Theory of Computation: CS-202

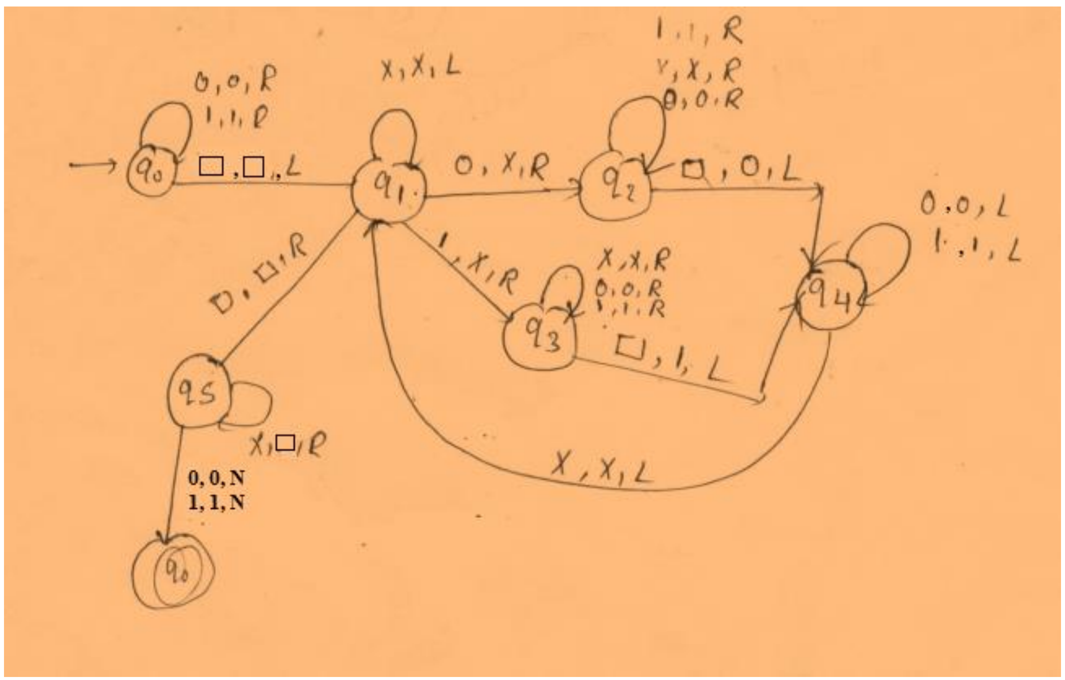
Turing Machine

Outline

- ❑ Variants of Turing machine
- ❑ Unrestricted Grammar
- ❑ Recursive Enumerable & Recursive Languages
- ❑ Halting problem of Turing Machine
- ❑ Computability & Decidability
- ❑ Post Correspondence Problem

