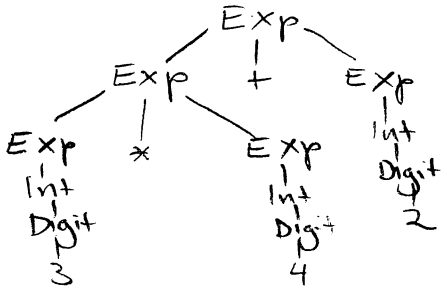
CFG Ambiguity [H-M-U 5.4]

Example:

Integer → Digit Integer | Digit
Digit → 0 | 1 | 2 | ⋯ | 9
Exp(ression) → Exp + Exp | Integer | Exp * Exp



Thus $Exp \overset{*}{\Rightarrow} 3*4+2$
ie., $3*4+2 \in L(Exp)$

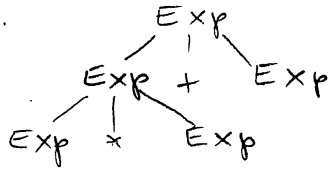
Observe: The same string can have multiple derivations, eg.

leftmost → $Exp \Rightarrow Exp + Exp \Rightarrow Exp * Exp + Exp \overset{*}{\Rightarrow} 3 * Exp + Exp \overset{*}{\Rightarrow} 3*4+Exp \overset{*}{\Rightarrow} 3*4+5$, and

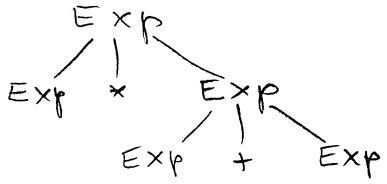
rightmost → $Exp \Rightarrow Exp + Exp \overset{*}{\Rightarrow} Exp + 2 \Rightarrow Exp * Exp + 2 \overset{*}{\Rightarrow} Exp * 4 + 2 \overset{*}{\Rightarrow} 3*4+2$

This is okay. It is usually not okay if there are multiple parse trees yielding the same string.

Eg.



versus



same yield, Exp * Exp + Exp, but very different meanings!

Def: A CFG is unambiguous if every terminal string has at most one parse tree.

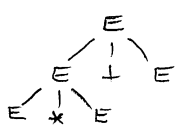
Equivalently, unambiguous ⇔ every terminal string has at most one leftmost derivation

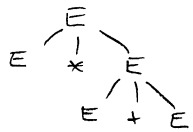
ambiguous ⇔ some terminal string has two or more parse trees (or leftmost derivations)

Removing ambiguity (leaving $L(G)$ the same)

Idea 1 Operator precedence

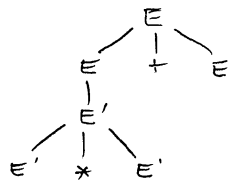
G: $E \longrightarrow \text{Num}(be-) \mid E+E \mid E*E \mid (E)$





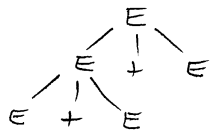
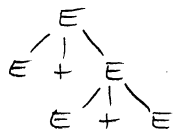
G': $E \longrightarrow E+E \mid E'$

$E' \longrightarrow \text{Num} \mid E'*E' \mid (E)$



"+" should have strictly lower precedence, ie. be higher in the parse tree, than "*"

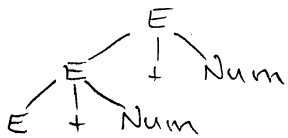
- but G' is still ambiguous!

$E+E+E$ has parse trees  and 

Idea 2 Force left associativity

H: $E \longrightarrow \text{Num} \mid E+(E)$ — we don't want to allow this E to be expanded further

H': $E \longrightarrow \text{Num} \mid E+\text{Num}$



G'': $E \longrightarrow E+E' \mid E'$

$E' \longrightarrow E'*E'' \mid E''$

$E'' \longrightarrow (E) \mid \text{Num}$

Idea: E can be split into a "+" but E' can only expand as a "*" or (E) and E'' can only be (E) or Number.

Note: Yacc/bison can automatically deal with precedence/associativity

Inherent ambiguity [H-M-U Ch. 5.4.4]

- $L = \{a^m b^m c^n d^n \mid m,n \geq 0\} \cup \{a^m b^n c^n d^m \mid m,n \geq 0\}$ is a context-free language (CFL)

Theorem: Every grammar G for L is ambiguous!

- essentially strings like $a^m b^m c^m d^m$ must have two parse trees, for m large enough.

Simplifying the presentation of Context-free Grammars

① Eliminate "useless" symbols
② Convert to "Chomsky normal form" (CNF)

Def: A variable or terminal X is **useful** if
- (a) X derives a terminal string, **and**
- (b) X is reachable from the start, i.e., $S \overset{*}{\Rightarrow} \alpha X \beta$

Eliminating useless symbols

(A) Eliminate symbols that cannot generate a terminal string
(B) Eliminate symbols that are not reachable

(A) **Example** G: $S \to AB \mid C$
$A \to 0B \mid C$
$B \to 1 \mid A0$
$C \to AC \mid C1$

G': $S \to AB$
$A \to 0B$, $B \to 1 \mid A0$

	symbols					
round	0	1	A	B	C	S
	✓	✓				
2	✓	✓		✓		
3	✓	✓	✓	✓		
4	✓	✓	✓	✓		✓
5	no changes ⇒ eliminate C					

Algorithm: Loop through the rules
- every terminal is generating
- if $A \to X_1 X_2 \cdots X_k$ and each X_j is generating, then so is A

updating the symbols in each round; until no changes are made in a loop.

(B) Use a graph traversal algorithm
- S is reachable
- If A is reachable and $A \to X_1 X_2 \cdots X_k$, then so are the X_j

	0	1	S	A	B
round 1			✓		
2			✓	✓	✓
3	✓	✓	✓	✓	✓

Important: Remember to eliminate non-generating symbols first.

Example: $S \to AB \mid a$
$A \to b$
should simplify to $S \to a$

In general, if $S \overset{*}{\Rightarrow} CD$ and C is not generating, then D is reachable, but this derivation can't go anywhere

Chomsky Normal Form (CNF)

Definition: A context-free grammar is in Chomsky normal form if all its productions are of the form

$$A \to BC \quad \text{or} \quad A \to a,$$

where A, B, C are variables and a is a terminal.

Theorem: For any context-free language L, there exists a context-free grammar G with

- $L(G) = L \setminus \{\varepsilon\}$
- G's symbols are all useful
- G is in Chomsky Normal Form.

Example:

$$E \to E + E \mid E * E \mid \text{Digit}$$
$$\text{Digit} \to 0 \mid 1 \mid 2 \mid \cdots \mid 9$$

in CNF:

$$E \to E\ \text{Eplus} \mid E\ \text{Etimes} \mid 0 \mid 1 \mid \cdots \mid 9$$
$$\text{Eplus} \to \text{PLUS}\ E$$
$$\text{Etimes} \to \text{TIMES}\ E$$
$$\text{PLUS} \to +$$
$$\text{TIMES} \to *$$

Why? - simpler because every production has one of two forms
- for proving the Pumping Lemma for context-free languages

Proof idea:

1. Eliminate useless symbols ✓
2. Eliminate ε productions $(L \to L \setminus \{\varepsilon\})$
   - find "nullable" symbols $A \overset{*}{\Rightarrow} \varepsilon$, and replace them
     if $A \to BC$ are B, C are nullable,
     then add $A \to BC \mid B \mid C$
     throw away $A \to \varepsilon$ productions
3. Eliminate unit productions
4. Convert to Chomsky Normal Form by splitting larger productions, adding new variables