CONTEXT-FREE LANGUAGES

iACADEMY
SCHOOL OF COMPUTING • SCHOOL OF BUSINESS • SCHOOL OF DESIGN
SCHOOL OF COMPUTING
MITCH M. ANDAYA

CONTEXT-FREE GRAMMARS

- Finite automata and regular expressions are tools that are used for describing regular languages.

- ***Context-free grammars*** are like finite automata and regular expressions. However, they are more powerful tools because they can also describe non-regular languages.

- In linguistics, a ***grammar*** is a system of rules by which sentences are constructed by putting together words of the language.

CONTEXT-FREE GRAMMARS

- Example of a simple grammar, G_1, in English:

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

To form the sentence "the girl smiles":

1. start with rule #1:

   <sentence> → <noun phrase> <predicate>

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

<sentence> → <noun phrase> <predicate>

2. Then using rule #2, replace <noun phrase> with <article> <noun>:

<sentence> → <article> <noun> <predicate>

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

<sentence> → <article> <noun> <predicate>

3. Then using rule #3, replace <predicate> with <verb>:

    <sentence> → <article> <noun> <verb>

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

<sentence> → <article> <noun> <verb>

4. Then using rule #5, replace <article> with *the*:

    <sentence> → the <noun> <verb>

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

<sentence> → the <noun> <verb>

5. Then using the rule #7, replace <noun> with *girl*:

   <sentence> → the girl <verb>

CONTEXT-FREE GRAMMARS

1. <sentence> → <noun phrase> <predicate>
2. <noun phrase> → <article> <noun>
3. <predicate> → <verb>
4. <article> → a
5. <article> → the
6. <noun> → boy
7. <noun> → girl
8. <verb> → smiles
9. <verb> → laughs

<sentence> → the girl <verb>

6. Then using the rule #8, replace <verb> with *smiles*:

    <sentence> → the girl smiles

CONTEXT-FREE GRAMMARS

- In theory of computation, a ***grammar*** is a system of rules by which strings are constructed by putting together symbols of the language.

  Example: The grammar, G_2, for language $L = \{w \in \{0,1\}^* \mid w = w^R\}$ is

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

$S \rightarrow \varepsilon$

$S \rightarrow 0$

$S \rightarrow 1$

$S \rightarrow 0S0$

$S \rightarrow 1S1$

To form the string 1010101:

0S0

$S \rightarrow 1S1$

1S1

$S \rightarrow 10S01$

0

$S \rightarrow 101S101$

$S \rightarrow 1010101$

CONTEXT-FREE GRAMMARS

<sentence> → <noun phrase> <predicate>
<noun phrase> → <article> <noun>
<predicate> → <verb>
<article> → a
<article> → the
<noun> → boy
<noun> → girl
<verb> → smiles
<verb> → laughs

- A context-free grammar is simply a set of rules called ***substitution rules*** or ***productions***.

- The left-hand side of each production has a symbol, called a ***variable***, followed by an arrow and a string.

  In grammar G_1, the variables are *<sentence>*, *<noun phrase>*, *<predicate>*, *<article>*, *<noun>*, and *<verb>*.

iACADEMY
SCHOOL OF COMPUTING • SCHOOL OF BUSINESS • SCHOOL OF DESIGN

CONTEXT-FREE GRAMMARS

<sentence> → <noun phrase> <predicate>
<noun phrase> → <article> <noun>
<predicate> → <verb>
<article> → a
<article> → the
<noun> → boy
<noun> → girl
<verb> → smiles
<verb> → laughs

- The right-hand side of each production has a string composed of variables and other symbols called ***terminals***.

  Variables are symbols that can be replaced while terminals are symbols that cannot be replaced.

CONTEXT-FREE GRAMMARS

start variable

<sentence> → <noun phrase> <predicate>
<noun phrase> → <article> <noun>
<predicate> → <verb>
<article> → a
<article> → the
<noun> → boy
<noun> → girl
<verb> → smiles
<verb> → laughs

- One variable is assigned as the ***start variable***.

  It usually appears on the left-hand side of the first production.

  In grammar G_1, the start variable is *<sentence>*.

CONTEXT-FREE GRAMMARS

- By following the rules of a grammar, each string of that language can be generated. The procedure is:

1. Write down the rule that contains the start variable. It is usually the first rule of the grammar.

2. Find a variable on the right-hand side of the rule written in step 1 and find another rule that starts with that variable. Replace or substitute the variable with the right-hand side of that rule.

3. Repeat step 2 until no variables remain.

- The sequence of substitutions performed to obtain a string is called a ***derivation***.

CONTEXT-FREE GRAMMARS

- Rules with the same left-hand side variable may be combined into a single rule with their right-hand side strings separated by a |.

For grammar G_1:

<article> → a
<article> → the
<noun> → boy
<noun> → girl
<verb> → smiles
<verb> → laughs

<article> → a | the
<noun> → boy | girl
<verb> → smiles | laughs

CONTEXT-FREE GRAMMARS

- Rules with the same left-hand side variable may be combined into a single rule with their right-hand side strings separated by a |.

  For grammar G_2:

$$
\begin{aligned}
S &\rightarrow \varepsilon \\
S &\rightarrow 0 \\
S &\rightarrow 1 \\
S &\rightarrow 0S0 \\
S &\rightarrow 1S1
\end{aligned}
\quad \Rightarrow \quad
S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1
$$

CONTEXT-FREE GRAMMARS

- All strings generated or derived from a grammar G constitute the ***language of that grammar*** and is written as $L(G)$.

  For grammar G_1:

  $L(G_1)$ = {the girl smiles, the girl laughs, the boy smiles, the boy laughs, a girl smiles, a girl laughs, a boy smiles, a boy laughs}

  For grammar G_2:

  $L(G_2) = \{w \in \{0,1\}^* \mid w = w^R\}$

- Any language that can be generated by some context-free grammar is called a ***context-free language***.

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

- A ***context-free grammar*** is a 4-tuple (V, Σ, R, S), where

    1. V is a finite set of variables,
    2. Σ is a finite set of terminals,
    3. R is a finite set of rules, with each rule being a variable on the left-hand side and a string of variables and/or terminals on the right-hand side, and
    4. S is the start variable.

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

- Examples:

  Given grammar G_3 as

  1. $A \rightarrow 0A1 \mid \varepsilon$

  The formal definition of grammar G_3 is $G_3 = (V, \Sigma, R, S)$

  where $V = \{A\}$
  $\Sigma = \{0, 1\}$
  $R = \{A \rightarrow 0A1 \mid \varepsilon\}$
  $S = A$

