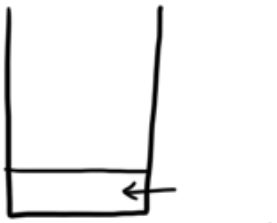

Context Free Languages.

→ Finite Automata has limits.

Here we use memory which is Infinite in Nature.

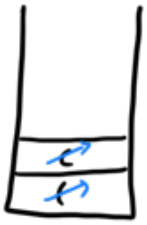
We use STACK here.

→ Operations on STACK — PUSH, POP



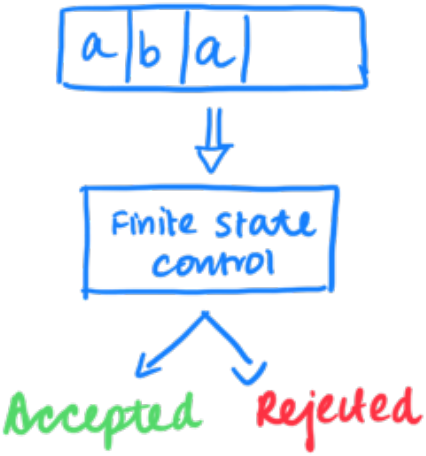
→ Matching Paranthesis

(( ))



Push the opening brackets

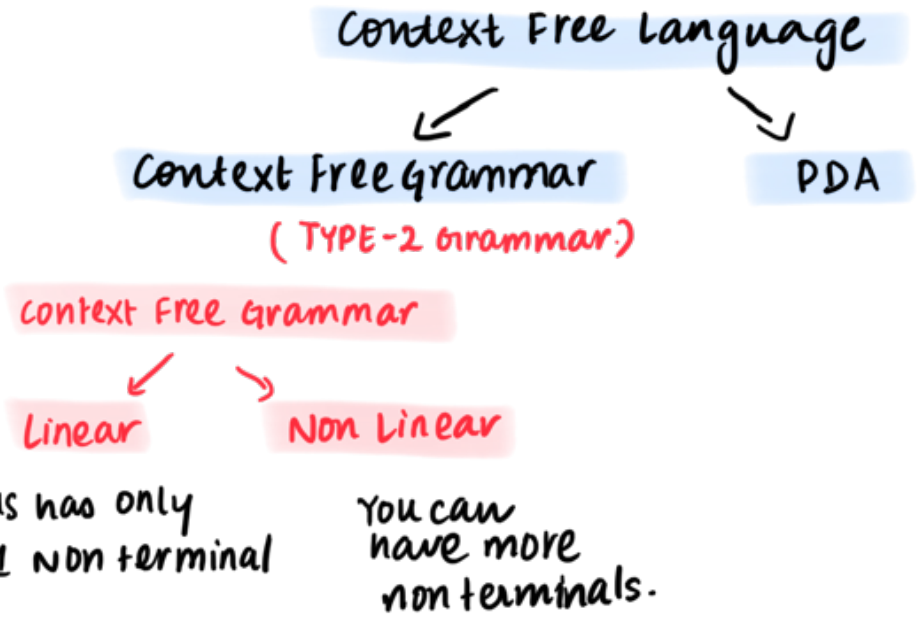
Pop on seeing closing brackets.



→ Here we describe CONTEXT FREE LANGUAGES by 7 tuples

↳ $Q, f, \Sigma, \delta, q_0, Z_0, \gamma$ → stack symbol

(bottom of stack)



* Context Free Languages are formal languages, they are the basis for programming languages. Responsible for Compiler Design.
* Syntax of a language - context free Grammar.

$G = (V, T, P, S)$ → start symbol

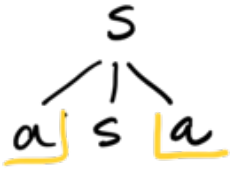
V → variables, T → Terminals, P → $V \to \alpha$, $\alpha \in (V \cup T)^*$

α is a grammar symbol

TERMINAL, NONTERMINAL

Why is it called context Free Grammar / Language ?

EX $S \to aSa \mid bSb \mid \lambda$



We dont bother about what's on the right/left.

LINEAR CONTEXT FREE GRAMMAR

Single non terminal on RHS

① Even palindromes $L=\{ w w^R, w \in \{a|b\}^* \}$

Minimal string $= \lambda$

$L = \{ \lambda, aa, bb, aba, bab, abba \cdots \}$

First symbol must be same as the last symbol.

$$S \rightarrow aSa \mid bSb \mid \lambda$$

a at start, a at end b at start, b at end

(2) $L=\{ wcw^R$, mid is detected by c $\}$

$$S \rightarrow aSa \mid bSb \mid c$$

$L = \{ c, aca, bcb, abcba \cdots \}$

(3) $L=\{ ww^R, w \in (ab)^* + (ba)^* \}$

$$S \rightarrow abSba \mid baSab \mid \lambda$$

starts with ab must end with ba

" " ba " " " ab

(4) $L = \{ a^n w w^R b^n \;,\; w \in \{a,b\} \}$

Already know this

Merging two languages.

