Theory of Automata and Formal Languages (ACSE0404)

Unit: V

Turing Machine

B. Tech

(Information Technology)

4th Semester

Index

- Course Outcomes
- Syllabus
- Contents of the Unit
- Objectives of the Unit
- CO- PO correlation w.r.t. Unit
- CO-PSO correlation w.r.t. Unit
- Prerequisite for the course
- Objectives of the topic
- Topic mapping with CO
- Video Links
- Daily Quiz
- MCQs
- Weekly assignment
- Old University Exam Paper
- Expected Questions for University Exams
- Summary
- References

Result Analysis

Subject Result:

Department Result:

Faculty-Wise Result:

End Semester Question Paper Template

M TECH

(SEM-V) THEORY EXAMINATION 20__-20__

COMPILER DESIGN

Time: 3 Hours **Total Marks: 100**

Note: 1. Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt all questions in brief. **2 x 10 = 20**

Q.No.	Question	Marks	CO
1		2	
2		2	
.		.	
10		2	

End Semester Question Paper Templates

SECTION B

2. Attempt any three of the following: 3 x 10 = 30

Q.No.	Question	Marks	CO
1		10	
2		10	
.		.	
5		10	

SECTION C

3. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

End Semester Question Paper Templates

4. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

5. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

6. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

End Semester Question Paper Templates

7. Attempt any one part of the following: 1 x 10 = 10

Q.No.	Question	Marks	CO
1		10	
2		10	

Evaluation Scheme

Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Scheme				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
1	AAS0404	Optimization and Numerical Techniques	3	1	0	30	20	50		100		150	4
2	AASL0401	Technical Communication	2	1	0	30	20	50		100		150	3
3	ACSE0403A	Operating Systems	3	0	0	30	20	50		100		150	3
4	ACSAI0402	Database Management Systems	3	1	0	30	20	50		100		150	4
5	ACSAI0401	Introduction to Artificial Intelligence	3	0	0	30	20	50		100		150	3
6	ACSE0404	Theory of Automata and Formal Languages	3	0	0	30	20	50		100		150	3
7	ACSE0453A	Operating Systems Lab	0	0	2				25		25	50	1
8	ACSAI0452	Database Management Systems Lab	0	0	2				25		25	50	1
9	ACSAI0451	Introduction to Artificial Intelligence Lab	0	0	2				25		25	50	1
10	ACSE0459	Mini Project using Open Technology	0	0	2				50			50	1
11	ANC0402 / ANC0401	Environmental Science*/ Cyber Security* (Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs **(For B.Tech. Hons. Degree)											
		GRAND TOTAL										**1100**	**24**

Subject Syllabus

B. TECH. SECOND YEAR			
Course Code	**ACSE0404**	**L T P**	**Credits**
Course Title	**Theory of Automata and Formal Languages**	**3 0 0**	**3**

Course objective:

To teach mathematical foundations of computation including automata theory, provide the design concepts of abstract computation model of finite automata, push down automata and turing Machine and familiarize the notions of algorithm, decidability, complexity, and computability.

Pre-requisites:

- Discrete Mathematics
- Fundamental of Computer System

Course Contents / Syllabus

UNIT-I	Basic Concepts of Formal Language and Automata Theory	8 Hours

Introduction to Theory of Computation- Alphabet, Symbol, String, Formal Languages, Grammar, Derivation and Language generation by Grammar, Chomsky Hierarchy, Finite Automata, Deterministic Finite Automaton (DFA)- Definition, Representation, Acceptability of a String and Language, Non-Deterministic Finite Automaton (NFA), Equivalence of DFA and NFA, NFA with ∈-Transition, Equivalence of NFA's with and without ∈-Transition, Finite Automata with output- Moore Machine, Mealy Machine, Equivalence of Moore and Mealy Machine, Minimization of Finite Automata, Myhill-Nerode Theorem, Simulation of DFA and NFA.

UNIT-II	Regular Language and Finite Automata	8 Hours

Regular Expressions, Transition Graph, Kleen's Theorem, Finite Automata and Regular Expression- Arden's theorem, Algebraic Method Using Arden's Theorem, Regular Grammars-Right Linear and Left Linear grammars, Conversion of FA into Regular grammar and Regular grammar into FA, Regular and Non-Regular Languages- Closure properties of Regular Languages, Pigeonhole Principle, Pumping Lemma, Application of Pumping Lemma.

Decidability- Decision properties, Finite Automata and Regular Languages, Simulation of Transition Graph and Regular language.

UNIT-III	Context Free Language and Grammar	8 Hours

Context Free Grammar (CFG)-Definition, Derivations, Languages, Derivation Trees and Ambiguity, Simplification of CFG, Normal Forms- Chomsky Normal Form (CNF), Greibach Normal Form (GNF), Pumping Lemma for CFL, Closure properties of CFL, Decision Properties of CFL

Subject Syllabus

UNIT-IV	Push Down Automata	8 Hours
Pushdown Automata- Definition, Representation, Instantaneous Description (ID), Acceptance by PDA, Nondeterministic Pushdown Automata (NPDA)- Definition, Moves, Pushdown Automata and Context Free Language, Pushdown Automata and Context Free Grammar, Two stack Pushdown Automata.		
UNIT-V	**Turing Machine and Undecidability**	**8 Hours**
Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Variations of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Closure Properties of Recursive and Recursively Enumerable Languages, Non-Recursively Enumerable and Non-Recursive Languages, Undecidability, Halting Problem, Undecidability of Halting Problem, Post's Correspondence Problem.		
Course outcome: After completion of this course students will be able to:		
CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6
Text books:		
(1) Introduction to Automata theory, Languages and Computation, J.E. Hopcraft, R. Motwani, and Ullman. 3rd edition, Pearson Education Asia. (2) Theory of Computer Science-Automata Language and Computation, K.L.P. Mishra, and N. Chandrasekharan, 3rd Edition, PHI. (3) An Introduction to Formal Languages and Automata, P. Linz, 6th Edition, Jones & Bartlett Learning Publication.		
Reference Books:		
(1) Finite Automata and Formal Languages- A simple Approach, A. M. Padma Reddy, Cengage Learning Inc. (2) Elements and Theory of Computation, C Papadimitrou and C. L. Lewis, PHI. (3) Introduction to languages and the theory of computation, J Martin, 3rd Edition, Tata McGraw Hill. (4) Introduction to The Theory of Computation, M Sipser, 3rd Edition, Cengage Learning Inc.		

Links

LINKS
https://nptel.ac.in/courses/111103016

Branch Wise Applications

- It **can compute man-made problems as well as natural phenomena**.

- Automata theory has a lot of applications in real life as well, such that: Lambda calculus, Combinatory logic, Markov algorithm, and Register, Natural Language Processing ,Compiler Design and Lexical analysis and Semantic analysis. Also, the concept of Formal languages and automata theory can overlap with other subjects as well.

Course Objective

- Introduce concepts in automata theory and theory of computation
- Identify different formal language classes and their relationships and PDAs.
- Applying the Concept of Turing Machine and LBAs.
- Prove or disprove theorems in automata theory using its properties
- Determine the decidability and intractability of computational problems and complexity theory.

Course Outcome

Course outcome: After completion of this course students will be able to:		
CO 1	Design and Simplify automata for formal languages and transform non-deterministic finite automata to deterministic finite automata.	K6
CO 2	Identify the equivalence between the regular expression and finite automata and apply closure properties of formal languages to construct finite automata for complex problems.	K3
CO 3	Define grammar for context free languages and use pumping lemma to disprove a formal language being context- free.	K3
CO 4	Design pushdown automata (PDA) for context free languages and Transform the PDA to context free grammar and vice-versa.	K6
CO 5	Construct Turing Machine for recursive and recursive enumerable languages. Identify the decidable and undecidable problems.	K6

Program Outcomes (POs)

Engineering Graduates will be able to:

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

Program Outcomes (POs)

Contd..

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

Program Outcomes (POs)

Contd..

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

CO-PO correlation matrix

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	2	2	3	3	2	2	-	-	2	1	-	3
CO2	**1**	**3**	**2**	**3**	**2**	**2**	**-**	**1**	**1**	**1**	**2**	**2**
CO3	2	2	3	2	2	2	-	2	2	1	2	3
CO4	2	2	2	3	2	2	-	2	2	1	1	3
CO5	3	2	2	2	2	2	-	2	1	1	1	2
Average	2	2.2	2.4	2.6	2	2	-	1.4	1.6	1	1.2	2.6

CO-PSO correlation matrix

CO	PSO			
	PSO1	PSO2	PSO3	PSO4
CO1	2	2	2	2
CO2	2	2	1	1
CO3	2	2	1	1
CO4	2	2	1	1
CO5	2	2	2	2
Average	2	2	1.4	1.4

Program Educational Objectives

PEO1: To have an excellent scientific and engineering breadth so as to comprehend, analyze, design and provide sustainable solutions for real-life problems using state-of-the-art technologies.

PEO2: To have a successful career in industries, to pursue higher studies or to support entrepreneurial endeavors and to face global challenges.

PEO3: To have an effective communication skills, professional attitude, ethical values and a desire to learn specific knowledge in emerging trends, technologies for research, innovation and product development and contribution to society.

PEO4: To have life-long learning for up-skilling and re-skilling for successful professional career as engineer, scientist, entrepreneur and bureaucrat for betterment of society

CO-PEO Mapping

- CO-PEO correlation matrix

Sr. No	Course Outcome	PEO1	PEO2	PEO3	PEO4
1	CO 1	1	2	2	2
2	CO 2	2	1	3	1
3	CO 3	2	3	1	2
4	CO 4	3	2	1	3
5	CO 5	1	2	2	1

Syllabus

UNIT-2

Turing Machines and Recursive Function Theory : Basic Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Modifications of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Halting Problem, Post's Correspondance Problem, Introduction to Recursive Function Theory.

UNIT-5

Topics	Duration (in Hours)
TURING MACHINE	2
Techniques for Turing Machine Construction	3
Universal Turing machine, Linear Bounded Automata,	1
Church's Thesis, Recursive and Recursively Enumerable language,	1
Halting Problem, Post's Correspondance Problem, Introduction to Recursive Function Theory.	2

Objective of Unit

Topic Objective: To understand the formal definition of Turing Machine.

Mathematical representation of Turing Machine

IDs of Turing Machine

Recap: In previous unit we have studied about Push Down Automata.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

CO-PO correlation matrix

CO4	2	2	2	3	2	2	-	2	2	1	1	3
CO5	3	2	2	2	2	2	-	2	1	1	1	2

CO-PSO correlation matrix

CO-PSO correlation matrix w.r.t. Unit-5

CO5	2	2	2	2

Prerequisite and Recap

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Recap: In previous unit we have studied about Push Down Automata.

Turing Machine(CO5)

Objective of the Topic

The objective of the topic is to make the student able to:

- To understand the formal definition of Turing Machine.
- Mathematical representation of Turing Machine
- IDs of Turing Machine
-

Turing Machine(CO5)

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Turing Machine	-		-	-	3

Unit 5 Syllabus

Turing Machines and Recursive Function Theory : Basic Turing Machine Model, Representation of Turing Machines, Language Acceptability of Turing Machines, Techniques for Turing Machine Construction, Modifications of Turing Machine, Turing Machine as Computer of Integer Functions, Universal Turing machine, Linear Bounded Automata, Church's Thesis, Recursive and Recursively Enumerable language, Halting Problem, Post's Correspondance Problem, Introduction to Recursive Function Theory.

Introduction (CO5)

Topic Objective: To understand the formal definition of Turing Machine.

Mathematical representation of Turing Machine

IDs of Turing Machine

Recap: In previous unit we have studied about Push Down Automata.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Introduction (CO5)

Objective: To understand the formal definition of Turing Machine.