Design a Turing Machine that computes string reversal

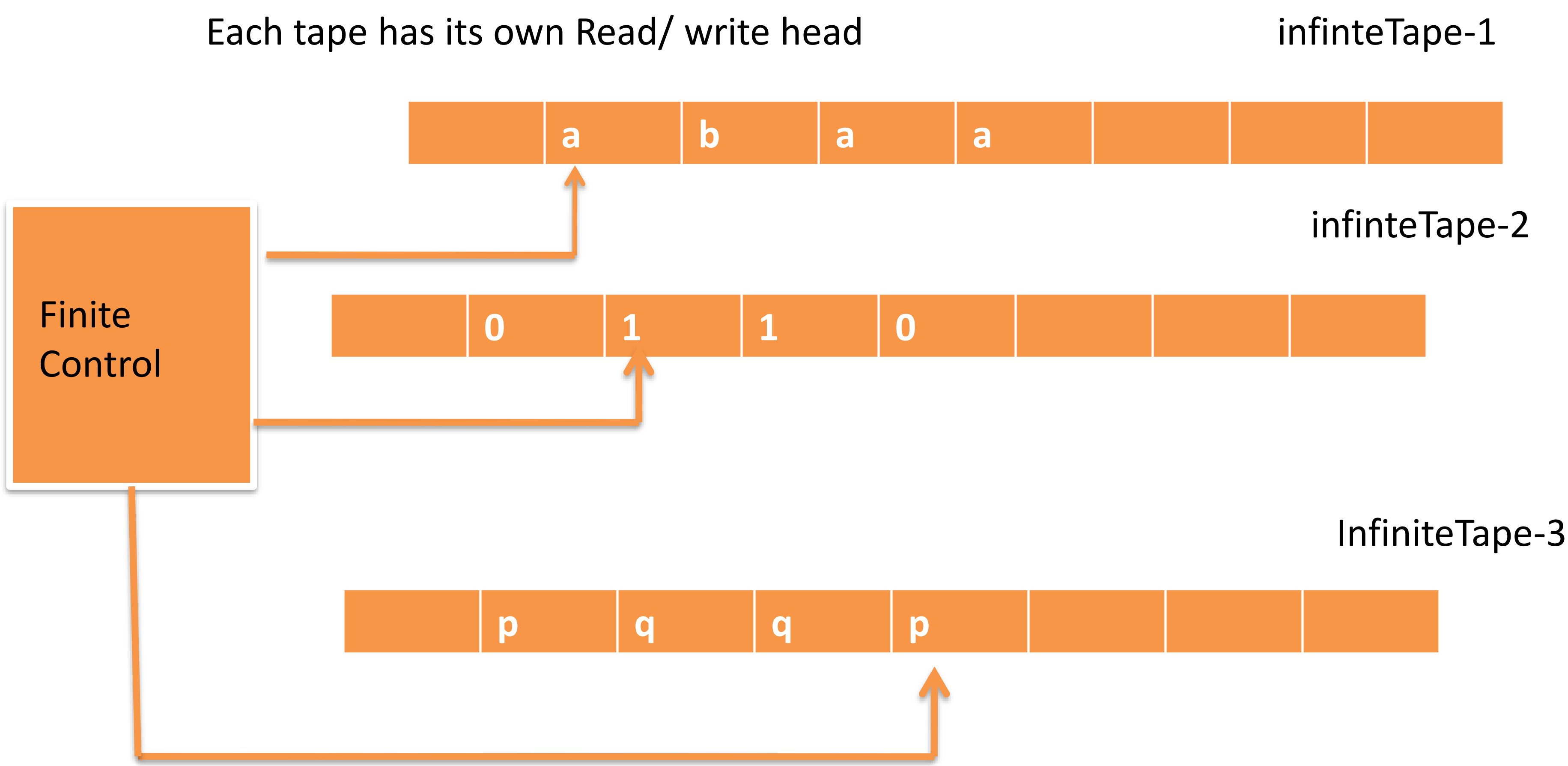
□	1	0	0	□	□	□



Variants of Turing Machine

- ❑Multi-tape Turing Machine
- ❑Multi-Track Turing Machine
- ❑Semi-infinite Tape Turing Machine
- ❑Non-Deterministic Turing Machine

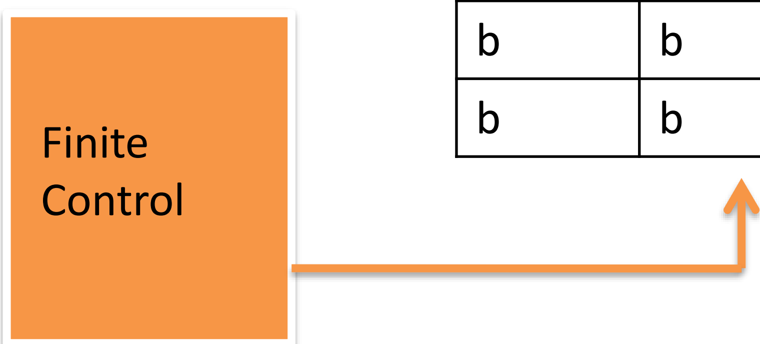
Multi-tape Turing Machine



$\delta(q_0, a1p)=(q_1, a0q, RRL)$

Multi-track Turing Machine

a	c	a	b			
b	b	a	b			
b	b	b	b			



Single tape is divided into multiple tracks

Semi infinite Tape Turing Machine

a	a	a	b	b		………

One end is infinite

Non-Deterministc Turing Machine

A Turing machine (TM) is defined by the seven-tuples:

$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$

Q	A finite set of internal states
Σ	A finite set of input alphabet
Γ	A finite set of symbols called tape alphabet
q_0	The initial/starting state, q_0 is in Q
$\square$	A special symbol called the blank symbol, is in Γ
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, where

$$\delta: Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L/R\}}$$

Linear Bounded Automata (LBA)

This model is important

- The set of context sensitive languages is accepted by the model.
- The infinite storage is restricted in size but not in accessibility to storage in comparison with Turing machine
- It is called Linear Bounded Automata because a linear function is used to restrict (or bound) the length of the tape.
- A LBA is a nondeterministic Turing machine which has a single tape whose length is not infinite but bounded by a linear function of the length of the input string.

Formal Definition of a Linear Bounded Automata (LBA)

A LBA is defined by the following set format:

$M = (Q, \Sigma, \Gamma, \delta, q_0, \square, ₵, \$, F)$

All the symbols have the same meaning as in the basic Turing Machine with the difference that the input alphabet Σ contains two different symbols ₵ and $.

- ₵ is called the left end marker (entered at leftmost cell of the input tape).
  It prevents the R/W head from getting off the left end of the tape.

- $ is called the right end marker (entered at rightmost cell of the input tape)
  It prevents the R/W head from getting off the right end of the tape.

Linear Bounded Automata (LBA)

Standard Turing machine has infinite length input tape.

In LBA, restrict the Turing machine to use the tape in which the input is present

LBA always halt.

₵	a	a	a	b	b	b	$

Context Sensitive Grammar (CSG)

A Grammar G=(V, T, S, P) is said to be context sensitive if all productions are of the form:

$$\alpha \to \beta$$

Where, $\alpha \in (V \cup T)^* \, V \, (V \cup T)^*$

$\beta \in (V \cup T)^+$

$|\alpha| \leq |\beta|$

$\Rightarrow$It does not incorporate ε productions.

Note: Add $S \to \varepsilon$ such that S should not occur twice on the R. H.S of any production.

Unrestricted Grammar

A Grammar G=(V, T, S, P) is said to be unrestricted if all productions are of the form:

$$\alpha \rightarrow \beta$$

Where, $\alpha \in (V \cup T) * V (V \cup T) *$

$\beta \in (V \cup T)^*$

For the Grammar G=(V, T, S, P)