Generate equal no. of a's and b's.
And instead of replacing it with λ, replace it with a even palindrome.

$S \to aSb \mid A$

Equal no. of a's and b's.

$A \to aAa \mid bAb \mid \lambda$

(5) $L = \{ a^n b^{n+1} \;,\; n \geq 0 \}$

Minimal = b.

$S \to aSb \mid b$

Equal a's and b's and one extra b.

(6) $L = \{ a^{n+2} b^n \;,\; n \geq 1 \}$

Minimal = aab

$S \to aSb \mid aa$

(7) $L = \{ a^n b^{2n}, n \geq 0 \}$

Minimal = λ

$S \rightarrow aSbb \mid \lambda$

(8) $L = \{ a^n b^{n-3}, n \geq 3 \}$

Minimal = aaa

$S \rightarrow aSb \mid aaa$

(9) $L = \{ a^n b^m, \quad n > m \}$

Minimal = a

$S \rightarrow aSb \mid aS \mid a$

(10) $L = \{ a^n b^m, n \neq m \}$ Minimal = a, b, aab, bba

$S \rightarrow A \mid B \rightarrow$ where # B's > A's

↓ where # A's > B's

$A \rightarrow aAb \mid aA \mid a$

$B \rightarrow aBb \mid bB \mid b$

(11) $L = \{ a^n b^m, n = 2 + (m \bmod 3) \}$

$m = 0$	$n = 2 + (0 \% 3) = 2$	aa
$m = 1$	$n = 2 + (1 \% 3) = 3$	$aaab$
$m = 2$	$n = 2 + (2 \% 3) = 4$	$aaaabb$

$m = 3$	$n = 2 + (3 \%_0 3) = 2$	aabbb
$m = 4$	$n = 2 + (4 \%_0 3) = 3$	aaabbbb
$m = 4$	$n = 2 + (5 \%_0 3) = 4$	aaaabbbb

No of a's are restricted to 2, 3, 4
and b is any number

$\%_0 3 \longrightarrow 0$
$\longrightarrow 1$
$\longrightarrow 2$
} Three possible remainders

$S \rightarrow aaA \mid aaabA \mid aaaabbA$

BASE CASES

$A \rightarrow bbbA \mid \lambda$

(Write the 3 Base CASES and then we can generate B's - multiples of 3. where this generation of B's as " " is optional)

(12) $L = \{a^n b^m,\ n \neq 2m\}$

{ aab X
aaaabb X }

$m = 0,\ n \neq 0$ → a^+

$m = 1,\ n \neq 2$ → aaa^+b

b | ab — {aaab aaaab aaaa

$m = 2,\ n \neq 4$

bb | abb aabb aaabb aaaa

In this case we dont want aab

The ones highlighted in GREEN are our base CASES:

we dont want to generate this

$S \rightarrow aaSb \mid A \mid B \mid aC$

$A \rightarrow aA \mid a$ For a+

$B \rightarrow Bb \mid b$ For b+

$C \rightarrow Cb \mid b$ For ab*

(13) $L = \{ a^{n+2} b^m \;,\; m > n,\; n \geq 0 \}$

n	m	STRING
0	>0	$a^2 b b^*$
1	>1	$a a^2 b b b^*$
2	>2	$a a a^2 b b b b^*$

Equal Number of a's and b's

for $a^2 b$

$S \rightarrow aSb \mid aab \mid Sb$ for just b's

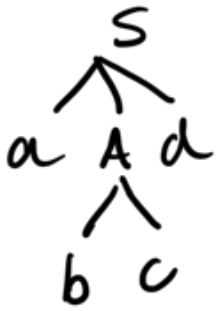
for the middle part

(14) $L = \{ a^n b^m c^m d^n , n, m \geq 1 \}$

Equal a's and d's
b's and c's

Minimal string = abcd

$S \rightarrow aSd \mid aAd$
$A \rightarrow bAc \mid bc$



(15) $L = \{ a^n b^m c^k , k = n+m , n, m, k \geq 0 \}$

$c = b + a$ { for every a one c
" " b one c }

$a^n b^m c^{n+m}$
$a^n b^m c^m c^n$

Minimal here is λ

$S \rightarrow aSc \mid \cancel{aAc} \mid A$ → Not required
$A \rightarrow bAc \mid \cancel{bc} \mid \lambda$