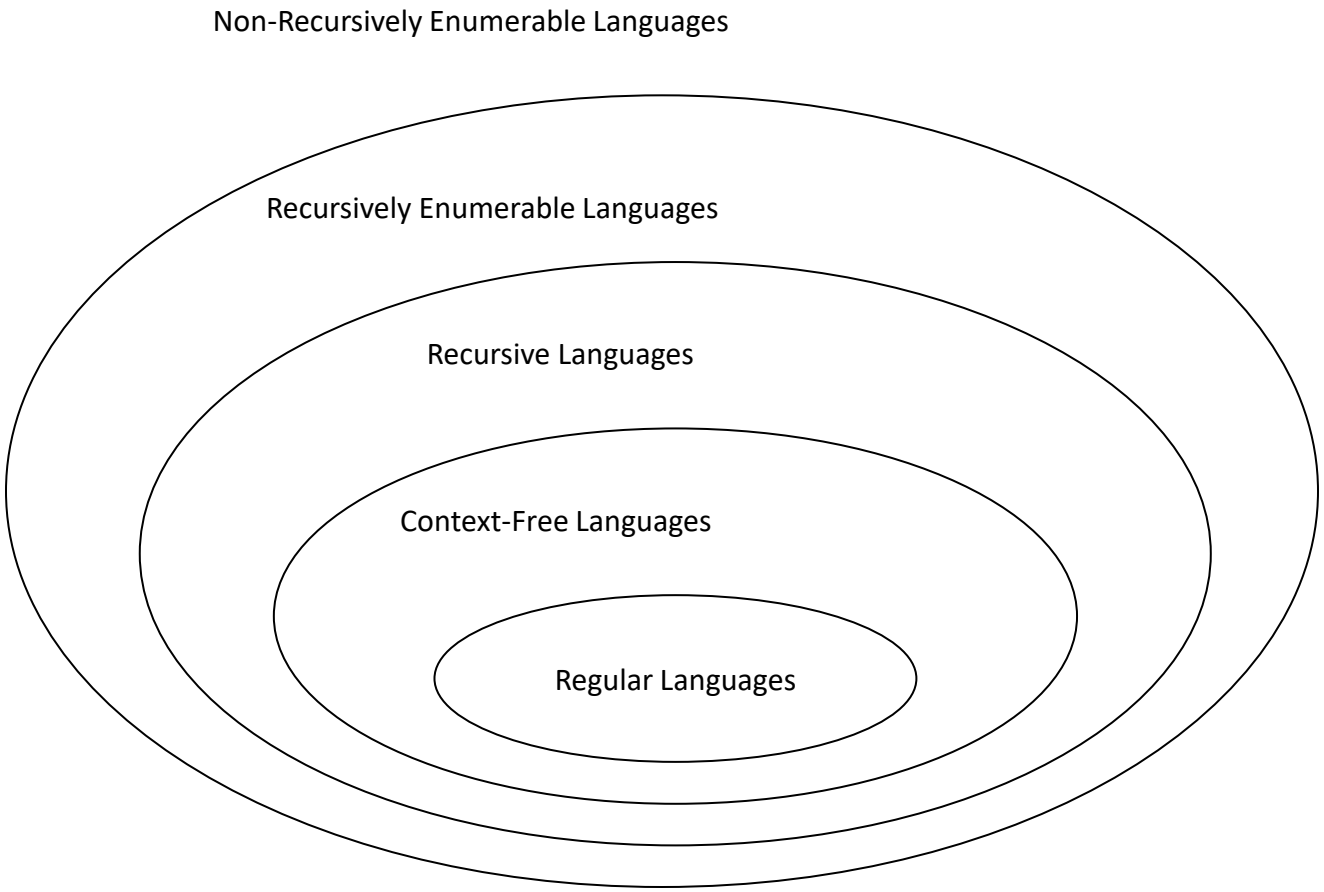
Turing machine was invented in 1936 by **Alan Turing**. It is an accepting device which accepts Recursive Enumerable Language generated by type 0 grammar.

There are various 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.

Introduction (CO5)

- **Generalize the class of CFLs:**



Formal definition of Turing machine(CO5)

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
q0: the initial state
F: a set of final states
B: a blank symbol used as a end marker for input
δ: a transition or mapping function.

- (q0, a) → (q1, X, R)
- That means in q0 state, if we read symbol 'a' then it will go to state q1, replaced a by X and move ahead right(R stands for right).

TM Example(CO5)

- Construct TM for the language L ={0^n1^n} where n>=1.

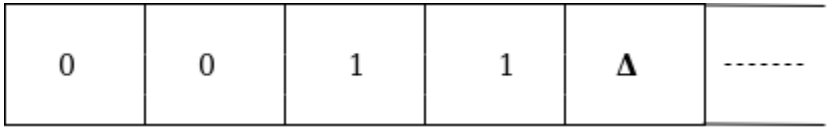
- We have already solved this problem by PDA. In PDA, we have a stack to remember the previous symbol. The main advantage of the Turing machine is we have a tape head which can be moved forward or backward, and the input tape can be scanned.

- The simple logic which we will apply is read out each '0' mark it by A and then move ahead along with the input tape and find out 1 convert it to B. Now, repeat this process for all a's and b's.

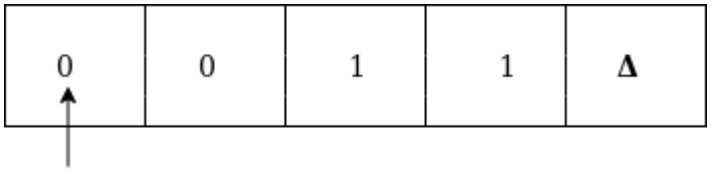
- Now we will see how this Turing machine work for 0011.

TM Example(CO5)

- The simulation for 0011 can be shown as below:

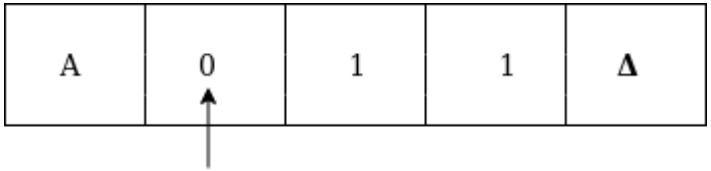


- Now, we will see how this turing machine will works for 0011. Initially, state is q0 and head points to 0 as:

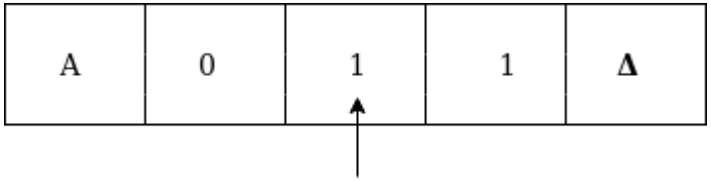


TM Example(CO5)

- The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A and head will move to the right as:

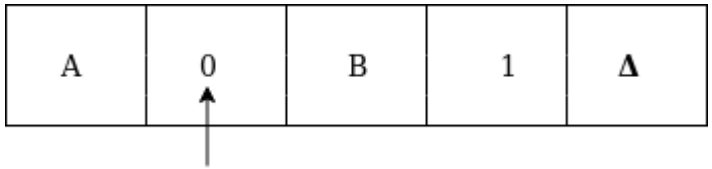


- The move will be δ(q1, 0) = δ(q1, 0, R) which means it will not change any symbol, remain in the same state and move to the right as:

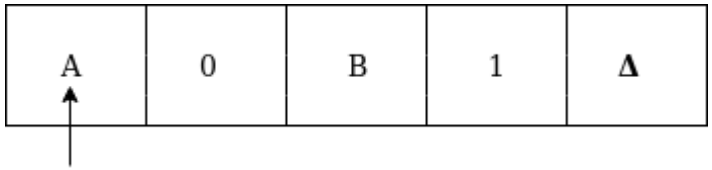


TM Example(CO5)

- The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:

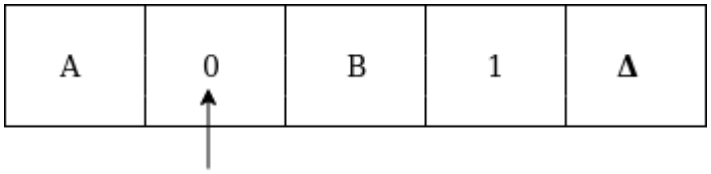


- Now move will be δ(q2, 0) = δ(q2, 0, L) which means it will not change any symbol, remain in the same state and move to left as:

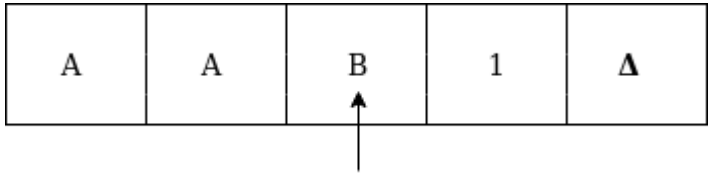


TM Example(CO5)

- The move will be δ(q2, A) = δ(q0, A, R), it means will go to state q0, replaced A by A and head will move to the right as:

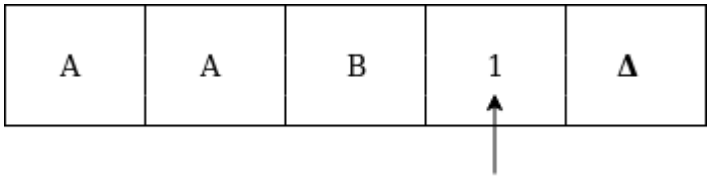


- The move will be δ(q0, 0) = δ(q1, A, R) which means it will go to state q1, replaced 0 by A, and head will move to right as:

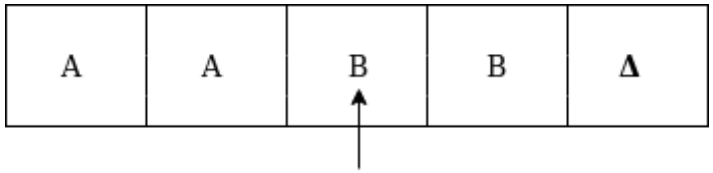


TM Example(CO5)

- The move will be δ(q1, B) = δ(q1, B, R) which means it will not change any symbol, remain in the same state and move to right as:

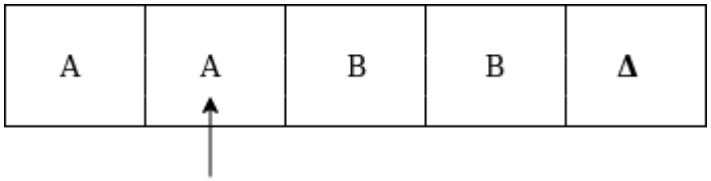


- The move will be δ(q1, 1) = δ(q2, B, L) which means it will go to state q2, replaced 1 by B and head will move to left as:

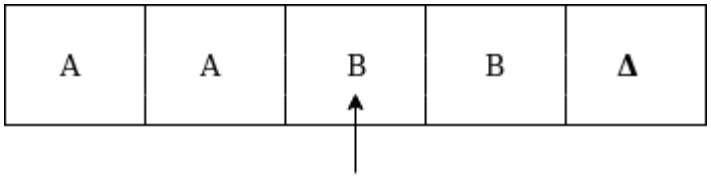


TM Example(CO5)

- The move δ(q2, B) = (q2, B, L) which means it will not change any symbol, remain in the same state and move to left as:

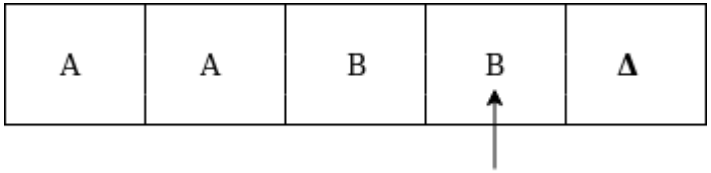


- Now immediately before B is A that means all the 0?s are market by A. So we will move right to ensure that no 1 is present. The move will be δ(q2, A) = (q0, A, R) which means it will go to state q0, will not change any symbol, and move to right as:

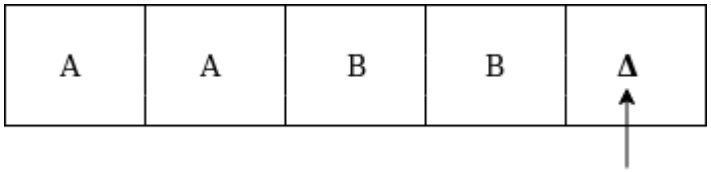


TM Example(CO5)

