THEORY OF COMPUTATION
CODE:303105306

UNIT:4 TURING MACHINE

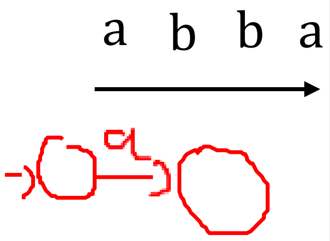
CHAPTER-4

Turing machines

Understanding of machines

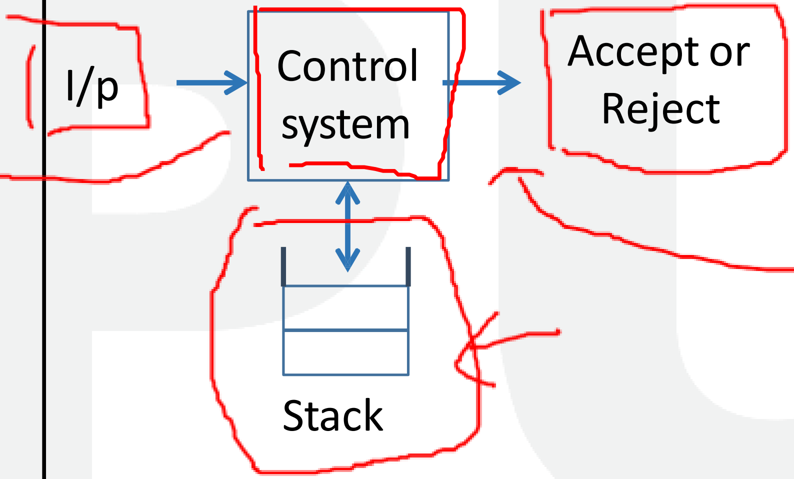
Finite Automata

- Input Strings



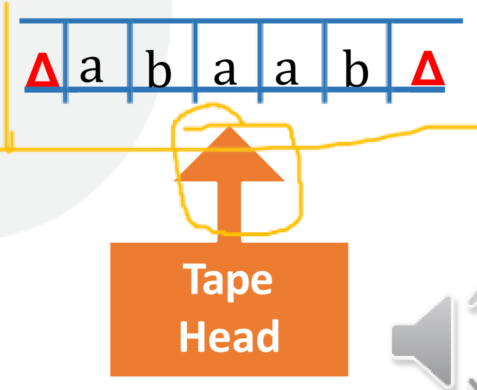
Pushdown Automata

- Input Strings & Stack

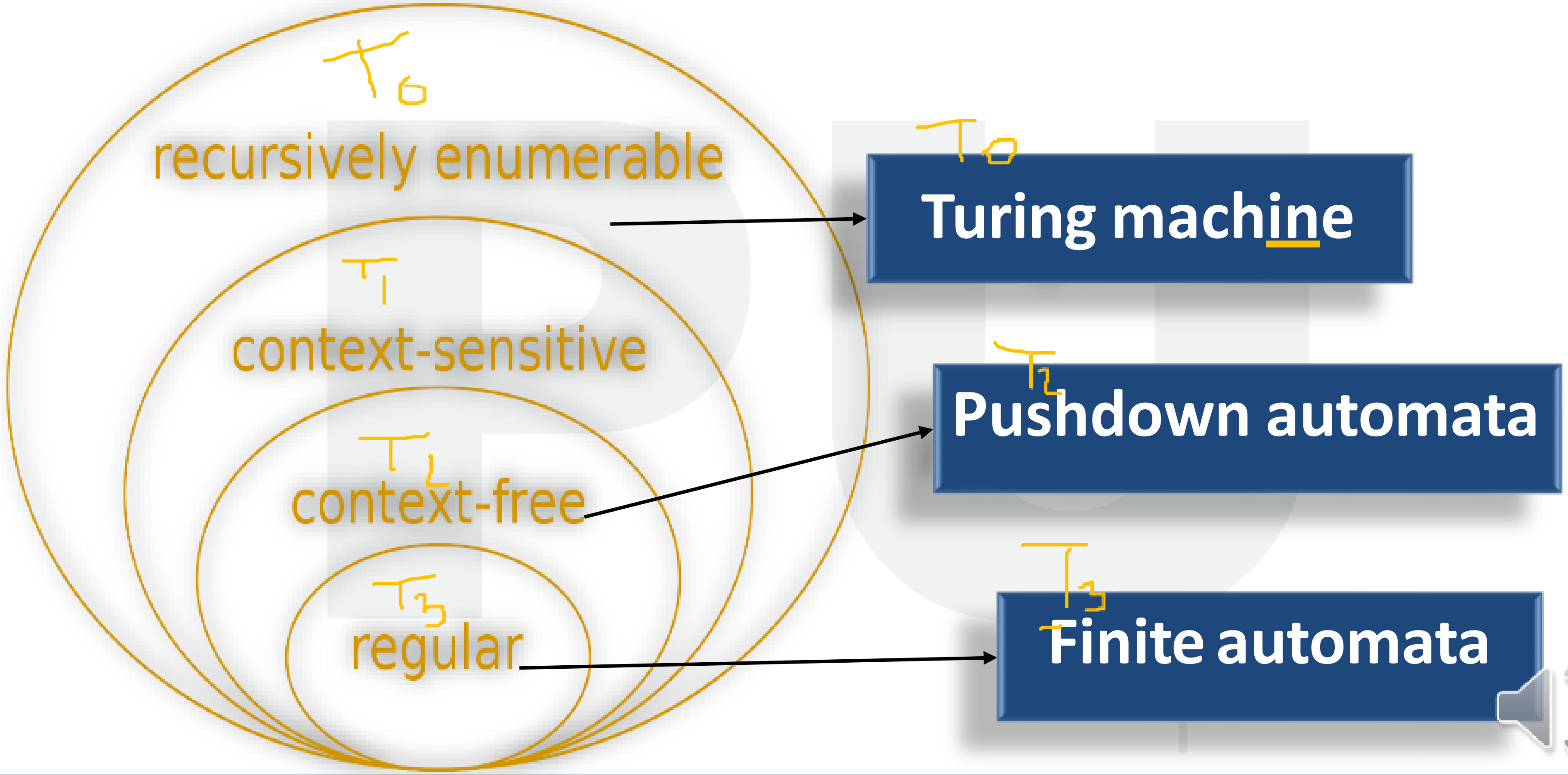


Turing Machine

- Input Strings
- Magnetic Tape
- Special symbol
- Control system

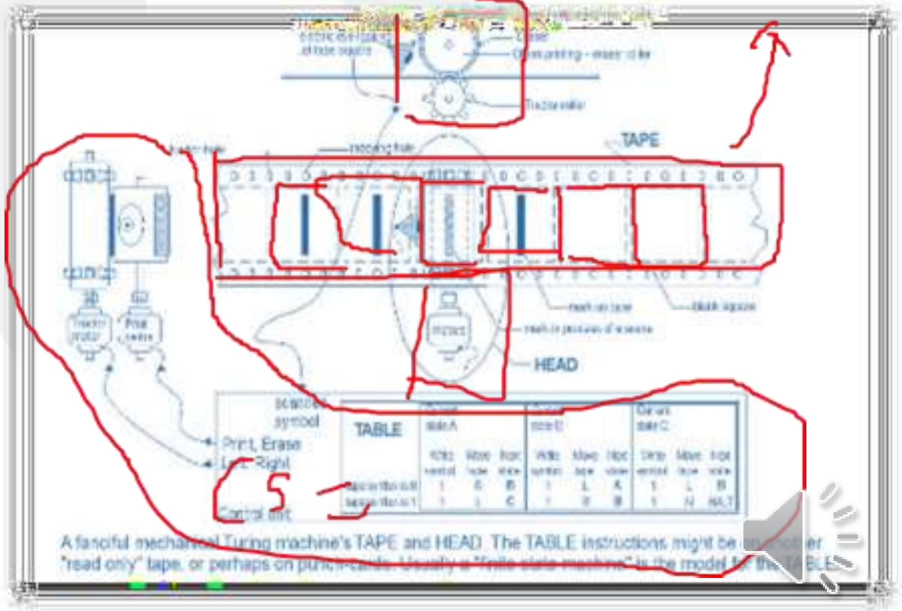


Chomsky-hierarchy



Turing Machine

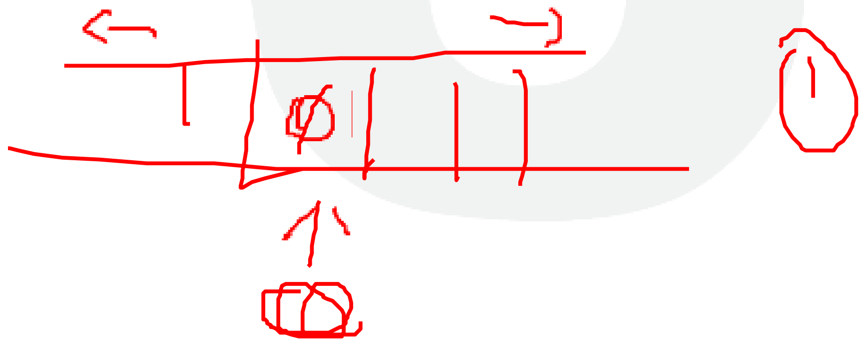
- A Turing Machine (TM) has finite-state control (like PDA), and an infinite read-write tape.
- The tape serves as both input and unbounded storage device.
- The tape is divided into cells, and each cell holds one symbol from the tape alphabet like 1 & 0.
- There is a special blank symbol B. At any instant, all but finitely many cells hold B.
- Tape head sees only one cell at any instant. The contents of this cell and the current state determine the next move of the TM.



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Contd.

- At any point of time, the head which is positioned over one of the squares on the tape. With this head, the machine can execute three very simple operations:
  - Read the symbol under the head on the square.
  - Edit the symbol by updating symbol or erasing it.
  - Shift the head to left or right by only one square, So that the machine can read and edit or erase the symbol of a neighboring square.

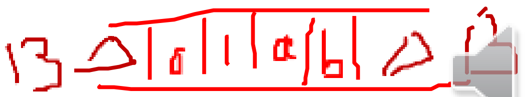


Definition of Turing machine

- Formally, a Turing machines a 7- tuple M= $\boxed{(Q,\Sigma,\Gamma, \delta, q0, B, F)}$ where
  - Q is the finite set of states of the finite control
  - Σ is the finite set of input symbols.
  - Γ is the finite set of tape symbols; Σ⊂Γ.
  - δ:Q × Γ → Q × Γ × {L, R} is the transition function, which is a partial function.
  - q ∈ Q is the start state.
  - B ∈ Γ is the blank symbol; B ∉ Σ.
  - F ⊆ Q is the set of final or accepting states.

