CONTEXT FREE GRAMMAR

COMP 4200 – Formal Language

OVERVIEW

- Grammar
  - Derivation from a grammar
  - Types
- Context-Free Grammars
  - Formal Definition
  - Examples of CFG
  - Designing CFG
  - Parse/derivation tree
  - Ambiguity
  - Simplification of CFG
  - Chomsky Normal Form
  - Greibach Normal form
- Pushdown Automata
  - Formal Definition
  - Examples of PDA
  - Equivalence with CFGs
- Non-Context Free Languages
  - The pumping lemma for context-free languages

GRAMMAR

- Regular grammar → set of rules that we use for proper composition.
- Grammar can be formally described using 4 tuples as G = {V,T,S,P}
  - V → set of variables/non terminal symbols
  - T → Terminal symbols
  - S → Start symbol
  - P → Production rule for terminal and non terminal symbols.
- Production rule, has the form a→β, where a and β are strings on V U T and atleast one symbol of 'a' belongs to V.
- Example: G = ({S,A,B}, {a,b}, S, {S → AB, A→a,B→b})

STEPS

- Write down the start variable. It is the variable on the left-hand side of the top rule, unless specified otherwise.
- Find a variable that is written down and a rule that starts with that variable. Replace the written down variable with the right-hand side of that rule.
- Repeat step 2 until no variables remain.

EXAMPLE

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Grammar G1 generates the string 000#111. The sequence of substitutions to obtain a string is called a derivation.
- A derivation of string 000#111 in grammar G1 is

EXAMPLE

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

$$B \rightarrow \#$$

- Grammar G1 generates the string 000#111. The sequence of substitutions to obtain a string is called a derivation.
- A derivation of string 000#111 in grammar G1 is
- $A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000B111 \Rightarrow 000\#111$.

G = ({S,A,B}, {A,B}, S, {S → AB, A→a,B→b})

- V = {S,A,B}
- T = {a,b}
- S = s
- P = S→AB, A→a, B→b
  - S → AB, we know that A→a
  - S → aB
  - S → ab

DERIVATION FROM A GRAMMAR

- The set of all the strings that can be derived from a grammar is said to be the Language generated from that grammar.
- Example: G = ({S,A}, {a,b}, S, {S → aAb, A→aaAbb,A→ε})
  - S → aAb, A → aaAbb
  - Expanding, S → aaaAbbb
  - S → aaaAbbb, A→ ε
  - S → aaabbb, String derived from grammar G
  - L = $a^n b^n$

DERIVATION FROM A GRAMMAR

- The set of all the strings that can be derived from a grammar is said to be the Language generated from that grammar.
- Example: $G = (\{S,A\}, \{a,b\}, S, \{S \rightarrow aAb, A \rightarrow aaAbb, A \rightarrow \varepsilon\})$

TYPES OF GRAMMAR

Grammar Type	Grammar Accepted	Language Accepted	Automaton
Type 0	Unrestricted grammar	Recursively enumerable language	Turing Machine
Type 1	Context-sensitive grammar	Context-sensitive language	Linear-bounded automaton
Type 2	Context-free grammar	Context-free language	Pushdown Automaton
Type 3	Regular grammar	Regular language	Finite state automaton

CONTEXT FREE GRAMMAR

- Methods of describing languages:
  - Finite automata
  - Regular expressions
- Some simple languages, such as $\{0^n1^n \mid n \geq 0\}$, cannot be described by these languages.
- We study Context-Free Grammars, a more powerful method of describing languages. Such grammars can describe certain features that have a recursive structure.
- The collection of languages associated with context-free grammars are called the Context-Free languages
- PDA: a class of machines recognizing the context-free languages A Context Free Grammar is a "machine" that creates a language. A language created by a CFG is called A Context Free Language

CONTEXT FREE LANGUAGE

Consider language $\{ 0^n 1^n \mid n \geq 0 \}$, which is nonregular.

- Start variable S with "substitution rules": $S \rightarrow 0S1$ $S \rightarrow \varepsilon$
- Rules can yield string $0^k 1^k$ by applying rule "$S \rightarrow 0S1$" k times, followed by rule "$S \rightarrow \varepsilon$" once.
- Derivation of string $0^3 1^3$ $S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 000S111 \Rightarrow 000\varepsilon111 = 000111$

EXAMPLE

- Example of CFG Example: Language $\{ 0^n 1^n \mid n \geq 0 \}$ has CFG $G = (V, \Sigma, R, S)$
- Variables $V = \{S\}$
- Terminals $\Sigma = \{0, 1\}$
- Start variable S
- Rules R: $S \rightarrow 0S1$ $S \rightarrow \varepsilon$
- Combine rules with same left-hand side in Backus-Naur (or Backus Normal) Form (BNF):

$$S \rightarrow 0S1 \mid \varepsilon$$

The following is an example of CFG, which we call G1.

$$A \rightarrow 0A1$$
$$A \rightarrow B$$
$$B \rightarrow \#$$

- A grammar consists of a collection of rules. Each rule appears as a line in the grammar, comprising a symbol and a string separated by an arrow.
- The symbol is called a variable.
- The string consists of variables and other symbols called terminals.
- One variable is designated as the start variable.
- It usually occurs on the LHS of the topmost rule.
- For example, grammar G1 contains 3 rules. G1's variables are A and B, where A is the start variable. Its terminals are 0, 1, and #.

