ENGR489 Meeting Minutes
QuickCheck for Whiley

Janice Chin
Primary Supervisor: David Pearce
Secondary Supervisor: Lindsay Groves

1 March 20th 2018

1.1 Present

- David Pearce

1.2 Action points were achieved since the last meeting

None as this is the first meeting.

1.3 Action points have yet to be achieved since the last meeting

None as this is the first meeting.

1.4 Action points were agreed to for the next meeting

1. Install Whiley command line tool.
2. Read research papers about QuickCheck to determine strategies for generating test cases.

1.5 Discussion Points

Objective 1: Take Whiley program and generate test cases from it using the Whiley compiler.

Assuming each Whiley program has specifications for each function.

To investigate:

- Learn and understand Whiley
- Whiley jar file for the compiler
- Whiley web project

- Maven, pom.xml for dependencies for the generator
- Need to know strategies for generating test cases

2 March 26th 2018

2.1 Present

- David Pearce

2.2 Action points were achieved since the last meeting

- Read some research papers about QuickCheck and other testing tools (DART, ArbitCheck)
- Read Whiley Getting Started Guide and started reading language specifications
- Got WhileyLabs working on my machine, could not get Whiley working on the command line.

2.3 Action points have yet to be achieved since the last meeting

None

2.4 Action points were agreed to for the next meeting

- Finish draft project proposal and send to supervisors to get it checked (by 30/3/2018).
- Reading more research papers. Suggest reading papers about American Fuzzy Lop.
- Begin implementation of project. The first steps are to create a Java project with the necessary hooks to access the Whiley compiler. And, then to be able to read a compiled Whiley file (*.wyil) to identify functions to test.

2.5 Discussion Points

Whiley can also pass functions as an argument to other functions.

- What testing method am I using? Random testing? Property testing?
  - Firstly, use random testing using the properties of the function. Pre-condition to for candidate values and post-condition to check test passed or failed. Then will look at limitations of the testing method

and decide whether to expand on the testing method (such as using dynamic symbolic execution).

- What types to use in test generation? Primitive, recursive etc
  - Primitive types: bool, byte, int, real, null, any, void
  - Array
  - Later on: records (closed), union
  - Even later: Recursive types
  - Extra (may not be implemented) - references, functionss
- Testing functions only or also test methods? Methods have side effects.
  - Test for functions first then by methods.
- Using pre- and post-conditions or property syntax?
  - Do not need to worry as the interpreter should evaluate the pre- and post-conditions. Properties are only used by the verifier and are specified in the pre- and post-conditions.