COMS W6156 - Topics in Software Engineering

Spring 2019

Tentative Proposal

Bruno Martins - bm2787

Rust is a new (version 1.0 was released in 2015) systems programming language developed by Mozilla as a fast and safe alternative to the dominant C and C++ languages. In their words: "Rust is a systems programming language that runs blazingly fast, prevents almost all crashes, and eliminates data races". While most of the language features borrows judiciously from other languages, there is one central novel feature that seems to be unique to Rust: the borrow checker. The borrow checker **statically** evaluates the ownership and the lifetime of all pieces of data in a program to find and prevent entire classes of bugs like data races and use-after-free.

My proposal is to put their claims to test. Would Rust prevent bugs in an actual, in production codebase? How easy is it to port code from C or C++ to Rust? Would original C/C++ code and Rust ported code be comparable in the first place? In order to investigate the matter, my plan for the midterm paper is to read and evaluate existing literature on Rust and the claims they make on safety.

For the final project, the idea is to pick an open source project, port part of it to Rust (since Rust is binary compatible with C) and report on my experience. At the moment I am leaning on picking one of the modules of the EPICS framework as the target open source project (since I am familiar with it). I'd rewrite parts of it in Rust as close to the current EPICS code as possible and look for compiler complaints (potentially new, unknown bugs.) I could rewrite the same parts, but from older, known buggy version to see if *these* known bugs would have been caught by Rust. I could also experiment with mutations on the concurrent parts to see how good Rust is in those cases.