Hence the equivalent Grammar is

$A_1 \rightarrow aA_1A_2 \mid bA_2 \mid aA_1Z_2A_2 \mid bZ_2A_2 \mid a$

$A_2 \rightarrow aA_1 \mid b \mid aA_1Z_2 \mid bZ_2$

$Z_2 \rightarrow aA_1A_1 \mid bA_1 \mid aA_1Z_2A_1 \mid bZ_2A_1 \mid aA_1A_1Z_2 \mid bA_1Z_2 \mid aA_1Z_2A_1Z_2 \mid bZ_2A_1Z_2$

Q3: Convert into Greibach NF

$S \rightarrow AB$

$A \rightarrow BS \mid b$

$B \rightarrow SA \mid a$

Ans: Rename the variable $S = A_1$, $A = A_2$, $B = A_3$

$A_1 \rightarrow A_2A_3$ $\quad 1 < 2$

$A_2 \rightarrow A_3A_1 \mid b$ $\quad 2 < 3$

$A_3 \rightarrow A_1A_2 \mid a$ $\quad 3 \not< 1$

$A_3 \rightarrow A_1A_2 \mid a$ is not in required form.

$A_3 \rightarrow A_2A_3A_2 \mid a$

$A_3 \rightarrow A_3A_1A_3A_2 \mid bA_3A_2 \mid a$

Now A_3 productions are

$A_3 \rightarrow bA_3A_2 \mid a \mid bA_3A_2Z_3 \mid aZ_3$

Z_3 pdn are

$Z_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z_3$

Substitute in

$A_2 \rightarrow A_3A_1 \mid b$

So A_2 productions are

$A_2 \rightarrow bA_3A_2A_1 \mid aA_3 \mid bA_3A_2Z_3A_1 \mid aZ_3A_1 \mid b$

Substitute in

$A_1 \rightarrow A_2A_3$

So A_1 pdns are

$A_1 \rightarrow bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2Z_3A_1A_3 \mid aZ_3A_1A_3 \mid bA_3$

Substitute in

$Z_3 \rightarrow A_1A_3A_2 \mid A_1A_3A_2Z_3$

$Z_3 \rightarrow bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2Z_3A_1A_3A_3A_2 \mid aZ_3A_1A_3A_3A_2 \mid bA_3A_3A_2$

$Z_3 \rightarrow bA_3A_2A_1A_3A_3A_2Z_3 \mid aA_1A_3A_3A_2Z_3 \mid bA_3A_2Z_3A_1A_3A_3A_2Z_3 \mid aZ_3A_1A_3A_3A_2Z_3 \mid bA_3A_3A_2Z_3$

Q4 Convert into GNF

$E \to E+T \mid T$, $T \to T*F \mid F$, $F \to (E) \mid a$

Ans: 1) Eliminate unit prods

$E \to E+T \mid T*F \mid (E) \mid a$

$T \to T*F \mid (E) \mid a$

$F \to (E) \mid a$

2) Rename the variables

$+ = A$, $* = B$, $) = D$

$E \to EAT \mid TBF \mid (ED \mid a$

$T \to TBF \mid (ED \mid a$

$F \to (ED \mid a$

A, B, D, E, F, T are renamed as $A_1, A_2, A_3, A_4, A_5, A_6$

$A = A_1$ $B = A_2$ $D = A_3$ $E = A_4$ $F = A_5$ $T = A_6$

$A_4 \to A_4A_1A_6 \mid A_6A_2A_5 \mid (\check{A}_4A_3 \mid \check{a}$

$A_6 \to A_6A_2A_5 \mid (\check{A}_4A_3 \mid \check{a}$

✓ $A_5 \to (\check{A}_4A_3 \mid \check{a}$

✓ are in required form

A_6 pdns are

✓ $A_6 \rightarrow CA_4A_3 \mid a \mid CA_4A_3Z_6 \mid aZ_6$

Z_6 pdns are

$Z_6 \rightarrow A_2A_5 \mid A_2A_5Z_6$

Substitute in A_6

A_4 pdns are

$A_4 \rightarrow A_6A_2A_5 \mid A_4A_1A_6 \mid CA_4A_3 \mid a$

✓ $A_4 \rightarrow CA_4A_3A_2A_5 \mid aA_2A_5 \mid CA_4A_3Z_6A_2A_5 \mid aZ_6A_2A_5 \mid CA_4A_3 \mid a \mid A_4A_1A_6$

Z_4 pdns are

$Z_4 \rightarrow A_1A_6 \mid A_1A_6Z_4$

✓ $Z_4 \rightarrow +A_6 \mid +A_6Z_4$

✓ $Z_6 \rightarrow *A_5 \mid *A_5Z_6$

Q5: Convert into GNF

$S \to AB \mid BC$

$A \to aB \mid bA \mid a$

$B \to bB \mid cC \mid b$

$C \to c$

Ans: $S \to aBB \mid bAB \mid aB \mid bBC \mid cCC \mid bC$

$C \to c$

$A \to aB \mid bA \mid a$

$B \to bB \mid cC \mid b$

Rename variable $S = A_1$, $A = A_2$, $B = A_3$, $C = A_4$

$A_1 \to A_2A_3 \mid A_3A_4$

$A_2 \to aA_3 \mid bA_2 \mid a$

$A_1 \to aA_2A_3 \mid bA_2A_3 \mid aA_3 \mid bA_3A_4 \mid cA_4A_4 \mid bA_4$

$A_3 \to bA_3 \mid cA_4 \mid b$

$A_4 \to c$

Q6: Reduce into GNF

$S \to SS$

$S \to OS1 \mid O1$

Ans: $S \to SS$

$S \to C_0SC_1 \mid C_0C_1$

$S \to C_0D$ $\quad C_0 \to 0$

$D \to SC_1$ $\quad C_1 \to 1$

$S \to C_0C_1$

$S \to SS$

Rename variables

$S = A_1$ $\quad C_0 = A_2$ $\quad C_1 = A_3$

$D = A_4$

$A_1 \rightarrow A_1 A_1$ $\quad A_2 \rightarrow 0$

$A_1 \rightarrow A_2 A_3$ $\quad A_3 \rightarrow 1$

$A_1 \rightarrow A_2 A_4$

$A_4 \rightarrow A_1 A_3$

~~$A_4 \rightarrow A_1 A_1 A_3$~~

$A_1 \rightarrow A_1 A_1 \mid A_2 A_3 \mid A_2 A_4$

$A_4 \rightarrow A_1 A_3$

$A_1 \rightarrow A_1 A_1 \mid 0 A_3 \mid 0 A_4$

A_1 pdn are

✓ $A_1 \rightarrow 0A_3 \mid 0A_4 \mid 0A_3 Z_1 \mid 0A_4 Z_1$

Z_1 pdn are

$A_1 \rightarrow A_1 \mid A_1 Z_1$

Z_1 pdn are

✓ $Z_1 \rightarrow 0A_3 \mid 0A_4 \mid 0A_3 Z_1 \mid 0A_4 Z_1 \mid 0A_3 Z_1 \mid 0A_4 Z_1 \mid$
$0A_3 Z_1 Z_1 \mid 0A_4 Z_1 Z_1$

A_4 pdn.

