GATEFREAKS

GATE/NET/PSU

COMPUTER SCIENCE

Theory of computation : Shortnotes

Sachin Michu
Alumnus IIT DELHI

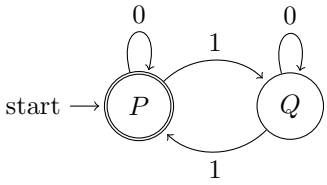
Neha Michu
Alumna IIT DELHI

August 6, 2018

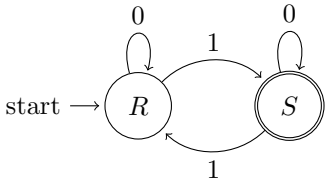
UNIT-1
Theory of Computation
Properties of Regular language and expression

Closure properties of regular languages :
If we can find finite automata corresponding to any language then that language is surely regular language . We will prove some of closure properties with the help of finite automata.
Union:$L(A)$ is regular language and $L(B)$ is regular language then $L(C)$=$L(A \cap B)$ is also regular language .
for example

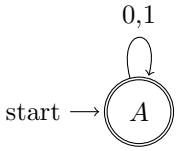
A:



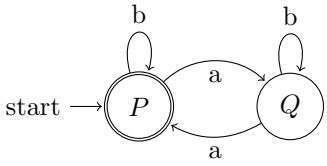
B:



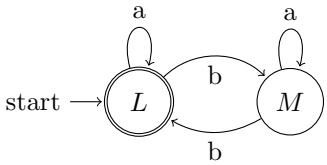
$A \cup B$:



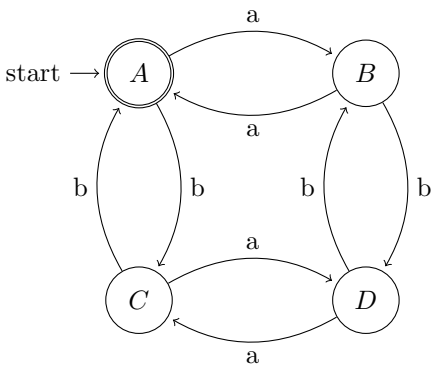
As FA(finite automata) is possible for $A \cup B$, So language corresponding is regular language .
NOTE: Regular languages are closed under the finite union but not closed under infinite union.
Intersection :$L(A_1)$ is regular language and $L(A_2)$ is regular language then $L(A_3)$=$L(A_1 \cap A_2)$ is also regular language .
for example
A_1



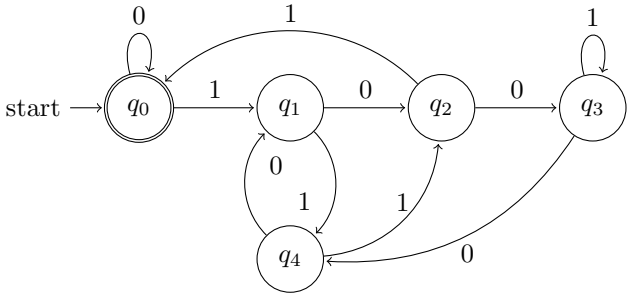
A_2



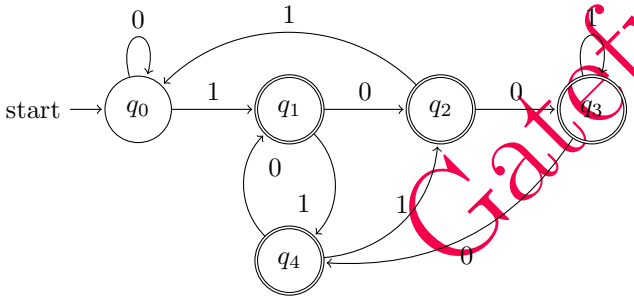
$A_1 \cap A_2$:



