BYJU'S EXAM PREP

Computer Science & IT

Theory of Computation

SHORT NOTES

Short Notes — THEORY OF COMPUTATION

1. **Symbol:** [Lowest unit] represented by Lower care or special symbols.
   * Also known as **Terminals.**
2. **Strings:** Seq. of symbols. Denoted by ω.
   * Stings should be finite, and Seq. matters.
3. **Length of String :**|ω| number of symbols involved in the String.
   ω_1 = abed ∴ $|\omega_1| = 4$
4. **Empty String or Null String:** (ϵ) String of Length O ie |ω| = 0.
   [* ϵ → String, not a symbol'
5. **Concatenation of Strings** (.) Concatenation of 2 Strings will always be a string.

   $\boxed{\omega_1 \cdot \omega_2 \neq \omega_2 \cdot \omega_1}$ $\quad \in \cdot any = any \cdot \epsilon = any$

   non-commutative,
6. **Prefix of a sting:** ω_1 = abcd ∴ Prefix (ω_1) = ϵ,a, ab, abc, abcd.
   ω_2 = aaa ∴ Prefix (ω_2) = ϵ, a, aa, aaa, aaaa.
   → [A String is a prefix of itself.]
   → [∈ is a prefix of every string.] (Trivial).
7. **Suffix of a sting:** ω_1 = abed ∴ Suffix = ϵ, d, cd, bcd, abcd.
   i.e., seq. of trailing symbols. → A string is a suffix of itself.
8. **Substring:** It is a string present in the string.
   string – {abc}
   Substring: {" ", a, b,c, ab, bc, abc}

   $\boxed{\text{max no of Sub str} = \frac{n(n+1)}{2} + 1 = \sum n + 1}$ → valid, when all symbols are unique [ie distinct].
9. **Reverse of a String of (ω^R):** ω = abcd; ω^R = dcba.
   [ω. ω^R, ω^R ω] ⇒ palindrome. Say, ω = ϵ, then ω^R = ϵ.
10. **Language:** Set of Strings. [May be finite or infinite]
    *Empty Language is possible. $\boxed{L = \{\} = \phi \neq \{\phi\}}$
11. **Alphabet:** (Σ) Non empty finite set of symbols on which the Language is defined.

    $[* \ w/o \ Alphabet, Language \ is \ not \ possible. \quad * \ infinite \ Alphabet \ not \ possible \ ]$

    * for L = { } and L = {ϵ}, any alphabet is valid
    * Σ doesn't contain ϵ.

12. Power of String [$\omega^n \mid n \in Z$]

ω = abc $\therefore \omega^\circ = \epsilon$;

$\omega^1 = \omega$ = abc

$\omega^2 = \omega . \omega$ = abc.abc

$\omega^3 = \omega\ \omega\ \omega$=abcabcabc.

13. Concatenation of Language: Just like cartesian prod, of 2 sets

L_1 = {a, b} L_2 = {1, 2}

$\therefore$ $L_1.L_2$ = {a1, a2, b1, b2}

* Concat of 2 languages is finite only if both of them are finite.

* If any of the Lang in the concat is ∞, then concat is ∞.

$$\boxed{\phi.\text{any} = \text{any} \cdot \phi = \phi}$$

14. Reverse of a Language [L^R]

L^R contain ω^R for $\forall \omega \in L$.

L= {ab, ba, aa bb, a, b}

$\therefore L^R$ = {ba, ab, aa, bb, a, b}

15. Power of Language [$L^n \mid n \in t$]

L = {a, b} L^0 = {ϵ},

$L^1 = L$ = {a, b}

$L^2 = L.L$ = {a, b} {a, b} = {aa, ab, ba, bb}

* the union and intersection of 2 languages will always be a Language.

16. Complement of a Language [L^c]:

$$\boxed{L^c = \text{Universal} - L} \quad \boxed{\begin{matrix} U^c = \phi \\ \phi^c = U \end{matrix}}$$

17. Set difference:

$$\boxed{L_1 - L_2 = L_1 \cap (L_2)^c}$$

18. Kleene Closure (*):

$$\boxed{a^* = \bigcup_{i=0}^{\infty} a^i} \Rightarrow \boxed{\begin{matrix} a^* = a^0 \cup a^1 \cup a^2 \cup a^3 \ldots\ldots \infty \\ = \{\epsilon, a, aa, aaa, \ldots\ldots\} \end{matrix}}$$

$$*\left[(a^*b^*)^* = (a+b)^*\right]$$

$$*\left(a^* + b^*\right)\{a, aa, \ldots . b, bb, \ldots, \epsilon\}$$

$$*\left[(a^*)^* = a^*\right]$$

19. Positive closure: (+)

$$a^+ = \bigcup_{i=1}^{\infty} a^i \qquad \begin{aligned} a^* &= a^+ + \in \\ a^+ &= a^* - \in \\ aa^* &= a^{+-} \end{aligned}$$

$$\phi^+ = \phi \text{ and } \left(a^+\right)^+ = a^+ \qquad (a^* b^*)^+ = (a+b)^*$$

- **Finite Automata:**

  It is used to recognize patterns of specific type input. It is the most restricted type of automata which can accept only regular languages (languages which can be expressed by regular expression using OR (+), Concatenation (.), Kleene Closure (*) like a*b*, (a+b) etc.)

- **Deterministic FA and Non-Deterministic FA:**

  In deterministic FA, there is only one move from every state on every input symbol but in Non-Deterministic FA, there can be zero or more than one move from one state for an input symbol.

  - **Note:**
    - Language accepted by NDFA and DFA are the same.
    - Power of NDFA and DFA is the same.
    - No. of states in NDFA is less than or equal to no. of states in equivalent DFA.
    - For NFA with n-states, in worst case, the maximum states possible in DFA is 2^n
    - Every NFA can be converted to corresponding DFA.

- **Identities of Regular Expression:**

  $\Phi + R = R + \Phi = R$

  $\Phi * R = R * \Phi = \Phi$

  $\varepsilon * R = R * \varepsilon = R$

  $\varepsilon^* = \varepsilon$

  $\Phi^* = \varepsilon$

  $\varepsilon + RR^* = R^*R + \varepsilon = R^*$

  $(a+b)^* = (a^* + b^*)^* = (a^* b^*)^* = (a^* + b)^* = (a + b^*)^* = a^*(ba^*)^* = b^*(ab^*)^*$

- **Complement of a FSM: [M^c]**

  Change: $\begin{matrix} \text{Final} \rightarrow \text{non Final} \\ \text{non Final} \rightarrow \text{Final} \end{matrix}$ $\Rightarrow$ this is only valid for ΔFAS, and invalid for NFA.

