CONTEXT FREE GRAMMAR

Dr. A. Beulah

AP/CSE

LEARNING OBJECTIVE

- To Understand the need of formal languages, and grammars (K3)
  - To Understand context free grammars

INTRODUCTION

- Context-free grammar is a 4-tuple **G = (N, T, P, S)** where
  - T is a finite set of terminal symbols
  - N is a finite set of nonterminals
  - P is a finite set of productions of the form

    $\alpha \rightarrow \beta$

    where $\alpha \in N$ and $\beta \in (N \cup T)^*$
  - $S \in N$ is a designated start symbol

$A \rightarrow \alpha$

NT

$(N \cup T)^*$

EXAMPLE

- $G = (\{E\}, \{+, -, *, /, (,), id\}, P, E)$,

  where P consists of

  $E \rightarrow E + E$

  $E \rightarrow E - E$

  $E \rightarrow E * E$

  $E \rightarrow E / E$

  $E \rightarrow - E$

  $E \rightarrow (E)$

  $E \rightarrow id$

NOTATIONAL CONVENTIONS USED

- Terminals
  $a,b,c,\ldots \in T$
  specific terminals: **0**, **1**, **id**, **+**
- Nonterminals
  $A,B,C,\ldots \in N$
  specific nonterminals: *expr*, *term*, *stmt*
- Grammar symbols
  $X,Y,Z \in (N \cup T)$
- Strings of terminals
  $u,v,w,x,y,z \in T^*$
- Strings of grammar symbols
  $\alpha,\beta,\gamma \in (N \cup T)^*$

CONTEXT FREE LANGUAGE

- The language generated by CFG G is defined as :
  - $L(G) = \{w \mid w \text{ is in } T^* \text{ and } S \stackrel{*}{\Rightarrow} w\}$. CFL
  - That is a string is in L(G) if
    - The string consists of terminals only
    - The string has to be derived from S only

- L is a Context Free Language (CFL), if it is L(G) for some CFG G.

APPLICATIONS OF CFG

- To design a Parser, a CFG is needed.
- The DTD (Document type Definitions) is a CFG whose language is a class of related documents.

EXAMPLE

- Construct a grammar for L=$\{0^n 1^{2n} / n \geq 1\}$

$n \geq 0$

ϵ 0

L=? L={011, 001111, 000111111 ....}

G=? $S \to 0S11 / 011$

- Construct a grammar for L={w/ |w| is odd} over {0,1}

L=? L={0,1, 000, 010, 101 ....}

G=? $S \to 0A \mid 1A$

$A \to 0S / 1S \mid \epsilon$

EXAMPLE

- Construct a grammar to generate the set of all strings over Σ={a,b} ending in a

L=? L={ a, ba, aa, baa, bba, aba, ... }

G=? ~~S→a|b~~ ~~S→Sa~~

S→b

~~S→aS|bA|~~
A→a

S→aS |bS |a

- Construct a grammar for L={w/ |w| is odd and its middle symbol is 0} over {0,1}

L=? L={0, 000, 001, 101 . . . . }

G=? S→A0B
~~A→1A|0A|ε~~

S→A0B
A→00B|01A

S→S0S

S→0|0S0|0S1|1S0|1S1

EXAMPLE

- What is the language generated by

S→aSb

S→ab

L=? $L=\{(ab)^n\}$ X

$L=\{ab, aabb, aaabbb \ldots\}$

ababab

$L=\{a^n b^n \mid n \geq 1\}$

- Construct a grammar for L={w/ w starts and ends with same symbol} over {0,1}

L=?

$L=\{0, 1, 00, 11, 010, 101, 1001, \ldots\}$

G=?

$S \rightarrow 0 \mid 1 \mid 0S0 \mid 1S1 \mid$

$\cancel{0S1} \mid \cancel{1S0}$

1101

1S1

101

$S \Rightarrow 1S1$

$\Rightarrow 1101$

$S \rightarrow 0A0 \mid 1A1 \mid 0 \mid 1$

$A \rightarrow 1A \mid 0A \mid \varepsilon$

EXAMPLE

- $L = \{(ab)^n / n \geq 1\} \cup \{(ba)^n / n \geq 1\}$

L=?

