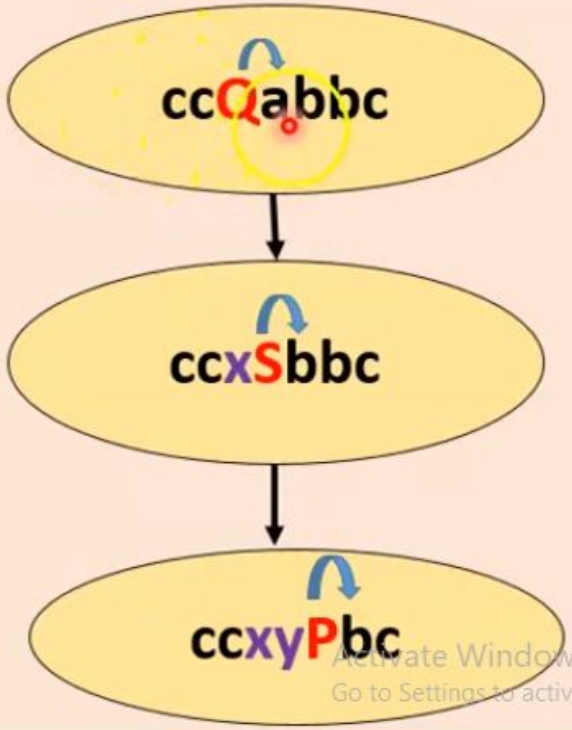
abaQbbaa

Deterministic TM and its computation history using Configuration

Deterministic TM

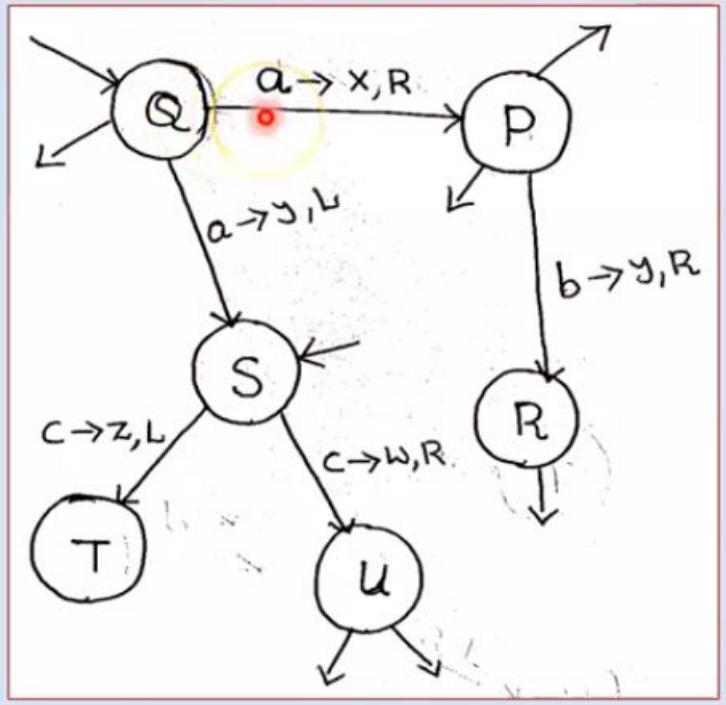


Configuration



Non-Deterministic TM and its computation history using Configuration

Non- Deterministic TM



Configuration