__/__/____

$G = (N, \Sigma, P, S)$

$P : N \times (N \cup \Sigma)^*$

$\alpha \xrightarrow[G]{1} B$

How to define

$\alpha = \alpha_1 A \alpha_2 \quad (\alpha_1, \alpha_2) \in (N \cup \Sigma)^*$

and $A \to \gamma \in P$ and $\beta = \alpha_1 \gamma \alpha_2$

$\alpha \xrightarrow[G]{1} \beta \quad \alpha \xrightarrow[G]{*} \beta$ [Reflexive closure]

$\alpha \xrightarrow[G]{0} \alpha$

$\alpha \xrightarrow[G]{n+1} \beta$ s.t $\exists\, \gamma$, $\alpha \xrightarrow[G]{n} \gamma \xrightarrow[G]{1} \beta$

$\alpha \xrightarrow[G]{*} \beta$ if $\alpha \xrightarrow[G]{n} \beta$ for some $n \geq 0$

$S \xrightarrow[G]{*} \alpha \quad \alpha \in (N \cup \Sigma)^*$ (Sentential form)

If $\alpha \in \Sigma^* \longrightarrow$ Sentence.

A subset $A \subseteq \Sigma^*$ is said to be a CFL if $\exists$ a CFG G s.t $L(G) = A$.

Given a grammar G & string $x \in L(G)$

Derivⁿ of x as per G, G_2

→ There is a choice on which var you want to expand left most derivation, Rightmost derivⁿ.

$S \rightarrow ABC$

$A \rightarrow aA \mid \epsilon$

$B \rightarrow bB \mid \epsilon$

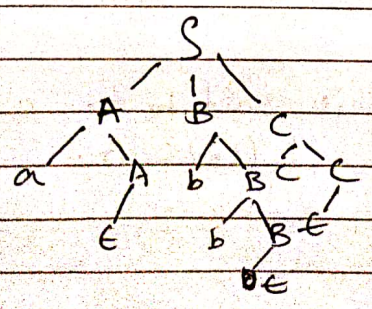
$C \rightarrow cC \mid \epsilon$

Left Most derivⁿ

$S \rightarrow ABC \rightarrow aABC \rightarrow aBC \rightarrow abBC \rightarrow abbC \rightarrow abbcC \rightarrow abbc$

Derivⁿ Tree / Parse tree

For a string, z is a tree representing the productions applied in derivation of z from the start Symbol S, independent of applicat.



Parse trees need not be unique for a particular string.

__/__/____

G ⟶ Ambiguous if $\exists w \in L(G)$ for which
- There are 2 or more distinct parse trees for w, or
- 2 or more left most derivations.

A language is ambiguous if it's not possible to have an unambiguous grammar for the language.

$L = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$ ⟶ Inherently Ambiguous

Can I put a bound on the structure of parse tree based on the string length?

We study restricted forms that are equivalent — but help in many applications

— Chomsky Normal Form (CNF)

A grammar CFG is CNF if all prodn's are of 2 forms:

$A \longrightarrow BC$

$A \longrightarrow a$

You can't generate ϵ using CNF.

Given a CFG G, you can convert this to a CNF G'

$L(G') - \{\epsilon\} = L(G)$

$S \to AS \mid \epsilon$

$A \to A1 \mid 0A1 \mid 01$

$L = \{(0^m 1^n)^* \mid m \le n\}$

$\Rightarrow$ $S \to AS \mid \epsilon$

$A \to 0A1 \mid B$ → Unambiguous

$B \to B1 \mid 01$

Theorem: Given a CFG G, you can convert this to a CNF G' s.t

$$L(G') = L(G) - \{\epsilon\}$$

We progressively get rid of other rules.

Step 1: Get rid of ϵ-prod^n rules.

You can't have these rules. ($A \to \epsilon$) ($S \to \epsilon$)

Instead put more production rules.

$B \to \alpha A \beta$
$A \to \epsilon$
$A \to a$
} New rule $B \to \alpha\beta$

Step 2: Get rid of unit produc^n rules

$A \to B$.

$A \to B$
$B \to \gamma$
} $\Rightarrow A \to \gamma$

After 1 & 2

All rules are of the form

i) $A \to a$

ii) $A \to (NU\Sigma)(NU\Sigma)^+$

Step 3 Take care of terminals in (ii)

For any terminal a, coin a new non-terminal A_a & replace a by A_a & add new rule

$A_a \to a$.

Step 4

All rules are of the form

i) $A \to a$

ii) $A \to B_1 \cdots B_K \quad K \geq 2$.

If $K=2 \to$ already CNF.

Rules : $A \to B_1 \cdots B_k \quad k \geq 3 \to$ Not in CNF

$A \to B_1 C$

$C \to B_2 \cdots B_k$ } Keep doing until all RHS are of length 2.

$\{a^n b^n \mid n \geq 0\}$

$S \to aSb \mid \epsilon$

↓ CNF

$L(G') = L(G) - \{\epsilon\}$

$S \to \epsilon$

$S \to aSb$ } $\Rightarrow$ New rule $S \to ab$

$S \to aSb \mid ab$ $\quad S \to ab \Rightarrow S \to AB$

$S \to aSb \Rightarrow S \to ASB$

$A \to a$

$B \to b$

$S \to ASB \Rightarrow S \to AC$

$C \to SB$

→ Final grammar

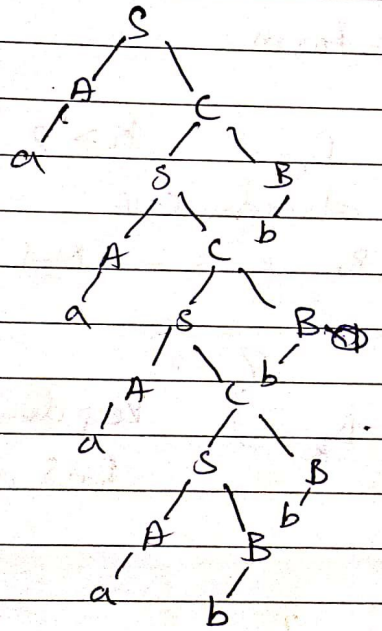
$S \rightarrow AB \mid AC$

$C \rightarrow SB$

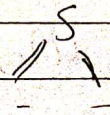
$A \rightarrow a$

$B \rightarrow b$

a^4b^4



Can I put a bound on the height of the parse tree



2 symbols level 1

level n → 2^n → n+1 level

A string of length 2^n what's the minimum height of the tree = n+1

→ To prove that a language is not CFG.