  * Do NOT reverse the arrows.

  * The complement of a regular language is always regular.

  * The concat. Of 2 languages are always regular.

- **Reverse of a Machine: [M^R]**

  → make the initial state as final state, and all final state as non-final. and Reverse the arrows.

  * Reverse of DFA could be NFA.

  * Reverse of RL is always RL.

  * if more than 1 initial state, then make new initial state with ϵ transition to them.

$$\cap \leq \#\text{states in DFA} \leq 2^n$$

  * n → no of states in NFA of equivalent Language.

- **Standard RL and NRL**

1. $\{a^n b\}\ R_L$
2. $\{a^n b^m\}\ n, m \geq 1\}\ R_L$
3. $\{a^n b^m \mid n \leq 7, m \leq 14\}\ R_L$
4. $\{a^n b^m \mid n = 2 \text{ or } 4\} R_L$
5. $\{a^n b^m \mid n \geq 1, m \geq 2\}\ R_L$
6. $\{a^n b^m \mid n \times m \geq 2\}\ R_L$
7. $\{a^n b^m \mid n \times m \leq 2\}\ R_L$
8. $\{a^n b^m \mid n + m = 6\}\ R_L$
9. $\{a^n b^m \mid n + m \geq 6\}\ R_L$
10. $\{a^n b^m \mid n + m \leq 6\}\ R_L$
11. $\{a^n b^m \mid n - m - 5\}\ NR_L$
12. $\{a^n b^m \mid n = m + 5\}\ NR_L$
13. $\{a^n b^m \mid 100 - n = m\}\ R_L$
14. $\{a^n b^m \mid n = 2m\}\ NR_L$
15. $\{a^n b^m \mid n > m\}\ NR_L$
16. $\{a^n b^m \mid n \geq 2m\}\ NR_L$
17. $\{a^n b^m \mid n / m = 5\}\ NR_L$
18. $\{a^n b^m \mid n^2 = m\}\ NR_L$
19. $\{a^p \mid P \rightarrow \text{Prime}\ \}NR_L$

20. $\{a^n b^n c^m \mid n \leq 7, m \geq 2\}\ R_L$

21. $\{a^n b^n c^m \mid n \geq 1, m \leq 2\}\ NR_L$

22. $\{a^n b^n c^m \mid n \geq 1, m \geq 1\}\ NR_L$

23. $\{a^n b^n c^m \mid n = m\}\ NR_L$

24. $\{a^n b^m c^p \mid n = m + p\}\ NR_L$

25. $\{a^n b^m c^p \mid n = m \times p\}\ NR_L$

26. $\{a^n b^m c^p \mid n = 2m + p\}\ NR_L$

27. $\{a^n b^n c^n \mid n \geq 1\}\ NR_L$

28. $\{a^p \mid P \text{ is Prime}, |p| \leq 30\}\ R_L$

29. $\{a^n b^n c^n \mid n \leq 10^7\}\ R_L$

30. $\{a^n a^n \mid n \geq 1\}\ R_L$

31. $\{w.w \mid w \in (a,b)^*\}\ NR_L$

32. $\{ww^R \mid w \in (a,b)^*\}\ NR_L$

33. $\{w(w^R)^* \mid w \in (a,b)^*\}\ R_L$

34. $\{w \# w^R \mid w \in (a,b)^*\}\ NR_L$

35. $\{w \mid w \in (a,b)^*\}\ R_L$

36. $\{w \times w^R \mid w, x \in (a,b)^*\}\ R_L$

37. $\{wxw^R \mid w, x \in (a,b)^+\}\ R_L$

38. $\{xww^R \mid w, x \in (a,b)^*\}\ R_L$

39. $\{xww^R \mid w, x \in (a,b)^+\}\ NR_L$

40. $\{ww^R x \mid w, x \in (a,b)^*\}\ R_L$

41. $\{ww^R x \mid w, x \in (a,b)^+\}\ NR_L$

42. $\{xcy \mid x, y \in (a,b)^*\}\ R_L$

43. $\{wxw \mid w, x \in (a,b)^+\}\ NR_L$

44. $\{wxw \mid w, x \in (a,b)^*\}\ R_L$

45. $\{w \times w \mid w \in (a,b)^* x \in (a,b)\}$ NR_L

46. $\{w \times w \mid w, x \in (a,b)^+, |w| \leq 3\}$ R_L

47. $\{a^n b^n c^m d^m \mid n, m \geq 1\}$ NR_L

48. $\{a^n b^m c^p d^q \mid n, m, p, 1 \geq 1\}$ R_L

49. $\{a^n b^m c^p d^q \mid n, m \geq 3, pq \leq 10\}$ R_L

50. $\{a^n cb^n \mid n \geq 1\}$ NR_L

51. $\{a^{2n} cb^{3m} \mid n, m \geq 1\}$ R_L

52. $\{awa \mid w \in (a,b)^*\}$ R_L

53. $\{waw \mid w \in (a,b)^*\}$ NR_L

54. $\{www^R \mid w \in (a,b)^*\}$ NR_L

55. $\{abb^n \mid n \geq 1\}$ R_L

56. $\{w \mid w \in (a,b)^*, |w| = \text{even}, \eta d(w) = , \text{ odd }\}$ R_L

57. $\{a^n b^n \mid n \geq 1, n \neq aa\}$ NR_L

58. $\{a^n b^m \mid n < m < 2^n\}$ NR_L

59. $\{a^n b^n \mid n > m > 10\}$ NR_L

60. $\{a^n b^m \mid n < m < 10\}$ R_L

61. $\{a^n b^{2m} \mid 3 < m < 4, n \geq 1\}$ R_L

62. $\{a^m b^n c^p \mid m + n + p = 10\}$ R_L

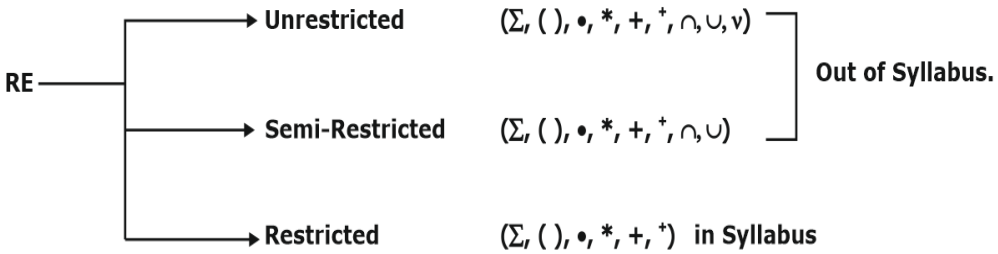
63. $\{w_1 \cdot w_2 \mid w_1, w_2 \in (a, b^*, |w| = |w_2|)\}$ R_L

64. $\{w \mid w \in (a,b)^* \mid \eta_a |w| \leq 2\}$ for every prefix of w of R_L.