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

Some sample derivations using grammar G_3:

$$A \rightarrow 0A1 \mid \varepsilon$$

$$
\begin{aligned}
A &\rightarrow 0\overset{0A1}{A}1 \\
&\rightarrow 00\overset{\varepsilon}{A}11 \\
&\rightarrow 0011
\end{aligned}
$$

$$
\begin{aligned}
A &\rightarrow 0\overset{0A1}{A}1 \\
&\rightarrow 00\overset{0A1}{A}11 \\
&\rightarrow 000\overset{\varepsilon}{A}111 \\
&\rightarrow 000111
\end{aligned}
$$

$$
\begin{aligned}
A &\rightarrow 0\overset{0A1}{A}1 \\
&\rightarrow 00\overset{0A1}{A}11 \\
&\rightarrow 000\overset{0A1}{A}111 \\
&\rightarrow 0000\overset{\varepsilon}{A}1111 \\
&\rightarrow 00001111
\end{aligned}
$$

$$L(G_3) = \{0^x1^x \mid x \geq 0\}$$

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

Given grammar G_4 as

1. $A \rightarrow B1$
2. $B \rightarrow 0B1 \mid \varepsilon$

The formal definition of grammar G_4 is $G_4 = (V, \Sigma, R, S)$

where $V = \{A, B\}$

$\Sigma = \{0, 1\}$

$R = \{A \rightarrow B1 , B \rightarrow 0B1 \mid \varepsilon\}$

$S = A$

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

Some sample derivations using grammar G_4:

$$A \to B1$$

$$B \to 0B1 \mid \varepsilon$$

$$\begin{aligned} A &\to \overset{0B1}{B}1 \\ &\to 0\overset{\varepsilon}{B}11 \\ &\to 011 \end{aligned}$$

$$\begin{aligned} A &\to \overset{0B1}{B}1 \\ &\to 0\overset{0B1}{B}11 \\ &\to 00\overset{\varepsilon}{B}111 \\ &\to 00111 \end{aligned}$$

$$\begin{aligned} A &\to \overset{0B1}{B}1 \\ &\to 0\overset{0B1}{B}11 \\ &\to 00\overset{0B1}{B}111 \\ &\to 000\overset{\varepsilon}{B}1111 \\ &\to 0001111 \end{aligned}$$

$$L(G_4) = \{0^x 1^{x+1} \mid x \geq 0\}$$

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

Given grammar G_5 as

1. $A \rightarrow (A) \mid AA \mid \varepsilon$

The formal definition of grammar G_5 is $G_5 = (V, \Sigma, R, S)$

where $V = \{A\}$

$\Sigma = \{(,)\}$

$R = \{A \rightarrow (A) \mid AA \mid \varepsilon\}$

$S = A$

FORMAL DEFINITION OF CONTEXT-FREE GRAMMAR

Some sample derivations using grammar G_5:

$$A \rightarrow (A) \mid AA \mid \varepsilon$$

$$\begin{aligned} A &\rightarrow (\overset{\varepsilon}{A}) \\ &\rightarrow () \end{aligned}$$

$$\begin{aligned} A &\rightarrow (\overset{AA}{A}) \\ &\rightarrow (\overset{(A)}{A}A) \\ &\rightarrow ((A)\overset{(A)}{A}) \\ &\rightarrow ((\overset{\varepsilon}{A})(A)) \\ &\rightarrow (()(\overset{\varepsilon}{A})) \\ &\rightarrow (()()) \end{aligned}$$

$$\begin{aligned} A &\rightarrow (\overset{AA}{A}) \\ &\rightarrow (\overset{(A)}{A}A) \\ &\rightarrow ((A)\overset{(A)}{A}) \\ &\rightarrow ((\overset{\varepsilon}{A})(A)) \\ &\rightarrow (()(\overset{AA}{A})) \\ &\rightarrow (()(\overset{(A)}{A}A)) \\ &\rightarrow (()((A)\overset{(A)}{A})) \\ &\rightarrow (()((\overset{\varepsilon}{A})(A))) \\ &\rightarrow (()(()(\overset{\varepsilon}{A}))) \\ &\rightarrow (()(()())) \end{aligned}$$

$L(G_5)$ is the language of all strings of properly nested parentheses.

LEFTMOST AND RIGHTMOST DERIVATIONS

- Consider the following grammar G_6 whose rules are

$$S \rightarrow A1B$$
$$A \rightarrow 0A \mid \varepsilon$$
$$B \rightarrow 0B \mid 1B \mid \varepsilon$$

The set of variables $V = \{S, A, B\}$, the set of terminals $\Sigma = \{0, 1\}$, and the start variable is S.

- In the process of deriving a string, there will be situations where there will be more than one variable on the right-hand side.

LEFTMOST AND RIGHTMOST DERIVATIONS

- The derivation obtained by substituting the leftmost variable at each step is called the ***leftmost derivation***.

  Example:

  In deriving the string 00101 using leftmost derivation:

$$
\begin{aligned}
S &\rightarrow A1B \\
A &\rightarrow 0A \mid \varepsilon \\
B &\rightarrow 0B \mid 1B \mid \varepsilon
\end{aligned}
$$

0A

$$S \rightarrow A1B$$

0A

$$\rightarrow 0A1B$$

ε

$$\rightarrow 00A1B$$

0B

$$\rightarrow 001B$$

1B

$$\rightarrow 0010B$$

ε

$$\rightarrow 00101B$$

$$\rightarrow 00101$$

LEFTMOST AND RIGHTMOST DERIVATIONS

- The derivation obtained by substituting the rightmost variable at each step is called the ***rightmost derivation***.

  Example:

  In deriving the string 00101 using rightmost derivation:

$$
\begin{aligned}
S &\rightarrow A1B \\
A &\rightarrow 0A \mid \varepsilon \\
B &\rightarrow 0B \mid 1B \mid \varepsilon
\end{aligned}
$$

$$
\begin{aligned}
S &\rightarrow A1B && 0B \\
&\rightarrow A10B && 1B \\
&\rightarrow A101B && \varepsilon \\
&\rightarrow A101 && 0A \\
&\rightarrow 0A101 && 0A \\
&\rightarrow 00A101 && \varepsilon \\
&\rightarrow 00101
\end{aligned}
$$

LEFTMOST AND RIGHTMOST DERIVATIONS

Example:

Consider grammar G_7 with the following rules:

$$S \rightarrow 0AB$$
$$A \rightarrow 1B1$$
$$B \rightarrow A \mid \varepsilon$$

Give the leftmost and rightmost derivation of the string 01111.

LEFTMOST AND RIGHTMOST DERIVATIONS

01111

$S \to 0AB$

$A \to 1B1$

$B \to A \mid \varepsilon$

Leftmost

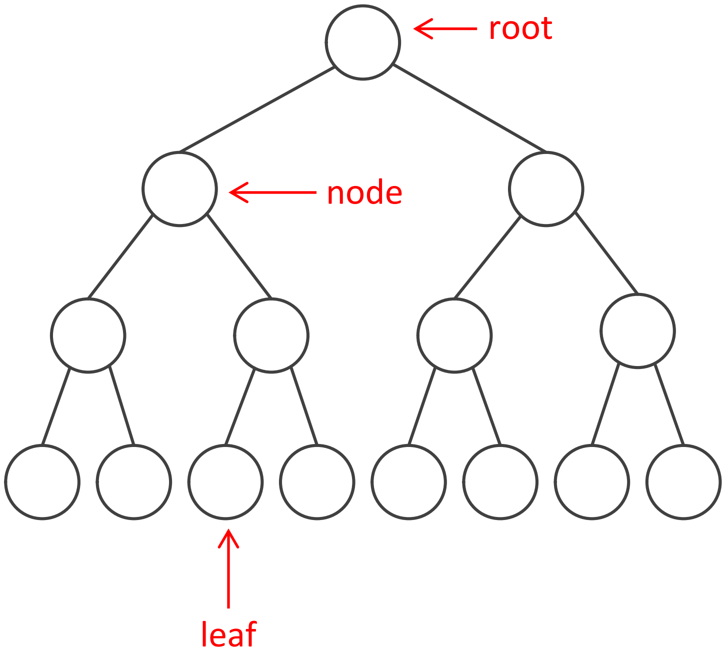
$$
\begin{aligned}
S &\to 0\overset{1B1}{A}B \\
&\to 01\overset{A}{B}1B \\
&\to 01\overset{1B1}{A}1B \\
&\to 011\overset{\varepsilon}{B}11B \\
&\to 01111\overset{\varepsilon}{B} \\
&\to 01111
\end{aligned}
$$