Pumping lemma for CFG

For every CFL A, $\exists\ k \geq 0$ such that every $z \in A$ of length at least k can be broken into 5 substrings:

$z = uvwxy$ such that

$|vwx| \leq k$ $\quad vx \neq \epsilon$ and for $\forall\ i \geq 0$

$uv^i wx^i y \in A$

Proof: Let G be a grammar for A in CNF.

Let n be # of non terminals of G.

Take $k = 2^n$

Take $z \in A$, $|z| \geq k$

→ Take the parse tree for z

Min ht $= n+1$

Take longest path in tree

length at least $n+1$

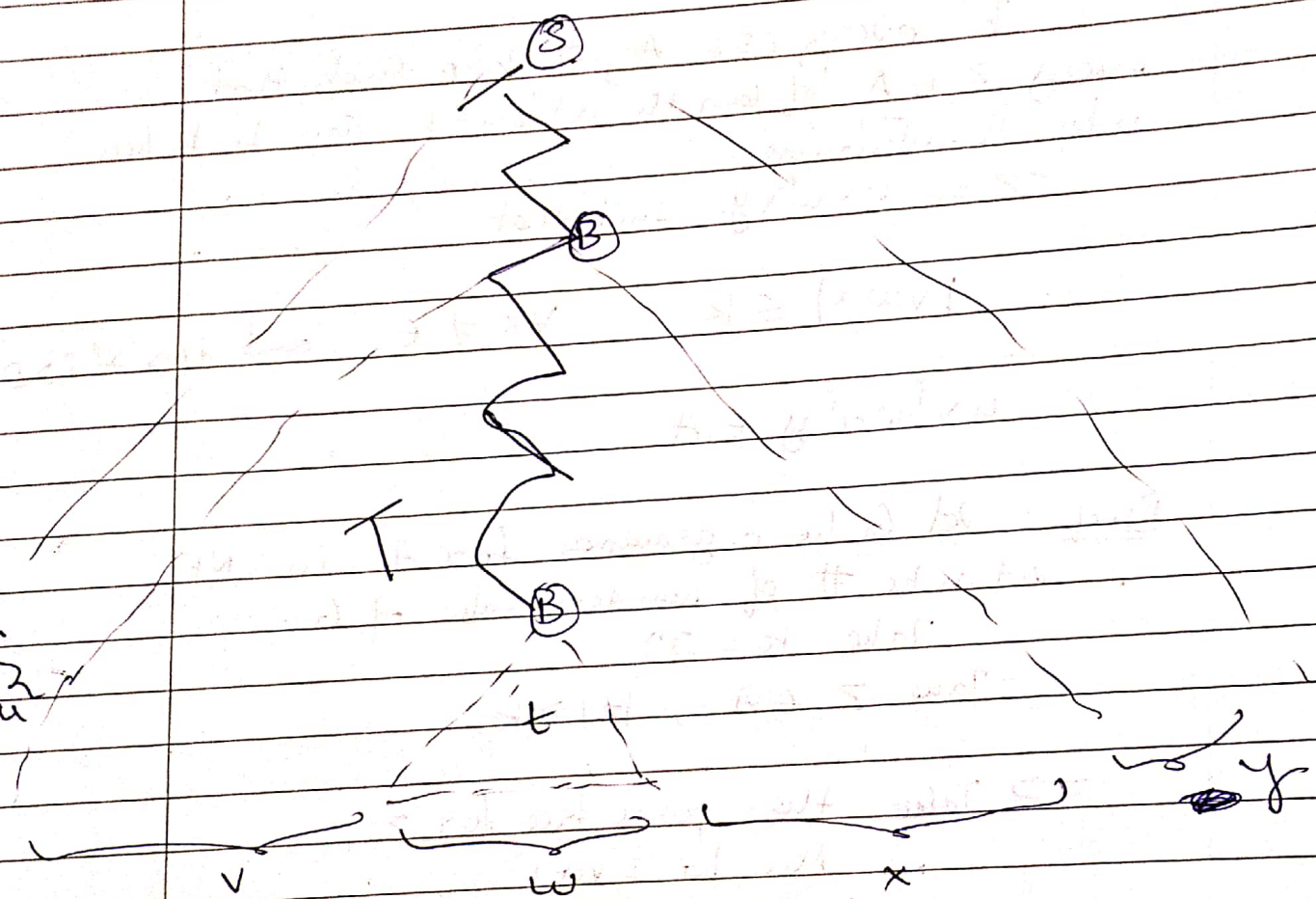
length $\geq n+1$

⇒ # of non terminals $\geq n+1$ on the path.

Pigeon hole → Atleast 2 non terminals are the same.

~~length $\geq n+1$~~

Take the last 2 occurrences from bottom



Replace t with T
Then we get $uv^i wx^i y$ $i^0 \geq 1$

replacing T with t
we get $uv^0 wx^0 y$.

Pumping Lemma for CFL

Any CFL A, $\exists a\ k \geq 0$ s.t for all strings $z \in A$, s.t $|z| \geq k$ its possible to break z, $z = uvwxy$, $|vwx| \leq k$, $vx \neq \epsilon$ s.t $\forall i \geq 0$ $uv^i wx^i y \in A$.

__/__/____

Proof

Let G be a grammar for A in CNF. Let n be the non-terminals

Take $k = 2^n$

Take a string $z \in A$

$|z| \geq k$

$\rightarrow |z| \geq 2^n$ ——— Ht of parse tree $\geq n+1$.

