**COMP 3002 Winter 2021 Assignment #6**

**Creating a Grammar Translator**

**Due: Tuesday Midnight March 9**

**Rerun the FSM builder from the last assignment to ensure complex operations following two operations are implemented…**

 parserFSMsForReduceAndMinus.txt (use this one first)

 scannerFSMsForReduceAndMinus.txt (use this one second)

**The new translator**

I have created a translator called “GrammarBuilder.st” which can create a grammar from a series of productions.. You will see that it contains a class method “promptForGrammar” to run it (where you will select one of the files from SampleGrammars. Start with toyLispGrammar until it works then go on to other examples. The last grammar to use should be toyScannerGrammar

But before you go too far, incorporate some of the code from the last assignment into this new translator. In particular, add the method processTypeNow: and the instance variable you added to make it work. Also, copy all your walk routines that build fsms.

You will note that I also added a class Grammar and a class Production to speed up the implementation. Feel free to change it.

**More details about this builder…**

It will need to run two passes. Pass1 to pick up all the nonterminals and then pass2 to build the grammars.

For pass1, you can add a new recursive routine, say walkTreePass1: which is used to recursively pick up the nonterminals and add them to the grammar (basically, looking only for leftPartWithLookahead: and leftPart: where the nonterminal is the left child; pick up the symbol, not the token).