$A \rightarrow 0A1$

$A \rightarrow B$

$B \rightarrow \#$

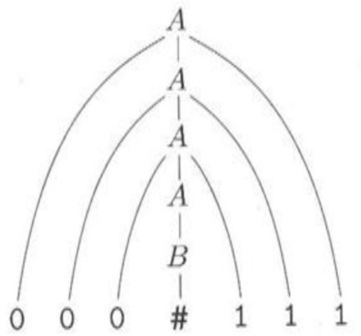
$A \rightarrow 0A1 \mid B$

For example, grammar G1 generates the string 000#111. The sequence of substitutions to obtain a string is called a derivation.

A derivation of string 000#111 in grammar G1 is

$A \rightarrow 0\ A1 \rightarrow 00\ A11 \rightarrow 000\ A111 \rightarrow 000\ B111 \rightarrow 000\ \#\ 111.$

Language defined by grammar G1 $\{0^n\#1^n \mid n \geq 0\}$



Parse tree for 000#111 in grammar G1

EXAMPLE

- Consider grammar G3 = ({S}, {a, b}, R, S). The set of rules, R, is
- $S \rightarrow aSb \mid SS \mid \varepsilon$
- What are some of the strings generated by this grammar?
- Can you verify if the strings abab, aaabbb, and aababb, belong to the grammar.

EXAMPLE

- CFG $G = (V, \Sigma, R, S)$ with

1. $V = \{S\}$
2. $\Sigma = \{0, 1\}$
3. Rules R: $S \to 0S \mid \varepsilon$

Then $L(G) = \{ 0^n \mid n \geq 0 \}$.

For example, S derives 0^3

$S \Rightarrow 0S \Rightarrow 00S \Rightarrow 000S \Rightarrow 000\varepsilon = 000$

Note that $\to$ and $\Rightarrow$ are different.

$\to$ used in defining rules

$\Rightarrow$ used in derivation

EXAMPLE

- CFG $G = (V, \Sigma, R, S)$ with

1. $V = \{S\}$
2. $\Sigma = \{0, 1\}$
3. Rules R: $S \rightarrow 0S \mid 1S \mid \varepsilon$

Then $L(G)=\Sigma^*$.

For example, S derives 0100

$S \Rightarrow 0S \Rightarrow 01S \Rightarrow 010S \Rightarrow 0100S \Rightarrow 0100$

EXAMPLE

Consider grammar G4 = (V, Σ ,R, <EXPR>)

V is {<EXPR> , <TERM> , <FACTOR>} and Σ is {a, +, x, (,)}.

