Page No.
Date : / /

Regular Grammar

G → L

conversion of a Grammar to Langue

for equivalent Grammar.

G_1 : $S \to aS \mid \epsilon$

G_2 : $S \to aS \mid aaS \mid \epsilon$ } equal

if $S \Rightarrow X \mid Y$

$S \to X \mid Y \mid Z$ } these are equal if Z is obtained from X, Y.

Solving a Grammar

To solve a Regular Grammar - two ways

Substitution | Conversion of M/C & solving it

to solve a CFG - substitution Method,

(You can solve it by M/C Method but here, M/C are difficult to design) (by using PDA

Substitution is difficult if grammar has so much dependency

Ques - 1. for ϕ - $G = (V, T, P, S)$

$T = \{\ \}$

$V = \{S, A\}$

$P = \{S \to A\}$

2. for ϵ $S \to \epsilon$

3. for a $S \to a$

4 for a+b $S \to a \mid b$

EXCELLENT

5 for aba + ba - $S \to aba|ba$

6. for a*

1. $S \to aS|\epsilon$ ✓
2. $S \to Sa|\epsilon$ ✓
3. $S \to a|SS|\epsilon$

three possible grammar

[only constant term in RHS of grammar - stopper]

7 for (ab)*

$S \to ab|SS|\epsilon$ ✓
$S \to abS|\epsilon$
$S \to Sab|\epsilon$

for a grammar to L^*

$S \to SS|\epsilon|L$

* if you see $S \to \cancel{aa}|SS|\epsilon$ just solve the left part & put *

~~related to *~~

$S \to aa|bb|SS|\epsilon \Rightarrow (aa+bb)^*$

for a^+ - ~~$S \to aS|\epsilon$~~ $S \to aS|a$ $S \to SS|a$
~~$S \to SS|a$~~ $S \to Sa|a$

SS will create ambiguity in Grammar.

Grammar ambiguity is different from language ambiguity

if L(G) is ambigous - every grammar of L is ambiguous & it can not be removed.

if L(G) is unambigous - a grammar of L exist which is unambigous.

EXCELLENT

D-CFL - D-PDA

Page No.
Date : / /

A regular or DCFL language can never be ambigous.
& ambiguity in language is very ~~less~~ rare.

A RL is always unambigous bcoz for each RL, DFA exist & DFA can not accept same string in two ways as no choice is allowed so ~~the~~ multiple derivation tree can not be generated. ~~& co~~

A D-CFL is DPDA and here also, no choices allowed

CFG corresponds to NPDA

- for ~~x e~~ is a ~~a~~ st
- For x → a string

x+ $S \to xS \mid x$
$S \to Sx \mid x$
$S \to x \mid SS$

$\{S \to aS \mid b = a^*b\}$

for $S \to xS \mid y$ $\{xe = x^*y\}$

for $(01)^*100$
$S \to 01S \mid 100$ (Right linear)

- for $100(01)^*$
$S \to S01 \mid 100$ Left linear

* $S \to Sx \mid y$ and $S \to xS \mid y$ are not equivalent

yx^* x^*y

EXCELLENT

Page No.
Date : / /

- Every left linear Grammar has equivalent right linear grammar.

- if x^*y use LLG
  xy^* use RRG

- a^*+b^* —— $S \to aS \mid \epsilon$
  $S \to bS \mid \epsilon$
  for +
  $S_1 \to aS_1 \mid \epsilon$
  $S_2 \to bS_2 \mid \epsilon$
  start symbol — $S \to S_1 \mid S_2$ — Substitution Method

- if $S_1 \Rightarrow aS \mid \epsilon$
  $S_2 \to aaS \mid \epsilon$
  R.E = $a^* + (aa)^* a^*$ — If you can simplify the given expression just simplify it.

