CSCI 3155: Homework Assignment 2

Due Thursday, September 10, 2009

For this assignment, each student will turn in a write-up. **You need to have your own write-up.**

You are welcome and encouraged to collaborate and discuss these questions in groups, particularly now that you at least know someone from Homework Assignment 1. Just be sure to acknowledge those with which you discussed.

This assignment refers to the grammar for the MYSTERY programming language:

http://www.cs.colorado.edu/~diwan/pldetective/pldsyntax.htm

**Exercise 1: Bookkeeping.** Indicate in a sentence or two how much time you spent on this homework, how difficult you found it subjectively, and what you found to be the hardest part. Also, indicate one thing you like about the class so far and one thing you would change about it. Any non-empty answer will receive full credit.

**Exercise 2: Skill 3.2.** Prove that the following grammar is ambiguous:

\[
\begin{align*}
\langle S \rangle & \rightarrow \langle A \rangle \\
\langle A \rangle & \rightarrow \langle A \rangle + \langle A \rangle \mid \langle id \rangle \\
\langle id \rangle & \rightarrow a \mid b \mid c
\end{align*}
\]

(Sebesta, Chapter 3, Problem 8)

**Exercise 3: Skill 2.2.** Describe the language defined by the following grammar:

\[
\begin{align*}
\langle S \rangle & \rightarrow \langle A \rangle \mid \langle B \rangle \mid \langle C \rangle \\
\langle A \rangle & \rightarrow a \langle A \rangle \mid a \\
\langle B \rangle & \rightarrow b \langle B \rangle \mid b \\
\langle C \rangle & \rightarrow c \langle C \rangle \mid c
\end{align*}
\]

(Sebesta, Chapter 3, Problem 10)
Exercise 4:  **Skill 2.3.** Consider the following grammar:

\[
\begin{align*}
\langle S \rangle & \rightarrow \langle A \rangle \ a \ \langle B \rangle \ b \\
\langle A \rangle & \rightarrow \langle A \rangle \ b \mid b \\
\langle B \rangle & \rightarrow \ a \ \langle B \rangle \mid a
\end{align*}
\]

Which of the following sentences are in the language generated by this grammar? For the sentences that are described by this grammar, demonstrate that they are by giving derivations.

1. baab
2. bbbab
3. bbaaaaa
4. bbaab

(Sebesta, Chapter 3, Problem 11)

Exercise 5:  **Skill 2.3.** Consider the following grammar:

\[
\begin{align*}
\langle S \rangle & \rightarrow \ a \ \langle S \rangle \ c \ \langle B \rangle \mid \langle A \rangle \mid b \\
\langle A \rangle & \rightarrow \ c \ \langle A \rangle \mid c \\
\langle B \rangle & \rightarrow \ d \mid \langle A \rangle
\end{align*}
\]

Which of the following sentences are in the language generated by this grammar? For the sentences that are described by this grammar, demonstrate that they are by giving parse trees.

1. abcd
2. acccbd
3. accbcc
4. acd
5. accc

(Sebesta, Chapter 3, Problem 12)

Exercise 6:  Consider the following two grammars for \(\langle Expr \rangle\). Note that the first grammar is part of \(\langle Expr \rangle\)'s grammar in MYSTERY. In both grammars, \(\langle Operator \rangle\) and \(\langle Operand \rangle\) are the same as in MYSTERY. You need not look at the definition of \(\langle Operand \rangle\) to answer this question.

\[
\begin{align*}
\langle Expr \rangle & \rightarrow \langle Operand \rangle \mid \langle Expr \rangle \ (Operator) \ \langle Operand \rangle \\
\langle Expr \rangle & \rightarrow \langle Operand \rangle \ \langle ExprSuffix \rangle \\
\langle ExprSuffix \rangle & \rightarrow \langle Operator \rangle \ \langle Operand \rangle \ \langle ExprSuffix \rangle \mid \varepsilon
\end{align*}
\]
1. **Skill 2.2.** Intuitively describe the expressions generated by the two grammars.

2. **Skill 2.4.** Do these grammars generate the same or different expressions? Explain.

**Exercise 7:** Consider the following two grammars for \(\langle\text{Operand}\rangle\). The first grammar is part of \(\langle\text{Operand}\rangle\)'s grammar in MYSTERY. In both grammars, \(\langle\text{Expr}\rangle\) is the same as in MYSTERY. You need not look at the definition of \(\langle\text{Expr}\rangle\) to answer this question.

\[
\langle\text{Operand}\rangle \rightarrow \text{Number} \mid \text{id} \mid \langle\text{Operand}\rangle[\langle\text{Expr}\rangle]
\]

\[
\langle\text{Operand}\rangle \rightarrow \text{Number} \langle\text{OperandSuffix}\rangle \mid \text{id} \langle\text{OperandSuffix}\rangle
\]

\[
\langle\text{OperandSuffix}\rangle \rightarrow [\langle\text{Expr}\rangle] \langle\text{OperandSuffix}\rangle \mid \epsilon
\]

The square brackets ([ . . . ]) are for array references.

1. **Skill 2.2.** Intuitively describe the expressions generated by the two grammars.

2. **Skill 2.4.** Do these grammars generate the same or different expressions? Explain.

**Exercise 8:** **Skill 3.4.** Write a program that demonstrates whether + has higher precedence than * or vice versa in the C++ programming language. Make sure that you are checking for precedence in your program and not for left or right associativity. Submit your program, the output produced by running your program, and an argument indicating how you arrived at the relative precedence of + and * based on the output.

**Exercise 9:** **Skill 2.5.** Give a BNF grammar for floating point numbers that are made up of a fraction (e.g., 5.6 or 3.123 or -2.5) followed by an optional exponent (e.g., E10 or E-10). The exponent, if it exists, is the letter “E” followed by an integer. For example, the following are floating point numbers: 3.5E3, 3.123E30, -2.5E2, -2.5E-2, 3.5. The following are not examples of floating point numbers: 3.E3, E3, 3.0E4.5.

For this exercise (which may be different than in class), let us assume that the tokens are individual characters. Specifically, the alphabet is as follows:

\[\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \text{E, -}, .\}\]

Your grammar should be complete (i.e., it should not count on a non-terminal that it does not itself define).
Exercise 10:  Skill 3.3. Looking at the syntax of MYSTERY determine if the + operator is left or right associative. Also, determine if + has higher, lower, or the same precedence as >. Explain the reasoning behind your answer.

Exercise 11:  Skill 3.1. From the syntax of MYSTERY, we see that a MYSTERY program is a block. A MYSTERY program starts by executing this block.

MYSTERY is designed to be as small as possible, yet large enough to illustrate many of the concepts that we will cover in this class. Thus, MYSTERY is missing many features that you may be used to in your favorite programming language. On the other hand, you will find later this semester that MYSTERY has some features that are more powerful than features in your favorite programming language.

1. Write a program that prints out 1 through 100 (inclusive). You should test your program using this form:

   http://machine.cs.colorado.edu/~diwan/pldsyntax.htm

   Use your account name as the “User id”, enter your code in the ”Input” space, and click “Submit”. This will compile and run your program and present any errors or output in a new browser window. You may use as many attempts as you wish. As an additional challenge to yourself, try to do this in as few attempts as you can. Be sure to brag about how many tries it took in your submission!

2. Write a program that has two procedures, sum and main. The procedure sum takes two integers as arguments and returns their sum. The procedure main calls sum with two arguments and prints out the result. You should use the same form to test your program:

   http://machine.cs.colorado.edu/~diwan/pldsyntax.htm