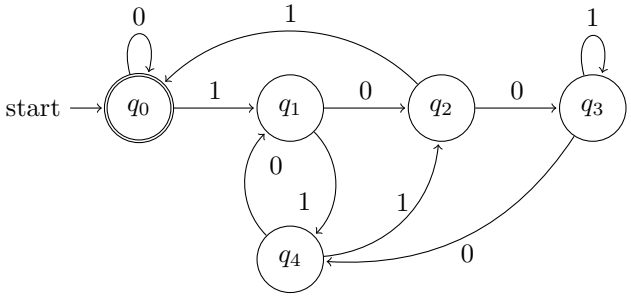
As FA(finite automata) is possible for $A_1 \cap A_2$, So language corresponding is regular language .
Complement:$L(A)$ is regular language and $L(\overline{A})$ is also regular language
for example
A:



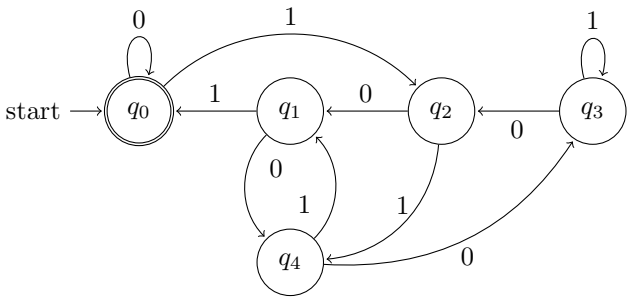
$\overline{A}$:



As FA(finite automata) is possible for $\overline{A}$, So language corresponding is regular language .
Reverse:$L(A)$ is regular language and $L(A^R)$ is also regular language
for example
A:



A^R:



As FA(finite automata) is possible for A^R , So language corresponding is regular language .
NOTE:Let A,B,C are regular languages then
♢ Reversal on union A=B+C will be $A^R = B^R + C^R$
♢ Reversal on concatenation A=B.C will be $A^R = C^R.B^R$
⇒ X be regular expression then L(X^R)=$(L(X))^R$
⇒ A=B^* then $A^R = (B^R)^*$
Difference :if L_1 is regular and L_1 is regular then $L_1 - L_2$ will surely be regular ,because $L_1 - L_2 = L_1 \cap \overline{L_2}$. As we discussed regular languages are closed under intersection and complement so $L_1 \cap \overline{L_2}$ will also be regular .
Kleene closure :if L is regular then L^* and L^+ are also regular .
Suffix: if L is regular language then ,suffix(L) = {y | xy ε L for some x ε Σ^*} is also the regular language .
Prefix: Assume that L is a regular language ,prefix(L)= {u | uvε L} is regular.
NOTE: suffix(L)=reverse(prefix(reverse(L)))
Half: Let L be a regular language. Then half(L) = {x | for some y such that $|x| = |y|$, xy is in L} is also a regular language .
Quotient: Let L_1 and L_2 be the regular languages then Quotient is defined as:
$L_1/L_2 = \{w \mid \exists x((x \in L_2) \wedge (wx \in L_1))\}$
(In other words, each string in L_1 / L_2 is the prefix of a string wx in L_1, with the remainder of the word being a string in L_2)
will also be a regular language.
homomorphism: A homomorphism on Σ is a function h : $\Sigma^* \to \Gamma^*$, where Σ and Γ are alphabets.
Let M = $x_1x_2x_3x_4.....x_n \in \Sigma^*$. Then
h(M) = $h(x_1)h(x_2)...h(x_n)$
and
h(X) = {h(M) | M ∈ X }
Example: Let h : $\{a, b\}^* \to \{0, 1\}^*$ be defined by h(a) = 01, and h(b) = ε Now
h(abab) = 0101.
If L is a regular language over alphabet Σ and h is a homomorphism on Σ , then h(L) is also regular.
NOTE: If A is some regular expression ,L(A) be the language corresponding to A .Then,
⇒ L(h(A)) = h(L(A))
⇒ h(ϵ)=ϵ
⇒ h(ϕ)=ϕ
⇒ A=B+C ,h(A) = h(B+C) = h(B) + h(C)
⇒ A=B.C , h(A)=h(B).h(C)
⇒ A=B^*, $L(h(F))^* = h(L(F))^*$
Inverse Homomorphism:
Let h : $\Sigma^* \to \Gamma^*$, where Σ and Γ are alphabets. Let L $\subseteq \Gamma^*$, and define
h^{-1}(L) = {w ∈ Σ^* | h(w) ∈ L}
h^{-1} (h inverse of L) is the set of strings w in Σ^* such that h(w) is in L.
For Example
Let L = L((aa + b)*).
Let h : $(0 + 1)^* \to (a + b)^*$ be defined by
h(0) = ab
h(1) = ba
Let $h^{-1}(L) = L((10)^*)$.
Explanation:
$(aa + b)^* \subseteq (a + b)^*$
$h^{-1}(L) = \{1010 \in \Sigma^* | h(1010) \in L\}$

