CS & IT ENGINEERING

Compiler Design

Lexical Analysis & Syntax Analysis

Lecture No. 4

By- DEVA Sir

- H.W. Questions
- Basics of parsing
  - CFG
    - Definitions
    - Types of Derivations
    - Ambiguous & Unambiguous

(18) `for(1,2,3);` ⇒ Syntax Error

(19) `fro(1,2,3);` ⇒ No Syntax Error
[either semantic error or Linker error]

(20) `for(2;2;2);` ⇒ No Compilation Error

(21) `fro(2;2;2);` ⇒ Syntax error

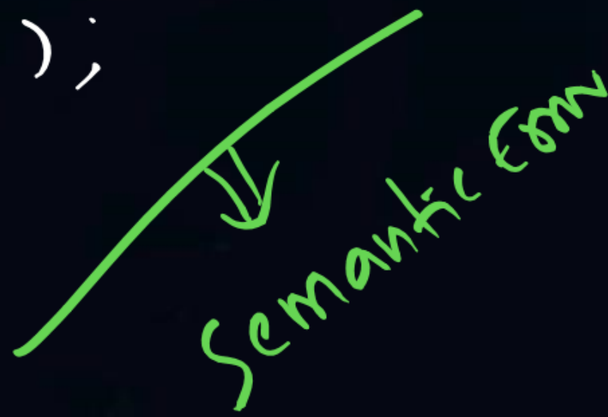
H.W.

```
#include <stdio.h>
void main( )
{
    f( );
}
```

Compiler assumes f( ) definition may exist in other file

Linker Error

```
Void main( )
{
    f( );
}
```



MCQ

Consider the following ANSI C program:

```
int main() {
    Integer x;
    return 0;
}
```

⇒ Semantic Error

⇒ No Syntax Error

typedef int Integer;

is missing

Which one of the following phases in a seven-phase C complier will throw an error?

[GATE-2021-CS: 1M]

- A Lexical analyzer
- B Syntax analyzer
- C Semantic analyzer
- D Machine dependent optimizer

MCQ

Consider the following statements:

I. Symbol table is accessed only during lexical analysis and syntax analysis. FALSE

II. Compilers for programming languages that support recursion necessarily need heap storage for memory allocation in the run-time environment. FALSE

III. Errors violating the condition any variable must be declared before its use are detected during syntax analysis. FALSE

Which of the above statements is/are TRUE?

[GATE-2020-CS: 1M]

A. I only

B. I and III only

C. II only

D. None of I, II and III

MCQ

A lexical analyzer uses the following patterns to recognize three tokens T_1, T_2 and T_3 over the alphabet {a, b, c}.

T_1: a? (b|c)* a = $(\varepsilon + a)(b+c)^* a$

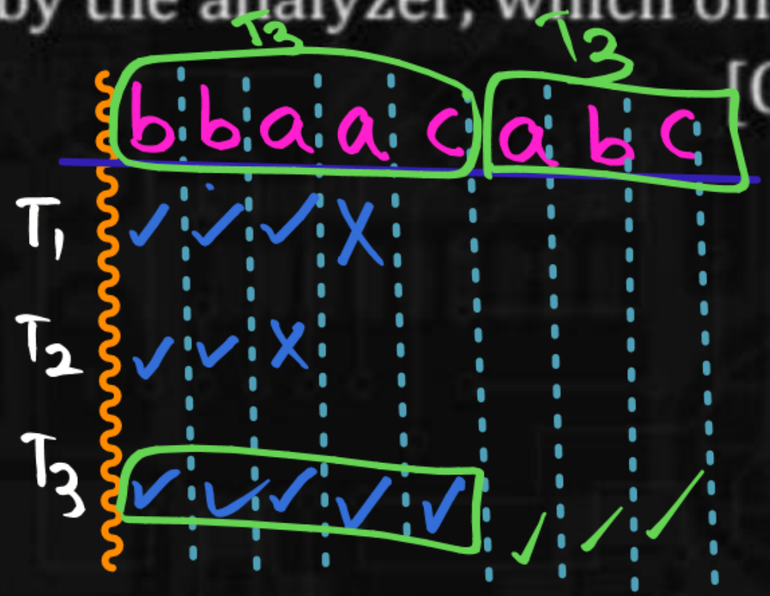
T_2: b? (a|c)* b = $(\varepsilon + b)(a+c)^* b$

