**Chapter 11: TURING MACHINE**

**Topic – 1: Introduction To Turing Machine (TM)**

**Introduction**

* **Recursive enumerable:** A language for which a **TM** can be constructed which **recognizes** all valid strings in it.
* Contains **external memory**, for accepting long inputs.
* **Unlimited** memory capability.
* Output can be gained from **input itself**, without having separate output.

**Formal Definition Of TM**

**{Q, Σ, T, q0, F, B, δ}**

**T = Tape symbol**

**B = Blank symbol meaning input tape is over**

**Transition Mapping**

* **Turing’s program** which tells which **direction** to move the tape at each stage is called a **triple**.

**(q0, a) 🡪 (q1, A, R)**

**It means state q0 transitions to state q1 on getting input symbol 'a'.**

**'a' is replaced with A & R means direction right.**

**Example**

**Ques: Construct TM for language {0n1n | n ≥ 1}**

**Ans:**

**Logic:**

**Step 1: First element is pointed by default.**

**Step 2: First 0 is replaced by A & transitions as (q0, 0) 🡪 (q1, A, R).**

**Step 3: Keeps moving in R direction until 1 is found.**

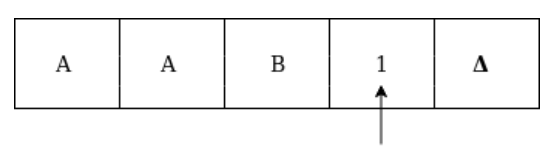
**Step 4: When 1 is found, transition (q1, 1) 🡪 (q2, B, L) occurs.**

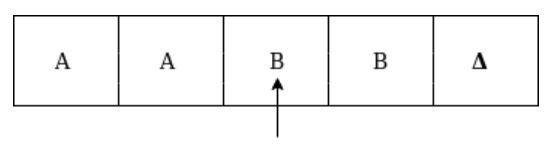
**Step 5: Repeat step 2,3,4 until all are replaced.**

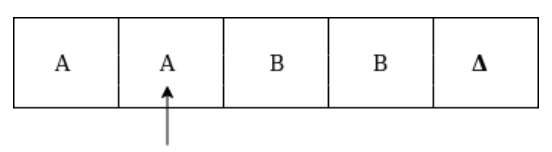
**Step 6: When all get replaced, we will move toward the blank symbol.**

**Aftermath:**

**After replacing last B, we move toward left.**

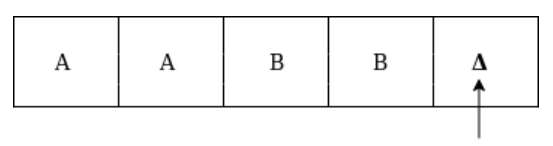
****

****

****

**At this stage when we counter an A immediately left to a B, we come to know that all the characters have been replaced.**

**So, we will make transition (q0, A) 🡪 (q3, B, R) & move right to find blank symbol (Δ) to end the process.**

****

**Now it makes transition (q3, Δ) 🡪 (q4, Δ, R) i.e. our HALT state.**

**Note!**

**🡪 The behaviour of pointer like changing direction on certain conditions & knowing when our objective has been achieved etc, varies language to language.**

**🡪 Remember the input tape we introduced in starting? That was not similar to Turing machine, but it was Turing machine itself!**

**🡪 In recent example, the machine changes second 'a' into A only after pointer moves & checks that previous symbol is A.**

**🡪 States in TM change only when either any change is made or we are at a cell giving us information about where to make change.**

