COM3064
Automata Theory

Week 6: Context Free Languages

Lecturer: Dr. Sevgi YİĞİT SERT
Spring 2023

Resources: Introduction to The Theory of Computation, M. Sipser,
Introduction to Automata Theory, Languages, and Computation, J.E. Hopcroft, R. Motwani, and J.D. Ullman
BBM401 Automata Theory and Formal Languages, İlyas Çiçekli

Grammars

- Grammars express languages
- Example: the English language

$$\langle sentence \rangle \rightarrow \langle noun\ \_\ phrase \rangle\ \langle predicate \rangle$$

$$\langle noun\ \_\ phrase \rangle \rightarrow \langle article \rangle\ \langle noun \rangle$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

A derivation of "the dog walks"

$$\langle sentence \rangle \rightarrow \langle noun\ \_\ phrase \rangle\ \langle predicate \rangle$$
$$\langle noun\ \_\ phrase \rangle \rightarrow \langle article \rangle\ \langle noun \rangle$$
$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

$$\langle article \rangle \rightarrow a$$
$$\langle article \rangle \rightarrow the$$

$$\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun\ _\ phrase \rangle\ \langle predicate \rangle \\
&\Rightarrow \langle noun\ _\ phrase \rangle\ \langle verb \rangle \\
&\Rightarrow \langle article \rangle\ \langle noun \rangle\ \langle verb \rangle \\
&\Rightarrow the\ \langle noun \rangle\ \langle verb \rangle \\
&\Rightarrow the\ dog\ \langle verb \rangle \\
&\Rightarrow the\ dog\ walks
\end{aligned}$$

$$\langle noun \rangle \rightarrow cat$$
$$\langle noun \rangle \rightarrow dog$$

$$\langle verb \rangle \rightarrow runs$$
$$\langle verb \rangle \rightarrow walks$$

Language of the grammar

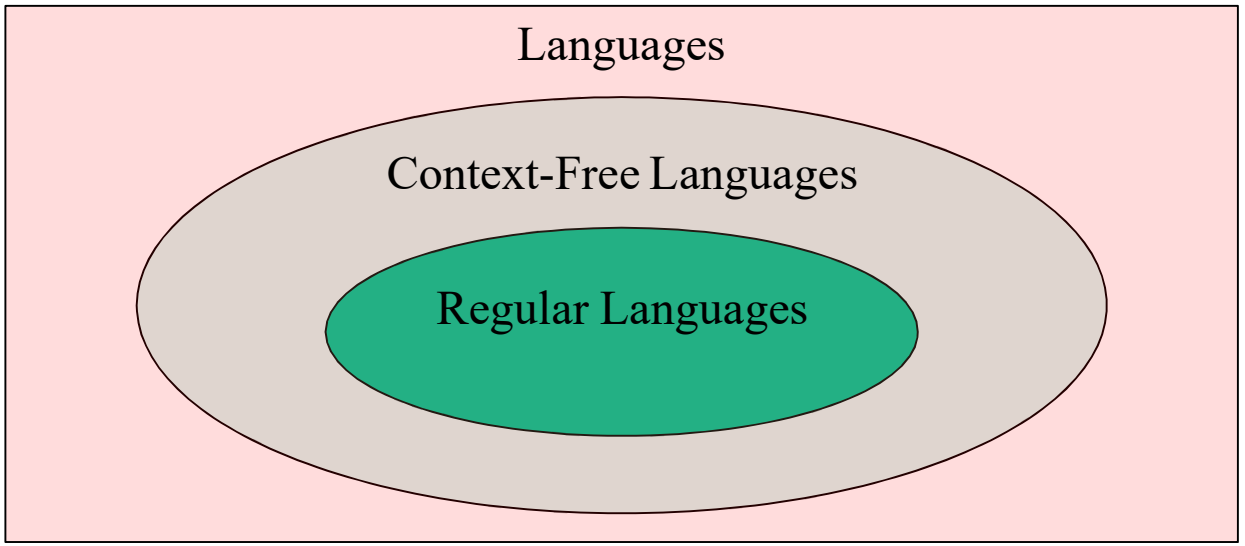
L = { “a cat runs”, “a cat walks”,
“the cat runs”, “the cat walks”,
“a dog runs”, “a dog walks”,
“the dog runs”, “the dog walks” }

Context-Free Grammars

- We have seen that many languages cannot be regular.
  - We need to consider larger classes of languages.
  - **Context-Free Languages (CFLs)** is a larger class of regular languages.
  - Every regular language is a context-free language.
- Regular expressions are used to define regular languages.
- **Context-Free Grammars (CFGs)** are used to define Context-Free Languages (CFLs).
- **Pushdown Automata** recognize CFLs.
- Context-Free Grammars (CFGs) played a central role in natural language processing, and compilers.
  - The syntax of a programming language is described by a **CFG**.
  - A parser for a programming language is a **pushdown automata**.

Context-Free Grammars and Context-Free Languages

- A **context-free grammar** is **a notation for describing languages**.
- CFGs are more powerful than REs, but they cannot still define all possible languages.
  - **CFGs** define **Context-Free Languages (CFLs).**
  - CFGs are useful to describe nested structures.
  - Since every regular language is a CFL, it can be defined by a CFG.
  - There are also languages that are not CFLs.



CFG – Example

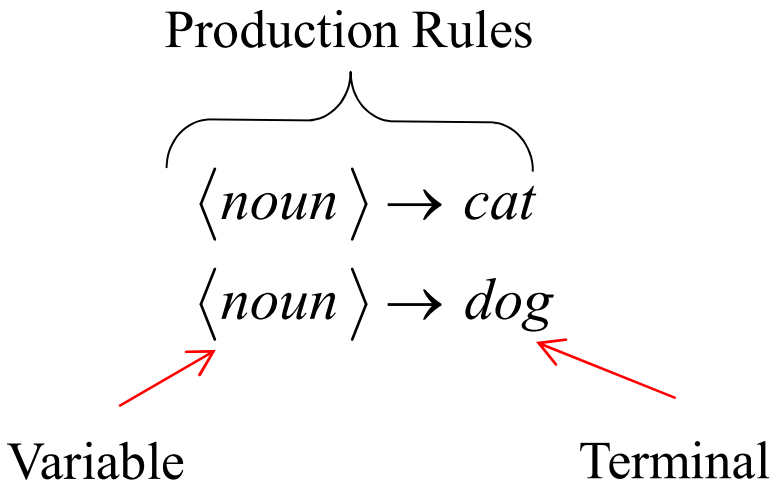
- The language $\{0^n 1^n \mid n \geq 0\}$ is not a regular language, but it is a CFL.
  - It can be defined by a CFG.

- A CFG for $\{0^n 1^n \mid n \geq 0\}$ is

$S \rightarrow \varepsilon$

$S \rightarrow 0S1$

- 0 and 1 are **terminals**.
- S is a **variable** (or **nonterminal**).
- S is also the **start symbol** of this CFG.
- $S \rightarrow \varepsilon$ and $S \rightarrow 0S1$ are **productions or rules**



Formal Definition of CFGs

- A **context-free grammar** $\boldsymbol{G}$ is a quadruple

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{T}, \boldsymbol{P}, \boldsymbol{S})$$

- $\boldsymbol{V}$ is a finite set of **variables** (**non-terminals**).
- $\boldsymbol{T}$ is a finite set of **terminals**.
- $\boldsymbol{P}$ is a finite set of **production rules** of the form $\boldsymbol{A} \rightarrow \alpha$, where $\boldsymbol{A}$ is a variable and $\alpha \in (V \cup T)^*$

  The left side of a production is a variable and its right side is a string of variables and terminals.

- $\boldsymbol{S}$ is a designated variable called the **start symbol**.

CFG - Example

- Consider the language of palindromes

  $L_{pal} = \{w \in \Sigma^* : w = w^R\}$

- Some members of L_{pal} : *abba bob ses tat*
- L_{pal} is NOT regular, but L_{pal} is a context-free language.

- Let $\Sigma = \{0,1\}$ be the alphabet for L_{pal}.
- In this case, ε, 0, 1, 00, 11, 000, 010, 101, 111, 0110,… will be in L_{pal}.

