BIPIN TRIPATHI KUMAON INSTITUTE OF TECHNOLOGY, DWARAHAT

<— TAFL ASSIGNMENT —>

SESSION : 2019-2020
SEMESTER : EVEN (4th)
SUBJECT : THEORY OF AUTOMATA AND FORMAL LANGUAGES [TCS 403]

SUBMITTED TO:-

DR. VISHAL KUMAR
[PROFESSOR,
C.S.E. DEPARTMENT]

SUBMITTED BY:-

NAME. AYUSH CHAUHAN
ROLL NO. 18018010013
BRANCH . C.S.E.
YEAR . IInd Year
SEMESTER . 4th (EVEN)

Sub. TAFL
Name. Ayush Chauhan
Roll No.. 18018010101013

TAFL ASSIGNMENT

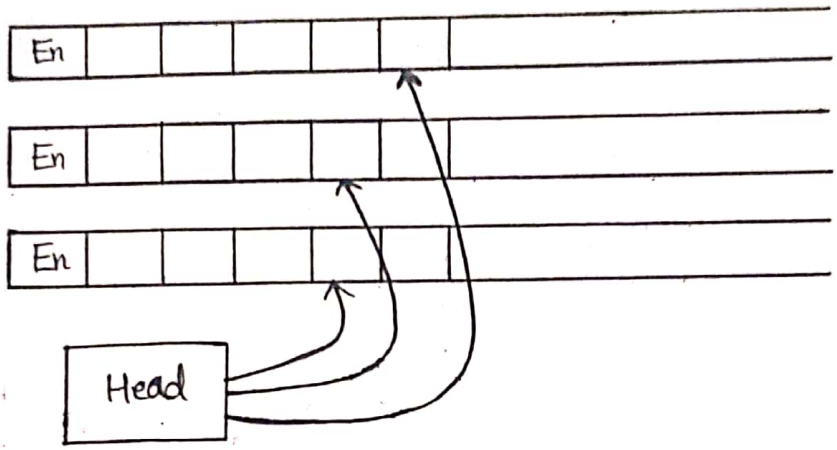
Q>1> Explain Church's thesis.

Ans. The Church-Turing thesis states the equivalence between the mathematical concepts of algorithm or computation and Turing-Machine. It asserts that if some calculation is effectively carried out by an algorithm, then there exists a Turing machines which will compute that calculation. The notion of algorithm, a step-by-step procedure or a defined method to peform calculations has been used informally and intuitively in mathematics for centuries. However, attempts to formalize the concept only begun in the beginning of the 20th century. Three major attempts were made: λ-calculus, recursive functions and Turing Machines. These three formal concepts were proved to be equivalent; all three define the same class of functions. Hence, Church-Turing thesis also states that λ-calculus and recursive functions also correspond to the concept of computability. Since the thesis aims to capture an intuitive concept, namely the notion of computation, it cannot be formally proven. However, it has gain widely acceptance in the mathematical and philosophical community.

Q>2> Write a short note on multi tape Turing machine.

Ans. Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads.

Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



A Multi-tape Turing machine can be formally described as a 6-tuple $(Q, X, B, \delta, q_0, F)$ where –

Q is a finite set of states
X is the tape alphabet
B is the blank symbol
δ is a relation on states and symbols where
$\delta : Q \times X^k \rightarrow Q \times (X \times \{Left\_shift, Right\_shift, No\_shift\})^k$
where there is k number of tapes
q_0 is the initial state
F is the set of final states

Every Multi-tape Turing machine has an equivalent single-tape Turing machine.

Q>3> Define concept and working of a Push down automata.

Ans. Pushdown Automata is a finite automata with extra

memory called stack which helps Pushdown automata to recognize Context Free Languages.
A Pushdown Automata (PDA) can be defined as:

- Q is the set of states
- Σ is the set of input symbols
- Γ is the set of pushdown symbols (which can be pushed and popped from stack).
- q_0 is the initial state
- Z is the initial pushdown symbol (which is initially present in stack)
- F is the set of final states.
- δ is a transition function which maps $Q \times \{\Sigma \cup \epsilon\} \times \Gamma$ into $Q \times \Gamma^*$. In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.

Instantaneous Description (ID) is an informal notation of how a PDA "computes" an input string and make a decision that string is accepted or rejected.
A ID is a triple (q, w, α), where:
q is the current state.
w is the remaining input.
α is the stack contents, top at the left.

Turnstile notation
$\vdash$ sign is called a "turnstile notation" and represents one move.
$\vdash^*$ sign represents a sequence of moves.

Working of PDA:-
A PDA is an abstract machine, it's not an actual device (although any one of numerous actual

devices can implement the abstraction). It "recognizes" phrases of some language by being in an "accept" state when the input has been exhausted. Alternatively, acceptance can be defined as the stack being empty when the input is exhausted.

