Title: Boogie feature extractor
Team members: Ankit Agrawal

Introduction:
A standard technique in program verification is to transform a given program into set of verification conditions whose validity implies that the program satisfies the correctness properties under consideration. Complex task of generating verification conditions for modern programming languages can be managed by separating tasks in 2 steps:
1. A transformation of program and it's proof obligations into an intermediate representation
2. Transformation of the intermediate representation to logical formulas.
Boogie is a intermediate language to accomodate encoding of verification conditions for object-oriented programs. Boogie exist for languages like Spec#, C, Dafny, Java bytecode with BML, Eiffel.

I worked with a tool developed by Yulia Demyanova which extracted important features in C source programs. I used these features to train a classifier for SMACK (a software verification tool) to predict the optimal parameter configuration which can result in improvements in runtime of the verifier. SMACK tool has several parameter configurations which directly impact the runtime to verify correctness of a program. Since the tool has several parameters, it makes sense to learn what parameter configuration is best suited to verify certain programs. The benchmark suite used is the sv-comp benchmarks and I plan to use the same benchmarks for this project.

Motivation:
1. Feature extraction on programs can help train a machine learning algorithm to learn a model to quickly analyze if the program is correct or not. In this case, it becomes extremely important to extract features that show some resemblance (pattern) to already verified programs.
2. Instead of writing a separate feature extraction tool for each of the source languages mentioned above, it makes more sense to write the tool for Boogie since Boogie can act as an intermediate input and output source (output for the translation of the source program and input to the feature extraction tool).
3. SMACK uses Boogie and generates .bpl files while trying to verify correctness of C programs..

Overview:
The main idea is to parse through boogie files (.bpl) and extract features based on the syntax/ semantics of the program.

Source program → Boogie → Generates .bpl file → Feature extractor tool → feature vector

The big picture here is to train a classifier on enough boogie files to be able to predict if a program is correct or not, with high accuracy. Since the benchmark suite is limited, it will be interesting to see if we have enough data/ benchmarks to create such classifiers. It will also be good to do analysis of runtime with other verification tools.

Approach:
1. Install boogie and generate boogie files.
2. Understanding the syntax of the generated boogie file.
3a. Install visual studio and figure out where the call to the parser is being made.
3b. Write a python script which reads the boogie file in a list.
4. Read the paper by Yulia "On the concept of variable roles and its use in Software Analysis" to understand what are considered as important features.

5. Write code to extract these features.
6. Format these features so they are easily readable.

Experiments:
1. Take train and testing samples from several categories of SV-benchmarks (C files) and generate boogie files. Then train a classifier on these categories individually and test if the model is able to correctly predict the correctness of program.
2. I would also like to compare the runtime of SMACK and the designed classifier to see if I can obtain any significant speed up.

Progress:
So far, I have installed Visual studio and cloned the boogie repository. I plan to run some boogie files (I got them from my work on SMACK) and locate the parser and go through it to see how I can invoke it to do a feature extraction.

Milestones:
1. March 15: Better understanding of scope of the problem, install boogie, generate boogie files and decide on either step 3a/ 3b.
2. March 30: Understanding the syntax of boogie code [2]
3. April 10 - Read the paper and extract the features
4. April 25 – Finish experimentation & submit observations

Related work:
1. Y. Demyanova, H. Veith, F. Zuleger. On the concept of variable roles and its use in Software Analysis
2. K. Rustan M. Leino. This is boogie2