(16) $L = \{ a^n b^m c^k , m = 2n , k = 2 , n \geq 0 \}$

Minimal when $n = 0$ cc

$n, 2n, 2$

$a^n b^n$ (a^n: A, b^n: B)

$S \to AB$

$A \to aAbb \mid \lambda \quad [n \geq 0]$

$B \to cc$

(17) $L = \{a^n b^m c^k, \quad m, n \geq 0$

$k = n + 2m$

$a^n b^m c^n c^{2m}$

$a^n b^m c^{2m} c^n$ (Minimal $= \lambda$)

$A \quad B \quad A$

$S \to aSc \mid A$

$A \to bAcc \mid \lambda$

(18) $L = \{ |W| \bmod 3 \neq |W| \bmod 2 \}$ $W = \{a\}^*$

$|W| \bmod 3$: 0 1 2 — $|W| \bmod 2$: 0 1

	$\|W\| \bmod 3$	$\|W\| \bmod 2$		
a^0	$=0$	$=0$	0 0	X
a^1	$=1$	$=1$	0 1	X
a^2 belongs	$=2$	$=0$	1 0	X
a^3	$=0$	$=1$	1 1	X
a^4	$=1$	$=0$	0 2	✓
a^5 belongs	$=2$	$=1$	2 0	✓
			2 1	✓
a^6	$=0$	$=0$	1 2	✓
a^7	$=1$	$=1$	2 2	✓

a^2 belongs. $= 2$ $= 0$

$L = \{ a^2, a^3, a^4, a^5, a^8, a^9, a^{10}, a^{11} \}$

a^6 a^6 a^6 a^6

Base cases a^2, a^3, a^4, a^5

Generate a sum multiple of 6's

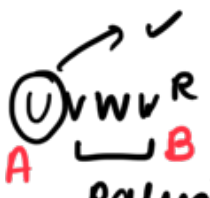
$S \rightarrow aaaaaaS \mid aa \mid aaa \mid aaaa \mid aaaaa$

Base cases.

END OF LECTURE 1

TOPIC 2 : CONTEXT FREE GRAMMAR (NON LINEAR)

Ex(1) $L = \{ UVWV^R, \; |U| = |W| = 2, \; |V| \geq 1, \; \Sigma = \{a, b\} \}$



Palindrome

$|U| = 2$

aa ab ba bb

$S \rightarrow AB$

$A \rightarrow aa \mid bb \mid ab \mid ba$

$B \rightarrow aBa \mid bBa \mid A$

$w = 2$

aa ab ba bb

B no a palindrome

A for w

Ex(2) $L = \{ n_a(w) = n_b(w), \; w \in \{a, b\}^* \}$

[Not same as $a^n b^n$
This is a subset of $a^n b^n$]

NO specific order

For "bb" and "aa" so that they can occur simultaneo

$S \rightarrow aSb \mid bSa \mid SS \mid \lambda$

Ex(3) $L = \{ n_a(w) = n_b(w) + 1 \}$

$S \rightarrow AaA$
$A \rightarrow aAb \mid bAa \mid AA \mid \lambda$

Ex(4) $L = \{ n_a(w) = 2\, n_b(w) \}$

Minimal = λ

(aab, aba, baa,)

$S \rightarrow aSaSb \mid bSaSa \mid aSbSa \mid \lambda \mid SS$

EX (4) $L = \{ n_a(w) > n_b(w) \}$

Minimal = a

$S \rightarrow$ (AaA) Minimal string

$A \to aAa \mid bAb \mid \underbrace{aA \mid Aa}_{\text{More a's.}} \mid \lambda \mid AA$

Ex (5) $L = \{ n_a(w) \neq n_b(w) \}$

↗ a's more ↘ b's more

$S \to AaA \mid BbB$

$A \to aAb \mid bAa \mid AA \mid \lambda \mid aA \mid Aa$

$B \to bBa \mid aBb \mid BB \mid \lambda \mid Bb \mid bB$

Ex (6) $L = \{ a^n b^{2n} \quad \cup \quad a^n b^{2n} \}$

$S1 \qquad\qquad S2$

$S \to S1 \mid S2$

$S1 \to aS_1b \mid \lambda$

$S2 \to aS_2bb \mid \lambda$

Context Free Languages closed under union.

Ex (7) Language of proper nesting.