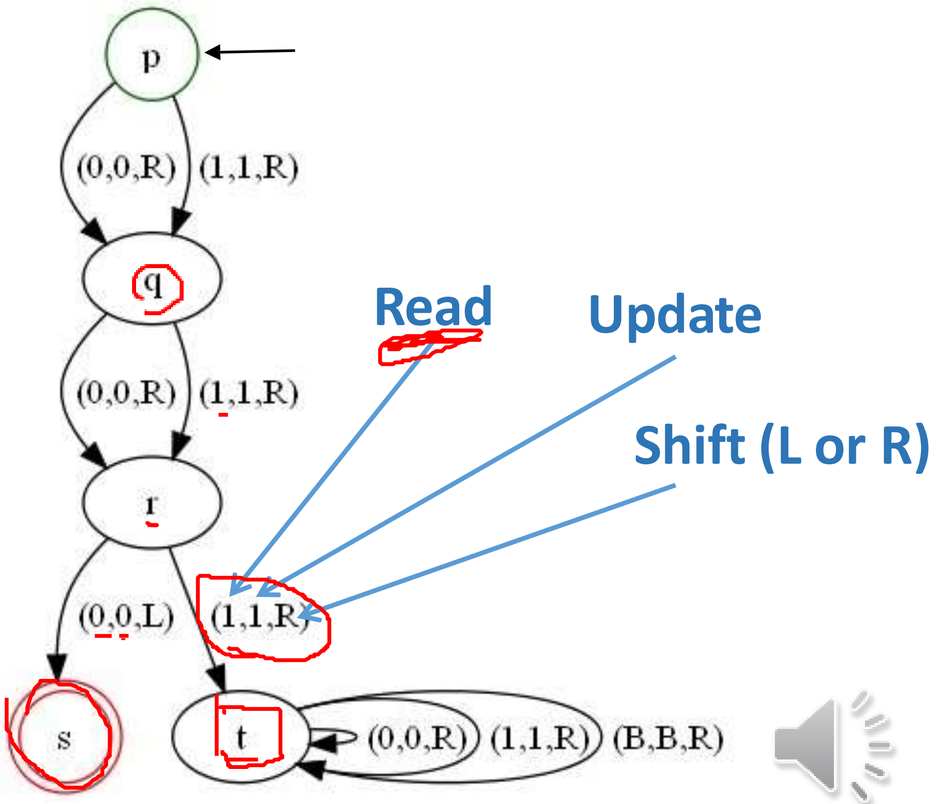
{S, P}

$$\begin{tikzcd} {} \arrow[r] & \boxed{S} \arrow[r] & \boxed{P} \end{tikzcd}$$



Examples

- TM that checks its third symbol is 0, accepts if so, and runs forever, if not (note it never rejects).
  - M=({p,q,r,s,t},{0,1,},{0,1,B},p,s,d)
    - δ(p,X) = (q,X,R) for X=0,1
      - δ(q,X) = (r, X , R) for X=0,1
      - δ(r,0) = (s,0,L)
      - δ(r,1) = (t,1,R)
      - δ(t,X) = (t,X,R) for X=0,1,B



Examples

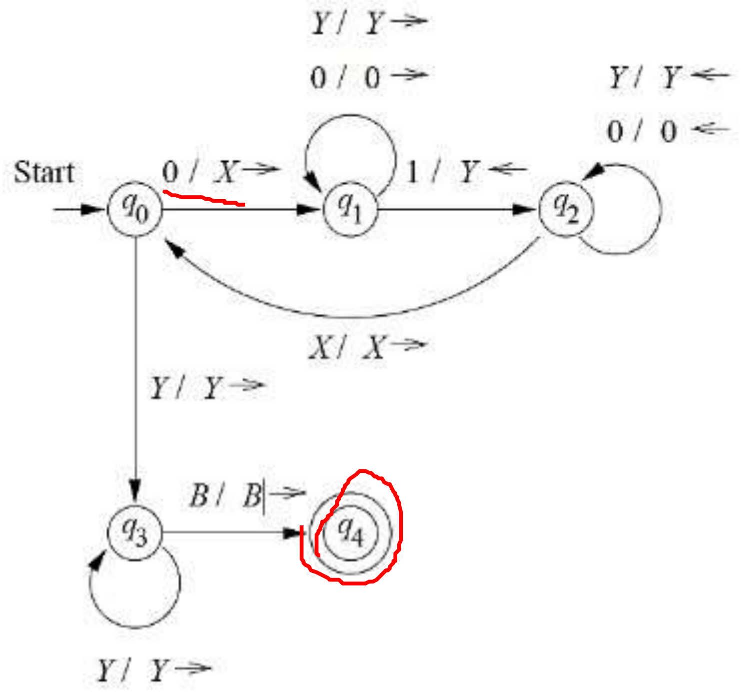
- TM for {0n1n : n >= 1}
  - M = ({q0, q1, q2, q3, q4}, {0, 1}, {0, 1,X, Y,B}, ±, q0,B, {q4})

