Compositional and Lightweight Dependent Type Inference for ML

He Zhu and Suresh Jagannathan
Computer Science Department of Purdue University

Motivation
- Consider an open-source bit-vector library
- Procedure has complex invariants
- buggy, our tool, discovers this invariant only using counterexample guided refinement
- Bug detection

Inference and Checking
Inference follows typing rules which are quite like traditional typing rules
Subtype constraint built for this application leads to the verification condition:
\[(x \geq 0 \land \forall y \leq x \Rightarrow \forall y \geq 0) \land (\forall y \geq 0 \land x \geq 0 \Rightarrow y \geq 0)\]

Refinement: Constraint Propagation

Counterexample path
Negation of the verification condition is supplied to SMT which may produce counterexample as satisfiable assignment
Build counterexample paths from counterexamples

Refinement and Dependent Types

Counterexample path:
\[\forall y \geq 0 \land x \geq 0 \Rightarrow \forall y \geq 0\]

Refinement: WP Generation

Framework

Benchmarks

Related Work
- Liquid Types
- Higher-Order Program Model Checking
- Dependent Types from Counterexamples
- Verifying Functional Programs using Abstract Interpreters