A general, expressive, and accurate diagnostic method for Haskell errors

**Motivation**

Type-inference error messages are sometimes confusing

![Haskell compiler: Expected 'Nat' is not a numerical type](image)

Haskell compiler: Expected 'Nat' is not a numerical type

**Overview**

Diagnosing Type Errors with Class
Danfeng Zhang and Andrew C. Myers
Cornell University

Dimitrios Vytiniotis and Simon Peyton-Jones
Microsoft Research Cambridge

**A general approach to diagnosing Haskell errors**

**SHErrLoc Overview**

**SHErrLoc Constraints**

A rich constraint language that models GHC type checking

**Syntax of Constraints**

- Constraint elements $E :\; \Rightarrow \; a \limp \{ \text{con} \{ E_0, \ldots, E_n \} \}$
- Quantified Axioms $Q :\; \forall \alpha \cdot \alpha_i \Rightarrow I$

**Inequalities**

- $\Gamma_1 \Rightarrow E_1 \leq E_2$

**Constraints**

- $\Gamma_1 \Rightarrow \alpha_1 \leq \alpha_2$

**Examples**

- $[\text{Int} \tau :: \tau \leq \text{D}] = \{ \text{Int} \}$
- $[\tau :: \tau \leq \text{D}] = \{ (\text{tup}_3 \tau) \}$

**Type Classes**

- `g politic 20 :: Int`
- `g politic 20 :: Integer`
- `g politic 20 :: Float`
- `g politic 20 :: Double`

**Type Inference Engine**

- Node: constraint elements
- Edge: partial orders on elements

**Finding informative paths**

- Saturate constraint graph
- Test the satisfiability of a partial order on end nodes
- Trivial paths are ignored (e.g., one end node is a variable)

**Inequalities**

- $\Gamma_1 \Rightarrow E_1 \leq E_2$

**Contraints may have hypotheses and axioms**

$(\text{Int} \leq \text{Eq}) \land (\forall a \cdot a \leq \{ [a] \leq \text{Eq} \} \Rightarrow [\text{Int} \leq \text{Eq}])$

**Inference Engine**

- Model constraints on end nodes
- Test the satisfiability of the partial order on end nodes

**Evaluation**

Correctness metric
- CE Benchmark: well-marked errors in the benchmark
- Helium Benchmark: user's fix with larger time stamp

**Graph Saturation**

Idea: add new graph nodes and edges during saturation

- $\text{Int} \leq \text{Cla} \Rightarrow \alpha \Rightarrow \{ \alpha \leq \text{Cla} \}$

**Inferring Likely Wrong Constraints**

Idea: redundant paths are useless in ranking

**Haskell-Specific**

- We handle the highly expressive type system of GHC
  - Type classes
  - Type families
  - GADTs
  - Type signatures

**Challenges**

Haskell features we handle