$\{0^n 1^n / n \geq 1\}$

$\rightarrow$ R

$\leftarrow$ L

B	0	0	1	1	B
	X	X	Y	Y	



Examples

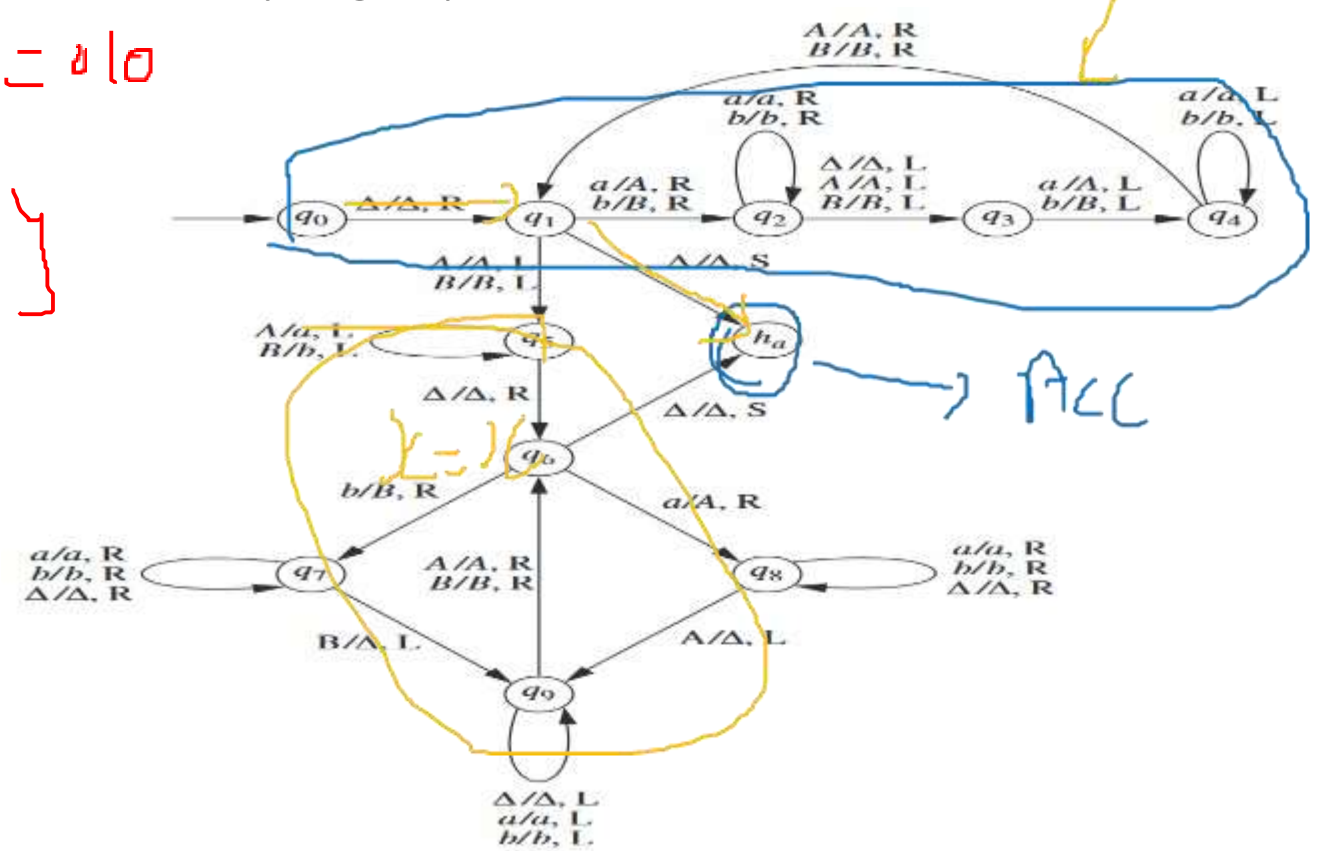
- A Turing machine accepting{xx|x∈{a, b}∗}