- The move δ(q0, B) = (q3, B, R) which means it will go to state q3, will not change any symbol, and move to right as:

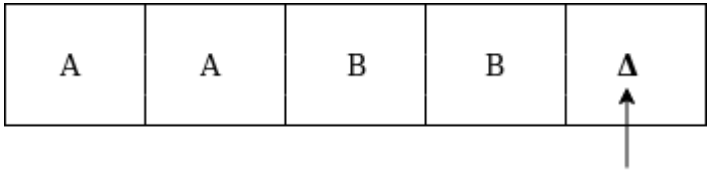


- The move δ(q3, B) = (q3, B, R) which means it will not change any symbol, remain in the same state and move to right as:

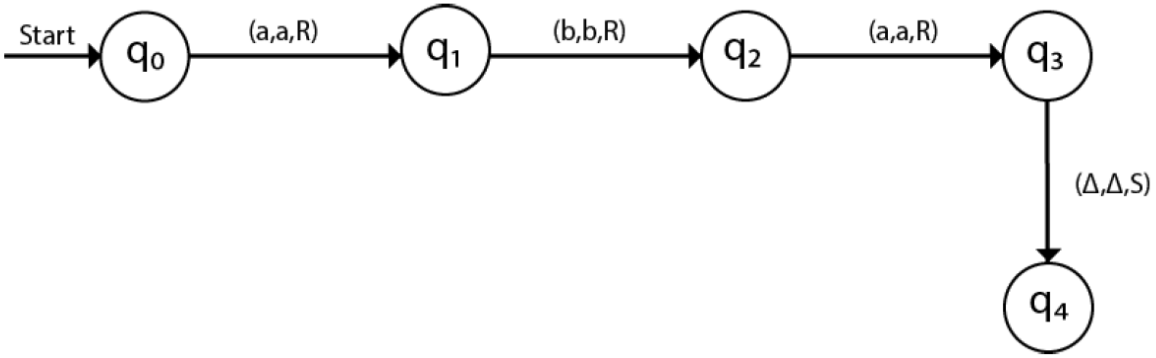


TM Example(CO5)

- The move δ(q3, Δ) = (q4, Δ, R) which means it will go to state q4 which is the HALT state and HALT state is always an accept state for any TM.



- The same TM can be represented by Transition Diagram:



TM Example(CO5)

- **Example #1:** {w | w is in {0,1}* and w ends with a 0}
  0, 00, 10, 10110
  Noteε
  $Q = \{q_0, q_1, q_2\}$
  $\Gamma = \{0, 1, B\}$
  $\Sigma = \{0, 1\}$
  $F = \{q_2\}$
  δ:

	0	1	B
->q_0	(q_0, 0, R)	(q_0, 1, R)	(q_1, B, L)
q_1	(q_2, 0, R)	-	-
q_2*	-	-	

  q_0 is the start state and the "scan right" state, until hits B
  - q_1 is the verify 0 state
  - q_2 is the final state

TM Example(CO5)

- **Example #2:** $\{0^n1^n \mid n \geq 1\}$

	0	1	X	Y	B
$->q_0$	(q_1, X, R)	-	-	(q_3, Y, R) ***0's finished***	-
q_1	$(q_1, 0, R)$ ***ignore1***	(q_2, Y, L)	-	(q_1, Y, R) ***ignore2***	-
(more 0's)					
q_2	$(q_2, 0, L)$ ***ignore2***	-	(q_0, X, R)	(q_2, Y, L) ***ignore1***	-
q_3	-	- (more 1's)	-	(q_3, Y, R) ***ignore***	
(q_4, B, R)					
q_4*	-	-	-	-	-

- **Sample Computation:** (on 0011), *presume state q looks rightward*

$q_0$0011BB.. |— X$q_1$011
|— X0$q_1$11
|— X$q_2$0Y1
|— q_2X0Y1
|— X$q_0$0Y1
|— XXq_1Y1
|— XXY$q_1$1
|— XXq_2YY
|— Xq_2XYY
|— XXq_0YY
|— XXYq_3Y B…
|— XXYYq_3 BB…
|— XXYYBq_4

Recap

Definition of Turing Machine.

Representation of Turing Machine

Acceptability and ID's of Turing Machine.

Daily Quiz

Q1: Definition of Turing Machine.

Q2: How we can represent of Turing Machine

Q3: what is ID's of Turing Machine.

Techniques of Turing Machine Construction(CO5)

Objective of the Topic

The objective of the topic is to make the student able to:

- How to design Turing machine for different mathematical models.

Prerequisite and Recap

Prerequisite: Representation of Turing Machine.

Recap: In previous lecture we have studied about representation and working of turing machine.

Construction of Turing Machine(CO5)

Topic mapping with Course Outcome

Topic	CO1	CO2	CO3	CO4	CO5
Construction of Turing machine	-		-	-	2

Construction of Turing Machine(CO5)

Making a TM for $\{0^n1^n \mid n >= 1\}$

Try n=2 or 3 first.

- q0 is on 0, replaces with the character to X, changes state to q1, moves right
- q1 sees next 0, ignores (both 0's and X's) and keeps moving right
- q1 hits a 1, replaces it with Y, state to q2, moves left
- q2 sees a Y or 0, ignores, continues left
- when q2 sees X, moves right, returns to q0 for looping step 1 through 5
- when finished, q0 sees Y (no more 0's), changes to pre-final state q3
- q3 scans over all Y's to ensure there is no extra 1 at the end (to crash on seeing any 0 or 1)
- when q3 sees B, all 0's matched 1's, done, changes to final state q4
- blank line for final state q4

Try n=1 next.

Make sure unbalanced 0's and 1's, or mixture of 0-1's," crashes" in a state not q4, as it should be

Construction of Turing Machine(CO5)

$q_0$0011BB.. |— X$q_1$011

|— X0$q_1$11

|— X$q_2$0Y1

|— q_2X0Y1

|— X$q_0$0Y1

|— XXq_1Y1

|— XXY$q_1$1

|— XXq_2YY

|— Xq_2XYY

|— XXq_0YY

|— XXYq_3Y B…

|— XXYYq_3 BB…

|— XXYYBq_4

Construction of Turing Machine(CO5)

Same Example #2: $\{0^n1^n \mid n \geq 1\}$

	0	1	X	Y	B
q_0	(q_1, X, R)	-	-	(q_3, Y, R)	-
q_1	$(q_1, 0, R)$	(q_2, Y, L)	-	(q_1, Y, R)	-
q_2	$(q_2, 0, L)$	-	(q_0, X, R)	(q_2, Y, L)	-
q_3	-	-	-	(q_3, Y, R)	(q_4, B, R)
q_4	-	-	-	-	-

Logic: cross 0's with X's, scan right to look for corresponding 1, on finding it cross it with Y, and scan left to find next leftmost 0, keep iterating until no more 0's, then scan right looking for B.

Construction of Turing Machine(CO5)

- The TM matches up 0's and 1's
- q_1 is the "scan right" state, looking for 1
- q_2 is the "scan left" state, looking for X
- q_3 is "scan right", looking for B
- q_4 is the final state

Can you extend the machine to include n=0? How does the input-tape look like for string epsilon?

- **Other Examples:**

000111	00
11	001
011	

Construction of Turing Machine(CO5)

- **Roger Ballard's TM for Example #2, without any extra Tape Symbol:** $\{0^n1^n \mid n \geq 0\}$

	0	1	B
q_0	(q_1, B, R)		(q_4, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	-	(q_3, B, L)	-
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4^*	-	-	-

Construction of Turing Machine(CO5)

Logic: cross 0's with X's, scan right to look for corresponding 1, on finding it cross it with Y, and scan left to find next leftmost 0, keep iterating until no more 0's, then scan right looking for B.

- The TM matches up 0's and 1's
- q_1 is the "scan right" state, looking for 1
- q_2 is the "scan left" state, looking for X
- q_3 is "scan right", looking for B
- q_4 is the final state

Can you extend the machine to include n=0? How does the input-tape look like for string epsilon?

- **Other Examples:**

000111	00
11	001
011	

Construction of Turing Machine(CO5)

- **And his example of a correct TM for the language that goes on infinite loop outside language:** $\{0^n1^n \mid n \geq 0\}$

	0	1	B
q_0	(q_1, B, R)	$(q_3, 1, L)$	(q_4, B, R)
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	(q_2, B, L)
q_2	-	(q_3, B, L)	-
q_3	$(q_3, 0, L)$	$(q_3, 1, L)$	(q_0, B, R)
q_4^*	-	-	-

Logic: This machine still works correctly for all strings in the language, but start a string with 1 (not in the language),
and it loops on B1 for ever.

Construction of Turing Machine(CO5)

- **Exercises:** Construct a DTM for each of the following.
  - {w | w is in {0,1}* and w ends in 00}
  - {w | w is in {0,1}* and w contains at least two 0's}
  - {w | w is in {0,1}* and w contains at least one 0 and one 1}
  - Just about anything else (simple) you can think of

Construction of Turing Machine(CO5)

- $\{ 0^{2^n} \mid n \geq 0\}$
- Stages
  1. Sweep left to right across the tape. Crossing off every other 0
  2. If in stage 1, the tape conatined a single 0, accept
  3. If in stage 1, the tape contained more than a single 0, and the number of 0's was odd, reject
  4. Return the head to the left-hand end of the tape
  5. Goto stage 1.

Construction of Turing Machine(CO5)

What is important

- We seek to convince the student that Turing machines are a powerful tool to describe algorithms
- The full set of details is often too complex to describe completely, because the details do not add to our understanding.
- But, we could use our high level descriptions to complete the formal description if we desired.

Construction of Turing Machine(CO5)

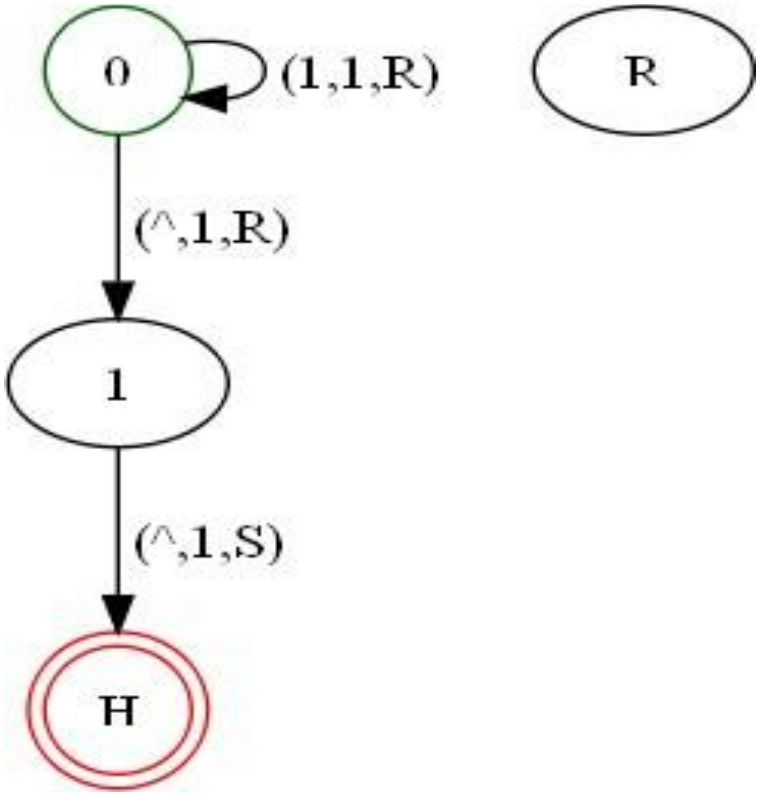
Turing machines with output

- A Turing machine can compute an output by leaving an answer on the tape when it halts.
- We must specify the form of the output when the machine halts.

