Turing Machines

Until right now we have seen PDA or FA. The main difference between PDA, FA and Turing machines is that PDA and FA move in only one direction while turning machine can move in both the directions. Turning machine can read and write a symbol on the tape.

Turning machine:

1. It can move left and right
2. It can do both read and write operations on the tape
3. It may not have all the configurations from a state and still be deterministic
4. If a string is not accepted by a turing machine, the machine will halt in a state other than the final state
5. If a string is accepted by a turing machine, the machine will halt in the final state
6. Turing machine is a hypothetical infinite tape with cells.

Let’s see an example:

Check whether the following string is accepted or not: B = Blank

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| … B | B | a | a | a | b | b | b | B | B … |



Let’s see one more example: L = {anbncn | n>=1}, make a turning machine for this one.



|  |
| --- |
|  |

**Turning machine: A transducer.**

FA and PDA are considered as acceptors. This means that when we send a symbol to these machines, they will tell us if they accept or reject the string or whether the input is present or absent in the language. On the other hand:

Turning machine can act as an acceptor and as a transducer. A transducer converts an input to an output. So turning machines can convert a given input to give an output too. Because turning machine has the capacity to read from the tape and write on it too, it has the capacity to change the symbols on the tape, meaning changing the input to some other output. Let’s take an example.

**Example 1: TM to find one’s complement of a binary string.**

*Note: It is a courtesy to leave the output at the start of the final modified string. Accha nahi lagta ki Blank pe chhod diya. Lol.*

Here the turning machine is not an acceptor of string, therefore there is no need of a final state. It can halt whenever the work has been done.

|  |
| --- |
|  |

**Example 2: Turing machine to find the 2’s complement of a given binary string**

Logic: From right, if zeros keep them as zeros, 1stone leave it as it is, 1’s complement the remaining symbols on the left of 1st 1.

String : 0111000

One’s : 1000111

Two’s: 1001000

|  |
| --- |
|  |

**Example 3: Turning machine as an adder**

Let’s use unary representation of the number.

3 = 111, 4 = 1111, 5 = 11111 and so on.



String will be given like: 11101111 (for adding 3 and 4) and we want in the end 1111111 (7) as the answer. Let’s do this.

|  |
| --- |
|  |

**Example 4: Turning machine as a comparator**

a=111 (number 3) , b=1111 (number 4). The turning machine should give me three things.

1. Whether a and b are equal (should halt in a state that indicates a is equal to b)
2. Whether a is greater than b (should halt in a state that indicates a >b)
3. Whether a is smaller than b (should halt in a state that indicates a < b)

Therefore, depending on the input our turning machine should always halt in one of the three states. Let’s see how can we do that.

|  |  |
| --- | --- |
|  |  |

*Note: addition and comparison are the basis operations for any mathematical operation. So by having a cascade of Turing machines you can represent functions like log, exponents, subtraction, division, multiplication by just addition and comparison operations. Therefore, Turing machine is mathematically complete. This means that given any mathematical function, Turing machine will be able to compute it because it is able to do addition and comparison. So* ***Turing machine can perform any mathematical function.***