Rightmost

$$
\begin{aligned}
S &\to 0A\overset{\varepsilon}{B} \\
&\to 0\overset{1B1}{A} \\
&\to 01\overset{A}{B}1 \\
&\to 01\overset{1B1}{A}1 \\
&\to 011\overset{\varepsilon}{B}11 \\
&\to 01111
\end{aligned}
$$

PARSE TREES

- The derivations obtained from a context-free grammar can be represented graphically using a tree structure called ***parse trees*** or ***derivation trees***.
- Parse trees give a visualization of the entire derivation of a string.
- The variables occupy the internal nodes of a tree with the start variable being the root of the tree. The terminals occupy the leaf at the bottom
- The children of an internal node (variables) are the right-hand side string of a rule used to expand the variable.

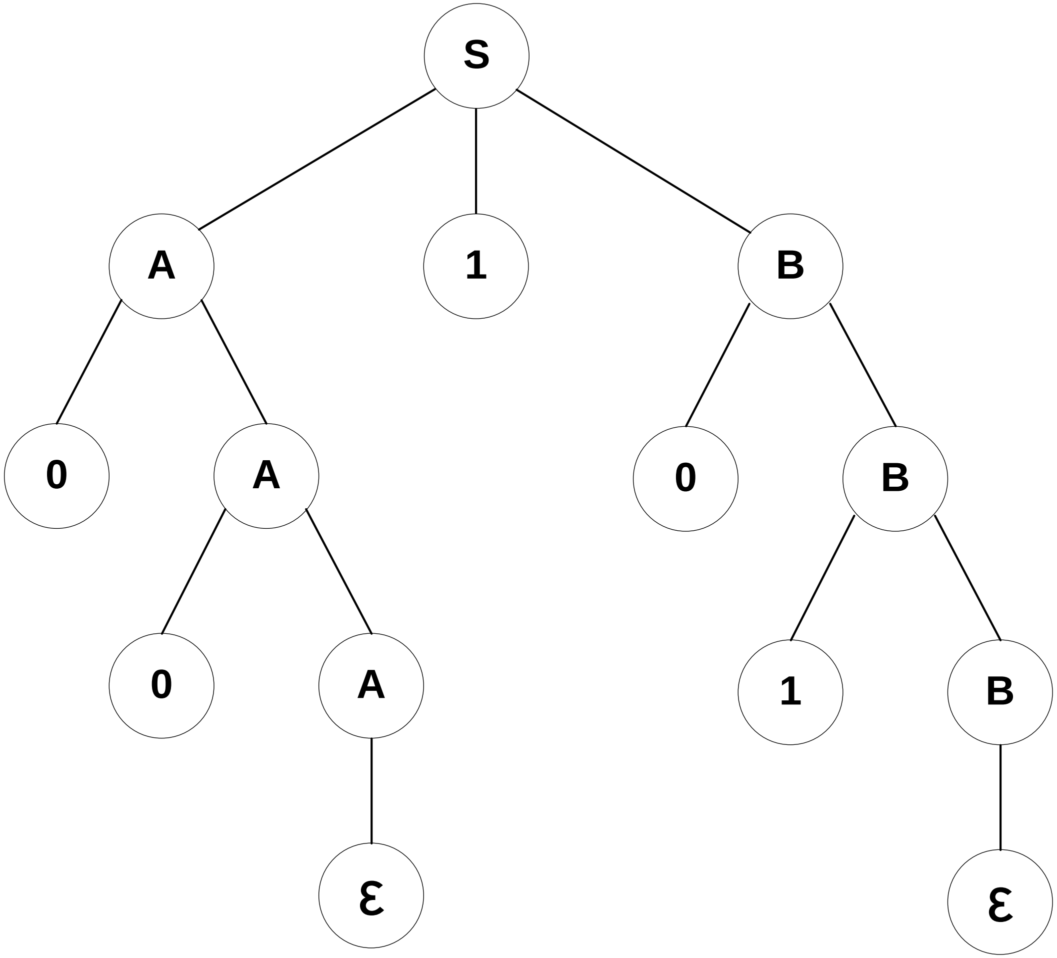


PARSE TREES

Example:

The parse tree for the derivation of the string 00101 using grammar G_6.

$$
\begin{aligned}
S &\rightarrow A1B \\
&\rightarrow A10B \\
&\rightarrow A101B \\
&\rightarrow A101 \\
&\rightarrow 0A101 \\
&\rightarrow 00A101 \\
&\rightarrow 00101
\end{aligned}
$$



AMBIGUITY IN CFGs

- Some grammars that may generate the same string in different ways.
- This means that a string may have different meanings or structures.
- For some situations, this may not cause any problem.
- But for applications like programming languages where it is important to have a unique structure or interpretation for each line of code, this may cause numerous errors.

AMBIGUITY IN CFGs

- Consider the following grammar G = (V, Σ, R, S) where

  V = {<expr>}
  Σ = {+, ×, a, b, c, (,)}
  S = {<expr>}

  and the set of rules *R* is

  1. <expr> → <expr>+<expr> | <expr>×<expr> | (<expr>) | a | b | c

AMBIGUITY IN CFGs

<expr> → <expr>+<expr> | <expr>×<expr> |
(<expr>) | a | b | c

The leftmost derivation for the string *a+b+c* is as follows:

a

<expr> → <expr> + <expr>   <expr> + <expr>

→ a + <expr>   b

→ a + <expr> + <expr>   c

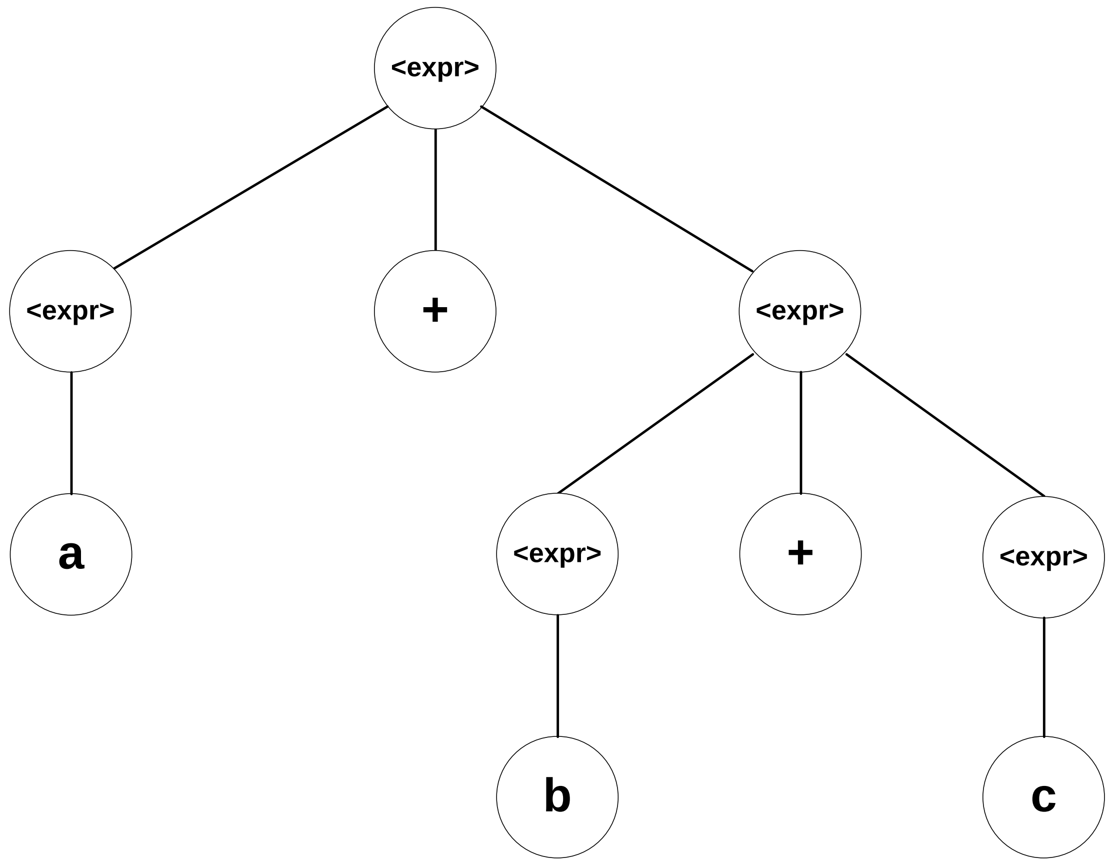
→ a + b + <expr>

→ a + b + c

AMBIGUITY IN CFGs

The parse tree for this derivation is

<expr> → <expr> + <expr>
→ a + <expr>
→ a + <expr> + <expr>
→ a + b + <expr>
→ a + b + c



AMBIGUITY IN CFGs

<expr> → <expr>+<expr> | <expr>×<expr> |
(<expr>) | a | b | c

However, there is another leftmost derivation for the string *a+b+c* which is as follows:

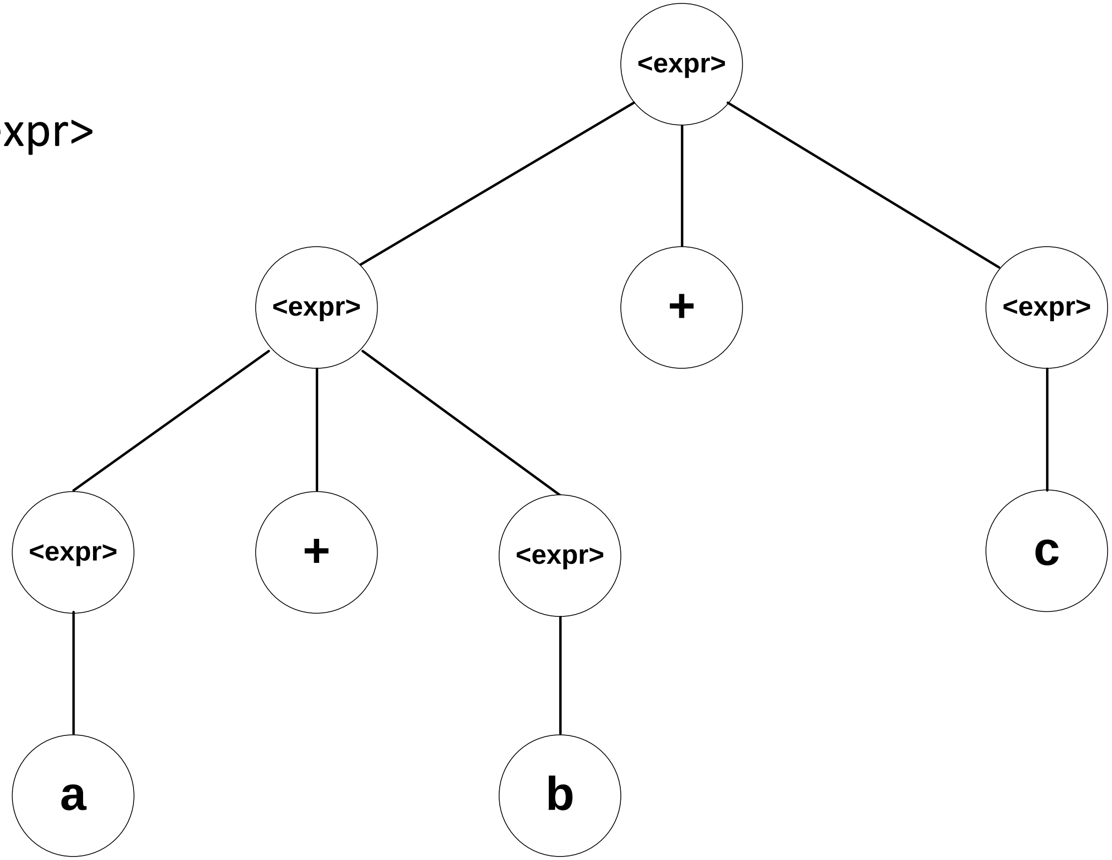
<expr> + <expr>

<expr> → <expr> + <expr>
→ <expr> + <expr> + <expr>
a
→ a + <expr> + <expr>
c
b
→ a + b + <expr>
→ a + b + c