Construction of Turing Machine(CO5)

Adding two to a number in unary

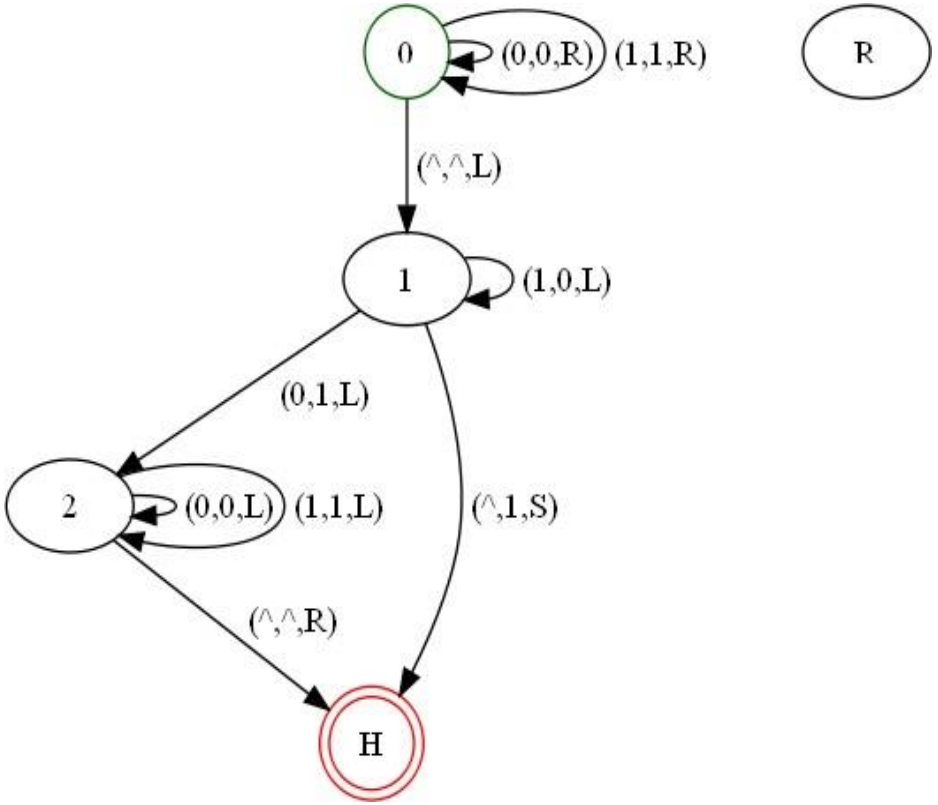
TM				
	Q	{0,	1,	H, R}
	Sigma	{1}		
	Gamma	{1,	^}	
	Delta	0 1	->	(1, R, 0)
		0 ^	->	(1, R, 1)
		1 ^	->	(1, S, H)
	q0	0		
	Accept	H		
	Reject	R		
	Blank	^		



Construction of Turing Machine(CO5)

Adding 1 to a Binary Number

TM Q {0, 1, 2, H, R}
Sigma {1, 0, ^}
Gamma {1, 0, ^}
Delta 0 0 -> (0, R, 0)
0 1 -> (1, R, 0)
0 ^ -> (^, L, 1)
1 0 -> (1, L, 2)
1 1 -> (0, L, 1)
1 ^ -> (1, S, H)
2 0 -> (0, L, 2)
2 1 -> (1, L, 2)
2 ^ -> (^, R, H)
q0 0 ^1011^
Accept H ^1010^
Reject R ^1000^
Blank ^ ^1100^



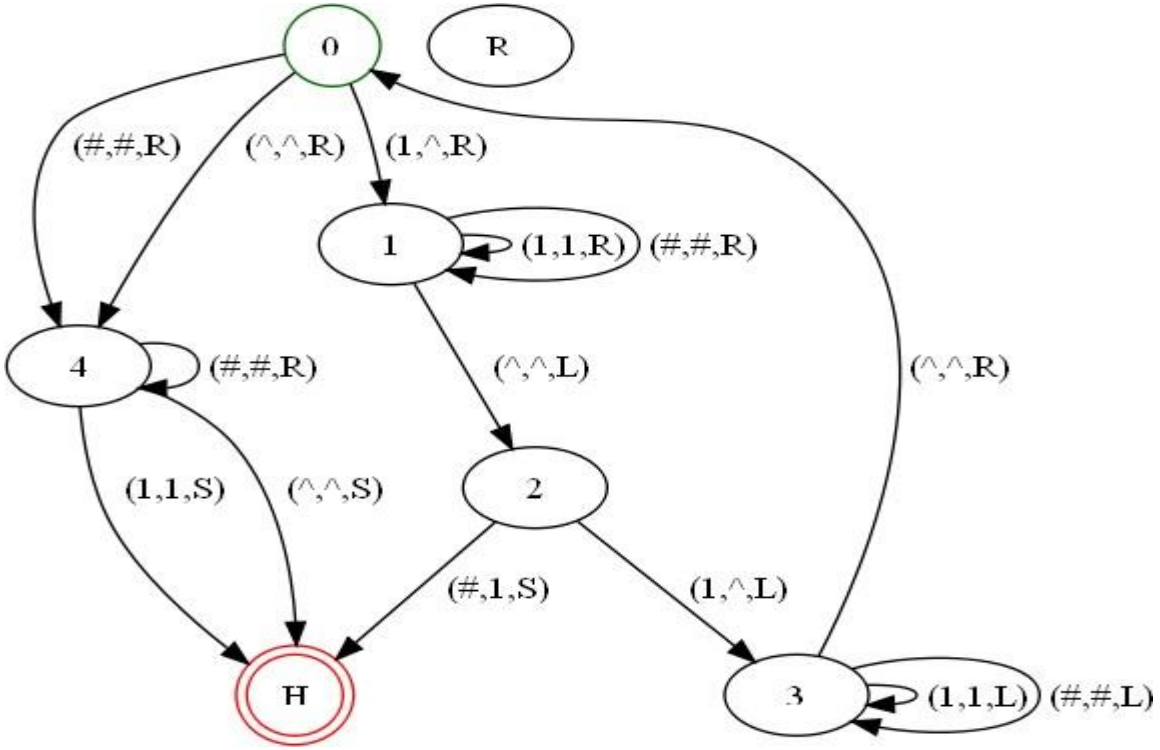
Construction of Turing Machine(CO5)

An equality Test

start = 0
blank = '^'
final = H

delta =
(0,1,^,R,1)
(0,^,^,R,4)
(0,#,#,R,4)
(1,1,1,R,1)
(1,^,^,L,2)
(1,#,#,R,1)
(2,1,^,L,3)
(2,#,1,S,H)
(3,1,1,L,3)
(3,^,^,R,0)
(3,#,#,L,3)
(4,1,1,S,H)
(4,^,^,S,H)
(4,#,#,R,4)

states = 0,1,2,3,4,H tape alphabet = 1,0,#,^ input alphabet = 1,0,#



Construction of Turing Machine(CO5)

- configurations for TM's are strings of the form □ q □ , where □, □ □ □* and q □ Q. (Assume that Q and □* are disjoint sets, guaranteeing unique parsing of configurations.)
- The string □ represents the tape contents to the left of the head.
- The string □ represents the non-blank tape contents to the right of the head, including the currently scanned cell.

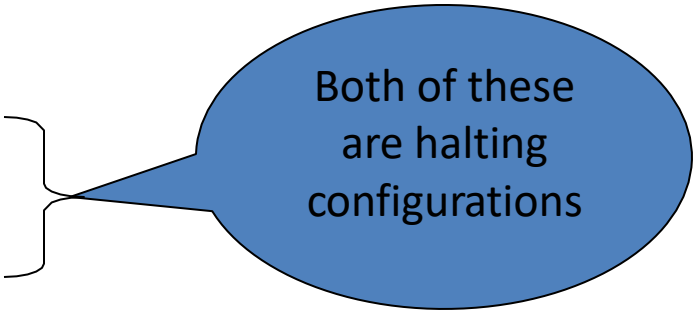
- Adding or deleting a few blank symbols at the beginning or end of an confiuration results in an equivalent configuartion. Both represent the same instant in the execution of a TM.

Construction of Turing Machine(CO5)

Sipser terminology

- Other authors call configurations instantaneous descriptions

- Starting Configuration
- Accepting Configuration
- Rejecting Configuration



Construction of Turing Machine(CO5)

- TM's transitions induce the relation |- between configurations.
- Let $\alpha = X_1 \ldots X_{i-1}\, q\, X_i \ldots X_k$ be an configuration.
- If $\delta(q,X_i)$ is undefined, then there are no configurations α' such that $\alpha \vdash \alpha'$.
- If $\delta(q,X_i)=(p,Y,R)$ then
  - $\alpha \vdash \alpha'$ holds for $\alpha' = X_1 \ldots X_{i-1}\, Y\, p\, X_{i+1} \ldots X_k$
- Similarly, if $\delta(q,X\_i)=(p,Y,L)$
- then $\alpha \vdash \alpha'$ holds for $\alpha' = X_1 \ldots pX_{i-1}YX_{i+1} \ldots X_k$
- When $\alpha \vdash \alpha'$ Sipser says: " α yields α' "

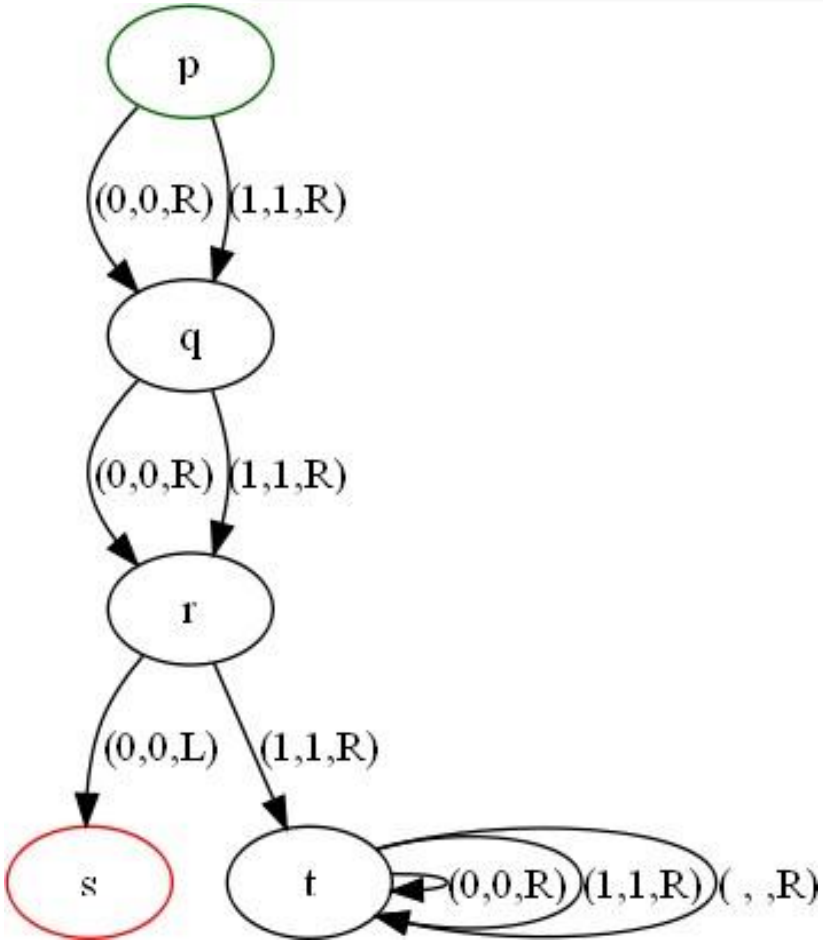
Construction of Turing Machine(CO5)

- If, in the first case, we have i=k, (that is we are at the end of the non-blank portion of the tape to the right) then we need to use the equivalent representation

- $\alpha = X_1 \ldots X_{k-1}\ q\ X_k\ B$

- for our formula to make sense. Similarly, we add a B to the beginning of α whenever necessary.