- **Regular Expression:** Mathematical formula used to rep. R_L only.

  * If RE for a language exists, then its R_L.

  * RE for a given Language, will generate only the strings in the given language.



- **Priority:** ( ), a*, a^+, *, + {order of evaluation.
- **Arden's Theorem** (only for ϵ free FSM)

$$\begin{array}{|ll|} \hline & R = Q + RP \\ \Rightarrow & R = QP^* \\ \hline \end{array}$$

R, Q, P are R.E., where Q ≠ ∈.

* it has unique solution ⇒ R = QP*

- **Algebraic Laws of RE :**

Commutative Law

(a) $r_1 + r_2 = r_2 + r_1$

(b) $r_1 . r_1 \neq r_2 . r_1$

Associative Law

(a) $r_1(r_2 + r_3) = (r_1 + r_2) + r_3$

(b) $(r_1 . r_2) . r_3 = r_1 . (r_2 . r_3)$

Distributive Law

(a) $r_1 + (r_2 . r_3) \neq (r_1 + r_2) . (r_1 + r_3)$ [not distributive like this]

(b) $r_1 . (r_2 + r_3) = r_1 . r_2 + r_1 . r_3$ [this is valid]

Identity Law

(a) $r_1 + \phi = r_1$

(b) $r_1, \epsilon = r_1$

Idempotent Law

(a) $r_1 + r_1 = r_1$ [valid]

(b) $r_1 \cdot r_1 \neq r_1$ [not valid]

- **Standard Results for Regular Expression:**

1. $\left(r^*\right)^* = r^*$
2. $\phi^* = \{\epsilon\}$
3. $\phi^+ = \phi$
4. $\epsilon^* = \{\epsilon\}$
5. $\epsilon^+ = \{\epsilon\}$
6. $r^+ = r \cdot r^* = r^* r$
7. $r^* = r^+ + \epsilon$
8. $\epsilon + r = \epsilon + r \neq r$
9. $rr \neq r$
10. $\epsilon + rr^* = r^*$

11. $$\begin{aligned}(a+b)^* &= (a^*+b)^* = \left(a+b^*\right)^* = \left(a^*+b^*\right)^* \\ &= \left(ab^*+ba^*\right)^* = \left((a+b)^*\right)^* \\ &= \left(a^*b^*+a^*b^*\right)^* = \left(b^*a^*\right)^* \\ &= \left(a^*b^*\right)^*\end{aligned}$$

12. $a^*, \phi = \phi\ r \cdot \phi = \phi$

- **Important Notes:**

1. Union of 2 RL is always RL and union ⇒ commutative and Associative.
2. Finite union of RL is always RL.

   * Infinite union of RL may or may not be RL.
3. Finite intersection of 2 RL is always RL.

   * Infinite intersection is not RL. (NRL)
4. Concatenation of 2 or more RL is always RL.
5. Subsets of RL may or may not be RL. $\left[a^n b^n \le a^n b^m\right]$
6. Subset of NRL may be RL.
7. RL ∪ DCFL = DCFL RL ∩ DCFL = DCFL
   RL ∪ CFL = CFL RL ∩ CFL = CFL
   RL ∪ CSL = CSL RL ∩ CSL = CSL
   RL ∪ REC = REC RL ∩ REC = REC
   RL ∪ R.E = RE RL ∩ RE = RE
8. Kleene closure (*) and the positive closure (+) of RL is always RL.
9. Reverse of RL is always RL.
10. Every NRL has at least 1 RL superset (universal Lang.)
11. RL ∪ (L) = RL
    need not be regular.
12. NRL ∪ NRL = (L) may or may not be RL.
13. RL ∪ (L) = NRL
    Always NRL
14. RL ∩ NRL = (L) may or may not be RL
15. NRL ∩ NRL = (L) may or may not be RL.
16. NRL ∩ (L) = RL may or may not be RL.

- **Derivation:**

  To derive a string from the grammar.

  → LMD: Leftmost symbol resolved first.

  → RMD: Rightmost symbols are resolved first.

  * Graphical Rep: Parse Tree, abstract syntax Tree, derivation Tree.

  1. If there exists LMD for a string, then there will exist RMD for it and vice versa.
  2. Parse Trees of LMD and RMD may not be the same.
  3. We may have more than 1 LMD/RMD for a string.
  4. If N no of LMD's are possible, then exactly N RMD's are also possible.

- **Ambiguous grammar: [USELESS]**

  If there exists more than 1 LMD or RMD for any given string in the grammar, then it's ambiguous.

  → Parse Tree & LMD or RMD may be diff.

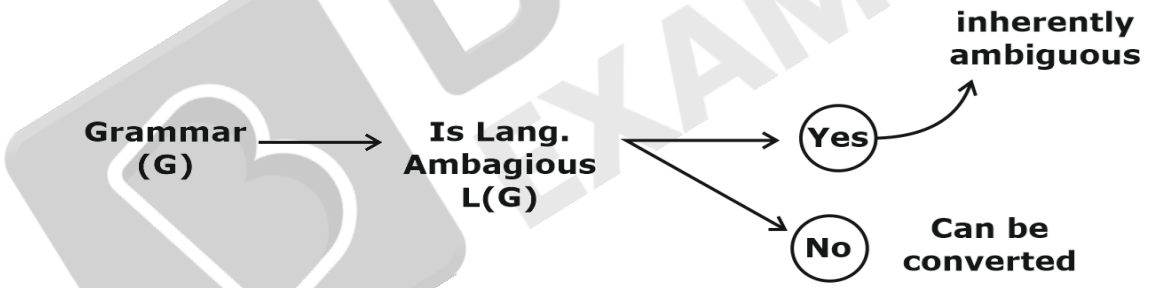
  → Grammar is ambiguous if we find atleast 1 string for which more than LMD/RMD exists.