T_3: c? (b|a)* c = $(\varepsilon + c)(b+a)^* c$

$x? = \varepsilon + x$

Note that 'x?' means 0 or 1 occurrence of the symbol x. Note also that the analyzer outputs the token that matches the longest possible prefix.

If the string *bbaacabc* is processed by the analyzer, which one of the following is the sequence of tokens it outputs? [GATE-2018-CS: 2M]



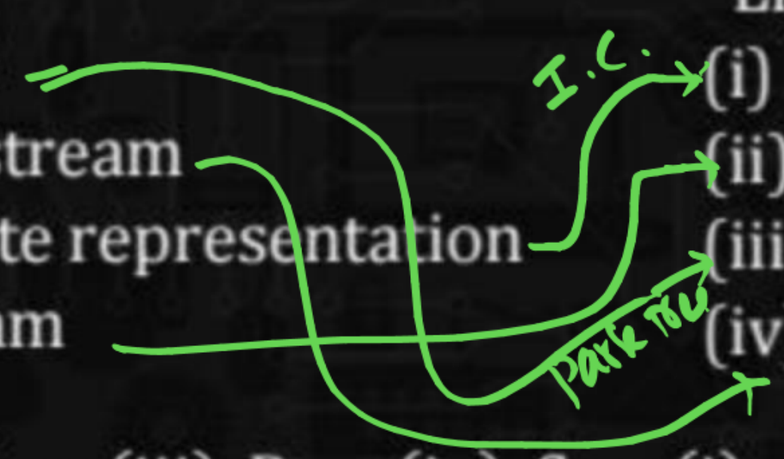
- (A) $T_1 T_2 T_3$
- (B) $T_1 T_1 T_3$
- (C) $T_2 T_1 T_3$
- (D) $T_3 T_3$ ✓ $T_3\ T_3$

MCQ

Match the following according to input from List-I to the compiler phase in the List-II that processes it:

List-I	List-II
(P) Syntax tree	(i) Code generator
(Q) Character stream	(ii) Syntax analyser
(R) Intermediate representation	(iii) Semantic analyser
(S) Token stream	(iv) Lexical analyser

[GATE-2017-CS: 1M]



(A) P → (ii), Q → (iii), R → (iv), S → (i)

(B) P → (ii), Q → (i), R → (iii), S → (iv)

(C) P → (iii), Q → (iv), R → (i), S → (ii)

(D) P → (i), Q → (iv), R → (ii), S → (iii)

(after changing given data)

P – i
Q – iv
R – i
S – iii

Given data

MCQ

Match the following

List-I	List-II
P: Lexical analysis	(i) Leftmost derivation
Q: Top down parsing	(ii) Type checking
R: Semantic analysis	(iii) Regular expressions
S: Runtime environment	(iv) Activation records

[GATE-2016-CS: 1M]

A. P→(i),Q→(ii),R→(iv),S→(iii)

B. P→(iii),Q→(i),R→(ii),S→(iv)

C. P→(ii),Q→(iii),R→(i),S→(iv)

D. P→(iv),Q→(i),R→(ii),S→(iii)

MCQ

In a compiler, keywords of a language are recognized during

[GATE-2021-CS: 1M]

A. parsing of the program

B. the code generation

C. the lexical analysis of the program

D. dataflow analysis

MCQ

Which data structure in a compiler is used for managing information about variables and their attributes?