Take the longest path from S to some terminal

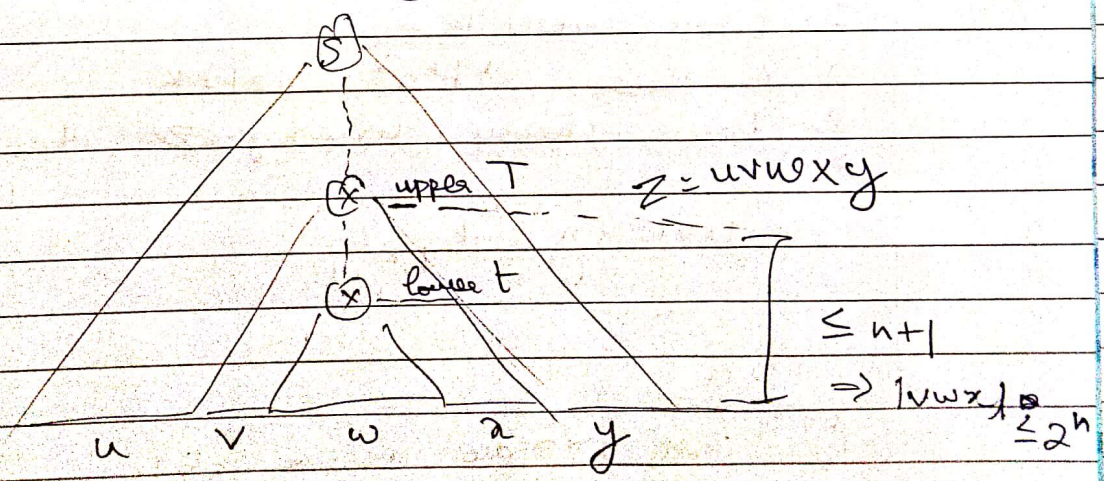
length $\geq n+1$.

at Least $n+1$ non terminals

By pigeonhole principle atleast 1 Non Terminal repeats, as only n non terminals exist in the Grammar.

~~Height of parse tree~~

Lets take 2 consecutive occurrences of X from the bottom of the two paths.



Contrapositive form

To prove that A is not Context Free

$\forall k \geq 0$,

$\exists$ a string $z \in A$ $|z| \geq k$ s.t

$\forall$ u v w x y

$z = uvwxy$

$|vx| > 0$, $|vwx| \leq k$

$\exists$ an i s.t $uv^i w x^i y \notin A$.

Game with demon

1. Demon chooses $k \geq 0$
2. You choose $z \in A$, $|z| \geq k$
3. Demon chooses $z = uvwxy$

$|vwx| \leq k$, $vx \neq \epsilon$

4. You choose $i \geq 0$ s.t $uv^i wx^i y \notin A$

$A = \{ a^n b^n a^n \mid n \geq 0 \}$

1. Demon chooses $k \geq 0$
2. You choose $a^k b^k a^k$ $|z| \geq k$
3. Demon chooses $uvwxy$ where $|vx| > 0$, $|vwx| \leq k$

$uvwxy = a^k b^k a^k$

Cases

i) Both v x are in one a^k block for $i \geq 2$ → that block has more a's than b's in neighbour block

ii) Both v & x are in b^k block.
$i=2$ No. of b's > No. of a's
(in 1 block) (in 1 block)

iii) v lies in a^k, x lies in b^k
$a^p b^q a^k$ one of p & $q > k \notin A$.

iv) v lies in b^k & x lies in a^k
$a^k b^q a^r$ one of $q, r > k$.

v) One of v or x contains both a's & b's.
⟶ Not of the form $a^*b^* \; a^* \in A$.

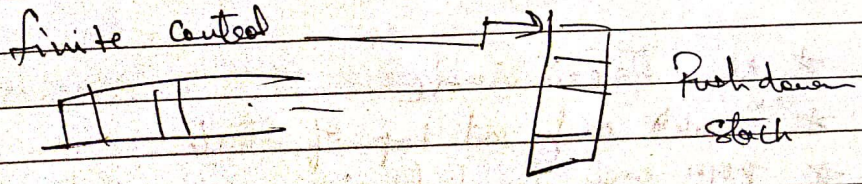
$A = \{ww \mid w \in \{0,1\}^*\}$

PDA

CFL's
- CFG
- PDA

Push down Automata

Non deterministic
NPDA



i/p tape

Scans current state P,
i/p symbol a, takes a trans^n
$\delta(P, a)$

DFA.

How do you start

1. Finite Control in start state s.

2. Scanning left most symbol on the i/p tape.

3. stack contains a designated start symbol 'I'

Transition

look at current state 'p', current i/p Symbol 'a', top of the stack 'A'.

Transition

→ Based on (p, a, A) move to state q, next symbol in i/p tape, pop A & push $B_1 \cdots B_k$ on the stack $k \geq 0$.

① look at current state 'p', current i/p Symbol 'a'

② Don't even look at i/p symbol 'p' ∈ 'A' → 'q' pop A, push $B_1 B_2 \cdots B_k$ ∈ transitions.

★ Non Deterministic
★ Can look only at top of stack.
★ pops one symbol from stack but is allowed to push more than one symbol.

__/__/____

Formal defⁿ of

$(Q, \Sigma, \delta, S, F, \Gamma, \perp)$

Γ : Stack Alphabet. $\perp$: Initial stack Symbol.

$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \longrightarrow Q \times \Gamma^*$

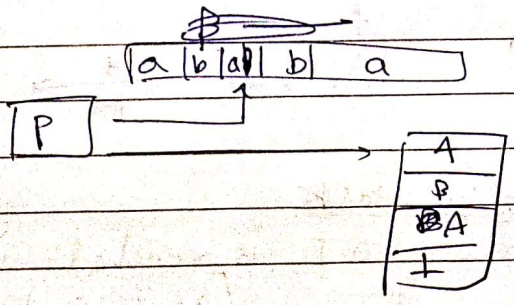
Types of transition :

1. $(P, a, A), (a, B_1 \cdots B_k) \in \delta$
2. $(P, \epsilon, A), (a, B_1 \cdots B_k) \in \delta$

Configuration at a time

(a, α, β)

- a → Current state
- α → remaining part of i/p
- β → stack content



Configuraⁿ : $(P, aba, ABA\perp)$

$: Q \times \Sigma^* \times \Gamma^*$

Feed x to PDA.

initial configuration.

$(S, x, \perp)$

$(S, x, \perp) \xrightarrow[M]{1} (\quad - \quad)$

$\xrightarrow[M]{*} (\quad)$

String x is accepted if it reaches final configuration

2 ways of accepting

1. By final state:

When the EOI is reached, m/c is in one of the final states.