- A CFG G_{pal} for L_{pal} is:

  $\mathbf{G_{pal} = (\{S\}, \{0,1\}, \{ S \rightarrow \varepsilon, S \rightarrow 0, S \rightarrow 1, S \rightarrow 0S0, S \rightarrow 1S1 \}, S)}$

- Sometimes, we use a shorthand for a list of production rules with the same left side.

  $\mathbf{S \rightarrow \varepsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1}$

Derivation

- Initially, we have a string that only contains the **start symbol**.
- We expand the **start symbol** using a production rule whose left side is the start symbol.
  - i.e. we replace the start symbol with a string which appears on the right side of a production rule belongs to the start symbol.
- If the resulting string contains at least one variable, we further expand the resulting string by replacing one of its variables with the right side (body) of one of its production rules.
  - We can continue these replacements until we derive a string consisting entirely of terminals.
- ***The language of the grammar is the set of all strings of terminals that we can be obtained in this way.***
- **Replacement of a variable (in a string) with the right side of one of its productions is called as derivation.**

Derivation $\Rightarrow$

- Suppose $\boldsymbol{G = (V, T, P, S)}$ is a CFG.
- Let $\alpha \boldsymbol{A} \beta$ be a string of terminals and variables where $\boldsymbol{A}$ is a variable.
  - i.e. α and β are strings in $(V \cup T)^*$, and A is V.
- Let $\boldsymbol{A} \rightarrow \gamma$ be a production rule of G.
- Then, we say that

$$\alpha \boldsymbol{A} \beta \Rightarrow \alpha \gamma \beta$$ is a **derivation**

- One derivation step can replace any variable in the string with the right side (body) of one of its productions.

Derivation $\Rightarrow$

- Grammar G : $S \rightarrow Ab$

$A \rightarrow aAb$

$A \rightarrow \lambda$

- Derivations :

$S \Rightarrow Ab \Rightarrow b$

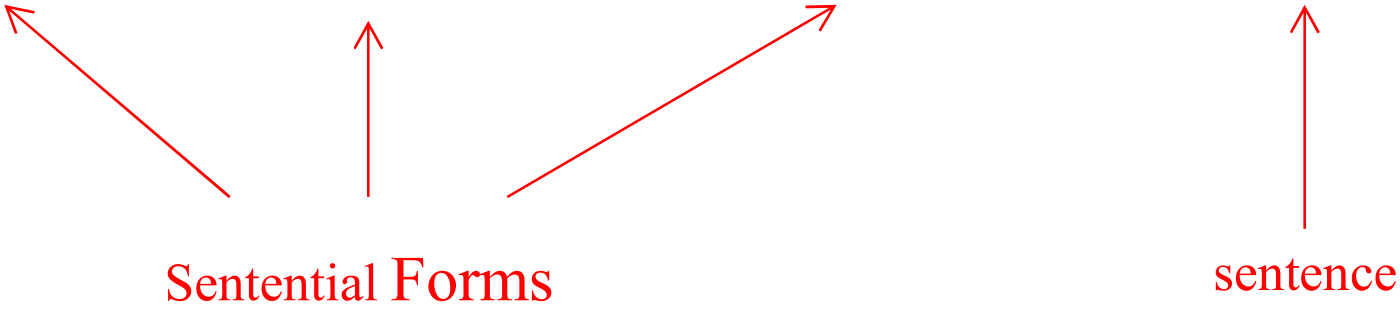
$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow abb$

$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aabbb$

Derivation $\Rightarrow$

- Sentential Form: A sentence that contains variables and terminals

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$



Derivation Sequence

- We can extend the derivation (⇒) relationship to represent zero or more derivation steps.
- We use symbol $\overset{*}{\Rightarrow}$ to denote zero or more steps of a **derivation sequence**.

- In other words, $\alpha \overset{*}{\Rightarrow} \beta$, β means that there is a sequence of strings $\gamma_1, \gamma_2, \ldots, \gamma_n$ for some $n \geq 1$ such that

  1. $\alpha = \gamma_1$,
  2. $\beta = \gamma_n$, and
  3. for $i = 1,2, \ldots, n-1$, we have $\gamma_i \Rightarrow \gamma_{i+1}$

Derivation Sequence – Example

- Let CFG **G = ({S}, {0,1}, { S→ε , S→0S1 }, S)**
- **S⇒ 0S1 ⇒ 00S11 ⇒ 000S111 ⇒ 000111** is a **derivation sequence**.
  - **S derives 000111; or 000111 is derived from S.**

- That is, **S** $\overset{*}{\Rightarrow}$**00011** and also
- **S** $\overset{*}{\Rightarrow}$ **000S111**
- **S** $\overset{*}{\Rightarrow}$ **00S11**
- **0S1** $\overset{*}{\Rightarrow}$ **000S111**
- **00S11** $\overset{*}{\Rightarrow}$**000111**

- **S ⇒ 0S1 ⇒ 00S11 ⇒ 0011** is a **derivation sequence**.

Derivation Sequence – Example

A CFG:

S → **ASB** | **c**

A → ε | **aA**

B → ε | **bB**

Derivation Sequences of **acb** from **S.**

S ⇒ ASB ⇒ aASB ⇒ aSB ⇒ acB ⇒ acbB ⇒ acb

S ⇒ ASB ⇒ ASbB ⇒ Asb ⇒ Acb ⇒ aAcb ⇒ acb

S ⇒ ASB ⇒ AcB ⇒ aAcB ⇒ aAcbB ⇒ acbB ⇒ acb

- We may select any non-terminal (variable) of the string for the replacement in each derivation step.

Leftmost and Rightmost Derivations

- **Leftmost Derivation** $\underset{lm}{\Rightarrow}$ always replaces the **leftmost variable** (in the string) with one of its rule-bodies

$$S \underset{lm}{\Rightarrow} ASB \underset{lm}{\Rightarrow} aASB \underset{lm}{\Rightarrow} aSB \underset{lm}{\Rightarrow} acB \underset{lm}{\Rightarrow} acbB \underset{lm}{\Rightarrow} acb$$

- **Rightmost Derivation** $\underset{rm}{\Rightarrow}$ always replaces the **righmost variable** (in the string) by one of its rule-bodies.

$$S \underset{rm}{\Rightarrow} ASB \underset{rm}{\Rightarrow} ASbB \underset{rm}{\Rightarrow} ASb \underset{rm}{\Rightarrow} Acb \underset{rm}{\Rightarrow} aAcb \underset{rm}{\Rightarrow} acb$$

Leftmost and Rightmost Derivations

S → ASB | c

A → ε | aA

B → ε | bB

Derivation Sequences of **acb** from **S.**

S ⇒ ASB ⇒ aASB ⇒ aSB ⇒ acB ⇒ acbB ⇒ acb is a **leftmost derivation**

S ⇒ ASB ⇒ aASB ⇒ aSB ⇒ AcB ⇒ aAcB ⇒ acb is a **rightmost derivation**

S ⇒ ASB ⇒ AcB ⇒ aAcB ⇒ aAcbB ⇒ acbB ⇒ acb

is **NOT** a leftmost or rightmost derivation.

The Language of a CFG

- If $G = (V, T, P, S)$ is a CFG, then **the language of G** is

$$L(G) = \{ w \in T^* : S \overset{*}{\Rightarrow} w \}$$

- i.e. **the set of strings of terminals (strings over T^*) that are derivable from S**

- If we call L(G) as a **context-free language**.
  - Ex: $L(G_{pal})$ is a context-free language.

- **For each CFL, there is a CFG, and each CFG generates a CFL.**
- Every regular language is a CFL.
  - That is, regular languages are a proper subset of context-free languages

The Language of a CFG - Example

A context-free grammar G :

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

A derivation :

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

The Language of a CFG - Example

A context-free grammar G : $S \to aSa$

$S \to bSb$

$S \to \lambda$

A derivation :

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

Another derivation :

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaba$$

The Language of a CFG - Example

A context-free grammar G : $S \to aSa$

$S \to bSb$

$S \to \lambda$

A derivation :

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$$

Another derivation :

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaba$$

$$L(G) = \{ww^R : \ w \in \{a,b\}^*\}$$

Parse Trees

- **Parse trees** are an alternative representation to derivations.