x x   x = 0 1 0

{ 0 1 0 | 0 1 0 }

x | x

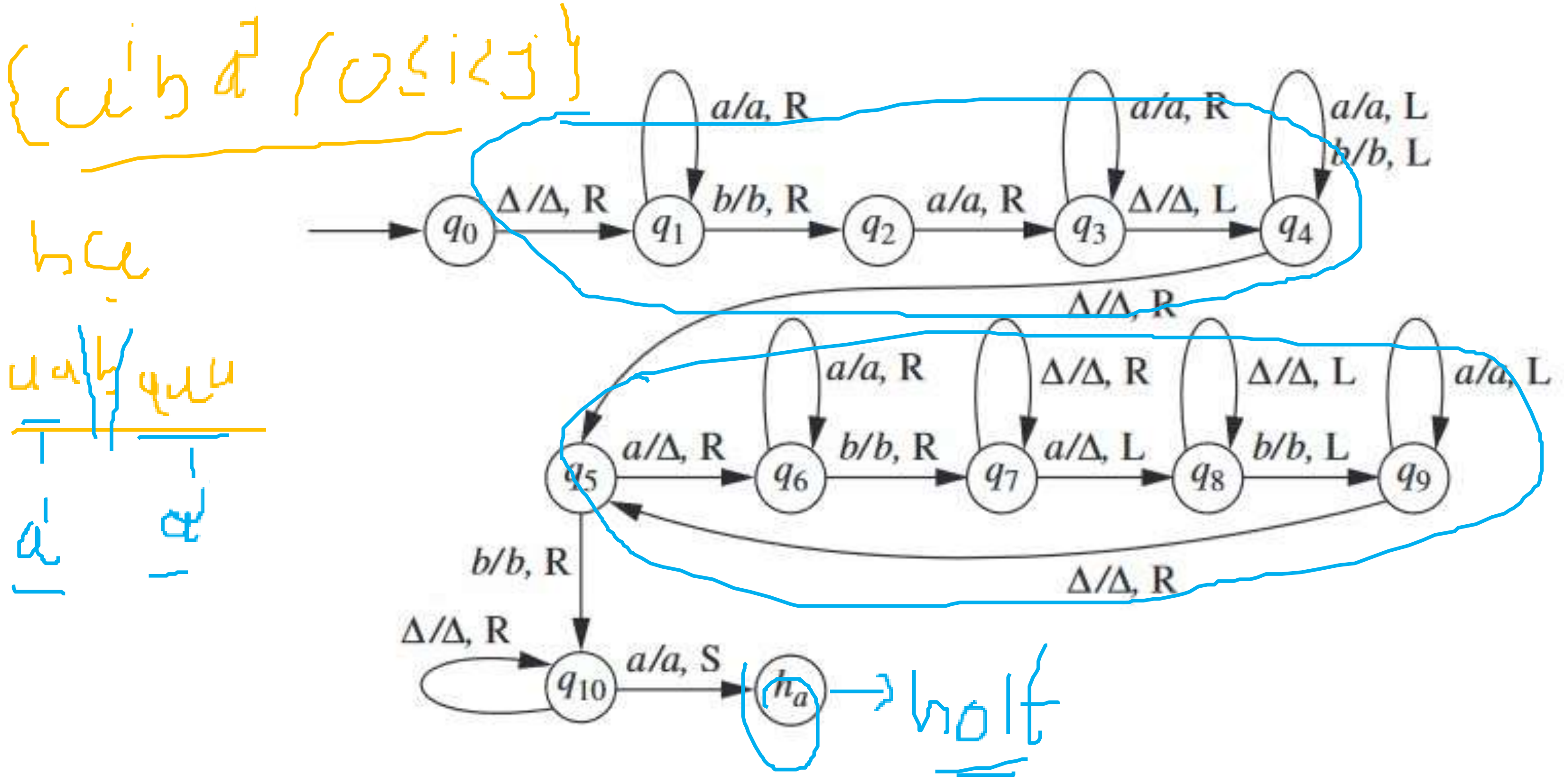
A B



Examples

- A Turing machine accepting{aibaj|0≤i<j}

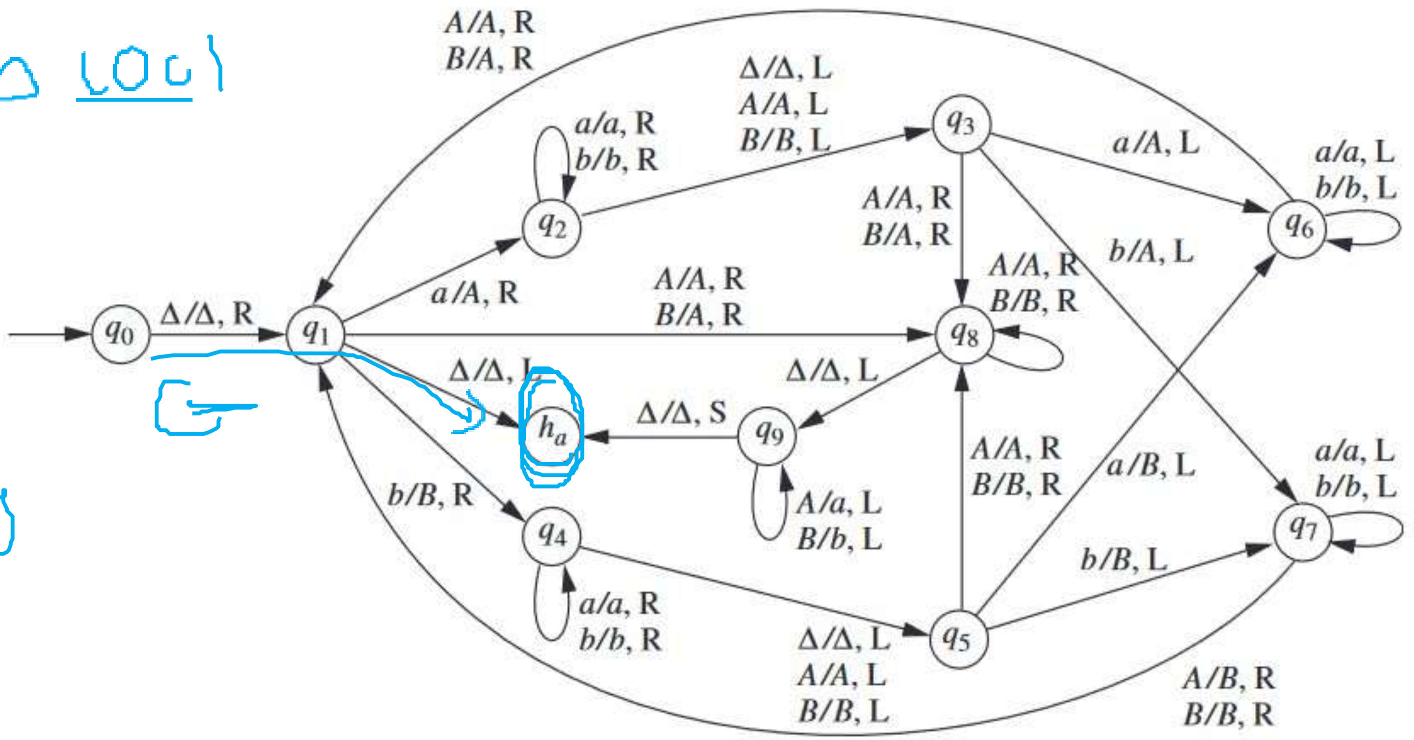
{a^i b a^j / 0≤i<j}



Examples

- A Turing machine computing the reverse function.