- a^*b^*
  $S \to S_1 S_2$
  $S_1 \to aS_1 \mid \epsilon$
  $S_2 \to bS_2 \mid \epsilon$

- When you do like $S \to S_1 S_2$ Grammar is not regular Grammar.
  (start)

Ques - RG_1 with S_1 produce L_1
RG_2 with S_2 —— L_2

New Grammar has
- all production of G_1
- G_2
- $S \to S_1$
- $S \to S_2$

Resulting Grammar is CFG as if G_1 = LLG with $L_1 \cup L_2$ G_2 = RRG

Page No.
Date : / /

{ It is CFG with regular language produce $L_1 \cup L_2$ }

if G_1 is left linear & G_2 is right linear if they get mixed it will not be a regular Grammar.

if we have $G_1 = RRG$
$G_2 = RRG$
$S \to S_1 S_2$
} - CFL with RL and $L_1 \cdot L_2$

for concatenation- if Grammar is to be regular.

- for a^*b^* $S \to aS | b^* | \epsilon$
  Here you cannot put the language
  ~~$S \to aS$~~ use variable of that langua

$S \to aS | bS | \epsilon$
$S_1 \to bS | \epsilon$
} then S_1 is known as Variable stopper

- $a^*b^*c^*$
  $S \to aS | A$
  $A \to bA | C$
  $C \to cC | \epsilon$

- $(a+b)^*$ — { $S \to aS | bS | \epsilon$ }
  $S \to Sa | Sb | \epsilon$
  $S \to a | b | SS | \epsilon$

EXCELLENT

Eg {a, b}

Page No.
Date : / /

$(a+b)^+$ $S \to a|b|aS|bS$
$S \to a|b|Sa|Sb$
$S \to a|b|SS$

for end with a $S \to aS|bS|a$
for end with b $S \to aS|bS|b$
for start with a $S \to Sa|Sb|a$

$(x+y)^*$ $S \to xS|yS|\epsilon$
$S \to Sx|Sy|\epsilon$

$(x+y)^+$ ~~$S \to xy|\epsilon$~~ $S \to xS|yS|x|y$
$(x+y)^*z \to S \to xS|yS|z|\epsilon$

if $(x+y)^*(10)^* \to S \to xS|yS|A$
$A \to 10A|\epsilon$

If $\{S \to xS|Sy|\epsilon \Rightarrow$ ~~$(a+b)^*(a^*b^*)$~~$\}$

$\{S \to 0|S|S11|\epsilon$ ~~$(00$~~ $((01)^*(11)^*)\}$

$\{S \to 01S|S11|011|\epsilon$ ~~$((01)^*(11)^*)011$~~

$\{(01)^*011(11)^*\}$

Linearity of a Grammar-

Left side must have single variable atmost but allow single variable is ~~allow~~ RHS at any pos^n

$\{V \to VT^* + T^*V + T^*VT^* + T^*\}$

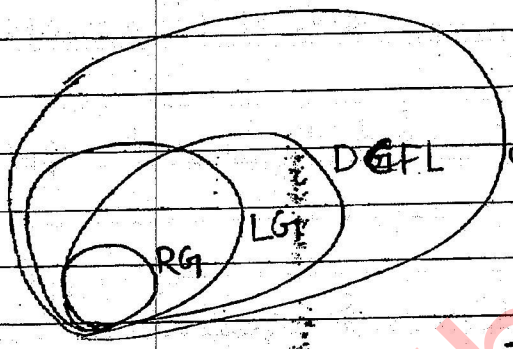
EXCELLENT

Initial state is only 1.

Page No.
Date : / /

Ex. $S \to Aaa \mid aaB \mid aBa \mid aa \mid \varepsilon$ (Linear Gramma

$S \to aBaA$ // Not a linear Grammar



17 APR

Algorithm in FA-

1. NFA to equivalent DFA → Subset construction Algo // do not work for null Move
2. DFA to equivalent min DFA
3. NFA with Null Moves to NFA without ε-moves.

} decidable

{ NFA Minimization is decidable but it is NP problem }

ε-closure form transition system (so many initial state)

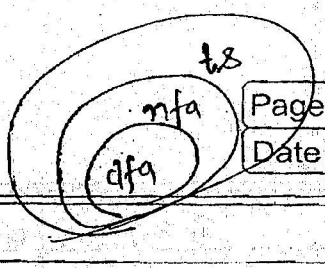
- NFA never has more than one initial state.