$A_4 \rightarrow A_1 A_3$

✓ $A_4 \rightarrow 0A_3 A_3 \mid 0A_4 A_3 \mid 0A_3 Z_1 A_3 \mid 0A_4 Z_1 A_3$

Q7: $S \rightarrow AB$

$A \rightarrow BSB \mid BB \mid b$

$B \rightarrow aAb \mid a$

Ans: Not in CNF

✓ $S \rightarrow AB$

✓ $A \rightarrow BD \mid BB \mid b$ $\quad S=A_1,\ A=A_2\ \ B=A_3$

✓ $D \rightarrow SB$ $\quad D=A_4\ \ E=A_5$

$B \rightarrow BAB \mid a$

~~a~~ ✓ $B \rightarrow BE \mid a$

~~b~~ ✓ $E \rightarrow AA$

$A_1 \rightarrow A_2 A_3$ $\quad A_3 \rightarrow a$

$A_2 \rightarrow A_3 A_4 \mid A_3 A_3 \mid b$ $\quad A_2 \rightarrow b$

$A_4 \rightarrow A_1 A_3$

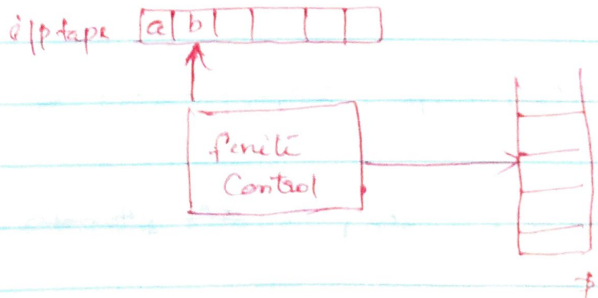
$A_2 \rightarrow aA_4 \mid aA_3 \mid b \mid$ ~~$aA_4Z_2 \mid aA_3Z_2 \mid bZ_2$~~

$A_1 \rightarrow aA_4A_3 \mid aA_3A_3 \mid bA_3 \mid$ ~~$aA_4Z_2A_3 \mid aA_3Z_2A_3 \mid bZ_2A_3$~~

$A_4 \rightarrow aA_4A_3A_3 \mid aA_3A_3A_3 \mid bA_3A_3$

Pushdown Automata (PDA)

* PDA recognizes Context free language which is type 2
* It's an extension of C-NFA with an infinite storage called stack
* It recognizes all regular, non-regular language and a ~~proper subset of~~ CFL. It's non-deterministic.
* It can accept a set of strings by reaching final state or emptying the stack whatever be the state
* It can store infinite inputs and compare words that $a^n b^n$, ww^R, which is not regular can be accepted by PDA
* PDA is non-deterministic in nature.
* PDA consists of ~~infinite~~ input tape, finite control and a stack storage



It has deterministic PDA which accepts all regulars and a proper subset of CFL.

PDA can be represented by $P = (Q, \Sigma, \Gamma, \delta, s, F, z_0)$

Q = Set of states

Σ - i/p alphabet

Γ - finite set of pushdown symbols

s - Start state

F - final state

δ - transition fn that maps $(Q \times \Sigma^* \times \Gamma^*) \rightarrow (Q \times \Gamma^*)$

z_0 - initial symbol of pushdown store

* Stack operations are performed in PDA, push the symbol, pop the symbol, a symbol on the stack can be replaced by another symbol, do nothing. It will

* $q_1 \xrightarrow{a, \epsilon \rightarrow c} q_2$ q_1 consumes a i/p, top of the stack ϵ, push operation, c is top of the stack → push operation

* $q_1 \xrightarrow{a, c \rightarrow \epsilon} q_2$ pop

* $q_1 \xrightarrow{a, \epsilon \rightarrow \epsilon} q_2$ do nothing

* $q_1 \xrightarrow{a, b \rightarrow c} q_2$ $(q_1, a, b) \rightarrow (q_2, c)$

* Acceptance by reaching final state and reading entire i/p $(q_0, w, z_0) \xrightarrow{*} (q_f, \epsilon, \alpha)$, remains some contents in the stack.
* Acceptance by reading entire i/p and emptying the stack $(q_0, w, z_0) \xrightarrow{*} (q_0, \epsilon, \epsilon)$

Instantaneous Description (ID)

* At a particular instant, what will be current i/p, current state and current stack symbol

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

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

$(q_0, \epsilon, z_0) \vdash (q_f, z_0)$

q_0, a, z_0

Designing of PDA

1. Design a PDA which accepts $L = \{ w \in \{a,b\}^* \mid w$ has equal no: of a's & b's $\}$

A: Hint: Every string having equal no: of a's and b's and no concern of position, a can come first or b can

$P = (Q, \Sigma, \delta, T, S, F, Z_0)$

$Q = \{q_0\}$ $\quad T = \{a, b, Z_0\}$

$\Sigma = \{a, b\}$ $\quad F = \{q_f\}$.

1 $(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

2 $(q_0, b, Z_0) \rightarrow (q_0, bZ_0)$

3 $(q_0, a, a) \rightarrow (q_0, aa)$

4 $(q_0, b, b) \rightarrow (q_0, bb)$

5 $(q_0, a, b) \rightarrow (q_0, \epsilon)$

6 $(q_0, b, a) \rightarrow (q_0, \epsilon)$

7 $(q_0, \epsilon, Z_0) \rightarrow (q_0, \epsilon)$ By Emptying the Stack

8 $(q_0, \epsilon, Z_0) \rightarrow (q_f, Z_0)$

		State	I/P	Stack	Transition used	
Ibs	1	q_0	abbbaaba	z_0	1	
	2	q_0	bbbaaba	az_0	1	
	3	q_0	bbaaba	z_0	6	
	4	q_0	baaba	bz_0	2	
	5	q_0	aaba	bb	4	
	6	q_0	aba	bz_0	6	
	7	q_0	ba	z_0	6	
	8	q_0	a	bz_0	2	
	9	q_0	~~zc~~ $\in$	z_0	5	
	10	q_0	$\in$	$\in$	7	(either by 7 or
	11	q_f	$\in$	z_0	8	8)

Q_2: Design PDA for $L = \{wcw^r \mid w \in \{a,b\}^*\}$

$P = (Q, \Sigma, \delta, \Gamma, q_0, F, Z_0)$

$Q = \{q_0, q_1\}$

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

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

$F = \{f\}$.