AMBIGUITY IN CFGs

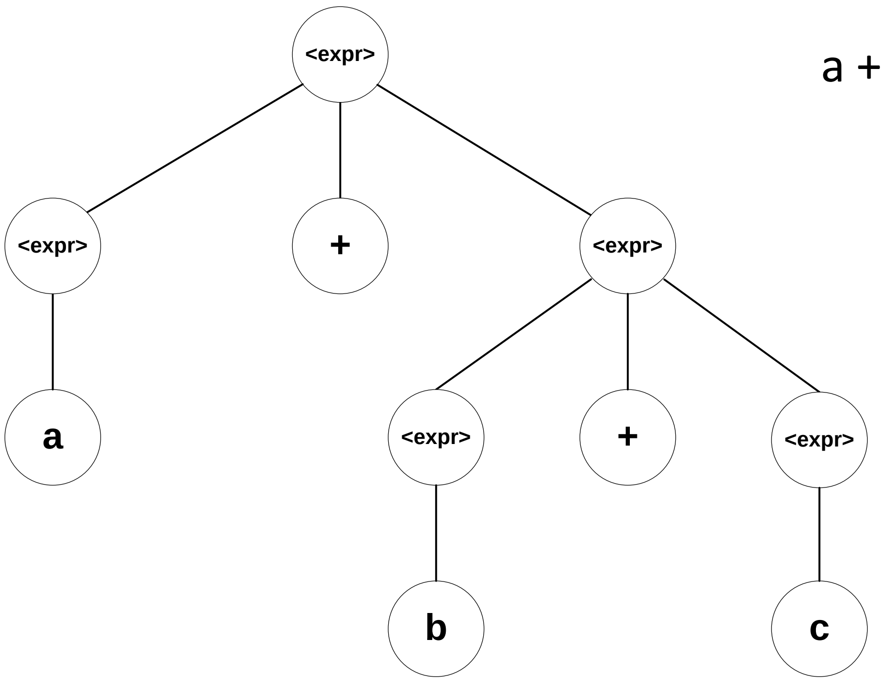
The parse tree for this derivation is

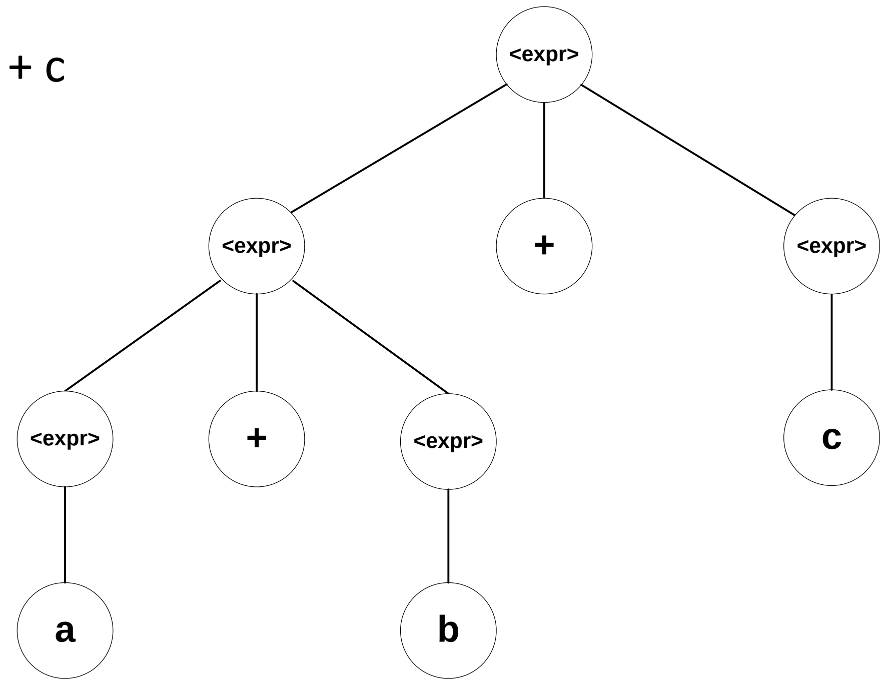
<expr> → <expr> + <expr>
→ <expr> + <expr> + <expr>
→ a + <expr> + <expr>
→ a + b + <expr>
→ a + b + c



AMBIGUITY IN CFGs

a + b + c





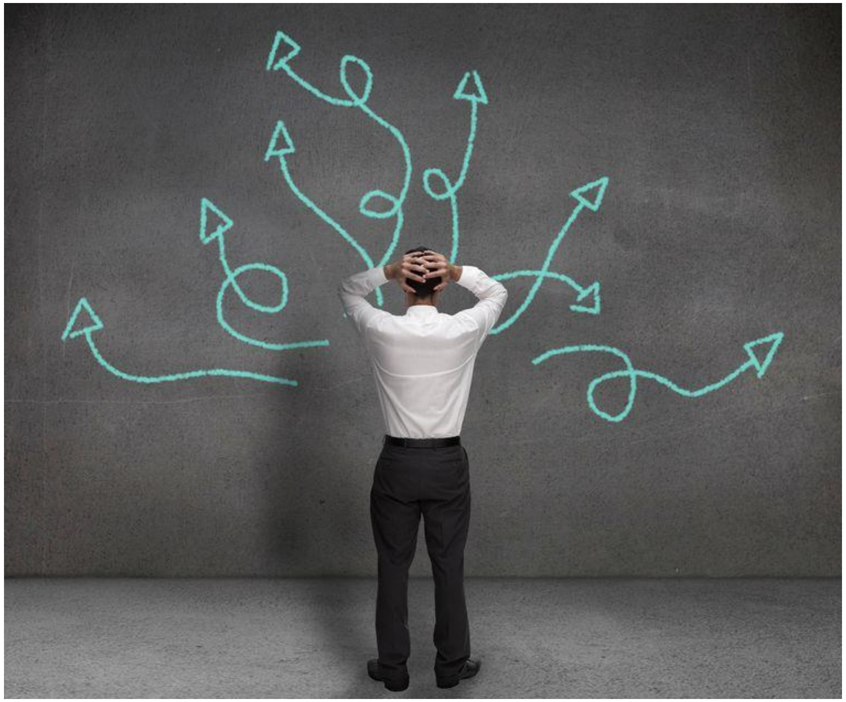
The first derivation actually added *b+c* first. And then it added *a* to this sum.

The second derivation actually added *a+b* first. And then it added this sum to *c*.

AMBIGUITY IN CFGs

- If a string can be derived from a grammar in more than one way, then the grammar is said to be *ambiguous* and the string is derived ***ambiguously***.

- In other words, when a string has two or more different leftmost derivations (or rightmost derivations) or two or more parse trees, then the grammar is ambiguous.



AMBIGUITY IN CFGs

<expr> → <expr>+<expr> | <expr>×<expr> |
(<expr>) | a | b | c

As another example, the leftmost derivation for the string *a+b×c* is as follows:

a

<expr> → <expr> + <expr>   <expr> × <expr>

→ a + <expr>

b

→ a + <expr> × <expr>

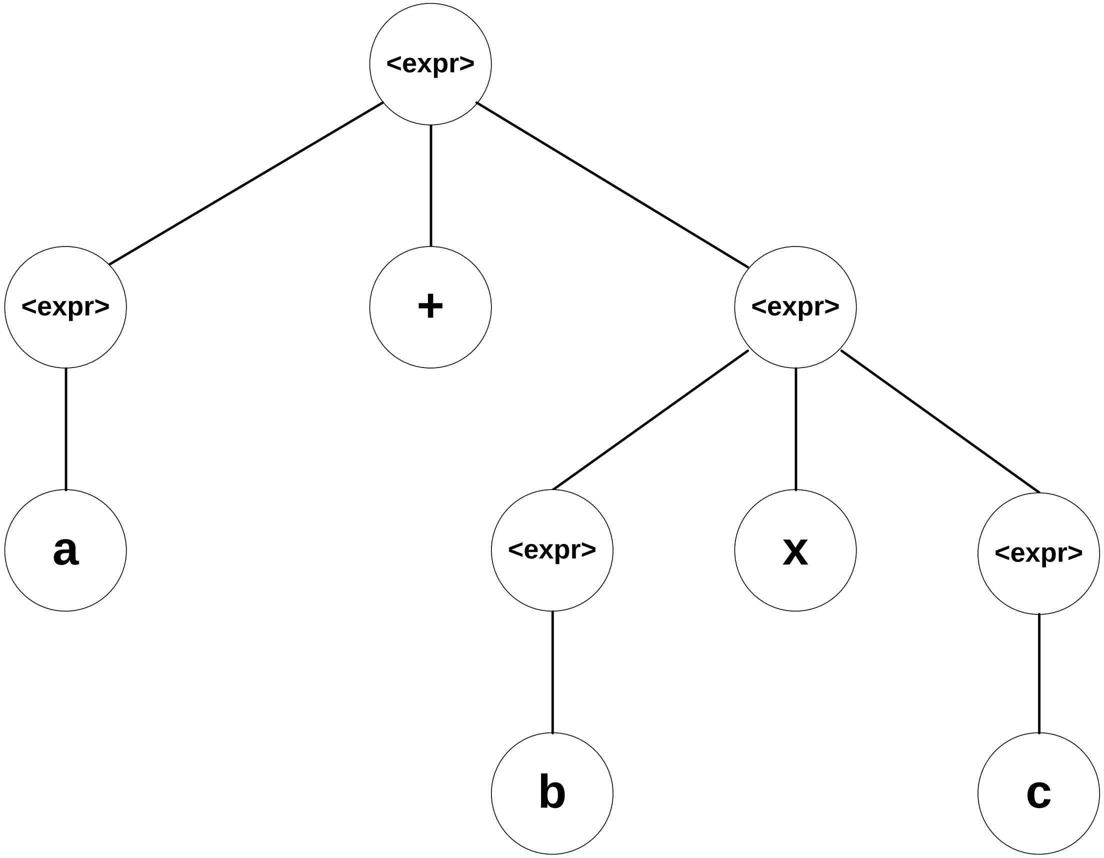
→ a + b × <expr>   c

→ a + b × c

AMBIGUITY IN CFGs

The parse tree for this derivation is

<expr> → <expr> + <expr>
→ a + <expr>
→ a + <expr> × <expr>
→ a + b × <expr>
→ a + b × c