The rules are

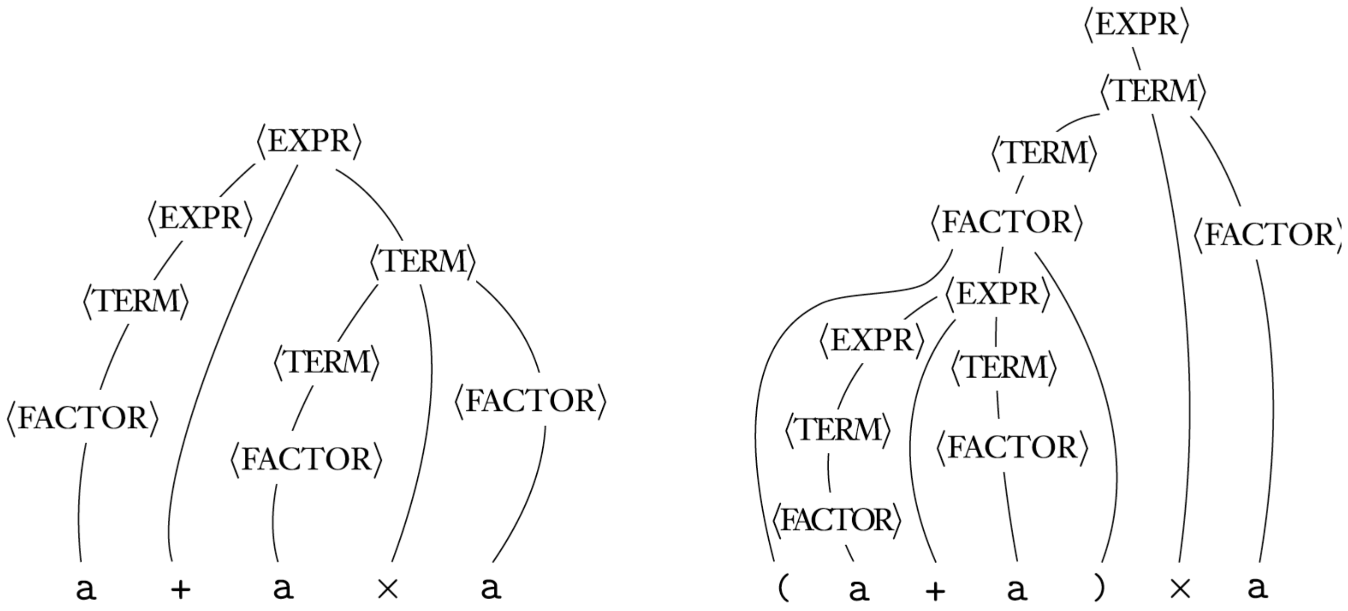
<EXPR> → <EXPR > + <TERM> | <TERM>

<TERM> → <TERM> * <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR >) | a

The two strings a + a * a and (a + a) * a can be generated with grammar G4.

PARSE TREE


⟨EXPR⟩
⟨EXPR⟩
⟨TERM⟩
⟨TERM⟩
⟨FACTOR⟩
⟨TERM⟩
⟨FACTOR⟩
⟨FACTOR⟩
a + a × a
⟨EXPR⟩
⟨TERM⟩
⟨TERM⟩
⟨FACTOR⟩
⟨FACTOR⟩
⟨EXPR⟩
⟨EXPR⟩
⟨TERM⟩
⟨TERM⟩
⟨FACTOR⟩
⟨FACTOR⟩
(a + a) × a

Consider grammar G4 = (V, Σ ,R, <EXPR>)

V is {<EXPR> , <TERM> , <FACTOR>} and Σ is {a, +, x, (,)}.

The rules are

<EXPR> → <EXPR > + <TERM> | <TERM>

<TERM> → <TERM> * <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR >) | a

<EXPR> → <EXPR> + <TERM>

→ <TERM> + <TERM>

→ <FACTOR> + <TERM>

→ a + <TERM>

→ a + <FACTOR>

→ a + (<EXPR>)

→ a + (<TERM>)

→ a + (<TERM> * <FACTOR>)

→ a + (<FACTOR> * <FACTOR>)

→ a + (a * <FACTOR>) = a + (a * a)

TRY YOURSELF!

- CFG $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{0, 1\}$, Rules R: $S \rightarrow 0S \mid 1S \mid 1$
- CFG $G = (V, \Sigma, R, S)$ with $V = \{S, Z\}$, $\Sigma = \{0, 1\}$, Rules R: $S \rightarrow 0S1 \mid Z$ $Z \rightarrow 0Z \mid \varepsilon$
- PALINDROME = $\{ w \in \Sigma* \mid w = wR \}$, where $\Sigma = \{a, b\}$. CFG $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{a, b\}$, Rules R: $S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$

TRY YOURSELF!

- CFG $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{0, 1\}$, Rules R: $S \rightarrow 0S \mid 1S \mid 1$. Derive 011
- CFG $G = (V, \Sigma, R, S)$ with $V = \{S, Z\}$, $\Sigma = \{0, 1\}$, Rules R: $S \rightarrow 0S1 \mid Z$

$Z \rightarrow 0Z \mid \varepsilon$. $L(G) = \{0^i 1^j \mid i>=j\}$

- PALINDROME = $\{ w \in \Sigma* \mid w = wR \}$, where $\Sigma = \{a, b\}$. CFG $G = (V, \Sigma, R, S)$ with $V = \{S\}$, $\Sigma = \{a, b\}$, Rules R: $S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$
- $A \rightarrow 0A$, $A \rightarrow 1B \mid 1$, $B \rightarrow 0A \mid 1B \mid 1$, Derive the string 10011011
- $S \rightarrow 0S1 \mid 1S0 \mid SS \mid \varepsilon$, derive the string 00011011.

CFG FOR SIMPLE ARITHMETIC EXPRESSIONS

- CFG $G = (V, \Sigma, R, S)$ with

1. $V = \{S\}$
2. $\Sigma = \{ +, -, \times, /, (,), 0, 1, 2, \ldots, 9 \}$
3. Rules R: $S \rightarrow S + S \mid S - S \mid S \times S \mid S/S \mid (S) \mid -S \mid 0 \mid 1 \mid \cdots \mid 9$

$L(G)$ is a set of valid arithmetic expressions over single-digit integers.

