Compiler Construction
Oblig 1: parsing

Investigate and characterize conflicts of the original grammar

The conflicts that occur in the original from my understanding seem to be shift/reduce problems in the way expressions are handled within the grammar. This is due to many of the expressions allowing more than one token/non-terminal to be reduced, a consequence of the grammar itself being ambiguous.

How many states do the 2 generated CUP grammars have?

The ambiguous grammar holds 174 states, meanwhile the unambiguous one generates 179 states. In order to bake in the precedence and associativity more states are needed, so that ambiguous grammar having more states is more

Discuss also whether the choice of the two grammars influences the generation of the AST. Is one of the two approaches easier to work with when it comes to generating an AST?

From testing, the parsing tree only deviates in certain expressions, where the precedence and associativity are baked in. One peculiar defect lies in the fact that multiple runs of the unambiguous grammar from testing sometimes caused different results depending on the run. These bugs are most likely caused by the poor structure of the current AST generation. Beyond expressions there doesn't seem to be any significant difference between the AST's that the two grammars generate.

As for which approach is easier to work, the answer is the ambiguous ones. Being able to define associativity and precedence, makes building from a structural standpoint easier. That being said, my AST generated is built around the ambiguous approach as it was much easier to work with, and it simply adapted to the unambiguous grammar later.

Disclaimers

The first disclaimer concerts the current. As it stands now it's very messy and can be hard to read. Due to time restraints I prioritized the AST output being correct over it being structurally sound and or “pretty”.

The second disclaimer concerns the file structure under the folder “syntaxTree”. I had a hard time adapting the build.xml file to include the sub folders i wanted to make under the syntaxTree folder. Furthermore when taking another stab at attempting to build better file/folder structure later on I also ran into a lot of java packaging issues. A better file structure is something I will probably need to incorporate later on to make my job easier, but due to time constraints I focused more on simply getting the oblig tasks done.