① Grammar for arith. expressions [c = constant]

CSCE 355
3/2/2022

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

shorthand:

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

$c * c + c$

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

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

Def: A grammar (CFG) is ambiguous if the exists a string of terminals derivable (from start symbol) by ~~one~~ more than one leftmost derivation.

Parse trees

Def: A parse tree of a given grammar G is a rooted, ordered tree such that every internal node is labeled by a nonterminal of G and, for any such node with label A (say),

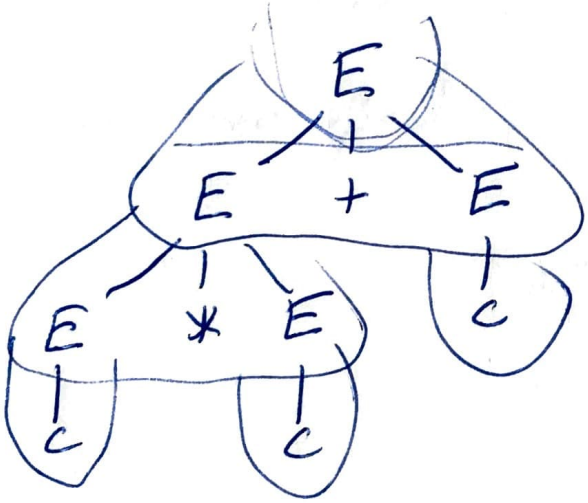
(2) the children of A, read left to right, form the (labels) body of a production whose head is A.

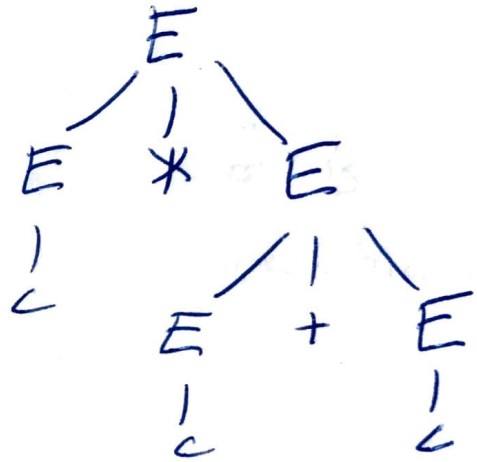
A parse tree is <u>complete</u> if

a) the root has the start symbol as its label,

b) every leaf is either a terminal or ε.

<u>Example</u>: Grammar above:





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

$\Rightarrow c*c+c$

Parse trees correspond one-to-one with leftmost derivations (which correspond one-to-one with rightmost derivations).

Def Given a parse tree, its yield is the string of terminals & nonterminals obtained by concatenating the leaf labels in left-to-right order (i.e., by an in-order traversal).

(3) **Prop:** Let G be CFG and w a string of terminals. $w \in L(G)$ iff there is a complete parse tree of G yielding w.

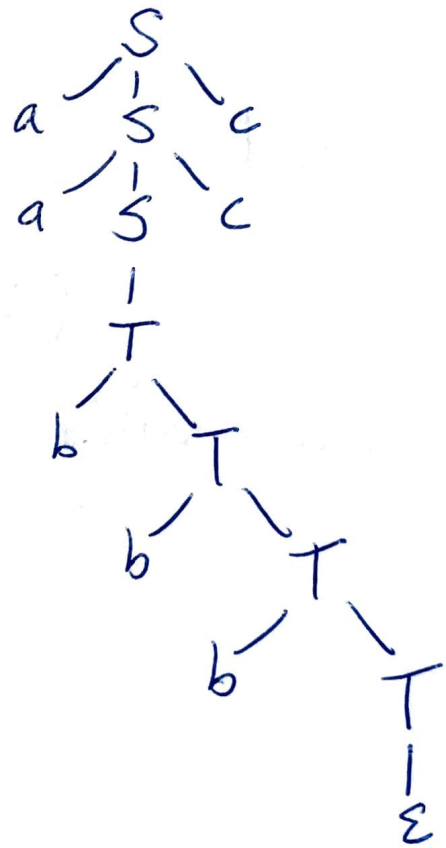
Ex:

$$\left.\begin{array}{l} S \to aSc \\ S \to T \\ T \to bT \\ T \to \varepsilon \end{array}\right\} G$$

$L(G) = \{a^n b^m c^n : m, n \geq 0\}$

Complete

Parse tree yielding $aabbbcc$



G is unambiguous

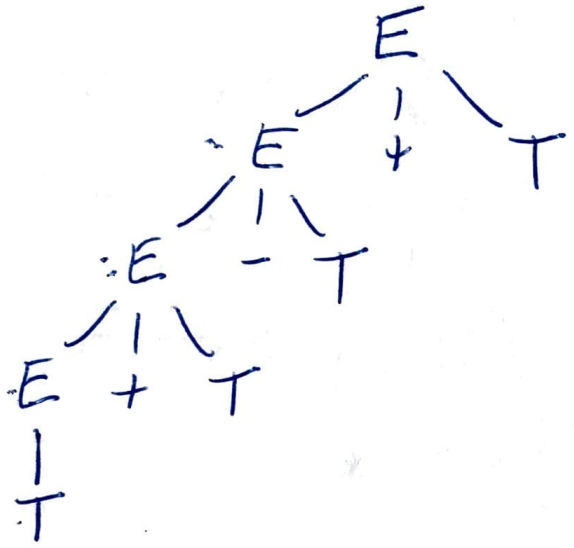
Unambiguous CFG for arith. exprs with constants (c), binary infix $+, -, *, /$ reflecting the precedence and associativity of the ops.