The current state of the PDA is a set of rules that determines what it should do based on the symbol under the read head and the symbol at the top of the stack. The things it can do are pop the top symbol of the stack, push some symbol onto the stack, advance the read head, and change to another state. Once, it has transitioned to another state, the process is repeated until the input is exhausted. That's all there is to "processing" the tape.

Q>4> Describe Halting Problem of Turing Machines.

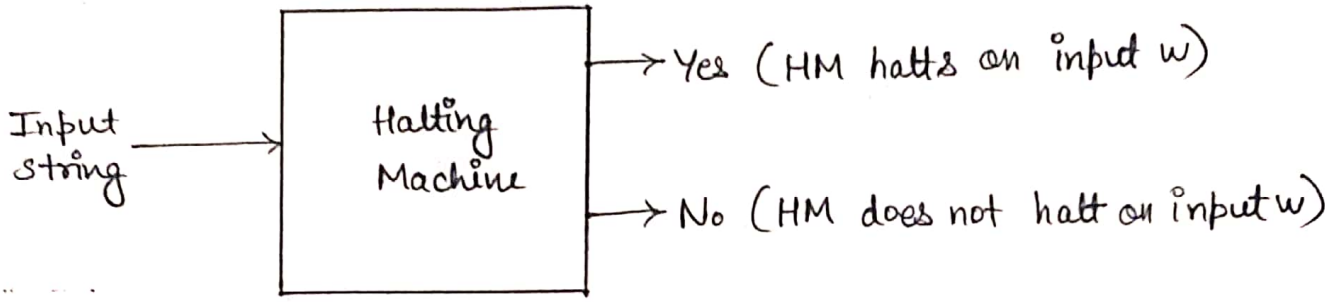
Ans. Turing Machine Halting Problem:—

Input: A Turing machine and an input string w.

Problem: Whether the Turing machine complete the calculation of the string w in a finite number of steps? Say yes or no.

Proof: Initially you should assume that a Turing machine is used to solve this problem and it will show that it is contradicting itself. This kind of Turing machine is called as a Halting machine which provides a 'yes' or 'no' in a finite amount of time.

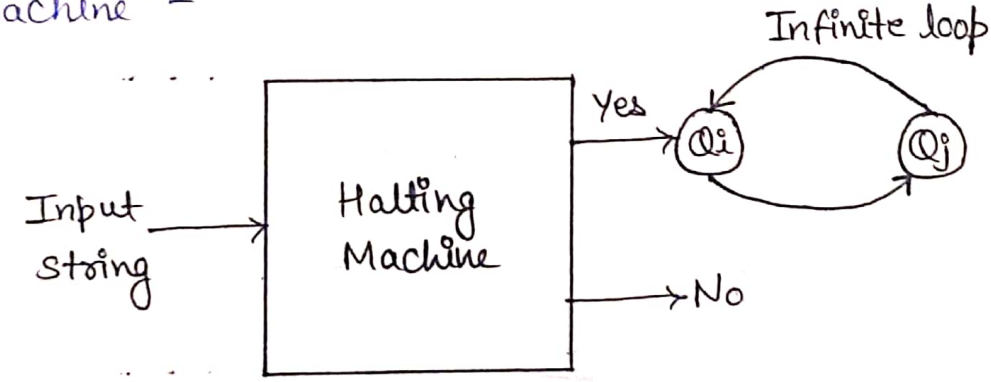
In case, if the halting machine completes in a finite amount of time, the output will be yes, otherwise it will be 'no'. Below mentioned block diagram of a Halting machine is -



Now we will design an inverted halting machine (HM) as -

- If H returns YES, then loop forever.
- If H returns NO, then halt.

Below mentioned block diagram of an 'Inverted halting machine' -



After that a machine (HM)2 which input itself is constructed as follows -

- If (HM)2 halts on input, loop forever.
- Else, halt.

After this we got a contradiction then the halting problem is undecidable.

Q>5> Describe Myhill Nerode Theorem.

Ans. > The Myhill-Nerode theorem is an important characterization of regular languages, and it also has many practical implications.

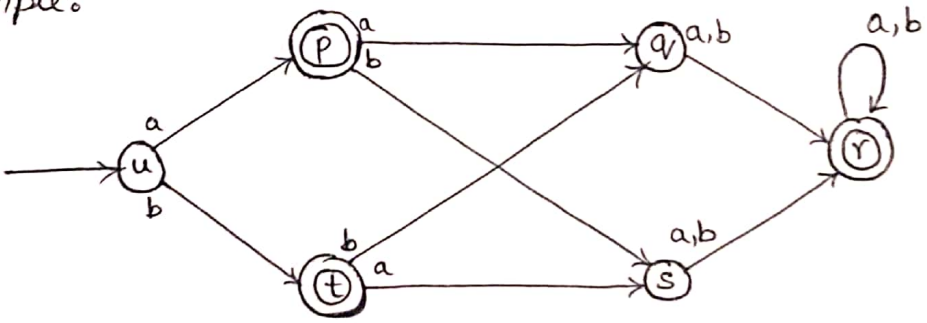
> One consequence of the theorem is an algorithm for minimizing DFAs which is a vital step in automata theory.

