

**Concepts of Programming Languages, CSCI 305, Fall 2021**  
**Precedence & Associativity in Context-Free Grammars**  
**Introduction to Parsing, Oct. 4**

Precedence & Associativity in Context-Free Grammars

1. Recall the sample grammar:

$\text{assign} \rightarrow \text{id} = \text{expr}$

$\text{id} \rightarrow A \mid B \mid C \mid D$

$\text{expr} \rightarrow \text{expr} + \text{expr} \mid \text{expr} * \text{expr} \mid (\text{expr}) \mid \text{id}$

This grammar is ambiguous. Modify the grammar so that it is unambiguous and uses the usual rules. That is,  $*$  has higher precedence than  $+$ , and both  $+$  and  $*$  are left associative.

2. Problem 2.19, page 107, from the text:

Develop a grammar for the language consisting of all well-formed regular expressions. Arrange for all operators to be left-associative. Give Kleene closure the highest precedence and alternation the lowest precedence.

Definition of well-formed regular expressions:

Basis:

a for any letter a (upper or lower case)

$\epsilon$  (the empty string)

Induction, given that  $R_1$  and  $R_2$  are regular expression:

$R_1 \circ R_2$  (concatenation)

$R_1 \mid R_2$  (alternation)

$R_1^*$  (Kleene closure)

Use parenthesis to avoid ambiguity

Use your grammar to create a parse tree for the well-formed regular expression:

$(a^* \mid a \circ a \circ b) \circ b$