Final Conf^n (q, ϵ, γ) $q \in F, \gamma \in \Gamma^*$

accept x if

$(S, x, \perp) \xrightarrow[M]{*} (q, \epsilon, \gamma)$

2. By Empty stack

When EOI is reached if stack is empty I accept the string. (q, ϵ, ϵ) $q \in Q$.

$\Rightarrow$ Q Accept x if

$(S, x, \perp) \xrightarrow[M]{*} (q, \epsilon, \epsilon)$

Example set of balanced parenthesis

$Q = \{q\}$ $\quad s = q$ $\quad F = \phi$

$\Sigma = \{[,]\}$ $\quad \Gamma = \{\perp, [\}$

$$\left.\begin{array}{l} (q, [, \perp), (q, [\perp) \\ (q, [, [), (q, [[) \\ (q,], [), (q, \epsilon) \\ (q, \epsilon, \perp), (q, \epsilon) \end{array}\right\} \delta$$

$(q,], \perp)$ ⟶ doesn't exist in δ, so this will not lead anywhere, thus can't reach (q, ϵ). (It's an NFA)

$L = \{ww^R \mid w \in \{0,1\}^*\}$

For w keep pushing symbols on the stack q_0.

At some point you make a guess that you are starting w^R

q_1 — I match the symbol with the top of the stack

If same, pop symbol, when you reach $\perp$ transit to q_2 (final state)

__/__/____

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

$S = q_0$

$F = \{q_2\}$

δ

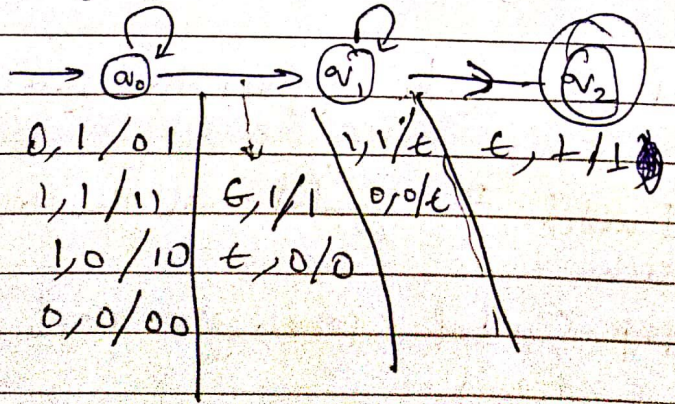
Forgot ⊥ transitions

$(q_0, 0, 1), (q_0, 01)$
$(q_0, 1, 1), (q_0, 11)$
$(q_0, 0, 0), (q_0, 00)$
~~$(q_0, 01, 0), (q_0, 01)$~~
$(q_0, 1, 0), (q_0, 10)$
$(q_0, \epsilon, \perp), (q_1, \perp)$
$(q_0, \epsilon, 0), (q_1, 0)$
$(q_0, \epsilon, 1), (q_1, 1)$

$(q_1, 1, 1), (q_1, \epsilon)$
$(q_1, 0, 0), (q_1, \epsilon)$
$(q_1, \epsilon, \perp), (q_2, \perp)$

$(q_0, \epsilon, \perp), (q_1, \epsilon\perp)$
$(q_0, \epsilon, 1) (q_1, 1)$
$(q_0, \epsilon, 0) (q_1, 0)$

Graphical notation.



__/__/__

$L = \{ w \in \{0,1\}^* \mid \#0(w) \geq \#1(w) \}$

PDA by empty stack

$\#0(w) = \#1(w)$ ⟶ Do for this, then we modify

Keep pushing symbols. If stack contains initial stack / top of stack is same as i/p

If top is diff, pop

$Q = q$

$S = q$

$F = \emptyset$

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

$\perp, \Gamma = \{\perp, 0, 1\}$

$(q, 1, \perp), (q, 1\perp)$

$(q, 1, 0), (q, \epsilon)$

$(q, 1, 1), (q, 11)$

$(q, 0, \perp), (q, 0\perp)$

$(q, 0, 0), (q, 00)$

$(q, 0, 1), (q, \epsilon)$

$(q, \epsilon, 0), (q, \epsilon)$

PDA by final state

↑↓

PDA by empty stack CFG

↑↓

CFG

__/__/____

CFG ⟶ PDA (empty stack)

$G = (N, \Sigma, P, S)$

Construct M st $L(M) = L(G)$
(by empty stack)

All rules are of the form

$$A \longrightarrow cB_1 \cdots B_k$$

$c \in \Sigma \cup \{\epsilon\}$

~~$A B_1 \cdots B_k \longrightarrow$ Non terminals~~

If

$A \longrightarrow AbB$

$\Rightarrow$ Make it

$A \longrightarrow AB_bB$

$B_b \longrightarrow b$

$A \longrightarrow cB_1 \cdots B_k$

$\Downarrow$

$(q, c, A), (q, B_1 \cdots B_k)$

$Q = \{q\}$

$\Sigma = \Sigma$

$S = q$

$F = \emptyset$

$\perp$ ~~$= S$~~ (Fresh)

$\Gamma = N$ (Fresh)

CFG → NPDA

G → All Production of form

$A \longrightarrow cB_1 \cdots B_k \qquad c \in \Sigma \cup \{\varepsilon\}$

↓

PDA transⁿ

PDA by empty stack

Single state q_1.