( ) ( ) or (( ))

or ((( ) ( )) ( ))

For every open bracket there is a closing bracket

Using stack we'll match

→ Helps in nesting ((( )))

$S \rightarrow (S) \mid SS \mid \lambda$

↳ we can have brackets side by side

Ex (8) Proper nesting of { }, [ ], ( )

{ [( )] } ( )

$S \rightarrow (S) \mid [S] \mid \{S\} \mid SS \mid \lambda$

we can have ( ) or [ ] or { }

Ex (9) Language to generate Arithmetic Expression

$\Sigma = \{ +, /, *, -, id, (,), num \}$

num → numbers

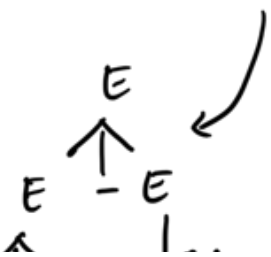
id → variable name

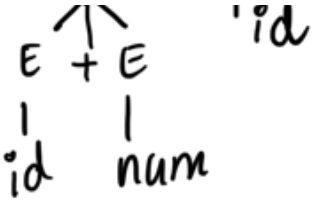
Like: a+4-c

E → Expression

$E \rightarrow E + E \mid E * E \mid E / E \mid E - E \mid (E) \mid num \mid id$

Generating parse tree a+4-c





(Q10) Language for nested if-else

```
if (condition)
    { if (condition)
          statement
      else
          statement
    }
else
    statement
```

```
if condition then
  { statement
  }
  else
      { if condn then
          { statement }
        else
      }
```

[{ } to prevent Dangling if, else]

Σ = { if, condn, then, statement, {, } }

{ statement } can be replaced

S → if condn then S | if condn then else S

| { statement }

Q (11) Variable declarations in C.

↳ DATA TYPES.

- int a, char a, byte a, short, float, double etc.
- type ↳ id

OR we can have list of id's int a,b,c ;

D → type list
list → list, id | id
Type → char | float | int | double

(list can be list of id's separated by comma OR can be a single id.)

(Q12) Nested while loop

```
while (condn) then
  {  while condn then
       { statement ;
       }
  }
```

S → while (condn) then S |
{statement }

Combination of all

S → while (Condn) S | {statement}
do S while (condn) |
If (condn) then S |
If (condn) S else S |
D | E

This combines while loop, do-while loop, if, if-else, Declaration, Expression evaluation

× END OF LECTURE 2 - UNIT 3 ×

PARSING & AMBIGUITY

Context Free Grammar ↳ Linear — Only 1 Non-terminal on RHS
↳ Non Linear — can have more

How do we know, S → SS which terminal to be expanded first

* If the right most terminal is expanded at every step, it is called

RIGHT MOST DERIVATION.

* If the left most terminal is expanded at every step, it is called

LEFT MOST DERIVATION.

LEFT MOST DERIVATION

Given $w = abba$

Grammar

$S \to aSb \mid bSa \mid \lambda \mid SS$

$S \overset{lm}{\Rightarrow} SS$

$S \overset{lm}{\Rightarrow} aSbS \quad (S \to aSb)$

$S \overset{lm}{\Rightarrow} abS \quad (S \to \lambda)$

$S \overset{lm}{\Rightarrow} abbSa$

$S \overset{lm}{\Rightarrow} abba$ //

RIGHT MOST DERIVATION.

$S \to aSb \mid bSa \mid \lambda \mid SS$

$w = abba$

$S \overset{rm}{\Rightarrow} SS$

$S \overset{rm}{\Rightarrow} SbSa$

$S \overset{rm}{\Rightarrow} Sba$

$S \overset{rm}{\Rightarrow} aSbba$

$S \overset{rm}{\Rightarrow} abb$

We say a GRAMMAR is ambigious if $\exists\ w \in L(G)$, $\exists$ 2 different structures / parse trees / or different derivations

2 Right Most, 2 left most.

Given G: $E \to E+E \mid E*E \mid E/E \mid E-E \mid id \mid num \mid (E)$

Given $w = id * id + id$

Left most Derivation -1

Left most Derivation 2

$E \Rightarrow E * E$

$E \Rightarrow id * E$

$E \Rightarrow id * E + E$

$E \Rightarrow id * id + id$

This is wrong addition happens first

$\longleftrightarrow$ Thus AMBIGUOUS.

$E \Rightarrow E + E$

$E \Rightarrow E * E + E$

$E \Rightarrow id * id + E$

$E \Rightarrow id * id + id$. Mult, firs

↳ Addition happe later

How do we know a grammar is ambiguous?

↳ Hit & Trial

↳ NO Algorithm

↳ Minimal string method.

How to eliminate ambiguity?