[GATE-2021-CS: 1M]

Identifiers

A. Abstract syntax tree

B. Symbol table

C. Semantic stack

D. Parse table

Context Free
(Structure)
(Syntax)

S + V + O

✓ I am student

✓ I are student

Context Sensitive
(Semantic)
(Meaning)

I am student (Semantically correct)

I are student (Semantically wrong)

Context Free
(Structure)
(Syntax)

Integer x ;

Syntax correct

Context Sensitive
(Semantic)
(Meaning)

Integer x ;

Semantically wrong

Context Free Grammar [CFG]

→ It can be used to represent Syntax/structure of programming languages.

$$CFG = (V, T, P, S)$$

- V → Set of Variables (non-terminals)
- T → Set of terminals
- P → Set of Rules (productions) (Production Rules)
- S → Start Symbol, $S \in V$

Each rule in P:

LHS → RHS

V → Any Sequence

$$V \rightarrow (VUT)^*$$

Identify CFGs :

① $S \to \varepsilon$ ✓

② $S \to Sab \mid a$ ✓

③ $S \to ABS \mid a$

$AB \to a$ ✗

④ $S \to aSb \mid SS \mid a$ ✓

⑤ $S \to S \mid a \mid \varepsilon$ ✓

⑥ $S \to aSb$

$aS \to bb$ ✗

S = Program

8 rules

Program → RI( ){D;} (rule 1)

R → Void | int (rule 2 | rule 3)

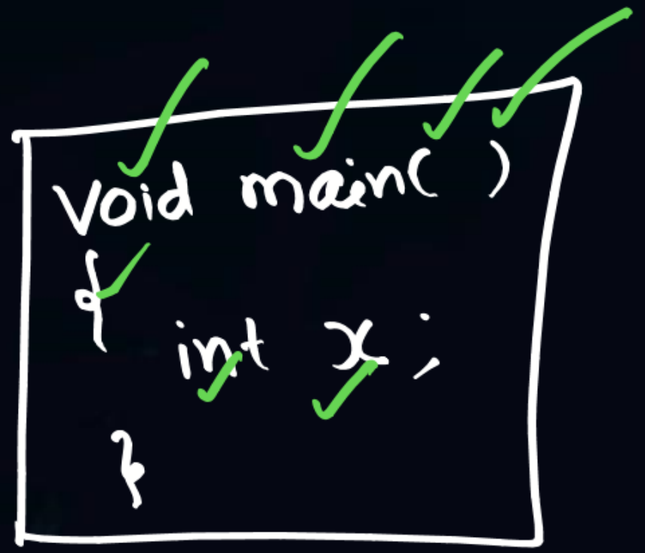
I → Identifier (4)

D → TI (5)

T → int | float | char (6 | 7 | 8)

V = {program, R, I, D, T}

T = {(,), {, ;, }, void, int, Identifier, float, char}



$S \rightarrow ab$

OR

$S \rightarrow Ab$

$A \rightarrow a$

OR

$S \rightarrow Ab$

$A \rightarrow aB$

$B \rightarrow \varepsilon$

OR

$S \rightarrow AB$

$A \rightarrow a$

$B \rightarrow b$

ab

Derivation of a String:

Linear
- I) Left Most Derivation (LMD)
- II) Right Most Derivation (RMD)

Non-linear
- III) Parse Tree (Derivation Tree) (PT)

