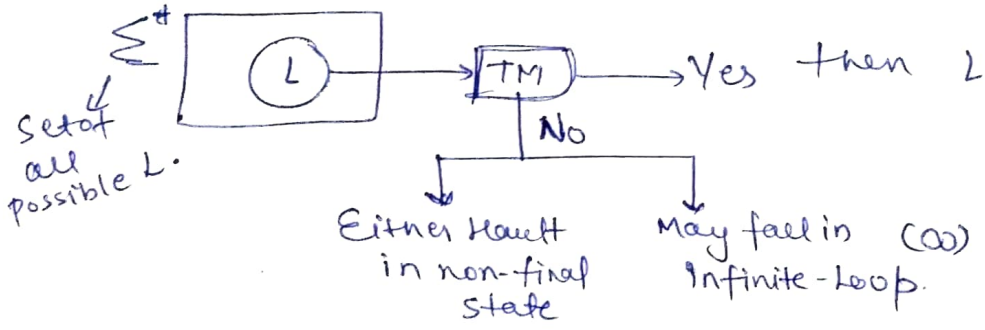
→ All the languages accepted by PDA must be accepted by LBA.

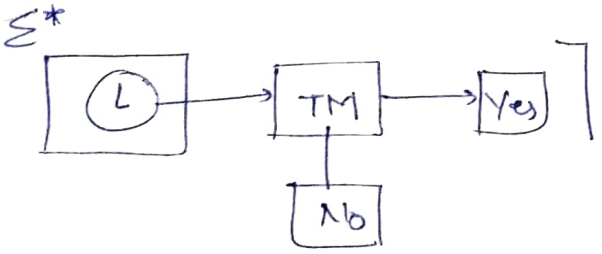
Recursively Enumerable and recursive languages

→ Language accepted by TM are called recursively enumerable languages. CFL, RL are the subset of recursively Enumerable.

imp.

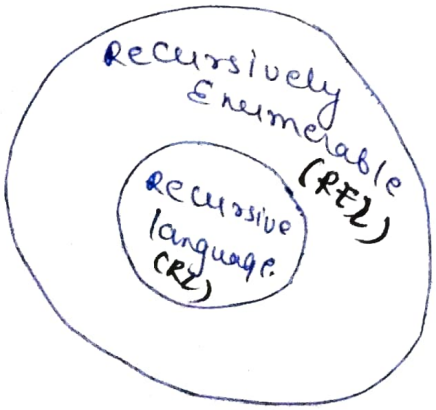
→ But ϵ doesn't accepted by TM, so we are talking about those languages that doesn't contain ϵ.





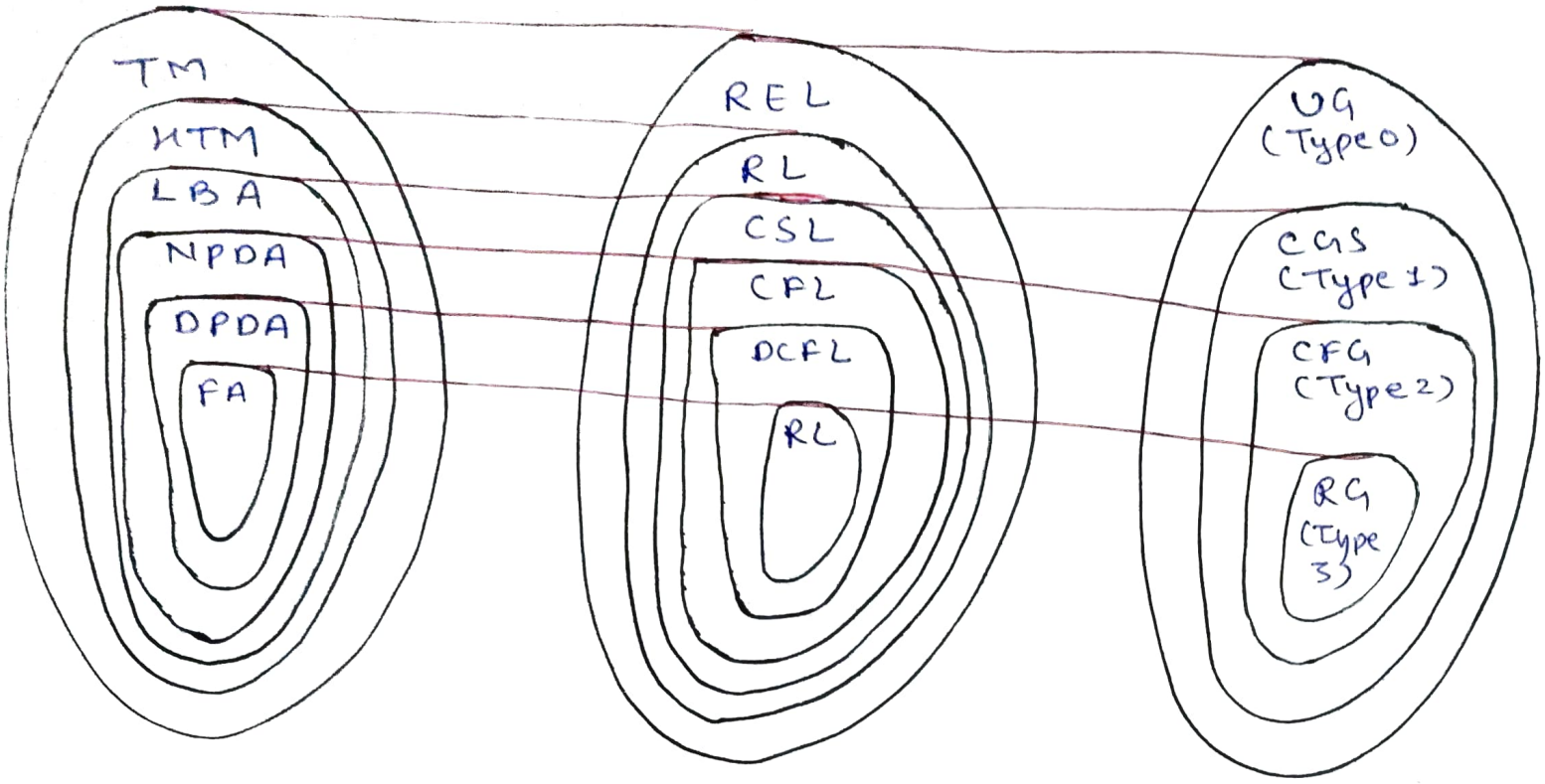
we don't want to stuck in infite loop. This type of TM called as 'Haulting turing machine.'