Subset Cons. Algo- (. I/P - NFA without null Moves or any transition system)

NFA with Null Move → Transition system w/o ε-move
↳ NFA w/o ε-move.

- A DFA to NFA is also decidable as DFA is also a NFA.

EXCELLENT



Page No.
Date : / /

1. Subset Construction Algo-

A. if you have n-state NFA, then its equivalent DFA has atmost 2^n states by using subset construction algo as

$Q = \{q_0, q_1, q_2\}$ $\quad 2^Q = \{\phi, \{q_1\} \ldots \{q_1, q_0, q_2\}\}$

B. Let I/P NFA $= M_N(Q, \Sigma, \delta, q_0, F)$

& equivalent o/P DFA $= M_D(Q', \Sigma', \delta', q_0', F)$

Here

1. $Q' \subseteq 2^Q$
2. $\Sigma' = \Sigma$
3. $q_0' = \{q_0\}$ ↳ as Now $q' \in 2^Q$ as states are in bracket
4. $F' \subseteq Q' \subseteq 2^Q$
5. $F' \not\subseteq 2^F$ as it will not allow some new elmt of Q. $F' \not\subseteq F$

Example

	0	1
$\rightarrow q_0$	q_1, q_2	—
q_1	q_2	q_1
q_2	q_0	q_1

Transition table on NFA

If you have NFA with DC only, just put ~~DC~~ trap.

EXCELLENT

D | NFA with n state → equivalent DFA can have atleast n and almost 2^n states

Page No.
Date: / /

Equivalent DFA is

M'	0	1
→ $\{q_0\}$	$\{q_0,q_1\}$	ϕ
$\{q_1,q_2\}$	$\{q_0,q_2\}$	$\{q_1\}$
$\{q_0,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_1\}$
ϕ	ϕ	ϕ
$\{q_0,q_1,q_2\}$	$\{q_0,q_1,q_2\}$	$\{q_1\}$
$\{q_1\}$	$\{q_2\}$	$\{q_1\}$
$\{q_2\}$	$\{q_0\}$	$\{q_1\}$

- Start procedure from initial state
- go on with each state entering
- if ~~all~~ no new stat came, just stop

$\{q_1,q_2\}$ - have behaviou of q_1,q_2.

ϕ is a valid state used as trap.

$\delta(\phi,0)=\phi$

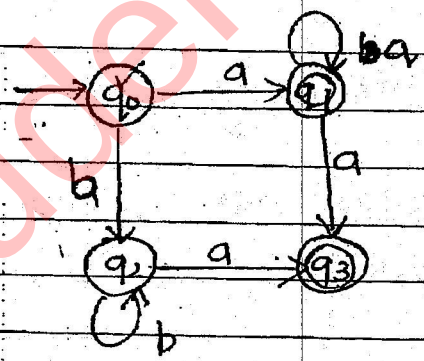
$\delta(\phi,1)=\phi$

- Any state containing F then become final state

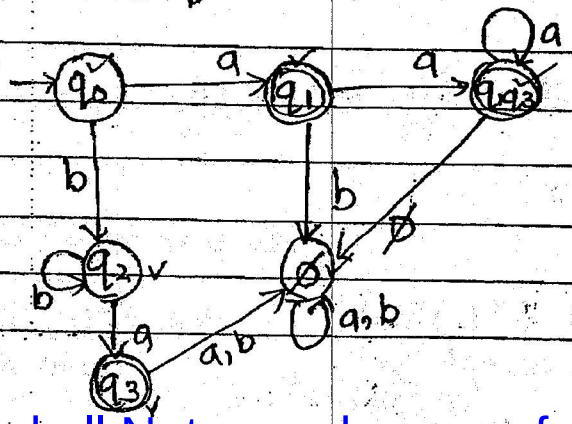
No of states in equivalent DFA = 7

No of final state = 4

Ques



equivalent DFA is (directly convert it)



No of state = 6

F state = 3

EXCELLENT

{F = can be ∅}

Page No.
Date : / /

- If Input NFA has an DC without choice, it will be having trap state.
- If I/P NFA has DC state with choice, it may or may not have trap state

Ex

	a	b
→q_0	q_1, q_2	q_2
q_1	q_2	-
q_2	q_0	q_2

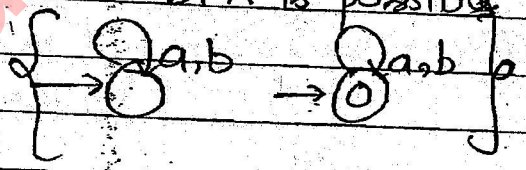
- but if I/P NFA has DC state with choice at initial state, it will have trap state.
- If a DFA has n state its equivalent NFA will have atmost n state.

