Software Engineering Assignment 1

Prabhat Kumar Jha

September 14, 2022

Reflection Report

In the session with Luigi, we discussed how the Pynblint is used for standalone notebooks and also for directories. The tool performs a static analysis of the structure of notebooks/directories (such as arrangements of code blocks)in order to report any break in the control flow. We also discussed how this tool can be improved by combining it with the static analyzer for python.

My Ph.D. project is about path-finding algorithms. We use techniques from formal methods and these techniques have some structural similarities with static analysis of programs. Since my work is mostly theoretical, I use Jupyter notebooks on rare occasions. The output of my research is mostly algorithms and mathematical proof of their correctness. Hence, Pynblint will not be very useful for this project. Nonetheless, I look forward to combining my research with implementations in the future, and then notebooks will be an important way of documenting. In that case, Pynblint can assist me to ensure that I follow good software engineering practices.

A possible improvement that occurs to me (maybe unrealistic) is to guide analysis with the user-provided specifications. There are already some commands for some specific specifications but what I am suggesting is to have a language structure with some primary commands, some conjunctive, and some quantifiers. I understand that there is a trade-off in complexity and expressiveness of specifications, but I doubt that Pynblint is on the lower than optimal side of the expressiveness/complexity curve.

In program analysis, specifically static analysis (using techniques like abstract interpretations), the main bottleneck arises from the specifications domains. Machine learning projects do not usually have a very well-defined specification and that (in my understanding) limits the static analysis for ML projects. Code structure can have some specifications using some good practices in software engineering but the code itself does not have a good (with respect to expressiveness) specification.

Models are even harder to analyze due to many reasons including a lack of theoretical foundations to help with understanding what the model is doing. Nonetheless, some heuristic tests might be performed on the model to increase trust. Similarly, some statistical tests might be performed on data to ensure the quality of data and to suggest possible pre-processing.