Recursively Enumerable Lang. includes both haulting TM and non-haulting TM. But the language accepted by haulting TM can be separately called as 'recursive language.'



→ Every recursive lang. is recursively enemerable languages.

The Big Picture



TM → Turing Machine
HTM → Haulting TM
LBA → Linear Bounded Automata
NPDA → Non-Deterministic PDA
DPDA → Deterministic PDA
FA → Finite Automata

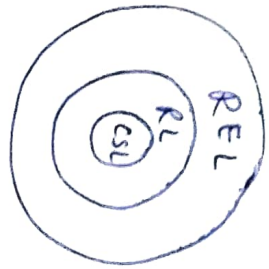
REL → Recursively Enumerable
RL → Recursive language
CSL → Context Sensitive Lang.
CFL → Contex Free Lang.
DCFL → Deterministic Context Free Lang.
RL → Regular Lang.

UG → Unrestricted Grammar
CSG → Context Sensitive Grammar
CFG → Context free Grammar.
RG → Regular Grammar

Note → Diagram contains information for various GATE problems.

Context Sensitive language

→ language accepted by Linear Bounded Automata (LBA).



REL → Recursively Enumerable
RL → Recursive lang.
CSL → Context sensitive lang.

Unrestricted Grammar → A grammar is called unrestricted grammar if all the productions are of the form $u \to v$

where $u \in (VUT)^+$ → doesn't contain $\in$

$v \to (VUT)^*$ → can contain $\in$

Turing machine → Recursively enumerable → Unrestricted Grammar

What lang. does the following unrestricted Grammar derive.

$S \to S_1B$

$S_1 \to aS_1b$

$bB \to bbbB$

$aS_1b \to aa$

$B \to \lambda$

$S \to S_1B$

$\to aS_1bB$

$\to aaB$

$\to aa$

$S \to S_1B$

$\to aS_1bB$

$\to aS_1bbbB$

$\to aabbbB$

$\to aabbb$

$L = \{ a^{n+1}, b^{n+k} : n \geq 1, k = -1, 1, 3, 5, \ldots \}$

CSG Example-1

A grammar is said to be CSG if all the productions are of form

$x \to y$

where $x, y \in (VUT)^+$ and

$|x| \leq |y|$

Always the grammar is non-contracting means size of string derived will remain same or increase after each step.

What is language generated by the following CSG:

$S \rightarrow abc \mid aAbc$

$Ab \rightarrow bA$

$AC \rightarrow Bbcc$

$bB \rightarrow Bb$

$aB \rightarrow aa \mid aaA$

Sol^n $S \rightarrow aAbc$
$\rightarrow abAC$
$\rightarrow abBbcc$
$\rightarrow aBbbcc$
$\rightarrow aaAbbcc$
$\rightarrow aabAbcc$
$\rightarrow aabbAcc$
$\rightarrow aabbBbccc$
$\rightarrow aabBbbccc$
$\rightarrow aaBbbbccc$
$\rightarrow aaabbbccc$

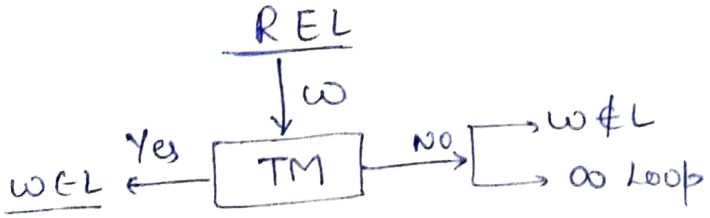
$L = \{a^n b^n c^n \mid n \geq 1\}$

→ Not asked in Gate exam yet Just remember for sake of information.