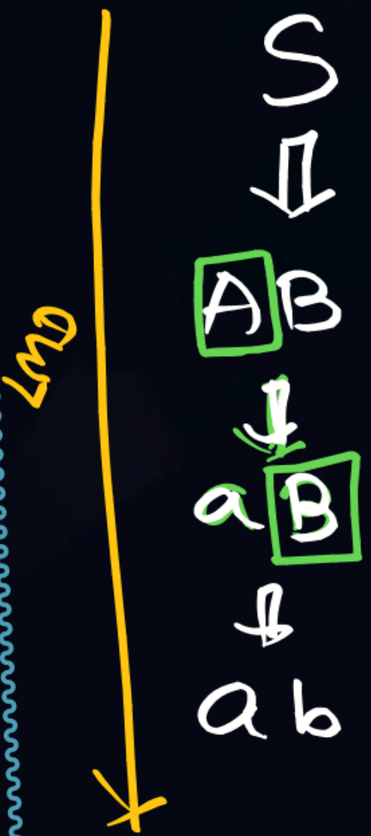
$S \to AB$
$A \to a$
$B \to b$

CFG

$w = ab$

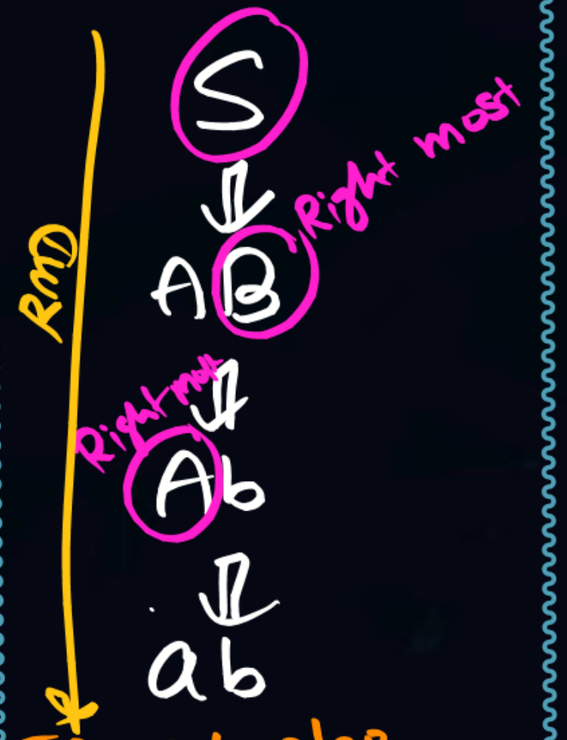
string

LMD



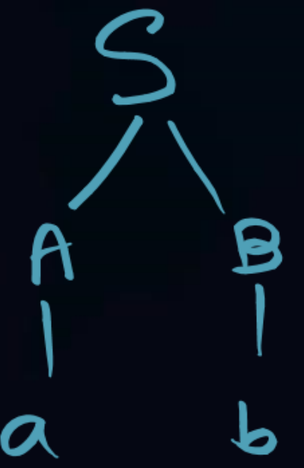
Note: In every step, left most nonterminal Substituted

RMD



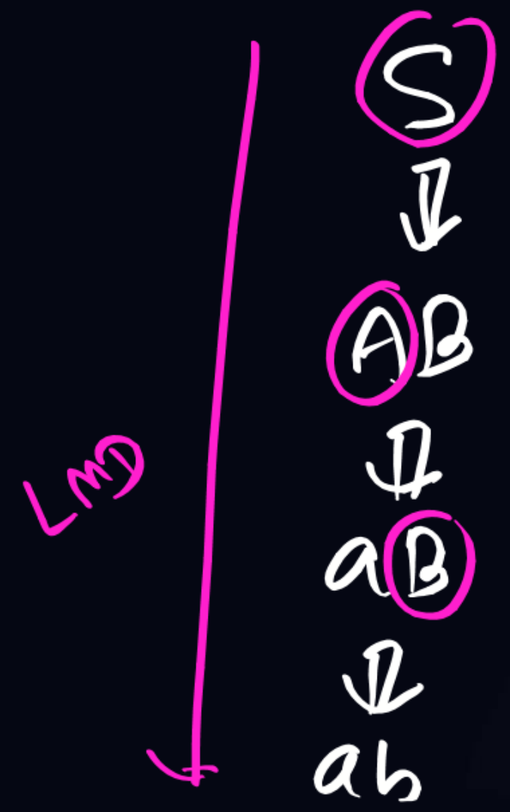
In each step, Right most nonterminal is Substituted

PT



Leaf : terminal or ε

Nonleaf : Non-term.



