A Mechanized Formalization of the WebAssembly Specification in Coq

Xuan Huang (xh5403@rit.edu)  Advisor: Dr. Matthew Fluet

Introduction

We developed a mechanized formalization of the W3C WebAssembly 1.0 specification in the Coq proof assistant, and (almost) completed a mechanized proof that the language is sound, implying both type safety and memory safety.


Methodology

Wasm is formalized as inductive tree structures (abstract syntax) with judgements and inference rules (operational semantics, static typing rules). We faithfully mechanized those mathematical definitions into Coq code with eyeball-closeness.

Spec (Prosé of Soundness)

Every thread in a valid configuration either runs forever, trapps, or terminates with a result that has the expected type. Consequently, given a valid store, no computation defined by instantiation or invocation of a valid module can “crash” or otherwise (mis)behave in ways not covered by the execution semantics given in this specification.

Spec (Statement of Preservation and Progress)

Theorem (Preservation):
If \( S, T : [S] \) and \( S \rightarrow S', T' \)
then \( S', T' : [S'] \) and \( S \leq S' \).

Theorem (Progress):
If \( S, T : [S] \) then either \( T \) is terminal or \( S \rightarrow S', T' \).

Background

Programming language theory: the formal study of PL, characterized by mathematically-precise definitions of PLs and programs.

Verification: proving the correctness of abstract systems or programs with respect to a specification or property, using formal methods.


Coq: a dependently-typed programming language enabling mechanized interactive theorem proving.

Challenges

In addition to its scale and the non-trivial soundness property, Wasm has unique features that complicate the language. Here is an example of a recursive factorial program written in C compiled to Wasm.

```
long fact(long n) {
    if (n == 0) return 1;
    else return n * fact(n - 1);
}
```

1. The recursive definition of evaluation contexts introduces non-determinism and requires manual decomposition to discover structure.
2. Branching in structured control flow references labels indexed by depth. Block types are preserved by stack unwinding and stack polymorphism.
3. Runtime structures (e.g., frames, instances, stores) not present in the source have complex invariants.

References

1. W3C, WebAssembly Core Specifications. https://www.w3.org/TR/wasm-core-1