$(q_1, c, A), (q_1, B_1 B_2 \cdots B_k)$

Ex:1 Balanced parenthesis

$S \longrightarrow [S] \mid SS \mid \varepsilon$

Change to

$S \longrightarrow [SB$

$S \longrightarrow SS$

$S \longrightarrow \varepsilon$

$B \longrightarrow]$

$\langle \{q_1\}, \{[,]\}, \{S, B\}, S, q_1, S, \phi \rangle$

i) $(q_1, [, S), (q_1, SB)$

ii) $(q_1, \varepsilon, S), (q_1, SS)$

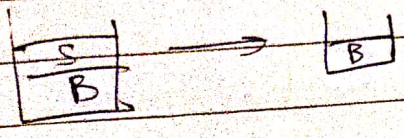
iii) $(q_1, \varepsilon, S), (q_1, \varepsilon)$

iv) $(q_1,], B), (q_1, \varepsilon)$

Let i/p string be [ ] [[ ]]

a) For '[' Pop S & push BS ... (acc to i)

b) For ']' according to (iv)



__/__/____

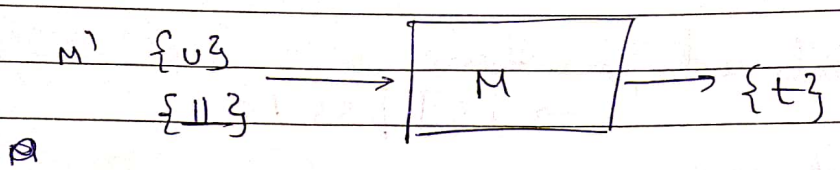
CFL's
- CFG
- (N)PDA
  - Final state
  - Empty state

⟹ Only one final state
Accepts by final state/empty stack "coincide"

↓

PDA that accepts by empty stack & has only 1 state

Use the same conversions as CFG → NPDA.



$M = \langle Q, \Sigma, \Gamma, \delta, s, \perp, F \rangle$

$M' = \langle Q \cup \{U, t\}, \Sigma, \Gamma \cup \{\perp\}, \delta', U, \perp, \{t\} \rangle$

$\delta' = \delta \cup$ ~~

↳ $(U, \epsilon, \perp), (s, \perp\perp)$

$(a, \epsilon, A), (t, A)$ $\quad a \in \Delta, A \in G.$

$(t, \epsilon, A), (b, \epsilon)$

what are $\Delta \times G$ ⟶ depends on whether we want (N)PDA that accepts by empty stack or Final state.

$$\Delta = \begin{cases} F & \text{Accepts by final state} \\ Q & \text{Accepts by empty stack} \end{cases}$$

$$G = \begin{cases} \Gamma' & \text{Final state} \\ \{\perp\} & \text{Empty stack.} \end{cases}$$

To show:

$L(M) = L(M')$

(1) If $x \in L(M) \Rightarrow x \in L(M')$

$\Rightarrow L(M) \subseteq L(M')$

(2) If $x \in L(M') \Rightarrow x \in L(M)$

$\Rightarrow L(M') \subseteq L(M)$

Case a : M accepts by final state.

$(S, x, \perp) \xrightarrow[M]{n} (q, \epsilon, x)$ $q \in F$, $x \in \Gamma^*$, For some $n \geq 0$

$(u, x, \perp\perp) \xrightarrow[M']{1} (S, x, \perp\perp)$

$(t, \epsilon, x\perp\perp) \xleftarrow[M']{1} (q, \epsilon, x\perp\perp)$ $\xleftarrow{\frac{n}{M'}}$

$\xrightarrow[M']{*} (t, \epsilon, \epsilon)$

Case b : Empty Stack

$(S, x, \perp) \xrightarrow[M]{n} (q, \epsilon, \epsilon)$ $q \in Q$.

M'

$(u, x, \perp\perp) \xrightarrow[M']{1} (S, x, \perp\perp) \xrightarrow[M']{n} (q, \epsilon, \perp\perp)$

$(t, \epsilon, t) \xleftarrow[M']{1} (t, \epsilon, \perp\perp)$ $\xleftarrow{\frac{1}{M'}}$

This shows (1)

__/__/____

Now Proving (2)

$(u, x, \Pi) \xrightarrow[M']{1} (s, x, \perp\Pi) \xrightarrow[M']{n} (q, y, r\Pi)$

$(t, \epsilon, \epsilon) \xleftarrow[M']{*} (t, y, r\Pi) \xleftarrow[M']{1}$

~~(y has to be ε)~~

From t only ε transitions are possible

$\Rightarrow y = \epsilon$ otherwise machine gets stuck.

$(s, x, \perp) \xrightarrow[M]{n} (q, \epsilon, r)$ → We need to show it's true for both Empty stack and Final state.

Case a: Empty stack.

r has to be ε. As stack should be empty.

$\Rightarrow (s, x, \perp) \xrightarrow[M]{n} (q, \epsilon, \epsilon)$

This final configuration can reach an end in M'.

Case b: Final state.

$(q, \epsilon F)$

$(s, x, \perp) \longrightarrow (q, \epsilon, r) \quad (q \epsilon F)$

∴ Since (1) & (2) are proved.

∴ $L(M) = L(M')$

$\langle Q, \Sigma, \Gamma, \delta, s, \perp, \{t\} \rangle$

Single final state t

Acceptance by either final state or Empty stack.

Empties stack once it reaches t.

PDA with a single state

Acceptance by empty stack.

Simulate all states on stack

$\Gamma' = Q \times \Gamma \times Q$

Elements are $\langle p A q \rangle$

$p, q \in Q$.

$A \in \Gamma$

Interpretation of $\langle p\ A\ q \rangle$

Currently in state p seeing A.

By the time m/c would've popped A it would be in state q.

↳ (Removing A & not adding anything else)

The m/c will be able to accept x if it was in state p & seeing A on stack. After scanning x it reaches q & pops A (empty stack)

$M' = (\{*\}, \Sigma, \Gamma', \delta', *, \langle s \perp t \rangle, \emptyset)$

$(p, c, A), (q, B_1 \cdots B_k)$

$((*, c, \langle p\ A\ q_k \rangle), (*, \langle q\ B_1\ q_1 \rangle, \langle q_1, B_2, q_2 \rangle, \ldots, \langle q_{k-1}, B_k, q_k \rangle))$

__/__/____

$(P, \epsilon, A), (\alpha, \epsilon)$

↓

$((*, \epsilon, \langle P A \alpha \rangle), (*, \epsilon))$

Propositional logic.

Constants T, ⊥ → false.

