Parse tree and ambiguity

Parse tree representation of the derivation.

- It is tree representation of the derivation.
- For a given derivation, there is only one parse tree.
- But, for a given parse tree, there may be many derivations.

$$\begin{array}{llll} 1. & P & \rightarrow & \epsilon \\ 2. & P & \rightarrow & 0 \\ 3. & P & \rightarrow & 1 \\ 4. & P & \rightarrow & 0P0 \\ 5. & P & \rightarrow & 1P1 \end{array}$$

Figure 5.1: A context-free grammar for palindromes

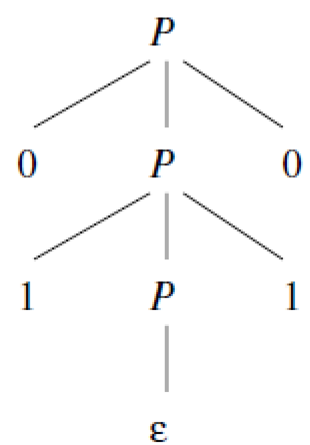


Figure 5.5: A parse tree showing the derivation $P \stackrel{*}{\Rightarrow} 0110$

1.	E	$\rightarrow$	I
2.	E	$\rightarrow$	$E + E$
3.	E	$\rightarrow$	$E * E$
4.	E	$\rightarrow$	(E)
5.	I	$\rightarrow$	a
6.	I	$\rightarrow$	b
7.	I	$\rightarrow$	Ia
8.	I	$\rightarrow$	Ib
9.	I	$\rightarrow$	$I0$
10.	I	$\rightarrow$	$I1$

Figure 5.2: A context-free grammar for simple expressions

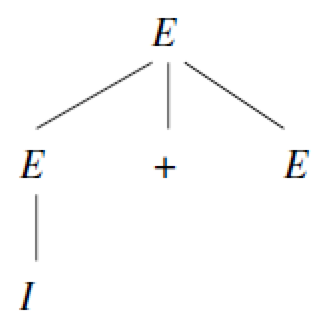


Figure 5.4: A parse tree showing the derivation of $I + E$ from E

1. Each interior node is labeled by a variable in V.

2. Each leaf is labeled by either a variable, a terminal, or ϵ. However, if the leaf is labeled ϵ, then it must be the only child of its parent.

3. If an interior node is labeled A, and its children are labeled

$$X_1, X_2, \ldots, X_k$$

respectively, from the left, then $A \to X_1 X_2 \cdots X_k$ is a production in P.

5.2.2 The Yield of a Parse Tree

1. The yield is a terminal string. That is, all leaves are labeled either with a terminal or with ϵ.

2. The root is labeled by the start symbol.

1.	E	$\rightarrow$	I
2.	E	$\rightarrow$	$E + E$
3.	E	$\rightarrow$	$E * E$
4.	E	$\rightarrow$	(E)
5.	I	$\rightarrow$	a
6.	I	$\rightarrow$	b
7.	I	$\rightarrow$	Ia
8.	I	$\rightarrow$	Ib
9.	I	$\rightarrow$	$I0$
10.	I	$\rightarrow$	$I1$

Figure 5.2: A context-free grammar for simple expressions

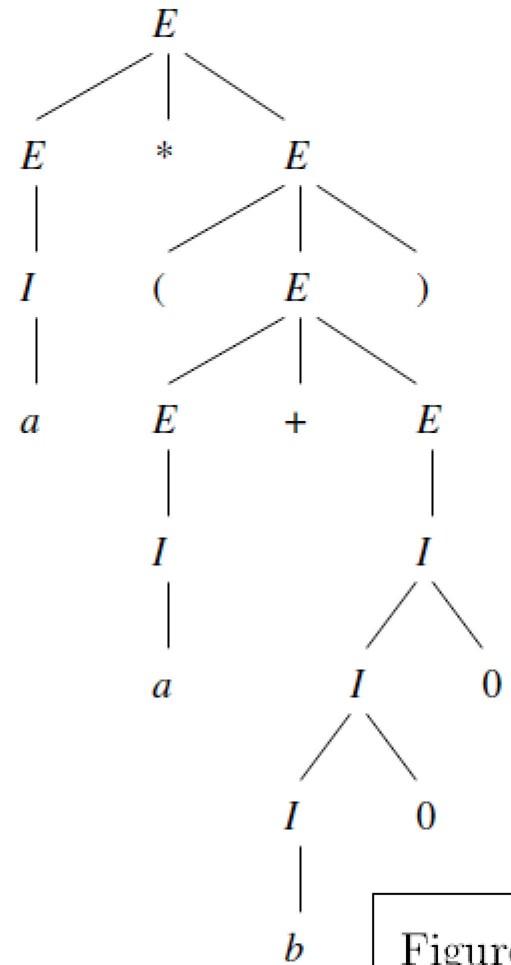


Figure 5.6:

Parse tree for the yield $a * (a + b00)$

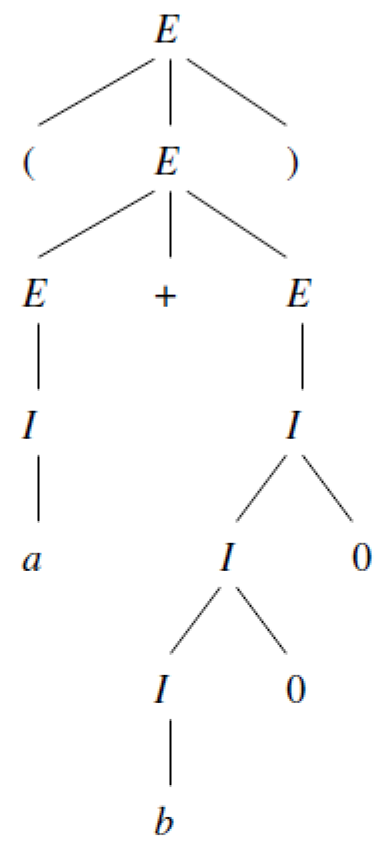
Parse tree representation of the derivation.

- It is tree representation of the derivation.
- For a given derivation, there is only one parse tree.
- But, for a given parse tree, there may be many derivations.

- For a given parse tree there is a unique leftmost derivation.
- Similarly, for a given parse tree there is a unique rightmost derivation.

$$E \underset{lm}{\Rightarrow} (E) \underset{lm}{\Rightarrow} (E+E) \underset{lm}{\Rightarrow} (I+E) \underset{lm}{\Rightarrow} (a+E) \underset{lm}{\Rightarrow}$$

$$(a+I) \underset{lm}{\Rightarrow} (a+I0) \underset{lm}{\Rightarrow} (a+I00) \underset{lm}{\Rightarrow} (a+b00)$$

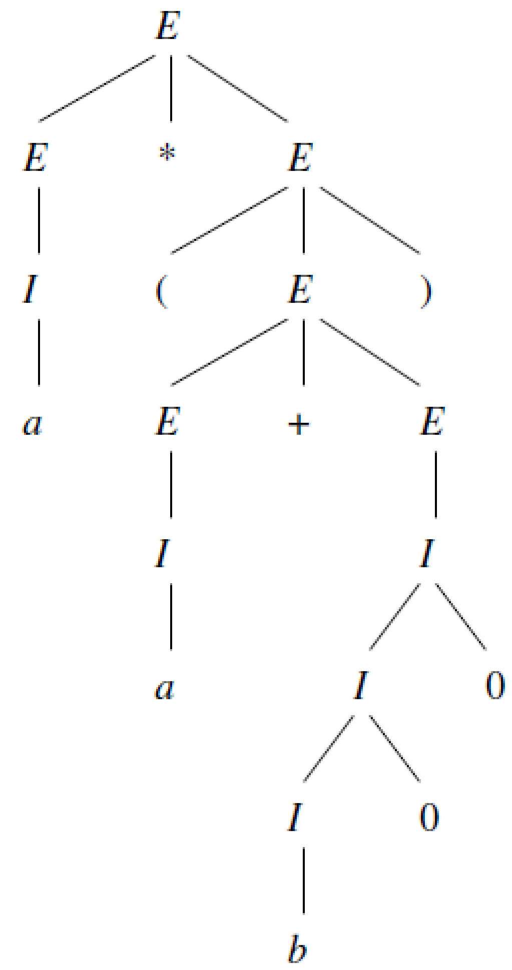


- Can you find the rightmost derivation?