AMBIGUITY IN CFGs

<expr> → <expr>+<expr> | <expr>×<expr> |
(<expr>) | a | b | c

However, there is another leftmost derivation for the string *a+b×c* which is as follows:

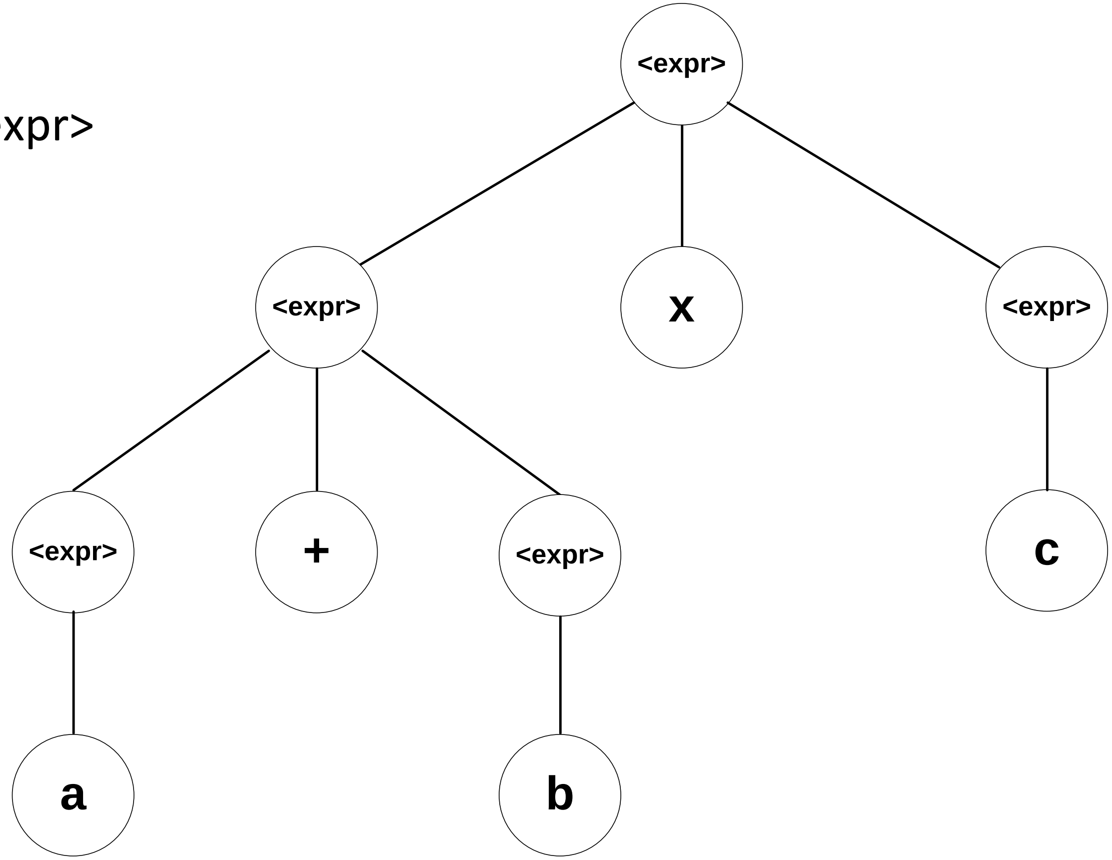
<expr> + <expr>

<expr> → <expr> × <expr>
→ <expr> + <expr> × <expr>
a → a + <expr> × <expr>
b → a + b × <expr> c
→ a + b × c

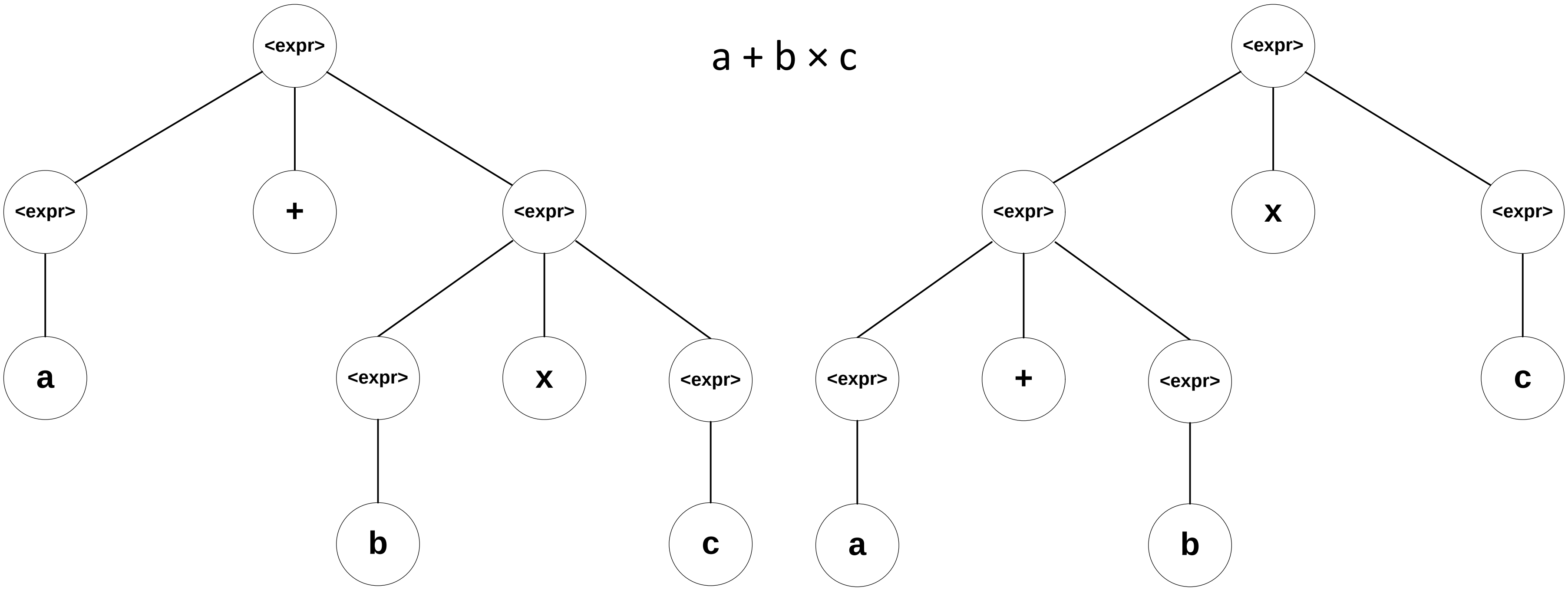
AMBIGUITY IN CFGs

The parse tree for this derivation is

<expr> → <expr> × <expr>
→ <expr> + <expr> × <expr>
→ a + <expr> × <expr>
→ a + b × <expr>
→ a + b × c



AMBIGUITY IN CFGs



The first derivation actually multiplied *b×c* first. And then it added *a* to the product.