Minimization of DFA

1. If DFA with n state, its min. DFA will have states as

$1 \leq$ no of state $\leq n$

- O/P Mm. DFA will n states if I/P DFA itself is minimal.
- for $\Sigma = \{a, b\}$ two DFA is possible with 1 state



- Working- it minimizes the machine by identifying equivalent states

ie, ∀ state $A \equiv B$

if $\forall\ w \in \Sigma^*$

$\delta^*(A, w)$ & $\delta^*(B, w)$ both same behaviour

will both accept w or reject it.

Here $\delta^*(A,w)$ & $\delta^*(B,w)$ one may on

Myhill-Nerode Theorem is used for Minimization of DFA and to check whether a lang. is regul

Page No.
Date :

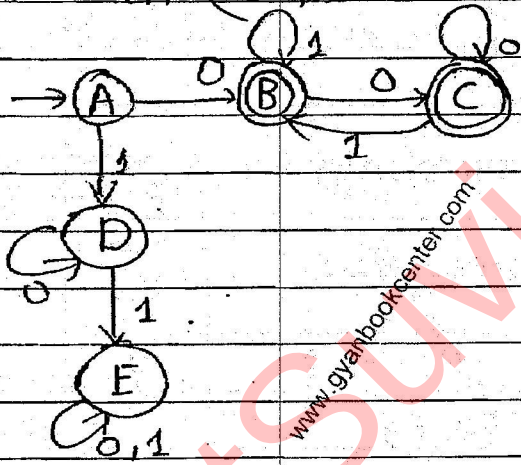
$\delta^*(A,\omega) \neq \delta^*(B,\omega)$ ⇒ it is not always required that they both must ~~reduce same~~ produce same state.

We can not go with checking equivalence as there may be ~~an~~ endless a case where no end exist.

To check equivalence, we use equivalence class Π

Here if $\Pi_k = \Pi_{k+1}$ (stop)

Ques –



Π_0 → all states which are 0equivalent
i.e which shows same behaviour at ∈ strings

behaviour of state is Accept or Reject.

So Here B, C Accept Null (as both are final state)
A, D, E reject Null

$\therefore$ $\Pi_0 = \{\{B,C\}\{A,D,E\}\}$

$\delta^*(B,\epsilon) = B$
$\delta^*(B,\epsilon) = C$
$\delta^*(A,\epsilon) = \phi$
$\delta^*(D,\epsilon) = \phi$
$\delta^*(E,\epsilon) = \phi$

B, C shows same behaviour and A, D, E show same behav.
{B, C} are putted together & {A, D, E} are putted together.

EXCELLENT

If I/P is a minimal DFA is given then it will partition into different state {A}, {B}, {C}, {D}, {E}

- this algo will take worst case time when ~~the~~ I/P itself is minimal DFA.

Page No.
Date : / /

$\Pi_0 = \{\{B, C\}\{A, D, E\}\}$

if two states are not k equivalence then they can never ~~have~~ be k+1 equal equivalence

B, A can never be equivalent

for

Π_1 = check for equivalence for B & C

check for 1-length ~~sent~~ string

if O/P of B & C belong to same block in Π_0 then they are Π_1 equivalent.