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow} I * E \underset{lm}{\Rightarrow} a * E \underset{lm}{\Rightarrow}$$

$$a * (E) \underset{lm}{\Rightarrow} a * (E + E) \underset{lm}{\Rightarrow} a * (I + E) \underset{lm}{\Rightarrow} a * (a + E) \underset{lm}{\Rightarrow}$$

$$a * (a + I) \underset{lm}{\Rightarrow} a * (a + I0) \underset{lm}{\Rightarrow} a * (a + I00) \underset{lm}{\Rightarrow} a * (a + b00)$$



- Can you find the rightmost derivation?

- Can you do this?

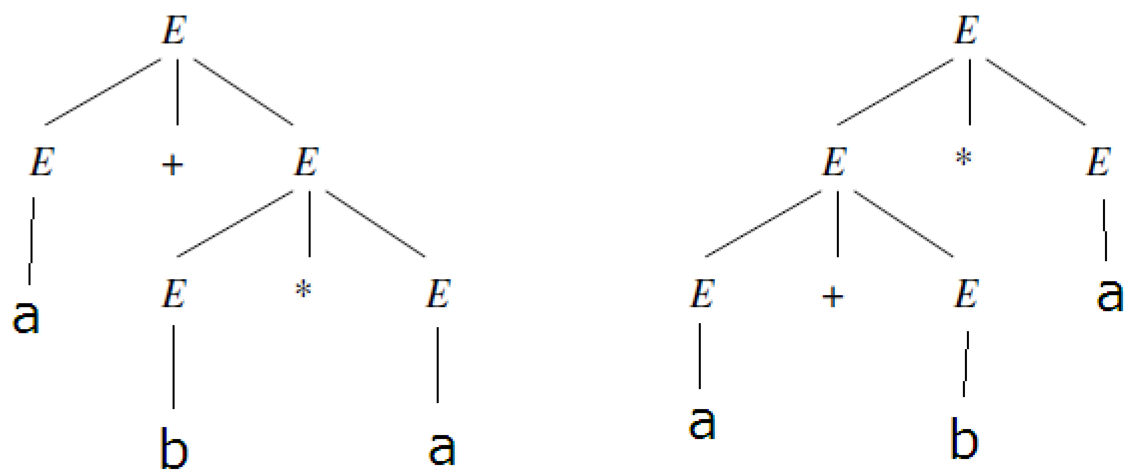
! **Exercise 5.2.2:** Suppose that G is a CFG without any productions that have ϵ as the right side. If w is in $L(G)$, the length of w is n, and w has a derivation of m steps, show that w has a parse tree with $n + m$ nodes.

Ambiguous grammar

- The CFG is ambiguous, if there is a string in the language for which there are more than one parse tree.
- This is equivalent to say, “there are more than one leftmost derivation for a string, hence the grammar is ambiguous”.
- Similarly, with rightmost derivation

- Two parse trees for the yield a+b*a

$$
\begin{aligned}
E &\rightarrow E + E \\
E &\rightarrow E * E \\
E &\rightarrow a \\
E &\rightarrow b
\end{aligned}
$$



Can we remove the ambiguity?

- Finding whether a given CFG is ambiguous or not is an undecidable problem !
- There are some CFLs for which it is impossible to have an unambiguous CFG.

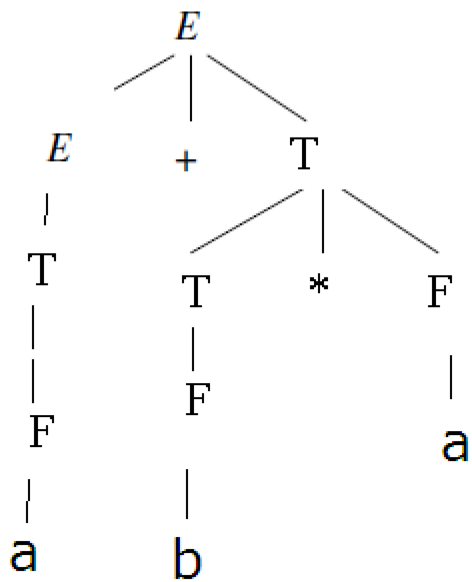
Can we remove the ambiguity?

- Finding whether a given CFG is ambiguous or not is an undecidable problem !
- There are some CFLs for which it is impossible to have an unambiguous CFG.