↳ Operators we can convert ↳ others we have to change

(Q) Given $S \to aS \mid Sa \mid \lambda$, check whether is it ambiguous or not.

if λ = Minimal string CAN'T PROVE

if a = Yes

$S \overset{lm}{\Rightarrow} aS$ (using $S \to \lambda$)

$S \overset{lm}{\Rightarrow} a$

$S \overset{lm}{\Rightarrow} Sa$ (using $S \to \lambda$)

$S \overset{lm}{\Rightarrow} a$

Two different structures / derivations

(Q) $S \to aS \mid bS \mid c \mid \lambda$

(Q) $S \to aSbS \mid bSaS \mid \lambda$

Minimal string = λ

= ab, ba

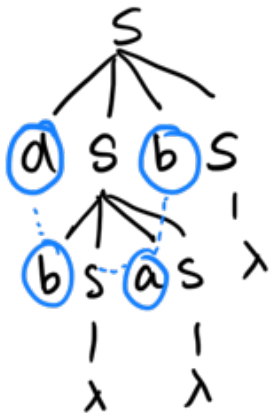
= abba

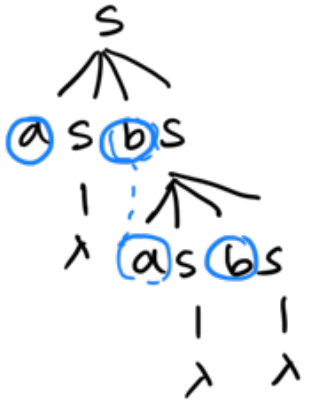
= abab

can't prove

can't prove

X





$S \overset{lm}{\Rightarrow} aSbS$

$S \overset{lm}{\Rightarrow} aSbSaSbS$

$S \overset{lm}{\Rightarrow} ababab$

$S \overset{lm}{\Rightarrow} aSbS$

$S \overset{lm}{\Rightarrow} abS$

$S \overset{lm}{\Rightarrow} abaSbS$

$S \overset{lm}{\Rightarrow} ababab$

(Q) $R \to R+R \mid RR \mid R^* \mid a \mid b \mid c$

a+bc

$R \Rightarrow R+R \quad (R \to RR)$

$\Rightarrow a+RR \quad (R \to b)$

$\Rightarrow a+bR \quad (R \to R*)$

$\Rightarrow a+bC \quad (R \to c)$

$R \Rightarrow RR$

$\Rightarrow R+RR$

$\Rightarrow a+RR$

$\Rightarrow a+bR$

$\Rightarrow a+bC$

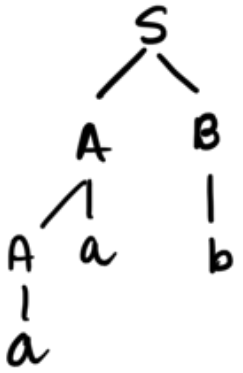
(Q) $S \to AB \mid aaB$

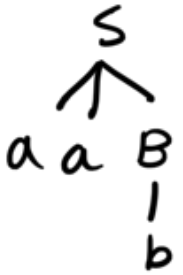
$A \to a \mid Aa$

$B \to b$

Minimal = ab ×

= aab ✓





$S \Rightarrow AB$

$\Rightarrow Aa B$

$\Rightarrow aaB$

$\Rightarrow aab$

$S \Rightarrow aaB$

$\Rightarrow aab$

(Q) $S \to 1S \mid 11S \mid \lambda$

λ – X

1 – X

11 – ✓

$S \Rightarrow 1S$

$\Rightarrow 11S$

$\Rightarrow 11$

$S \Rightarrow 1S$

$\Rightarrow 11S$

$\Rightarrow 11$

LANGUAGE AMBIGUITY :-

Grammar Ambiguity vs Language Ambiguity

For a given language $L = \{G_1, G_2, G_3, \ldots, G_n\}$ } Multiple Grammars.

If all different grammars for L are ambiguous, the language is ambiguous.

(Q) a^n , $n \geq 0$

$S \Rightarrow aS \mid \lambda$ } unambiguous.

$\Leftarrow \Rightarrow S \mid \lambda$ } Multiple

$S \Rightarrow aS \mid a \mid \lambda$

$S \Rightarrow Sa \mid a \mid \lambda$

$S \Rightarrow aS \mid Sa \mid \lambda$

Ambiguous

Multiple Grammars for same language

not inherently

$\therefore$ Language is unambiguous.

