CSCE 355
2/28/2024

Ambiguity
Parse trees

regex ⇒ CFG

Recall: Grammar for arith exprs:

$$E \to c \mid E+E \mid E-E \mid E*E \mid E/E \mid (E)$$

c * c + c

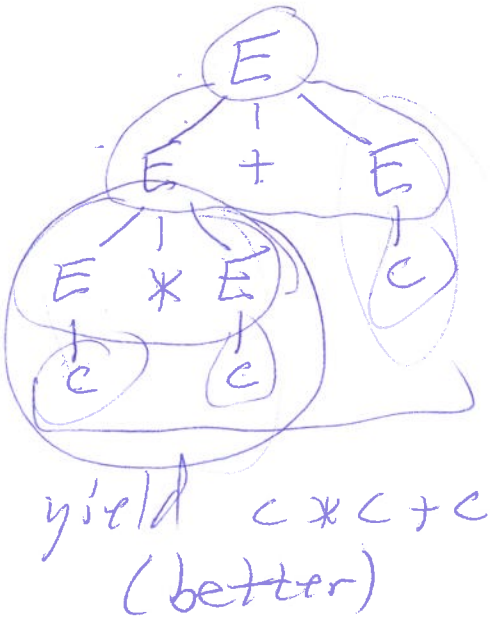
2 leftmost derivations:

$$E \Rightarrow E+E \Rightarrow E*E+E \Rightarrow \cdots \Rightarrow c*c+c$$

$$E \Rightarrow E*E \Rightarrow c*E \Rightarrow c*E+E \Rightarrow \cdots \Rightarrow c*c+c$$

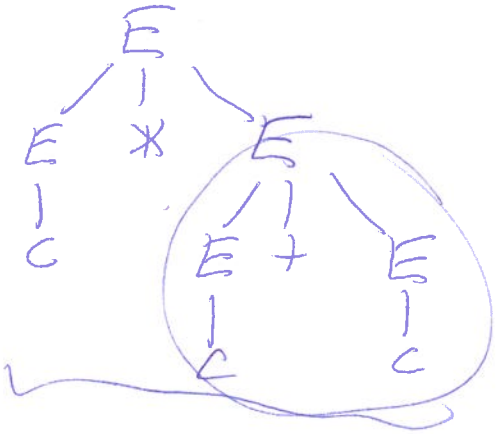
A grammar G is ambiguous if there exists a string in $L(G)$ with 2 or more leftmost derivations.

Parse trees: Above:



yield c * c + c

(better)



c * c + c

Def: Given a grammar G, a **parse tree** of G is a rooted, ordered tree whose nodes are labeled with grammar symbols of G (or with ε) such that every internal node is labeled by a variable and the children of the node, read left to right, form the body of a production with that variable as the head.

Ex: $A \to aBcA$ is a production of G,

could have  in a parse tree of G.

If $A \to \varepsilon$ then could have  in a parse tree.

Def: The **yield** of a parse tree is the concatenation of the leaves (in-order traversal, left to right).

Def: A parse tree of G is **complete** if
1) the root is labeled with the start symbol, and
2) every leaf is either a terminal or ε.

Prop: ~~Ther~~ For any CFG G and ~~$w \in L(G)$~~ w a string of terminals,

$w \in L(G)$ if and only if there is a complete derivation of w $\Leftrightarrow$ complete parse tree of G yielding w

Cor: $L(G) = \{w : w$ is the yield of some parse tree$\}$

Prop: There is a 1–1 correspondence between complete leftmost derivations and ~~pp~~ complete parse trees.

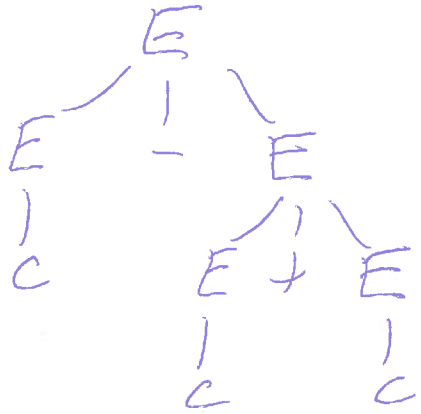
Cor: G is ambiguous if there are 2 or more complete parse trees of G with the same yield.

We (i.e., compilers) prefer unambiguous grammars, so that programs can be ~~passed~~ parsed in only one way.

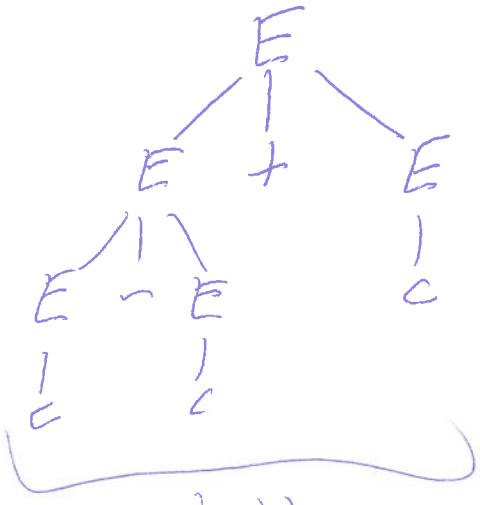
Want an unambiguous grammar for arith exprs.