The second derivation actually added *a+b* first. And then it multiplied the sum to *c*.

AMBIGUITY IN CFGs

- Because the multiplication operation has a higher precedence over the addition operation, the first derivation will produce the correct result.

- If ambiguity in a CFG will produce erroneous results, the grammar must be rewritten to remove any ambiguities.

FORMAL DEFINITION OF AMBIGUITY

- A string *w* is derived ***ambiguously*** in context-free grammar *G* if it has two or more distinct leftmost derivations, two or more distinct rightmost derivations, or two or more distinct parse trees.

- Grammar *G* is ***ambiguous*** if it generates some string ambiguously.

CHOMSKY NORMAL FORM

- Context-free grammars do not have any restrictions or limitations on how the right-hand side of each production or rule is written. A rule may be as simple as

$$A \rightarrow 0$$

- Or it may be as complicated as

$$A \rightarrow BCD010EFG101101BEC$$

- Complex productions such as the one shown above may render the context-free grammar difficult to understand, analyze, and use.

CHOMSKY NORMAL FORM

- Simplified versions of context-free grammars are very useful in analyzing the grammar, constructing proofs about certain properties of the grammar, testing if a particular string is a member of the language of the grammar, etc.

- These simplified versions are called ***normal forms***.

- One popular normal form of context-free grammars is the ***Chomsky Normal Form***.

FORMAL DEFINITION OF CHOMSKY NORMAL FORM

- A context-free grammar is in the ***Chomsky Normal Form*** if every rule is of the form:

$$A \rightarrow BC$$
$$A \rightarrow a$$

where:

1. *A*, *B*, and *C* are any variables
2. *a* is any terminal
3. *B* and *C* may not be the start variable
4. the rule $S \rightarrow \varepsilon$ where *S* is the start variable, is allowed

FORMAL DEFINITION OF CHOMSKY NORMAL FORM

- Example:

Grammar G_1: $S \rightarrow XY \mid 0 \mid \varepsilon$
$X \rightarrow YY \mid 1$
$Y \rightarrow 0$

Grammar G_2: $S \rightarrow XS \mid XYY \mid 0X \mid 0$
$X \rightarrow SX \mid 11 \mid \varepsilon$
$Y \rightarrow 1$

Grammar G_1 is in the Chomsky normal form while grammar G_2 is not.

CONVERTING A CFG INTO ITS CNF

- Case Study: Convert the following grammar G_3 into the Chomsky normal form.

$$S \rightarrow ASA \mid 0B$$

$$A \rightarrow B \mid S$$

$$B \rightarrow 1 \mid \varepsilon$$

CONVERTING A CFG INTO ITS CNF

Step 1:

Create a new start variable S_o and the production $S_o \rightarrow S$, where S is the original start variable of the grammar being converted.

$S \rightarrow ASA \mid 0B$
$A \rightarrow B \mid S$
$B \rightarrow 1 \mid \varepsilon$

$\Rightarrow$

$S_o \rightarrow S$
$S \rightarrow ASA \mid 0B$
$A \rightarrow B \mid S$
$B \rightarrow 1 \mid \varepsilon$

CONVERTING A CFG INTO ITS CNF

Step 2:

For every variable A that is not a start variable, remove all $A \rightarrow \varepsilon$ rules (***the ε - productions***).

Then, for each occurrence of A on right hand side of a production, add a new rule with the A deleted.

For example, if there is a rule $B \rightarrow AC$, add the rule $B \rightarrow C$.

If there is a rule $B \rightarrow A$, add the rule $B \rightarrow \varepsilon$, unless $B \rightarrow \varepsilon$ was previously removed.

CONVERTING A CFG INTO ITS CNF

Removing $B \rightarrow \varepsilon$:

$S \rightarrow 0$

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \\
A &\rightarrow B \mid S \\
B &\rightarrow 1 \mid \varepsilon
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \mid 0 \\
A &\rightarrow B \mid S \mid \varepsilon \\
B &\rightarrow 1
\end{aligned}
$$

$A \rightarrow \varepsilon$

iACADEMY

SCHOOL OF COMPUTING • SCHOOL OF BUSINESS • SCHOOL OF DESIGN

CONVERTING A CFG INTO ITS CNF

Removing $A \rightarrow \varepsilon$:

$S \rightarrow SA \mid AS \mid S$

$$\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \mid 0 \\
A &\rightarrow B \mid S \mid \varepsilon \\
B &\rightarrow 1
\end{aligned}$$

$$\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \mid 0 \mid \\
&\qquad SA \mid AS \mid S \\
A &\rightarrow B \mid S \\
B &\rightarrow 1
\end{aligned}$$

CONVERTING A CFG INTO ITS CNF

Step 3:

Remove all rules of the form $A \rightarrow B$ (***the unit productions***).

For each rule $B \rightarrow w$ (where w is a string of variables and terminals) that appears, add the rule $A \rightarrow w$ unless $A \rightarrow w$ was previously removed.

Rules of the form $A \rightarrow A$ can just simply be removed since these are useless productions.

CONVERTING A CFG INTO ITS CNF

Removing $S \rightarrow S$:

$$\begin{aligned} S_o &\rightarrow S \\ S &\rightarrow ASA \mid oB \mid o \mid \\ &\qquad SA \mid AS \mid S \\ A &\rightarrow B \mid S \\ B &\rightarrow 1 \end{aligned}$$

$$\begin{aligned} S_o &\rightarrow S \\ S &\rightarrow ASA \mid oB \mid o \mid \\ &\qquad SA \mid AS \\ A &\rightarrow B \mid S \\ B &\rightarrow 1 \end{aligned}$$

CONVERTING A CFG INTO ITS CNF

Removing $A \rightarrow B$:

$S_o \rightarrow S$
$S \rightarrow ASA \mid oB \mid o \mid SA \mid AS$
$A \rightarrow B \mid S$
$B \rightarrow 1$

$A \rightarrow 1$

$S_o \rightarrow S$
$S \rightarrow ASA \mid oB \mid o \mid SA \mid AS$
$A \rightarrow S \mid 1$
$B \rightarrow 1$

CONVERTING A CFG INTO ITS CNF

Removing $A \rightarrow S$:

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \mid 0 \mid \\
&\qquad SA \mid AS \\
A &\rightarrow S \mid 1 \\
B &\rightarrow 1
\end{aligned}
$$

$$
\begin{aligned}
S_0 &\rightarrow S \\
S &\rightarrow ASA \mid 0B \mid 0 \mid \\
&\qquad SA \mid AS \\
A &\rightarrow 1 \mid ASA \mid 0B \mid \\
&\qquad 0 \mid SA \mid AS \\
B &\rightarrow 1
\end{aligned}
$$

$A \rightarrow ASA|0B|0|SA|AS$

CONVERTING A CFG INTO ITS CNF

Removing $S_o \rightarrow S$:

$S_o \rightarrow ASA \mid oB \mid o \mid SA \mid AS$

$S_o \rightarrow S$
$S \rightarrow ASA \mid oB \mid o \mid SA \mid AS$
$A \rightarrow 1 \mid ASA \mid oB \mid o \mid SA \mid AS$
$B \rightarrow 1$

$S_o \rightarrow ASA \mid oB \mid o \mid SA \mid AS$
$S \rightarrow ASA \mid oB \mid o \mid SA \mid AS$
$A \rightarrow 1 \mid ASA \mid oB \mid o \mid SA \mid AS$
$B \rightarrow 1$

CONVERTING A CFG INTO ITS CNF

Step 4:

Convert all the rules into the proper form by removing the rules of the form:

$$A \rightarrow w_1 w_2 \ldots w_k$$

where $k \geq 3$ and w_i is a variable or terminal

and adding the rules $A \rightarrow w_1 T_1$, $T_1 \rightarrow w_2 T_2$, $T_2 \rightarrow w_3 T_3$, and so on and so forth (where T_1, T_2, and T_3 are new variables).