(a) Union of $a^n b^n c^m$ and $a^n b^m c^m$ $\quad n, m \geq 1$

$S_1 \rightarrow Ac$

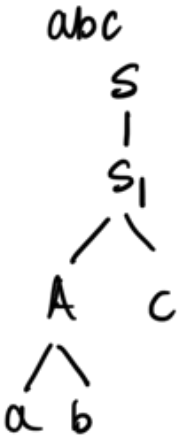
$A \rightarrow aAb \mid ab$

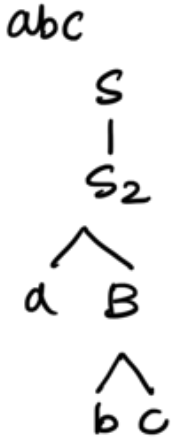
$S \rightarrow S_1 \mid S_2$

$S_2 \rightarrow aB$

$B \rightarrow bBc \mid bc$

Intersection $\Rightarrow$ Equal $a^n b^n c^n$





any language with intersection like this will always have 2 LMD's, 2 RMD's

The grammar is Ambiguous.

$\therefore$ This language is AMBIGUOUS.

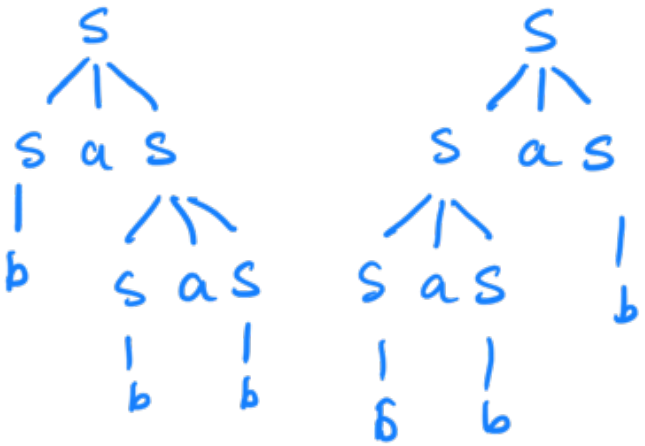
(Q) Can you prove that the following grammars represent an inherently ambiguous language.

Given G_1, prove that it is ambiguous
check if you can provide an unambiguous grammar (if possible). This proves that the language is unambiguous.

Ex 1: G_1: $S \rightarrow SaS \mid b$

STEP 1: PROVE AMBIGUITY

$w = babab$



$S \Rightarrow \underline{S}aS$
$\Rightarrow bS$
$\Rightarrow basas$
$\Rightarrow babaS$
$\Rightarrow babab$

$S \Rightarrow SaS$
$\Rightarrow \underline{S}asaS$
$\Rightarrow basa\underline{S}$
$\Rightarrow babaS$
$\Rightarrow babab$ //

STEP 2: understand what is $L(G_1)$
Types of strings here

$L = \{ ba\underline{b}, ba\underline{ba}\underline{b}, \underline{ba}\underline{ba}\underline{ba}\underline{b} \cdots \}$

$(ba)^* b$

Equal ba and extra b.

STEP 3: Find the grammar.

$S \rightarrow Ab$

$A \rightarrow baA \mid \lambda$

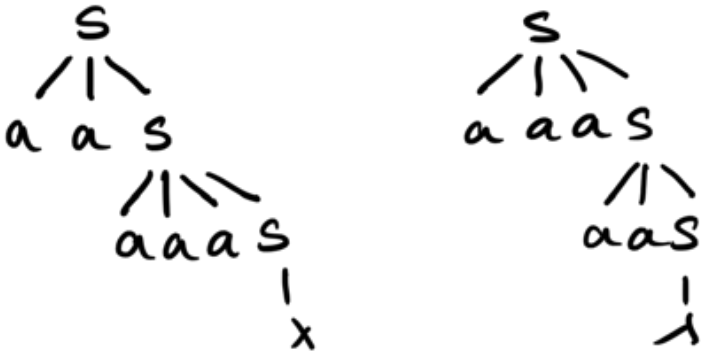
This is unambigous

If you can provide a regex for the language that language is unambigous.

(Q2) $S \rightarrow aaS \mid aaaS \mid \lambda$

Can you prove L is inherently ambiguous?

Prove Ambiguity : $w = aaaaa$



Step 2: Understand language

$L = \{ \underline{aa}, \underline{aaa}, \underline{aaaa}, \underline{aa}\underline{aaa}, \underline{aaaaaa} \cdots \}$

$((aa)\, a^* + \lambda)$

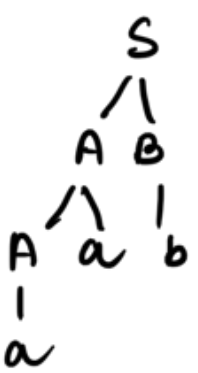
$S \rightarrow aaA \mid \lambda$

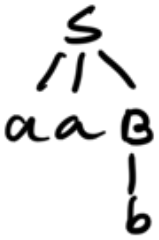
$A \rightarrow aA$

So L is unambiguous.