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

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

$(q_0, b, z_0) \vdash (q_0, bz_0)$

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

$(q_0, a, b) \vdash (q_0, ab)$

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

$(q_0, c, a) \vdash (q_1, a)$

$(q_0, c, b) \vdash (q_1, b)$

$(q_0, c, z_0) \vdash (q_1, z_0)$

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

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

$(q_1, \epsilon, z_0) \vdash (q_f, z_0)$

Q3: Design a PDA which $L = \{a^n b^n\}$

$P = (Q, \Sigma, \Gamma, \delta, S, F, z_0)$

$Q = \{q_0, q_1, \}$

$S = \{q_0\}$

$F = \{q_f\}$

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

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

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

$(q_1, b, a) \vdash (q_1, \epsilon)$

$(q_1, \epsilon, z_0) \vdash (q_1, \epsilon, \epsilon)$

Q4. Design a PDA for R.E $r = 0^* 1^*$

$Q = \{q_0, q_1\}$

$F = \{q_0, q_1\}$

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

$\Gamma = \{Z_0, Z'\}$

$(q_0, \epsilon, Z_0) \to (q_0, Z'Z_0)$

$(q_0, 0, Z_0) \to (q_0, Z'Z_0)$

$(q_0, 1, Z_0) \to (q_1, Z'Z_0)$

$(q_1, 1, Z_0) \to (q_1, Z'Z_0)$

$(q_1, 0, Z_0) \to (q_2, Z'Z_0)$

or

$(q_0, \epsilon, Z_0) \to (q_0, Z_0)$

$(q_0, 0, Z_0) \to (q_0, Z_0)$

$(q_0, 1, Z_0) \to (q_1, Z_0)$

$(q_1, 1, Z_0) \to (q_1, Z_0)$

$(q_1, 0, Z_0) \to (q_2, Z_0)$

Q5. Design PDA for $L = 0^+ 1^+$

$(q_0, 0, Z_0) \to (q_0, Z_0)$

$(q_0, 1, Z_0) \to (q_1, Z_0)$

$(q_1, 1, Z_0) \to (q_1, Z_0)$

$(q_1, 0, Z_0) \to (q_2, Z_0)$

$Q = \{q_0, q_1, q_2\}$

$\delta = \{q_0\}$

$F = \{q_1\}$

$\Gamma = \{Z_0\}$.

Q6: Design a PDA for $L = \{ a^n b^{n+1} \mid n = 1, 2, 3, \ldots \}$

$(q_0, a, z_0) \vdash (q_0, a z_0)$

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

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

$(q_1, b, a) \vdash (q_1, \epsilon)$

$(q_1, b, z_0) \vdash (q_2, z_0)$

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

$Q = \{ q_0, q_1, q_2 \}$

$S = \{ q_0 \}$

$F = \{ q_2 \}$

$\Gamma = \{ z_0 \}$

Q7: Design a PDA $\{ a^n b^m a^n \mid m, n \geq 1 \}$ by null store

D $(q_0, a, z_0) \vdash (q_0, a z_0)$

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

$(q_0, b, a) \vdash (q_1, aa)$

$(q_1, b, a) \vdash (q_1, aa)$

$(q_1, a, a) \vdash (q_2, \epsilon)$

$(q_2, a, a) \vdash (q_2, \epsilon)$

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

Q8 Design a PDA accepting all palindromes $\{a, b\}$

A: palindrome is of odd length and even length. $w = w^R$.
If w is of odd length, middle symbol is not compared with anyone. If its even length, 1st half last symbol is compared with 2nd half 1st symbol if matches, proceeds, otherwise halts

even → odd p.

1 $\delta(q_0, a, z_0) \to (q_0, az_0)$ (q_1, z_0)
2 $\delta(q_0, a, a) \to (q_0, aa)$ (q_1, a)
3 $\delta(q_0, b, z_0) \to (q_0, bz_0)$ Reading i/p = (q_1, z_0)
4 $\delta(q_0, b, b) \to (q_0, bb)$ pushing onto stack (q_1, b)
5 $\delta(q_0, a, b) \to (q_0, ab)$ (q_1, b)
6 $\delta(q_0, b, a) \to (q_0, ba)$ (q_1, a)
7 $\delta(q_0, \epsilon, z_0) \to (q_1, z_0)$ (Reached 1st half)
8 $\delta(q_0, \epsilon, a) \to (q_1, a)$
9 $\delta(q_0, \epsilon, b) \to (q_1, b)$
10 $\delta(q_1, a, a) \to (q_1, \epsilon)$
11 $\delta(q_1, b, b) \to (q_1, \epsilon)$
12 $\delta(q_1, \epsilon, z_0) \to (q_2, z_0)$ reaches final state.

Q.10 Design a PDA for $a^n b^{2n}$

1 $(q_0, a, z_0) \rightarrow (q_1, az_0)$

2 $(q_1, a, a) \rightarrow (q_1, aa), (q_2, \epsilon, a)$

3 $(q_2, b, a) \rightarrow (q_3, \epsilon)$

4 $(q_3, b, a) \rightarrow (q_2, \epsilon)$

5 $(q_4, \epsilon, z_0) \rightarrow (q_4, \epsilon)$ by emptying the stores

③ $(q_1, b, a) \rightarrow (q_2, \epsilon)$

④ $(q_2, b, a) \rightarrow (q_3, \epsilon)$

⑤ $(q_3, b, a) \rightarrow (q_2, \epsilon)$

⑥ $(q_3, \epsilon, z_0) \rightarrow (q_4, z_0)$

aabbbb

aaabbbbbb

Q.11: Design a PDA for $\{a^n b^m c^n \mid m, n \geq 1\}$

1 $(q_0, a, z_0) \rightarrow (q_0, az_0)$

2 $(q_0, a, a) \rightarrow (q_0, aa)$

3. $(q_0, b, a) \rightarrow (q_1, a)$

4 $(q_1, b, a) \rightarrow (q_1, a)$

5) $(q_1, c, a) \rightarrow (q_2, \epsilon)$

6 $(q_2, c, a) \rightarrow (q_2, \epsilon)$

7 $(q_2, \epsilon, z_0) \rightarrow (q_2, \epsilon)$.