Length of Derivation

=

Derivation Length

=

No. of steps

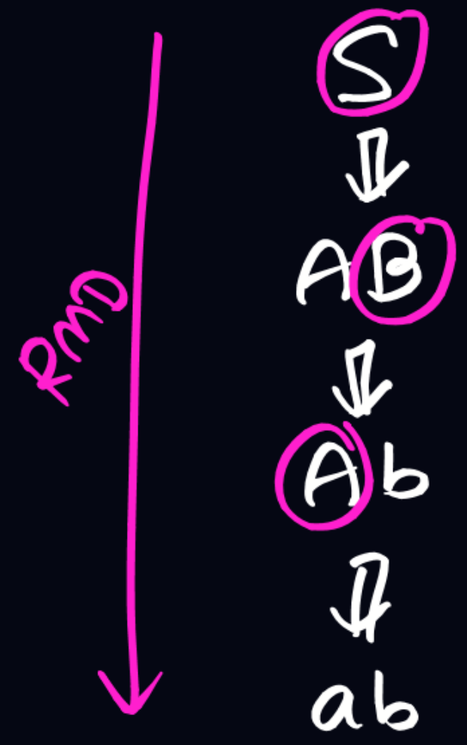
=

No. of substitutions

$= 3$

What is LMD order?

= SAB

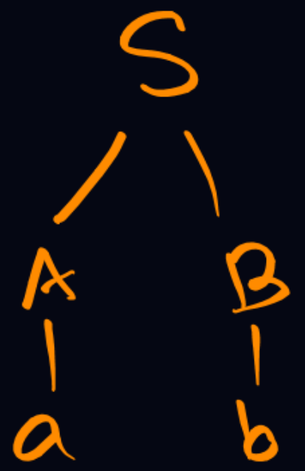


Length of derivation = 3

RMD order:

S, B, A

Note: LMD and RMD need not be same



Derivation Length for "ab"

=

No. of non leaf nodes

=

3

$S \to a \mid Aa$

$A \to \varepsilon \mid b$

$\Rightarrow$

$S \to a$

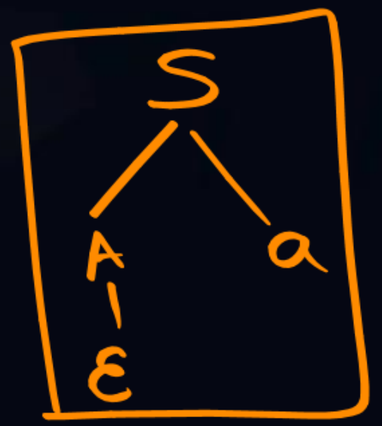
$S \to Aa$

$A \to \varepsilon$

$A \to b$

$w = a \Rightarrow$ How many derivations for w ?

OR



$= a$

$\varepsilon a = a$

= 2 derivations

= 2 Parse Trees

= 2 LMDs

= 2 RMDs

For particular string:

No. of derivations

=

No. of PTs

=

No. of LMDs

=

No. of RMDs

$S \to AB$

$A \to a$

$B \to b$

$w = ab$

No. of LMDs for ab

$=$

No. of RMDs for ab

$=$

1

$\Rightarrow$ In this example :

LMD & RMD are different

No. of steps in LMD

$=$

No. of steps in RMD

$=$

3

① $S \rightarrow SS \mid \varepsilon$

$w = \varepsilon$

No. of derivations = ?

$S \Rightarrow \varepsilon$

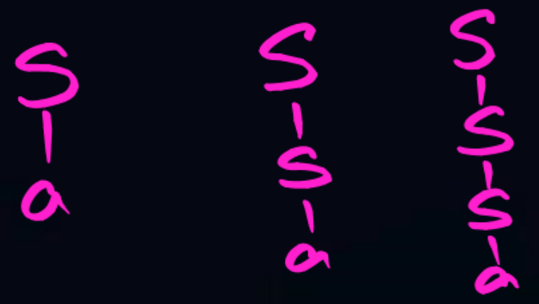
$S \Rightarrow SS \Rightarrow \varepsilon\varepsilon = \varepsilon$

$S \Rightarrow SS \Rightarrow S\varepsilon \Rightarrow SS\varepsilon \Rightarrow \varepsilon\varepsilon\varepsilon = \varepsilon$

. . . . . .