h(1010)=baabbaab $\in (aa+b)^*$

Example 14. Let Σ = {a,b}, and Δ = {0,1}. Let L = (00∪1)* and h(a) = 01 and h(b) = 10.
•$h^{-1}(1001)$ = {ba}, $h^{-1}(010110)$ = {aab}
•h^{-1}(L) = (ba)*
What is h(h^{-1}(L))? (1001)* ⊆L
Note: In general h(h^{-1}(L)) ⊆L⊆ h-1(h(L)), but neither containment is necessarily an equality.
Concatenation: Regular languages are closed under concatenation operation .
Some important results on regular expressions: Let A,B and C be some regular expressions , then
• ϕ +A=A
• ϕ.A=ϕ
• $\phi^* = \epsilon$
• $\epsilon^* = \epsilon$
• $A^{**} = A^*$
• A+A=A
•$A.A \neq A$
•$\epsilon + A \neq A$
•$\epsilon.A = A$
•$(A^*B^*)^* = (A+B)^*$
•$(A^*B^* + C^*)^* = (A+B+C)^*$
•$A.A^* = A^+$
•$\epsilon + A.A^* = A^*$
•$A(BA)^* = (AB)^*A$
•$A.B \neq B.A$
• A.(B.C)=(A.B).C
• A.(B+C)=A.B+A.C
•$A + BC \neq (A+B)(A+C)$
• Number of nodes in the derivation tree of CNF is $3*2^k - 1$ **Regular or Not ?**:
Let $\Sigma = \{0,1\}$ then ,

⇒ $\{0^i1^k | i \neq k\}$ is not regular because there is comparison between i and k , which is not allowed in regular language .

⇒ $\{0^i1^k | i = k\}$ is not regular because there is comparison between i and k , which is not allowed in regular language .

⇒ $\{0^i1^k | i \geq k \vee i \leq k\}$ is regular because there is no relation between i and k.

⇒ $\{0^i1^k | i > k \wedge i < k\}$ is regular because $\{0^i1^k | i > k \wedge i < k\} = \{\phi\}$

⇒ Union of non- regular languages can be regular language. For example Union of $\{0^i1^j | i = j\}$ and $\{0^i1^j | i \neq j\}$ is $\{0^*1^*\}$

⇒ Intersection of non- regular languages can be regular language. For example Intersection of $\{0^i1^j | i = j\}$ and $\{0^i1^j | i \neq j\}$ is $\{\phi\}$

⇒ Complement of non-regular language is always non- regular

⇒ Concatenation of non-regular languages can not be regular

⇒ $\{0^i |$ i is prime number$\}$ is neither regular nor Context free .

⇒ $\{0^i |$ i is Composite number$\}$ is not regular

⇒ $\{0^i1^j |$ gcd(i,j)=1 $\}$ is not regular

⇒ Regular languages are not closed under infinite union , infinite intersection , superset, subset .