Δ a b a Δ b a b

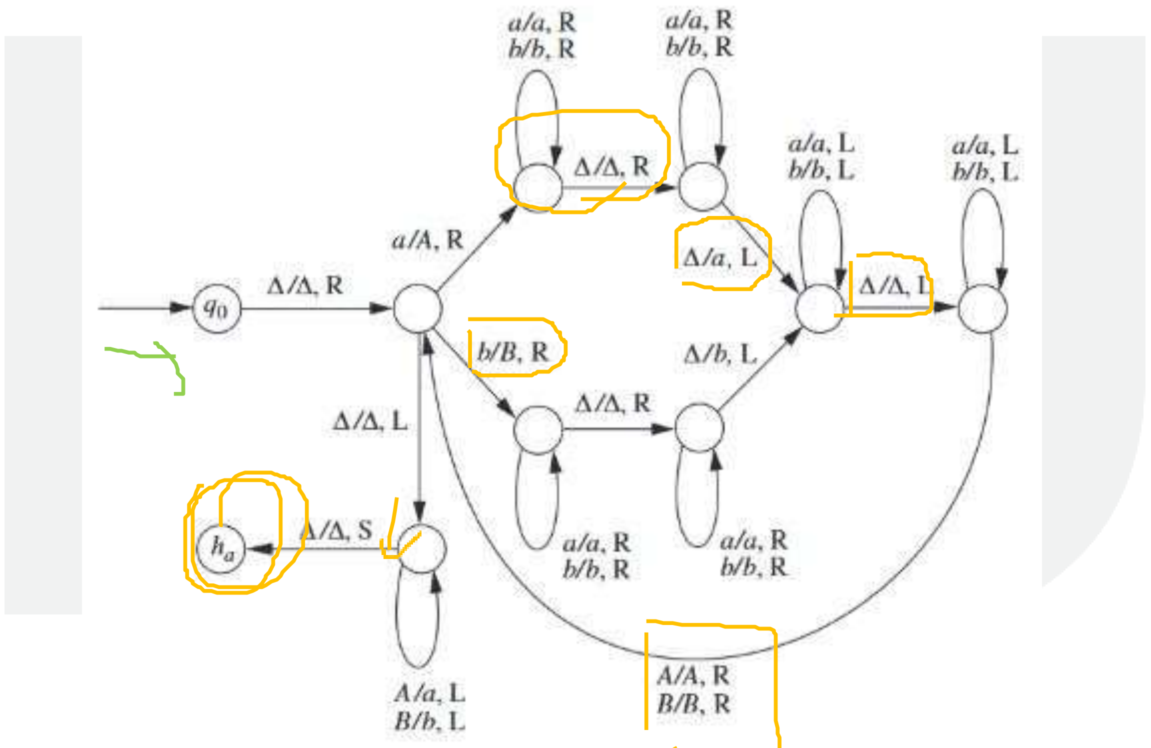


Parul® University

DIGITAL LEARNING CONTENT

Examples

•A Turing machine to copy strings.



Turing Recognizable and Decidable

- A Turing machine M recognizes language L if L=L(M). We say L is Turing-recognizable(or simply recognizable) if there is a TM M such that L=L(M).
- A Turing machine decides language L if L=L(M) and M halts on all inputs. We say L is decidable if there is a TM M that decides L.
- Both types of machine halt in the Accept state on strings that are in the language.
  - A Decider also halts if the string is not in the language.
  - A Recognizer MAY or MAY NOT halt on strings that are not in the language.
- On all input:
  - A Decider MUST halt (in Accept or Reject state)
  - A Recognizer MAY or MAY NOT halt on some strings

L_1, L_2
/De

$L_1 \cup L_2 = L_3$

Re /De

/Re

Closure properties

-Union

Both decidable and Turing recognizable languages are closed under union.

•For decidable languages the proof is easy. Suppose L1 and L2 are two decidable languages accepted by halting TMs M1 and M2 respectively. The machine for L1 U L2 is designed as follows:

•Given an input x, simulate M1 on x. If M1 accepts then accept, else simulate M2 on x. If M2 accepts then accept else reject.

•Now suppose L1 and L2 are two Turing recognizable languages accepted by TMs M1and M2 respectively. Since L1 and L2 are Turing recognizable languages, there fore for strings that do not belong to these languages, the corresponding machines may not even halt. The previous strategy will not work because we can have a scenario where M2 accepts x but M1 loops forever.

$\overset{*}{L_1} = L_2 \qquad \overset{x}{L_2} - L_3$

De $\quad$ D+ $\quad$ Re

Contd.

- Concatenation
  - Both decidable and Turing recognizable languages are closed under concatenation.
  - I will give the proof for Turing recognizable languages. The proof for decidable languages is similar. Let L1and L2 be two Turing recognizable languages. Given an input w, use non determinism and guess a partition w (say w=xy). Now run the respective Turing machines of L1and L2 on x and y respectively. If both accepts then accept else reject.
- Star
  - Both decidable and Turing recognizable languages are closed under star operation.
  - This is also similar to concatenation. Non deterministically first guess a number k, and then guess a k partition of the given input. Now for each string in the partition, check whether it belongs to the original language.

$L_1 \cap L_2 = L_3$

De De

Re Re

Contd.