- If $v \in L(G)$, for some CFG, then v has a parse tree, which tells us the (syntactic) structure of w.
  - If G is unambiguous, v can have only one parse tree.
  - If G is ambiguous, v may have more than one parse tree.
  - Ideally there should be only one parse tree for each string in the language. This means that the grammar should be unambiguous.
    - We may remove the ambiguity from some of ambiguous grammars in order to obtain unambiguous grammars by making certain assumptions.
    - Unfortunately, some CFLs are inherently ambiguous and they can be only defined by ambiguous grammars.

Constructing Parse Trees

- Let $G = (V, T, P, S)$ be a CFG.

- **A tree is a parse tree for G if:**

  1. **Each interior node is labeled by a variable in V.**
     - **The root must be labeled by the start symbol S.**
  2. **Each leaf is labeled by a symbol in $T \cup \{\varepsilon\}$.**
     - **Any ε -labeled leaf is the only child of its parent.**
  3. **If an interior node is labeled by the variable A, and its children (from left to right) labeled $X_1, X_2, \ldots, X_k$ then $A \rightarrow X_1 X_2 \ldots X_k \in P$.**

Parse Tree - Example

Grammar:

$S \rightarrow ASB \mid c$

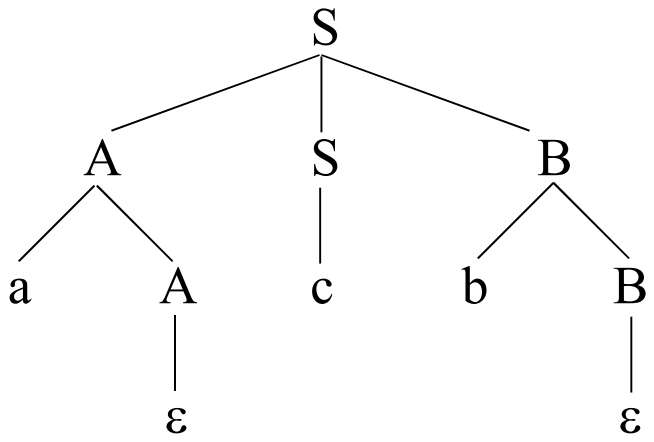
$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

A Derivation Sequence of **acb**

$S \Rightarrow ASB \Rightarrow aASB \Rightarrow aSB \Rightarrow acB \Rightarrow acbB \Rightarrow acb$

Parse tree of **acb**



Parse Tree & Derivation

Grammar:

$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

S

S

Parse Tree & Derivation

Grammar:

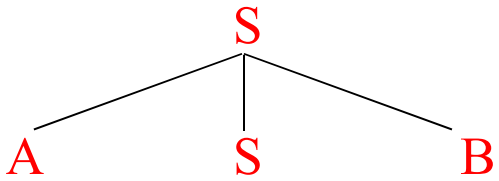
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB$



Parse Tree & Derivation

Grammar:

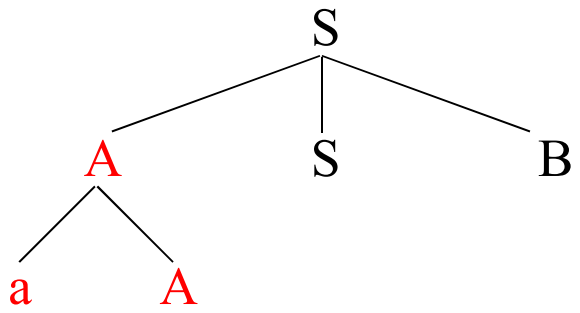
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB \Rightarrow aASB$



Parse Tree & Derivation

Grammar:

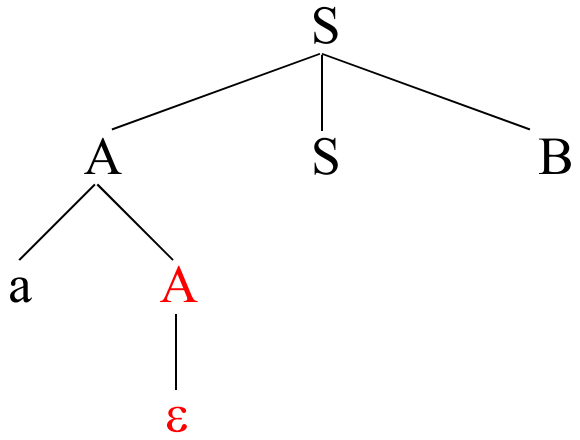
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB \Rightarrow aASB \Rightarrow aSB$



Parse Tree & Derivation

Grammar:

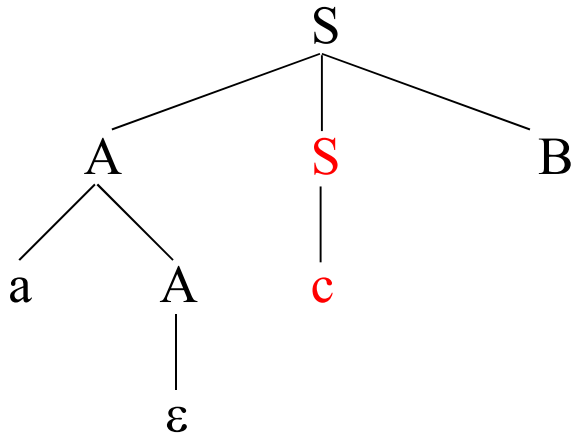
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB \Rightarrow aASB \Rightarrow aSB \Rightarrow acB$



Parse Tree & Derivation

Grammar:

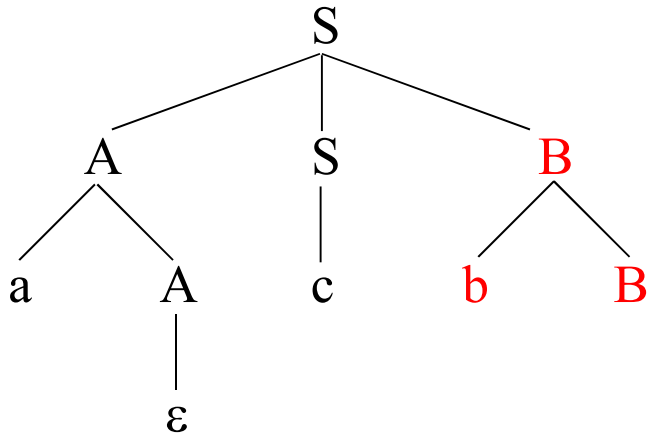
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB \Rightarrow aASB \Rightarrow aSB \Rightarrow acB \Rightarrow acbB$



Parse Tree & Derivation

Grammar:

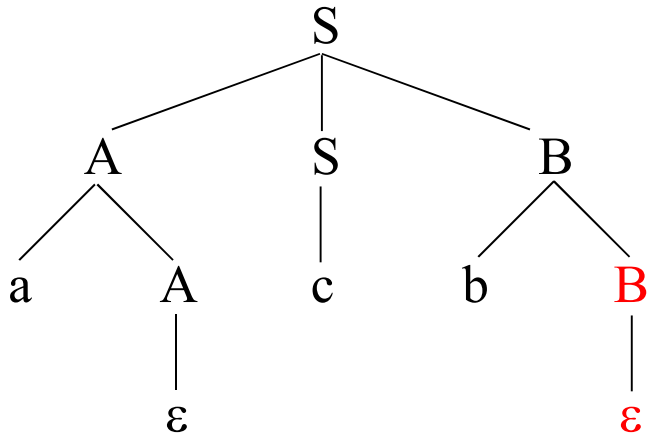
$S \rightarrow ASB \mid c$

$A \rightarrow \varepsilon \mid aA$

$B \rightarrow \varepsilon \mid bB$

- Each derivation step corresponds to the creation of an inner node (by creating its children).

