Towards a Self-Certifying Compiler for WebAssembly

Anton Xue <antonxue@seas.upenn.edu> and Kedar Namjoshi <kedar.namjoshi@helsinki.fi> labs.com

Works on hard-coded source CFG, optimized CFG, and proof relations

Testing against real-world WASM code

Write out simple optimizations like block merging and constant propagation

Converting from CFG back to WASM can be a challenge

Being able to generate valid WASM is still key to making this technique useful

Hacking existing WebAssembly toolchains' optimizations to generate proofs is hard

Primitive pipeline written with self-certifying optimizations on the way

Basic interfacing with Z3

High-level goal: smoothly integrate formal methods into software development

Insight: easier for developers to leverage techniques they are familiar with

Idea: SMT-backed checker API that exposes familiar features is likely easier to use

Instruction set for a browser-native stack-based VM

Compilation target for your favorite language

Next Steps

Challenge 2: Integration with Existing WASM Compiler Tooling

Hacking existing WebAssembly toolchains' optimizations to generate proofs is hard

Requires understanding other people's software architecture

E.g. Binaryen has wild C++ inheritance structures for its optimizations

Conclusion

We present a work-in-progress of bringing self-certifying compiler optimizations to WebAssembly. In addition to the goal of making WebAssembly language tooling more robust, self-certification as a framework has potential to tighten the gap between theory and practice: to increase adoption of formal methods in real-world software engineering. By allowing developers to work with familiar techniques while preserving formal rigor, it becomes easier—and practical—to write correct code.