If w_i is a terminal, replace it with a new variable T_i and add the rule $T_i \rightarrow w_i$.

CONVERTING A CFG INTO ITS CNF

Applying Rule 4:

$T_1 \rightarrow SA$ $\qquad$ $T_2 \rightarrow 0$

$S_0 \rightarrow ASA \mid 0B \mid 0 \mid SA \mid AS$

$S \rightarrow ASA \mid 0B \mid 0 \mid SA \mid AS$

$A \rightarrow 1 \mid ASA \mid 0B \mid 0 \mid SA \mid AS$

$B \rightarrow 1$

$S_0 \rightarrow AT_1 \mid T_2B \mid 0 \mid SA \mid AS$

$S \rightarrow AT1 \mid T_2B \mid 0 \mid SA \mid AS$

$A \rightarrow 1 \mid AT_1 \mid T_2B \mid 0 \mid SA \mid AS$

$B \rightarrow 1$

$T_1 \rightarrow SA$

$T_2 \rightarrow 0$

The resulting grammar G_3 in Chomsky normal form will be

$$
\begin{array}{ll}
S_0 & \rightarrow AT_1 \mid T_2B \mid 0 \mid SA \mid AS \\
S & \rightarrow AT1 \mid T_2B \mid 0 \mid SA \mid AS \\
A & \rightarrow 1 \mid AT_1 \mid T_2B \mid 0 \mid SA \mid AS \\
T_1 & \rightarrow SA \\
T_2 & \rightarrow 0 \\
B & \rightarrow 1
\end{array}
$$

CONVERTING A CFG INTO ITS CNF

- Example:

  Convert the following grammar G_4 into the Chomsky normal form.

$$\begin{aligned} A &\rightarrow BAB \mid B \mid \varepsilon \\ B &\rightarrow 00 \mid \varepsilon \end{aligned}$$

  Step 1: Add $S_0 \rightarrow A$

$$\begin{aligned} S_0 &\rightarrow A \\ A &\rightarrow BAB \mid B \mid \varepsilon \\ B &\rightarrow 00 \mid \varepsilon \end{aligned}$$

CONVERTING A CFG INTO ITS CNF

Step 2: Remove ε - productions

Remove $A \rightarrow \varepsilon$:

$S_o \rightarrow \boldsymbol{\varepsilon}$

$S_o \rightarrow A$
$A \rightarrow BAB \mid B \mid \varepsilon$
$B \rightarrow oo \mid \varepsilon$

$A \rightarrow BB$

$S_o \rightarrow A \mid \varepsilon$
$A \rightarrow BAB \mid B \mid BB$
$B \rightarrow oo \mid \varepsilon$

CONVERTING A CFG INTO ITS CNF

Step 2: Remove ε - productions

Remove $B \rightarrow \varepsilon$:

$$
\begin{aligned}
S_o &\rightarrow A \mid \varepsilon \\
A &\rightarrow BAB \mid B \mid BB \\
B &\rightarrow OO \mid \varepsilon
\end{aligned}
$$

$A \rightarrow AB \mid BA$

$$
\begin{aligned}
S_o &\rightarrow A \mid \varepsilon \\
A &\rightarrow BAB \mid B \mid BB \\
&\quad \mid AB \mid BA \\
B &\rightarrow OO
\end{aligned}
$$

CONVERTING A CFG INTO ITS CNF

Step 3: Remove unit-productions

Remove $A \rightarrow B$:

$A \rightarrow oo$

$$
\begin{aligned}
S_o &\rightarrow A \mid \varepsilon \\
A &\rightarrow BAB \mid B \mid BB \\
&\quad \mid AB \mid BA \\
B &\rightarrow oo
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
S_o &\rightarrow A \mid \varepsilon \\
A &\rightarrow BAB \mid BB \mid AB \\
&\quad \mid BA \mid oo \\
B &\rightarrow oo
\end{aligned}
$$

CONVERTING A CFG INTO ITS CNF

Step 3: Remove unit-productions

Remove $S_o \rightarrow A$:

$A \rightarrow BAB \mid BB \mid AB \mid BA \mid oo$

$S_o \rightarrow A \mid \varepsilon$
$A \rightarrow BAB \mid BB \mid AB \mid BA \mid oo$
$B \rightarrow oo$

$S_o \rightarrow BAB \mid BB \mid AB \mid BA \mid oo \mid \varepsilon$
$A \rightarrow BAB \mid BB \mid AB \mid BA \mid oo$
$B \rightarrow oo$

iACADEMY
SCHOOL OF COMPUTING • SCHOOL OF BUSINESS • SCHOOL OF DESIGN

CONVERTING A CFG INTO ITS CNF

Step 4: Convert all the rules into the proper form.

$T_1 \rightarrow AB$

$S_0 \rightarrow BAB \mid BB \mid AB \mid BA \mid 00 \mid \varepsilon$

$A \rightarrow BAB \mid BB \mid AB \mid BA \mid 00$

$B \rightarrow 00$

$T_2 \rightarrow 0$

$S_0 \rightarrow BT_1 \mid BB \mid AB \mid BA \mid T_2T_2 \mid \varepsilon$

$A \rightarrow BT_1 \mid BB \mid AB \mid BA \mid T_2T_2$

$B \rightarrow T_2T_2$

$T_1 \rightarrow AB$

$T_2 \rightarrow 0$

CONVERTING A CFG INTO ITS CNF

The resulting grammar G_4 in Chomsky normal form will be:

$$
\begin{aligned}
S_o &\rightarrow BT_1 \mid BB \mid AB \mid BA \mid T_2T_2 \mid \varepsilon \\
A &\rightarrow BT_1 \mid BB \mid AB \mid BA \mid T_2T_2 \\
B &\rightarrow T_2T_2 \\
T_1 &\rightarrow AB \\
T_2 &\rightarrow 0
\end{aligned}
$$

iACADEMY
SCHOOL OF COMPUTING • SCHOOL OF BUSINESS • SCHOOL OF DESIGN

EXERCISES

- Let *G* be the grammar:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T \times F \mid F$$
$$F \rightarrow (E) \mid a$$

Give the parse trees for each string:

a. a+a+a+a

b. ((a))

- Given the following grammar:

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

Show that the grammar is ambiguous by giving two leftmost derivations for the string 0101.

EXERCISES

- Convert the following context-free grammar into its Chomsky normal form:

$$S \rightarrow ASB \mid \varepsilon$$
$$A \rightarrow 0AS \mid 0$$
$$B \rightarrow S1S \mid A \mid 00$$