S derives string $2 \times (3 + 4)$

CFG for Simple Arithmetic Expressions

- CFG $G = (V, \Sigma, R, S)$ with

1. $V = \{S\}$
2. $\Sigma = \{ +, -, \times, /, (,), 0, 1, 2, \ldots, 9 \}$
3. Rules R: $S \rightarrow S + S \mid S - S \mid S \times S \mid S/S \mid (S) \mid -S \mid 0 \mid 1 \mid \cdots \mid 9$

L(G) is a set of valid arithmetic expressions over single-digit integers.

S derives string $2 \times (3 + 4)$

$S \Rightarrow S \times S \Rightarrow S \times (S) \Rightarrow S \times (S + S) \Rightarrow 2 \times (S + S) \Rightarrow 2 \times (3 + S) \Rightarrow 2 \times (3 + 4)$

DERIVATION TREE

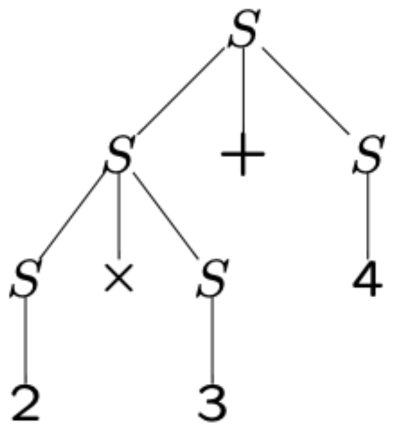
- CFG

$S \rightarrow S + S \mid S - S \mid S \times S \mid S/S \mid (S) \mid -S \mid 0 \mid 1 \mid \cdots \mid 9$

- Can generate string $2 \times 3+4$ using derivation

$S \Rightarrow S + S \Rightarrow S \times S + S \Rightarrow 2 \times S + S \Rightarrow 2 \times 3 + S \Rightarrow 2 \times 3+4$

- Leftmost derivation: leftmost variable replaced in each step.
- Corresponding derivation (or parse) tree



DESIGNING CFG

For example, to get a grammar for the language $\{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$, first construct the grammar

$S_1 \rightarrow 0 S_1 1 \mid \varepsilon$

for the language $\{0^n 1^n \mid n \geq 0\}$ and the grammar

$S_2 \rightarrow 1 S_2 0 \mid \varepsilon$

for the language $\{1^n 0^n \mid n \geq 0\}$ and then add the rule

$S \rightarrow S_1 \mid S_2$ to give the grammar

$S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow 0 S_1 1 \mid \varepsilon$

$S_2 \rightarrow 1 S_2 0 \mid \varepsilon$

EXAMPLE

- Generate CFG for given language $L=\{a^n b^n / n \geq 0\}$
- Let $L=\{a^n b^{2n} / n \geq 1\}$.Find CFG
- $L=\{ a^n b a^n / n \geq 1\}$. Find CFG.
- $L=\{wcw^r$ /w is element of $(a,b)^*\}$. Find the CFG.

CONTEXT FREE LANGUAGE (CFL)

- The CFG is given as $S \rightarrow aSb/ab$. Find the CFL.

CONTEXT FREE LANGUAGE (CFL)

- The CFG is given as $S \rightarrow aSb/ab$. Find the CFL.
- Let $G=(\{S\},\{a,b\},\{S \rightarrow aSb, S \rightarrow ab\},\{S\})$
- $S \rightarrow ab$
- $S \rightarrow aSb$
  - $S \rightarrow aabb$
- $S \rightarrow aSb$
  - $S \rightarrow aaSbb$
  - $S \rightarrow aaabbb$
- $L(G) = \{ab,aabb,aaabbb,\ldots.\}$ Therefore,$L(G)=\{a^nb^n/n\geq 1\}$

EXAMPLE

- Find the CFL for $S \rightarrow a\,B / bA$, $A \rightarrow a / aS / bAA$, $B \rightarrow b / bS / aBB$

TRY YOURSELF!

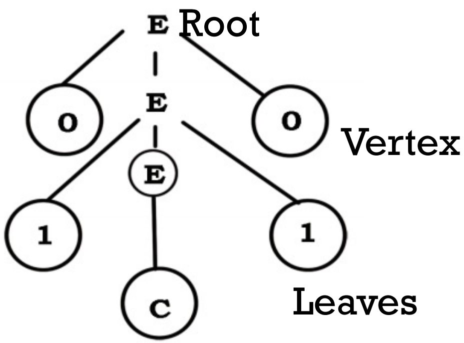
- Find L(G) $S \rightarrow aSa/bSb/\varepsilon$
- Find L(G) $S \rightarrow aSa/bSa/\varepsilon$
- Find L(G) $S \rightarrow aSa/bSb/a/b$
- Find L(G) $S \rightarrow SS/bS/a$
- Find L(G) for $S \rightarrow aS/bS/a$
- Find L(G) $S \rightarrow aSa/bSb/\varepsilon$

TRY YOURSELF!