Binary operators $\rightarrow, \wedge, \vee, \leftrightarrow$

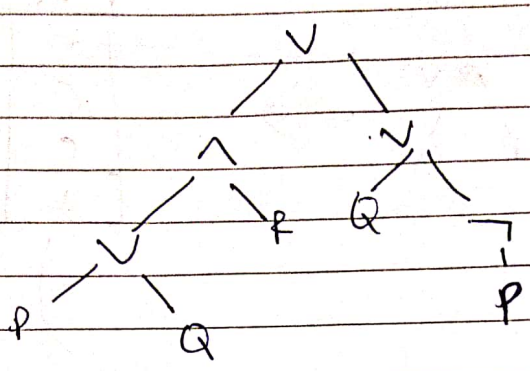
Unary " $\neg$

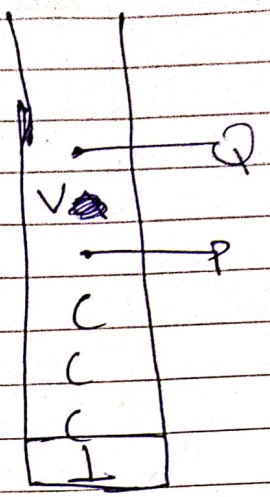
Variables P, Q, R

Parenthesis (,)

CFG $\leftrightarrow$ PDA

$((P \vee Q) \wedge R) \vee (Q \wedge (\neg P))$





Stack for Parsing ⟶

Parsing

1. Start with the initial symbol on stack
   Scan from left to right.
   → a. If symbol is '(', push
   b. If symbol is a variable or constant, push.
   c. If symbol is an operator, push
   d. If the symbol is ')', reduce.

Reduce

(i) Pop the top of stack. This should be a variable/Const
Make this as the right operand for the node. } Node

(ii) Pop the next element. This should be an operator for the node. If binary continue

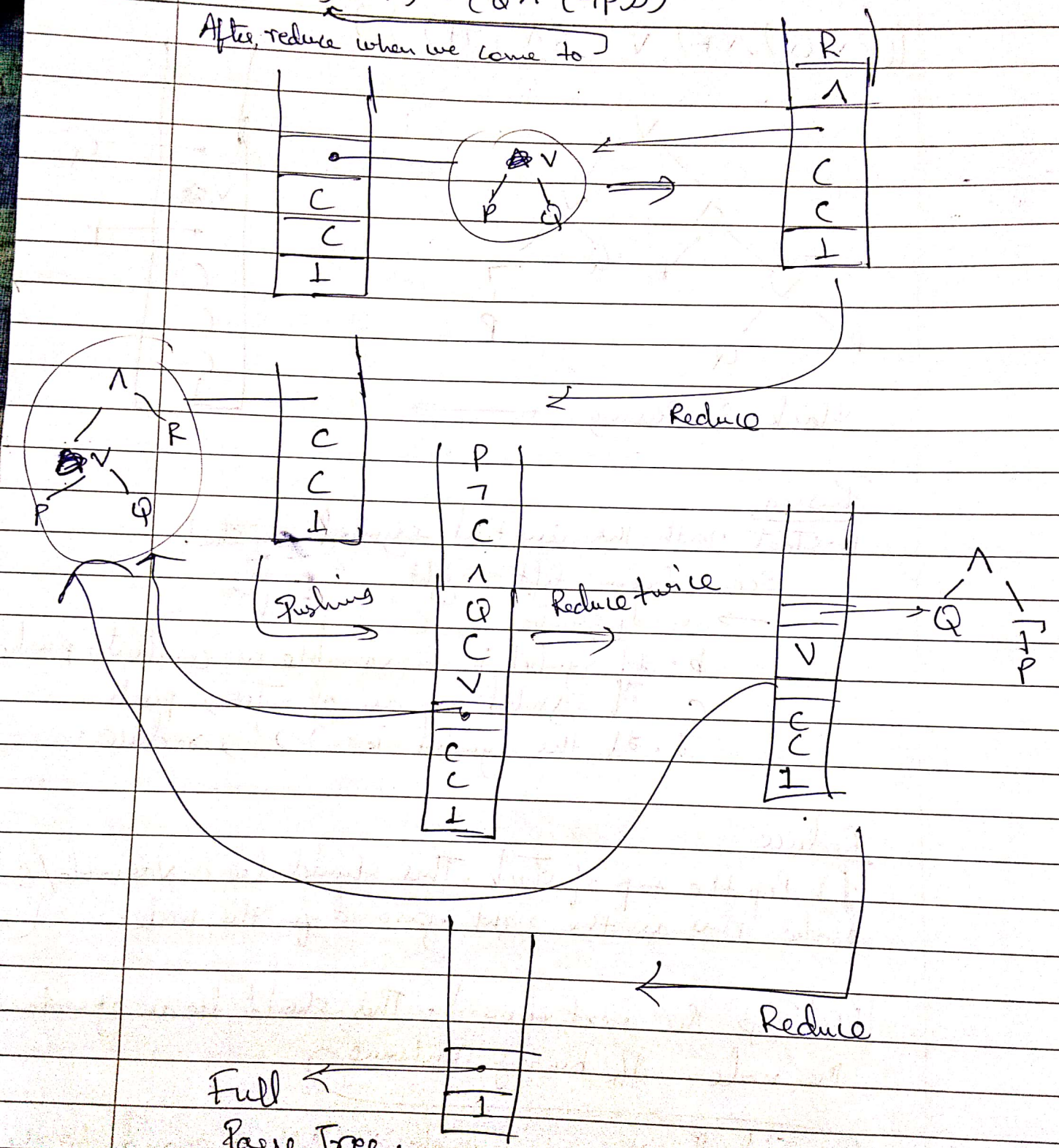
↳ iii) Pop the top, make it left operand for the Node.

iv) Pop the top, it should be '('

v) Push pointer on the top.

$((((P \vee Q) \wedge R) \vee (Q \wedge (\neg P)))$

After, reduce when we come to



a/b.c

a.b/c a/bc (a/b).c

Arithmetic Expression

$E \rightarrow E+E \mid E-E \mid E/E \mid E.E \mid -E \mid C \mid V \mid (E)$

Ambiguous.

$C \rightarrow 0 \mid 1$

$V \rightarrow a \mid b \mid c$

$\rightarrow$ unary .,/ > +,− (Partial Order) — Precedence.

Parsing :−

Scan from left to right

a. Variable / Constant → Push

b. See operator (B) → Find previous operator (A)

If B has higher precedence than A → Push B

Lower " than A → Reduce

Same " as A → Reduce.

Once expression has ~~returned~~ ended → Reduce.