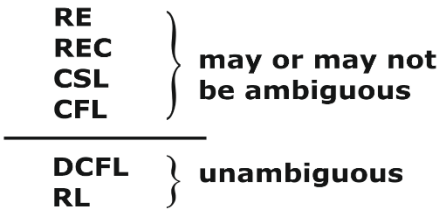
  → No algorithm exists to check ambiguity. [undecidable]

  → There are some grammars from which we can remove ambiguity.]

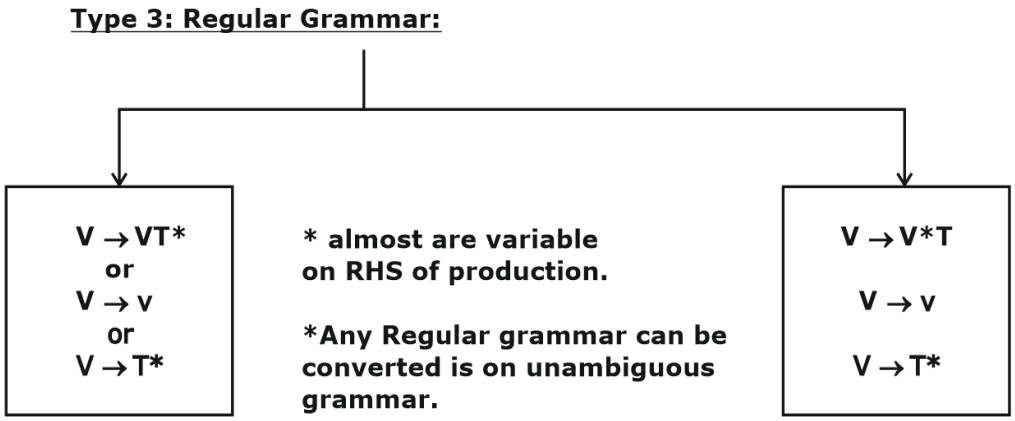
  → There are some grammars from which we can never remove ambiguity.] **Inherently ambiguous.**

  - **NOTE:**



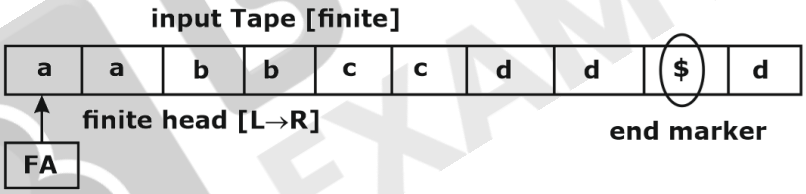


- **Type 3: Regular Grammar**



- **Moore Machine:**
  - Moore machines are finite state machines with output value and its output depends only on the present state.
  - Mealy Machine: Mealy machines are also finite state machines with output value and its output depends on present state and current input symbol.

- **Mathematical Model of FSM:**

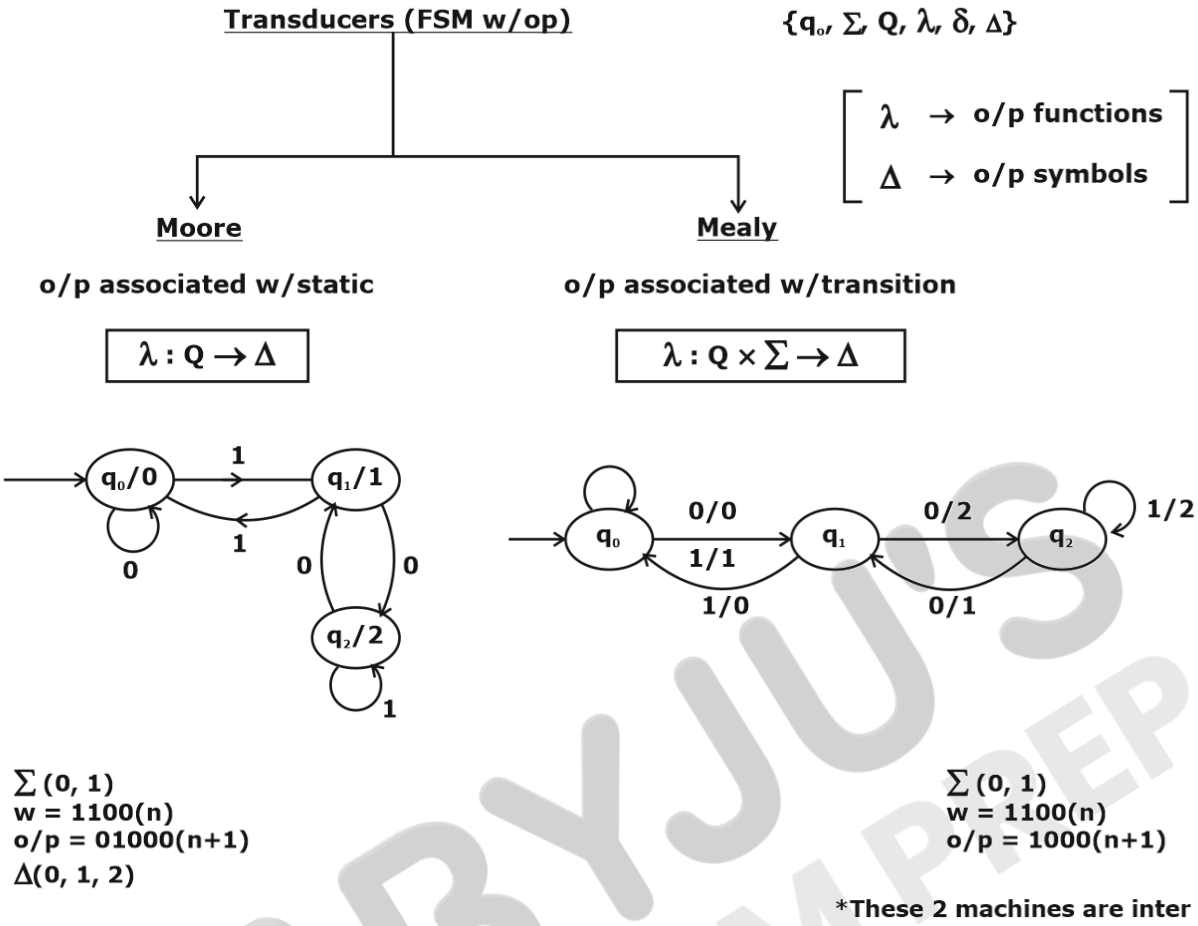


- **NOTE**

$$\text{DFA} + \infty \text{ stack} = \text{DPDA}$$
$$\text{NFA} + \infty \text{ stack} = \text{NPDA}$$

$$\text{Power (NPDA)} > \text{Power (DPDA)}$$

* $\text{DFA} + \text{Finite Stack} = \text{DFA}$
  $\text{NFA} + \text{Finite Stack} = \text{NFA}$



- **Note:**

  1. [Moore → Mealy] → no change in no. of states.
  2. [Mealy → Moore] → no. of states increases.