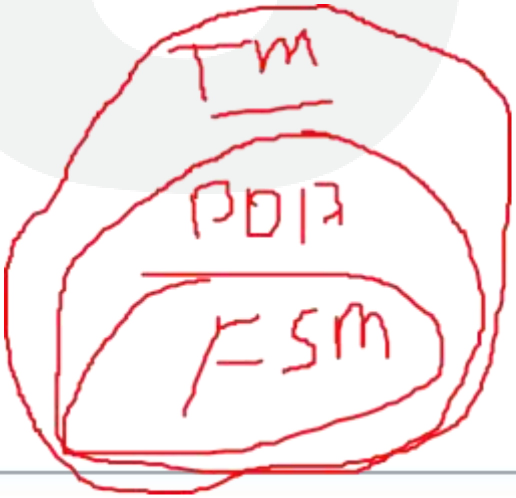
- Intersection
  - Both decidable and Turing recognizable languages are closed under intersection. Run the TMs of both the languages on the given input. Accept if and only if both the machines accept. In the case of intersection we can run the TMs of L1 and L2 one after the other (as opposed to union).
- Complementation
  - Decidable languages are closed under complementation. To design a machine for the complement of a language L, we can simulate the machine for L on an input. If it accepts then accept and vice versa.

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-4

Variants of Turing machines



Parul® University

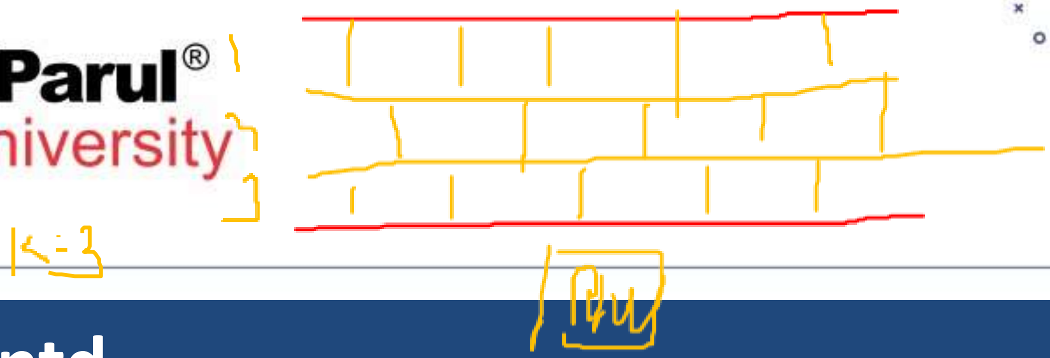
DIGITAL LEARNING CONTENT

Variants of Turing machines

1. Multiple track Turing Machine
2. Two-way infinite Tape Turing Machine
3. Multi-tape Turing Machine
4. Multi-tape Multi-head Turing Machine
5. Multi-dimensional Tape Turing Machine
6. Multi-head Turing Machine
7. Non-deterministic Turing Machine

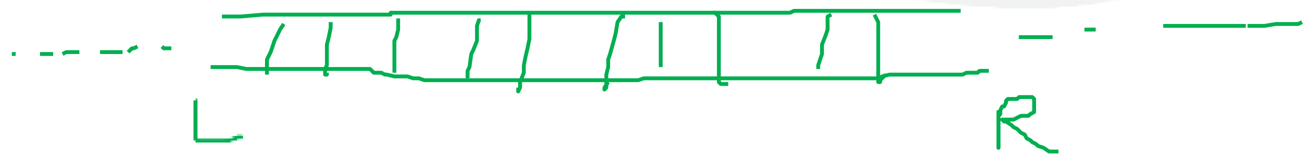
L =

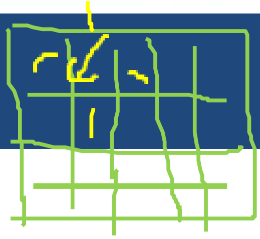
same
power



Contd.

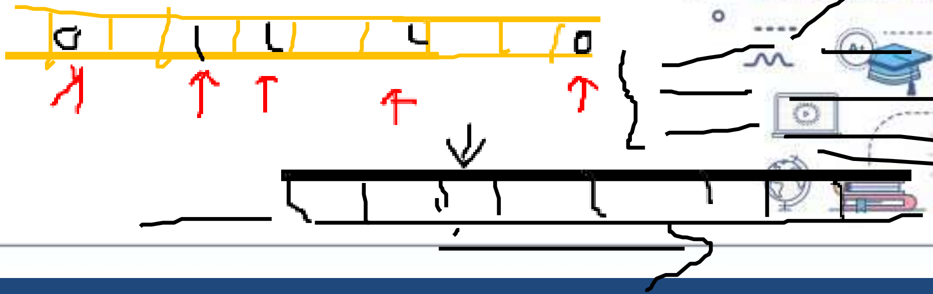
1. **Multiple track Turing Machine:**
   - A k-tack Turing machine(for some k>0) has k-tracks and one R/W head that reads and writes all of them one by one.
   - A k-track Turing Machine can be simulated by a single track Turing machine
2. **Two-way infinite Tape Turing Machine:**
   - Infinite tape of two-way infinite tape Turing machine is unbounded in both directions left and right.
   - Two-way infinite tape Turing machine can be simulated by one-way infinite Turing machine(standard Turing machine).





Contd.

3. **Multi-tape Turing Machine:**
   - It has multiple tapes and controlled by a single head.
   - The Multi-tape Turing machine is different from k-track Turing machine but expressive power is same.
   - Multi-tape Turing machine can be simulated by single-tape Turing machine.
4. **Multi-tape Multi-head Turing Machine:**
   - The multi-tape Turing machine has multiple tapes and multiple heads
   - Each tape controlled by separate head. Multi-Tape Multi-head Turing machine can be simulated by standard Turing machine.
5. **Multi-dimensional Tape Turing Machine:**
   - It has multi-dimensional tape where head can move any direction that is left, right, up or down. Multi dimensional tape Turing machine can be simulated by one-dimensional Turing machine



Contd.