(4)

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

E = expr
T = term
F = factor

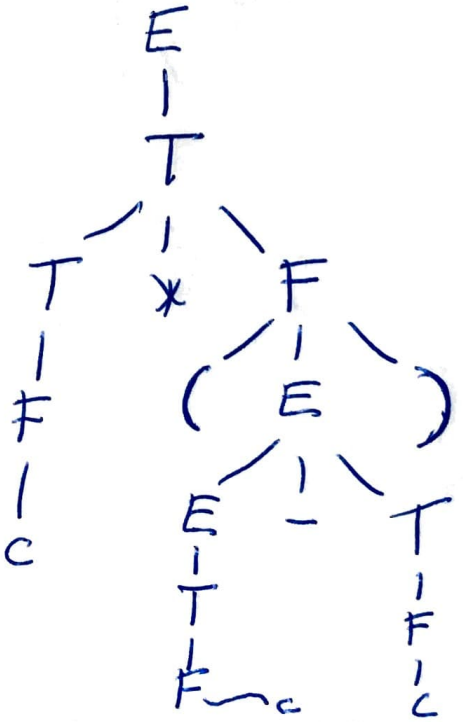
Parse tree for $T+T-T+T$:



$T \rightarrow T*F \mid T/F \mid F$

$F \rightarrow c \mid (E)$

Complete parse tree yielding $c*(c-c)$



(5) Another unambiguous grammar for the same language:

$E \rightarrow T\ T'$

$T' \rightarrow +\,T\ T' \mid -\,T\ T' \mid \varepsilon$

$T \rightarrow F\ F'$

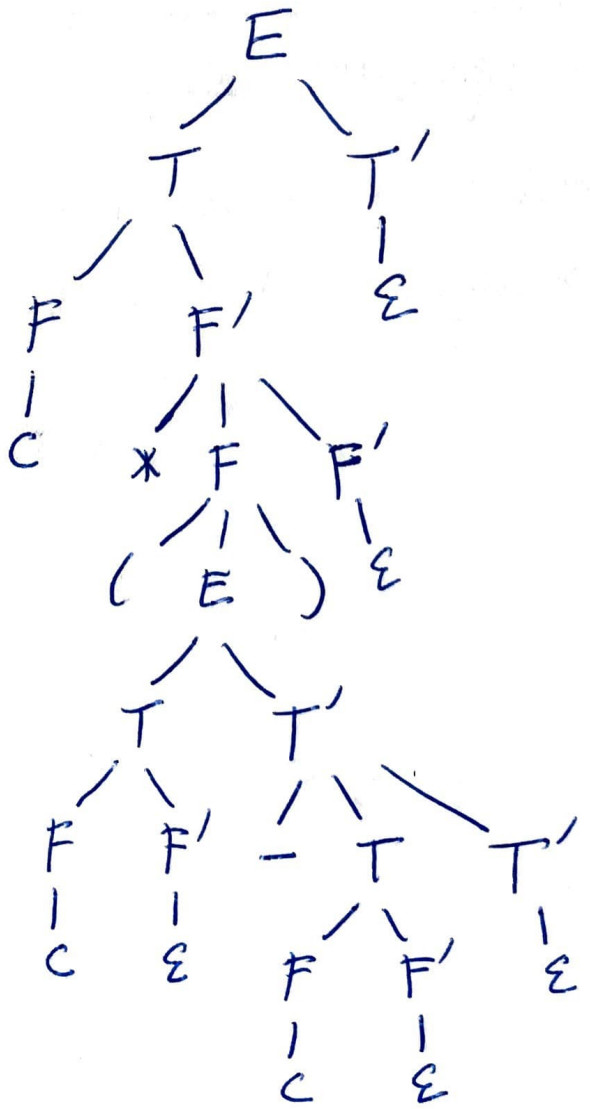
$F' \rightarrow *\,F\ F' \mid /\,F\ F' \mid \varepsilon$

$F \rightarrow c \mid (E)$

T = term

T' = more terms

Complete Parse tree for $c*(c-c)$:



$S \to E\ ;$

$S \to \{\ L\ \}$

$S \to$ ~~d~~ while $(E)\ S$

$S \to$ if $(E)\ S$

$S \to$ if $(E)\ S$ else S

$L \to L\ S$

$L \to \varepsilon$

S = statement

L = statement list

Thm: Every regular language is a context-free language.

Proof Idea: Convert any regex (over some Σ) to an equiv grammar with token alphabet Σ.

Atomic regexes:

$\emptyset \longrightarrow \langle \{S\}, \Sigma, S, \{S \to S\} \rangle$

$(a \in \Sigma)$ $a \longrightarrow \langle \{S\}, \Sigma, S, \{S \to a\} \rangle$

Nonatomic regexes: Assume CFG $G_1 = \langle V_1, \Sigma, S_1, P_1 \rangle$ for $L(s)$ and $G_2 = \langle V_2, \Sigma, S_2, P_2 \rangle$ for $L(t)$

(s) + (t):

(7) WLOG $V_1 \cap V_2 = \emptyset$. (By renaming variables if necessary)

$\Longrightarrow$ Grammar for $L_1 L_2$

New start symbol S and

production set ~~$P_1 \cup P_2 \cup S$~~

$P_1 \cup P_2 \cup \{S \to S_1 S_2\}$

so $\langle \{S\} \cup V_1 \cup V_2, \Sigma, S, \quad \rangle$

L^* G_1, G_2 as before

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

L^* G_1 as before

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