- Consider the grammar G=({A,S},{a,b},P,S) where p consist of $S \rightarrow aAS/a$ $A \rightarrow SbA/SS/ba$ Draw the derivation for string"aabbaa"
- Draw the derivation for the string abb where $S \rightarrow aAB$ $A \rightarrow bBb$ $B \rightarrow A/\varepsilon$

PARSE OR DERIVATION TREE

- Parse tree is an ordered rooted tree that graphically presents the semantic information of string derived from CFG.
- Any one production or derivation derived from tree format.
- It is the graph form of representation.
- RULE
  - Start with 'S'.→ Root
  - The final answer should be terminal. →Leaves
  - The derivation should be applied from left to right.
  - The intermediate derivation should be terminal or nonterminal. → vertex

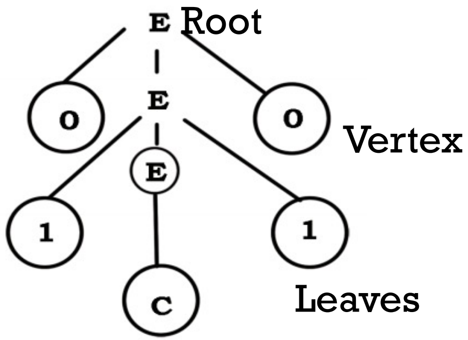


- W = 01C10 , E → 0E0 E →1E1 E → C
- DERIVATION E => 0E0

  => 01E10 ∴ E => 1E1

  => 01C10 ∴ E => C

  => E => 01C10
- PARSE TREE (OR) DERIVATION TREE



TYPES OF DERIVATION

- LEFT MOST DERIVATION (LMD): If at each step-in derivation, a production is applied to the left most variable (or) left most non-terminal then the derivation method is called left most derivation.
- Example w = id+id*id E → E+E , E → E*E, E → id

E → E+E

E → id+E ∴ E=>id

E → id+E*E ∴ E=>E*E

E → id+id*E ∴ E=>id

E → id+id* id

- RIGHT MOST DERIVATION (RMD): A derivation in which the right most variable is replaced at each step then, the derivation method is called right most derivation.
- Example

$E \rightarrow E+E$

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

$E \rightarrow E+E*id \therefore E \Rightarrow id$

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

EXAMPLE

- Draw derivation for the string abbabba For thee CFG given G where production is $S \rightarrow bA/aB$ $A \rightarrow a/aS/Baa$ $B \rightarrow b/bS/Abb$
  - s → aB
  - s → aBB
  - s → abSB
  - s → abSbs
  - s → abbAbs
  - s → abbAbbA
  - s → abbabbA
  - s → abbabba

EXAMPLE

Draw derivation tree for the string abaaba for the CFG given by G where p is S → aSa/bSb/b/a/ε

Solution:

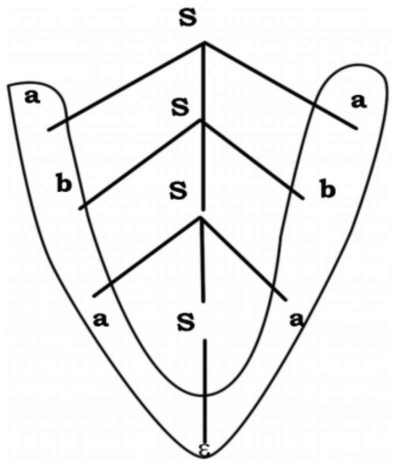
S → aSa

S → abSba

S → abaSaba

S → abaεaba

S → abaaba



TRY YOURSELF!

- Draw derivation tree for the string aabbaabba For thee CFG given G where production is $S \rightarrow bA/aB$ $A \rightarrow a/aS/bAA$ $B \rightarrow b/bS/aBB$
- Draw the derivation tree for the string abb where $S \rightarrow aAB$ $A \rightarrow bBb$ $B \rightarrow A/\varepsilon$.
- Draw the derivation tree for the given graph CFG G=(v,t,p,s) where p={$S \rightarrow aSb/ \varepsilon$} and the input string :aabb

EXAMPLE

- Draw derivation for the string abbabba For thee CFG given G where production is $S \rightarrow bA/aB$ $A \rightarrow a/aS/Baa$ $B \rightarrow b/bS/Abb/BB$
  - $s \rightarrow aB$
  - $s \rightarrow aBB$
  - $s \rightarrow abSB$
  - $s \rightarrow abSbS$
  - $s \rightarrow abbAbS$
  - $s \rightarrow abbAbbA$
  - $s \rightarrow abbabbA$
  - $s \rightarrow abbabba$

TRY YOURSELF!