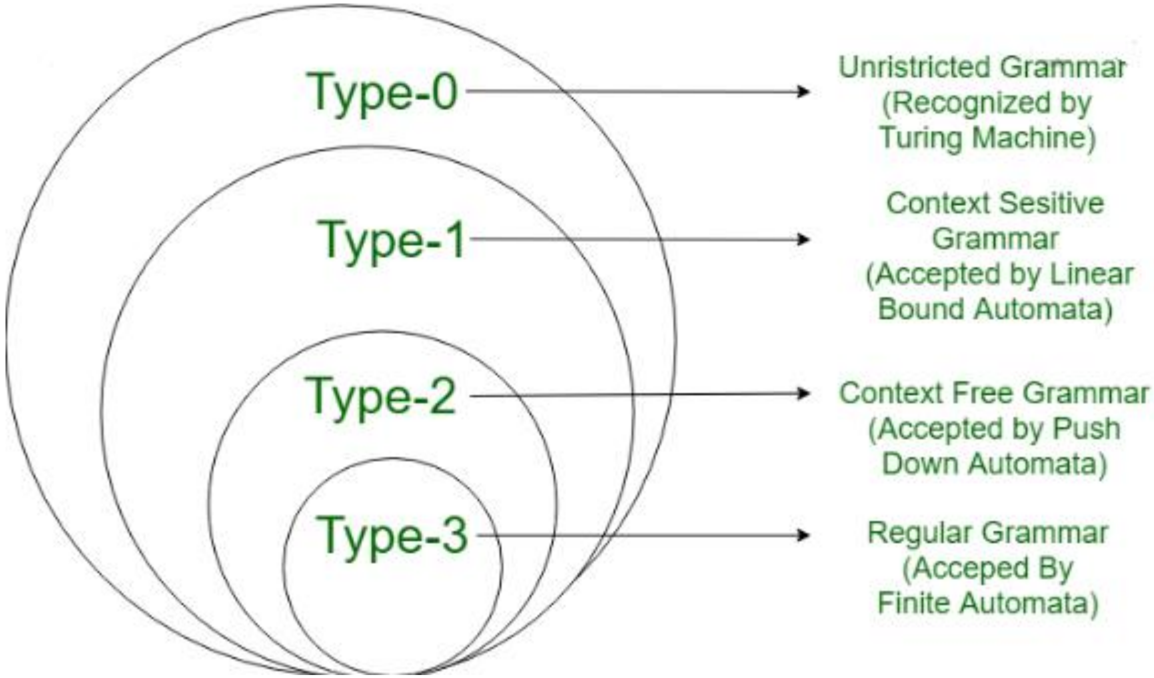
Regular Language
Recognized by
Finite Automata

Regular Grammar
$\alpha \rightarrow \beta$
Where,
$\alpha \in V$
$\beta \in VT^* | T^*$
Or $\beta \in T^* V | T^*$

Context free Language
Recognized by
Push Down Automata

Context free Grammar
$\alpha \rightarrow \beta$
Where,
$\alpha \in V$
$\beta \in (V \cup T)^*$

Chomsky Hierarchy



Recursive Enumerable Language

Language recognized by Turing machine (TM) is called recursive enumerable language.

A language $L \subseteq \Sigma^*$ is recursive enumerable if there exists a TM M such that $L=T(M)$.

Or

Consider a language L over Σ, TM can accept or reject or it may stuck infinite loop (no halting), such language is called recursive enumerable language.

Recursive Language

A language $L \subseteq \Sigma^*$ is recursive if there exists some TM M that satisfies the following two conditions.

(i) If $w \in L$ then M accepts w (that is, reaches an accepting state on processing w) and halts.

(ii) If $w \notin L$ then M eventually halts, without reaching an accepting state.

Decidability

A problem with two answers (Yes/No) is decidable if the corresponding language is recursive. In this case, the language *L* is also called *decidable.*

For a given problem, if a TM can accept or reject it, called Decidable problem.

Example: is ‘x’ a prime number?

For a given instance every problem will be always decidable.

A_{DFA}= {(B,w)| B accepts the input string w}
A_{DFA} is Decidable?

To prove this, we can construct a TM that always halts and also accepts $A_{DFA.}$

TM can be defined as follows:

1. Let B be a DFA and w an input string . (B, w) is an input for the TM M.
2. Simulate B and input w in the TM M.
3. If the simulation ends in an accepting state of B, then M accepts w. if it ends in a nonaccepting state of B, then M rejects w.

Note: It is evident that M accepts (B, w) iff w is accepted by the DFA B.

Un-decidability

A problem/language is undecidable if it is not decidable.

