COMP 5800 : Secure Compilation – Final Project Abstract

James Clark

November 8, 2022

1 Problem Description:

The IMP language, as annotated on the Xavier Leroy website (xavierleroy.org), has basic syntactic and semantic specification for arithmetic expressions, boolean expressions, and commands. Arithmetic expressions provide functionality for basic mathematical operations. Boolean expressions allow for specifying truth values, and also unary and binary operators over Boolean expressions. Command expressions are used for explicit control flow of the evaluation of the expression, providing the ability to assign integer values to variables in the store, perform conditional statement evaluation, perform loops, and can be specified in sequence. The IMP language specification is limited in two discernible ways. First, the IMP specification does not support more complex types such as arrays, pointers, functions, or the ability to perform heap memory allocation and access. Second, the IMP language specification does not provide any secure coding abilities at the source level. Programs written in the IMP language can be correctly compiled into a target language. In order to prove a correct compilation, the compilation's preservation of operational semantics requires explicit verification. In the base specification the compilation is specified and proved correct, through inductive reasoning over a transitional simulation of the stack machine language, in Coq. This is relatively easier to do than with languages supporting more complex types and functions and languages supporting safe coding abilities.

2 Purpose of Study:

As an exploratory study into the complexity of formally verifying an IMP language compilation scheme, the IMP language is extended to include more expressive memory operations. The functional source language features that are added are namely: arrays, pointers, and heap memory allocation, access, and freeing. The target stack machine language will duly be extended to provide operations which allow for preservation of operational semantics in compilation. For each memory operation that is added at the source and target level, there

will be a formal verification of the extended compilation relation using Coq. A stretch goal is to have a security enforcing compilation relation, that will implicitly eliminate out-of-band array access, and convert all null pointer de-references to a safe memory de-reference analogue. A final stretch goal is to extract the compiler to OCaml and run some basic tests cases.

3 Planned Deliverables:

• Commitment – Extend the IMP language specification to support pointers.
• Commitment – Extend the compilation scheme to preserve operational semantics in pointer related expressions.
• Commitment – Prove the compilation scheme to be correct in Coq.
• Stretch goal – Extend IMP and verify the compilation for arrays, and heap memory allocation, access, and freeing.
• Stretch goal – Modify the compilation scheme to enforce safe memory access at the target level.
• Stretch goal – Extract the compiler to Ocaml and run some basic unit tests.

4 Milestones:

• November 15 – Began first steps of extending IMP specification to support pointers at the source level.
• November 22 – Successfully extended source IMP specification to support pointers at the source and target level. Began reasoning over proofs of correctness in Coq.
• December 6 – Successfully extended source IMP specification to support pointers at source and target level, and successfully rewrote all proofs of correctness in Coq.