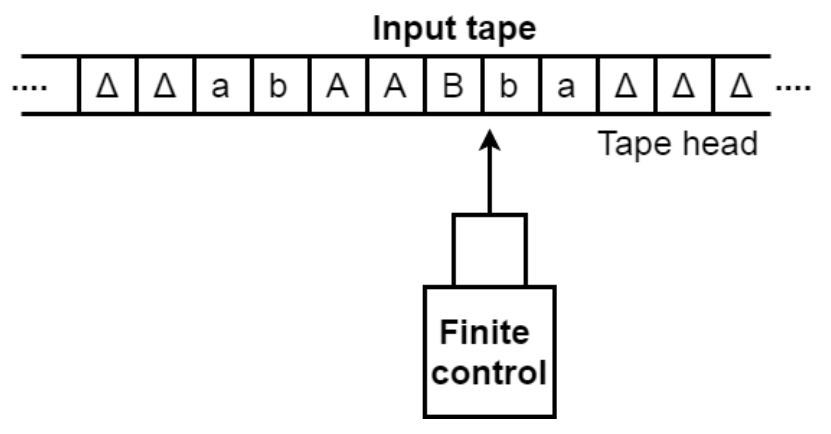
**🡪 Transition mappings tell everything about current cell only along with direction.**

**Topic – 2: Basic TM Model**

**Features**

* There can be **infinite** number of cells in input tape.
* **External characters** can be used in building logic for machine.
* The pointer we talked about was actually **finite control** itself.

**Example Diagram**



**Topic – 3: Language Accepted By TM**

**Introduction**

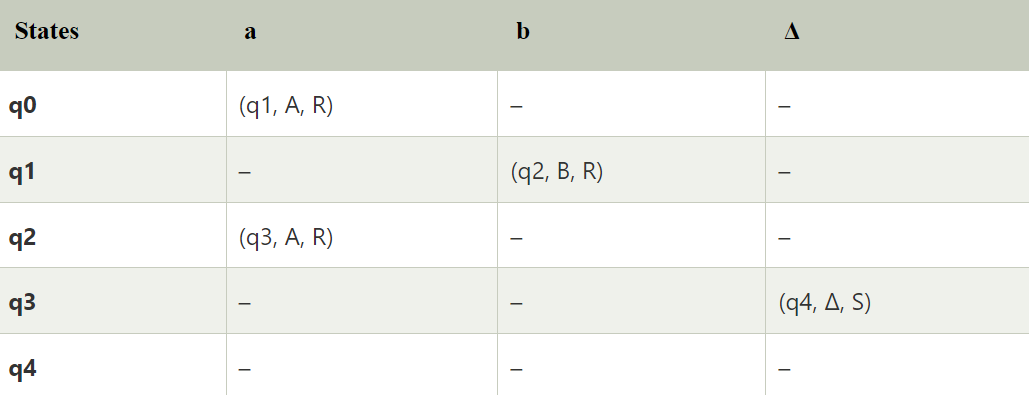
* **TM** also accepts **calculative functions** (arithmetic functions).
* We can construct **transition table** for **Turing machine** too.

**Example**

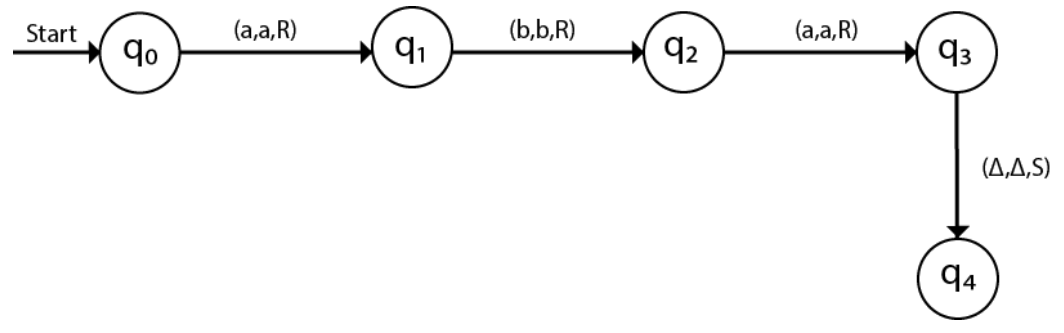
**Ques: Construct Turing machine for language "aba" over {a, b}. Also make its transition table.**