  * In mealy m/e if $|\Phi| = m$, and $|\Delta| = n$, the

  No of states in Moore = $(m \cdot n) \Rightarrow$ max possible.

- **Push Down Automata:**

  Pushdown Automata has extra memory called stack which gives more power than Finite automata. It is used to recognize context-free languages.

- **Deterministic and Non-Deterministic PDA:** In deterministic PDA, there is only one move from every state on every input symbol but in Non-Deterministic PDA, there can be more than one move from one state for an input symbol.
  - **Note:**
    - Power of NPDA is more than DPDA.
    - It is not possible to convert every NPDA to corresponding DPDA.
    - Language accepted by DPDA is a subset of language accepted by NPDA.
    - The languages accepted by DPDA are called DCFL (Deterministic Context Free Languages) which are a subset of NCFL (Non-Deterministic CFL) accepted by NPDA.

- **Type 2: Context free Grammar: (CFG)**

  * Single variable defines anything.

  * Every regular Grammar is CFG.

  $$\boxed{\begin{matrix} V \to (V+T)^* \\ [V \to \text{any}] \end{matrix}}$$

  - **NOTE**

    1. To check if 2 grammars are equivalent or not is undecidable.
    2. To check if 2 Regular Grammar are equivalent or not is decidable.

Type 1 : Context Sensitive	**Type 0 : Unrestricted Grammar**
$\alpha \to \beta$	$\alpha \to \beta$
1. $\alpha \in (V+T)^+$	**i)** $\alpha \in (V+T)^+$
2. $\beta \in (V+T)^+$	**ii)** $\beta \in (V+T)^*$
3. $\lvert LHS \rvert \quad \lvert RHS \rvert$	**No restrictions.**

  * Turing M/e for L = {ϵ} is not possible.

  * Context sensitive grammar does not produce ϵ.

- **CNF: Chomsky Normal Form:**

  The productions are of type $\boxed{V \to VV \text{ or } V \to T}$

  * In CNF, ϵ is not allowed

- * Algorithm for converting: (CFG → CNF)

  1. Remove ϵ - Production (unless inherent)
  2. Remove unit Production
  3. Simplify the Grammar:

  (a) Remove <u>useless</u> variables.

  (b) Remove <u>unsearchable</u> variables.

  $\boxed{\text{* No of steps in Derivation : } 2N-1}$

- **GNF: Greibach Normal Form:**

- **Reduction of the form:** $\boxed{V \to TV^*}$ ϵ - production not allowed in GNF.

  * Exactly one terminal followed by any o. of variables.

  * ϵ is allowed only in the start variable.

  * Any Grammar (CFG) can be converted to on GNF

  …… to convert: (CFG → GNF)

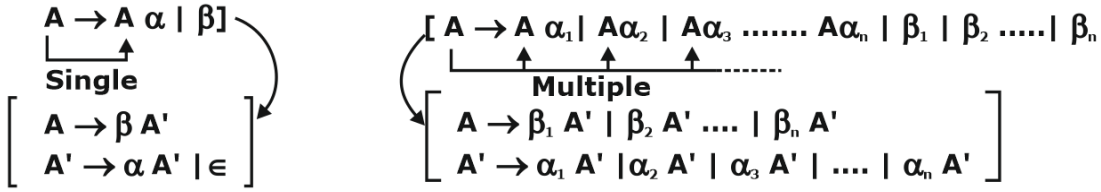
  ***** If the length of the string is N, then we need N steps for derivation from GNF.

  1. Remove Left recursion
  2. Remove ϵ-Production

3. Remove unit production

4. Simplify the grammar.

- **Removal of Left recursion :**



* Left Recursion can be removed from every CFG.

* Indirect left recursion is also possible.

- **Removal of ϵ-Production:**

$$\begin{matrix} S \to AB \\ A \to aA|\in \\ B \to bB|\ b \end{matrix} \Rightarrow \begin{bmatrix} S \to AB|\ B \\ A \to aA|\ a \\ B \to bB|\ b \end{bmatrix} \qquad \begin{matrix} S \to AB \\ A \to aA|\in \\ B \to bB|\in \end{matrix} \Rightarrow \begin{bmatrix} S \to AB|B|A|\epsilon \\ A \to aA|\ a \\ B \to bB|\ b \end{bmatrix}$$

* If lang. of grammar contains ϵ; then can't remove.

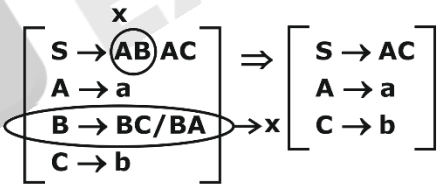
- **Removal of unit Production:**

[V → V] this is unit production.