G=?

$S \rightarrow A / B$

$A \rightarrow \ldots$ (1)

$B \rightarrow \ldots$ (2)

EXAMPLE

- Construct grammar for set of all palindromes over $\Sigma=\{0,1\}$

L=?

G=?

DERIVATIONS

- $G = (\{E\}, \{+, -, *, /, (,), id\}, P, E)$,

  where P consists of

$E \rightarrow E + E$

$E \rightarrow E - E$

$E \rightarrow E * E$

$E \rightarrow E / E$

$E \rightarrow -E$

$E \rightarrow (E)$

$E \rightarrow id$

$\boxed{id * (id + id)}$

lm, rm

id * id

$E \Rightarrow E * E$

$\Rightarrow id * E \quad (E \rightarrow id)$

$\Rightarrow id * id \quad (E \rightarrow id)$

DERIVATIONS

- $E \Rightarrow E+E$

- E+E derives from E
  - we can replace E by E+E
  - to able to do this, we have to have a production rule $E \rightarrow E+E$ in our grammar.

$$
\begin{aligned}
E &\Rightarrow E+E \\
&\Rightarrow id+E \\
&\Rightarrow id+id
\end{aligned}
$$

$E \rightarrow id$

- A sequence of replacements of non-terminal symbols is called a **derivation** of id+id from E.

DERIVATIONS

- $\alpha_1 \Rightarrow \alpha_2 \Rightarrow \ldots \Rightarrow \alpha_n$ (α_n derives from α_1 or α_1 derives α_n)

  $\Rightarrow$: derives in one step

  $\overset{*}{\Rightarrow}$: derives in zero or more steps

  $\overset{+}{\Rightarrow}$: derives in one or more steps

DERIVATIONS

- A **left-most derivation** of a sentential form is one in which rules transforming the left-most non-terminal are always applied.
- $\overset{lm}{\Rightarrow}$: leftmost derivation

- A **right-most derivation** of a sentential form is one in which rules transforming the right-most non-terminal are always applied
- $\overset{rm}{\Rightarrow}$: rightmost derivation

