**CS332 Mod01 HW2**

1. (10 pts) Create a grammar, G, for an *expression language* that allows creation of mathematical expressions using the plus (+) and multiplication (\*) operators. Let ∑ = {x, y, z, int, +, \*, (, ) } where *int* is a stand-in for any integer value (this is commonly done). The empty string, λ, is not in L. Valid strings in L are
   * single variables or integers: x, y, z, int
   * simple mathematical expressions: x + int, x + y, x + y + int\*z
   * mathematical expressions using parenthesis to show precedence: (x + y) \* int, (int \* y + int) \* (x + z)
   * parenthesis may be nested, with no limit, but must be properly nested:  int \* ( x + (y \* int + (y + z) ) ).

1. S 🡪 A

2, 3, 4, 5. A 🡪 x | y | z | int

6, 7, 8, 9. A 🡪 x B A | y B A | z B A | int B A

10, 11, 12, 13. A 🡪 (x B A) | (y B A) | (z B A) | (int B A)

14, 15. B 🡪 + | \*

NOTE: I GOT THIS WRONG BECAUSE IT DOESN’T ALLOW FOR ALL STRINGS IN THE LANGUAGE – EXAMPLE: (x + y) \* z

1. (5 pts) Use your grammar, G, to derive the string u = int.

u = int

|  |  |
| --- | --- |
| S |  |
| A | 1 |
| int | 5 |

1. (5 pts) Use your grammar G to derive the string v = x + int \* (y \*z + int).

v = x + int \* (y \* z + int)

|  |  |
| --- | --- |
| S |  |
| A | 1 |
| xBA | 6 |
| x + A | 14 |
| x + int B A | 9 |
| x + int \* A | 15 |
| x + int \* ( y B A) | 11 |
| x + int \* ( y \* A) | 15 |
| x + int \* ( y \* z B A) | 8 |
| x + int \* ( y \* z + A) | 14 |
| x + int \* ( y \* z + int) | 5 |

1. (5 pts) State where your grammar lies in the Chomsky Hierarchy with justification.
   1. This is a Type 2, context free, grammar. This is the case because there is only ever one non-terminal on the LHS at one time and the number of terminals in the RHS never decreases.
2. (5 pts) The MU puzzle is a famous problem created by Douglas Hofstader in *Godel, Escher, Bach*. It is presented in near-BNF form [Note 1], though he doesn't tell the reader that. The alphabet, ∑ = { M, I, U } (yes, the terminals are capital letters). The production rules use *x* and *y* to represent any string, and are:

* xI →→ xIU (you can add a U to any string ending in I)
* Mx →→ Mxx (if a string begins with M, you can duplicate the part of the string after the M)
* xIIIy →→ xUy (if you have three I's in a row, you can replace it with a U)
* xUUy →→ xy (if you have two U's in a row, you can eliminate them)

You don't have to solve the puzzle, but here it is: starting with MI, can you derive MU?

Here's the question -- where does this grammar fall in the Chomsky Hierarchy, and why?

Note 1: He's speaking to a general audience, so adjusts his language. He uses capitals for terminals, and lower case for generic strings. He requires that the puzzle start with MI, so does not need a start symbol.

* 1. It is unrestricted because the number of terminals in the RHS can decrease due to the 3rd rule and the only type in the Chomsky Hierarchy that accepts that is type 0, an unrestricted grammar.