CS 463 - Homework 3

Defining Languages

Deadline: Thursday, February 15th 2024, 5pm

- Work must be uploaded to gradescope ***per-question***; text boxes and picture/pdf file uploads are ok.

---

1. [12pts] Consider the following language description: its sentences only contain balanced brackets of various kinds: `()`, `{}`, and `[]` all may be included. The empty string is also in this language. Whether they are one after the other, or nested, they must balance each open with a matching-style close afterwards. Some valid sentences (members of the language):

   `()[]{}` `(())))` `(([{}]))(())()()`

   Some invalid sentences (not members of the language):

   `[` `([)]` `)(` `(no)`

a. Write out the production rules for this language. It must not be ambiguous.

b. write a derivation for the sentence: `{[]}()`

---

2. [12pts] Given the following production rules:

```
S → S+N | N
N → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

   For each sentence, do two things: draw a parse tree, and write out a leftmost derivation. Don't skip any steps!

a. `2+3`
b. `5+8+9`

---

3. [12pts] Given these production rules:

```
S → S * S | N
N → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0
```

a. Draw two unique valid parse trees of the sentence `3*4*5`
b. Is this ambiguous or unambiguous? (Yes or No)

---

4. [10pts] Given the production rules below, is the plus operator (+) left-associative, right-associative, or neither? You need to provide a sample sentence and a parse tree of it that supports your answer. (hint: very short sentences should suffice).

```
S → S + T | T
T → 1 | 2 | 3
```

---

5. [16pts] Given the set of terminals: `terminals = {l, o, m, f, g}`

a. Write production rules for a language that accepts palindromes of any non-zero length.
b. Write a derivation for `lolol`

---

6. [10pts] consider a language where it is chosen that `or` has higher precedence than `and`. The terminals are:

```
terminals = { true, false, and, or}
```

a. Write out the producton rules. Requirement: don't allow any ambiguity.
b. Next, draw a parse tree and derivation for `false or true and true or false` . (Hint: remember, there are no parenthesis terminals! They should not show up here.)

---

7. [16pts] The following production rules are ambiguous:

```
Expr  →  Expr / Expr | Expr * Expr | ( Expr ) | Num
Num   →  0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

a. Prove it is ambiguous by providing a specific example sentence that has two different parse trees for the sentence, and draw out both parse trees.
   - (Hint: there are relatively short sentences that do this).
   - Make sure you don't skip over any non-terminals in your parse trees! (No need to write out the derivation for this question).

b. Rewrite the production rules so that it:
   - accepts the exact same sentences as before (no more, no less)
   - is unambiguous
   - is always right-associative (an unusual choice for multiplication/division, but our languages can do whatever we choose).
   - doesn't introduce any extra precedence between multiplication and division. (PEMDAS is really P-E-MD-AS. Or, "B-E-DM-AS" in England: "brackets, exponents, [etc.]")

---

8. [12pts] Answer each (yes/no):

a. in an unambiguous language, can operators with the same precedence have the same associativity?

b. in an unambiguous language, can operators with different precedence have the same associativity?

c. can production rules be both ambiguous and unambiguous?

d. can production rules contain both left-associative and right-associative operators?