- But, the situation is not so unpromising.
- For many situations in practice, we can handcraft unambiguous CFG for a given ambiguous one.

$$
\begin{aligned}
E &\rightarrow T \mid E + T \\
T &\rightarrow F \mid T * F \\
F &\rightarrow a \mid b
\end{aligned}
$$

An unambiguous expression grammar



This is the only parse tree for a+b*a

- For an expression grammar, injecting precedence and associativity of operators can make them unambiguous.

$$
\begin{array}{llll}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E+E \\
3. & E & \rightarrow & E*E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

Figure 5.2: A context-free grammar for simple expressions

$$
\begin{array}{lll}
E & \rightarrow & T \mid E+T \\
T & \rightarrow & F \mid T*F \\
F & \rightarrow & I \mid (E) \\
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1
\end{array}
$$

Figure 5.19: An unambiguous expression grammar

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Figure 5.19: An unambiguous expression grammar

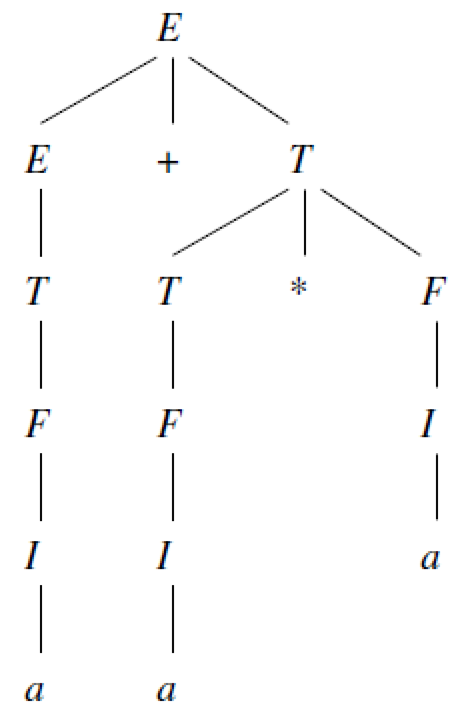


Figure 5.20: The sole parse tree for $a + a * a$

$$
\begin{array}{llcl}
1. & E & \rightarrow & I \\
2. & E & \rightarrow & E + E \\
3. & E & \rightarrow & E * E \\
4. & E & \rightarrow & (E) \\
\\
5. & I & \rightarrow & a \\
6. & I & \rightarrow & b \\
7. & I & \rightarrow & Ia \\
8. & I & \rightarrow & Ib \\
9. & I & \rightarrow & I0 \\
10. & I & \rightarrow & I1
\end{array}
$$

Figure 5.2: A context-free grammar for simple expressions

- With above we get two parse trees for the same yield.

$$
\begin{array}{lcl}
I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \\
F & \rightarrow & I \mid (E) \\
T & \rightarrow & F \mid T * F \\
E & \rightarrow & T \mid E + T
\end{array}
$$

Figure 5.19: An unambiguous expression grammar

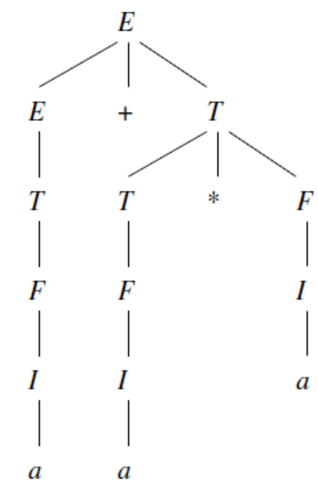


Figure 5.20: The sole parse tree for $a + a * a$

Inherent ambiguity

- A CFL is said to be inherently ambiguous, if every CFG that generates the language is ambiguous.
- Note, in this case we say CFL is ambiguous.
  - Earlier we said CFG is ambiguous.

An example of ambiguous CFL

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

That is, L consists of strings in $\mathbf{a}^+\mathbf{b}^+\mathbf{c}^+\mathbf{d}^+$ such that either:

1. There are as many a's as b's and as many c's as d's, or
2. There are as many a's as d's and as many b's as c's.

An example of ambiguous CFL

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

That is, L consists of strings in $\mathbf{a}^+\mathbf{b}^+\mathbf{c}^+\mathbf{d}^+$ such that either:

1. There are as many a's as b's and as many c's as d's, or
2. There are as many a's as d's and as many b's as c's.

$$\begin{array}{lcl} S & \rightarrow & AB \mid C \\ A & \rightarrow & aAb \mid ab \\ B & \rightarrow & cBd \mid cd \\ C & \rightarrow & aCd \mid aDd \\ D & \rightarrow & bDc \mid bc \end{array}$$

