Contextual Typing

Xu Xue and *Bruno C. d. S. Oliveira*

The University of Hong Kong

Type Inference and what we believe …

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
  - unambitious in complete type inference;
  - the places to put the annotations should be easy to predict;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;
  - better error report;
  - better performance;
  - etc.

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;
- Guidelines are easy to follow;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;
- Guidelines are easy to follow;
  - for language designers;
  - and programmers;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;
- Guidelines are easy to follow;
- Scalability is necessary;

Type Inference and what we believe ...

- Let type annotations be reasonable and meaningful;
- Type information propogation is local;
- Guidelines are easy to follow;
- Scalability is necessary;
- Implementation can be easily derived.

Bidirectional Typing

- Merge type inference and type checking by two modes;

Bidirectional Typing

- Merge type inference and type checking by two modes;
  - Inference mode: $\Gamma \vdash e \Rightarrow A$

Bidirectional Typing

- Merge type inference and type checking by two modes;
  - Inference mode: $\Gamma \vdash e \Rightarrow A$
  - Checking mode: $\Gamma \vdash e \Leftarrow A$

Bidirectional Typing

- Merge type inference and type checking by two modes;
  - Inference mode: $\Gamma \vdash e \Rightarrow A$
  - Checking mode: $\Gamma \vdash e \Leftarrow A$
- Mode-correct bidirectional type systems can be directly implemented;

Bidirectional Typing

- Merge type inference and type checking by two modes;
  - Inference mode: $\Gamma \vdash e \Rightarrow A$
  - Checking mode: $\Gamma \vdash e \Leftarrow A$
- Mode-correct bidirectional type systems can be directly implemented;

```
infer :: Env → Term → Type
check :: Env → Term → Type → Bool
```

Bidirectional Typing

- Merge type inference and type checking by two modes;
  - Inference mode: $\Gamma \vdash e \Rightarrow A$
  - Checking mode: $\Gamma \vdash e \Leftarrow A$
- Mode-correct bidirectional type systems can be directly implemented;

```
infer :: Env → Term → Type
check :: Env → Term → Type → Bool
```

- Types are propogated to neighbouring expressions;

Bidirectional Typing: Problems

- Trade-off between expressive power and backtracking;
  - more expressive, less syntax-directness;
  - all-or-nothing inference strategy;
- Unclear annotatability and rule duplication;
- Inexpressive subsumption.

Our Proposal: Contextual Typing

- Quantitative Type Assignment Systems (QTASs);
  - as a specification for programmers;
  - tells you where the annotations are needed;
  - parametrised with a counter: $\Gamma \vdash_n e : A$
- Syntax-directed Algorithmic Type Systems;
  - is decidable;
  - parametrised with a context: $\Gamma \vdash \Sigma \Rightarrow e \Rightarrow A$

QTAS: STLC

$$\textsc{DLit}\quad \frac{}{\Gamma \vdash_0 i : \mathtt{Int}}$$

$$\textsc{DVar}\quad \frac{x : A \in \Gamma}{\Gamma \vdash_0 x : A}$$

$$\textsc{DAnn}\quad \frac{\Gamma \vdash_\infty e : A}{\Gamma \vdash_0 (e : A) : A}$$

$$\textsc{DLam}\quad \frac{\Gamma, x : A \vdash_\infty e : B}{\Gamma \vdash_\infty \lambda x.\ e : A \to B}$$

$$\textsc{DApp1}\quad \frac{\Gamma \vdash_0 e_1 : A \to B \qquad \Gamma \vdash_\infty e_2 : A}{\Gamma \vdash_0 e_1\ e_2 : B}$$

$$\textsc{DApp2}\quad \frac{\Gamma \vdash_\infty e_1 : A \to B \qquad \Gamma \vdash_0 e_2 : A}{\Gamma \vdash_\infty e_1\ e_2 : B}$$

$$\textsc{DSub}\quad \frac{\Gamma \vdash_0 e : A \qquad A = B}{\Gamma \vdash_\infty e : B}$$

Algo: STLC

$$\frac{}{\Gamma \vdash \square \Rightarrow i \Rightarrow \texttt{Int}}\ \textsc{ALit}$$

$$\frac{x : A \in \Gamma}{\Gamma \vdash \square \Rightarrow x \Rightarrow A}\ \textsc{AVar}$$

$$\frac{\Gamma \vdash A \Rightarrow e \Rightarrow B}{\Gamma \vdash \square \Rightarrow e : A \Rightarrow A}\ \textsc{AAnn}$$

$$\frac{\Gamma \vdash \boxed{e_2} \mapsto \Sigma \Rightarrow e_1 \Rightarrow A \to B}{\Gamma \vdash \Sigma \Rightarrow e_1\ e_2 \Rightarrow B}\ \textsc{AApp}$$

$$\frac{\Gamma, x : A \vdash B \Rightarrow e \Rightarrow C}{\Gamma \vdash A \to B \Rightarrow \lambda x.\ e \Rightarrow A \to C}\ \textsc{ALam1}$$

$$\frac{\Gamma \vdash \square \Rightarrow e_2 \Rightarrow A \qquad \Gamma, x : A \vdash \Sigma \Rightarrow e \Rightarrow B}{\Gamma \vdash \boxed{e_2} \mapsto \Sigma \Rightarrow \lambda x.\ e \Rightarrow A \to B}\ \textsc{ALam2}$$

$$\frac{\Gamma \vdash \square \Rightarrow g \Rightarrow A \qquad \Sigma \neq \square \qquad A \approx \Sigma}{\Gamma \vdash \Sigma \Rightarrow g \Rightarrow A}\ \textsc{ASub}$$

Soundness

Soundness

$$If\ \Gamma \vdash \square \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_0 e : A.$$

Soundness

$$If\ \Gamma \vdash \square \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_0 e : A.$$

$$If\ \Gamma \vdash A \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_\infty e : A.$$

Soundness

$$If\ \Gamma \vdash \square \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_0 e : A.$$

$$If\ \Gamma \vdash A \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_\infty e : A.$$

Completeness

Soundness

$$If\ \Gamma \vdash \square \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_0 e : A.$$

$$If\ \Gamma \vdash A \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_\infty e : A.$$

Completeness

$$If\ \Gamma \vdash_0 e : A, then\ \Gamma \vdash \square \Rightarrow e \Rightarrow A.$$

Soundness

$$If\ \Gamma \vdash \square \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_0 e : A.$$

$$If\ \Gamma \vdash A \Rightarrow e \Rightarrow A, then\ \Gamma \vdash_\infty e : A.$$

Completeness

$$If\ \Gamma \vdash_0 e : A, then\ \Gamma \vdash \square \Rightarrow e \Rightarrow A.$$

$$If\ \Gamma \vdash_\infty e : A, then\ \Gamma \vdash A \Rightarrow e \Rightarrow A.$$

Recap

- Contextual typing is a lightweight approach to type inference
  - that exploits partially known contextual information;
- It enables several improvements over bidirectional typing

Code Block

```
infer :: Int → Int → Int

infer n1 n2 = n1 + n2
```