- Consider the grammar G=({A,S},{a,b},P,S) where p consist of $S \rightarrow aAS/a$ $A \rightarrow SbA/SS/ba$ Draw the derivation for string"aabbaa"
- Draw the derivation for the string abb where $S \rightarrow aAB$ $A \rightarrow bBb$ $B \rightarrow A/\varepsilon$

TYPES OF DERIVATION

- LEFT MOST DERIVATION (LMD): If at each step-in derivation, a production is applied to the left most variable (or) left most non-terminal then the derivation method is called left most derivation.
- Example w = id+id*id E → E+E , E → E*E, E → id

E → E+E

E → id+E ∴ E=>id

E → id+E*E ∴ E=>E*E

E → id+id*E ∴ E=>id

E → id+id* id

- RIGHT MOST DERIVATION (RMD): A derivation in which the right most variable is replaced at each step then, the derivation method is called right most derivation.
- Example

E → E+E

E → E+E*E ∴ E=>E*E

E → E+E*id ∴ E=>id

E → E+id*id => id+id*id

TRY YOURSELF!

- Draw derivation tree for the string aabbaabba For thee CFG given G where production is $S \rightarrow bA/aB$ $A \rightarrow a/aS/bAA$ $B \rightarrow b/bS/aBB$
- Draw the derivation tree for the string abb where $S \rightarrow aAB$ $A \rightarrow bBb$ $B \rightarrow A/\varepsilon$.
- Draw the derivation tree for the given graph CFG G=(v,t,p,s) where p={$S \rightarrow aSb/ \varepsilon$} and the input string :aabb

AMBIGUITY

- CFG G is ambiguous if string w ∈ L(G) having different parse trees (or equivalently, different leftmost derivations).
- A string is derived ambiguously in a CFG if it has two or more different leftmost derivations
- A grammar that produces more than one parse tree (or) derivation tree for some string then the grammar is said to be an ambiguous grammar.
- An ambiguous grammar produces more than one LMD (or) more than 1 RMD then, the given grammar is said to be an ambiguous grammar.
- Leftmost derivation : At every step in the derivation the leftmost variable is replaced
- A grammar is ambiguous if it generates *some* string ambiguously
- Some context free languages are inherently ambiguous, that is, every grammar for the language is ambiguous

- LMD:

E => id+E ∴ E => id

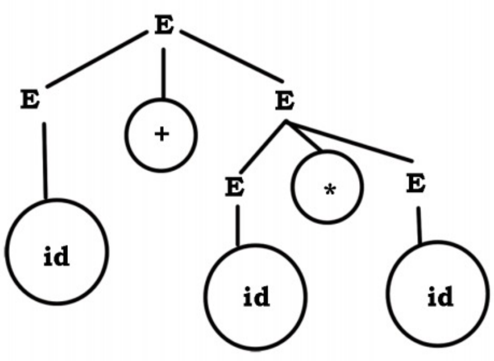
E => id+E*E

E => id+id*E

E => id+id*id

E=> id+id*id

PARSE TREE



Rules:
E → E+E ,E → E*E ,E → (E) ,E → id
w= id+id*id

- RMD:

E => E*E

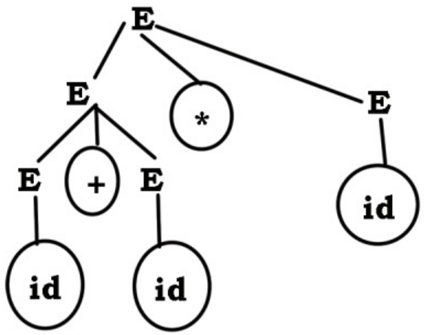
E => E+E*E

E => E+E*id

E => E+id*id

E => id+id*id

Therefore, the above grammar is ambiguous.

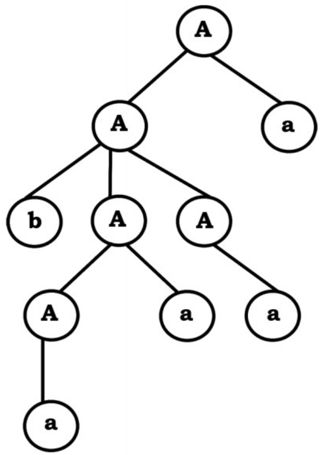


Show that CFG having productions $A \rightarrow a/Aa/bAA/AAb/AbA$ is ambiguous. The string can be baaaa

Show that CFG having productions $A \rightarrow a/Aa/bAA/AAb/AbA$ is ambiguous.