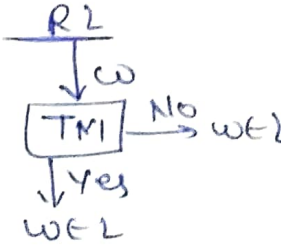
Important theorem on recursive and RE languages

If a language L and its complement L' both recursively enumerable, then both languages are recursive.

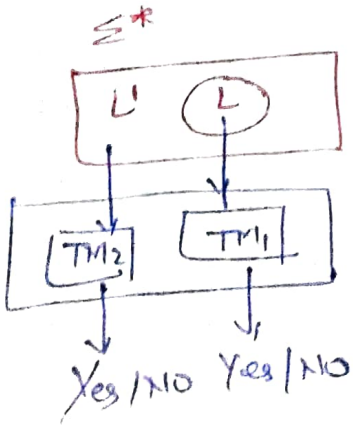
If L is recursive then L' is also recursive and consequently both are recursively enumerable.



(No Membership Algo.)



(Membership Algo exist)



TM'

→ To remove the problem of ∞ loop we make a TM combination of two TM accepting L and L' so that atleast one get the answer.

Countability

RL → CFL → CSL → RE → REL

A language can be said as a proper subset of another language if it has atleast one language that is not accepted by that language but accepted by super set language

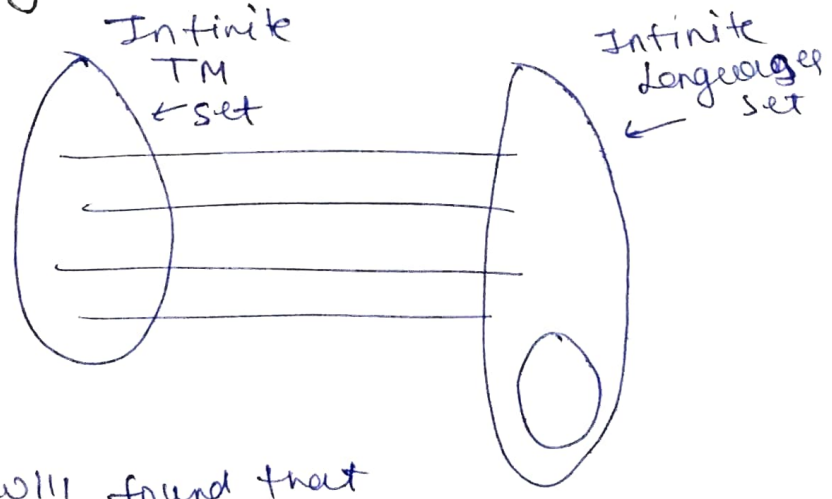
→ RL doesn't accept $a^n b^n$ but CFL accept it.

→ Similarly CFL doesn't accept $a^n b^n c^n$ but CSL accept it.

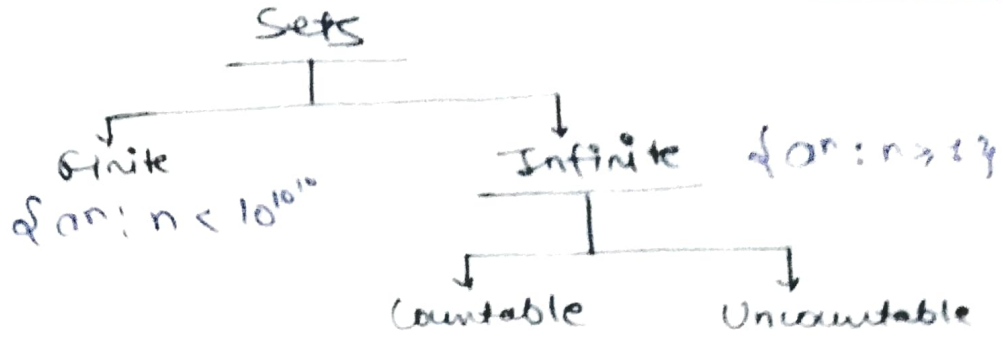
→ But in the case of CSL and RE and RE and REL we can say that CSL ⊂ RE and RE ⊂ REL but we can't predict that particular language as we were predicting in above two cases.

So we can show that **CSL ⊂ RE** and **RE ⊂ REL** using some set theorems.

Q For every language is there a TM accepting ?



→ We will found that there exist some languages that can't accept by TM too. So there are some more languages other than Recursively Enumerable.



Countable → A set 'S' is said to be countable, if all the elements of set can be put in one-one correspondence with set of natural numbers

Uncountable → A set is uncountable, if it is infinite and not countable.