* Q: Construct a PDA $L = \{ a^i b^j c^k \mid i=j \text{ or } j=k, i>0, j>0, k>0 \}$ by final state

$a^i b^i c^k$ where $i=j$ if 0, $a^0 b^0 c^k$

$\delta(q_0, \epsilon, z_0) = \{(q_1, z_0), (q_2, z_0), (q_3, z_0)\}$

$\delta(q_4, \epsilon, z_0) \to (q_1, z_0)$

1. $\delta(q_1, c, z_0) \to (q_1, z_0)$
2. $\delta(q_2, a, z_0) \to (q_2, aZ_0)$
3. $\delta(q_2, a, aZ_0) \to (q_2, aa)$
4. $\delta(q_2, b, a) \to (q_4, \epsilon)$
5. $\delta(q_4, b, aZ_0) \to (q_4, \epsilon Z_0)$
6. q_1 for $a^0 b^0 c^k$ & q_2 for $a^i b^i c^k$, q_3 for $a^i b^j c^k$ where $j=k$, $a^i b^j c^j$, $a^i b^0 c^0$.
7. $\delta(q_3, a, z_0) \to (q_3, aZ_0), (q_5, \epsilon, z_0)$
8. $\delta(q_5, b, z_0) \to (q_6, bZ_0)$
9. $\delta(q_5, b, bZ_0) \to (q_6, bb)$
10. $\delta(q_6, c, b) \to (q_4, \epsilon)$
11. $\delta(q_4, c, bZ_0) \to (q_5, \epsilon)$
12. $\delta(q_4, \epsilon, z_0) \to (q_5, \epsilon)$

$F = \{q_3, q_4\}$

$S = \{q_0\}$

$\Gamma = \{a, b, z_0\}$

$(q_3, \epsilon, z_0) \to (q_f, z_0)$

Q12: Design a PDA for $L = \{ a^m b^n, m<n \}$

1. $(q_0, a, z_0) \rightarrow (q_0, az_0)$
2. $(q_0, a, a) \rightarrow (q_0, aa)$
3. $(q_0, b, a) \rightarrow (q_1, \epsilon)$
4. $(q_1, b, a) \rightarrow (q_1, \epsilon)$
5. $(q_1, b, z_0) \rightarrow (q_2, z_0)$
6. $(q_2, b, z_0) \rightarrow (q_2, z_0)$
7. $(q_2, \epsilon, z_0) \rightarrow (q_f, z_0)$

Q13: Design a PDA for $L = \{ a^m b^n, m>n \}$

1. $(q_0, a, z_0) \rightarrow (q_0, az_0)$
2. $(q_0, a, a) \rightarrow (q_0, aa)$
3. $(q_0, b, a) \rightarrow (q_1, \epsilon)$
4. $(q_1, b, a) \rightarrow (q_1, \epsilon)$
5. $(q_1, \epsilon, a) \rightarrow (q_2, a)$
6. $(q_2, \epsilon, a) \rightarrow (q_f, a)$

Pumping Lemma for Context-Free Language

* Similar to R.L pumping Lemma
* The string z can be divided into 5 parts, we pump 2nd and 4th substrings and that also in z.
* used to prove certain L are not context-free.

Theorem

* Let L be a CFL. Then there exist a constant n such that if z is any string such that $|z|$ is atleast n, we can write $z = uvwxy$ subject to foll. conditions

1. $|vwx| \leq n$, ie middle portion is not so long
2. $|vx| \neq \epsilon$, v & x are substring is to be pumped ie atleast one string must not be empty.
3. for all $i \geq 0$, uv^iwx^iy is in L. ie 2 strings must be pumped any no. of times, ie also a member of L.

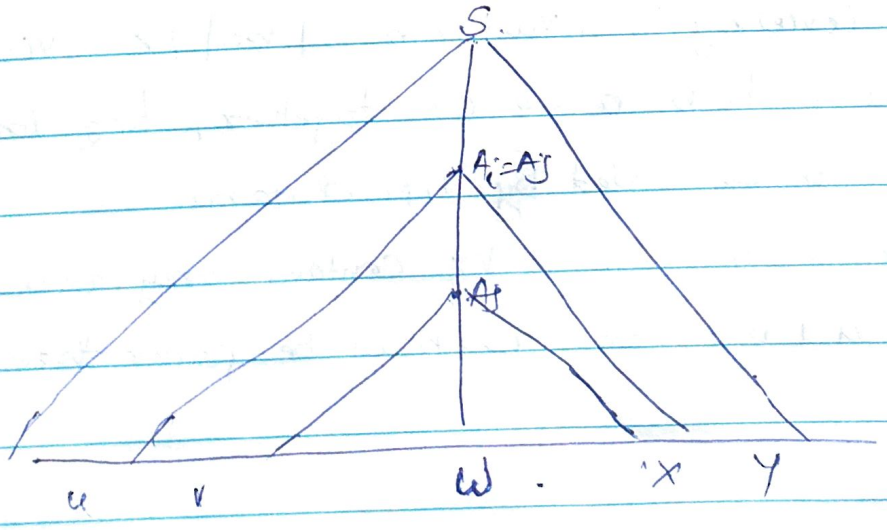
Proof

Take A CNF G will generate $L - \{\epsilon\}$ Bcz we take a string z ie sufficiently large greater than n

* Let G have m variables Choose $n = 2^m$. Suppose z is length of atleast n

* Any parse tree whose longest path is of length m or less must have a yield of length 2^{m-1} or less. $2^{m-1} = n/2$ or less. Such a parse tree can't have a yield z, bez z is too long. Thus, any parse tree having a yield z has a path length of atleast m+1

* The longest path in the tree for z, where k is atleast m and path is of length k+1. Since $k \geq m$, there are atleast m+1 occurrence of variables $A_0, A_1 \ldots A_k$. As there are only m different variables in V, atleast 2 of last m+1 must be same variable



Application of pumping Lemma

- It's used for proving certain Languages are not context free.
1. Pick $|z| \geq n$
2. Break z into uvwxy, $|vwx| \leq n$ and $|vx| \neq \epsilon$
3. Choose i $uv^iwx^iy \notin L$.

Eg: S.T $a^n b^n c^n \mid n \geq 1$ is not CFL

Ans: all strings $a^+b^+c^+$ with equal no: abc, aabbcc

Suppose L is CFL.

Then there is an integer n. Pick $z = a^n b^n c^n$.