⇒ CSL, Recursive, REL are closed under infinite union , infinite intersection.

⇒ CFL and DCFL are closed under infinite union , infinite intersection.

⇒ Language having equal number of 0's and 1's is non-regular

⇒ **Language on input symbols {0,1}having equal numbers of 01 and 10 is regular .**

⇒ $\{a^n b^k | k = n, n \geq 1\}$ is not regular .

⇒ $\{a^n b^n c^m | n, m \geq 1\}$ is not regular.

⇒ $ww^r$is not regular .

⇒ ww is not regular .

⇒ $\{a^n b^k | k = 2n, n \geq 1\}$ is not regular .

⇒ $\{a^n b^m c^{n+m} | m, n \geq 1\}$ is not regular .

⇒ $\{a^n b^{2n+1} | n \geq 1\}$ is not regular .

⇒ $\{a^m b^n a^n b^m | m, n \geq 1\}$ is not regular .

⇒ $\{a^m b^m c a^n b^n | m, n \geq 1\}$ is not regular .

⇒ $\{a^n b^k | k \neq n, n \geq 1\}$is not regular .

⇒ $\{a^{2m} b^m a^{3n} b^n | m, n \geq 1\}$ is not regular .

⇒ $\{a^n b^k | 2k = 3n, n \geq 1\}$is not regular .

⇒ Every Finite subset of a non-regular set is regular .

⇒ $\{ww^R | w \in \{0,1\}^+\}$ is not regular

⇒ $\{ww^R x | w, x \in \{0,1\}^+\}$ is not regular

⇒ **$\{wxw^R | w, x \in \{0,1\}^+\}$ is regular**

⇒ $\{xww^R | w \in \{0,1\}^+\}$ is not regular

⇒ $\{a^n b^n c^n | n > 0\}$ is not CFL but CSL

⇒ $\{a^n b^n c^{2n} | n > 0\}$ is not CFL but CSL

⇒ $\{ww | w \in (0,1)^*\}$ is not CFL but CSL

⇒ $\{a^n b^n c^{2n} | n > 0\}$ is not CFL but CSL

⇒ $\{a^n |$n is prime number$\}$ is not CFL but CSL

⇒ $\{a^n |$n is perfect square$\}$ is not CFL but CSL

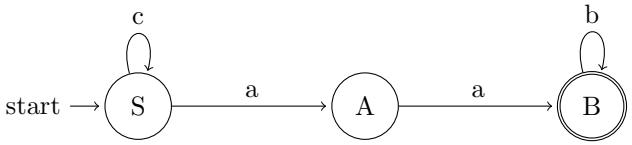
⇒ $\{a^n |$n is perfect cube$\}$ is not CFL but CSL

	DCFL and CFL	Regular	CFL	DCFL	CSL	REC	RE
1	Union	✓	✓	X	✓	✓	✓
2	Intersection	✓	X	X	✓	✓	✓
3	Kleen closure	✓	✓	X	✓	✓	✓
4	Positive closure	✓	✓	X	✓	✓	✓
5	Concatenation	✓	✓	X	✓	✓	✓
6	Intersection with regular expression	✓	✓	✓	✓	✓	✓
7	Complement	✓	X	✓	✓	✓	X
8	Difference	✓	X	X	✓	✓	X
9	Reverse	✓	✓	X	✓	✓	✓
10	homomorphism	✓	✓	X	X	X	✓
11	inverse homomorphism	✓	✓	✓	✓	✓	✓

Grammar:
"**Left linear to Right Linear Grammar:**
i)If the left linear grammar contains S → p, then put that rule in the right linear grammar.
ii)If the left linear grammar contains A → p, then put this rule in the right linear grammar: S → pA
iii)If the left linear grammar contains B → Ap, then put this rule in the right linear grammar: A → pB
iv)If the left linear grammar contains S → Ap, then put this rule in the right linear grammar: A → p": Introduction to Formal Languages by Gyorgy Revesz

Linear Grammar:
A grammar G ={V,T,S,P,} is said to be right - linear. if at each step, a production is applied to the **Rightmost** variable -occurrence in the current string.
A grammar G ={V,T,S,P,} is said to be left - linear. if at each step, a production is applied to the **leftmost** variable -occurrence in the current string.

