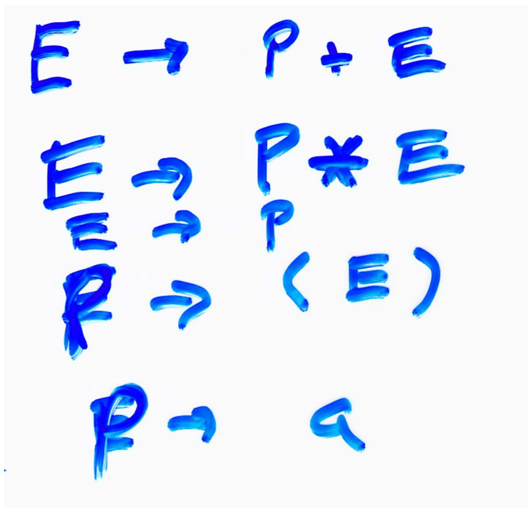
Trees, Derivations and Ambiguity

A grammar



A tree

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 => P+E => a+E => a+P => a+a

2) E => P+E => P+P => a+P => a+a

3) E => P+E => P+P => P+a => 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 → E+E | E*E | (E) | a

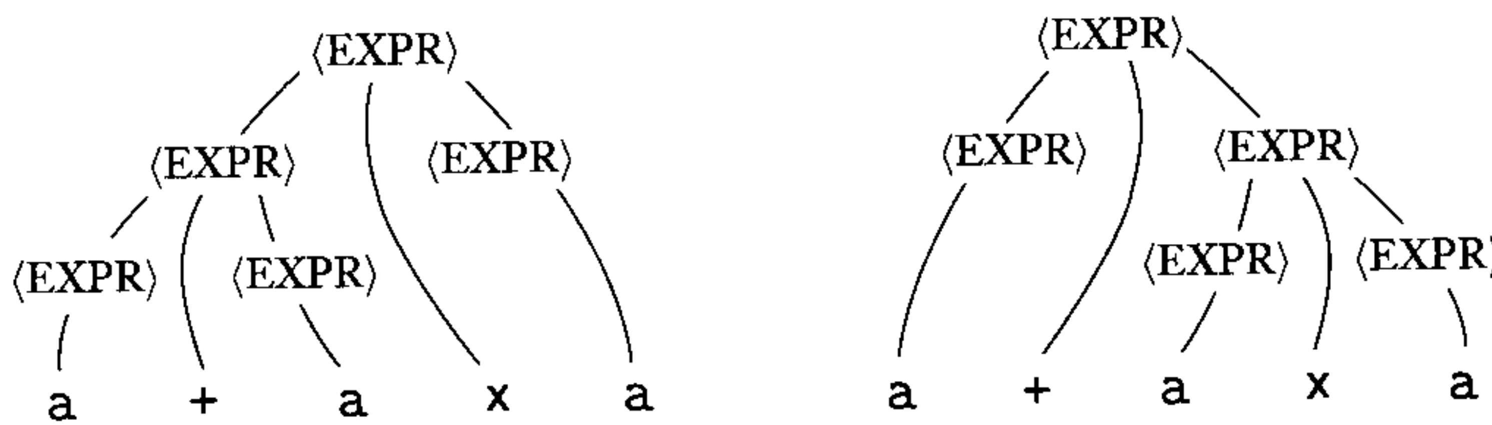


FIGURE 2.6
The two parse trees for the string a+a×a in grammar G_5

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 != a+(a*a); which is right?

$E \to E+E \mid E*E \mid a$

ambiguous

$E \Rightarrow E+E$
$\Rightarrow a+E$
$\Rightarrow a+E*E$
$\Rightarrow a+a*E$
$\Rightarrow a+a*a$

