Building Verified Language Tools in Operational Type Theory

Aaron Stump

Computational Logic Center
Computer Science Department
The University of Iowa

Thanks to Morgan Deters and Todd Schiller.

Funding from NSF CAREER.

From Meta-Theory to Tools

- Mechanized meta-theory great.

- Verified language tools also great!

- The combination definitely the greatest.

Meta-theory and Tools for LF

- Paper meta-theory for LF [Harper+05],[Watkins+02].
- Machine-checked meta-theory for LF [Urban+08].
- Unverified tools for LF: TWELF, FLIT, SC, LFSC.
- Verified tool (this talk): GOLFSOCK.
  - Verify that optimized LF checker builds type-correct LF.
  - Uses a declarative presentation of LF.
  - That presentation could be simpler.
  - More meta-theory would be needed.
  - Balance simplicity of specification, ease of verification.

Incremental Checking

- Basic idea: interleave parsing and checking [Stump08].
- Combine with bidirectional type checking.
    - Synthesizing: $\Gamma \vdash t \Rightarrow T$.
    - Checking: $\Gamma \vdash t \Leftarrow T$.
- ASTs built for subterms iff they will appear in the type T. E.g.,

  ```
  (refl x+y) => x+y == x+y
  ```

    - AST must be built for `x+y`.
    - But not `(refl x+y)`.
- C++ implementation: small footprint, fastest checker I know.

A Need for Correctness

- LF with Side Conditions (LFSC) proposed for SMT.
  - Satisfiability Modulo Theories.
  - SMT solvers check large formulas, produce big proofs.
  - Must check proofs efficiently.
  - LFSC provides flexible intermediate proof language.
- Problems with C++ (proof checker):
  - Lack of memory safety => many days with VALGRIND.
  - Optimizations reduce trustworthiness.
- As features are added to checker, trust diminishes.
- Additional assurance is required.

GOLFSOCK: Towards A Verified LF Checker

- Goal: implement verified LFSC checker.
- GOLFSOCK: incremental LF checker in GURU.
  - GURU is a verified programming language.
  - Combines a dependently type PL, logical theory (OpTT).
  - Supports mutable state, non-termination, input/output.
  - Type/proof checker, compiler to efficient C code.
  - Beating native code OCAML on small testcases.
- Status:
  - GOLFSOCK implemented.
  - Running reasonably fast: 40% slower than C++ version.
  - Specification: ASTs we build are type correct LF.
  - Expressed with dependent types, declarative LF.
  - A number of lemmas still to prove.
  - 4300 lines code, proof.
  - 13000 lines standard library (e.g., tries).