$z = uvwxy$, $|vwx| \leq n$, $|vx| \neq \epsilon$, vwx can't envolve both a's & c's together, bc's last a & 1st c is separated by n+1 distance

vwx has no c, vx contains only a's b's, atleast one a & b's So c must be fewer than a's & b's

Suppose L is CFL [illegible]

$|vwx| \leq n$, $|vx| \neq \epsilon$. Choose $|vx| = m$

We know that $m \leq n$., Choose $|uv^2wx^2y| = |uvwxy| + |vx|$
$n^2 + m$

$n^2 < n^2 + m \leq n^2 + n < (n+1)^2$

which clearly lies b/w $n^2 < (n+1)^2$

eg3. Let $L = ww$ $w = \{0,1\}^*$

$z = 0^n 1^n 0^n 1^n = uvwxy$, $|vwx| \leq n$, $|vx| \neq \epsilon$

Suppose vwx contains only 0's 0^{n+1}

Ambegous Grammar

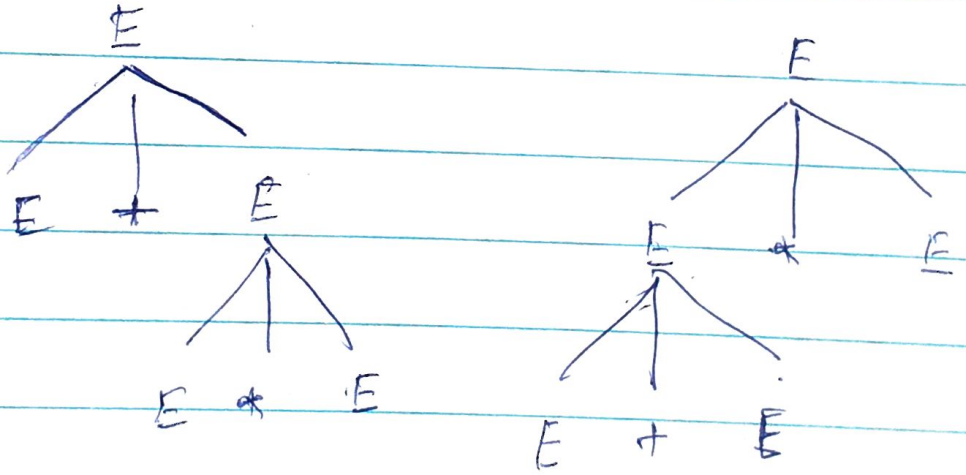
It produce more than one parse tree for this type of grammar. It's said to be ambegous grammar

1) $E \rightarrow E+E \mid E*E$

Ans. Give a sentential for $E+E*E$

$E \Rightarrow E+E \Rightarrow E+E*E$

$E \Rightarrow E*E \rightarrow E+E*E$



Removing ambegulty - Removing left recursion and specify the precedence.

Inherent Ambiguity

* For every palin, it generates one diff parse tree
* A CFL is said to be inherently ambiguous if all its grammars are ambiguous

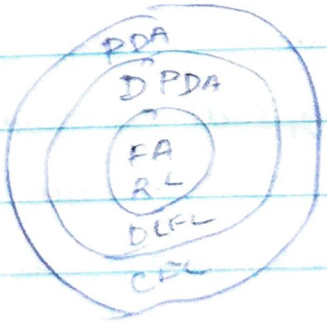
Deterministic PDA

There will be no choice of 2 kinds

A PDA is deterministic if

1) $\delta(q, a, x)$ has atmost one member for any q in Q a in Σ or $a = \epsilon$ and x in Γ
2) If $\delta(q, a, x)$ is non-empty, for some a in Σ, then $\delta(q, \epsilon, x)$ must be empty

* DPDA accept all regular L and a proper subset of CFL. wcw^r will accept, ww^r won't accept

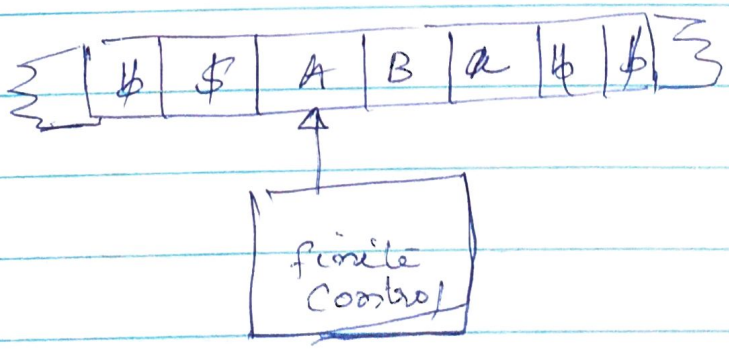


* DPDA accept unambiguous grammars

Turing machines

- H's ~~dem~~ Automata that recognizes type 0 grammars and accept Recursive and recursive enumerable language
- H has unrestricted grammar where $|\alpha\beta\gamma| \rightarrow |\alpha\lambda\gamma|$ No restrictions on the grammar.
- There are some problems that can't be solved by a computer. These type of problems are called undecidable problems. We can't write algorithm for this
- But TM can use to solve decidable & undecidable problem
- In 1936 A.M. Turing has proposed turing m/c.
- CHURCH-TURING THESIS states that an effective procedure that can be carried by a computer or a human can be carried out by a turing m/c.
- An effective procedure is a step by step instructions to do a pgm. It may halt or mayn't halt. For eg: If we want to check a no: is prime >n
  1st pick n+1, check whether this is prime or not then go to n+2 & soooo. This will never end and go in a loop. So can't write an alg

* An algorithm is a step by step procedure and it will halt and definitely give an answer
* for eg: If a no: is prime or not, If no: is prime give answer yes, otherwise no. So we can write algorithm for this
* Turing mc can be defined by finite control and infinite tape
* TM can be represented by 7-tuples $(Q, \Sigma, \delta, q_0, B, \Gamma, F)$ where Q - finite set of states
Σ - finite i/p alphabet
δ - transition fn
q_0 - start state
B - set of Blank symbol
Γ - set of tape symbols



Next move is defined by state and current i/p

$\delta(q, a) \rightarrow (q, X, R/L)$

(state, tape symbol, direction of move)

state i/p

* TM can act as 1) i/p/o/p device 2) Whether a string is a member of this Language? TM acceptors 3) Computational device
* TM has an infinite tape which store finite no: of symbols. i/p and o/p are stored in same tape. Rest of the space is written as blank symbol. Both ends will grow infinitely. tape is divided into one cell each and each cell contains one symbol at a time.
* R/W head can move leftwards or Rightwards one block at a time based on finite control.
* After the pbm, TM will enter into final state.

1 Design a TM for $L = a^n b^n$

Ans: Logic: Match 1st a with last b, 2nd a with 2nd last b and so on

; aaabbb; At q_0, Read a + write *, change state q_1, move right

↑ q_0

; *aabbb; At q_1, Read a + write a, Remain in same state and move right (i.e. skip all a's & b's until reaches end i.e. ;)

↑↑↑↑↑↑ $q_1 q_1 q_1 q_1 q_1 q_1$

; *aabbb; ; At q_1, read ; + write ; change state to q_2 & move left

↑ q_1

; *aabbb; ; At q_2, if u see b + write *, change to q_3 & move left

↑ q_2

; *aabb*; ; At q_3, if a or b +, write a or b, remain in q_3, move left until *

↑↑↑↑↑ $q_3 q_3 q_3 q_3 q_3$

; *aabb*; ; At q_3, if * + write *, change to q_0, move right

↑↑ $q_3 q_0$ At q_0, process repeats.