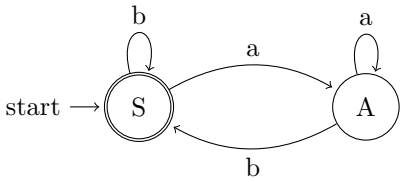
Example 1



G={V,T,P,S}

V={S,A,B}
T={a,b,c}
S=S
Right Linear grammar
P=S→cS
S→aA
A→a
A→aB
B→bB
B→b

Example 2 Right linear grammar



Right Linear grammar

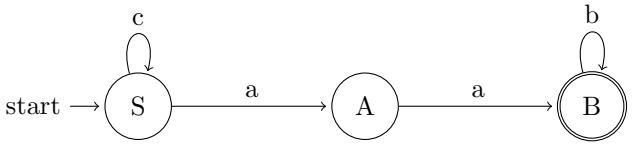
G={V,T,P,S}

V={S,A}
T={a,b}
S=S

Right linear grammar :
P: S→bS
S→ε
S→b

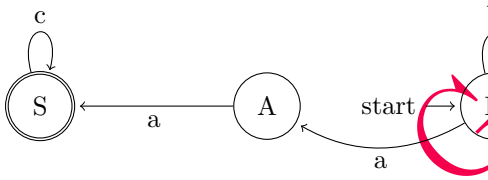
S→aA
A→aA
A→b

Example for left Linear Grammar



G={V,T,P,S}

V={S,A,B}
T={a,b,c}
S=S
Right Linear grammar
P=S→cS
S→aA
A→a
A→aB
B→bB
B→b
Right linear grammar to left linear Grammar :
First design the automata corresponding to right linear Grammar then do the following changes
•reverse the arrow direction
• Make final state as Initial state and Initial state as Final state



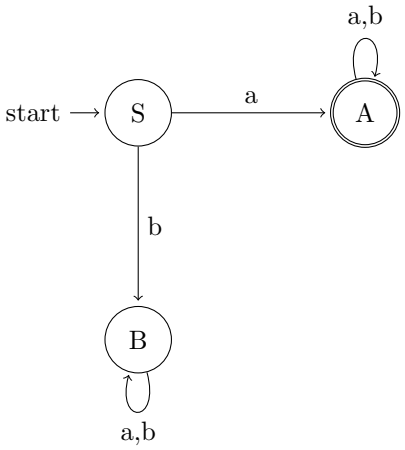
Left Linear Grammar

P:
B→Bb
B→Aa
A→Sa
S→Sc
S→c

How to find complement of an automata (explain for DFA and NFA), language and regular expression

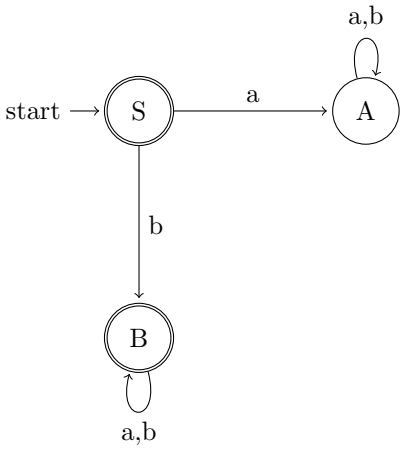
Answer: In the case of DFA ,We just make the **non-final state** to **final State** and make the **final state** to **non-final State.**
{Q,Σ,δ,q_0,F} after taking complenment {Q,Σ,δ,q_0,Q-F}