Infinite parse trees

② $S \rightarrow S \mid a \mid ab$



$S \rightarrow S$
$S \rightarrow a$
$S \rightarrow ab$

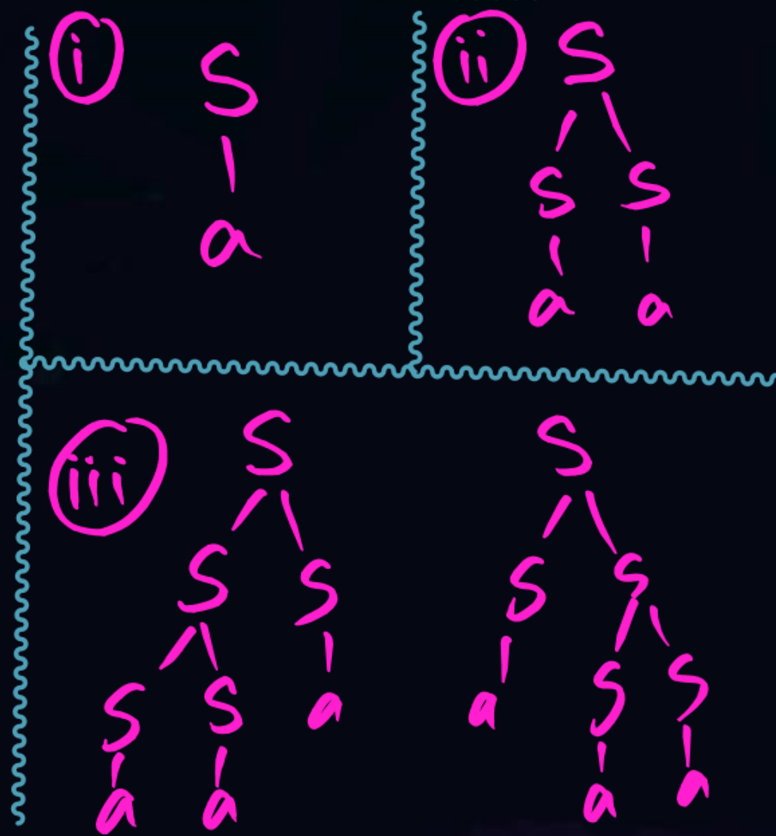
How many Parse Trees for "a" ? = Infinite

③ $S \rightarrow SS \mid a$

i) How many PTs for "a" ? = 1

ii) How many PTs for "aa" ? = 1

iii) How many PTs for "aaa" ? = 2



④ $S \rightarrow SSS \mid a$

i) How many PTs for "a" ? = 1

ii) " " for "aa" ? = 0

iii) " " " "aaa" ? = 1

Types of CFGs based on derivations:

① Ambiguous CFG

$S \to \varepsilon \mid a \mid b \mid A$

$A \to a$

there is some string, which has more than one derivation.

② Unambiguous CFG

$S \to \varepsilon \mid a \mid b$

$\varepsilon \Rightarrow$ 1 PT
$a \Rightarrow$ 1 PT
$b \Rightarrow$ 1 PT

Every string has only 1 PT

Types of CFGs based on derivations:

① Ambiguous CFG

Some string, >1 PT

$\exists w \in L(CFG) \Rightarrow >1$ PT

② Unambiguous CFG

Every string, only 1 PT

$\nexists w \in L(CFG) \Rightarrow 1$ PT

Types of CFGs based on derivations:

① Ambiguous CFG

↓

Easy to write

② Unambiguous CFG

↓

Better

Identify Ambiguous and Unambiguous CFGs.