Size-order → Finite < Countable < Uncountable

We have to prove that TM are countable and languages are uncountable

Ex → **Countable**

Set of Even No. = { 0, 2, 4, 6, 8, ---- } (2i)
↓ ↓ ↓ ↓ ↓ ↓
Set of Natural No = { 1, 2, 3, 4, 5, ---- } (i+1)

for each element 2i (i∈(0,---)) there exist an element (i+1) in natural no. and we also have one-one matching so set of even no. is countable.

Index: No. from natural number mapped with set

Ex → 1 is index of 0, 2 is index of 2. etc.

Means each element must have index in natural no.

for Odd no = { 1, 3, 5, 7 ---- } (2i+1)
↓
Natural no. = { 1, 2, 3, 4, ---- } (i+1) → index
↳ Countable.

S → Set of all real numbers

$S = \{0 \quad 1 \quad 2\}$

$N = \{1, 2, 3, 4, 5 \ldots\}$

Here only between 0 and 1 there are many numbers that can mapped with all natural numbers.

So, Set of real number is 'Uncountable'.

Alternative definition of countability

A set is said to be countable if there exists an ① enumeration method using which all the elements of the set can be generated and for any particular element, ② it takes only finite number of steps to generate it. The finite number of steps taken to generate an element can be used as its index and hence a ③ mapping into natural numbers.

Ex → Set of all Even no.

① Enumeration method → for (i=0 to ∞) {
generate (2i) }

o/p	0	2	4	6	8	----
No. of steps required to generate o/p	1	2	3	4	5	

② → 0 require 1 step, 2 required 2 steps (1 for 0 and 1 for 2), 4 required 3 steps (1 for 0, 1 for 2 and 1 for 4) and so on.

③ → Since all the steps are natural no. so. we can easily mapped to it.

→ Similarly we can do it for set of odd numbers.

Set of real numbers

→ Enumeration method → It is very typical to obtain, or even impossible. So we can say that it is uncountable.

Set of all quotient are countable ? → Yes, countable

Quotient of form (p/q) such that $p, q \in Z+$

$S = \{ 1/1, 1/2, 2/1, 1/3, 1/4, \ldots \}$

First no of set is $1/1$ - p/q then $p+q = 2$

we start from 2

$p+q$	(2)	(3)		(4)			(5)
p/q	$1/1$	$1/2$	$2/1$	$1/3$	$2/2$	$3/1$	$1/4$, ...
	↓	↓	↓	↓	↓	↓	↓
	1	2	3	4	5	6	7

Mapping with natural no

Here we do that, add the value of p and q $(p+q)$ in increasing order of (p/q).

Set of all strings over any finite alphabet are countable

$\Sigma = \{ a, b \}$

$\Sigma^* = \{ \epsilon, a, b, aa, ab, ba, bb, \ldots \}$

Enumeration method → fix the length of string and write all possible combination at that length and then increase length.

$\{ \epsilon, a, b, aa, ab, ba, bb, \ldots$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7$ — Steps required to generate are finite.

→ Every subset of countable is either finite or countable.

Set of all Turing Machine is countable

$\Sigma = \{0, 1\}$

$\Sigma^* = \{\epsilon, 0, 1, 10, 11, 00, 01, \ldots\}$

→ we can say that Σ^* is countable coe just prove in previous example

→ Every turing can be encoded as a string of 0's and 1's.

→ set of all turing machines (S) is subset of Σ^*.

$$S \subseteq \Sigma^*$$

→ Every subset of countable set is either finite or countable.

9mb.

So we can say that set of all turing machines are countable.

Implication of the fact that Turing Machine are countable

→ No. of Recursively Enumerable lang are countable.

→ No. of Recursive Language is countable.

→ CSL are countable.

→ CFL are countable.

→ RL are countable.

→ PDA are countable.

→ LBA are countable.

→ FA are countable.

→ Diagonalization method to prove that set of all languages are uncountable.

$\Sigma = \{a, b\}$, Σ^* is countable.

So, 2^{Σ^*} is set of all possible language and we have to prove that,

2^{Σ^*} is countable or not.

Theorem

If 'S' is countable infinite then, 2^S is uncountable.

Ex → $S = \{a, b\}$ $2^S = \{\{\}, \{a\}, \{b\}, \{a, b\}\}$

Note → Sir explains the proof but I didn't complete it bcoz it is not important for gate.

So, just focus on result that set of all languages 2^{Σ^*} are uncountable.

Problems on Countability

→ If S_1 and S_2 are countable sets then $S_1 \cup S_2$ is Countable and $S_1 \times S_2$ is Countable.

→ The cartesian product of finite number of countable sets is countable.

→ The set of all languages that are not recursively enumerable is uncountable.