$B \xrightarrow{0} C$
$C \xrightarrow{0} C$ } same ~~block~~ block

$B \xrightarrow{1} B$
$C \xrightarrow{1} B$ } same block

$\Pi_1 = \{\{B, C\}, \{A\}, \{D, E\}\}$

$A \xrightarrow{0} B$
$D \xrightarrow{0} D$ } not in same block

separate A & D

$A \xrightarrow{0} B$
$E \xrightarrow{0} E$ } not in same block

so ~~sp~~ separate A & E

$D \xrightarrow{0} D$
$E \xrightarrow{0} E$ } same block

$D \xrightarrow{1} E$
$E \xrightarrow{1} E$ } same block but we cannot put bar

$B, C \xrightarrow{0} B$ and $B, C \xrightarrow{1} C$ } always be in same block

∴ if something is like permanent

If $B \xrightarrow{0} D$, $C \xrightarrow{0} B$ } you can not put bar as it may get divided.

Max equivalence class - no of states

Page No.
Date : / /

$\Pi_2 = \{\{B, C\}, \{A\}, \{D, E\}\}$

$\Pi_1 = \Pi_2$

i.e K-equivalent
& -equivalent

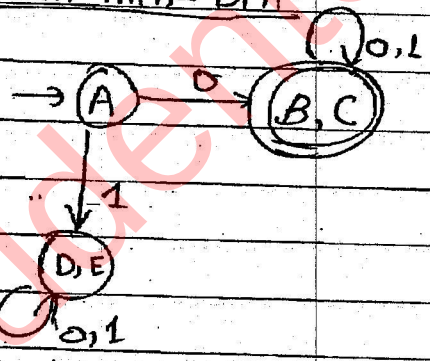
Steps -

for Π_0 - put final states & Non final state separated as ϵ-string is only accepted by final string.

for $\Pi_1 \cdots \Pi_k$

1. check for 1 length string for each state in same block
2. if $\Pi_k = \Pi_{k+1}$ stop
   else again

∴ equivalent min- DFA -



- Limitation-
  - This algo is incapable of removing non reachable states
  - so when a DFA contain Non-reachable state, firstly remove them.

EXCELLENT

Page No.
Date : / /

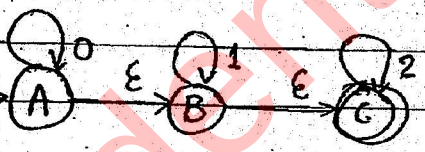
E-NFA - NFA having atleast one ε-move.

ε-closure

if a ε-nfa of n states with k ε-move & p size alphabet then its transition system with no ε-move will also have n states as no new states get added or no previous getting deleted.

~~O/P~~ :
~~If a ε-nfa of n states &~~
I/P -
I/P - ε-nfa of n state
O/P - nfa with no of state
Ans - @ atmost 2^n

Ex $0^*1^*2^*$

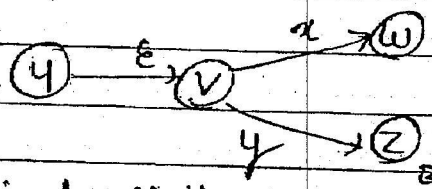


Transition state with ε-move-

ε-closure (A) = $\delta^*(A, \varepsilon)$
ε-closure (A) = {A, B, C}
ε-closure (B) = {B, C}
ε-closure (C) = {C}

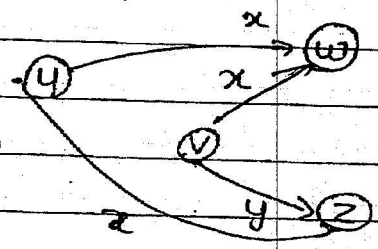
Page No.
Date : / /

State steps to remove ε move



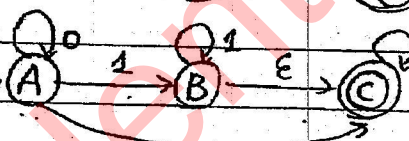
1. duplicate all the arrow on v are as if originating from u (duplicate all the arrow originating for v as it is originating from u)
2. if u is a initial state then make v as initial state.
3. if v is a final state the make u as final state.