$S \Rightarrow ASB \Rightarrow aASB \Rightarrow aSB \Rightarrow acB \Rightarrow acbB \Rightarrow acb$



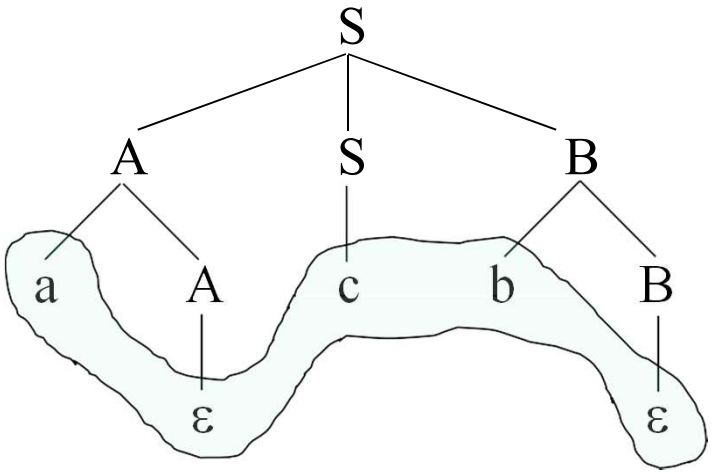
The Yield of a Parse Tree

- **The concatenation of the labels of the leaves in left-to-right order is called the yield of the parse tree.**

- The yield of the parse tree is a string of terminals.
  - The set of all yields of all parse trees of a CFG G is the language of G.

- Yield Example:

  a ε c b ε = acb



For every parse tree, there is a unique leftmost, and a unique rightmost derivation.

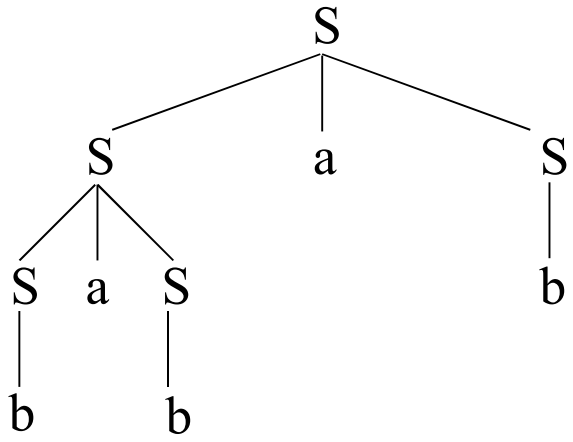
Ambiguous Grammars

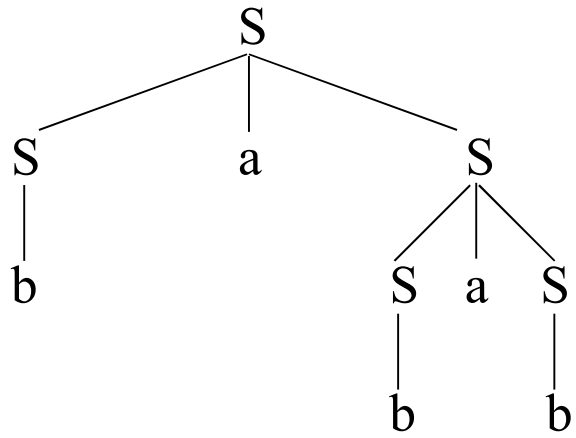
- **A CFG is ambiguous if it produces more than one parse tree for a string in the language.**
  - **i.e. there is a string in the language that is the yield of two or more parse trees.**

Example:

$S \rightarrow SaS \mid b$ is an ambiguous grammar.

There are two parse trees for the string **babab**





Ambiguity, Leftmost and Rightmost Derivations

- If there are two different parse trees for a string in the language, they must produce two different **leftmost derivations** for that string.
  - Conversely, two different leftmost derivations of a string produce two different parse trees for that string.
- Likewise for rightmost derivations.

- Thus, equivalent definitions of **ambiguous grammar** are:

1. **A CFG is ambiguous if there is a string in the language that has two different leftmost derivations.**
2. **A CFG is ambiguous if there is a string in the language that has two different rightmost derivations.**

Ambiguity, Leftmost and Rightmost Derivations

$S \rightarrow SaS \mid b$

- There are two **leftmost** derivation sequences for the string **babab**

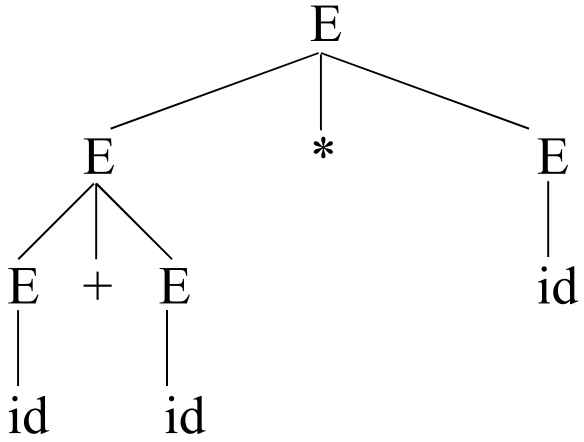
1. $\mathbf{S \underset{lm}{\Rightarrow} SaS \underset{lm}{\Rightarrow} SaSaS \underset{lm}{\Rightarrow} baSaS \underset{lm}{\Rightarrow} babaS \underset{lm}{\Rightarrow} babab}$
2. $\mathbf{S \underset{lm}{\Rightarrow} SaS \underset{lm}{\Rightarrow} baS \underset{lm}{\Rightarrow} baSaS \underset{lm}{\Rightarrow} babaS \underset{lm}{\Rightarrow} babab}$

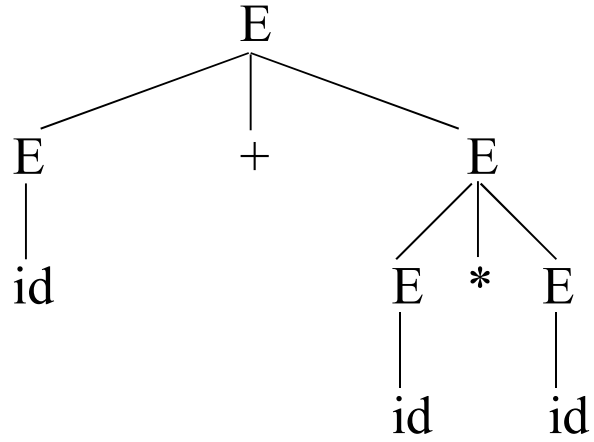
- There are two **rightmost** derivation sequences for the string **babab**

1. $\mathbf{S \underset{rm}{\Rightarrow} SaS \underset{rm}{\Rightarrow} Sab \underset{rm}{\Rightarrow} SaSab \underset{rm}{\Rightarrow} Sabab \underset{rm}{\Rightarrow} babab}$
2. $\mathbf{S \underset{rm}{\Rightarrow} SaS \underset{rm}{\Rightarrow} SaSaS \underset{rm}{\Rightarrow} SaSab \underset{rm}{\Rightarrow} Sabab \underset{rm}{\Rightarrow} babab}$

Ambiguity

- An ambiguous grammar for expressions: **E → E+E | E*E | E^E | id | (E)**
- 2 parse trees, 2 leftmost and 2 rightmost derivations for the expression **id+id*id**





$E \underset{lm}{\Rightarrow} E*E \underset{lm}{\Rightarrow} E+E*E \underset{lm}{\Rightarrow} id+E*E \underset{lm}{\Rightarrow} id+id*E \underset{lm}{\Rightarrow} id+id*id$

$E \underset{lm}{\Rightarrow} E+E \underset{lm}{\Rightarrow} id+E \underset{lm}{\Rightarrow} id+E*E \underset{lm}{\Rightarrow} id+id*E \underset{lm}{\Rightarrow} id+id*id$