Construction of Turing Machine(CO5)



- Here is the sequence of configurations of our example machine, showing its execution with the given input 0101:
- p0101 |- 0q101 |-01r01 |- 0s101
- The machine halts, since there are no moves from the state s. When the input is 0111, the machine goes forever, as follows:
- p0111 |- 0q111 |- 01r11 |- 011t1 |- 0111t |- 0111Bt |- 0111BBt |-

Construction of Turing Machine(CO5)

- We define the language of the TM M to be the set L(M) of all strings w □ □*
- such that: Q_0 w |-* □ q_{accept} □
- for any □, □
- Languages accepted by TM's are call *recursively enumerable* (RE). Sipser calls this Turing-recognizable
- **Example**. For our example machine, we have L(M)= (0+1)(0+1)0(0+1)*
- If the machine recognizes some language, and also halts for all inputs. We say the language is Turing-decidable.

Construction of Turing Machine(CO5)

- Some text books define an alternative way of defining a language associated with a TM M. (But not Sipser, though the idea is still interesting).
- We denote it H(M), and it consists of strings that cause the TM to halt. Precisely, a string w □ □* belongs to H(M)
- iff $q_0 w \vdash^* \square p X \square$
- where □(p,X) is undefined.

- **Example**. For our example machine, we have
- H(M)= □ + 0 + 1 + (0+1)(0+1) + (0+1)(0+1)0(0+1)*

Construction of Turing Machine(CO5)

Equivalence of Acceptance by Final State and Halting

- How would we prove such an equivalence?

- 1. Construct a TM that accepts by Halting from an ordinary one.
- 2. Construct an ordinary TM from one that accepts by halting.

Recap

Construction method of Turing Machine.

Numerical of Turing Machine.

Daily Quiz

Q1: Design a Turing machine for 0n1n2n, n>0;

Q2:Desing a Turing machine for f(m,n)=m*n;

Universal Turing Machine (CO5)

Topic Objective: To understand the Universal Turing Machine.

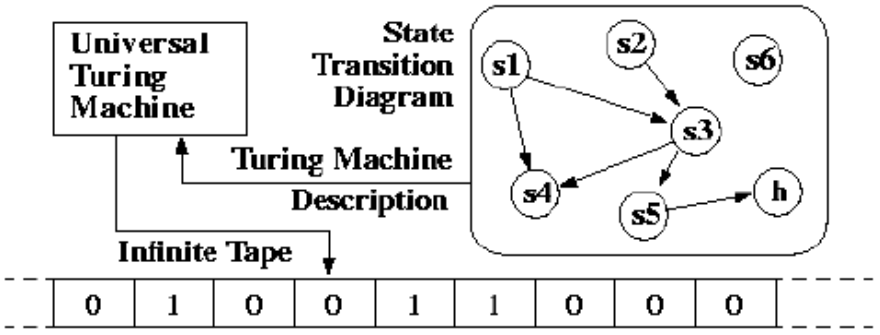
Recap: Till now we have studied about Turing Machine.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Universal Turing Machine (CO5)

Objective: To understand the Universal Turing Machine.

- A Turing Machine is the mathematical tool equivalent to a digital computer.
- It was suggested by the mathematician Turing in the 30s, and has been since then the most widely used model of computation in computability and complexity theory.
- The model consists of an input output relation that the machine computes.
- The input is given in binary form on the machine's tape, and the output consists of the contents of the tape when the machine halts.



Universal Turing Machine (CO5)

- What determines how the contents of the tape change is a finite state machine (or FSM, also called a finite automaton) inside the Turing Machine.
- The FSM is determined by the number of states it has, and the transitions between them.
- At every step, the current state and the character read on the tape determine the next state the FSM will be in, the character that the machine will output on the tape (possibly the one read, leaving the contents unchanged), and which direction the head moves in, left or right.
- The problem with Turing Machines is that a different one must be constructed for every new computation to be performed, for every input output relation.
- This is why we instroduce the notion of a universal turing machine (UTM), which along with the input on the tape, takes in the description of a machine M.
- The UTM can go on then to simulate M on the rest of the contents of the input tape. A universal turing machine can thus simulate any other machine.

Linear Bounded Automata (CO5)

Topic Objective: To understand the Linear Bounded Automata.

Recap: Till now we have studied about Turing Machine and Universal Turing Machine.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Linear Bounded Automata (CO5)

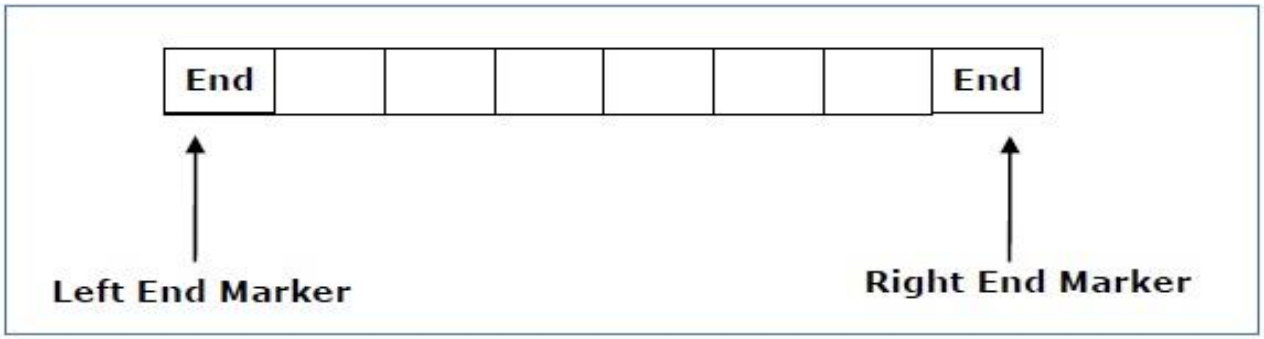
Objective: To understand the Linear Bounded Automata.

A linear bounded automaton is a multi-track non-deterministic Turing machine with a tape of some bounded finite length.

- **Length = function (Length of the initial input string, constant c)**

Here,

- **Memory information ≤ c × Input information**



- **Link: https://www.youtube.com/watch?v=kFH4X69L1JU**

Linear Bounded Automata(CO5)

A linear bounded automaton can be defined as an 8-tuple (Q, X, ∑, q_0, ML, MR, δ, F) where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- **∑** is the input alphabet
- **q_0** is the initial state
- **M_L** is the left end marker
- **M_R** is the right end marker where $M_R \neq M_L$
- **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

Church's Thesis (CO5)

Topic Objective: To understand the importance of Church Thesis in Theory of Computation.

Recap: Till now we have studied about Turing Machine, Universal Turing Machine and Linear Bounded Automata.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Church's Thesis (CO5)

Objective: To understand the importance of Church Thesis in Theory of Computation.

- In 1936, Alonzo Church created a method for defining functions called the λ-calculus. Within λ-calculus, he defined an encoding of the natural numbers called the Church numerals.
- Also in 1936, before learning of Church's work, Alan Turing created a theoretical model for machines, now called Turing machines, that could carry out calculations from inputs by manipulating symbols on a tape.
- A Turing machine is an abstract representation of a computing device.
- It is more like a computer hardware than a computer software.

- Link: https://www.youtube.com/watch?v=EEfNU8AoA-8

Church's Thesis (CO5)

- The Church-Turing thesis concerns an effective or mechanical method in logic and mathematics.
- A method, M, is called 'effective' or 'mechanical' just in case:
- M is set out in terms of a finite number of exact instructions (each instruction being expressed by means of a finite number of symbols);
- M will, if carried out without error, always produce the desired result in a finite number of steps;

Church's Thesis (CO5)

- M can (in practice or in principle) be carried out by a human being unaided by any machinery except for paper and pencil;
- M demands no insight or ingenuity on the part of the human being carrying it out.
- They gave an hypothesis which means proposing certain facts.
- The Church's hypothesis or Church's turing thesis can be stated as:
- The assumption that the intuitive notion of computable functions can be identified with partial recursive functions.
- This statement was first formulated by Alonzo Church in the 1930s and is usually referred to as Church's thesis, or the Church-Turing thesis.
- However, this hypothesis cannot be proved.

Decision Problems(CO5)

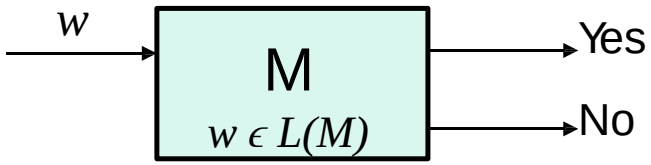
- Informally, a (decision) “problem” is a yes/no question about an infinite set of possible *instances*.
- Example 1: “Does graph G have a *Hamilton cycle* (cycle that touches each node exactly once)?
  - Each undirected graph is an instance of the “Hamilton-cycle problem.”
- Example 2: “Is graph G k-colorable ?
  - Each undirected graph, and value k, is an instance of the “graph coloring problem.”

Decision Problems(CO5)

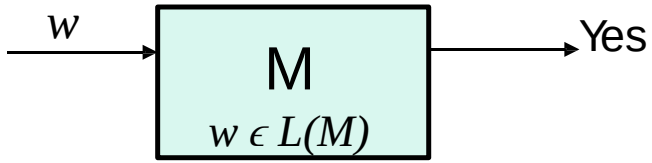
- Formally, a problem is a language.
- Each string encodes some instance.
- The string is in the language if and only if the answer to this instance of the problem is “yes.”

Decision Problems(CO5)

- **Recursive Language:** A language L is recursive language if there is a Turing machine that accepts the language and halts on all inputs.



- **Recursively Enumerable Language:** if there is a Turing machine that accepts the language by halting when the input string is in the language.
  - The machine may or may not halt if the string is not in the language.

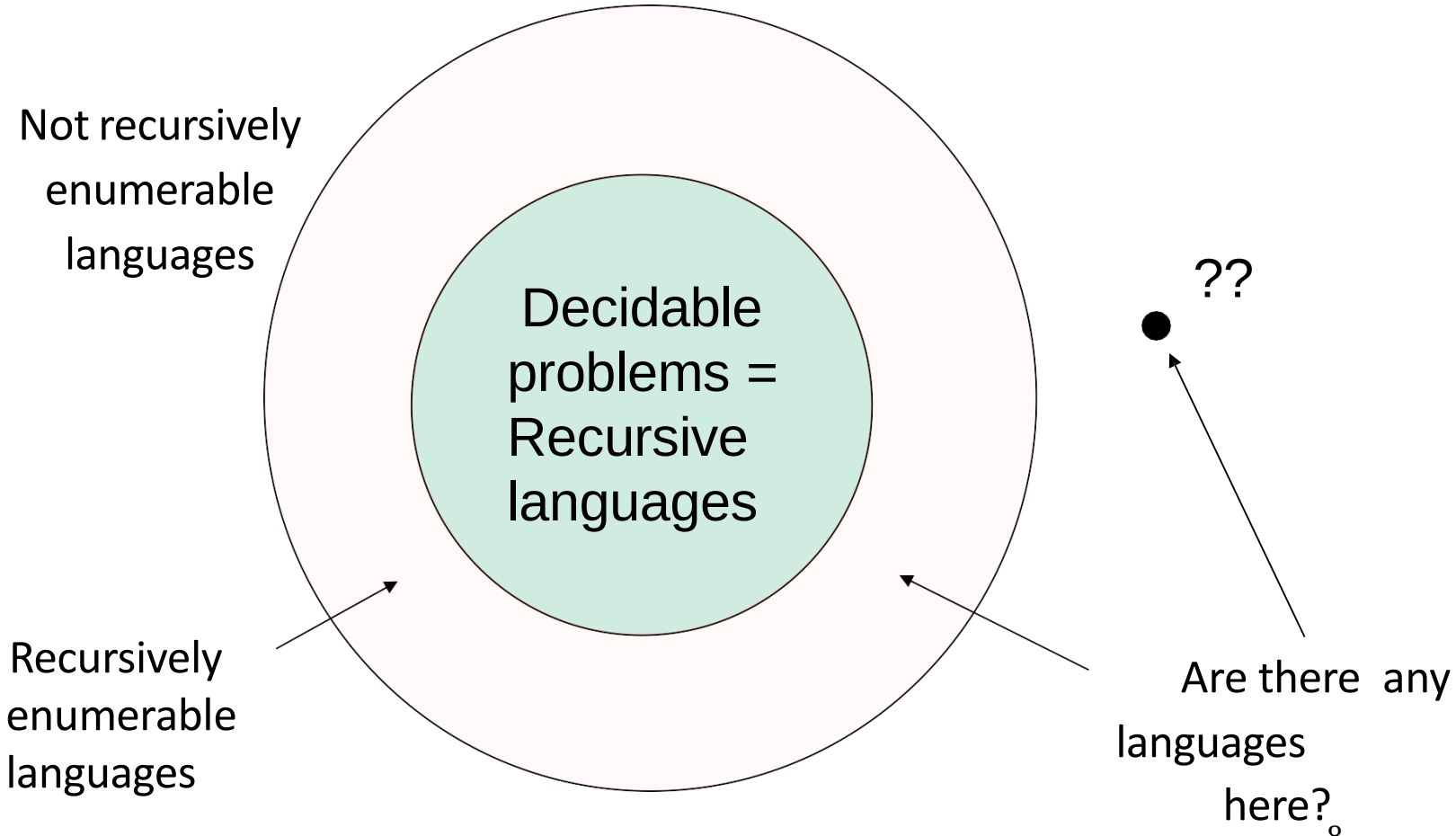


Decision Problems(CO5)

Decidable Problems

- A problem is *decidable*
- if there is an algorithm to answer it.
  - Recall: An "algorithm," formally, is a TM that halts on all inputs, accepted or not.
  - Put another way, "decidable problem" = "recursive language."
- Otherwise, the problem is *undecidable*.