LM AND RM DERIVATIONS

- Leftmost Derivation
- id+id*id

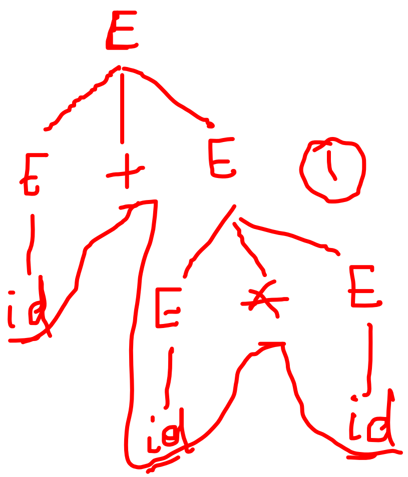
$E \overset{lm}{\Longrightarrow} E+E$

$\overset{lm}{\Longrightarrow} id+E$ 1

$\overset{lm}{\Longrightarrow} id+E*E$

$\overset{lm}{\Longrightarrow} id+id*E$

$\overset{lm}{\Longrightarrow} id+id*id$



id+id*id

- Rightmost Derivation
- id+id*id

$E \overset{rm}{\Longrightarrow} E+E$

$\overset{rm}{\Longrightarrow} E+E*E$ 1

$\overset{rm}{\Longrightarrow} E+E*id$

$\overset{rm}{\Longrightarrow} E+id*id$

$\overset{rm}{\Longrightarrow} id+id*id$

2 → LMD

2 → RMD

2 → PT

AG

LM AND RM DERIVATIONS

- Leftmost Derivation
- id+id*id

$$E \xrightarrow{lm} E*E$$

$$\xrightarrow{lm} E+E*E$$ ②

$$\xrightarrow{lm} id+E*E$$

$$\xrightarrow{lm} id+id*E$$

$$\xrightarrow{lm} id+id*id$$

- Rightmost Derivation
- id+id*id

$$E \xrightarrow{rm} E*E$$

$$\xrightarrow{rm} E*id$$ ②

$$\xrightarrow{rm} E+E*id$$

$$\xrightarrow{rm} E+id*id$$

$$\xrightarrow{rm} id+id*id$$

PARSE TREE

Dr. A. Beulah

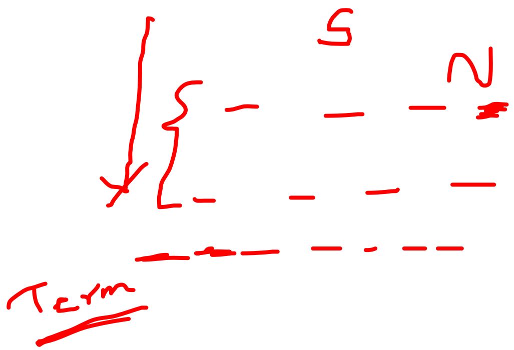
AP/CSE

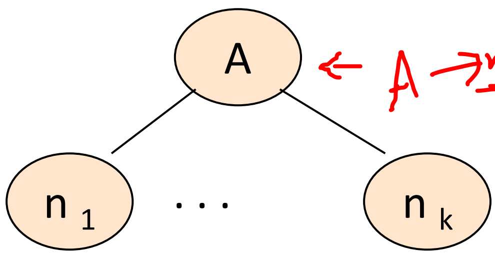
LEARNING OBJECTIVE

- To understand parse tree and ambiguous grammar (K3)

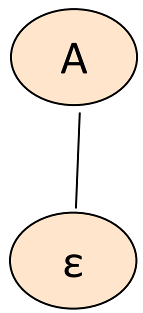
PARSE TREE

- Let G=(N, T, P, S) be a CFG.
- A tree is a derivation (or) parse tree for G if :
  - Every node has a label which is a non-terminal (or) terminal (or) ε, (i.e.) N ∪ T ∪ {ε}.
  - The label of the root is S (Start symbol)
  - The internal nodes must be in N (non - terminal) labeled as A.
  - If A is a label for a node and nodes $n_1, n_2, \ldots\ldots\ldots n_k$ are the sons of node n, in order from the left, then $A \rightarrow n_1 n_2 \ldots n_k$ must be a production in P.
  - If a node has label ε, then the vertex is a leaf and is the only son of its father.





$\Leftarrow A \rightarrow n_1 n_2 \cdots n_k$



$\Leftarrow A \rightarrow \varepsilon$

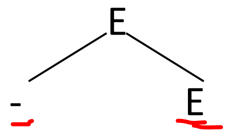
DERIVATION TREE/ PARSE TREE

$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow - E$
$E \rightarrow (E)$
$E \rightarrow id$

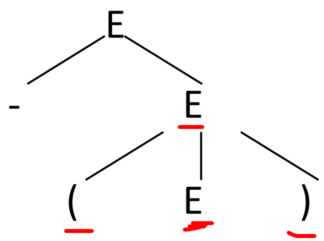
$$
\begin{aligned}
E &\overset{lm}{\Rightarrow} -E \\
&\overset{lm}{\Rightarrow} -(E) \\
&\overset{lm}{\Rightarrow} -(E+E) \\
&\overset{lm}{\Rightarrow} -(id+E) \\
&\overset{lm}{\Rightarrow} -(id+id)
\end{aligned}
$$

- Inner nodes of a parse tree are non-terminal symbols.
- The leaves of a parse tree are terminal symbols.
- A parse tree can be seen as a graphical representation of a derivation.