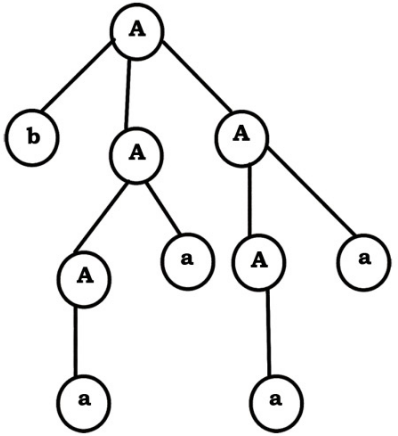
Parse tree 1 $A \rightarrow Aa \Rightarrow bAAa \Rightarrow bAaaa$

$A \rightarrow baaaa$



- Parse tree 2 $A \rightarrow bAA \Rightarrow bAaA \Rightarrow bAaAa$

  $A \rightarrow baaaa$



- Therefore, the above grammar is ambiguous.

TRY YOURSELF!

- Prove that given CFG is ambiguous, S →0B/1A A → 0/0S/1AA B → 1/1S/0BB. The string can be randomly chosen. For reference I have chosen "0011010"

- Show that grammar is S =>aSbS/ bSaS/ ε ambiguous.

RESOLVING AMBIGUITY

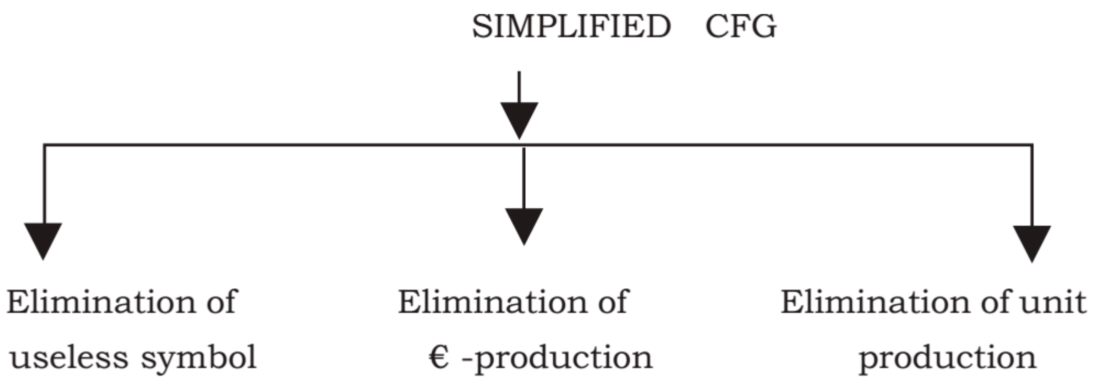
- Designing unambiguous grammars is tricky and requires planning from the start.
- It's hard to start with an ambiguous grammar and to manually convert it into an unambiguous one.
- Often, must throw the whole thing out and start over.
- We have just seen that this grammar is ambiguous:

  E → E+E|E*E|id

  If we take a string id+id*id or id+id+id we get two parse trees.
- Goals:
  - Eliminate the ambiguity from the grammar.
  - Make the only parse trees for the grammar the ones corresponding to operator precedence.

SIMPLIFYING GRAMMAR

SIMPLIFIED CONTEXT FREE GRAMMAR



1. Eliminate useless symbols i.e.,symbols or terminals which do not appear in any derivation of a terminal string from start symbol.
2. Eliminate ε-productions which are of the form $A \rightarrow \varepsilon$ form some variable A.
3. Eliminate unit production which are of the form $A \rightarrow B$ for variables A and B.

ELIMINATING USELESS PRODUCTIONS

- **Non-generating symbols** are those symbols which do not produce any terminal string.
- **Non-reachable symbols** are those symbols which cannot be reached at any time starting from the start symbol.
- Find the non-generating symbols, i.e., the symbols which do not generate any terminal string. If the start symbol is found non-generating, leave that symbol. For removing non-generating symbols, remove those productions whose right side and left side contain those symbols.
- Now find the non-reachable symbols, i.e., the symbols which cannot be reached, starting from the start symbol. Remove the non-reachable symbols.

