RustProof

A Stab at Formal Verification in Rust

- Bradley Rasmussen
- Michael Salter
- Matthew Slocum

Preface

We are enthusiastic students, not experts

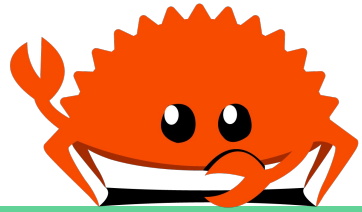
This project is strictly a proof-of-concept

Despite this, we believe there is some value here

Contents

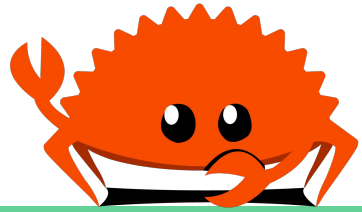
Hardships of Formal Verification

1. Tedious and Slow
2. No replacement for good testing
3. “Wouldn’t formal methods be nice?”
4. Examples of modern implementations:
    a. Isabelle
    b. Haskell
    c. Frama-C



Benefits of Formal Verification

1. Reliability
2. Security
3. Start making promises
4. Let’s automate formal methods



Some Theory

1. Hoare Logic (briefly)
2. Weakest Preconditions
3. Verification Conditions

Hoare Logic Primer

{P} S {Q}

- P : Precondition
- S : Statement
- Q : Postcondition

Example:

- P: { x < 4 }
- S: y = x + 1
- Q: { y < 10 }

If P is true, then Q should also be true after executing the code in S...

Weakest Preconditions

What's a "weakest" precondition (WP)?

Where does it come from?

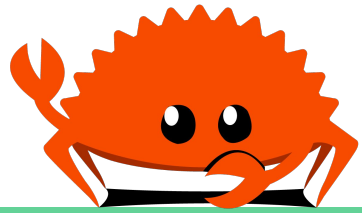
1. Given: {P} S {Q}
2. Apply S on Q to produce WP
   - WP : Weakest Precondition
   - S : Statement
   - Q : Postcondition

Example:

- P: { x < 4 }
- S: y = x + 1
- Q: { y < 10 }

Substitute

- WP: { y < 10 } where y = x + 1
- WP: { x + 1 < 10 }



Verification Condition Generation

Verification Condition: $P \rightarrow WP$

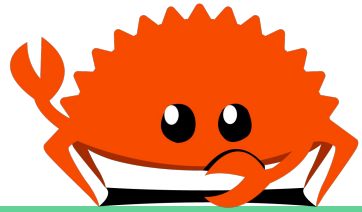
How to check VC for all x?

Check satisfiability of: $\neg(P \rightarrow WP)$

Example:

- P: $\{ x < 4 \}$
- WP: $\{ x + 1 < 10 \}$

- $P \rightarrow WP$:
  $(x < 4) \rightarrow (x + 1 < 10)$
- VC:
  $\neg((x < 4) \rightarrow (x + 1 < 10))$



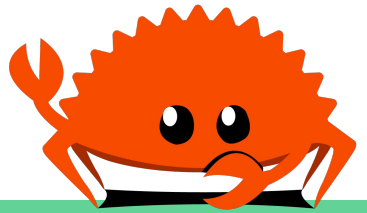
Introducing RustProof

(finally)

1. Compiler plugin
2. Why we did this with MIR
3. Applying Theory
    a. Parsing with LALRPOP
    b. VC Generation
    c. SMT-LIB and Z3

Compiler Plugins

- Why a plugin?
  - Access to user code
  - Compiler does most of the work
  - Fully automated with compilation
  - Experimentation!
- Only available on nightlies
- Compile-time analysis

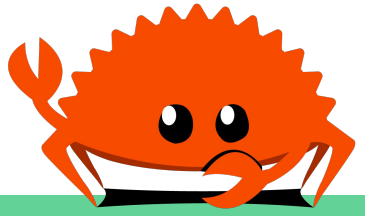


Why we did this with MIR

User code can be complex

MIR makes it simple!

- Basic blocks
- Simplified control flow
- Mostly assignment statements



Parsing, VC Generation, Solving

When RustProof analyzes code:

1. Validate user's pre- and postconditions
2. Parse and store the conditions
3. Traverse MIR blocks and generate WP
4. Create VC from WP and precondition
5. Output VC to Z3 and wait for results
6. Display results

Problems we encountered

Keeping up-to-date with nightly

Testing

- Running tests on code that only runs when other code is compiled?
- Compiling tests... at compile-time?

Supported Rust Language Features

- Integer arithmetic
- Boolean arithmetic
- Assertions
- Conditionals

Demo

http://viewpure.com/6AONwoGaquw?start=0&end=0

Questions?

RustProof

Crate: https://crates.io/crates/rustproof

Github: https://github.com/Rust-Proof/rustproof

Dependencies

libsmt.rs: https://github.com/sushant94/libsmt.rs

LALRPOP: https://github.com/nikomatsakis/lalrpop

Z3: https://github.com/Z3Prover/z3

Sponsor a capstone project like this!

http://wiki.cs.pdx.edu/capstone/