step 1

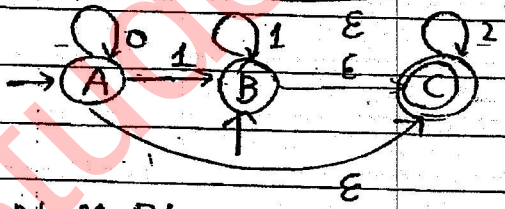


Ex :-

A→B

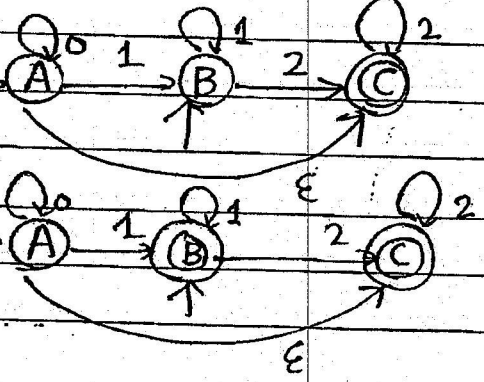


- duplication of arrow



- initial state

Remove Null B/w B & C

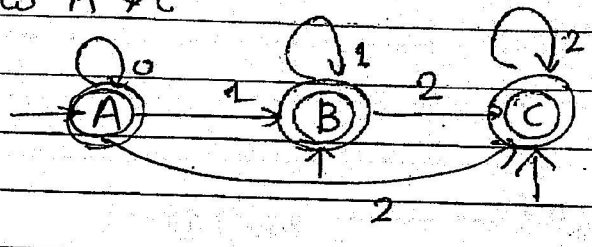


- final state travel backward
- Initial state travel forward

EXCELLENT

Page No.
Date : / /

b/w A & C



Conversion of ε to NFA

EXCELLENT

Page No.
Date : / /

Regular? CFL? CSL

Guidelines:-

1. A Finite Language is always regular

Ex $\{a^n b^n \mid n \geq 0\}$ - Not regular
$\{a^n b^n \mid n \leq 10\}$ - Regular
$\{a^n b^n \mid n \geq 10\}$ - Not a Regular language

$\{a^n \mid n \geq 0\}$ - ~~Linear~~ Regular as FA can be drawn.
$\{a^{3n+1} \mid n \geq 0\}$ - Regular
↳ if powers are linear then grammar is reg

Non Linear powers strings are ~~CFL~~ or CSL

$\{a^{n^2} \mid n \geq 0\}$ - Not Regular
$\{a^{n^2} \mid n \leq 5\}$ - Regular

~~A finite~~ A infinite language with non-linear power is always CSL.

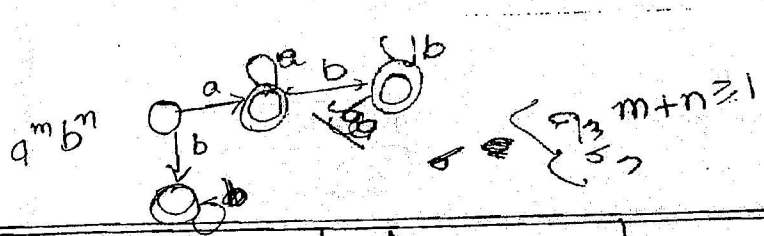
A infinite language with linear power is always regular when it does not require any string ~~machine~~ matching or any storage

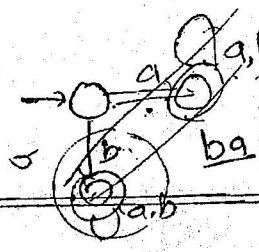
$\{2^n \mid n \geq 0\}$ - Not a regular as multiplication is required

$\{a^m b^n \mid m+n = 10\}$ Regular (FA & PDA can't do multiplication & division)
$\{a^m b^n \mid m+n = p\}$ - Not regular

$\{a^m b^n c^p \mid m+n = 10\}$ Regular
$\{a^m b^n c^p \mid m \times n = 10\}$ Regular
$\{a^m b^n c^p \mid m+n = p\}$ CSL

EXCELLENT





Page No.
Date : / /

$\{a^m b^n c^p \mid m-n=10\}$

↓ Not Regular
as it is infinite & it requires comparison.