(Q) $S \rightarrow AB \mid aaB$
$A \rightarrow a \mid Aa$
$B \rightarrow b$

Proving ambiguity :- $w = aab$





Language = { ab, aab, aaab , aaaab ... }

Regex $a^*(ab)$

$S \rightarrow Aab$
$A \rightarrow aA \mid \lambda$

} Unambiguous.

If the grammar contains operators we can utilize assosiativity & precedence.

Example

E → E + E | E - E | E * E | E / E | (E) | id | num

For id + id * id.

Associativity

* - Left to Right

Precedence

* > +

(Operation that must be done first must be in the lower part of the parse tree.

If the operator is left associative (*, /, +, -)

+, - ↓ low precedence

*, / ↓ high precedence

E → E + T | E - T | T

Left Associative E will be the leftmost symbol.

lower precedence will come first

T → T * F | T / F | F

Left Associative.

F → (E) | id

Bracket has highest priority

2^{3^2}

2^{9}

2^{9}

↑ Power operator

Right Associative.

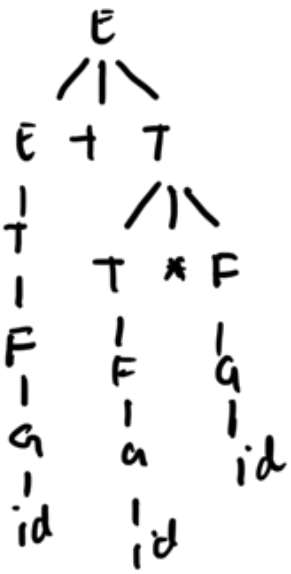
E → E + T | E - T | T

T → T * F | T / F | F

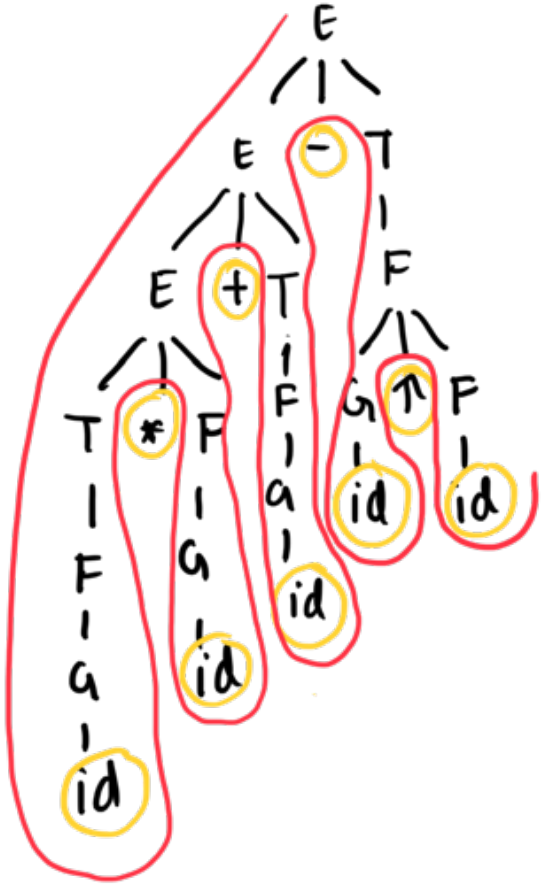
F → G ^ F | G Right Associative.