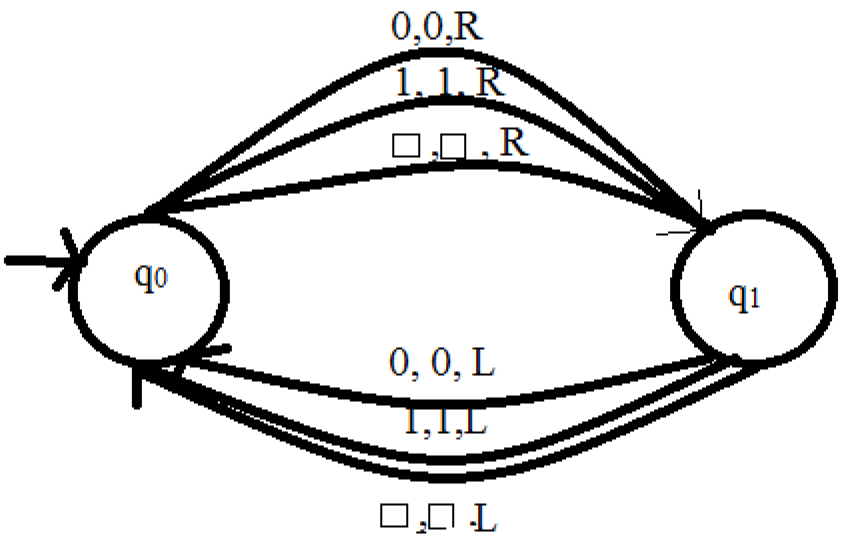
Note: A decidable problem is called a solvable problem and an undecidable problem an unsolvable problem.

Uncomputable (undecidable) problems

- Many well-defined (and apparently simple) problems cannot be solved by any computer
- Examples:
  - For any program x, does x have an infinite loop?
  - For any two programs x and y, do these two programs have the same input/output behavior?

A_{TM}= {(M, w)| The Turing Machine M accepts w}, A_{TM} is undecidable.

Halting problem of Turing Machine



Reduction technique

Reduction technique: It is used to prove the undecidability of halting TM.

We say that problem *A* is reducible to problem *B* if a solution to problem *B* can be used to solve problem *A*.

For example, if *A* is the problem. of finding some root of $x^4 - 3x^2 + 2 = 0$ and *B* is the problem of finding some root of $x^2 - 2 = 0$, then *A* is reducible to *B*.

As $x^2 - 2$ is a factor of $x^4 - 3x^2 + 2$. a root of $x^2 - 2 = 0$ is also a root of $x^4 - 3x^2 + 2 = 0$.

Note: If *A* is reducible to B and *B* is decidable then *A* is decidable.

If *A* is reducible to *B* and B is undecidable. then *A* is undecidable.

Post Correspondence Problem

The Post Correspondence Problem (PCP) was first introduced by Emil Post in 1946. Later, the problem was found to have many applications in the theory of formal languages. The problem over an alphabet Σ belongs to a class of yes/no problems and is stated as follows:

Consider the two lists $x = (x_1, x_2 .. x_n)$, $y = (y_1, y_2 .. y_n)$ of nonempty strings over an alphabet $\Sigma = \{0, 1\}$.

The PCP is to determine whether or not there exist $i_1, i_2 .. i_n$ *where,* $1 \leq i_j \leq n$ *Such that*

$$x_{i1} \ldots \ldots x_{im} = y_{i1} y_{i2} \ldots y_{im}$$

Note: The indices i_j's need not be distinct and *m* may be greater than *n*.
Also, if there exists a solution to PCP, there exist infinitely many solutions.

PCP (Example)

Does the PCP with two lists $x = (b, bab^3, ba)$ and $y = (b^3, ba, a)$ have a solution?

We have to determine whether or not there exists a sequence of substrings of x such that the string formed by this sequence and the string formed by the sequence of corresponding substrings of y are identical. The required sequence is given by $i_1 = 2$, $i_2 = 1$, $i_3 = 1$, $i_4 = 3$, i.e. (2, 1, 1,3), and $m = 4$. The corresponding strings are

$$\underset{x_2}{\boxed{bab^3}} \quad \underset{x_1}{\boxed{b}} \quad \underset{x_1}{\boxed{b}} \quad \underset{x_3}{\boxed{ba}} \quad = \quad \underset{y_2}{\boxed{ba}} \quad \underset{y_1}{\boxed{b^3}} \quad \underset{y_1}{\boxed{b^3}} \quad \underset{y_3}{\boxed{a}}$$

Thus the PCP has a solution.

Thank you