Trees, Derivations and Ambiguity
A grammar

\[
\begin{align*}
E & \rightarrow P + E \\
E & \rightarrow P * E \\
P & \rightarrow (E) \\
P & \rightarrow a 
\end{align*}
\]

A tree

\[
\begin{align*}
E & \downarrow \\
P & + E \\
P & + P \\
P & + a \\
P & \rightarrow a + a 
\end{align*}
\]

3 derivations correspond to same tree (same rules being used in the same places, just written in different orders in the linear derivation)

1) \( E \rightarrow P + E \rightarrow a + E \rightarrow a + P \rightarrow a + a \)

2) \( E \rightarrow P + E \rightarrow P + P \rightarrow a + P \rightarrow a + a \)

3) \( E \rightarrow P + E \rightarrow P + P \rightarrow P + a \rightarrow a + a \)

But only one leftmost derivation corresponds to it (and vice versa). (see HW#7 for more)
Another grammar for the same language:

\[ E \rightarrow E+E \mid E^*E \mid (E) \mid a \]

This grammar is ambiguous: there is a string in \( L(G) \) with two different parse trees, or, equivalently, with 2 different leftmost derivations. Note the pragmatic difference: in general, \((a+a)^*a \neq a+(a^*a)\); which is right?
\[ E \rightarrow E + E \mid E \ast E \mid a \]

**Figure 2.6**
The two parse trees for the string \( a + a \ast a \) in grammar \( G_5 \)

*Leftmost derivation*

\[
E \Rightarrow_{L} E + E \Rightarrow_{L} E + E \ast E \Rightarrow_{L} a + E \ast E \\
\Rightarrow_{L} a + a \ast E \\
\Rightarrow_{L} a + a \ast a
\]
The “E, P” grammar again

This grammar is unambiguous.
(Why? Very informally, the 3 E rules generate $P((\text{'+'} \cup \text{'*'} \text{'*'}))P)^*$ and only via a parse tree that “hangs to the right”, as shown.)

But it has another undesirable feature: Parse tree structure does not reflect the usual precedence of * over +. E.g., tree at lower right suggests “$a * a + a == a * (a + a)$”
EXAMPLE 2.4

Consider grammar $G_4 = (V, \Sigma, R, \langle EXPR \rangle)$.

$V$ is $\{\langle EXPR \rangle, \langle TERM \rangle, \langle FACTOR \rangle\}$ and $\Sigma$ is $\{a, +, x, (, )\}$. The rules are

\[
\begin{align*}
\langle EXPR \rangle & \rightarrow \langle EXPR \rangle + \langle TERM \rangle \mid \langle TERM \rangle \\
\langle TERM \rangle & \rightarrow \langle TERM \rangle \times \langle FACTOR \rangle \mid \langle FACTOR \rangle \\
\langle FACTOR \rangle & \rightarrow ( \langle EXPR \rangle ) \mid a
\end{align*}
\]

The two strings $a + a \times a$ and $(a + a) \times a$ can be generated with grammar $G_4$. The parse trees are shown in the following figure.

A more complex grammar, again the same language. This one is unambiguous and its parse trees reflect usual precedence/associativity of plus and times.
Can we always tweak the grammar to make it unambiguous?

No! This language is a CFL; see grammar at left. Easy to see this G is ambiguous. Hard to prove, but true, that every G for this L is also ambiguous. Hopefully this is fairly intuitive—strings of the form $a^n b^n c^n$ can come from the $i=j$ or $j=k$ path.

$G$ is ambiguous
$L$ is inherently ambiguous, meaning every $G$ for $L$ is ambiguous
Some closure results for CFLs
Theorem

\( \text*{CFL's are closed under } \\
U, \cdot, \ast \\
\text{Conv.}
\text{all regular languages are CFL's.}
\text{Ref.}
\text{Give CFL's for } \emptyset, \{a\}, \{a, b\}, \{a \text{ or } b\}, \text{ and } a \{a, b\} \ast 1 \\
\text{CFL}
Concept

$G_c = (V_c, \Sigma, R_c, S_c)$

be 2 CFGs

with $V_1 \cap A_2 = \emptyset$

let $S \notin V_1 \cup V_2$

Build new grammar

$G = (V, \Sigma, R, S)$

$V = V_1 \cup V_2 \cup \{S\}$

$R = R_1 \cup R_2 \cup \{S \rightarrow S_1, S_2\}$

\[\forall x \in V_1 \forall y \in V_2\]

$S_1 \Rightarrow^* x \Rightarrow S_2 \Rightarrow^* y$

\[\therefore S \Rightarrow^* S_1, S_2 \Rightarrow^* x, S_2 \Rightarrow^* y\]

\[\therefore S \Rightarrow^* x, y \in L(G)\]