Bachelor

Isabella Odorico Schultz, Anders Friis Persson og Fie K. Hammer

January 2, 2023

Subject: 'High Assurance'

The Bachelor's project has a focus on the notion of High Assurance, where we strive to investigate this specific system which is designed with an intent to be highly secure and reliable. This notion is typically used in critical infrastructure or other applications where a failure could have serious consequences. We recognize that the system often is the subject of rigorous testing and evaluation, in order to ensure that they meet strict security and reliability standards, which we would like to explore using mechanical proofs as support. The project will be written in RUST, since this language is often mentioned in the context of high assurance systems, since it is also designed to be safe, fast, and concurrent.

Our understanding of 'High Assurance' so far

High assurance refers to a high level of security and reliability that is necessary to ensure that a software system functions correctly and predictably. This is especially relevant for systems that are safety-critical, such as systems for aviation and space, medical equipment, and financial transaction systems. To ensure high assurance, it is often necessary to thoroughly review systems to verify that they meet their specifications and security requirements.

Our understanding of 'Mechanical Proofs' so far

Mechanical proofs are techniques used to automate the verification that a software system meets their specifications and security requirements. This may include the use of formal proofs and tools such as model checkers and proof assistants to prove that a system behaves as expected. Mechanical proofs can also be used to find defects and vulnerabilities in systems before they are implemented, which can contribute to their reliability and robustness.

Possible discussions point

- Techniques for verifying the correctness of a Rust codebase, including the use of model checkers and proof assistants.
- Formal proofs and their application in the verification of a Rust codebase.
- Challenges in verifying the correctness of a Rust codebase, including handling asynchronous code and error handling.
- Comparison of different strategies for verifying correctness, including manual verification, use of model checkers, and use of formal proofs.
- Best practices for ensuring high assurance in a Rust codebase, including the use of formal proofs and tools for verifying correctness.
- Potential future developments in the area of high assurance and mechanical proofs, such as better integration of tools and techniques within Rust development.

Possible learning objectives

1. Understand the basics of the Rust programming language, including its syntax, data types, and control structures.
2. Understanding the way that mechanical proofs work with regard to High Assurance of given code snippets.
3. Learning how to create and execute mechanical proofs with Rust-based code
4. Gaining knowledge about High Assurance and the way the Rust code can be securely written.
5. Gaining knowledge about how computers and mechanical proofs can detect security flaws with regards to code itself, and learning how we in the future can become better at detecting flaws within systems and code itself.