Design of a typechecker

David Himmelstrup

May 8th, 2019

Haskell Suite

- haskell-src-exts :: String $\rightarrow$ AST SrcLoc
- haskell-scope :: AST SrcLoc $\rightarrow$ AST Origin
- haskell-tc :: AST Origin $\rightarrow$???

- haskell-src-exts :: String $\rightarrow$ AST SrcLoc
- haskell-scope :: AST SrcLoc $\rightarrow$ AST Origin
- haskell-tc :: AST Origin $\rightarrow$???
- typing-haskell-in-haskell :: AST $\rightarrow$ [TypeSig]

Use-cases

- Compilers
- Documentation systems
- Type-directed source code suggestions
- Teaching

GHC

- Shuffles code
- Deletes code
- Adds new code
- Names every type variable 'p'

```
length "pie"
```

`length "pie"`

$$\forall a.[a] \to Int \longrightarrow [Char] \to Int$$

@Char

`length "pie"`

$$\forall a.[a] \to \mathit{Int} \quad \longrightarrow_{@\mathit{Char}} \quad [\mathit{Char}] \to \mathit{Int}$$

$\underline{\mathrm{f}}$

$$\underline{\texttt{f}}$$

$$\forall a.a \to \forall b.b \qquad \longrightarrow \qquad \forall ab.a \to b$$

$$\Lambda ab.\lambda arg.\texttt{f @a arg @b}$$

$$\underline{\texttt{f}}$$

$$\forall a.a \to \forall b.b \qquad \longrightarrow \qquad \forall ab.a \to b$$

$$\Lambda ab.\lambda arg.\texttt{f @a arg @b}$$

haskell-tc :: AST Origin $\rightarrow$ AST Typed

Type signature for bindings, coercions for each usage site.

Problems

1. Code shuffle

   Annotate AST with mutable references. Apply TC algorithm. Freeze AST.

2. Naming type variables
   - No scoping rules.
   - Preference to user-written types.
   - No shadowing.

Problems

1. Code shuffle
   Annotate AST with mutable references. Apply TC algorithm. Freeze AST.
2. Naming type variables
   - No scoping rules.
   - Preference to user-written types.
   - No shadowing.

Problems

1. Code shuffle
   Annotate AST with mutable references. Apply TC algorithm. Freeze AST.
2. Naming type variables
   - No scoping rules.
   - Preference to user-written types.
   - No shadowing.

Naming

```
const x _ = x
  where
    id x = x
```

Naming

```
const :: ∀ab.a → b → a
const x _ = x
  where
    id :: ∀c.c → c
    id x = x
```

Naming

```
outer x = x
  where
    inner :: a -> a
    inner y = const x y
```

Naming

```
outer :: ∀b.b → b
outer x = x
  where
    inner :: ∀a.a → a
    inner y = const @b @a x y
```

Naming

```
id1 x = id2 x
id2 x = id1 x
```

Naming

$$
\begin{aligned}
&\mathsf{id1} \;::\; \forall ab.a \to b \\
&\mathsf{id1}\;\; \mathsf{x} = \mathsf{id2}\;\; \mathsf{x} \\
&\mathsf{id2} \;::\; \forall ab.a \to b \\
&\mathsf{id2}\;\; \mathsf{x} = \mathsf{id1}\;\; \mathsf{x}
\end{aligned}
$$

haskell-tc

- Pure API (with ST under the hood)
- Annotates, never modifies
- Human-readable output
- Aims to support Haskell2010 + RankNTypes