→ Above all three statement can extend to more sets such that

$S_1 \cup S_2$ can be $S_1 \cup S_2 \cup S_3 \cdots$

Introduction to Computability and Decidability

→ Algorithm is a TM that is suppose to halt but TM is not going to be an Algorithm.

→ Since TM are countable so algorithm are always countable.

→ for some problem algorithm might not exist.

Computability

- function

$f(n) = n^2 + 1$

- If there is an Algo. or TM (Kaust) except 'n' as input and produce n^2+1 as output.
- fixed up the domain (value of n) first.

Decidability

- Problem : A statement whose o/p is True or false.
- Domain should be fixed first.
- There exist an Algo that take input gives o/p as True or False and get halt.

→ We are more interested in Decidability because the TM that can take decision also gives the output.

→ Both the things are actually same but context is different.

→ ~~Output is~~ depends upon the domain selected.

Ex → A given grammar 'G' is ambiguous.

D = Set of all CFG.

Above problem is undecidable because Domain have only a particular CFG but G can have other type of Grammar too that is out of bound.

→ Given an instance of a problem, it is always decidable.

→ If P_1 is ~~undecidable but~~ a problem so that there is an algorithm that converts P_1 to some problem P_2 so that P_2 have an algorithm to make it decidable so, P_1 is also decidable.

P_1 —Algo→ P_2 —Algo→ | Yes / No] Reducability

(Decidable)

→ The answer to P_2 should be the answer to P_1.

if P_2 → decidable
then P_1 → decidable

→ Imp

If P_1 is already proven to be Undecidable then P_2 must be undecidable.

P_1 —Algo→ P_2
(Undecidable) (Undecidable)

→ Haulting Problem is undecidable. so if you can convert it to some another problem then all the problem in the chain derived by haulting problem are undecidable.

Haulting Problem —Algo→ P_1 —Algo→ P_2 —Algo→

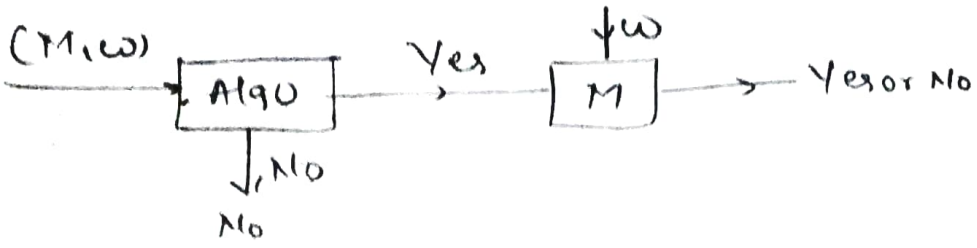
↓

Turing Machine Haulting Problem

Given the description of TM 'm' and and i/p 'w', does 'm' when started with 'w' as its i/p eventually halts?

Theorem → If the halting problem were decidable, then every RE Language would be recursive, Consequently the halting problem is Undecidable.

→ Since REL would never be equals to Recursive so, halting problem is undecidable.



first pass the Machine and String to Algo. that can verifies that on given String, Machine will halt or not

If o/p is Yes then pass string to Machine and get o/p that string belongs to lang. or not.

Some Undecidable Problems based on TM halting problem

→ The state entry problem is given a TM, a state $q \in Q$ and $w \in \Sigma^+$. Decide whether or not the state 'q' is ever entered when M is applied to 'w'.

This is undecidable.

→ Given a TM 'M', whether or not M halts if started with a blank tape. This is undecidable.

→ Almost any problem related to RE language is undecidable.

Note → Just by-heart the above statements.

Post Correspondance Problem (PC)

Given two sequences of 'n' strings on some alphabet. Σ, say $A = w_1, w_2 \cdots w_n$ and $B = v_1 v_2 \cdots v_n$, we say these exists a PC solution for pair (A, B) if there is a non-empty sequence of integers $i, j, \cdots k$ such that

$$w_i w_j \cdots w_k = v_i v_j \cdots v_k$$

PC problem is to device an algo. that will tell us for any (A, B) whether or not there exist a PC-solution.

→ In CFL, ambiguity problems are undecidable problem.

Ex.

$A =$

w_1	w_2	w_3
a	ab	bba

$B =$

w_1	v_2	v_3
baa	aa	bb

Solution is (3, 2, 3, 1) → PC solution

$w_3 w_2 w_3 w_1$ = bbaabbbaa

$v_3 v_2 v_3 v_1$ = bb aa bb bba

→ Same string can be derived in two ways. So this is an ambiguity problem.

→ Halting Problem → PC Problem → Ambiguity Problem

(Undecidable) (Undecidable) (Undecidable)

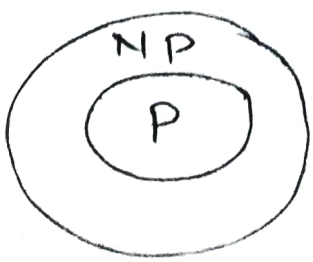
P and NP problems (Complexity classes)

* P-Class → Set of all languages that are accepted by some 'deterministic' TM in polynomial time.
  ↓
  $O(n^k)$

* NP-Class → The set of all languages accepted by non-deterministic TM in polynomial time.
  ↓
  $O(n^p)$

→ All the practical Algo. are P class and NP class Algo. are heuristics.

