Max Wang and Chris Lipscomb

**Reflection**

Overall, developing the Fitch/Sequent Automated Proof Converter was an interesting project. We decided to use Java as our programming language because of its useful libraries, like DocumentBuilder, which made it easier for us to parse in our XML files. The discussion back in March where we constructed the structure for our XML file was very helpful. We only had to make small changes to the initial design.

We decided to use a class hierarchy to help organize/define a proof in Java. In addition to Fitch and Sequent classes, we created Sentence, Step, and Proof classes. This also made our software very extensible. To help in the conversion, we created a SequentStep class, which extended the Step class, to keep track of the assumption base for Sequent proof steps.

Both conversion had their own challenges, although converting from Sequent to Fitch was more difficult. When converting from Fitch to Sequent, the main challenge was creating the assumption base for each step. When converting from Sequent to Fitch, a recursive function was needed to create separate <proof></proof> tags for each subproof.

To help in testing, we created .bram files for all the basic inference rules (And Intro/Elim, Or Intro/Elim, etc). This also revealed special cases we needed to account for. For example, Or Elimination and Biconditional Introduction both require 2 subproofs. We considered this as an edge case scenario, which complicated the code a bit.

In conclusion, we successfully produced a working project with, to our knowledge, no bugs. In addition, we created pretty-print parsers to make the XML files more readable. In addition to conversions to other Proof Systems, an improvement to the software would be to parse in a pretty-print proof text file, which is much easier to create than a proof in our XML structure. Lastly, a UI could be nicer for this project that allows you to browse through your files, select the appropriate one, displays the pretty-print version of the proof, and has a button that allows you to convert the proof into another system would be a great addition to this software. The code for the project can be found here:  
<https://github.com/Risch1997/ProofConverter>

**Instructions**

In order to use this software, you simply need to download the proofConverter.jar file from the Github repository, and you need to install Java on your computer so that it can be run via the command line. Once this is done, you may open a command line, and use the command:

Java –jar proofConverter.jar command arg1 arg2

Where “command” refers to the specific command you want to use, “arg1” and “arg2” are the command-specific arguments needed to use the specific command. Use the “help” command with no arguments in order to learn about the other commands.