; **abb*; ; At q_1, read * +, write *, move left, change to q_2

↑↑↑↑ $q_1 q_1 q_1 q_1$

; **abb*; ; ***b**;

↑ q_2 ↑↑ $q_1 q_1$

; **ab**; ; ****b***;

↑↑↑ $q_3 q_3$ ↑ q_2

; **a b***; ; *****b****; At q_0, * → accept

↑ q_0 ↑↑ $q_3 q_0$

	a	b	*	;
write * for 1st a → q_0	q_1 * R		Accept	
write * for 2nd a → q_1	q_2 * R			
Skip all a's & b's → q_2	q_2 a R	q_2 b R	q_3 * L	q_3 ; L
write * for b → q_3	—	q_4 * L		
Skill all a's & b's until * → q_4	q_4 a L	q_4 b L	q_0 * R	

3. Design a TM for ww^R for palindromes

Ips: ; baabaab;

Logic: Mark 1st symbol & move till end. Check the last symbol with marked symbol. If same proceed. Move until a *.

; baabaab; q_0, Read b → write *, move right until sees ;

; * aabaab; At q_1, when ; → write ; Change to q_2 & move left

; * aabaa*; If b only, make it *

; * aabaa* ;

		a	b	*	;
write * for a →	q_0	q_1 * R		Accept	
Skip all a's & b's Right →	q_1	q_1 a R	q_1 b R	q_2 * L	q_2 ; L
write * for b →	q_2	—	q_3 * L		
← Skip a's & b's left	q_3	q_3 a L	q_3 b L	q_0 * R	

2 Design a TM for $L = \{a^{2n} b^n\}$

Ans: Logic: Read 1st a, write *; Read second a, write *, Skip all a's & b's and convert last b with * . Again Repeat the process

; a a a a b b ; Read q_0, a → write *, Right q_1

↑ q_0

; * a a a b b ; Read q_1, a → , write *, Right q_2

↑ q_1

; * * a a b b ; Skip all a's & b's until ; q_2, move left

↑ ↑ ↑ ↑ ↑ change to q_3

q_2 q_2

; * * a a b * ; q_3, Read b → write *, Change to q_4

↑ ↑ ↑ ↑

q_4 q_4 q_4

; * * a a b * ; q_4, * → q_0, *, R

↑ ↑ ↑ ↑

q_0 Repeat process

; * * a b a a * ;
↑

; * * a b a * * ;
↑ ↑ ↑ ↑

; * * * b * * * ;
↑ ↑

; * * * * * * * ;
↑ ↑

		a	b	*	;
Mark 1st symbol →	q_0	q_1 * R	q_4 * R	Accept (Error)	
Move right to make a	q_1	q_1 a R	q_1 b R	q_2 * L	q_2 ; L
Mark last a →	q_2	q_3 * L	—	Accept (Odd length centre a)	
← Move left	q_3	q_3 , a, L	q_3 b L	q_0 * R	—
Move right to make b →	q_4	q_4 a R	q_4 b R	q_5 * L	q_5 ; L
Write * for b →	q_5	—	q_3 * L	Accept (Odd length centre b)	

4 Design a TM for $a^n b^n c^n$

		a	b	c	x	y	z	;
Mark a →	q_0	$q_1 X R$	–	–	–	$q_4 Y R$	–	–
Mark b →	q_1	$q_1 a R$	$q_2 Y R$	–	–	$q_1 Y R$	–	–
Mark c →	q_2	–	$q_2 b R$	$q_3 Z L$	–	–	$q_2 Z R$	–
← Move left till u see X & Right move until →	q_3	$q_3 a L$	$q_3 b L$	–	$q_0 X R$	$q_3 Y L$	$q_3 Z L$	–
	q_4	–	–	–	–	$q_4 Y R$	$q_4 Z R$	$q_5 ; L$
	q_5	–	–	–	$q_5 a L$	$q_5 b L$	$q_5 c L$	Accept

; aaabbbccc;
↑ q_0

; xaabbbccc;
↑↑↑ q_1 q_1

; xaaybbbccc;
↑↑↑ q_2 q_2 q_2

; xaaybbzccc;
↑ ↑ q_3 q_3

; xaaybbzccc;
↑↑↑↑↑ q_3 q_3

; xaaybbzccc;
↑ q_0

; xxaaybbzccc;
↑↑↑ q_1 q_1 q_1

; xxayybbzccc;
↑↑↑ q_2 q_2 q_2

; xxayyybzccc;
↑ q_3

; xxayyybzzcc;
↑↑↑↑↑↑ q_3 q_3

; xxayyybzzzc;
↑ q_0

; xxxyyybzzzc;
↑↑↑ q_1 q_1 q_1

; xxxyyyzzzc;
↑↑↑↑ q_2 q_2 q_2 q_2

; xxxyyyzzzc;
↑↑↑↑ q_3 q_3

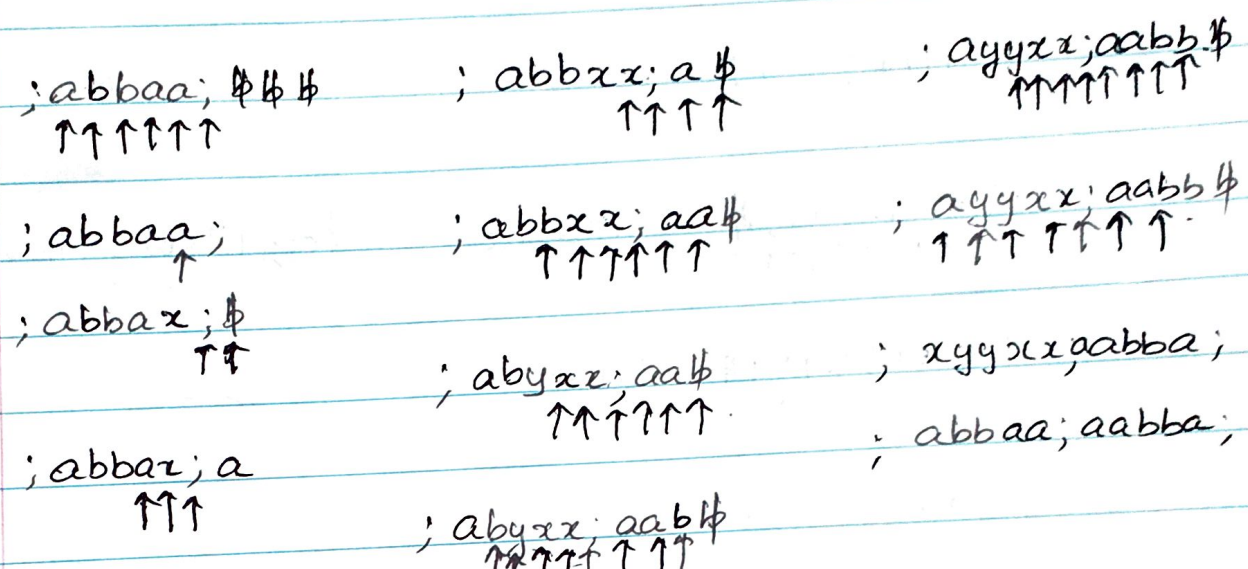
; xxxyyyzzzc;
↑ q_4

; xxxyyyzzzc;
↑↑↑↑↑↑↑↑ q_4 q_4 q_4 q_4 q_4 q_4 q_4 q_4

; xxxyyyzzzc;
↑ q_5

; aaabbbccc;