→ There is no restriction on no. of tapes to use.



NDTM ⟶ DTM

$O(n^k)$ $O(2^n)$

Ex $(a \cup b \cup c)$; $a, b, c \in (0,1)$

- We have to give a combination of a, b, c over 0, 1 so that it gives 0/p 1.

NDTM → Gives just the answer.

DTM → generate all combination over n to give answers.

⟶ Till now it is not known that P = NP or not.

Decidability Table

① Does $w \in L$? (Membership Problem): (RL : D, DCFL : D, CFL : D, CSL : D, RL : D, REL : UD)

② Is $L = \phi$? (Emptiness Problem) (RL : D, DCFL : D, CFL : D, CSL : UD, RL : UD, REL : UD)

Regular lang. → RL

Recursive lang. → RL

③ Is $L = \Sigma^*$? (Completeness Problem) (RL : D, DCFL : UD, CFL : UD, CSL : UD, RL : UD, REL : UD)

④ Is $L_1 = L_2$? (Equality Problem) (RL : D, DCFL : UD, CFL : UD, CSL : UD, RL : UD, REL : UD)

⑤ Is $L_1 \subseteq L_2$? (Subset Problem) (RL : D, DCFL : UD, CFL : UD, CSL : UD, RL : UD, REL : UD)

⑥ $L_1 \cap L_2 = \phi$ (RL : D) DCFL : UD, CFL : UD and all other are UD).

(7) Is L is finite or not? (finiteness) (RL: D, DCFL: D, CFL: D, CSL: UD, RL: UD, REL: UD)

(8) Is complement of 'L' a language of same type or not? (RL: D, DCFL: D, CFL: UD, CSL: D, RL: D, REL: UD)

(9) Is intersection of two languages of same type (RL: D, DCFL: UD CFL: UD and all other are D)

(10) Is L regular language. (RL, DCFL: D, and all other are Undecidable).

Properties of CFL's

(1) CFL's are closed under union, concat and Kleene closure.

L_1 → CFG_1 → $S_1 \rightarrow$

L_2 → CFG_2 → $S_2 \rightarrow$

→ On combine both L_1 and L_2 new grammar will contain

$S \rightarrow S_1 \mid S_2$

↓

Also a CFG.

→ So, $L_1 \cup L_2$ is also CFL.

→ $L_1 . L_2$ is also a CFL. ($S \rightarrow S_1 . S_2$)

→ L_1^* is also a CFL.

↓

CFG

$S_1 \rightarrow SS_1 \mid \epsilon$

where S is Grammar of L as $S \rightarrow$

(2) CFL is not closed under intersection and complements.

$L_1 \cap L_2 \rightarrow L'$ (need not be CFL everytime)

Ex → $L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$

$L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$

$L_1 \cap L_2$

Then, $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ → No CFG for this.

→ $L_1 \cap L_2 = (\overline{L_1 \cup L_2})$ → No context free

But if L_1 is also context free then $(\overline{L_1 \cup L_2})$ is also CFL and complement of $(L_1 \cup L_2)$ is also context free but its not true as we seen in intersections

Decidable problems of CFL

→ Membership, Emptiness and finiteness are decidable in CFL. (Remember it)

→ Go through the table of closure Properties of Languages given in 'Question folder of TOC'

Properties of Regular Languages

Closed under operation means after applying an operation on some input belongs to a domain then result will be of same domain.

Ex → Sum of two Natural is also a natural no.

① $L_1 \cup L_2$ → Union operation is closed for both languages.

② $L_1 . L_2$ → Concatenated language will also regular.

③ L_1^* → Kleen closure is also regular.

④ $\overline{L_1}$ → $\boxed{\overline{L_1} = \Sigma^* - L_1}$ → So it is also closed under complement

L_1 (DFA) $\xrightarrow{\text{Complement}}$ $(\overline{L_1})$ (DFA)

$(Q, \Sigma, \delta, q_0, F)$ $(Q, \Sigma, \delta, q_0, Q - F)$

⑤ $L_1 \cap L_2$ → Example of Constructive proofs

$L_1 \cap L_2 = (\overline{\overline{L_1} \cup \overline{L_2}})$ → It is also regular because complementation and Union are also regular.

Note → Regular languages are closed under Union, Intersection, Concatenation, Complement and Kleen Closure operations.

Regular Languages are also closed under difference and reversal.

① $L_1 - L_2 = (L_1 \cap \overline{L_2})$ → So difference is closed.

② DFA (L) —Reverse→ DFA (L^R)

→ Regular Languages are closed under Homomorphism.

The function $h: \Sigma \longrightarrow r^*$ is called homomorphism.

