Automated Verification Tool for Swift Code Bug Detection

Tommy Ngo

Motivation

- ***Why is verification needed?*** Writing error-free code can be challenging, especially as the complexity of the code increases. An Automated Verification Tool can help detect common errors in Swift code before they cause problems.
- ***Why is it interesting?*** By detecting errors early in the development process, developers can save time and effort by fixing them before they become more difficult to track down. An Automated Verification Tool can also help improve the overall quality and reliability of the code.

Current Status

- Overview of the methodology you plan to follow: The development of the tool will follow a methodology that includes identifying common errors in Swift code, translating these errors into properties that can be verified by the tool, writing tests to verify these properties, analyzing the results, improving the tool, and using it regularly to detect errors in Swift code.
- Progress made so far: So far, I have researched different verification tools and have decided to use SwiftCheck for my project. I have also begun defining the properties that my tool will verify:
  - *Identify common errors in Swift code*
  - *Translate common errors into propert*
  - *Write tests to verify the properties*

Current Status

- **Identify common errors in Swift code:** The first step in defining the properties to be verified is to identify common errors that can occur in Swift code. For example, some common errors might include force unwrapping of nil values, out-of-bounds array access, and division by zero.
- **Translate common errors into properties:** Once I have identified common errors that can occur in Swift code, I can translate them into properties that can be verified by my tool. For example, a property for force unwrapping of nil values might be "The code should not force unwrap optional values without checking for nil" or "The code should handle optional values safely using optional binding or optional chaining."
- **Write tests to verify the properties:** After defining the properties to be verified, I can write tests that will check if my code meets these properties. For example, I could write a test that attempts to force unwrap an optional value that is nil and checks if the code handles this situation safely.

Issues

- One question that we are currently considering is how to choose the right verification tool for the project.
- Another question is how to define the properties that the tool will verify.
- We are also considering how to write tests that will be used to verify the properties.
- Finally, we are considering how to analyze the results of the tests and improve the tool.