> Theorem:

The Myhill Nerode Theorem states that for a language L such that $L \subseteq \Sigma^*$, the following statements hold good:-

- There is a DFA that accepts L (L is regular).
- There is a right invariant equivalence relation ~ of finite index such L is a union of some of the equivalence classes of ~.
- ~L is of finite index.

Example:



Step 1: Consider every final-nonfinal state pair and tick it working only on the lower triangular part of the table.

States	u	p	t	q	s	r
u						
p	✓					
t	✓					
q		✓	✓			
s		✓	✓			
r	✓			✓	✓	

Step 2: Consider all the un-ticked areas of step1. For an input (either a or b) for each un-ticked state, see the intermediate state. For the area (r,t):

(r,a) => {r} and (t,a) => s

So, here the intermediate state is 's'

Now, check if {r,s} is ticked in step1.

If yes, tick {r,t} as well.

Similarly, {q,u} and {r,q} are also ticked.

Step 3: Continue step 2 until all states have been processed. Once no more can be ticked, algorithm terminates.

Hence, here {s,u} is also ticked.

Final table now becomes:

States	u	p	t	q	s	r
u						
p	✓					
t	✓					
q	✓	✓	✓			
s	✓	✓	✓			
r	✓	✓	✓	✓	✓	

Step 4: Check the spaces which are still un-ticked and such states can be merged together.

In the final minimized DFA, q-s are the new states and p-t are the new states.

Q>6- Explain recursive function theory.

Ans. By introducing one more operation on functions, we define the class of recursive functions.

Let $g(x_1, x_2, x_3 - x_n + y)$ be a total function over N. g is a regular function over N, g is a regular function if there exist some natural no. y_0 such that $g(x_1, x_2 - x_n + y_0) = 0$ for all the values $x_1, x_2 - x_n$ in N.

For instance $g(x,y) = \min(x,y)$ is a regular function since $g(x,0) = 0 \ \forall x$ in N. But $f(x,y) = |x-y|$ is not regular since $f(x,y) = 0$ only, when $x = y$ so we can't find y such that $f(x,y) = 0 \ \forall x$ in N.

A function $f(x_1, x_2 - x_n)$ over N is defined from a total function $g(x_1, x_2 - x_n, y)$ by minimization if:

a) $f(x_1, x_2 - x_n)$ is the least value of y's such that $g(x_1, x_2 - x_n, y) = 0$ if it exists.
The least value is denoted by $\mu y(x_1, x_2, x_3 - x_n, y) = 0$

b) $f(x_1, x_2, x_n)$ is undefined if there is no y such that $g(x_1, x_2 - x_n, y) = 0$.

A function is recursive if it can be obtained from the initial functions by a finite no. of set of applications of composition, recursion and minimization over regular functions.

Q>7> Explain Post correspondence problem.

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

Given the following two lists, M and N of non-empty strings over Σ-

$M = (x_1, x_2, x_3, \ldots, x_n)$
$N = (y_1, y_2, y_3, \ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2, \ldots, i_k$, where $1 \le i_j \le n$, the condition $x_{i_1} \ldots x_{i_k} = y_{i_1} \ldots y_{i_k}$ satisfies.

Ex1:

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

Sol:

	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 $k=3$.

Ex2:

Find whether the lists $M = (ab, bab, bbaaa)$ and $N = (a, ba, bab)$ have a Post Coorespondence Solution?

Sol:

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because-
$|x_2x_1x_3| \neq |y_2y_1y_3|$ (Lengths are not same)
Hence, it can be said that this Post Correspondence Problem is undecidable.

Q>8> Prove that: There exists a language over Σ that is not recursively enumerable.

Ans. We will use an indirect argument to show that the language over Σ that is not recursively enumerable.

Lemma 1: If L and its complement are recursively enumerable, then L is recursive.

Proof: Let M be a TM that enumerates L and M' be a TM that enumerates the complement of L. Given any string S, we can decide whether S is in L as follows. Begin running M and M', interleaving their executions so that each one eventually gets an arbitrary amount of runtime. If S in L, M will eventually list it at which point we know S in L and we halt-accept. If S is not in L and we halt-reject. Thus, for any S, we can halt accept if S is in L or halt-reject otherwise. Therefore, L and its complement are recursive.