2 Design a TM to copy a string in reverse



		a	b	x	y	⊔	:
Move until;	q_0	$q_0 a R$	$q_0 b R$				$q_1 ; L$
Mask last a	q_1	$q_2 x R$	$q_5 y R$				
rightmove till space & mask a	q_2	$q_2 a R$	$q_2 b R$	$q_2 x R$	$q_2 y R$	$q_3 a L$	$q_2 ; R$
leftmove of generated till ;	q_3	$q_3 a L$	$q_3 b L$	—	—	—	$q_4 ; L$
leftmove of masked string	q_4	$q_2 x R$	$q_5 y R$	$q_4 x L$	$q_4 y L$	—	$q_6 ; R$
right- for b till space	q_5	$q_5 a R$	$q_5 b R$	$q_5 x R$	$q_5 y R$	$q_3 b L$	$q_5 ; R$
Recovers	q_6	$q_6 a R$	$q_6 b R$	$q_6 a R$	$q_6 b R$	Halt	$q_6 ; R$

generated

TM as i/p - o/p device

1. Design a TM to copy a string

Assumption: Remaining symbols filled with space

Ans: ;aabba; ␣␣␣␣␣ Read 1st and mark

;xabba; a ↑↑↑↑↑ when get ; , move right & write marked symbol

;xxbba;aa␣␣ ↑↑↑↑↑↑

;xxybba;aab␣␣ ↑↑↑↑↑↑

;xxyyba;aabb␣ ↑↑↑↑↑↑

;xxyyx;aabba

		a	b	x	y	␣	;
Mark a & b	q_0	q_1 x R	q_3 y R				q_4 ; R
1st a →	q_1	q_1 a R	q_1 b R			q_2 a L	q_1 ; R
←	q_2	q_2 a L	q_2 b L	q_0 x R	q_0 y R		q_2 ; L
1st b →	q_3	q_3 a R	q_3 b R			q_2 b L	q_3 ; R
	q_4	q_4 a R	q_4 b R	—	—	q_5 ; L	
	q_5	q_5 a L	q_5 b L				q_6 ; L
	q_6	—	—	q_6 a L	q_6 b L	—	Halt

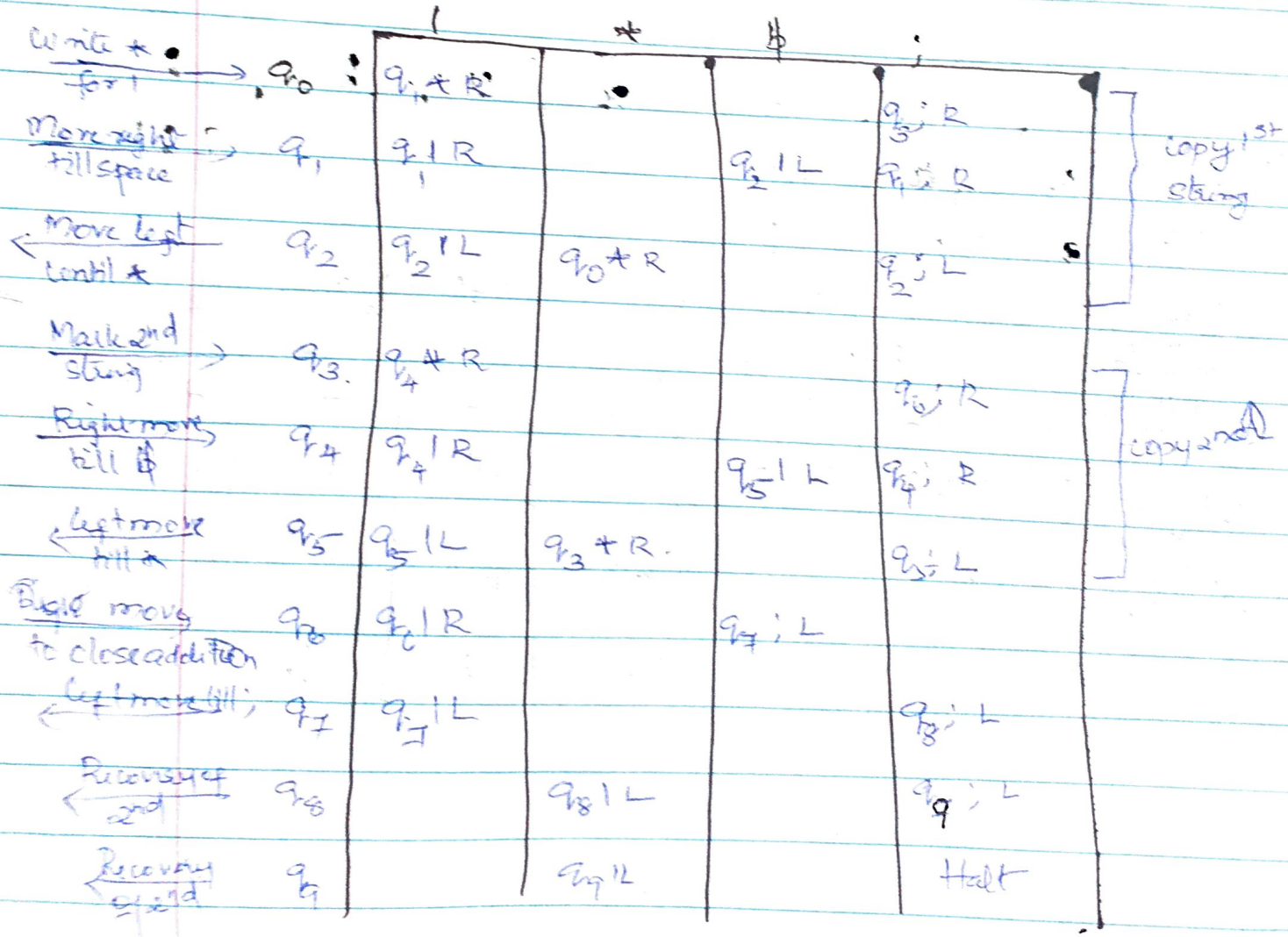
3 Design a TM to add 2 unary no:

; 1111 ; 111 ;