Unambiguous Grammer for Arithmetic Expression

$E \rightarrow E+F \mid E-F \mid F$

$F \rightarrow F.G \mid F/G \mid G$

$G \rightarrow -G \mid H$

$H \rightarrow C \mid V \mid (E)$

$C \rightarrow 0 \mid 1$

$V \rightarrow a \mid b \mid c$

__/__/____

Closure properties of CFL (Closure satisfied when ticked)

i) Union ✓

$S \rightarrow S_1 | S_2$

$L_1 \cup L_2$

ii) Concatenation ✓

$S \rightarrow S_1 S_2$

iii) Kleene Closure ✓

iv) Reversal. ✓ → Reverse RHS of all productions

v) Homomorphism ✓ on the terminal

$h(0) = ab$

$h(1) = \epsilon$

Use the Grammar substitute for the terminals

vi) Inverse Homomorphism ✓

$h(0) = ab$

$h(1) = \epsilon$

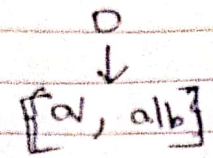
Simulate ab on other PDA.

Let P' be the new PDA $\{a, b\}$

States of P' = $[a, c]$ a is state of P

Finite $\{ a\ b\ b\ \epsilon \}$ c is any suffix under homomorphism.

P' start state $[q, \epsilon]$



transitions.

$([q,\epsilon], x, X) \;/\; ([q, h(x)], X)$

$\delta(q, d, X) = (q', \gamma)$ ⟶ Transit^n present in original PDA.

$([q, d\alpha], \epsilon, X), ([q'\alpha], \gamma)$

$\underset{\alpha \in \Sigma^*}{d \in \Sigma}$ $\delta(q, \epsilon, X) = (q', \gamma)$ ⟶ Transit^n in original PDA.

$([q, \epsilon], x, X), ([q, h(x)], X)$

$x \neq \epsilon$

$([q, \epsilon], 0, 1) \rightarrow ([q, ab], \perp)$

$([q, t], c, X) \longrightarrow [q, h(c)], X$ → Filling your buffer.

$(q, d, X) \longrightarrow (q', \gamma)$

vii) Intersection ✗

Counter Example :

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

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

$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ → NOT CFL

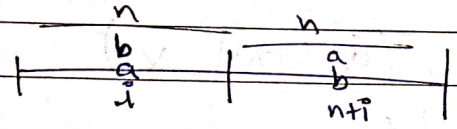
viii) Complement ✗

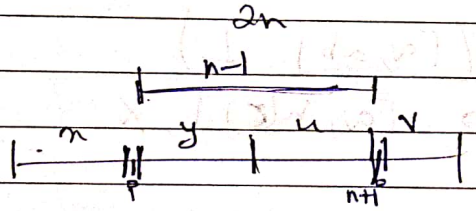
$L_1 \cap L_2 = \overline{(\bar{L}_1 \cup \bar{L}_2)}$

If CFL's were closed under complement then $L_1 \cap L_2$ would also be closed under intersection, which we know is False.

$L = \{ww \mid w \in \{a,b\}^*\}$

I → Strings of odd length
strings of even length that are asymmetric





If $i = a \Rightarrow n+i = b$
$i = b \Rightarrow n+i = a$.

$|x| = |y| \quad |u| = |v|$

(or) $x\ a\ y\ u\ b\ v$
$x\ b\ y\ u\ a\ v$

$S \rightarrow A \mid B \mid AB \mid BA$
$A \rightarrow CAC \mid a$
$B \rightarrow CBC \mid b$
$C \rightarrow a \mid b$

ix) Difference ✗

x) Intersection with RL's. ✓

L_1 : CFL → PDA
L_2 : RL → DFA
} → Simulate Simultaneously

$\delta([q,p],a,X) = ([\delta(q,a),p'],\gamma)$

$a \in \Sigma \cup \{\epsilon\}$ s.t $(p,a,X),(p',\gamma) \in \delta$

$L = \{x \in \{a,b,c\}^* \mid \#a(x) = \#b(x) = \#c(x)\}$

Show that L is not a CFL

Assume L is a CFL

$L \cap a^*b^*c^*$ should be a CFL

$= \{a^n b^n c^n \mid n \geq 0\}$ ⟶ not a CFL.

CFG G, string x.

Is $x \in L(G)$? → $|x|$

$|x| = n$

n Nonterminals

n Terminals for each

$S \to AB \to ACD \to \_\_\_$

$S \xrightarrow[G]{2n-1} x$

Try out all possible derivⁿs of length $2n-1$
$\leq K^{2n-1}$

$S \rightarrow aSb \mid \epsilon$

K = No. of prodⁿ

G: $S \rightarrow AB \mid BA \mid SS \mid AC \mid BD$

$A \rightarrow a$

$B \rightarrow b$

$C \rightarrow SB$

$D \rightarrow SA$

x → | a | a | b | b | a | b |
0 1 2 3 4 5 6

x_{ij} ⟶ substring b/w i x j

	0					
A	1					
∅	A	2				
∅	S	B	3			
S	C	∅	B	4		
D	S	∅	S	A	5	
	C	∅	C	S	B	6

→ what can derive. (non terminals)

Substrings of length 1
$x_{01}\ x_{12}\ x_{23} \cdots x_{51}$

$x_{02} \rightarrow x_{01} x_{12} \Rightarrow x_{02} \rightarrow AA$ → rule not present
thus 0, 2 in table is ∅.

$x_{03} \rightarrow x_{01} x_{13}$ AS
$x_{02} x_{23}$ ∅B
} Neither possible.

Keep filling this way.

Time Complexity — $O(Kn^3)$

DPDA	NPDA
1) Frozen <_____, →1> (8tupple) ↳ end of i/p marker	
2.) Transit: $(q, a, A), (q', \gamma)$ $Q \times (\Sigma \cup \{\lambda\} \cup \{\in\}) \times \Gamma \to Q \times \Gamma^*$	
3.) Bottom of stack always contain ⊥ $(q, a, \perp), (q', \beta\perp)$	

__/__/____

Unrestricted language.

4 tuple $G = \langle V \Sigma S, P \rangle$

P is of the form $\alpha \longrightarrow \beta$

$\alpha, \beta \in (V \cup \Sigma)^*$

Except that α contains atleast 1 variable.