6. **Multi-head Turing Machine:**

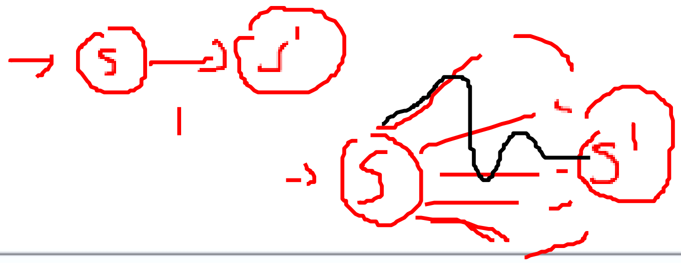
5

- A multi-head Turing machine contain two or more heads to read the symbols on the same tape.
- In one step all the heads sense the scanned symbols and move or write independently. Multi-head Turing machine can be simulated by single head Turing machine.

7. **Non-deterministic Turing Machine:**

- A non-deterministic Turing machine has a single, one way infinite tape. For a given state and input symbol has atleist one choice to move (finite number of choices for the next move), each choice several choices of path that it might follow for a given input string.
- A non-deterministic Turing machine is equivalent to deterministic Turing machine.

NTM $\Leftrightarrow$ DTM



Deterministic vs. non-determinism Turing Machines

- Deterministic TM's owe their name to the fact that each computation can be viewed as a linear ordered sequence (finite or infinite) of configurations. The first element contains the initial state and each configuration C′ follows directly configuration C in the sequence if the TM can change C into C′ as a consequence of a legal move of the machine.
- Nondeterministic machines are not restricted to such ordered computations. At each moment in time a non-deterministic TM may be allowed to make several moves (or equivalently to choose among a number of possibilities). Hence its computation may "branch" in many possible ways. The resulting computation is best described therefore as a rooted tree of configurations. Each deterministic computation corresponding to a particular sequence of choices is called a computation path.

[illegible]

T, N

Unrestricted grammars.

S →

- Definition: an unrestricted grammar is a 4-tuple (T,N,P,S), consisting of:
  - T = set of terminals (the legal "tokens" of the language) {a, b, c}
  - N = set of non-terminals (aka variables)
  - P = as set of productions, each of the form: T,N → T, N

    v → w

    where v and w are strings consisting of non-terminals and terminals.
  - S = a special nonterminal called the start symbol.

This type of grammar is also known as a **Type 0** grammar. It is equivalent to a **Turing Machine**. Given a Type 0 grammar, there exists a TM that can "recognize" the same language as that grammar. And, conversely, a Type 0 grammar can always be found for the language recognized or generated by any TM.

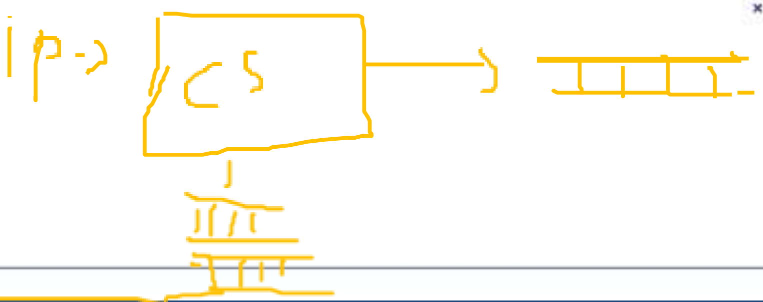
Example

- Example: Grammar for L={ ww : w∈(a, b)∗}

- S→S′Z
- S′→aS′A
- S′→bS′B
- S′→ ø

w = abb

{abbabb}



TMs Enumerator.

An enumerator is multi-tape Turing Machine, with a special output tape which is write-only.

- Write only means (a) symbol on output tape does not affect transitions, and (b) tape head only moves right.
- Initially all tapes blank (no input). During computation the machine adds symbols to the output tape. Output considered to be a list of words (separated by special symbol #).
- **Definition**: An enumerator M is said to enumerate a string W if and only if at some point M writes a word W on the output tape.

Contd.

- Enumerate all the strings for L={$a^n b^n c^n$|n≥0}

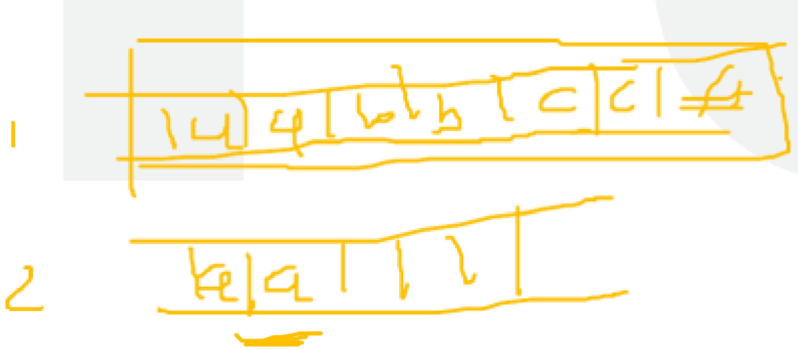
Solution. The L can be generated by two-tape TM E with following steps:

(1) Write # # on tape 1 and 2 for ε.

(2) Add a on tape 2,

(3)Copy a equal to size of tape 2 on tape 1, then by same size the b is written on tape 1, then by same size c is written tape 1, followed with #

(4) go to step 2.



DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in