**The Problem**

What is abstract interpretation?
- Approximation of program properties such as termination, runtime errors, reachability, ...
- Examples: static Analyzers, type Checkers
- Used in compilers, for debugging, and verification

**Soundness proofs of abstract interpreters**

We describe concrete and abstract interpreter with a single shared proofs more compositional

Arrow-based abstract interpreters make soundness proofs of abstract interpreters

**Our Solution**

**Soundness of pure functions**

Parametricity is essential for soundness of arrow-based abstract interpreters

We require that all pure functions $(\text{arr } f)$ uses types with non-standard orderings parametrically

**Fixpoints**

Calculating fixpoints of arrow-based abstract interpreters

Challenge: Finding precise and sound fixpoint combinators that requires no explicit soundness proof of eval

We adapted the technique of calculating fixpoints from Abstracting Abstract Machines (Van Horn).
Function calls are bound to a finite amount of addresses.
If the interpreter runs out of addresses, it overapproximates.

**Benefits**

- Decompose soundness proof into smaller proof obligations
- Construction of a soundness proof of eval without manual proof effort