Ex → $\Sigma = \{a, b\}$ $\quad r = \{0, 1, 2\}$

$h(a) = 01$

$h(b) = 112$

for every alphabet in 'Σ' there is a string in another set of alphabet 'r'

① Homomorphic image of string Ex.

$h(aba) = \underline{01112}\underline{01}$

② Homomorphic image of Lang.

$L_1 = a^*b$ (Regular Exp.)

$= \underline{(01)^*(112)}$.

RL are closed under Inverse Homomorphism

The inverse of homomorphic image of a language L is -

$$h^{-1}(L) = \{x \mid h(x) \text{ is in } L\}$$

for a string w, $h^{-1}(w) = \{x \mid h(x) \in L\}$

Ex Let $\Sigma = \{0,1,2\}$ and $\Delta = \{a,b\}$

Define 'h' by $h(0) = a$, $h(1) = ab$, $h(2) = ba$

→ Let $L_1 = \{ababa\}$

$$h^{-1}(L_1) = \{110, 022, 102\}$$

$L_2 = a(ba)^*$

$= \{a, aba, ababa, \ldots\}$

$h^{-1}(L_2) = \{0, \underline{10}, \underline{110}, \underline{\underline{110}}, \ldots\}$

$\underline{02}$ $\underline{\underline{022}}$ $\underline{\underline{\underline{0222}}}$

$\underline{102}$ $\underline{\underline{1022}}$

Generalised version

$h^{-1}(L_2)\{1^*0, 02^*, 1^*02^*\}$

Let $\Sigma = \{0,1\}$ and $\Delta = \{a,b\}$. Defining h by $h(0) = aa$ and $h(1) = aba$. Let $L = (ab+ba)^*a$.

→ $L = \{a, aba, baa, ababa, abbaa, \ldots\}$

x ✓ x x x

then $h^{-1}(L) = \{1\}$

$h(h^{-1}(L)) = \{aba\} \neq L$

x → can't generated

✓ → can be generated from L.

$h(h^{-1}(L)) \subseteq L$

RL are closed under Quotient operation

Right Quotient → Let L_1 and L_2 be languages on the same alphabet. Then the right quotient of L_1 and L_2 is defined as —

$$\frac{L_1}{L_2} = \{x : xy \in L_1 \text{ for some } y \in L_2\}$$

Ex- $L_1 = \{01, 001, 0001, 101, 1101\}$

$L_2 = \{01\}$

$\rightarrow \frac{L_1}{L_2} = \{\epsilon, 0, 00, 1, 11\}$

Ex→ $L_1 = 10^*1$, $L_2 = 0^*1$

$L_1 = \{11, 101, 1001, 10001, \ldots\}$

$L_2 = \{01, 1, 001, 0001, \ldots\}$

Divide each element of L_1 by each element of L_2 to get $\frac{L_1}{L_2}$,

$$\frac{L_1}{L_2} = \{1, 10, 100, 1000 \ldots\}$$

→ In case, $\frac{101}{11}$ → ϕ, output will be ϕ. It is not always necessary that o/p is definitely found.

Regular Sets are closed under INIT Operation.

INIT → set of all prefixes.

Ex→ $L = \{a, ab, aabb\}$

$INIT(L) = \{\epsilon, a, \epsilon, a, ab, \epsilon, a, aa, aab, aabb\}$

→ Construct a DFA and make states as final state except dead state, it will accept Init(L).

RL are closed under substitution.

Ex→ $\Sigma = \{a, b\}$

$f(a) = 0^*$

$f(b) = 01^*$

$L = a + b^*$

$f(L) = 0^* + (01^*)^*$

→ $\Sigma \subset \Gamma$ [$\Sigma = \{a, b\}$, $\Gamma = \{a, b, x, y, B\}$

→ In TM, (tape + Control Unit).

→ (FA + stack) → (PDA) + stack → TM

TM: $\delta \rightarrow (Q \times \Gamma) \rightarrow (Q \times \Gamma \times \{L, R\})$

→ Tape is unbounded and deterministic.

Decidability Table

Problem	RL	DCFL	CFL	CSL	RL	REL
$w \in L$ (Membership Problem)	✓	✓	✓	✓	✓	×
$L = \phi$ [Emptiness Problem]	✓	✓	✓	×	×	×
$L = \Sigma^*$ [Completeness Problem]	✓	✓	×	×	×	×
$L_1 = L_2$ [Equality]	✓	✓	×	×	×	×
$L_1 \subseteq L_2$ [Subset]	✓	×	×	×	×	×
$L_1 \cap L_2 = \phi$ [Disjoint]	✓	×	×	×	×	×
Part 2						
finiteness	✓	✓	✓	×	×	×
Regular Lang	✓	✓	×	×	×	×
Complement of L is of same type or not	✓	✓	×	✓	✓	×
$\cap$ of two Lang, one of same type	✓	×	×	✓	✓	✓

Closed Under table

	RL	DCFL	CFL	CSL	RL	REL
∪	✓	×	✓	✓	✓	✓
∩	✓	×	×	✓	✓	✓
L^c	✓	✓	(×)	✓	✓	×
•	✓	×	✓	✓	✓	✓
*	✓	×	✓	✓	✓	✓
+	✓	×	✓	✓	✓	✓

(CFL) = R

✓ CFG is of type-2 as $A \to \alpha \;\forall\; \alpha \in (V \cup T)^*$

→ Non-halting TM : Infinite loop

→ All the computation can be done by TM.