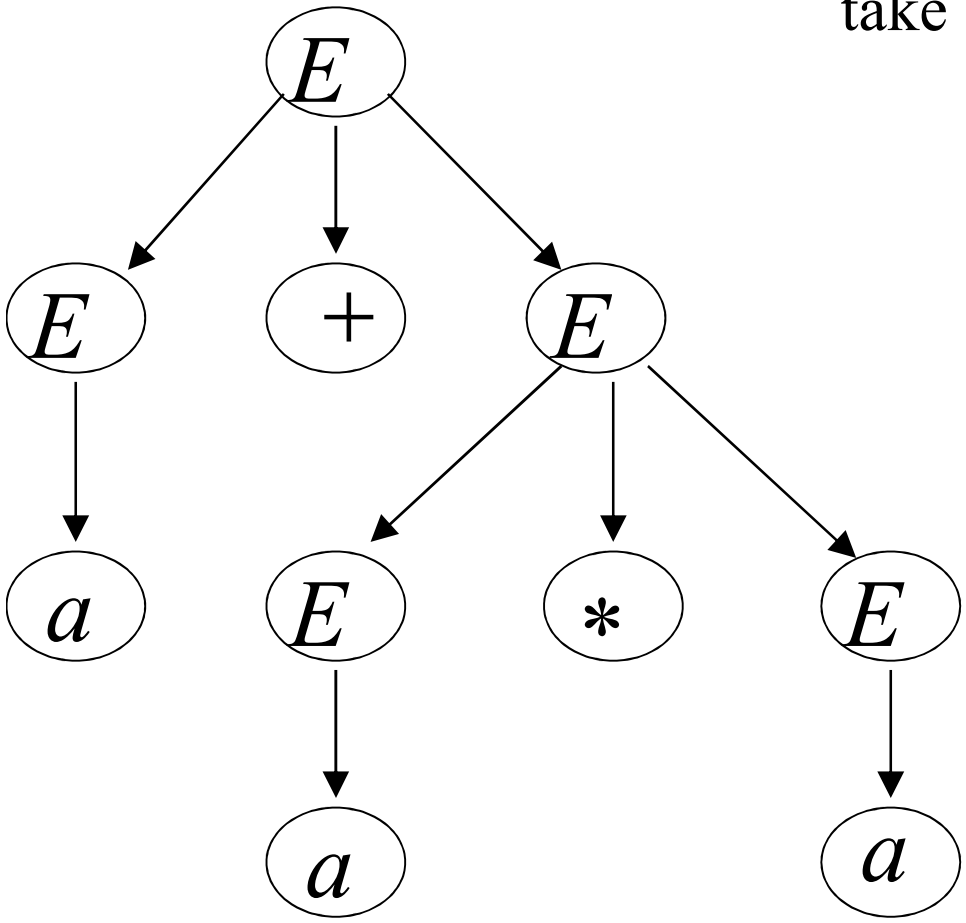
$E \underset{rm}{\Rightarrow} E*E \underset{rm}{\Rightarrow} E*id \underset{rm}{\Rightarrow} E+E*id \underset{rm}{\Rightarrow} E+id*id \underset{rm}{\Rightarrow} id+id*id$

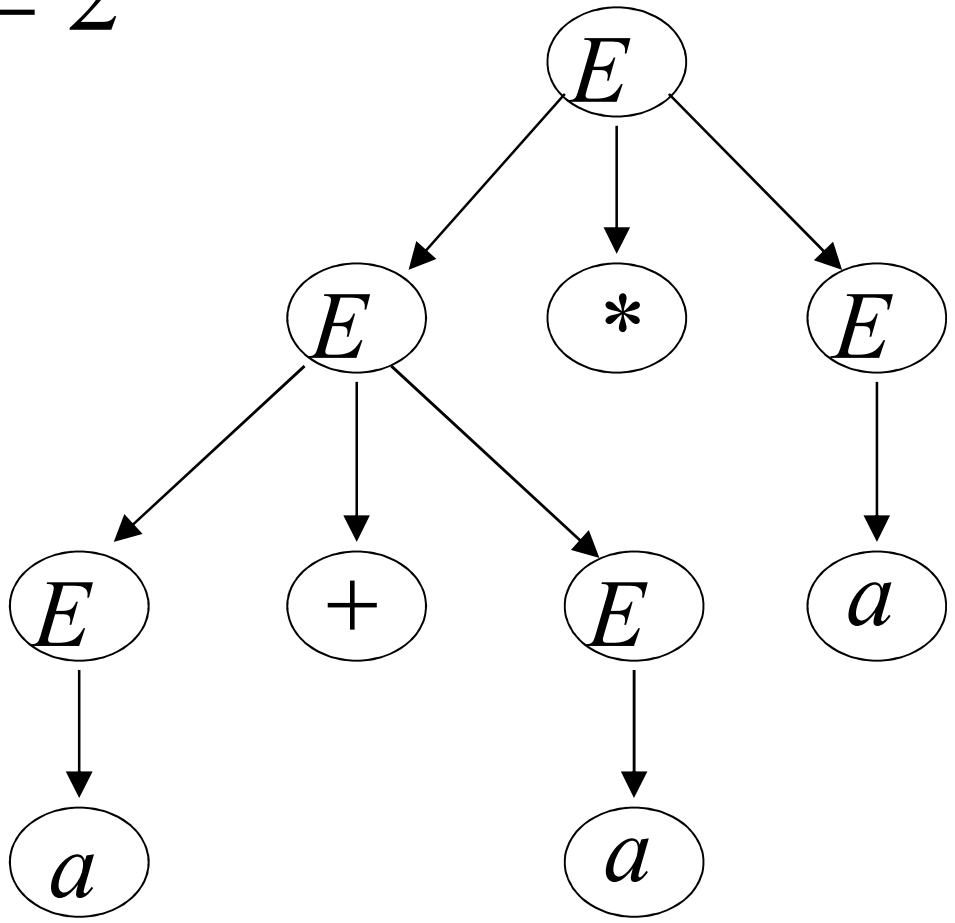
$E \underset{rm}{\Rightarrow} E+E \underset{rm}{\Rightarrow} E+E*E \underset{rm}{\Rightarrow} E+E*id \underset{rm}{\Rightarrow} E+id*id \underset{rm}{\Rightarrow} id+id*id$

Why do we care about ambiguity?