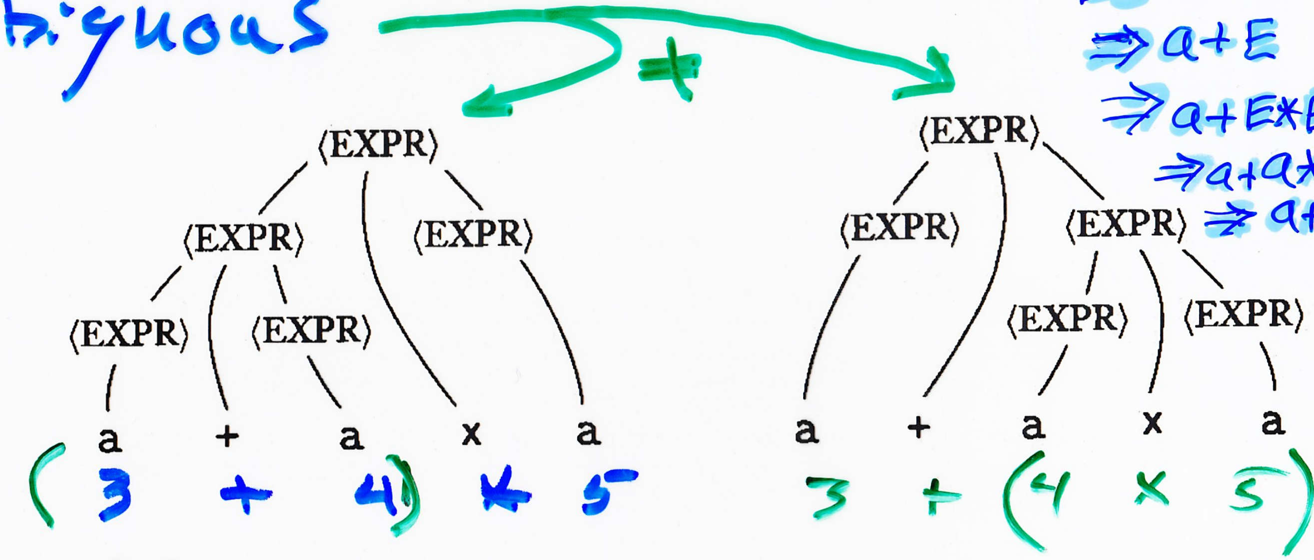


FIGURE 2.6

The two parse trees for the string a+a×a in grammar G_5

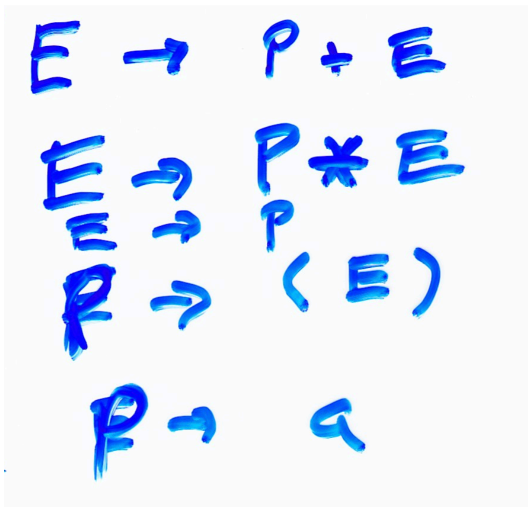
Leftmost deriv

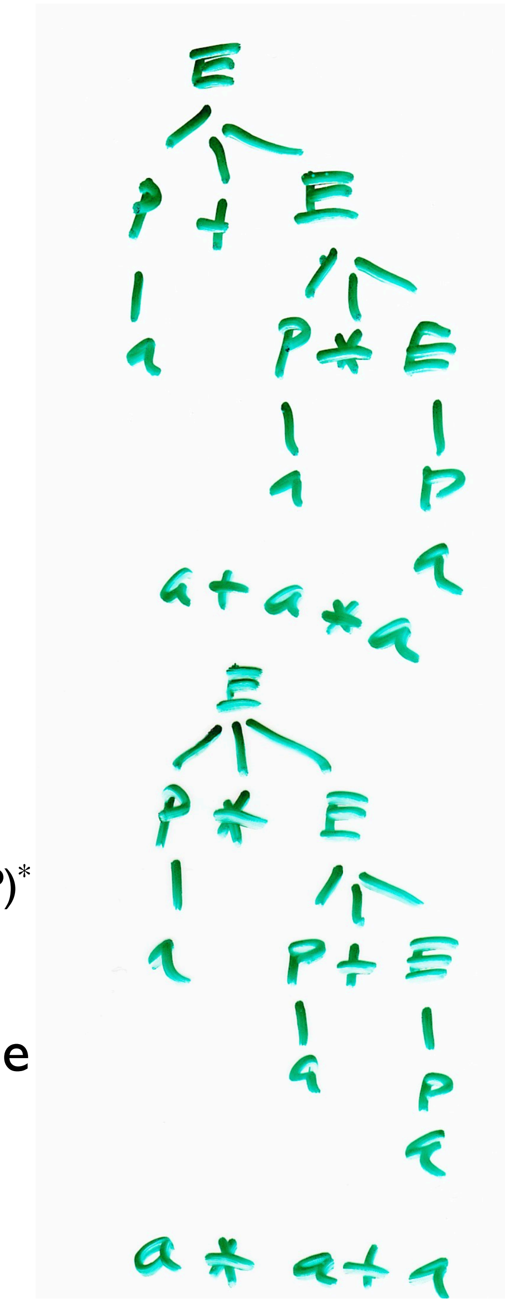
$E \Rightarrow_L E*E \Rightarrow_L E+E*E \Rightarrow_L a+E*E$
$\Rightarrow_L a+a*E$
$\Rightarrow_L a+a*a$

non-leftmost deriv: $E*E \Rightarrow E*a \Rightarrow E+E*a \Rightarrow E+a*a$

20-1

The "E, P" grammar again





This grammar is *un*ambiguous.

