Rust's Features

David Atkinson, Sherry Nguyen, and Rami AlQunaibit

April 21, 2018

Mini-Project

In this project our team has decided to explore the rich features of Rust programming language. Our main focus is on the macros which allow users to create abstractions at the syntactic level. Macros expand the syntactic form given as an argument, before any code is run. Thus, they allow the user to capture many code patterns that Rust's core abstractions cannot, by operating directly on a program's AST. (There are potential tie-ins here with our interpreters.) In contrast to Rust, C uses a simple macro system based on direct substitution, meaning that it is less powerful and more error prone than it could be.

Rust also uses the ownership and lifetime memory management systems. This reduces the overall complexity of memory management, while maintaining most of the control that comes from explicit memory management (such as in C). In Scala, such management is done automatically by the JVM. Rust attempts to combine the two approaches, hoping to the get the benefits of both. Understanding scope is crucial to understanding Rust's approach to memory management, so we think this would be easy to relate to the course. As of now, we think these would be the two main focuses of our presentation. We might still modify some of the ideas and improve the project.