→ Types of Modified TM : ① TM with stay option, ② Multitape TM, ③ Offline TM ④ Jumping TM ⑤ Non-erasing ⑥ Always writing ⑦ Multi Dimensional ⑧ Non deterministic turing machine

✓ ⑨ NDTM and DTM both are equivalent in power.

→ All the TM can be represented as a string of 0, 1.

→ ✓ If we fixed the length of Tape from both the sides TM become Linear Bounded Automata.

→ Comparison : FA < PDA < LBA < TM.

→ ✓ Lang. accepted by TM called REL but accepted by halting TM called recursive lang.

→ LBA accepts CSL → CSG (Type 1).

→ ✓ Unrestricted Grammar : $u \to v \mid u \in (V+T)^+$, $v \in (V+)^*$

→ ✓ CSG : $x \to y$: $x, y \in (V \cup T)^+$, $|x| \le |y|$

→ ✓ If L and L' are recursive and both are REL or if L and L' are REL then both are RL.

→ Finite < Countable < Uncountable.

Ex: set of all quotient (P/Q), Σ^*, set of all TM, 2^S is uncountable when S is set.

→ If S_1 and S_2 are countable then $S_1 \cup S_2$ and $S_1 * S_2$ are countable.

→ Set $(REL)^C$ is uncountable.

→ To find CFL's are RL's or not is undecidable problem

→ Halting problem is undecidable.

→ TM will accepts the string in finite steps is decidable.

→ REL = Turing recognizable & RL are Turing decidable [Recursive] & recognizable.

→ In Myhill-Nerode we get minimal DFA. Each state of DFA represents a different language and all the Languages are mutually exclusive.

→ Every regular language has a regular subset and ($\emptyset^*$) a super set of $\{\Sigma^*\}$

→ Every subset of a finite regular lang. is regular.

→ * (closure) not complement

→ $\phi \cup \epsilon = \epsilon$, $\phi \cdot a^* = \phi$, $\phi^* = \epsilon$

→ The class of regular lang. is not closed under infinite union.

→ Rice Theorem: Trivial: Property accepted by all the TM → (decidable)

Non trivial → Property accepted by some REL (TM) → (Undecidable)

→ Complement of CFL is recursive.

→ Content free language is regular: Undecidable.

→ State entry problem for TM is undecidable, but decidable in finite steps

→ We can convert RG → CFG using some Algo.

→ A language is recursive if it can be effectively enumerated in lexographical order.

→ $L_1 = \{a^n : n>0\}$ & $L_2 = \{b^n : n>0\}$
but $L_1 \cdot L_2 = \{a^n b^m : n,m>0\}$ closed under concatenation.

→ DFA for the Null language has only one state that accepts nothing. → (A) a,b

→ Operations over ϕ : $\phi \cap A = \phi$, $\phi \cdot A = \phi$, $\phi^* = \epsilon$, $\phi \cup A = A$

→ ϕ is also a regular language.

→ $\Sigma^* \cup (\text{anything}) = \Sigma^*$

→ In right linear or left linear grammar we have only one non-terminal on RHS and only one on LHS of production.

→ In CFG only one at LHS and there can be more than one at RHS.

→ finite state automation halts on all input is decidable.

→ TM prints a specific letter is elaboration of state entry problem which is undecidable.

→ TM will give o/p of any operation if it halts and halting problem is undecidable.

→ Machine accepts the string in finite steps is decidable.

→ L is decidable and L' is reducible to L. It means L' is decidable.

L is undecidable and L is reducible to L'. It means L' is undecidable.

→ WOR is accepted by NPDA so CFL.

→ If L_1 and L_2 is regular then $L_1 \cup L_2$ is regular but converse is not true.

→ If L_1 and L_2 are CFL but not regular then $L_1 \cap L_2$ is CFL but not regular depends on situation.

$L_1 = \{a^n b^n \mid n \geq 0\}$ $L_2 = \{a^n b^n c^m : n, m \geq 0\}$

$L_1 \cap L_2$ can be CFL but not regular.

→ $L_1 = \{a^n b^m \mid n > m \text{ & } n, m \geq 0\}$

$L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$

$L_3 = L_1 \cap L_2 = \{a^n \mid n > 0\}$

→ Revise all CFL and RL examples.

→ The max. no. of reduce moves taken by Bottom-up parser with no epsilon & unit production is $n-1$.