Decision Problems(CO5)

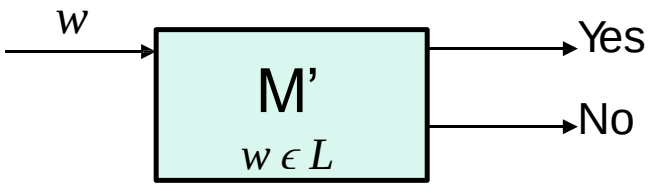


Closure Properties of Recursive and RE Languages(CO5)

- **Next topic is Decidability**
  - Review Lab notes on Math review – diagonalization etc.
- First let's look at closure properties of these classes of languages
- Both closed under union, concatenation, star, reversal, intersection, inverse homomorphism.
- Recursive closed under difference, complementation.
- RE closed under homomorphism.

Proving Closure Properties…methodology**(CO5)**

- ***Observe:*** *To prove the closure properties we have to construct a Turing machine, i.e., an algorithm (!!!), to accept the language*
  - *Construction shown using a flowchart & combining other "algorithms"*
    - *Getting more and more like programming!*
- To prove a language L (constructed from other recursive languages) is recursive, provide an algorithm described by a 'flowchart' below
  - To show it is RE, the machine halts only if w is in the language



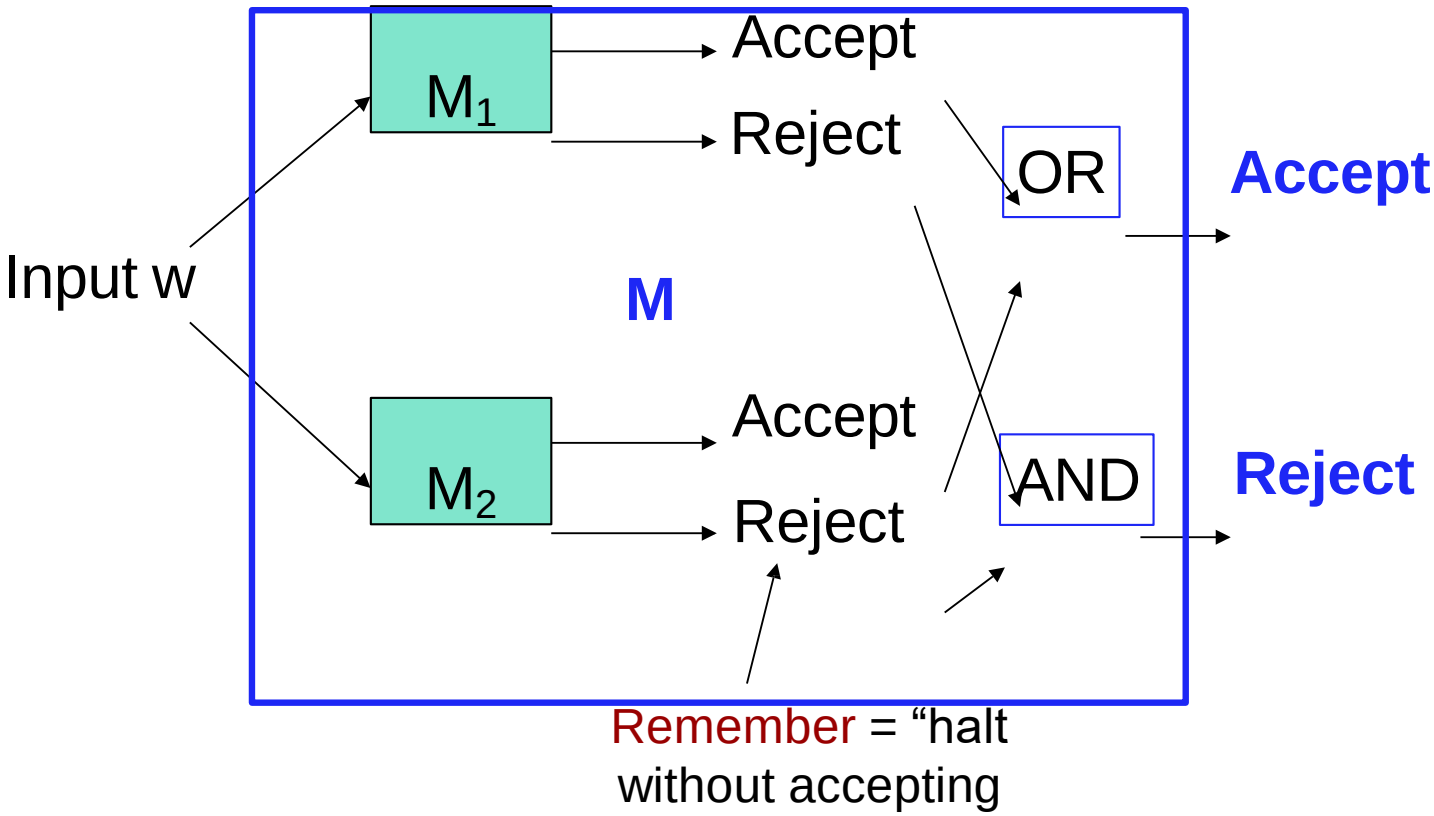
Closure under Union**(CO5)**

- Let $L_1 = L(M_1)$ and $L_2 = L(M_2)$.
- Assume M_1 and M_2 are single-semi-infinite-tape TM's.
- Construct 2-tape TM M to copy its input onto the second tape and simulate the two TM's M_1 and M_2 each on one of the two tapes, "in parallel."
- Recursive languages: If M_1 and M_2 are both algorithms, then M will always halt in both simulations.
- RE languages: accept if either accepts, but you may find both TM's run forever without halting or accepting.

Algorithm/Picture of Union**(CO5)**

M must halt on all inputs:

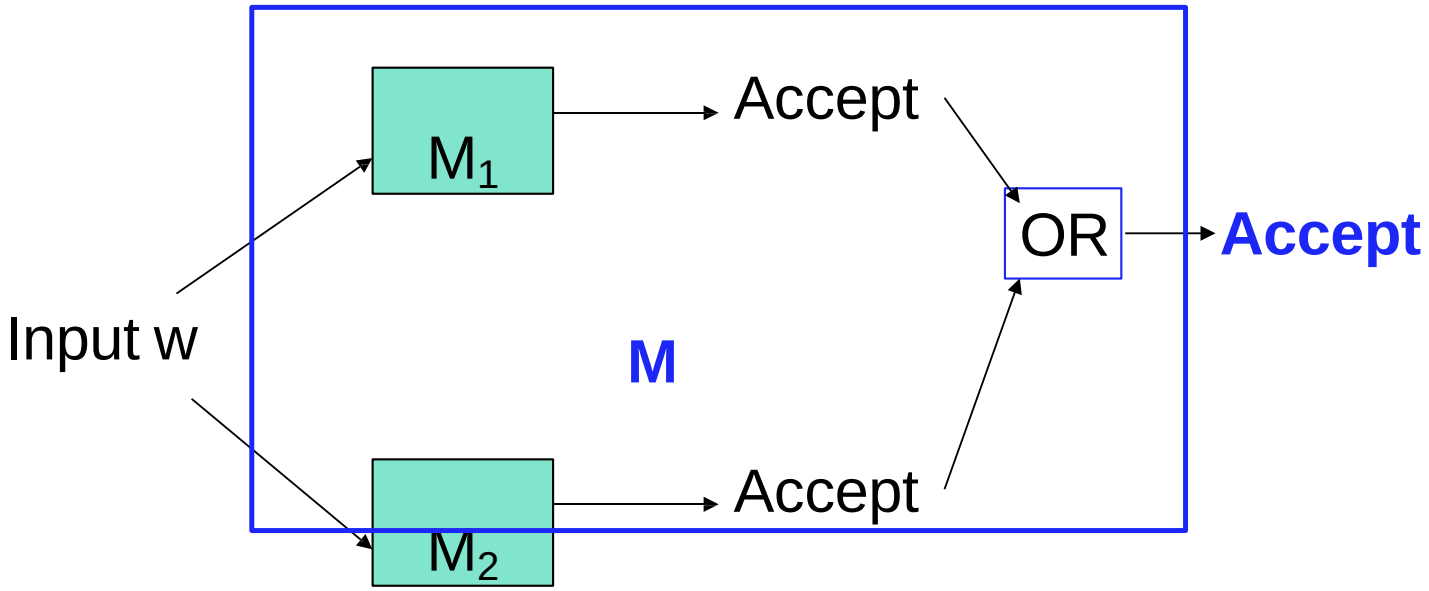
accepts if w is in either, rejects if w not in either



93

Picture of Union of RE Sets**(CO5)**

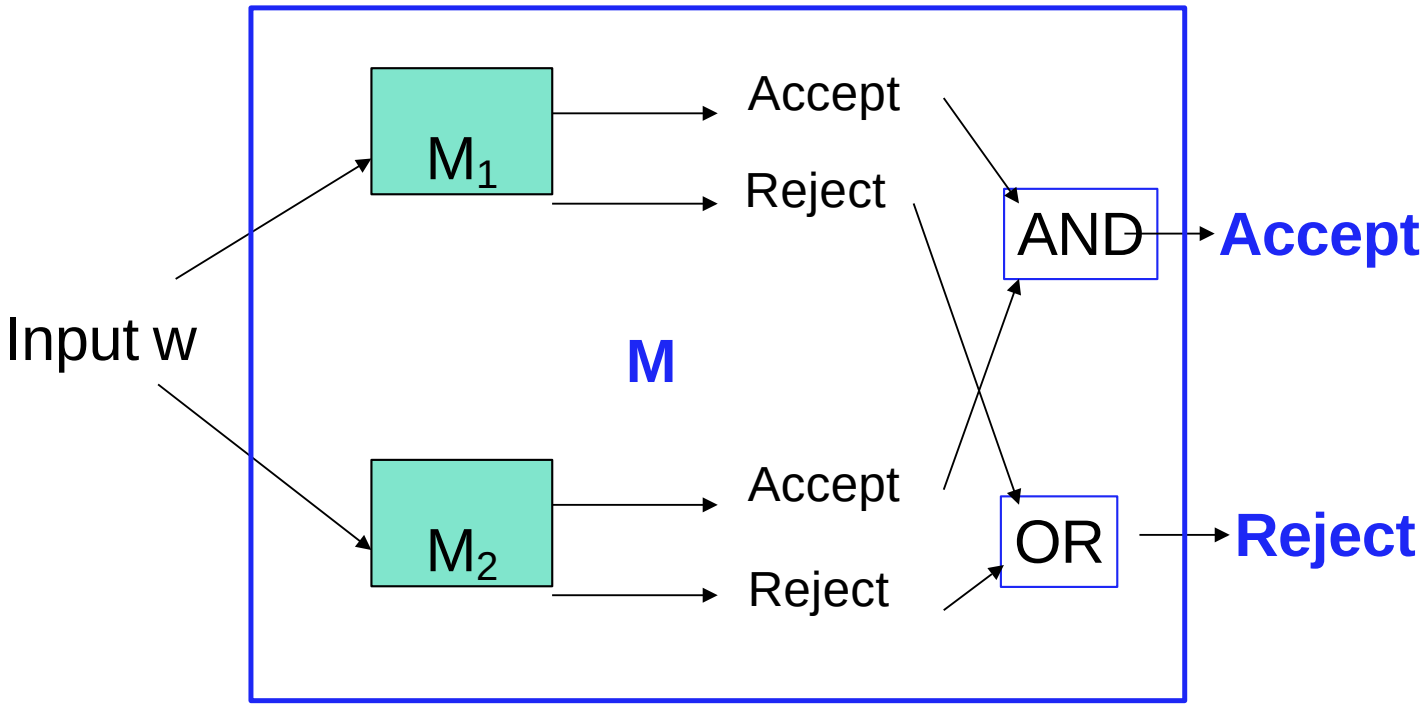
M must halt and accept if w is in either language,
else it may reject and halt or may not halt