**Ans:**

**Transition table:**

****

**State diagram:**

****

**Example – II**

**Ques: Construct a TM for language {0n1n2n | n ≥ 1}.**

**Ans:**

**Algorithm:**

**When turning a 2 as C, move right once to check if end has come or not.**

**If we see that we are on Δ, then move toward left afterwards.**

**This cuts down the computation time, though there can be multiple approaches.**

**Note!**

**🡪 There can be multiple Turing machines for a given language.**

**Example – III**

**Ques: Construct a TM to check if a number of even length is palindrome or not.**

**Ans:**

**Algorithm:**

**Step 1: Check the first symbol from left.**

**Step 2: Then compare it to first symbol from right.**

**Step 3: Then check the symbol right to last symbol checked from left.**

**Step 4: Then compare it with so from right.**

**Step 5: When we encounter the replaced symbol immediately next to another one, we will know that its time to HALT.**

**Step 6: Or if a symbol doesn’t match, we have to HALT there too.**

**Example – IV**

**Ques: Construct a TM to add two unary numbers.**

**Ans:**

**Unary sum example:**

**3 + 2 = 111 + 11 = 11111**

**It is showing behaviour of concatenation.**

**Algorithm:**

**Step 1: When the addition operator ('+' mostly) is noticed, replace it with 1.**

**Step 2: Replace first 1 from right with Δ.**

**Topic – 5: Undecidability**

**Introduction**

* Problems which can be answered with **Yes** or **No**, are known as **solvable/decidable**.
* Other kind of problems are called **unsolvable/undecidability**.

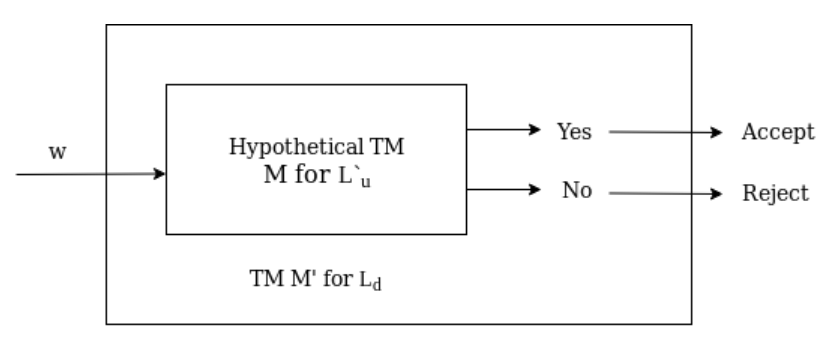
**Theorem**

* **LU** is **recursively enumerable** but not **recursive**.
* **LU**here means **universally undecidable languages**.

**Theorem’s Proof**

* Let’s assume that **LU** is **recursively enumerable language** & thus **recursive**.
* So, the complement of **LU** i.e. **L’U** is also **recursive**.
* To let **LU**be accepted by TM, we will modify it to a language **Ld**.
* But it **won’t** be a **recursive enumerable** anymore, thus **LU**is **not** recursive.

**Machine For Ld**



**Topic – 6: Undecidability Problem About TM**

**Reduction**

* ***Reduction*** is a technique used to convert an **undesired language** into a **desired** one.
* But their properties **remain same** after **reduction**.
* These properties include being **recursive**, **recursively enumerable**, **undecidable** etc.
* For proof, these can be explained by stating how properties get transferred in **reduced forms**.

**Empty & Non-Empty Languages**

**Le = Empty language**

**Lne = Non-empty language**

**Mi = TM for binary string w**

**Mi doesn’t accept input for L(Mj) if w Є Le.**

**Else w is in Lne.**

**Le = {M | L(M) = φ}**

**Lne = {M | L(M) ≠ φ}**

**Means both are each other’s complement.**

**Topic – 7: Post Correspondence Problem (PCP)**

**Introduction**

* This topic is related to **undecidability of string**, not **TM**.
* In layman’s language, it’s a way to **arrange symbols** from **two separate sets** in such a way that they give **equal/same strings**.