Example



L_1={a,aa,aaa,aaab,}

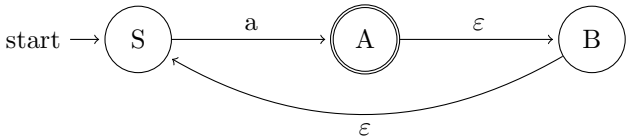
complement



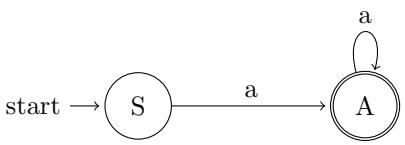
L_1={ε,b,bb,bbb,bbba,bbbbaaa,}

Complement of NFA

steps
1. first we have convert NFA to DFA
2.Than make the **non-final state** to **final State** and make the **final state** to **non-final State.**



Convert NFA to DFA (Gate Problem)



Complement of NFA L={ ε }

Normal Form:

A grammar is in a normal form if its production rules have a special structure:

• Chomsky Normal Form: Productions are of the form A → BC or A → a, where A,B,C are variables and a is a terminal symbol.

• Greibach Normal Form Productions are of the form A → $a\alpha$, whereα ∈V* and A ∈ V

• If G is a context free grammar and in chomsky normal form, w is a string of length n in L(G) then length of the derivation of w in G will be 2n-1

PDA:

PDA accepts input either by empty stack method or by final state method , both methods are equivalent i.e. there are some algorithms which can convert PDA accepting by Final state to PDA accepting by Empty state and vice versa .

There are 2 types of PDA's:NPDA and DPDA .Power of NPDA(number of language accepted by NPDA)is more as compared to DPDA .

⇛ PDA program for the $\{a^m b^n a^n b^m | m, n \geq 1\}$

Ans1.

$(q_0, a, z_0) \rightarrow (q_0, az_0)$

$(q_0, a, a) \rightarrow (q_0, aa)$

$(q_0, b, b) \rightarrow (q_0, bb)$

$(q_0, b, a) \rightarrow (q_0, ba)$

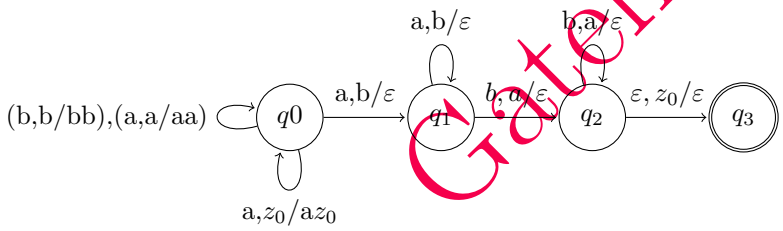
$(q_0, a, b) \rightarrow (q_1, \varepsilon)$