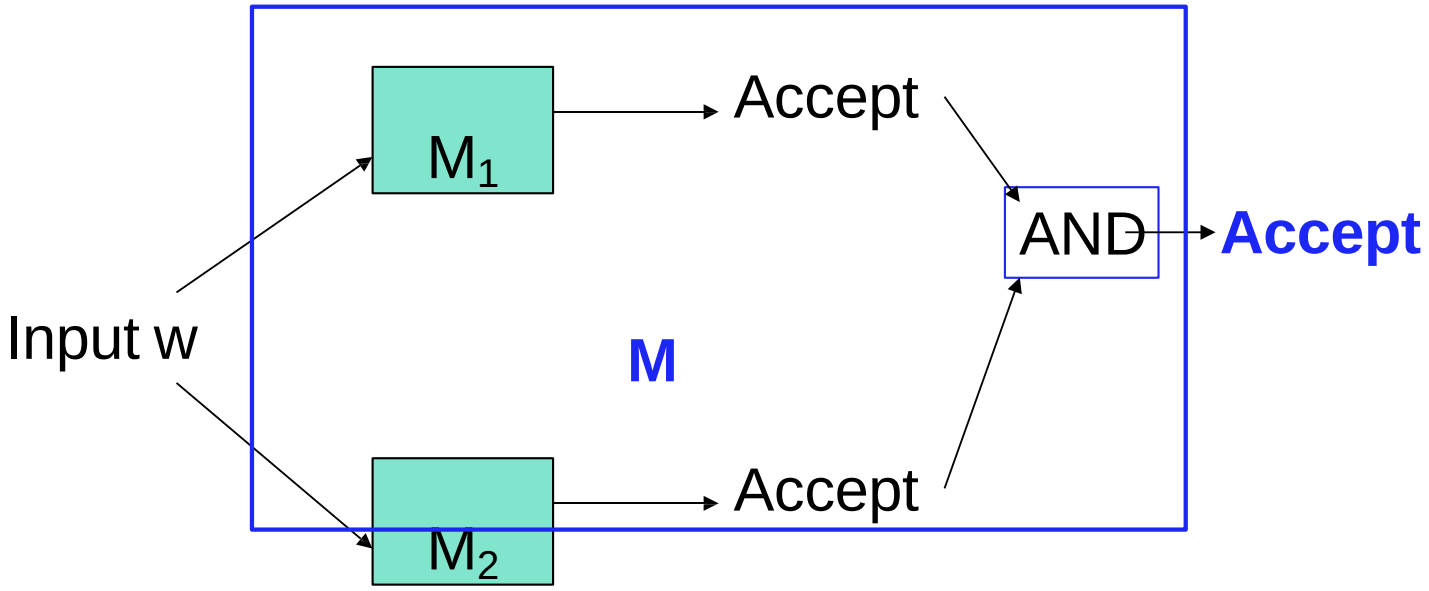
Closure under other set operations**(CO5)**

- Recursive Languages are closed under
  - Union, Intersection, Concatenation, Star Closure
  - Complementation. Set difference
  - Reversal
  - Inverse Homomorphism
- Recursively Enumerable (RE) languages are closed under
  - Union, Intersection, Concatenation, Star Closure
  - Reversal
  - Homomorphism
  - Inverse Homomorphism

Intersection of Recursive Sets – Same Idea**(CO5)**

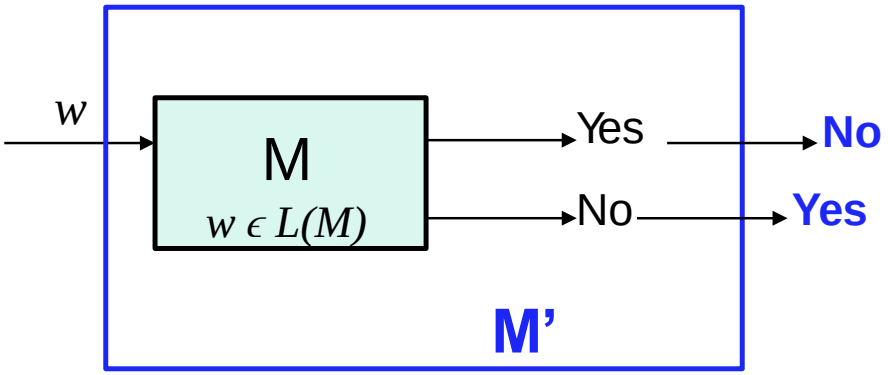


Intersection of RE Sets**(CO5)**



Observe: if w is in the intersection then both machines will accept and halt on w => This machine M will halt and accept w

Complement of Recursive Languages**(CO5)**



Star Closure**(CO5)**

- Same ideas work for each case.
- RE: guess many breaks, accept if M_1 accepts each piece.
- Recursive: systematically try all ways to break input into some number of pieces.

Reversal**(CO5)**

- Start by reversing the input.
- Then simulate TM for L to accept w if and only w^R is in L.
- Works for either Recursive or RE languages.

Recursive Enumerable (CO5)

Topic Objective: To understand the Recursive Enumerable Language

Recap: Till now we have studied about Turing Machine, Universal Turing Machine, Linear Bounded Automata and Church Thesis.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

.

Recursive Enumerable (CO5)

Objective: To understand the Recursive Enumerable Language.

- RE languages or type-0 languages are generated by type-0 grammars.
- An RE language can be accepted or recognized by Turing machine which means it will enter into final state for the strings of language and may or may not enter into rejecting state for the strings which are not part of the language.
- It means TM can loop forever for the strings which are not a part of the language.
- RE languages are also called as Turing recognizable languages.

- Link: https://www.youtube.com/watch?v=KJD_F-QGLmo

Halting Problem (CO5)

Topic Objective: To understand the Halting Problem in theory of computations.

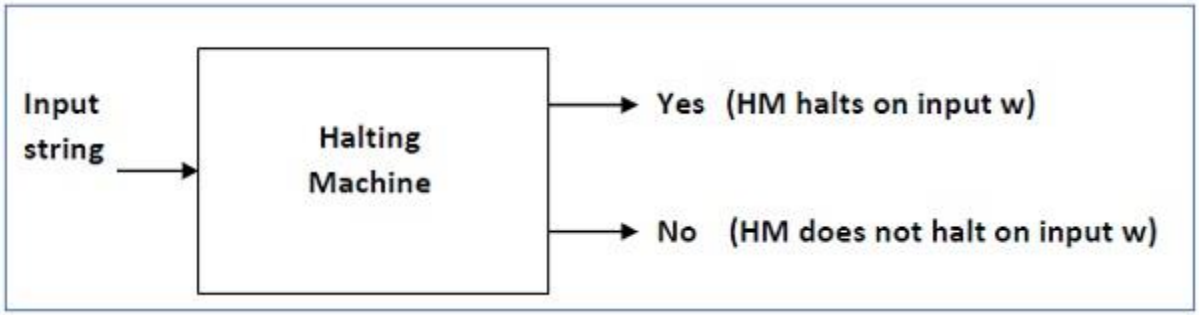
Recap: Till now we have studied about Turing Machine, Universal Turing Machine, Linear Bounded Automata, Church Thesis and Recursive Enumerable Language.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Halting Problem (CO5)

Objective: To understand the Halting Problem in theory of computations.

- **Input** – A Turing machine and an input string **w**.
- **Problem** – Does the Turing machine finish computing of the string **w** in a finite number of steps? The answer must be either yes or no.

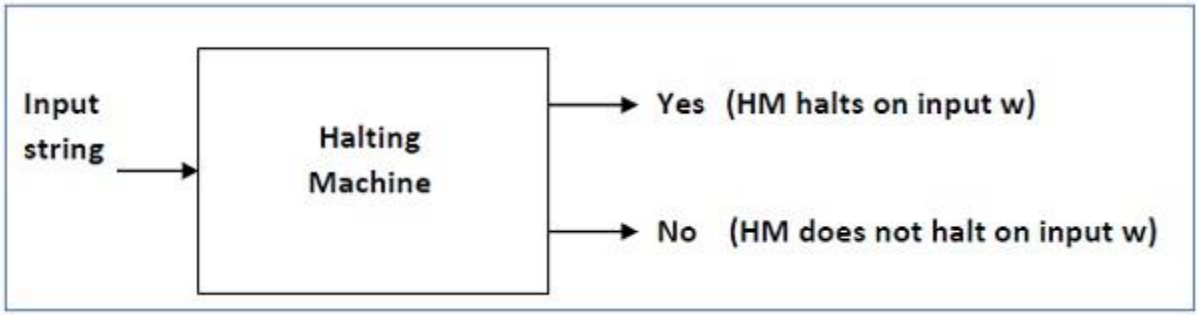


- Link for Further Explanation:
- https://www.youtube.com/watch?v=mx1mnyagfdM&list=PL4B084328ED81F3AF

Halting Problem (CO5)

Objective: To understand the Halting Problem in theory of computations.

- **Input** – A Turing machine and an input string **w**.
- **Problem** – Does the Turing machine finish computing of the string **w** in a finite number of steps? The answer must be either yes or no.



- Link for Further Explanation:
- https://www.youtube.com/watch?v=mx1mnyagfdM&list=PL4B084328ED81F3AF

Undecidable Problems (CO5)

Objective: To understand the **Undecidable Problems** in theory of computations.

Undecidable Problems

A problem is undecidable if there is no Turing machine which will always halt in finite amount of time to give answer as 'yes' or 'no'. An undecidable problem has no algorithm to determine the answer for a given input.

Examples

Ambiguity of context-free languages: Given a context-free language, there is no Turing machine which will always halt in finite amount of time and give answer whether language is ambiguous or not.