① $S \to a$

~~Ambiguous~~

Unambiguous ✓

$a \Rightarrow$ 1 PT

② $S \to a \mid \varepsilon$

~~Ambiguous~~

Unambiguous ✓

$\varepsilon \Rightarrow$ 1

$a \Rightarrow$ 1

③ $S \to AB \mid \varepsilon$

$A \to \varepsilon$

$B \to a$

~~Ambiguous~~

Unambiguous ✓

$\varepsilon \Rightarrow$ 1 PT

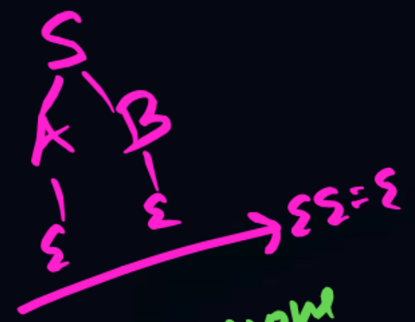
$a \Rightarrow$ 1 PT

④ $S \to AB$

$A \to \varepsilon$

$B \to \varepsilon$

$\varepsilon \Rightarrow 1$ PT



Unambiguous

⑦ $S \to AB$

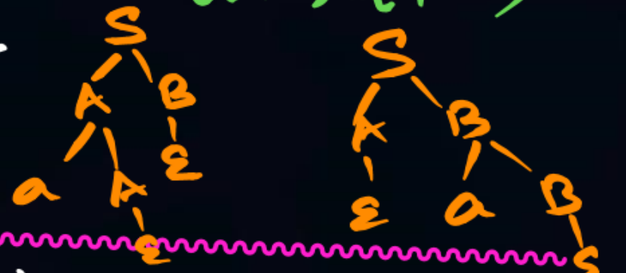
$A \to aA \mid \varepsilon$

$B \to aB \mid \varepsilon$

Ambiguous

$\varepsilon \Rightarrow 1$ PT

$a \Rightarrow 2$ PTs



⑤ $S \to AB$

$A \to a \mid \varepsilon$

$B \to b \mid \varepsilon$

$\varepsilon \Rightarrow 1$ PT

$a \Rightarrow 1$ PT

$b \Rightarrow 1$ PT

$ab \Rightarrow 1$ PT

Unambiguous

⑧ $S \to SS \mid \varepsilon$

Ambiguous

⑥ $S \to AB$

$A \to aA \mid \varepsilon$

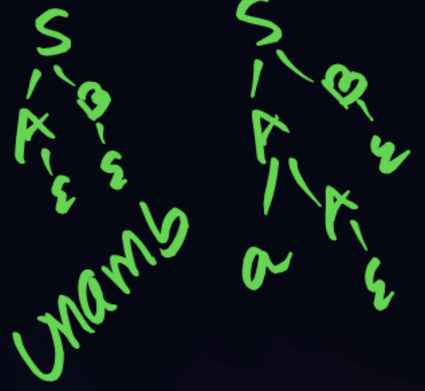
$B \to bB \mid \varepsilon$

$\varepsilon \Rightarrow 1$ PT

$a \Rightarrow 1$ PT

$b \Rightarrow 1$ PT

...



⑨ $S \to SS \mid a$

Amb.

$a \Rightarrow 1$ PT

$aa \Rightarrow 1$ PT

$aaa \Rightarrow 2$ PTs

...

$S \to AB$

$A \to aA \mid \varepsilon$

$B \to bB \mid \varepsilon$

Unambiguous

$\varepsilon \to 1$

$a \to 1$

$b \to 1$

$aa \to 1$

$ab \to 1$

$bb \to 1$

$aaa \to 1$

$aab \to 1$

$abb \to 1$

$bbb \to 1$

.

.

.

1

(10) $S \rightarrow SaS \mid b$

(11) $S \rightarrow Sa \mid bS \mid c$

(12) $E \rightarrow E+E \mid a$

(13) $S \rightarrow AB \mid ab$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bB \mid \varepsilon$

H.W.

Summary

- CFG
- Derivations
- Amb CFG
- Unamb CFG


THANK YOU GW
SOLDIERS !