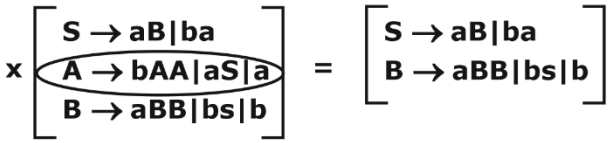
E → E + T | T	E → E + T | T * F | F	E → E + T | T * F | (E)|id
T → T * F | F	T → T * F | (E) | id	T → T * F | (E) | id
F → (E) | id	F → (E) | id	F → (E) id

- **Simplification of Grammar:**

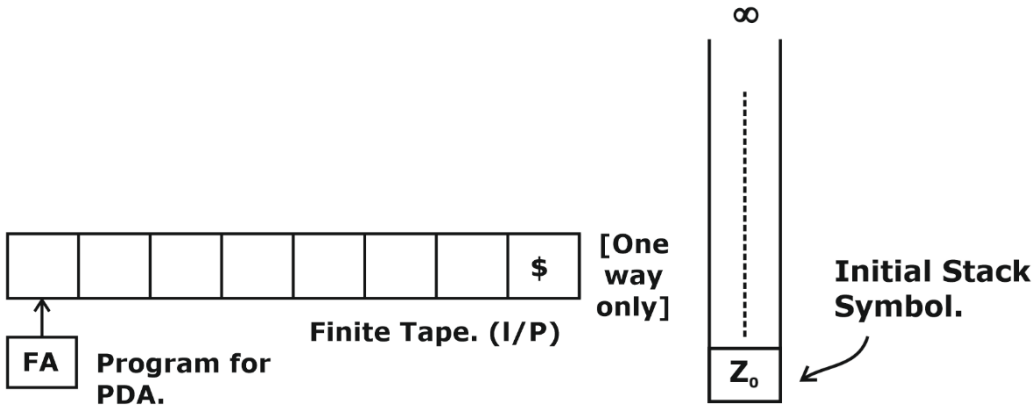
(a) useless variable: Variables that don't have terminating points are useless.



(b) unreachable variable: If the variable doesn't lie in the tree of short var.



- **Mathematical Model of PDA:**



- **7 types:** (Q, q_0, f, Σ, δ, Γ, Z_0)

  7 → Stack Alphabet. F → Final Static (can be empty).

  Z_0 → Init. Stack Symbol.

$$\text{DPDA}: \delta: Q \times \{\Sigma \cup \in\} \times \Gamma \to Q \times \text{push} / \text{pop} / \text{skip}$$
$$\text{NPDA}: \delta: Q \times \{\Sigma \cup \in\} \times \Gamma \to 2Q \times \text{push} \,|\, \text{pop} \,|\, \text{skip}$$

- **Acceptance by PDA** → [(a) empty stack (b) Final static] inter convertible, exception exits.
- **Instantaneous Description of PDA:**

$\delta(q_0, a, Z_0) \vdash (q, \text{push}|\text{pop}|\text{skip}.)$

- **Push:**

$\delta(q_0, a, Z_0) \vdash (q_1, AZ_0)$

$\delta(q_1, a, A) \vdash (q_1, AA)$

$\delta(q_1, b, A) \vdash (q_2, BA)$

- **pop:**

$\delta(q_1, b, A) \vdash (q_2, \epsilon)$

$\delta(q_2, b, B) \vdash (q_3, \epsilon)$

- **Skip:**

$\delta(q_3, C, \underline{B}) \vdash (q_3, \underline{B})$

* We can push more than one element at a time, and also pop more than one element at a time.

$$\text{CFL} \cup \text{CFL} = \text{CFL}$$
$$\text{DCFL} \cup \text{CFL} = \text{CFL}$$

$$\text{DCFL} \cup \text{DCFL} = \text{CFL}$$

$$\text{DCFL} \cap \text{DCFL} = \text{CSL}$$

$$\text{DCFL}^C = \text{DCFL} \qquad \text{CFL}^C = \text{CSL}$$
$$\text{CSL}^C = \text{CSL (May be CFL)}$$

- **Set difference Properties:**

DCFL – DCFL = CSL	DCFL – R = DCFL
R – DCFL = DCFL	CFL – DCFL = CFL
DCFL – CFL = CSL	CFL – CFL = CSL

- **Important Points:**

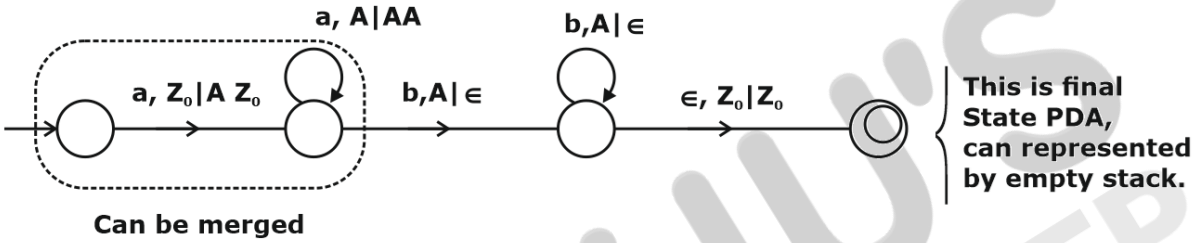
  * By PDA we can't do more than 1 comparison.

  * We can't recognize non-linear power PDA.

  * String matching not possible [ie L = {ww} not CFL]

