SJTU Programming Languages and Verification

Research Areas: Theories, Tools and Applications of PL & Verification.

Applications

We develop approaches to formally verify the **correctness** and **security** of **systems software**.

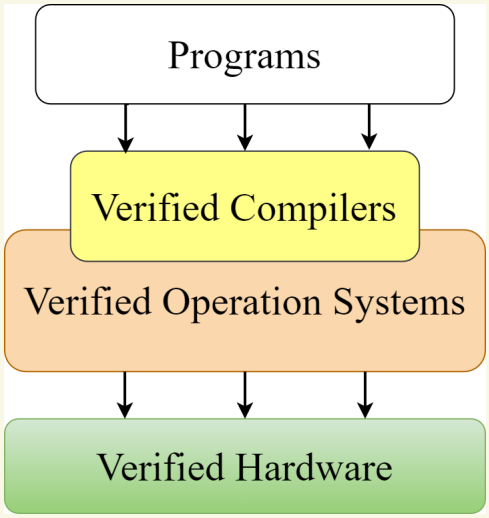


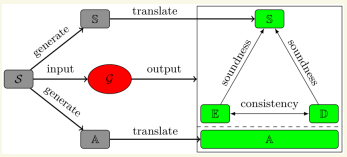
Figure: A Verified Systems Software Stack

Compilers & OS Verification

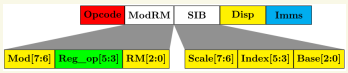
- Verified Compilers for Modular Programs
- Novel Memory Models for Verified Compilation
- Verified Program Loaders for Compilers & OS

Hardware & Architecture Verification

- Verified Machine Instruction Encoders & Decoders



(a) Auto-verification Framework



(b) X86 Instruction Format

⇑ Provide Foundations ⇑

Theories & Tools

We study programming language and verification theories rooted in **mathematical logic**.

Programming Languages

- Imperative Programming Languages
  - Based on **Turing Machines**, such as C/C++, Java, Rust and Python
  - Design **Rust-like** languages based on novel **type systems** for systems programming languages.
- Functional Programming Languages
  - Based on λ-**calculus**, such as Ocaml, Haskell, Erlang and Scala
  - We develop **front-ends** linking functional languages to LLVM.

Formal Verification

- Type theory
  - Uniform representation of proofs & programs
  - We develop our projects by using the proof assistant **Coq** based on a dependent type theory.
- Proof theory
  - Investigation of proofs as mathematical objects
  - We are in the core development team for the theorem prover **Abella** based on proof theory.

Abella

Miscellaneous

- **Faculty**: Yuting Wang (Email: `yuting.wang@sjtu.edu.cn`)
- **Selected Publications** (Top-Tier Conferences in PL & Verification): POPL2022, CAV2021, OOPSLA2020, POPL2019.
- **Collaborators**: Zhong Shao (Yale University, Department Chair of Computer Science), Gopalan Nadathur (University of Minnesota).

Yuting Wang
yuting.wang@sjtu.edu.cn
https://jhc.sjtu.edu.cn/~yutingwang/

上海交通大学
约翰·霍普克罗夫特计算机科学中心
John Hopcroft Center for Computer Science