(Why? Very informally, the 3 E rules generate P(('+'∪'*')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)"

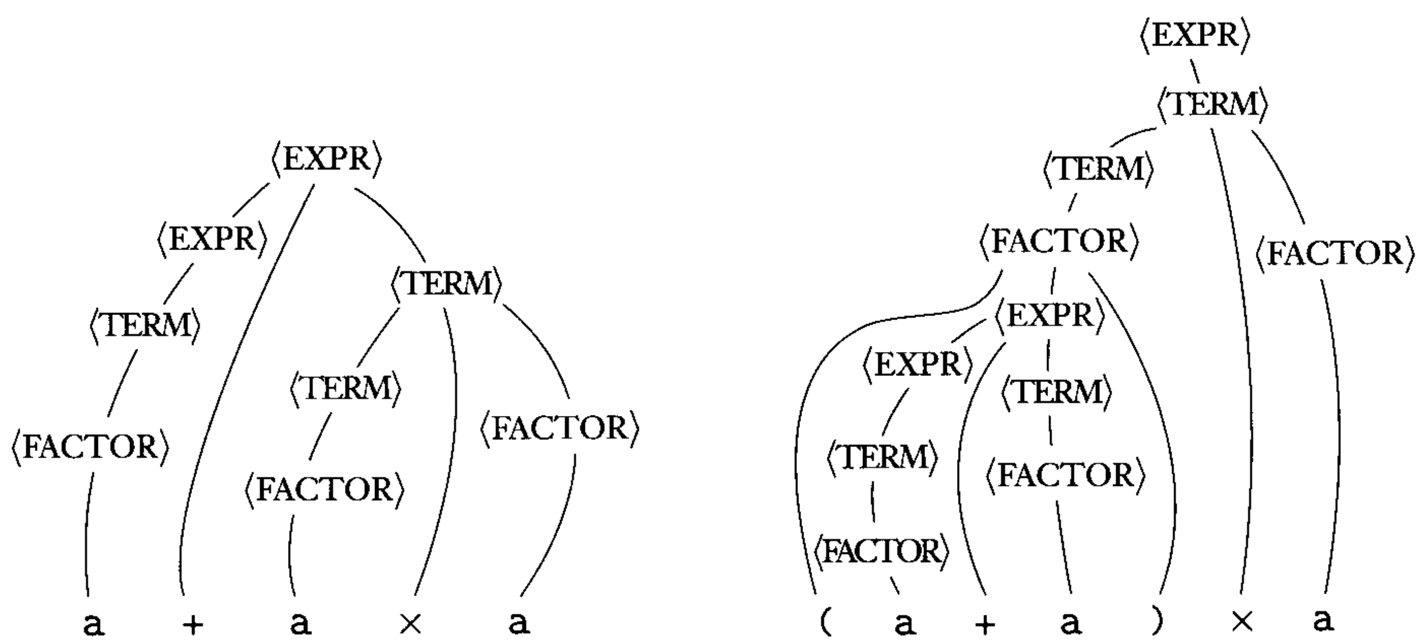
EXAMPLE 2.4

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

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

$$\begin{aligned}
\langle \text{EXPR} \rangle &\to \langle \text{EXPR} \rangle \texttt{+} \langle \text{TERM} \rangle \mid \langle \text{TERM} \rangle \\
\langle \text{TERM} \rangle &\to \langle \text{TERM} \rangle \texttt{x} \langle \text{FACTOR} \rangle \mid \langle \text{FACTOR} \rangle \\
\langle \text{FACTOR} \rangle &\to \texttt{(} \langle \text{EXPR} \rangle \texttt{)} \mid \texttt{a}
\end{aligned}$$

The two strings a+axa and (a+a)xa 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.

$$L = \{ a^i b^j c^k \mid i=j \text{ or } j=k \}$$

$S \to AC \mid DB$

$A \to aAb \mid \varepsilon$

$C \to cC \mid \varepsilon$

$D \to aD \mid \varepsilon$

$B \to bBc \mid \varepsilon$

$a^{10} b^{10} c^{22}$

$a^{10} b^{22} c^{22}$

$a^{10} b^{10} c^{10}$

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

CFL's are closed under

$\cup$, $\circ$, $*$

Corr.

all regular languages are CFL's.

Pf:

Give CFL's for $\emptyset$, $\{\varepsilon\}$, $\{a\}$ for each

$a \in \Sigma$

Concat

$G_i = (V_i, \Sigma, R_i, S_i)$

be 2 CFG's

with $V_1 \cap V_2 = \phi$

let $S \notin V_1 \cup V_2$

Build new grammar

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

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

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

$\forall x \in L_1 \; \forall y \in L_2$

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

$\therefore \; S \underset{L}{\Rightarrow} S_1 S_2 \underset{L}{\Rightarrow^*} x S_2 \underset{L}{\Rightarrow^*} x y$

$\therefore \; L_1 \cdot L_2 \subseteq L(G)$