$(q_1, a, b) \rightarrow (q_1, \varepsilon)$

$(q_1, b, a) \rightarrow (q_2, \varepsilon)$

$(q_2, b, a) \rightarrow (q_2, \varepsilon)$

$(q_2, \varepsilon, z_0) \rightarrow (q_3, \varepsilon)$



⇛ PDA program for the ww^r

$(q_0,a,z_0)=(q_0,a,z_0)$

$(q_0,b,z_0)=(q_0,b,z_0)$

$(q_0,a,b)=(q_0,a,b)$

$(q_0,b,a)=(q_0,b,a)$

$(q_0,a,a)=(q_0,a,a)$

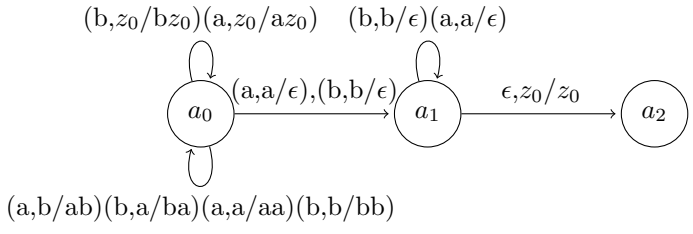
$(q_1,a,b)=(q_0,\epsilon)$

$(q_1,b,b)=(q_1,\epsilon)$

$(q_1,a,a)=(q_1,\epsilon)$

$(q_1,b,b)=(q_1,\epsilon)$

$(q_2,\epsilon,z_0)=(q_2,z_0)$



Ambiguous languages : Inherently ambiguous languages such that all the context free grammars generating them are ambiguous .
A grammar which is ambiguous can generate a regular language .for example
S$\rightarrow SS|ab|ba|\varepsilon$ =(ab+ba)* {$\because S \rightarrow SS|a|b|\varepsilon$ =(a+b)*}
Turing Machine:
Recursive languages are Turing decidable languages and Turing machine always halt,whether in accepting or non accepting state
Recursively enumerable are Undecidable and Turing machine always halt on the accepting state only .RE is also known as Turing recognizable languages .

Deterministic Turing machine(DTM) is equivalent to Non-Deterministic Turing machine(NDTM) and conversion between DTM and NDTM is possible .
Decidable/Not-Decidable:
Decidable(recursive):
For a language L if there is some Turing Machine that accepts every string in L and rejects every string not in L, then L is a decidable language.
Semi- Decidable(recursively enumerable):
if there is some Turing machine that accepts every string in L and either rejects or loops on every string not in L, then L is Semi-decidable. For example:Halting Problem , State Entry Problem , Blank Tape Halting Problem , Post Correspondence Problem , Modified Post Correspondence Problem. Undecidable(Not RE): No Turing machine exists that can even partially solve the problem in its true cases.For example: Encoding TM, Diagonalization. If M is a Turing Machine
Decidable:
It decidable whether an arbitrary Turing machine(TM) has k states, k is some constant value.
It decidable whether an arbitrary TM halts after k steps, k is some constant value.
It decidable whether an arbitrary TM accepts a string w in k steps, k is some constant value.
A TM M, a string s and an integer k, M accepts s with k steps.
{$< M >$ |M has 193 states}
{$< M >$ |M uses at most 32 tape cells on blank input}
Undecidable:
Whether Given Turing machine can find the product of two numbers.
Whether 2 Turing machine accepts same languages is undecidable. It undecidable whether an arbitrary TM ever prints a specific letter
{$< M >$ |M halts on blank input}
{$< M >$ | on input 0011 M at some point writes the symbol $ on its tape}
Rice's theorem : All non-trivial Questions are undecidable ,for example Given Turing machine accepts a string of length 50 is undecidable.
• Subset of Regular language is Regular : False
• Subset of Context Free language is Context Free language : False
• Subset of Context sensitive language is Context sensitive language : False
• Subset of Recursive language is Recursive language : False
take example as (a+b+c)*
• Set of all languages accepted by Turing machine are countable.

If A is reducible to B, and B is decidable, then A is decidable.
i) if A is reducible to B, and B is recursive, then A is recursive.
If A is undecidable, and reducible to B, then B is undecidable.
i) if B is recursively enumerable, and A is reducible to B, then A is recursively enumerable.
ii) if A is not recursively enumerable, and reducible to B, then B is not recursively enumerable

Countable:
Set of all negative integers
Set of all strings over $\sum$
Set of all Recursively enumerable languages are countable. (In short, all regular, CFG, R, RE languages are countable)
Set of all languages over $\sum$ accepted by Turing machines
Not countable:
Set of real numbers between 0 and 1

Set of all languages over $\sum$

Gatefreaks believe in providing quality. Qualifying any exam is not difficult if you have proper guidance and quality material. Our focus is to provide you the best error free content. We provide complete material including short notes, detailed notes, previous year solved papers and practice tests as a part of our intensive classroom program.For further details visit www.gatefreaks.com