G → (E) | id

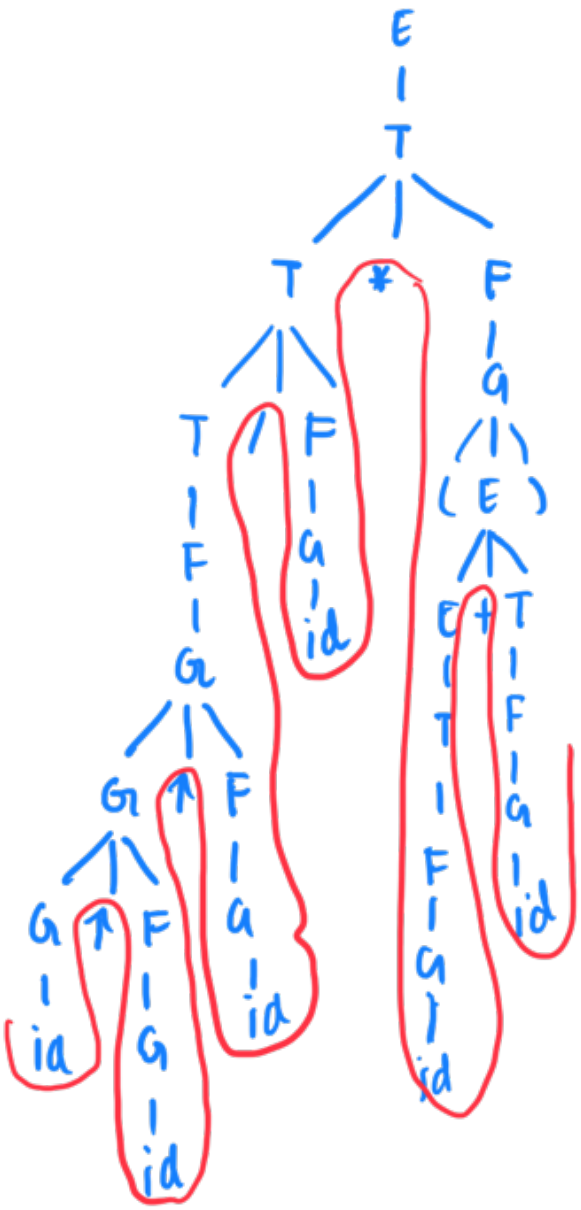
Ex id + id * id



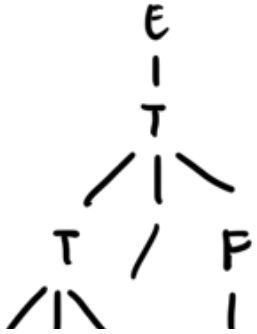
Ex id * id + id - id ↑ i

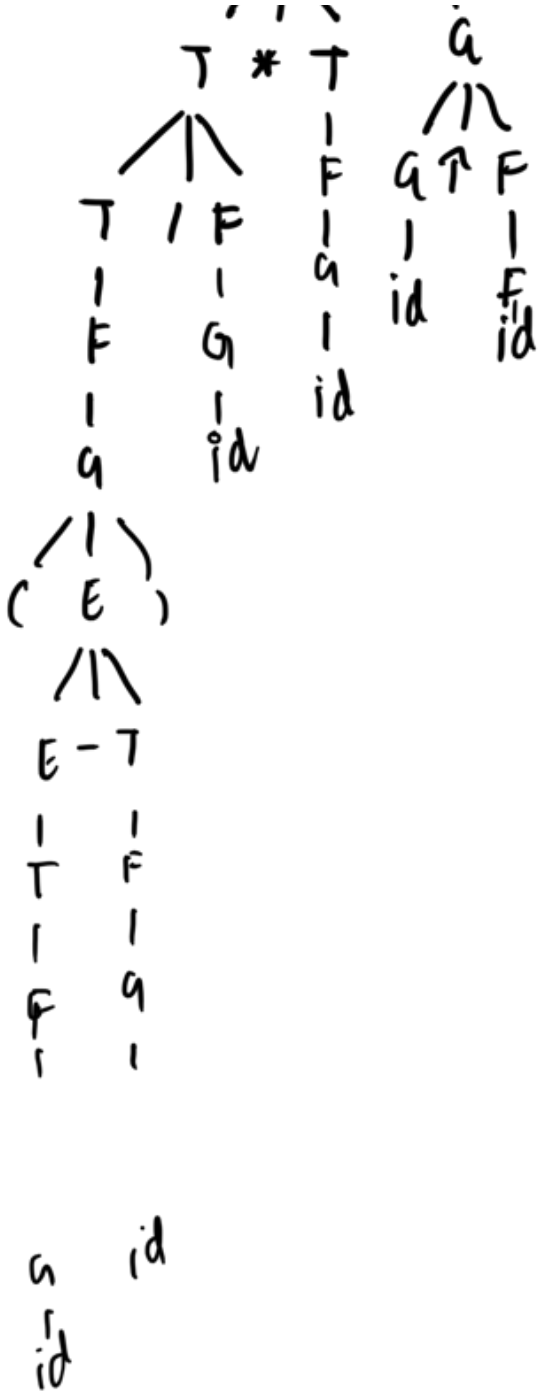


① id ↑ id ↑ id / id * (id + id)



(Q) (id - id) / id * id / id ↑ id





(Q) $R \rightarrow R+R \mid RR \mid R^* \mid a \mid b \mid c$

How do we eliminate ambiguity ?

+ } Associative
• } Precedence
*

lowest precedence

highest precedence

R → R+T | T
T → T.F | F
F → F* | a | b | C

a+bc*

(Q) convert given grammar to unambiguous.

B → B || B | B && B | ! B | true | false

&& And
|| OR
Not !

Precedence ↓

B → ! B | T
T → T || F | F
F → F && G | G

Gr → True|false

(Q) A → A $ B | B

B → B # C | C

C → C @ D | D

D → b

Highest @ > # > $ → left

→ left

@ → Left Associative