c − c + c

2 parse trees:



or



better

Precedences:

lowest +, − : left associative

*, / : left associative

highest: ( ) : nonassociative

Unambiguous CFG:

$E \rightarrow E + T \mid E - T \mid T$

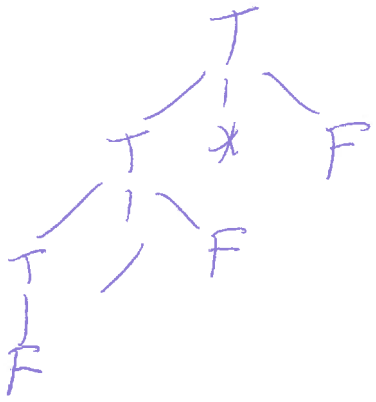
E = "expression"
T = "term"

eg.

yields T − T + T + T

$T \to T * F \mid T / F \mid F$ F="factor" (5

E.g



yields $F / F * F$

$F \to c \mid (E)$

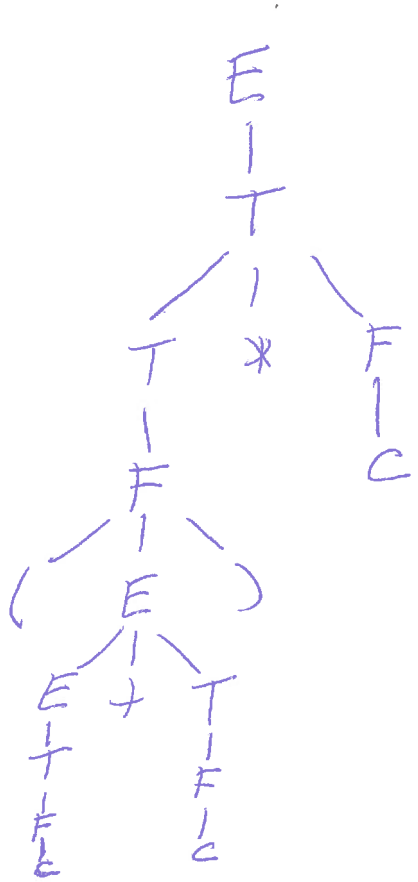
Summarize:

$E \to E + T \mid E - T \mid T$

$T \to T * F \mid T / F \mid F$

$F \to c \mid (E)$

Parse tree for $(c + c) * c$:



Another unambiguous grammar for arith exprs: (top-down)

$E \to T\ T'$

$T' \to +\ T\ T' \mid -\ T\ T' \mid \varepsilon$

$T \to F\ F'$

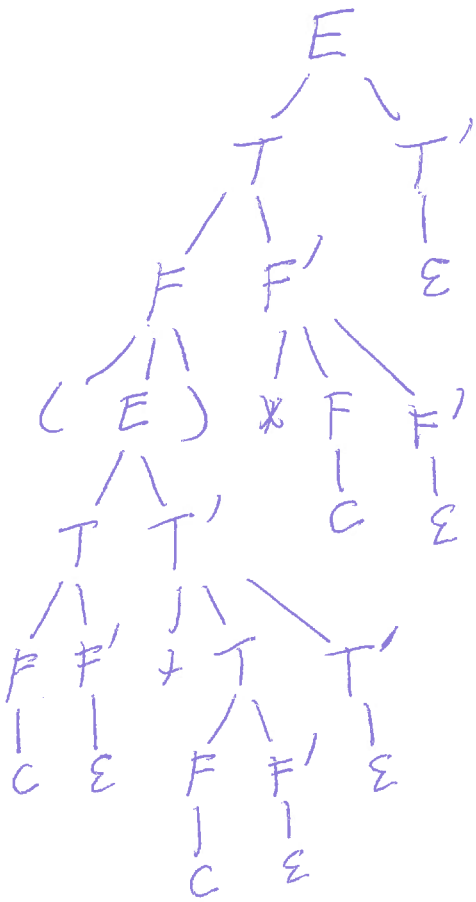
$F' \to *\ F\ F' \mid /\ F\ F' \mid \varepsilon$

$F \to c \mid (E)$

T' = "more terms"

F' = "more factors"

Parse tree yielding $(c+c)*c$



Converting a regex into an equivalent CFG

By induction/recursion on the syntax of the regex (over Σ)

r	equivalent G
$\emptyset$	$\langle \{S\}, \Sigma, S, \emptyset \rangle$ or $\{S \to S\}$
$(a \in \Sigma)$ a	$\langle \{S\}, \Sigma, S, \{S \to a\} \rangle$
$s + t$	have $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$ equivalent to s and $G_2 = \langle V_2, \Sigma, S_2, P_2 \rangle$ equiv to t WLOG, $V_1 \cap V_2 = \emptyset$ build $\langle \{S\} \cup V_1 \cup V_2, \Sigma, S, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\} \rangle$ (S new symbol not in V_1 or V_2)

L^+

G_1, G_2 as above.

$\langle \{S\} \cup V_1 \cup V_2, \Sigma, S, P_1 \cup P_2 \cup \{S \to S_1 S_2\} \rangle$

L^*

G_1 as before

$\langle \{S\} \cup V_1, \Sigma, P_1 \cup \{S \to S S_1 \mid \varepsilon\} \rangle$