Final Year Project Proposal

General Topic:

A formalisation of Separation Logic in Agda allowing for mechanical verification of properties of programs such as but not limited to correctness.

What is the problem?

In safety-critical or security-critical systems where the cost of an undetected error is high, we want to be completely sure that there are no bugs in our software. Dijkstra once remarked, 'program testing can be used to show the presence of bugs, but never to show their absence!' To show their absence a program must be proved correct. The necessary theory for proving programs correct has been around for decades but practical implementations have lagged behind. This is changing however with new efforts such as the Verified Software Toolchain project, the Iris Project, and Infer, a 'tool to detect bugs in Java and C/C++ ... code before it ships.' These projects build upon the theory of Separation Logic, which currently has no known implementation in Agda, the dependently typed programming language and an interactive theorem prover. Such an implementation may be useful for formalising proofs pertaining to properties of programs as well as providing the beginnings of another framework within which separation logic could be used to develop more reliable software.

Methods:

My methods for this project are going to involve an implementation of Hoare Logic and then Separation Logic in the Agda programming language as well as a simple language for this logic to be applied to. Before this, however, I will read up on the tools mentioned above and the relevant literature to ensure I properly understand the context in which my project is situated.

Milestones:

1. Lit Review: Become familiar with the theory and existing tools. By: (Week 6)
2. Implementation of Hoare logic and simple language. By: (Week 8)
3. Proof of correctness of very simple program via Hoare logic in Agda. By: (Week 9)
4. Separation Logic in Agda and pointers within the simple language. By: (early Semester 2)
5. Additional Lemmas and tools to aid the construction of proofs: By: (End)
6. More interesting/ambitious program examples and proofs utilising the implementation. By: (End)