# Discussion:  Formal Verification & Proof Techniques

## Topics

- How to turn existing verified compilers into secure compilers?
- Compositional compiler correctness
- Secure compilation techniques

## Verified Compilers -> Secure?

Can existing verified compilers be converted to:

- secure compilers?
  - side channels
  - robustly safe, etc., extending CompCert (Catalin)
- compositionally correct compilers?
  - compositional CompCert (Gil)
  - CakeML, Vellvm, etc.

*When do we need entirely new proof architectures?*

*What are good strategies for reusing mechanized proof effort?*

# Compositional Compiler Correctness

Better techniques for verifying compositional correctness?

- Logical relations, PILS, interaction semantics (& structured simulations), multi-language semantics, others?
  - guidelines for which technique is suitable when
- How to reduce effort when verifying multi-pass compilers (vertical compositionality/transitivity)?
- Can separate passes be verified using different techniques?

- Infrastructure for mechanizing proofs? (e.g., Iris for logical relations?)

## Secure Compilation

Proof methods that preserve classes of security properties

- Back-translation (in the presence of an adversary)
- Relational refinement (Toby and Kedar)

# Secure Compilation

Proof methods that preserve classes of security properties

- Back-translation (in the presence of an adversary)
- Relational refinement (Toby and Kedar)

Back-translation techniques

- partial evaluation (terminating source & target languages)
- universal embedding (target has types not representable in source)
- wrappers (source, target have isomorphic types)
- approximate back-translation

Back-translation based on traces

## Secure Compilation

Proof methods that preserve classes of security properties

- Back-translation (in the presence of an adversary)
- Relational refinement (Toby and Kedar)

Back-translation techniques

- partial evaluation (terminating source & target languages)
- universal embedding (target has types not representable in source)
- wrappers (source, target have isomorphic types)
- approximate back-translation

Back-translation based on traces

Reusing proof machinery (correct compiler, back-translation)

## Secure Compilation

Translation validation & relational refinement techniques

- What is the witness for translation validation?
- Complexity of witness generation and checking?