**Definition**

* Contains **two lists** of strings having **equal length**.
* These **lists** are generally represented as **A** & **B**.

**A = w1, w2, w3, …, wn**

**B = x1, x2, x3, …, xn**

**There exists a non-empty set of integers I:**

**i1, i2, i3, …, i4**

**This set ensures that:**

**wi1, wi2, wi3, …, win = xi1, xi2, xi3, …, xin**

* We have to try **various combinations** of **I** to get **wi = xi**.
* This gives us **solution** to **PCP**.

**Example**

**Ques: Find solution for PCP for given sets over {a, b}.**

**A = {b, bab3, ba}**

**B = {b3, ba, a}**

**Ans: A possible solution is 2,1,1,3.**

**Equation:**

**w2 w1 w1 w3 = x2 x1 x1 x3**

**Solution:**

**w2 w1 w1 w3**

**= bab3 . b . b . ba**

**= bab6a**

**x2 x1 x1 x3**

**= ba . b3 . b3 . a**

**= bab6a**

**Note!**

**🡪 Each element in set must be used at least once when constructing solution.**

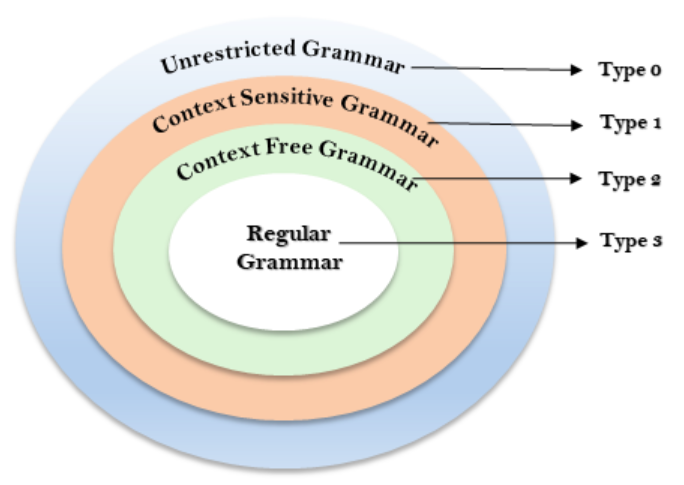
**🡪 The sequence of indices must be same however in both.**

**🡪 There can be multiple solutions to same problem.**

**Topic – 8: Chomsky’s Hierarchy**

**Introduction**

* ***Chomsky’s hierarchy*** tells us about various **classes** of languages accepted by different machines.



**Unrestricted Grammar (Type 0)**

* There is **no restriction** on grammar rules in it.
* Can be **easily constructed** using Turing machines.

**Example:**

**bAa 🡪 aa**

**S 🡪 s**

**Context-Sensitive Grammar (Type 1)**

* Includes **type 0** grammar in it.
* Can have **more than one** symbol on **LHS**.
* Number of symbols on **LHS** must be **same or less** than on **RHS**.
* Only **start symbol** can use **Ԑ**.

**Example:**

**S 🡪 AT**

**T 🡪 xy**

**A 🡪 a**

**Context-Free Grammar (Type 2)**

* Includes **type 0** & **type 1** grammar in it.
* As we discussed in many previous chapters about **CFG**.
* **LHS** contains a **terminal** & **RHS** contains combination of any number of **terminals** and **non-terminals**.

**Regular Grammar (Type 3)**

* Includes **type 0**, **type 1** & **type 2** grammar in it.
* These can be represented as **regular expressions**.
* And thus, these can be represented as **DFA** or **NFA** as well.
* It imposes **most restriction** on construction of languages.

**Representation:**

**V 🡪 T\*V | T\***

**V = Terminal symbol**

**T = Non-terminal symbol**

**Example:**

**A 🡪 xy**