$$a + a * a$$

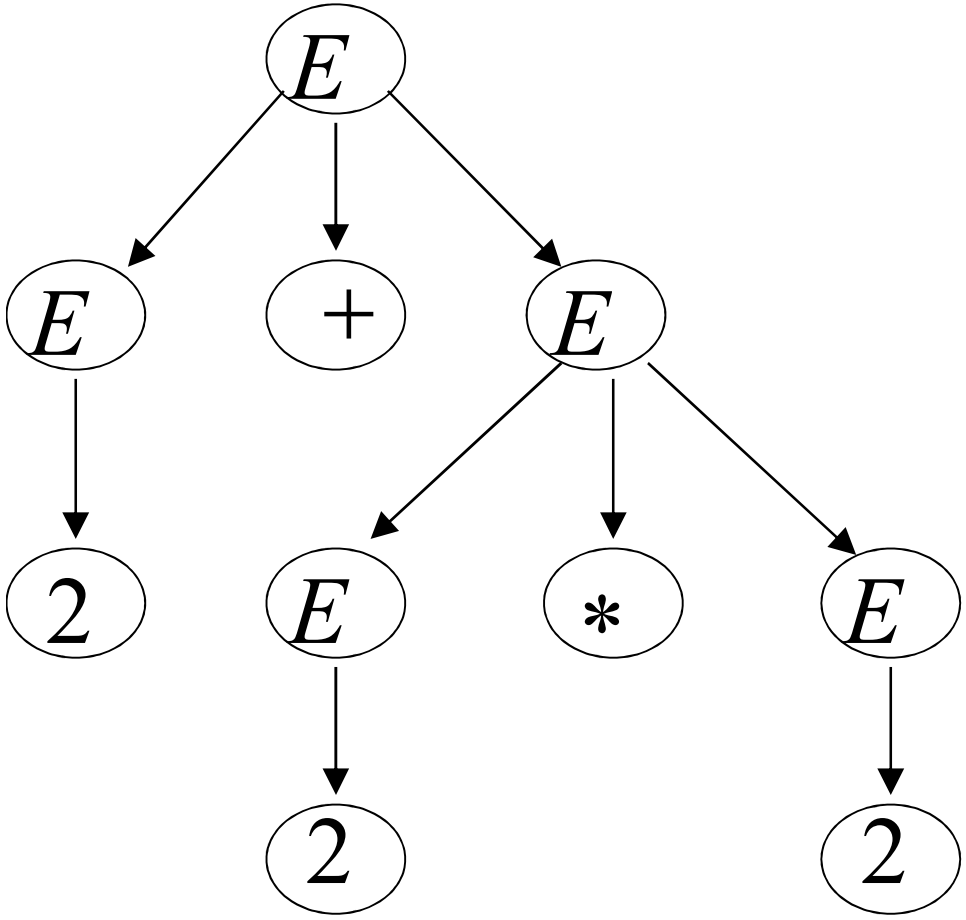
take $a = 2$

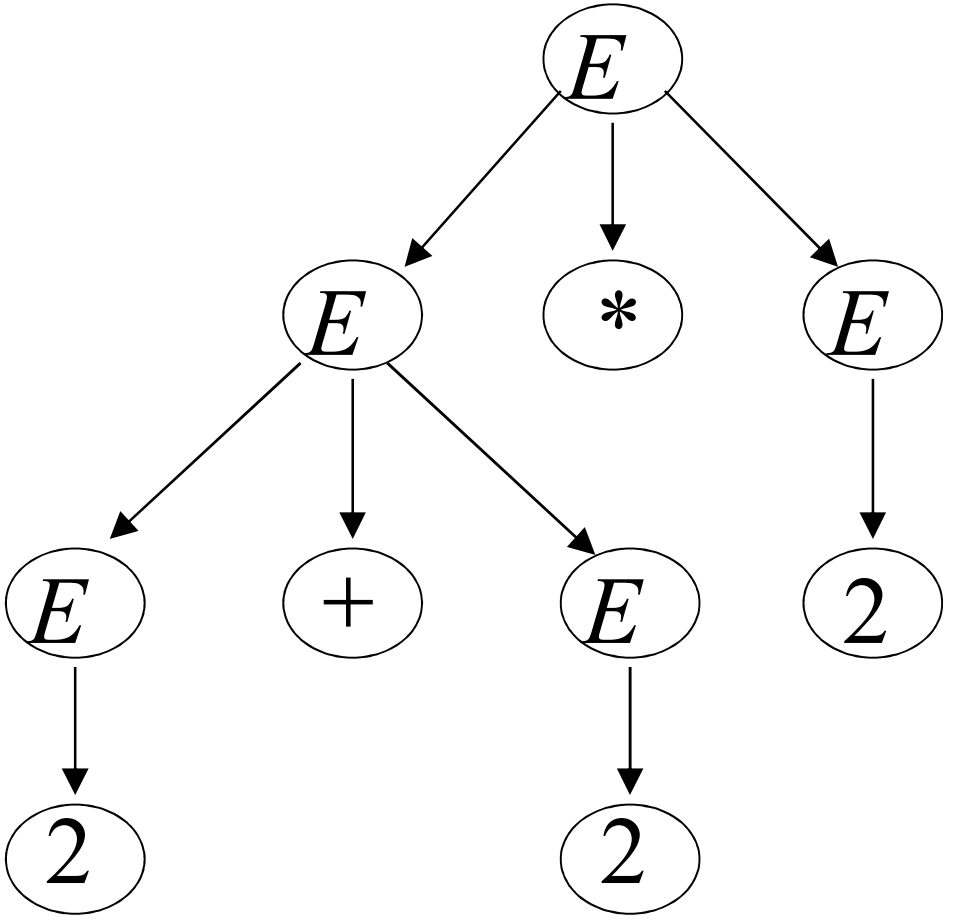




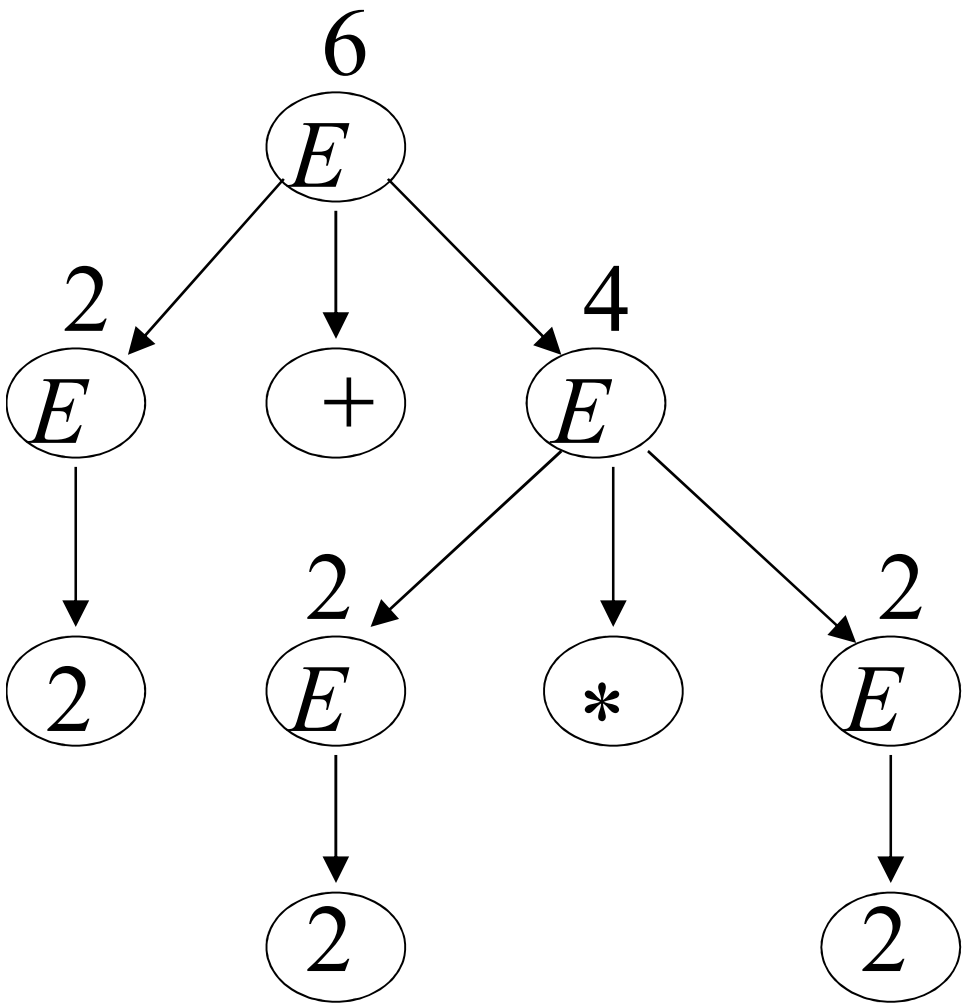
Why do we care about ambiguity?