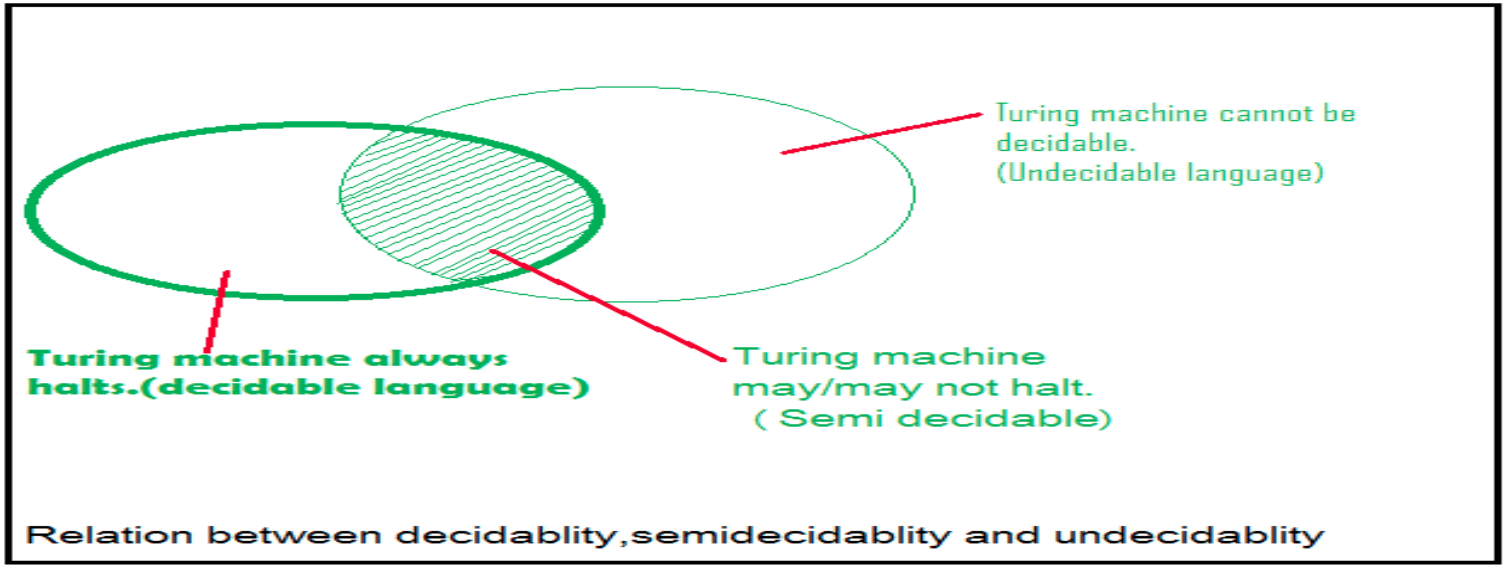
Equivalence of two context-free languages: Given two context-free languages, there is no Turing machine which will always halt in finite amount of time and give answer whether two context free languages are equal or not.

Everything or completeness of CFG: Given a CFG and input alphabet, whether CFG will generate all possible strings of input alphabet ($\sum^*$)is undecidable.

Regularity of CFL, CSL, REC and REC: Given a CFL, CSL, REC or REC, determining whether this language is regular is undecidable.

Undecidable Problems(CO5)

- Two popular undecidable problems are halting problem of TM and PCP (Post Correspondence Problem). Semi-decidable Problems
- A semi-decidable problem is subset of undecidable problems for which Turing machine will always halt in finite amount of time for answer as 'yes' and may or may not halt for answer as 'no'.
- Relationship between semi-decidable and decidable problem has been shown in Figure 1 as:



Relation between decidablity,semidecidablity and undecidablity

Undecidable Problems(CO5)

- **Rice's Theorem**
- Every non-trivial (answer is not known) problem on Recursive Enumerable languages is undecidable.e.g.; If a language is Recursive Enumerable, its complement will be recursive enumerable or not is undecidable.

- **Reducibility and Undecidability**
- Language A is reducible to language B (represented as $A \leq B$) if there exists a function f which will convert strings in A to strings in B as:

- $w \varepsilon A <=> f(w) \varepsilon B$
- Theorem 1: If $A \leq B$ and B is decidable then A is also decidable.
- Theorem 2: If $A \leq B$ and A is undecidable then B is also undecidable.

Undecidable Problems(CO5)

- Theorem 1: If A≤B and B is decidable then A is also decidable.
- Theorem 2: If A≤B and A is undecidable then B is also undecidable.
- Question: Which of the following is/are undecidable?
- G is a CFG. Is L(G)=ϕ?
- G is a CFG. Is L(G)=∑*?
- M is a Turing machine. Is L(M) regular?
- A is a DFA and N is an NFA. Is L(A)=L(N)?
- A. 3 only
- B. 3 and 4 only
- C. 1, 2 and 3 only
- D. 2 and 3 only

Undecidable Problems(CO5)

- **Explanation:**
- Option 1 is whether a CFG is empty or not, this problem is decidable.
- Option 2 is whether a CFG will generate all possible strings (everything or completeness of CFG), this problem is undecidable.
- Option 3 is whether language generated by TM is regular is undecidable.
- Option 4 is whether language generated by DFA and NFA are same is decidable. So option D is correct.
- Question: Which of the following problems are decidable?

Undecidable Problems(CO5)

- Does a given program ever produce an output?
- If L is context free language then L' is also context free?
- If L is regular language then L' is also regular?
- If L is recursive language then L' is also recursive?
- A. 1,2,3,4
- B. 1,2
- C. 2,3,4
- D. 3,4

Undecidable Problems(CO5)

- **Explanation:**
- As regular and recursive languages are closed under complementation, option 3 and 4 are decidable problems.
- Context free languages are not closed under complementation, option 2 is undecidable.
- Option 1 is also undecidable as there is no TM to determine whether a given program will produce an output. So, option D is correct.
- Question: Consider three decision problems P1, P2 and P3. It is known that P1 is decidable and P2 is undecidable. Which one of the following is TRUE?

- A. P3 is undecidable if P2 is reducible to P3
- B. P3 is decidable if P3 is reducible to P2's complement
- C. P3 is undecidable if P3 is reducible to P2
- D. P3 is decidable if P1 is reducible to P3

Undecidable Problems(CO5)

- **Explanation:**
- Option A says P2≤P3. According to theorem 2 discussed, if P2 is undecidable then P3 is undecidable. It is given that P2 is undecidable, so P3 will also be undecidable. So option **(A) is correct**.
- Option C says P3≤P2. According to theorem 2 discussed, if P3 is undecidable then P2 is undecidable. But it is not given in question about undecidability of P3. So option **(C) is not correct.**
- Option D says P1≤P3. According to theorem 1 discussed, if P3 is decidable then P1 is also decidable. But it is not given in question about decidability of P3. So option **(D) is not correct**.
- Option (B) says P3≤P2'. According to theorem 2 discussed, if P3 is undecidable then P2' is undecidable. But it is not given in question about undecidability of P3. So option **(B) is not correct**.

Post Correspondence Problem (CO5)

Topic Objective: To understand the importance of post correspondence problem.

Recap: Till now we have studied about Turing Machine, Universal Turing Machine, Linear Bounded Automata, Church Thesis, Recursive Enumerable Language and Halting Problem.

Prerequisite: Basic review of model to perform different operations like addition, subtraction, multiplication.

Post Correspondence Problem (CO5)

Objective: To understand the importance of post correspondence problem.

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet ∑ is stated as follows −

- Given the following two lists, **M** and **N** of non-empty strings over ∑
- M = (x_1, x_2, x_3,……, x_n)
- N = (y_1, y_2, y_3,……, y_n)
- We can say that there is a Post Correspondence Solution, if for some i_1,i_2,…… i_k, where $1 \leq i_j \leq n$, the condition x_{i1} …….x_{ik} = y_{i1} …….y_{ik} satisfies.

- Link for further explanation :
- https://www.youtube.com/watch?v=YSr5zmVqZLI

Post Correspondence Problem(CO5)

Problem:

Find whether the lists M = (abb, aa, aaa) and N = (bba, aaa, aa) have a Post Correspondence Solution?

Solution:

	x_1	x_2	x_3
M	Abb	aa	aaa
N	Bba	aaa	aa

- Here,
- **$x_2x_1x_3$ = 'aaabbaaa'**
- and **$y_2y_1y_3$ = 'aaabbaaa'**
- We can see that
- **$x_2x_1x_3 = y_2y_1y_3$**
- Hence, the solution is **i = 2, j = 1, and k = 3.**

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details

Youtube/other Video Links

- https://www.youtube.com/watch?v=IhyEGNn-7Uo
- https://www.youtube.com/watch?v=BR6fHjKFqa0
- https://www.youtube.com/watch?v=vhSMp91tS6k
- https://www.youtube.com/watch?v=mPec64RUCsk
- https://www.youtube.com/watch?v=moPtwq_cVH8

Daily Quiz

1. Design a TM to compute the function $f(n)=n^2$
2. Explain Church's Thesis.
3. Define Post Correspondence Problem and Modified Post Correspondence Problem.
4. Design a TM to accept the language " The set of strings with an equal number of 0's and 1's.
5. Design a right shift TM over an alphabet{0,1}
6. Write short note on Halting Problem.

Daily Quiz (Continued)

7. Define Basic model of Turing Machine.
8. Explain the techniques for Turing Machine construction.
9. Explain Church's Thesis.
10. Design a TM to compute the function $f(n)=n^2$
11. Define Post Correspondence Problem and Modified Post Correspondence Problem.
12. Write short note on Universal Turing Machine.
13. When a language is said to be recursive or recursively enumerable.
14. Write short note on Halting Problem.
15. Design a right shift TM over an alphabet{0,1}.

Weekly Assignment

- https://noidainstituteofengtech-my.sharepoint.com/:b:/g/personal/dileepkumar_it_niet_co_in/EYG37gFau8JMjbWTNty6U8IBmT2dZAlKvKq4_AncVizN-w?e=xfyJXk

MCQ s

1. Find the odd one :

a. Multiple track
b. Subroutines
c. Recursion
d. Shifting over

2. Which operation is not a part of TM?

a. Enter Accepting State
b. Enter Non Accepting state
c. Enter infinite loop and never halts
d. None of the above

MCQ s (Continued)

3. Which of the statement is not true for TM?

a. Computers of functions on non negative numbers
b. Language Recognition
c. Generating devices
d. None

4. A finite control of TM is used in order to hold a finite amount of data. Say True or False:

a. TRUE
b. FALSE
c. May Be
d. Cant Say

MCQ s (Continued)

5. Find the odd one :

a. Multiple track

b. Subroutines

c. Recursion

d. Shifting over

6. Which operation is not a part of TM?

a. Enter Accepting State

b. Enter Non Accepting state

c. Enter infinite loop and never halts

d. None of the above

MCQ s (Continued)

7. Which of the statement is not true for TM?

a. Computers of functions on non negative numbers

b. Language Recognition

c. Generating devices

d. None

8. X is a simple mathematical model of a computer. X has unrestricted and unlimited memory. X is a FA with R/W head. X can have an infinite tape divided into cells, each cell holding one symbol.
Name X?

a. NDFA

b. PDA

c. TM

d. None of the above

Old Question Papers

- *https://noidainstituteofengtech-my.sharepoint.com/:f:/g/personal/dileepkumar_it_niet_co_in/Ej8ZcmLFiTVMkjGTa0CJypMBIy5NJeLQzVtfB4fGN_OBpg?e=ACFSl5*

Expected Questions for University Exam

1. Design a TM to reverse a string over an alphabet {0,1}.
2. Design a TM to check whether a string over {a,b}contains equal number of a's and b's.
3. Design a TM that replace every occurrence of abb by baa.
4. Design a TM for addition of Unary numbers.
5. Design a TM for subtraction of a Unary numbers.

Summary

- Turing machines use an infinite tape divided into a number of rectangular cells and a tape head that can read and write symbols and can move in both the directions left and right.
- The input for a transition function of a TM are current state and current tape symbol.
- The output of the transition function are the state in which TM has to be after reading the current tape symbol.

References

1. Aho, Hopcroft and Ullman, *The Design and Analysis of Computer Algorithms,* Addison Wesley.
2. Aho, Sethi, Ullman, *Compilers Principles, Techniques and Tools,* Pearson Education, 2003.
3. Hopcroft and Ullman, *Introduction to Automata Theory, Languages and Computation,* Addison Wesley.
4. Kohavi, ZVI, *Switching And Finite Automata Theory*, Tata McGraw-Hill, 2006.
5. Lewis and Papadimitriou, *Elements of the Theory of Computation,* Prentice-Hall.
6. Martin, *Introduction to Languages and the Theory of Computation,* McGraw-Hill, 2nd edition,1996.
7. Mishra, KLP, Chandrasekaran, N. *Theory of Computer Science,* (Automata, Languages and Computation) PHI, 2002.

Thank You