- if any lang. is infinite if
  - Non linear power then it is not regular
  - infinite sub-mul (+, -, *, /) then it is not regular
  - infinite comparison ——
  - infinite string machine ——

FA - can never do string Matching
PDA can do one string matching but in reverse
ie PDA can do 1 comparison ie $\{ww^R\}$
CSL - can do more than one string matching

$\{ww^R \mid w \in \{0,1\}^*\}$ - CFL

$\{ww^R \mid w \in \{0,1\}\}$ - Regular

$\{ww^R w \mid w \in \{0,1\}^*\}$ - Not CFL & not regular

$\{a^{2n} b^n c^n \mid n \geq 0\}$ not CFL as require two comparison

$\{a^m b^n \mid m+n \geq 1\}$ regular

- $\{a^n b^m c^m d^n \mid m,n \geq 0\}$ - CFL as n, m has 1-1 comparison.

You can do one comp. on one variable in PDA but you can do 2 comp at two diff variable.

EXCELLENT

$\{a^n b^m c^n d^m\}$ → 2 comp on n
2 comp on m
↳ Not a PDA

$\{a^m b^n c^n d^m\}$ → PDA or CFL

$\{a^n b^m c^n\}$ - PDA as | when b came, donot push it to stack. }

$\{a^{m+n} b^m c^n \mid m,n \geq 0\}$ - accepted by PDA

Language not accepted by CSL = which are recursive

$\{(ab)^n (cd)^n \mid n \geq 0\}$ ~~Regular~~ PDA

$\{a^m b^n \mid m,n \geq 0\}$ - Regular
$\{a^{m+1} b^{3n+1} \mid m,n \geq 0\}$ - CFL

- $\{ww^R \mid w \in \{0,1\}^*\}$ - CFL
$\{ww^R \mid w \in \{0,1\}\}$ - Regular
$\{ww^R \mid w \in \{0,1\}^*, |w| \leq 10\}$ Regular
$\{w_1 w_2^R \mid w_1, w_2 \in (0,1)^*\}$ → Regular
$(0+1)^* (0+1)^*$

$L = \{wxw^R \mid w \in (0+1)^*, x \in (0+1)^*\}$ - Regular $\lceil w(0+1)^* w^R = (0+1)^* \rceil$
as
$\{x \mid x \in (0+1)^*\}$ is subset of L
if $w = \epsilon$
$\epsilon x \epsilon = x = (0+1)^*$

Page No.
Date : / /

$\{ wxw^R \mid w \in (0,1)^* \; x \in (0,1) \}$ - CFL

* All palindrome on binary alphabet are CFL
All palindrome on unary alphabet are regular as all string will be palindrom

$\{ wxw^R \mid w \in (0)^* \; x \in \{0\} \}$
↳ Regular

Regular is not closure over subset, superset, infinite intersection, infinite union, infinite difference.

If Regular is closure over subset having then $(0+1)^*$ is regular & then any language can become regular as $(0^n 1^n 0^n)$ ab.

Superset - $L = \phi$ is closet R.L.
then superset of L is covering all language
∴ All language cannot be regular

Infinite Intersection & Union -

$\{a^n b^n \mid n \geq 0\}$ → also regular
If U is regular then $\{\phi\} \cup \{ab\} \cup \{aabb\} \ldots$
will make it regular

infinite Intersection - $A \cap B = \overline{\overline{A} \cup \overline{B}}$
↓ Not regular as U is also not regular

Infinite difference $A - B = A \cap \overline{B}$ → Not regular

EXCELLENT

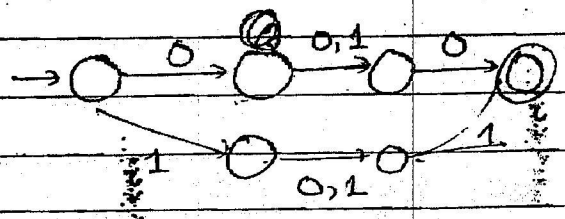
$x(0+1)^+$

Page No.
Date : / /

$\{wxw^R \mid w \in (0,1)^*, x \in \{0,1\}^*\}$ – Regular

$\{wxw^R \mid w \in (0,1) \quad x \in \{0,1\}\}$

$= 0(0+1)0 + 1(0+1)1$



$\{wxw^R \mid w \in \{0,1\}^*, x \in \{0,1\}^+\}$ – Regular as x $(0+1)^+$ cover all cases