$$2 + 2 * 2$$

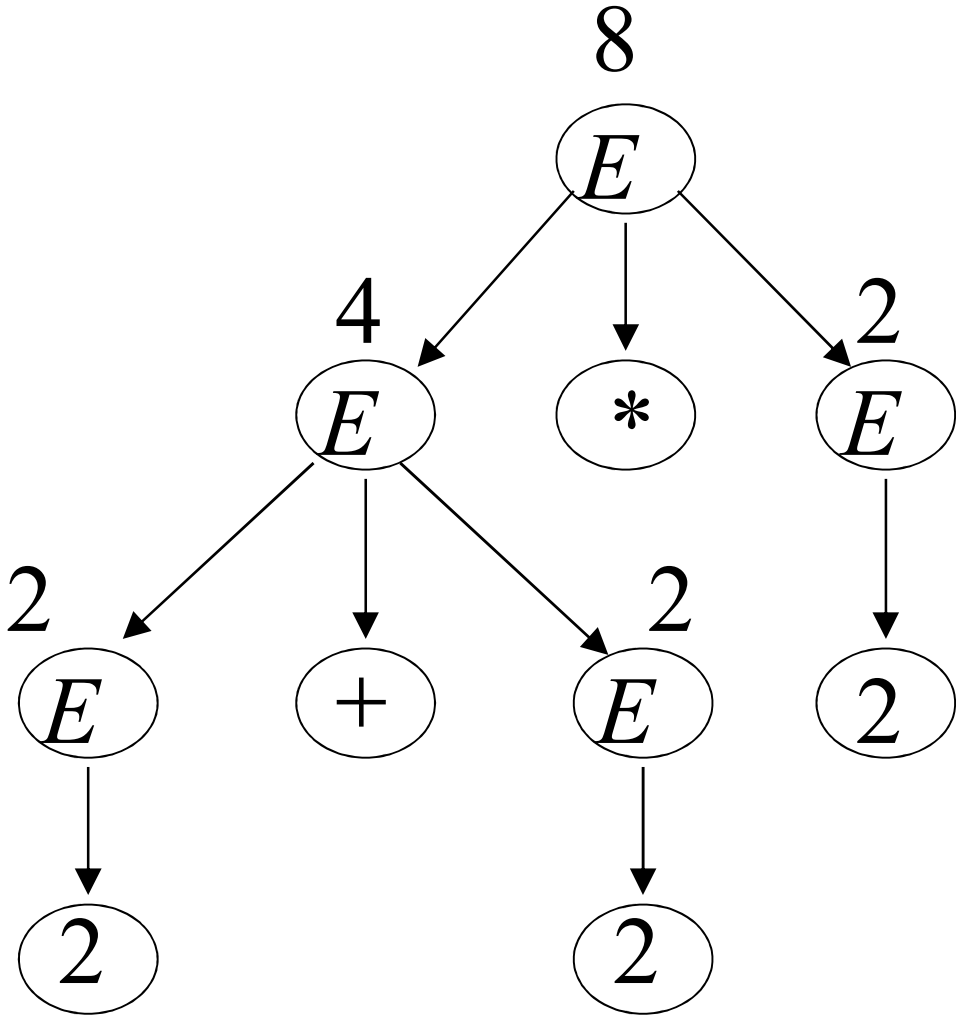




$$2 + 2 * 2 = 6$$



$$2 + 2 * 2 = 8$$

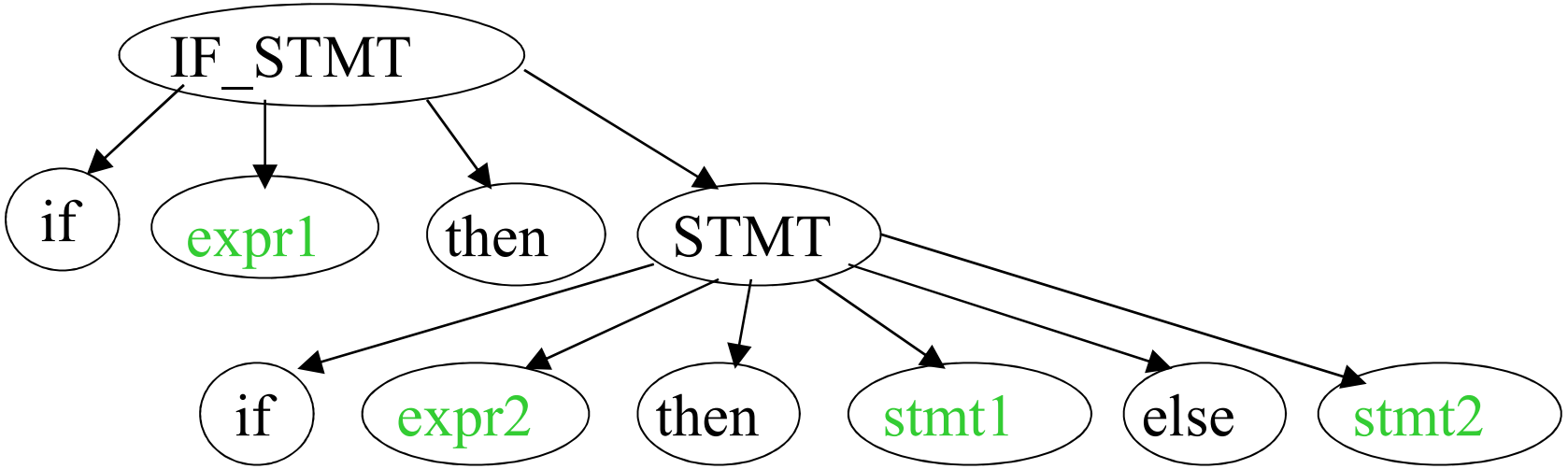


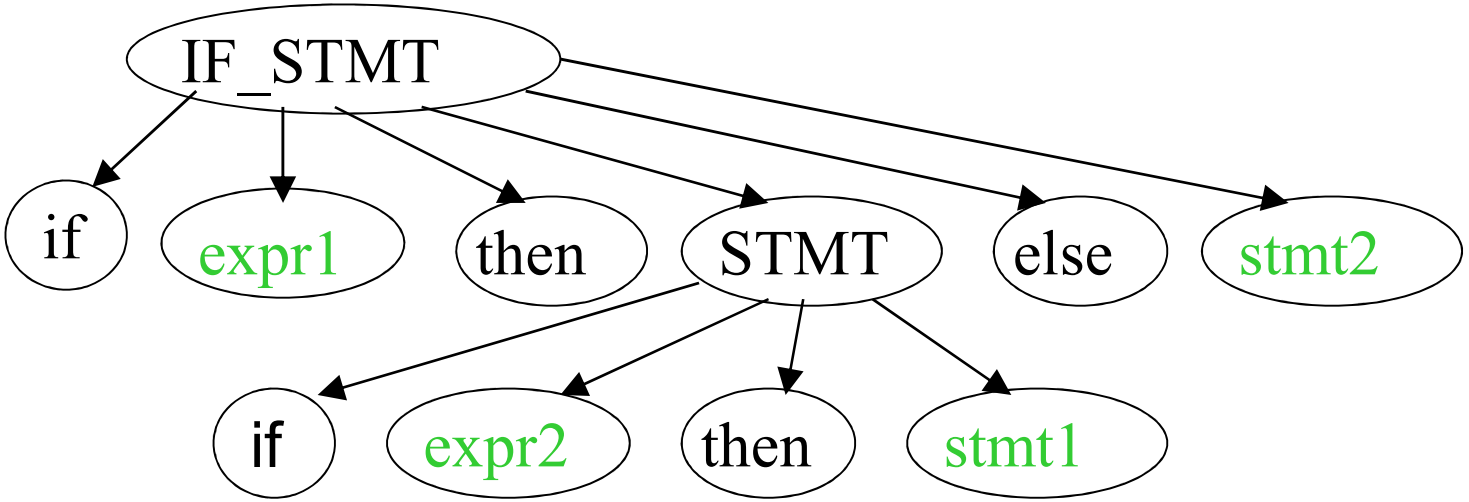
Another ambigious grammar

IF_STMT → if EXPR then STMT |

if EXPR then STMT else STMT

If expr1 then if expr2 then stmt1 else stmt2





Ambiguity

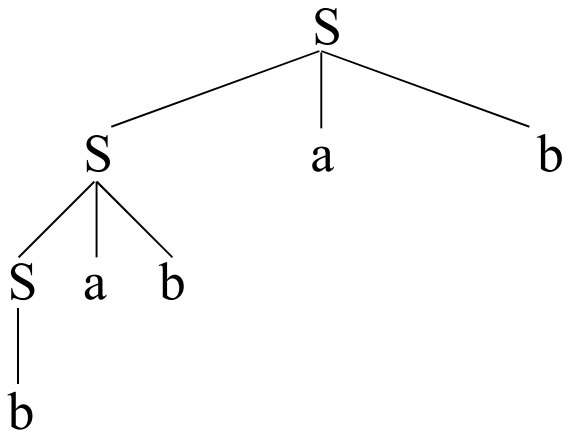
- We can create an equivalent CFG (which produces the same language) by eliminating the ambiguity from the following ambiguous CFG.

  $S \rightarrow SaS \mid b$

- In the following unambiguous CFG, we prefer left groupings.