- **Some Important PDA's:**

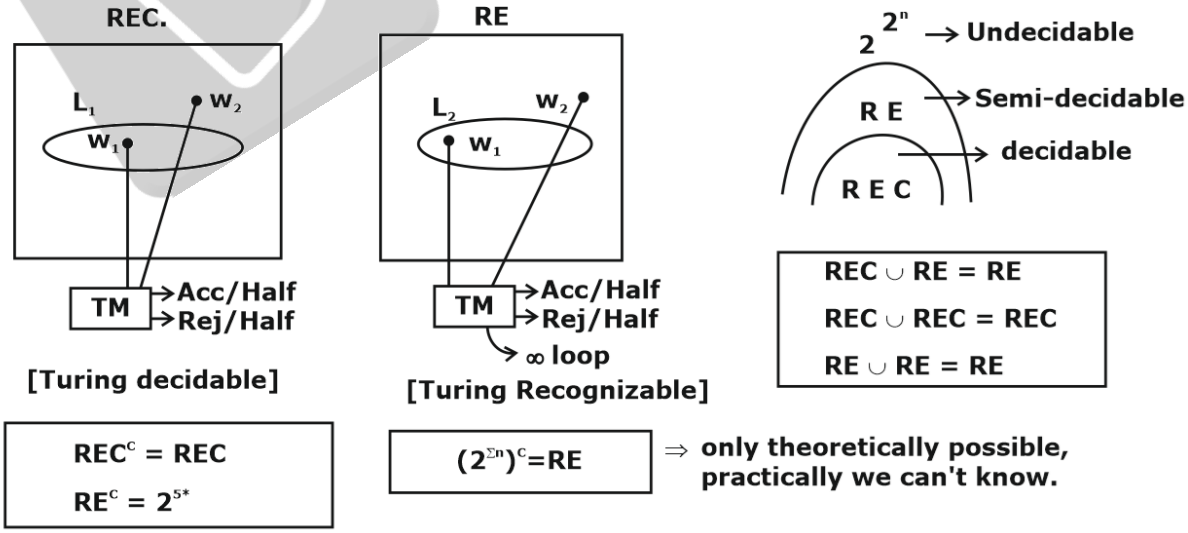
  1. L = $\{a^n b^n \mid n \geq 1\}$



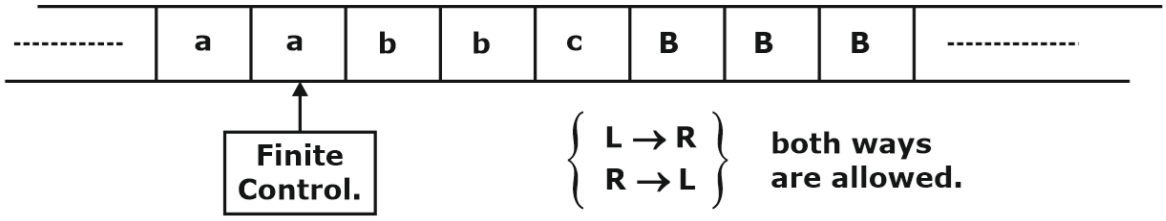
- **Linear Bound Automata:**
  - Linear Bound Automata has a finite amount of memory called tape which can be used to recognize Context Sensitive Languages.
  - LBA is more powerful than push down automata.

FA < PDA < LBA < TM

TOC – REC and RE



- **Mathematical Model of T.M. :**



- **7 Type:** {q_0, Q, Σ, B, ⌝, F, δ}

  ⌝ ⇒ Tape Alphabet B Blank ⇒ Symbol

FA+ ≥ 2Stack = TM
FA = ∞Tape but no mem. = FA
DFA + both = 2DF = FA
FA + Queue = TM
FA + 2eninter = TM
PDA = stack = TM

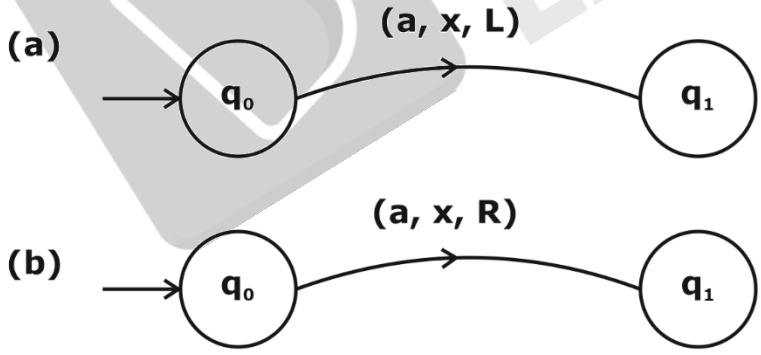
$DTM\ \delta : Q \times \urcorner \rightarrow Q \times \urcorner \times (L/R)$
$NTMS : Q \times \urcorner \rightarrow 2Q \times \urcorner \times (L/R)$

- **Instantaneous descriptions:**

$$\delta(q_0, a) \vdash (q_1, x, L/R)$$

⇒ q_0 state reads a, goes to state q_1, and written x in place of 'a', and moves left or right.

Ex:



- **Standard well known Problems :**
  1. **State entry problem:** [Given a T.M., a state q ∈ Q and w ∈ Σ^+, decide if the state q is ever entered?] Undecidable.
  2. **Halting problem:** [Given description of TM and ip: w ∈ Σ^+, does the m/c started w/ 'w' as I/P halt?] Undecidable.
  3. **Blank Tape halting problem:** [Given a T.M. if the m/c halts or not when started w/ Blank Tape?] Undecidable.

- **Note:**

  The Language which contains encoded T.M as a string (L_v) is a recursive Language, and so we can make halting. TM (HTM) for that.

  $\therefore L_v$ is R.E.C.

- **Universal Turing M/C**

  It's a T.M., where provided i/p itself is a T.M.

  $U = \{L = \{< TM, w >\}\}$

  * The set of all strings of the form <m,w> accepted by UTM is called universal Language.

  $L_U = \{< TM, w, < TM_2, w, ...\}$

- **Countability:**

  [Finite no of interval b/w w number/item]→ Countable

  N = {1, 2, ... ∞} Countable

  Even = {2, 4 ..... ∞} Countable

  Prime = {2, 3, 5, 7 ..... ∞} Countable

  R = {- ∞ .... – 0.1, ... 0.2 ..... ∞} Uncountable

  $\boxed{\Sigma^* = (a+b)^* = \text{Countable}}$

  - **Note:**

    → Subsets of countable sets are always countable.

    → Set of all TM's are countable.

    → Set of all RE's are countable.

    → Union of 2 or more countable set is always countable.

    → Power set of ∞ countable set = uncountable

    $\boxed{P(\infty \text{ countable}) = \text{uncountable}}$

    $\boxed{P(\text{uncountable}) = \text{uncountable}}$

    $\boxed{2^{\Sigma^*} = \text{uncountable}}$

    → Cartesian Product of 2 countable sets is always countable.

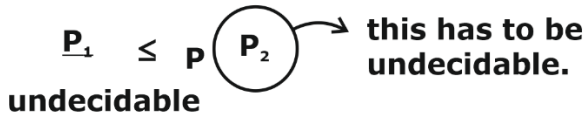
    → Complement of countable, may or may not be countable.

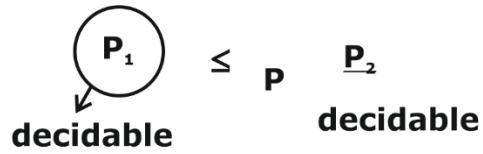
    → Complement of uncountable may or may not be countable.

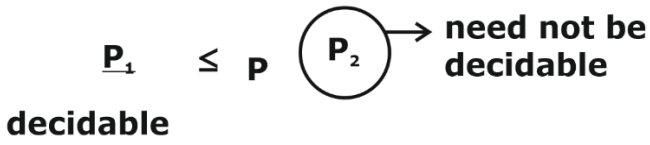
    → Intersections may or may not be countable.

