Concepts of Programming Languages, CSCI 305, Fall 2021
Homework #5, complete by Oct. 15

1. You have already done part (a) of exercise 2.9 from the text (page 107). Do parts (b) and (c).

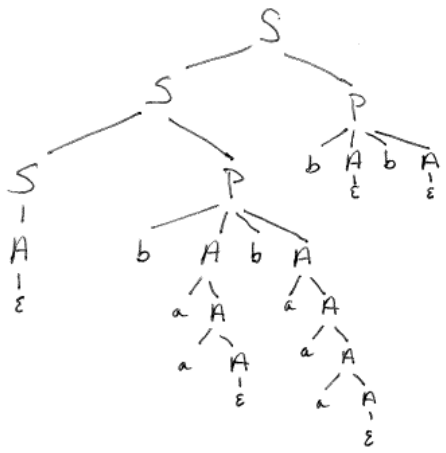
    a. Part a. Describe in English the language defined by the regular expression a* (ba*ba*)*. Your description should be a high-level characterization- one that would still make sense if we were using a different regular expression for the same language.

    Any string of a's and b's containing an even number, possibly 0, of b's.

    b. Write an unambiguous context-free grammar that generates the same language.

    S→ S P | A
    P→ b A b A
    A→ a A | ε

    c. Using your grammar for part (b), give a canonical (right-most) derivation of the string b a a b a a a b b .



S ⇒ SP ⇒ S b A b A ⇒ S b A b ⇒ S b b ⇒ S P b b ⇒ S b A b A b b ⇒
S b A b a A b b ⇒ S b A b a a A b b ⇒ S b A b a a a A b b ⇒
S b A b a a a b b ⇒ S b a A b a a a b b ⇒
S b a a A b a a a b b S ⇒ S b a a b a a a b b ⇒
A b a a b a a a b b ⇒ b a a b a a a b b

2. Do exercise 2.10 from the text.
   Give an example of a grammar that captures right associativity for an exponent operation (e.g. ** in Fortran).

   Example of right associativity:

   x **2 **3 = x ** (2 ** 3)

   To get this, make the parse tree expand to the right.

   Grammar:

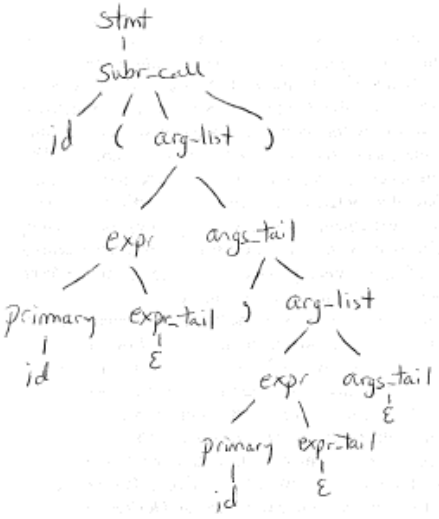
   expr → expr1 ** expr | expr1
   expr1 → num | id

3. Do exercise 2.13, parts (a) and (b)

Grammar:

1. stmt → assignment
2. stmt → subr_call
3. assignment → id := expr
4. subr_call → id (arg_list)
5. expr → primary expr_tail
6. expr → op expr
7. expr → ε
8. expr_tail → op expr
9. expr_tail → ε
10. primary → id
11. primary → subr_call
12. primary → (expr)
13. op → + | - | * | /
14. arg_list → expr args_tail
15. args_tail → , arg_list
16. args_tail → ε

a.) Construct a parse tree for the input string foo(a,b),



b.) Give a canonical (right-most) derivation of this same string.

stmt ⇒ subr_call ⇒ id (arg_list) ⇒ id (expr args_tail)
⇒ id (expr , arg_list) ⇒ id (expr , expr args_tail)
⇒ id (expr , expr) ⇒ id (expr , primary expr_tail)
⇒ id (expr , primary) ⇒ id (expr , id)
⇒ id (expr , id) ⇒ id (primary expr_tail , id)
⇒ id (primary , id) ⇒ id (id , id)

4. Give a context free grammar for the language on $\sum=\{a,b\}$ defined by $L=\{w \mid (n_a(w) > n_b(w)\}$.
The set of strings over the alphabet {a,b} with more a's than b's
$n_a(w)$ is the number of a's in the string w.
$n_b(w)$ is the number of b's in the string w.

$S \rightarrow EAE$
$A \rightarrow aA \mid a$
$E \rightarrow EaEbE \mid EbEaE \mid \varepsilon$

This is an example of an ambiguous grammar.

(The following is too flexible:
$S \rightarrow aSb \mid bSa \mid A \mid SS$
$A \rightarrow aA \mid a$
The SS provides needed flexibility, but S needs to get to ε.
Currently, can't derive the string baaab.)

5. Give a context-free grammar generating the language
$L = \{x_1\#x_2\#...\#x_k \mid k \geq 1$, each $x_i \in \{a,b\}^*$, and for some i and j, $x_i = x_j^R\}$

Hint - it is helpful to break the above problem into cases:
Case 1: x_i appears at the front of the string and x_j appears at the end of the string.
Case 2: x_i appears at the front, but x_j does not appear at the end.
Case 3: x_i does not appear at the front, but x_j appears at the end.
Case 4: x_i does not appear at the front, and x_j does not appear at the end of the string.

$S \rightarrow C_1 \mid C_2 \mid C_3 \mid C_4$

$C_1 \rightarrow aC_1a \mid bC_1b \mid \#X\# \mid \# \mid \varepsilon$
$X \rightarrow aX \mid bX \mid \#X \mid \varepsilon$

$C_2 \rightarrow U\#X$
$U \rightarrow aUa \mid bUb \mid \#X\# \mid \# \mid \varepsilon$

$C_3 \rightarrow X\#V$
$V \rightarrow aVa \mid bVb \mid \#X\# \mid \# \mid \varepsilon$

$C_4 \rightarrow X\#W\#X$
$W \rightarrow aWa \mid bWb \mid \#X\# \mid \# \mid \varepsilon$