; * 111 ; 111 ; $ ↑↑↑↑↑↑↑↑ till space — Copy 1st string, then 2nd string

; * 111 ; 111 ; 1 $ ↑ ↑ ↑ ↑ ↑ ↑ till *

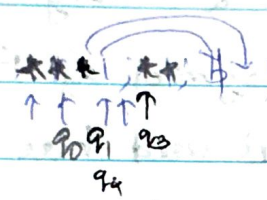
; * * 11 ; 111 ; 11 $



4 Design TM to subtract 2 no:

Assumption: 1st no > second

Logic: Mark 1st symbol & move till ; Mark 1st symbol of second and move Left



		1	*	⊔	;
Mark 1st →	q_0	q_1 * R			q_7 ; R
move right till end of 1st →	q_1	q_1 1 R			q_2 ; R
Mark 2nd & Start →	q_2	q_3 * L	q_2 * R	⊔	q_5 ; ⊔
← Left move till end of 1st	q_3		q_3 * L		q_4 ; L
← Skip 1st until *	q_4	q_4 1 L	q_0 * R		
move right till space →	q_5	q_5 1 R		q_6 1 L	
← Left move till end of 2nd	q_6	q_6 1 L			q_3 ; L
Right move to close subtraction →	q_7	q_7 1 R	q_7 * R	q_8 ; L	q_7 ; R
← Left move of ground	q_8	q_8 1 L			q_9 ; L
← Recover 1*	q_9		q_9 1 L		q_{10} ; L
← Recover 1st	q_{10}		q_{10} 1 L		Halt

5 Design a TM to multiply 2 unary no.

4x3

;1111;111;

Logic: Mark 1st symbol of 1st string & Mark each symbol of 2nd string and write into space. Return to 2nd of 1st string and write each symbol of 2nd string again & Continues the process till every symbol of 1st string is marked. Repitetive addition is used here. We are writing 3, 4 times

;1111;111;
↑

;*111;111;
↑↑↑↑

;*111;*11;1
↑↑↑↑

;*111;**1;11
↑↑↑↑

;*111;***;111
↑↑↑

;**11;***;111
↑↑↑↑↑

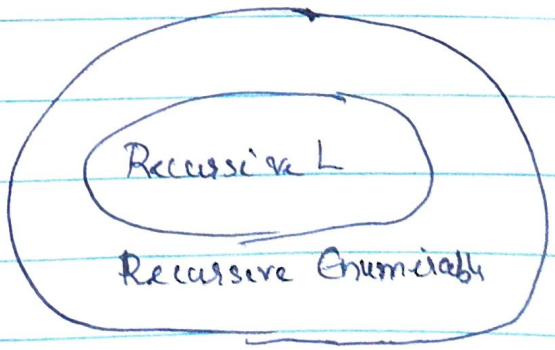
		1	*	#	;	
Mark first →	q_0	$q_1 * R$			$q_7 ; R$	
Move right till end of 1st	q_1	$q_1 1 R$			$q_2 ; R$	
Mark 2nd →	q_2	$q_3 * R$			$q_5 ; L$	
move till space →	q_3	$q_3 1 R$		$q_4 1 L$	$q_3 ; R$	Copy 2nd string
← left move till *	q_4	$q_4 1 L$	$q_2 * R$		$q_4 ; L$	
← left Recover 2nd.	q_5		$q_5 1 L$.		$q_6 ; L$.	
← Skip till *	q_6.	$q_6 1 L$	$q_0 * R$	—	—	
closing →	q_7	$q_7 1 R$	$q_7 * R$	$q_8 ; L$	$q_7 ; R$	
← left move till end of 1st	q_8	$q_8 1 L$			$q_9 ; L$	
← till end of 2nd	q_9.	$q_9 1 L$			$q_{10} ; L$	
	q_{10}	$q_{10} 1 L$			Halt	

Module 4

Recursive and Recursive Enumerable Languages

- When a TM executes its i/p, 4 possible outcomes
  1) Halts & accept the i/p
  2) Halts & reject the i/p
  3) Never halts (i.e falls into loop)
  4) Crash

* A TM accept the language ie Recursively Enumerable ie it accepts some i/p and some could enter into a loop. That is its undecidable
* A TM also accepts recursive language (which every string of the language accepts and string which are not in language should be rejected. It's decidable



* Every recursive L is also R.E

- A pbm is undecidable if there exist no algorithm that takes i/p an instance of pbm and determine whether that instance is yes or no

Properties of Recursive & R.E Language

- A Language L is said to be R.E if there exist a TM which accepts all words in L and rejects for ∀ word in $\bar{L}$ (compliment of L)
- This reduction involves combining several TM to form a composite m/c. The state of composite TM has a component for each individual component m/c
- Given an alg, we can follow composite TM to perform one action if alg accepts and another if it doesn't accept

Theorem

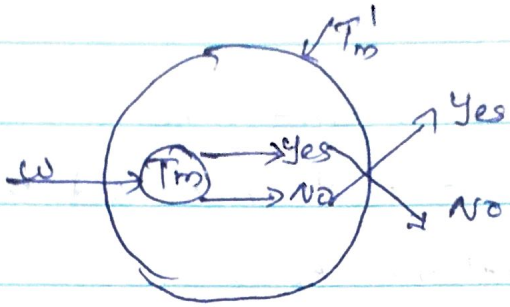
Compliment of recursive language is recursive

Proof

Let L be recursive language and T_m be a turing m/c that halts on all i/p and accepts L. Let construct a turing m/c T_m' for T_m so that if T_m enters a final state on i/p w, then T_m' halts without accepting

If T_m halts without accepting, T_m' enters a final state

Since one of these 2 events occurs, T_m' is an algorithm $T(T_m')$ is component of L and thus complement of L is recursive language.

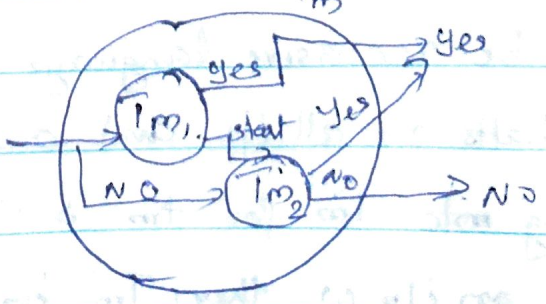


Theorem2

Union of 2 recursive language is recursive

Proof

Let L_1 and L_2 be 2 recursive language accepted by T_{m_1} & T_{m_2}. We construct a turing mc T_m, which 1st simulate T_{m_1}. If T_{m_1} accepts, then T_{m_2} accepts. If T_{m_1} rejects, then T_m simulates T_{m_2} and accepts iff T_{m_2} accepts. Since both T_{m_1} and T_{m_2} are algorithms, so T_m is guaranteed to halt T_m



Theorem 3

Union of 2 R.E language is recursively enumerable

Proof