Theory Homework 3

CS320 Concepts of Programming Languages

Due: November 1, 2022

Context-Free Grammars

We will use the following grammar in problems 1 through 3:

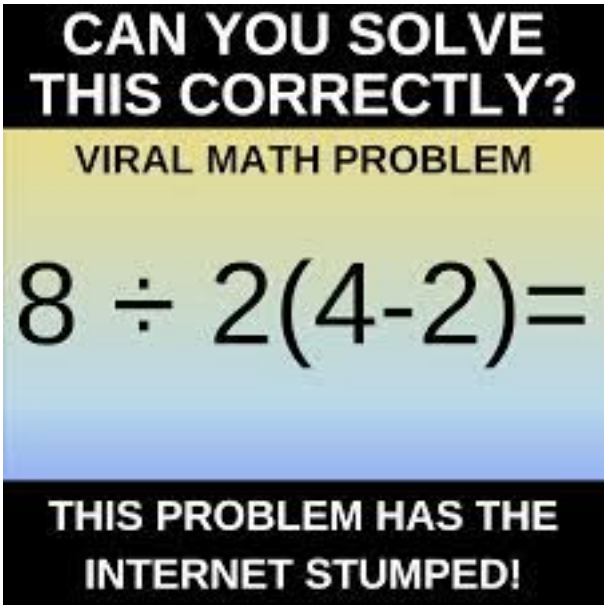
```
<sentence> ::= <noun-phrase><verb-phrase>
<noun-phrase> ::= <article><noun>
<article> ::= a | an | the
<noun> ::= man | shirt | mood | trade | book
<verb-phrase> ::= <verb> | <verb><noun-phrase>
<verb> ::= eats | trades | books | is | refuses
```

1. Derive 3 valid sentences from the grammar.

2. Draw a syntax tree for any of the sentences you created.

3. Is "man eats" a valid sentence? If so, draw a syntax tree. If not, explain why not.

In grade school we were introduced to an unambiguous grammar to interpret arithmetic expressions. There are multiple mnemonics to remember this, but a common one is PEMDAS. Unfortunately, the exact way PEMDAS works is a common confusion, which results in highly controversial discussions over images such as this:



However, if we review PEMDAS, we can make this expression unambiguous! PEMDAS stands for Parenthesis, Exponent, Multiplcation and Division, and Addition and Subtraction. This means that we calculate expressions in parentheses first, then in exponents, then we do multiplication or division based on order from left-to-right, and finally we do addition or subtraction based on order from left-to-right.

This is also described by the extended Backus-Naur form for arithmetic:

```
<expr> ::= <expr> <addop> <term> | <term>
<term> ::= <term> <mulop> <term> | <factor>
<factor> ::= <const> | ( <expr> )
<const> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
<addop> ::= + | -
<mulop> ::= * | /
```

Note: The starting symbol of the EBNF is <expr>. We sometimes write $*$ as $\times$ or by concatenating with an expression in parenthesis like $a(f)$. Similarly, we sometimes write / as $\div$. Also, since exponents aren't included in the EBNF for arithmetic given, we won't include them. Thus, we interpret the equation as $8 \div 2 \times (4 - 2)$.

4. This grammar is actually ambiguous! Prove that it is ambiguous by drawing two distinct parses for a single valid sentence. Hint: the outcomes based on the ambiguity in the grammar is the one the Internet usually argues between.

5. Make a change to the grammar to disambiguate it. Your newly modified grammar should accept the same sentences as the original. Explain your reasoning. (Hint: Very little needs to be changed.)

6. Use PEMDAS and either your fixed EBNF for arithmetic or the provided one to draw a syntax tree for the math problem in the picture above so that we can calculate the correct answer and save the Internet!

Regular Expressions and Regular Grammars

7. Show that the following grammar is ambiguous.

```
<S> ::= <A><B> | aa<B>
<A> ::= a | <A>a
<B> ::= b<T>
<T> ::= epsilon
```

8. Create a regular expression that describes the grammar in the previous problem.

9. Find a regular expression for $L = \{a^n b^m : n \geq 1, m \geq 1, nm \geq 3\}$. In words, this says that strings will be a non-empty string of a's and b's where there is at least one a, at least one b, and the product of the number of a's and b's is at least three.

10. Find a regular expression for $L = \{vw : v, w \in \{a, b\}^*, |v| = 2\}$. In words, this says that the strings will be a concatenation of two strings from $\{a, b\}^*$ (where $*$ is the Kleene star) where the left string has length exactly 2.