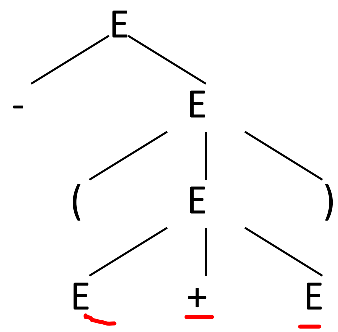
$E \Rightarrow -E$



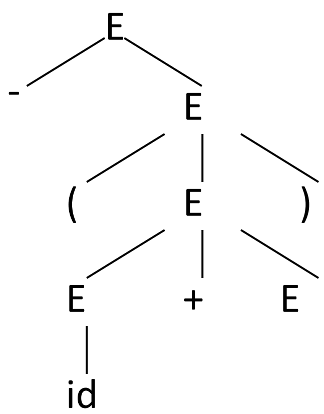
$\Rightarrow -(E)$



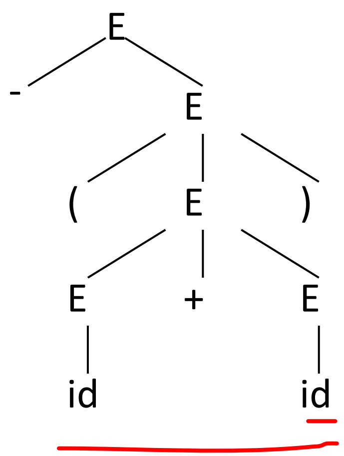
$\Rightarrow -(E+E)$



$\Rightarrow -(id+E)$



$\Rightarrow -(id+id)$

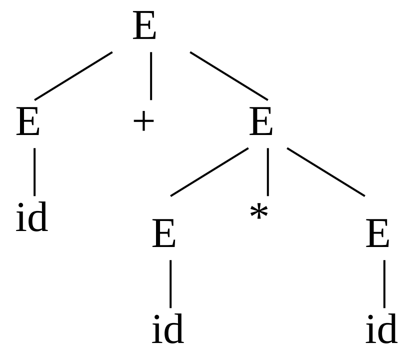


AMBIGUOUS GRAMMAR

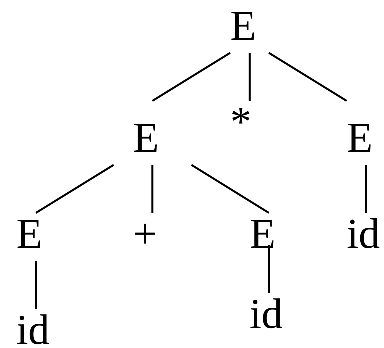
- A grammar G is ambiguous if there is a word $w \in L(G)$ having at least two different parse trees
- CFG is ambiguous $\Leftrightarrow$ any of following equivalent statements:
  - $\exists$ string w with more than one derivation trees.
  - $\exists$ string w with more than one leftmost derivations.
  - $\exists$ string w with more than one rightmost derivations.

AMBIGUOUS GRAMMAR

$$
\begin{aligned}
E &\stackrel{lm}{\Rightarrow} E+E \\
&\stackrel{lm}{\Rightarrow} id+E \\
&\stackrel{lm}{\Rightarrow} id+E*E \\
&\stackrel{lm}{\Rightarrow} id+id*E \\
&\stackrel{lm}{\Rightarrow} id+id*id
\end{aligned}
$$



$$
\begin{aligned}
E &\stackrel{lm}{\Rightarrow} E*E \\
&\stackrel{lm}{\Rightarrow} E+E*E \\
&\stackrel{lm}{\Rightarrow} id+E*E \\
&\stackrel{lm}{\Rightarrow} id+id*E \\
&\stackrel{lm}{\Rightarrow} id+id*id
\end{aligned}
$$



