Hybrid Type Checking

An implementation of $\lambda^H$

David Waern
Rickard Nilsson
Hybrid Type Checking

- Cormac Flanagan POPL 2006

- A combination of Static and Dynamic checking
  If a specification can't be checked statically, it will be checked dynamically

- Dynamic Type Casts
  Casts are inserted when static checking fails to prove or disprove
Hybrid Type Checking

- Precise specifications are supported
- Advanced specifications and static analyses can be tried out
- Selectable trade-off between compilation speed and coverage
Typed $\lambda$-calculus

Refinement types

$$\text{Natural} = \{x : \text{Int} \mid x > 0\}$$

Dependent function types

$$f : m : \text{Int} \to n : \text{Natural} \to \{x : \text{Int} \mid x = m + n\}$$

Undecidable type checking
Implementation

- Follows Flanagan's description closely

- Haskell
Parser

- Parsec parser combinator library

- Tested with QuickCheck
Type checker / Compiler

- Basic structural static type checking
- Actual checking done in subtyping function
- Casts injected if sub typer fails
Subtyping

- Simple rejections
- Accepts types with structurally equal predicates
- Refinement predicate evaluation for applications of constants
- Easily extendable
- Possible to plugin a theorem prover
Interpreter

- Evaluates inserted casts, which may fail
Demonstration