- Remove the useless symbols from the given CFG.

$$S \rightarrow AC,$$

$$S \rightarrow BA,$$

$$C \rightarrow CB,$$

$$C \rightarrow AC,$$

$$A \rightarrow a,$$

$$B \rightarrow aC/b$$

- Those symbols which do not produce any terminal string are non-generating symbols. Here, C is a non-generating symbol. So, we have to remove the symbol C.
- Minimized grammar will be: S → BA, A → a, B → b
- Now, we have to find non-reachable symbols, the symbols which cannot be reached at any time starting from the start symbol. There is no non-reachable symbol in the grammar.
- The minimized grammar is, $S \rightarrow BA, A \rightarrow a, B \rightarrow b$

EXAMPLE

Remove the useless symbols from the given CFG.

1. S → aB/bX, A → Bad/bSX/a, B → aSB/bBX, X → SBD/aBx/ad
2. S → AB | CA, A→ a, B→BC|AB, C→aB|b
3. S→aA|a|Bb|cC, A→aB, B→a|Aa, C→cCD, D→ddd

TRY YOURSELF!

- $S \rightarrow aC|SB$, $A \rightarrow bSCa|ad$, $B \rightarrow aSB|bBC$, $C \rightarrow aBC|ad$
- $S \rightarrow aAa$, $A \rightarrow bBB$, $B \rightarrow ab$, $C \rightarrow aB$
- $S \rightarrow aS|A|C$, $A \rightarrow a$, $B \rightarrow aa$, $C \rightarrow aCb$

REMOVAL OF UNIT PRODUCTIONS

- Any production rule of the form $A \rightarrow B$, where A,B ∈ Non terminal is called Unit production.
- Steps:
  - To remove A → B, add production, A→X to the grammar rule whenever B → x occurs in grammar. X → Null
  - Delete A→B
  - Repeat step 1 until all the unit productions are removed.

REMOVAL OF UNIT PRODUCTIONS

- A unit production is a production which is of form $A \rightarrow B$, where both A and B are variables.
- Production in the form ***non-terminal*** $\rightarrow$ ***single non-terminal*** is called unit production.
- Let there be a grammar $S \rightarrow AB$, $A \rightarrow E$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow b$, $E \rightarrow a$
- From here, if we are going to generate a language, then it will be generated by the following way
  - $S \rightarrow AB \Rightarrow EB \Rightarrow aB \Rightarrow aC \Rightarrow aD \Rightarrow ab$ => 6 steps
- The grammar, by removing unit production and as well as minimizing, will be
  - $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$ => 3 steps

TRY YOURSELF!

- $S \rightarrow 0AB$ $A \rightarrow 01|B$ $B \rightarrow 0A|1$
- $S \rightarrow 0A|1B|C$ $A \rightarrow 0S|00$ $B \rightarrow 1|A$ $C \rightarrow 01$
- Remove the unit production from the following grammar. $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow b$
- Remove the unit production from the following grammar. $S \rightarrow aX/Yb/Y$, $X \rightarrow S$, $Y \rightarrow Yb/b$
- Remove the unit production from the following grammar. $S \rightarrow AA$, $A \rightarrow B/BB$, $B \rightarrow abB/b/bb$

REMOVAL OF NULL PRODUCTIONS

- In CFG, a nonterminal symbol A is a nullable variable if there is a production, A → ε or there is a derivation that starts at A and leads to ε.
- Steps:
  - To remove A → ε look for all productions whose right side contains A.
  - Replace each occurrences of A in each of these productions with ε.
  - Add the resultant production to the grammar.
- Example: S → ABAC, A→aA/ε, B→bB/ε, C→c
  - Null productions: A→ ε and B→ ε
- To eliminate A→ ε, look for production whose right side contains A.
  - S → ABAC, replace A→ ε
  - Possible outcomes, S→ABC/BAC/BC
  - A→aA, replace A→ ε , A→a
  - S→ABAC/ABC/BAC/BC, A→aA/a, B→bB/ε, C→c

REMOVAL OF NULL PRODUCTIONS

- To eliminate $B \rightarrow \varepsilon$, look for production whose right side contains B.
  - $S \rightarrow ABAC$, replace $B \rightarrow \varepsilon$
  - Possible outcomes, $S \rightarrow AAC/AC/C$
  - $B \rightarrow bB$, replace $B \rightarrow \varepsilon$, $B \rightarrow b$
  - $S \rightarrow ABAC/ABC/BAC/BC/AAC/AC/C$, $A \rightarrow aA/a$, $B \rightarrow bB/b$, $C \rightarrow c$

TRY YOURSELF!

- $S \rightarrow aSa \mid bSb \mid \varepsilon$
- $S \rightarrow aS \mid A,\ A \rightarrow aA \mid \varepsilon$
- $S \rightarrow a \mid Ab \mid aBa,\ A \rightarrow b \mid \varepsilon,\ B \rightarrow b \mid A$
- $S \rightarrow AB,\ A \rightarrow aAA \mid \varepsilon,\ B \rightarrow bBB \mid \varepsilon$