EXAMPLE

$S \to SbS / a$

$\underline{\underline{AG}}$

abab a

EXAMPLE

$S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

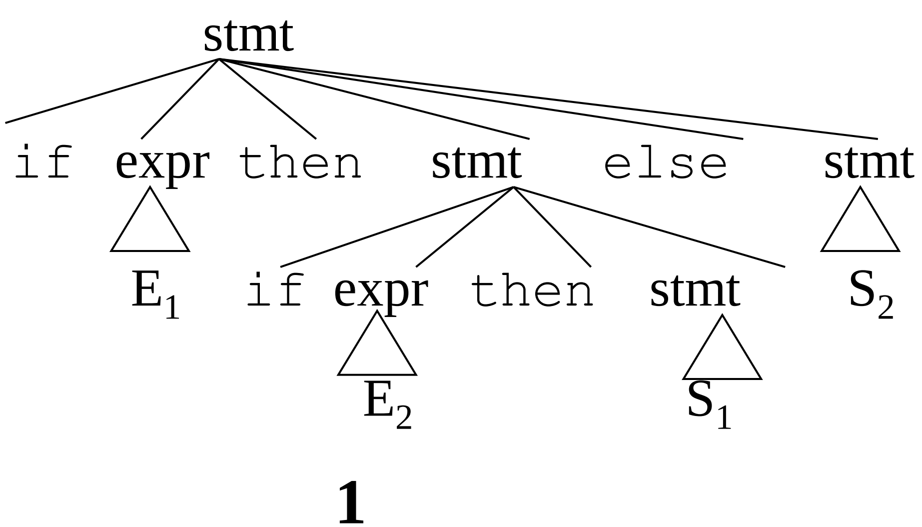
$B \rightarrow 1 \mid 1S \mid 0BB$

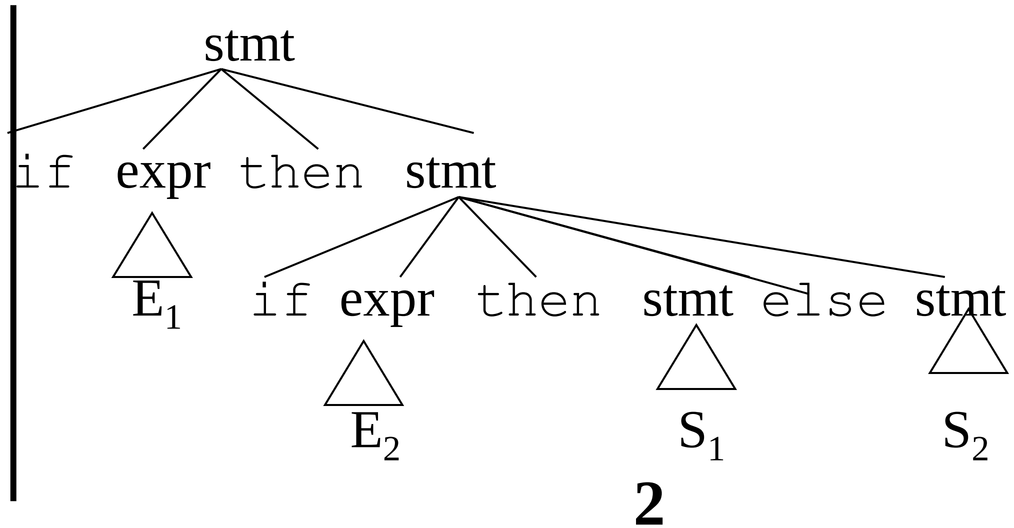
AG 00110101

AMBIGUOUS GRAMMAR

stmt → `if` expr `then` stmt |
`if` expr `then` stmt `else` stmt | otherstmts

$$\texttt{if } E_1 \texttt{ then } \texttt{if } E_2 \texttt{ then } S_1 \texttt{ else } S_2$$



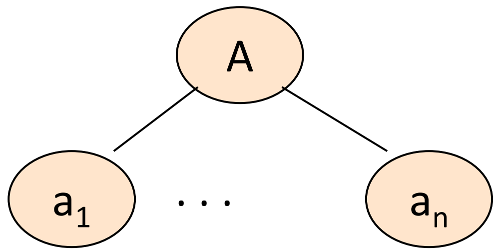


RELATIONSHIP BETWEEN DERIVATION AND DERIVATION TREES

- Let G = (N, Σ, P, S) be a context free grammar (CFG). Then $S \Rightarrow^* \alpha$ if and only if there is a derivation tree for G which yield α.
- We'll prove for any A in N, $A \Rightarrow^* \alpha$ if and only if there is a A- tree which yields α :
  - Part 1
    - If there is a parse tree with root labeled A and yield α, then $A \Rightarrow^* \alpha$.
  - Part 2
    - If $A \Rightarrow^* \alpha$, then there is a parse tree with root A and yield α.

PROOF – PART 1

- Induction on the *height* (length of the longest path from the root) of the tree.
- Basis: height 1. Tree looks like



- $A \rightarrow \alpha_1 \ldots \alpha_n$ must be α production.
- Thus, $A \Rightarrow \alpha_1 \ldots \alpha_n$.