Lemma 2: The language of encodings of TM that accepts something is recursively enumerable.

Proof: The set of all TM encodings is countable, and so is the set of all possible tape inputs. Thus, the set (M,S) of pairs of machines and inputs is countable. We may therefore, assume some ordering of these pairs $P_1, P_2 \rightarrow P_k$, for each pair = (M,S), begin executing machine M on input S, interleaving the executions of pairs $P_1, P_2 - P_k -$ so each eventually gets an arbitrary amount of runtime. If P^k enters the halt-accept state, we

may immediately list M as TM that accepts something (namely, the corresponding s), and we can even terminate all other running instances checking the same M (and forego starting any new ones). Any machine M that accepts some input will eventually be started and and will eventually halt-accept on an input, so all machines are eventually enumerated.

Lemma 3: The language of encodings of TM that accept nothing is not recursive.

Proof: This is a direct result of 'Rice's Theorem'. The property "accepts nothing" is a semantic property of the languages itself and is true for some, but not all, languages; ∴ no TM can decide whether another TM accepts a language with the property or not.

To Prove: Their exist language over Σ that is not recursively enumerable.

Proof: Assume this language is recursively enumerable we have already proven in Lemma 2 that its complement is recursively enumerable. By Lemma 1, then, both languages are recursive. However, Lemma's proves that the languages is not recursive. This is contradiction. The only assumption was that language is recursively enumerable, so that assumption must have been false; so the language is not recursively enumerable.

Q>9> Write short notes on:
i. Universal Turing Machine ii. Decidable languages.

Ans. i. Universal Turing Machine:

A Universal Turing Machine (UTM) is a Turing machine that can simulate an arbitrary Turing machine on arbitrary input. The universal machine essentially achieves this by reading both the description of the machine to be simulated as well as the input there of from its own tape.

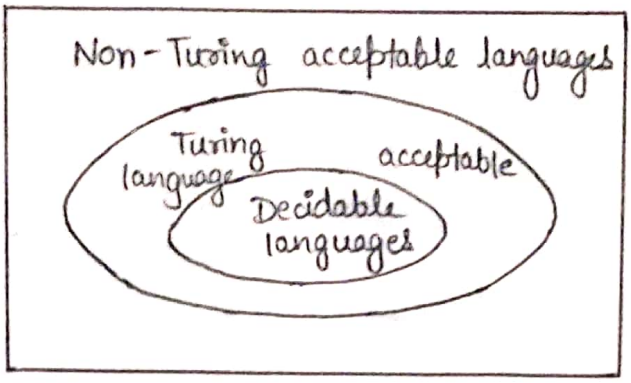
To understand this definition, we need to know that the alphabet and the finite state machine are considered to be a part of the machine, while the tape contents isn't.

So, if two turing machines differ only by tape contents they are the same machine running on different inputs.

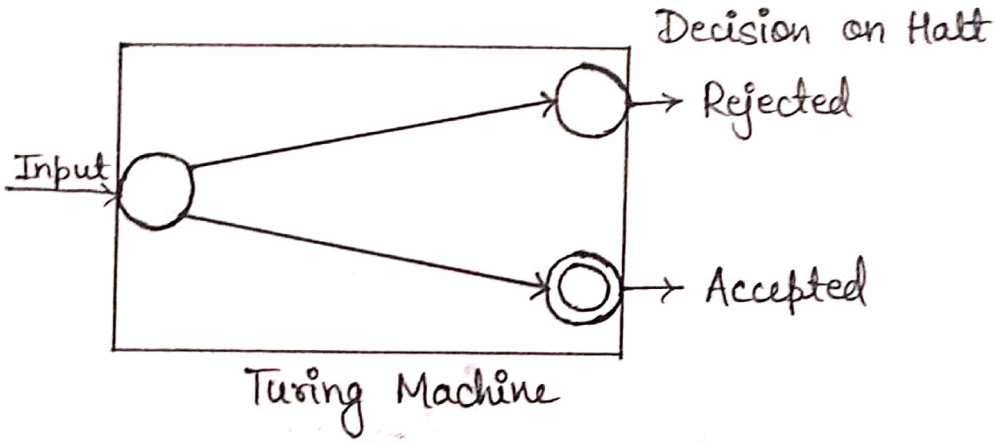
The universal machine is thus an alphabet and a FSM, which can be used to simulate any other machine by changing only the initial state of the tape.

ii. Decidable languages:

A language is called Decidable or Recursive if there is a Turing machine which accepts and halts on every input string w. Every decidable language is Turing-Acceptable.



A decision problem P is decidable if the language L of all yes instances to P is decidable.
For a decidable language, for each input string, the TM halts either at the accept or the reject state as depicted in the following diagram—



Thank You!!