Church- Turing Thesis

- A TM can be designed to solve any problem that is solvable by any effective procedure
- so any algorithmic computation that can be performed mathematically (or by mech. means) can also be performed by some TM.

* Recursively enumerable languages.

✓ A language that is accepted by a TM. is called recursively enumerable lang.

Let L be recursively enumerable lang. & M be a TM s.t. L = L(M).

so, for every $w \in L$.

$$q_0 w \overset{*}{\underset{M}{\vdash}} x_1 q_f x_2, \text{ with } q_f \in F$$

For some $w \notin L$

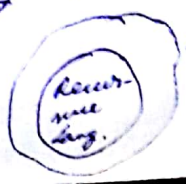
M may halt in a non-final state or it never halts & goes on in an ∞ loop.

* Recursive Languages.

- A language is called recursive if it is accepted by at least 1 TM that halts on all inputs. (note that halting may or may not be preceded by acceptance).

- By the CYK alg., every CFL is in a recursive set.

It can be proved that the recursive set is a proper subclass of recursively enumerable set.



← Set of rec. enumerable lang.

Countable sets

- In a countable set, the elements of the set can be arranged in some order.

Eg:- The set of all non-neg. even integers can be arranged in the order 0, 2, 4, ...

Since any positive integer in occurs in position $n+1$, the set is ~~considered~~ countable.

- To prove a set to be countable, a method is to be designed to so that all elements from the set can be arranged in some sequence.

We call such a method as an enumeration procedure

* (or we can generate the elements 1 after another, can be done if arrangement fixed)

- Since an enumeration procedure is some kind of mechanical process, we can use a TM to define it formally.

Let S = a set of strings on Σ.

Then an enumeration ~~process~~ procedure for S is a TM that can carry out the sequence of steps:

$$q_0 B \vdash^* q_{s_1} x_1 \# s_1 \vdash_M^* q_{s_2} x_2 \# s_2 \vdash_M^* \ldots$$

with $x_i \in T^* - \{\#\}$, $s_i \in S$ in such a way that any s_i is produced in a finite no. of steps.

$s_i \rightarrow$ ith element of countable set.

$q_{s_i} \rightarrow$ state of machine where string is generated

$x_i \rightarrow$ part of output generated

Strictly speaking an enumeration procedure cannot be called an algorithm since it will not terminate when S is ∞.

Theorem: The set of TM's, although infinite, is countable [Proof not given].

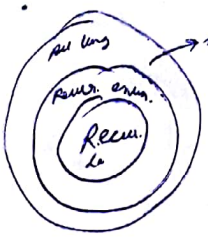
∴ as a TM can be represented as a string which can be fed to a UTM.

∴ the no. of TM's is infinite but countable from above theorem.

& encoded description of each TM on some alphabet can eventually be generated.

Theorem:
Let S be an ∞ countable set.
Then it's power set 2^S is not countable [Proof not given].

Theorem:
For any non empty Σ, there exists languages that are not recursively enumerable.



Proof:
Every subset of Σ^* is a language

2^{Σ^*} : the set of all languages.

Since Σ^* is ∞, the set of all lang. on Σ is not countable

But the set of all TM can be enumerated no

all recursively enumerable lang. is