$S \longrightarrow aSc | B$

Generating $\{a^n b^n c^n \mid n \geq 0\}$

$S \longrightarrow ABCS \mid T_c$

$CAB \longrightarrow AC$

$BA \longrightarrow AB$

$CB \longrightarrow BC$

$C T_C \longrightarrow T_C c$

$T_C \longrightarrow T_B$

$B T_B \longrightarrow T_B b$

$T_B \longrightarrow T_A$

$A T_A \longrightarrow T_A a$

$T_A \longrightarrow \epsilon$

$S \longrightarrow S'z$

$S' \longrightarrow aS'A$

$S' \longrightarrow bS'B$

$S' \longrightarrow \epsilon$

$AZ \longrightarrow Xz$

$AX \longrightarrow XA$

$BX \longrightarrow XB$

$aX \longrightarrow aa$

$bX \longrightarrow ba$

$BZ \longrightarrow YZ$

$AY \longrightarrow YA$

$BY \longrightarrow YB$

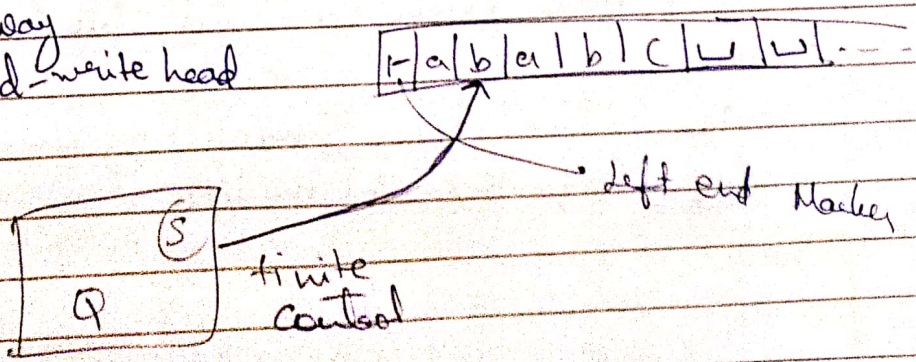
$bY \longrightarrow bb$

$aY \longrightarrow ab$

Turing m/c

1936 — Alan Turing → Defines what is Computable.

Informal Descrip^{n} :- Deterministic single Tape



__/__/__

Computation

Start:- Finite control is in start state 's'.
The tape contains the i/p immediately after the symbol ⊢.
The remaining cells are blank 'u'

The reading head points to the left end marker.

Transition

At any given point the m/c can see
- the current state 'p'
- the symbol at the i/p tape 'a'.

based on this
- change to a new state 'q'
- write a symbol 'b' on the tape
- move either left/right by one cell.

Termination

- If m/c reaches accept state 't'.
- If m/c reaches the reject state 'r' it rejects i/p.

- For certain i/p s, m/c may never reach 't' or 'r' it loops (infinite) for the i/p.

Formal defⁿ

$(Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, t, r, s)$

- Γ: Tape Alphabet, $\Sigma \subset \Gamma$
- $\vdash$: left end marker, $\in \Gamma - \Sigma$
- $\sqcup$: Special blank Symbol, $\in \Gamma - \Sigma$
- $t, r, s \in Q$

$\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$

Restrictions

1. left end marker $\vdash$ is kept as it is (and move R)

$\forall p, \delta(p, \vdash) = (q, \vdash, R)$ for some q.

2. Once you are in 't' or 'r', you stay there

$\forall b \quad \delta(t, b) = (t, b', d)$ for some b', d

$\delta(r, b) = (t, b'', d')$ " " b'', d'

$d \times d' \in \{L, R\}$

$b', b'' \in \Gamma$

$L = \{ a^n b^n c^n \mid n \geq 0 \}$

$\vdash aaabbbccc$

① check if my string is of the form $a^* b^* c^*$.

- keep scanning from left to rt check if of the form $a^* b^* c^*$

- Replace last '$\sqcup$' with '$\dashv$'.

(2) Check if #a = #b = #c

Scan from rt to left, erase one 'c', one 'b', one 'a'

Scan from left to rt, " one 'a', one 'b', one 'c'

Or you erase everything & make a pass b/w ⊢ ⊣

all blanks → accept

If in any pass some symbols are there, but not others, reject.

	⊢	a	b	c	⊔	⊣
S	$(S, \vdash, R)$	(S, a, R)	(q_1, b, R)	(q_2, c, R)	$(q_3, \dashv, L)$	—
q_1	—	$(r, \rightarrow)$	(q_1, b, R)	(q_2, c, R)	$(q_3, \dashv, L)$	—
q_2	—	$(r, \rightarrow)$	$(r, \rightarrow)$	(q_2, c, R)	$(q_3, \dashv, L)$	—
q_3	$(t, \rightarrow, \rightarrow)$	$(r, \rightarrow, \rightarrow)$	$(r, \rightarrow, \rightarrow)$	$(q_4, \sqcup, L)$	$(q_3, \sqcup, L)$	—
q_4	$(r, \rightarrow)$	$(r, \rightarrow)$	$(q_5, \sqcup, L)$	(q_4, c, L)	$(q_4, \sqcup, L)$	—
q_5	$(r, \rightarrow)$	$(q_6, \sqcup, L)$	(q_5, b, L)	(q_5, c, L)	$(q_5, \sqcup, L)$	—
q_6	$(q_7, \vdash, R)$	(q_6, a, L)	—	—	$(q_6, \sqcup, L)$	—
q_7	—	$(q_8, \sqcup, R)$	$r \rightarrow$	$r \rightarrow$	$(q_7, \sqcup, R)$	$t \rightarrow$
q_8	—	$(q_8, \sqcup, R)$	$(q_9, \sqcup, R)$ $r \rightarrow$	$(q_8, \sqcup, R)$	$r \rightarrow$	
q_9	—	—	(q_9, b, R)	$(q_{10}, \sqcup, R)$	$(q_9, \sqcup, R)$	$r \rightarrow$
q_{10}	—	—	—	(q_{10}, c, R)	$(q_{10}, \sqcup, R)$	$(q_3, \dashv, L)$