One CFG, for the CFL

An example of ambiguous CFL

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

That is, L consists of strings in $\mathbf{a}^+\mathbf{b}^+\mathbf{c}^+\mathbf{d}^+$ such that either:

1. There are as many a's as b's and as many c's as d's, or
2. There are as many a's as d's and as many b's as c's.

$$\begin{array}{lcl} S & \rightarrow & AB \mid C \\ A & \rightarrow & aAb \mid ab \\ B & \rightarrow & cBd \mid cd \\ C & \rightarrow & aCd \mid aDd \\ D & \rightarrow & bDc \mid bc \end{array}$$

One CFG, for the CFL

1. $S \underset{lm}{\Rightarrow} AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aabbB \underset{lm}{\Rightarrow} aabbcBd \underset{lm}{\Rightarrow} aabbccdd$
2. $S \underset{lm}{\Rightarrow} C \underset{lm}{\Rightarrow} aCd \underset{lm}{\Rightarrow} aaDdd \underset{lm}{\Rightarrow} aabDcdd \underset{lm}{\Rightarrow} aabbccdd$

Two leftmost derivations for the same string

$$
\begin{array}{lcl}
S & \rightarrow & AB \mid C \\
A & \rightarrow & aAb \mid ab \\
B & \rightarrow & cBd \mid cd \\
C & \rightarrow & aCd \mid aDd \\
D & \rightarrow & bDc \mid bc
\end{array}
$$

One CFG, for the CFL

1. $S \underset{lm}{\Rightarrow} AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aabbB \underset{lm}{\Rightarrow} aabbcBd \underset{lm}{\Rightarrow} aabbccdd$
2. $S \underset{lm}{\Rightarrow} C \underset{lm}{\Rightarrow} aCd \underset{lm}{\Rightarrow} aaDdd \underset{lm}{\Rightarrow} aabDcdd \underset{lm}{\Rightarrow} aabbccdd$

Two leftmost derivations for the same string

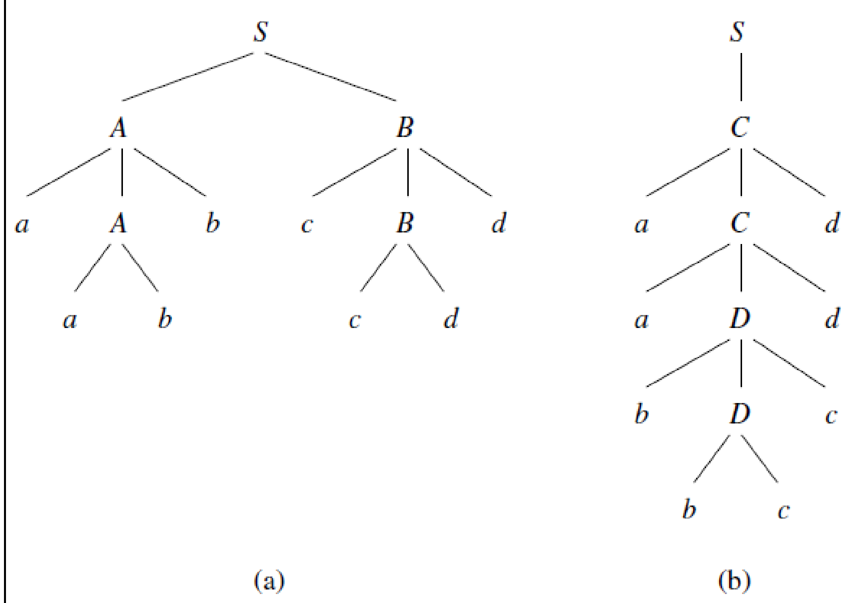


Figure 5.23: Two parse trees for *aabbccdd*

How to understand that every CFG is ambiguous.

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

That is, L consists of strings in $\mathbf{a}^+\mathbf{b}^+\mathbf{c}^+\mathbf{d}^+$ such that either:

1. There are as many a's as b's and as many c's as d's, or
2. There are as many a's as d's and as many b's as c's.

- Proof is complicated.
- But the essence is, the grammar has two parts one generating strings in each of the above union.
- There are some strings that are common between these two parts. These can be generated from two ways.

* **Exercise 5.4.1:** Consider the grammar

$$S \to aS \mid aSbS \mid \epsilon$$

This grammar is ambiguous. Show in particular that the string aab has two:

a) Parse trees.

b) Leftmost derivations.

c) Rightmost derivations.