$(0+1)^*$ ∈ $(0+1)^+$

$\{wxw^R \mid w \in (0,1)^+ \quad x \in (0,1)^+\} \rightarrow$ Regular

$\{wxw^R \mid w \in (0,1)^+ \quad x \in (0,1)^*\}$ – Regular

~~$\{wxw \mid$~~

$\{ww \mid w \in (0+1)^*\}$ – CSL

$(0+1)^*(0+1)^*$

$(0+1)^*$

$\{ww \mid w \in (0,1)\}$ – Regular

$\{wxw \mid w \in (0,1)^* \; x \in (0,1)^*\}$ – Regular as you can put every thing as x

$\{wxw \mid w \in (0,1)^*, x \in (0,1)^+\}$ – Regular

$\{wxw \mid w \in (0,1)^+ \quad x \in (0,1)^*\}$ – CSL

$0x0 + 1x1$

$0(1+0)^*0 + 1(1+0)^*1$

as but $\{w = 0,1$ not covered

0 not covered by either $0(1+0)^*0$ / $1(1+0)^*1$ EXCELLENT

Page No.
Date : / /

$\{ wxw \mid w \in (0,1)^+ \quad x \in (0,1)^+ \}$

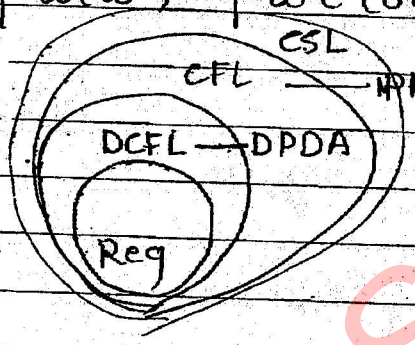
$0(0+1)^+0 + 1(0+1)^+\phi$ → Also CSL

$\{ xwwR \mid w, x \in (0,1)^+ \}$ — CFL

$x00 + x11$ not covering $w=01$

$wwRx \mid w, x \in (0,1)^+]$ — ~~CFL~~ ~~Regular~~ CFL

$\{ w(w^R)^* \mid w \in (0,1)^* \}$ - Regular.



DPDA is having less power than NPDA.

CLOSURE PROPERTIES

- Union
- Intersection
- ~~Identity~~ Complement
- concatenation
- Iteration / Closure
- Reversal of L / Transpose of L
- ~~Homo~~ Homomorphism of L
- Inverse Homomorphism.

EXCELLENT

Page No.
Date : / /

01 $(\ )^+$ @ w.ε (0∓)

	Regular	DCFL	CFL	CSL	REC	RE
∪	✓	×	✓	✓	✓	✓
∩	✓	×	×	✓	✓	✓
¯	✓	✓	×	✓	✓	×
·	✓	×	✓	✓	✓	✓
*	✓	×	✓	✓	✓	✓
L^R	✓	×	✓			
$h(L)$	✓	×	✓			
$h^{-1}(L)$	✓	✓	✓			
L∪R	✓	✓	✓	✓	✓	
L∩R	✓	✓	✓		✓	
L-R	✓	✓	✓	✓	✓	

No inform available

Regular language is closed for every op^n except $\subseteq$, $\supseteq$, in$\cap$ in $\cup$, in $\bar{}$.

No language is closed for $\subseteq$, $\supseteq$, in $\cap$, in $\cup$, in $\bar{}$.

R → Regular language

~~Excel~~

Every language is closed for $\cup$, $\cap$, $\bar{}$ with regular expression

$L_1 \cup L_2$ = X (may or may not be DCFL)
DCFL DCFL

$L_1 \uparrow \cup L_2 \uparrow$ → push up
CFL CFL = ✓ (may

∴ L_1, L_2 Union give CFL.

If you wanna correct answer push to known the boundary.

as $\overline{L_1}$ = X (may) comp of CFL is CSL
DCFL

$L_1 \uparrow$ $\bar{L_1}$
CFL → CSL ⇒ ✓

EXCELLENT