PART 1

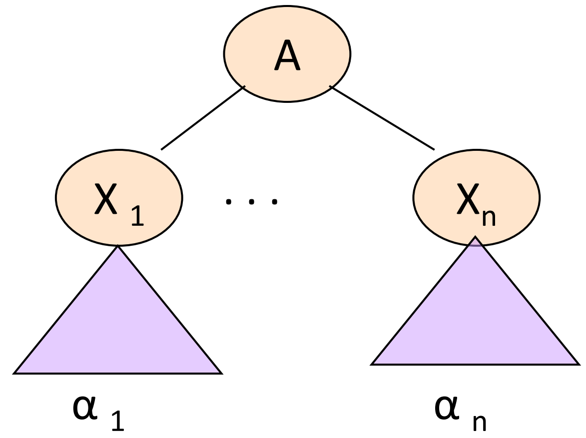
- Induction
- Assume (1) for trees of height >1
- By IH, $X_i \Rightarrow^* \alpha_i$.
  - Note: if X_i is a terminal, then $X_i = \alpha_i$.
- Thus, $A \Rightarrow X_1 \ldots X_n$

$$\Rightarrow^* \alpha_1 X_2 \ldots X_n$$

$$\Rightarrow^* \alpha_1 \alpha_2 X_3 \ldots X_n$$

$$\Rightarrow^* \ldots$$

$$\Rightarrow^* \alpha_1 \ldots \alpha_n.$$

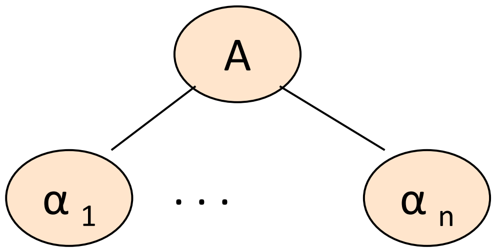


PART 2

- Given a derivation of a terminal string α, we need to prove the existence of a parse tree.
- The proof is an induction on the length of the derivation.

PART 2

- Basis
- If $A \Rightarrow^* \alpha_1 \ldots \alpha_n$ by a one-step derivation, then there must be a parse tree ($A \rightarrow \alpha_1 \ldots \alpha_n$)

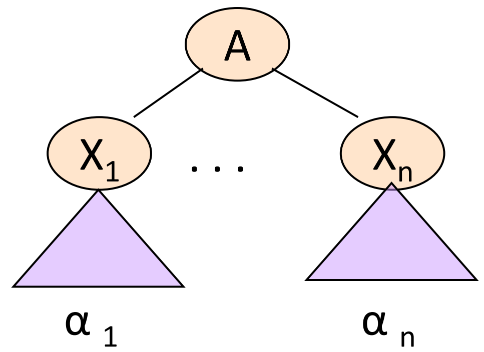


PART 2

- Induction
- Assume (2) for derivations of fewer than $k > 1$ steps, and let $A \Rightarrow^* \alpha$ be a k-step derivation.
- First step is $A \Rightarrow X_1 \ldots X_n$.
- Key point: w can be divided so the first portion is derived from X_1, the next is derived from X_2, and so on.
  - If X_i is a terminal, then $\alpha_i = X_i$.

PART 2

- That is, $X_i \Rightarrow^* \alpha_i$ for all i such that X_i is a variable.
  - And the derivation takes fewer than k steps.
- By the IH, if X_i is a variable, then there is a parse tree with root X_i and yield α_i.
- Thus, there is a parse tree



SUMMARY

- Discussion about context free grammar
- Language of CFG
- Derivations from a grammar for a string/word
- Parse tree for a string/word
- Ambiguous grammar

TEST YOUR KNOWLEDGE

- What the does the given CFG defines?
  S→aSbS|bSaS|e and w denotes terminal
  a) wwr
  b) wSw
  c) Equal number of a’s and b’s
  d) None of the mentioned
- A grammar G=(V, T, P, S) is ___________ if every production taken one of the two forms:
  B→aC
  B→a
  a) Ambiguous
  b) Regular
  c) Non Regular
  d) None of the mentioned

LEARNING OUTCOME

On successful completion of this topic, the student will be able to:

- Understand the need of formal languages, and grammars (K3)
  - Understand context free grammars

REFERENCE

- Hopcroft J.E., Motwani R. and Ullman J.D, “Introduction to Automata Theory, Languages and Computations”, Second Edition, Pearson Education, 2008