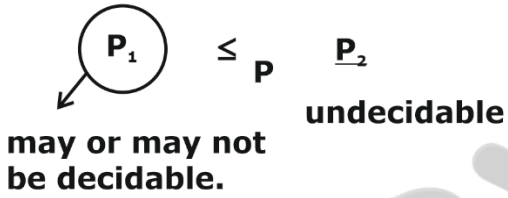
- **Computational Complexity:**

  *** We cannot reduce an undecidable problem to a decidability in polynomial time.**

1. 

2. 

3. 

4. 

- **<u>Turing Machine</u>:**
  - Turing machine has infinite size tape, and it is used to accept Recursive Enumerable Languages.
  - Turing Machine can move in both directions. Also, it doesn't accept ε.
  - If the string inserted is not in language, the machine will halt in non-final state.

- **Deterministic and Non-Deterministic Turing Machines:**

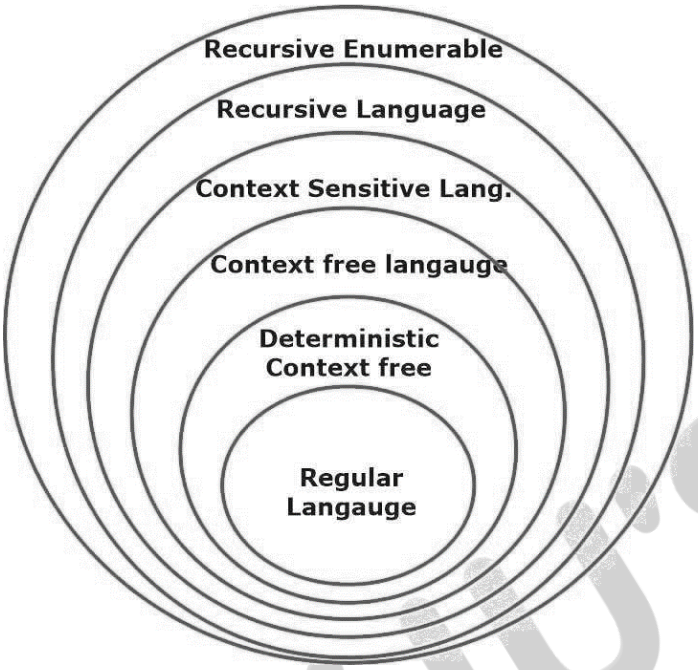
  In a deterministic Turing machine, there is only one move from every state on every input symbol but in a Non-Deterministic Turing machine, there can be more than one move from one state for an input symbol.

  - **Note:**
    - Language accepted by NTM, multi-tape TM and DTM are the same.
    - Power of NTM, Multi-Tape TM and DTM is the same.
    - Every NTM can be converted to corresponding DTM.

- **Chomsky Classification of Languages:**

Grammar Type	Production Rules	Language Accepted	Automata	Closed Under
Type-3 (Regular Grammar)	A→a or A→aB where A,B ∈ N(non terminal) and a∈T(Terminal)	Regular	Finite Automata	Union, Intersection, Complementation, Concatenation, Kleene Closure
Type-2 (Context Free Grammar)	A→ρ where A ∈ N and ρ ∈ (T∪N)*	Context Free	Push Down Automata	Union, Concatenation, Kleene Closure
Type-1 (Context Sensitive Grammar)	α→β where α, β∈ (T∪N)* and len(α) <= len(β) and α should contain at least 1 non terminal.	Context Sensitive	Linear Bound Automata	Union, Intersection, Complementation, Concatenation, Kleene Closure
Type-0 (Recursive Enumerable)	α →β where α, β ∈ (T∪N)* and α contains at least 1 non-terminal	Recursive Enumerable	Turing Machine	Union, Intersection, Concatenation, Kleene Closure

- **Relationship between these can be represented as:**



- **Decidable and Undecidable Problems:**
  - A language is **Decidable or Recursive** if a Turing machine can be constructed which accepts the strings which are part of language and rejects others. e.g.; A number is prime or not is a decidable problem.
  - A language is **Semi–Decidable or Recursive Enumerable** if a Turing machine can be constructed which accepts the strings which are part of language and it may loop forever for strings which are not part of language.
  - A problem is **undecidable** if we can't construct an algorithms and Turing machine which can give yes or no answer. e.g; Whether a CFG is ambiguous or not is undecidable.

Decidability Table						
Problem	**RL**	**DCFL**	**CFL**	**CSL**	**RL**	**REL**
Membership Problem	D	D	D	D	D	UD
Emptiness Problem	D	D	D	UD	UD	UD
Completeness Problem	D	UD	UD	UD	UD	UD
Equality Problem	D	D	UD	UD	UD	UD
Subset Problem	D	UD	UD	UD	UD	UD
L1 $nL2 = \phi$	D	UD	UD	UD	UD	UD
Finiteness	D	D	D	UD	UD	UD
Complement is of same type	D	D	UD	D	D	UD
Intersection is of same type	D	UD	UD	UD	UD	UD
Is L regular	D	D	UD	UD	UD	UD

- **Closure Properties:**

Operation	REG	DCFL	CFL	CSL	RC	RE
Union	Y	N	Y	Y	Y	Y
Intersection	Y	N	N	Y	Y	Y
Set difference	Y	N	N	Y	Y	N
Complementation	Y	Y	N	Y	Y	N
Intersection with a regular language	Y	Y	Y	Y	Y	Y
Union with a regular language	Y	Y	Y	Y	Y	Y
Left Difference with a regular language (L-regular)	Y	Y	Y	Y	Y	Y
Right Difference with a regular language (Regular-L)	Y	Y	N	Y	Y	N
Concatenation	Y	N	Y	Y	Y	Y
Kleene star	Y	N	Y	Y	Y	Y
Kleene plus	Y	N	Y	Y	Y	Y
Reversal	Y	Y	Y	Y	Y	Y
Epsilon-free homomorphism	Y	N	Y	Y	Y	Y
Homomorphism	Y	N	Y	N	N	Y
Inverse homomorphism	Y	Y	Y	Y	Y	Y
Epsilon-free substitution	Y	N	Y	Y	Y	Y
Substitution	Y	N	Y	N	N	Y
Right quotient with a regular language	Y	Y	Y	N	Y	Y
Left quotient with a regular language	Y	Y	Y	N	Y	Y
Subset	N	N	N	N	N	N

- **Countability :**
  - All strings over any finite alphabet are countable.
  - Every subset of a countable set is either finite or countable.
  - Set of all Turing Machines are countable.
  - The set of all languages that are not recursive enumerable is Uncountable.

****