  $S \rightarrow Sab \mid b$

- Now, there is only one parse tree for the string **babab**



Ambiguity – Operator Precedence

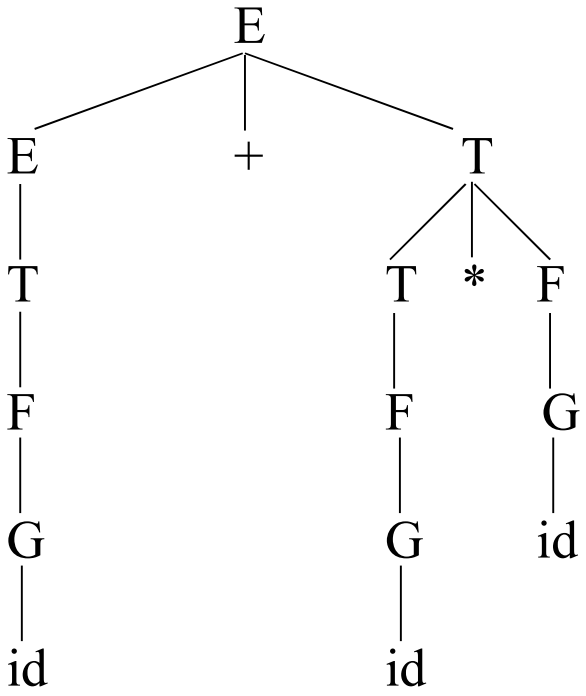
- Ambiguous grammars (because of ambiguous operators) can be disambiguated according to the precedence and associativity rules.

  E → E+E | E*E | E^E | id | (E)

- Disambiguated grammar:

  E → E+T | T
  T → T*F | F
  F → G^F | G
  G → id | (E)

  parse tree for **id+id*id**



Inherent Ambiguity

- Some CFLs may have both ambiguous grammars and unambiguous grammar.
  - In this case, we may disambiguate their ambiguous grammars.
- Unfortunately, there are some CFLs that do not have any unambiguous grammar.
- **A context free language L is said to be inherently ambiguous if all its grammars are ambiguous.**
- If even one grammar for L is unambiguous, then L is an unambiguous language.
  - Our expression language is an unambiguous language.
  - Even though the first grammar for expressions is ambiguous, there is another language for the expressions language is unambiguous.

Inherent Ambiguity

- **A context free language L is said to be inherently ambiguous if all its grammars are ambiguous.**

Example: Consider $L = \{a^n b^n c^m d^m : n \geq 1, m \geq 1\} \cup \{a^n b^m c^m d^n : n \geq 1, m \geq 1\}$

A grammar for L is

$S \rightarrow AB \mid C$

$A \rightarrow aAb \mid ab$

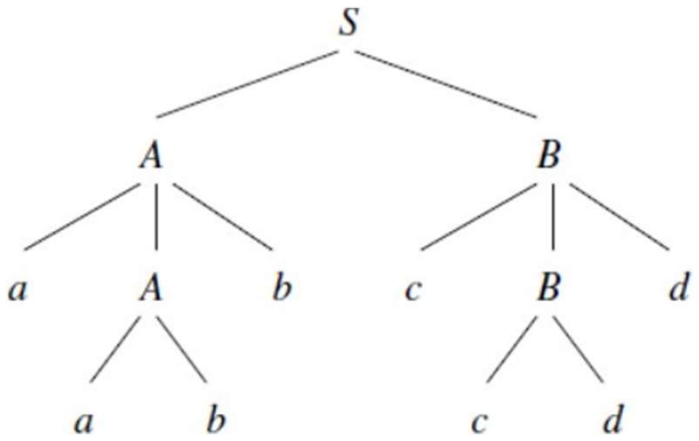
$B \rightarrow cBd \mid cd$

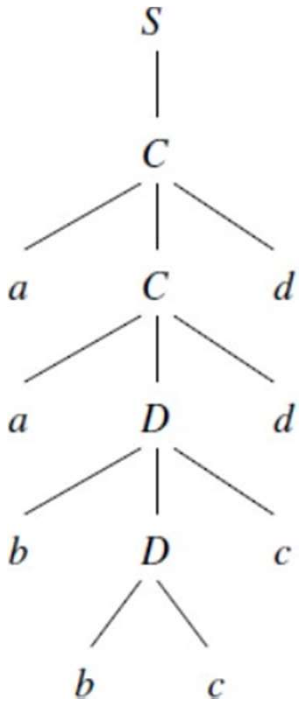
$C \rightarrow aCd \mid aDd$

$D \rightarrow bDc \mid bc$

Inherent Ambiguity

- The parse trees for the string **aabbccdd**.





S → AB | C
A → aAb | ab
B → cBd | cd
C → aCd | aDd
D → bDc | bc

$S \underset{lm}{\Rightarrow} AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aabbB$
$\underset{lm}{\Rightarrow} aabbcBd \underset{lm}{\Rightarrow} aabbccdd$

$S \underset{lm}{\Rightarrow} C \underset{lm}{\Rightarrow} aCd \underset{lm}{\Rightarrow} aaDdd \underset{lm}{\Rightarrow}$
$aabDcdd \underset{lm}{\Rightarrow} aabbccdd$

- It can be shown that every grammar for L behaves like the one above. The language L is inherently ambiguous.

Is Every Regular Language a CFL?

- Every regular language is a CFL.

Create a CFG for a given regular language whose DFA is given:

- A DFA $M = (Q, \Sigma, \delta, q_0, F)$ is given, define the CFG $\boldsymbol{G = (V, T, P, S)}$ as follows:

  $\boldsymbol{V} = \{ S_i \mid q_i \text{ is in } Q \}$

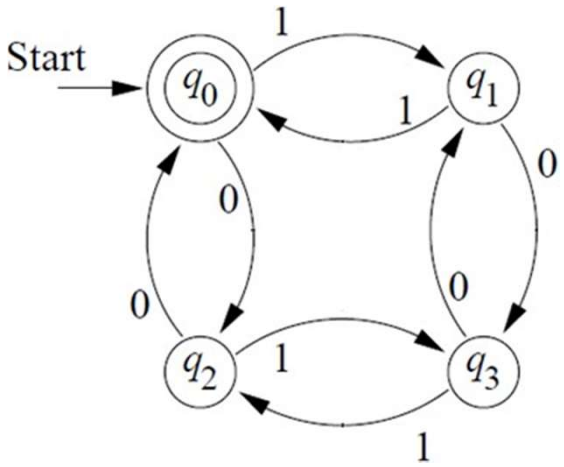
  $\boldsymbol{T} = \Sigma$

  $\boldsymbol{P} = \{ S_i \rightarrow aS_j \mid \delta(q_i, a) = q_j \} \cup \{ S_i \rightarrow \varepsilon \mid q_i \text{ is in } F \}$

  $\boldsymbol{S} = S_0$

Is Every Regular Language a CFL?

- Create a CFG for the following DFA:



$$S_0 \rightarrow 0S_2 \mid 1S_1 \mid \varepsilon$$

$$S_1 \rightarrow 0S_3 \mid 1S_0$$

$$S_2 \rightarrow 0S_0 \mid 1S_3$$

$$S_3 \rightarrow 0S_1 \mid 1S_2$$

CFG – Questions

- Design context-free grammars for the following languages:

- $\{0^n 1^m : n > m \geq 0\}$

  $S \rightarrow 0S1 \mid 0A$

  $A \rightarrow \varepsilon \mid 0A$

- The strings of 0's and 1's that contain equal number of 0's and 1's.

  $S \rightarrow 0S1S \mid 1S0S \mid \varepsilon$

CFG – Questions

- Design context-free grammars for the following language:
- $\{0^n 1^n : n \geq 0\} \cup \{1^n 0^n : n \geq 0\}$

  $S \rightarrow A \mid B$

  $A \rightarrow 0A1 \mid \varepsilon$

  $B \rightarrow 1B0 \mid \varepsilon$
- Is this grammar ambiguous?

  YES: Two leftmost derivations for ε

  $S \underset{lm}{\Rightarrow} A \underset{lm}{\Rightarrow} \varepsilon$ $\qquad$ $S \underset{lm}{\Rightarrow} B \underset{lm}{\Rightarrow} \varepsilon$
- Disambiguate this grammar.

  $S \rightarrow A \mid B \mid \varepsilon$

  $A \rightarrow 0A1 \mid 01$

  $B \rightarrow 1B0 \mid 10$