Type-based Program Analysis

IPA Spring Days 2013

Ruud Koot
Utrecht University
r.koot@uu.nl

May 13, 2012

Definitions

Syntax

$$\begin{array}{rcl} e & ::= & x \mid \lambda x.\, e \mid e_1\ e_2 \\ & \mid & (e_1, e_2) \mid \textbf{true} \mid \textbf{false} \\ & \mid & \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \end{array}$$

Types

$$\tau ::= \tau_1 \to \tau_2 \mid \tau_1 \times \tau_2 \mid \textbf{bool}$$

Typing rules

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ [\text{T-Var}] \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.e : \tau_1 \to \tau_2}\ [\text{T-Abs}]$$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ e_2 : \tau_1}\ [\text{T-App}]$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}\ [\text{T-Pair}]$$

$$\frac{}{\Gamma \vdash \textbf{true} : \textbf{bool}}\ [\text{T-True}] \qquad \frac{}{\Gamma \vdash \textbf{false} : \textbf{bool}}\ [\text{T-False}]$$

$$\frac{\Gamma \vdash e_1 : \textbf{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 : \tau}\ [\text{T-If}]$$

Typing derivation

Example

$$(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ \mathbf{true}$$

Derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{x_1:\mathbf{bool}\to\mathbf{bool},\ldots\vdash x_1:\mathbf{bool}\to\mathbf{bool}}\quad\dfrac{}{\ldots,x_2:\mathbf{bool}\vdash x_2:\mathbf{bool}}}{x_1:\mathbf{bool}\to\mathbf{bool},x_2:\mathbf{bool}\vdash x_1\ x_2:\mathbf{bool}}}{x_1:\mathbf{bool}\to\mathbf{bool}\vdash\lambda x_2.x_1\ x_2:\mathbf{bool}\to\mathbf{bool}}}{\vdash\lambda x_1.\lambda x_2.x_1\ x_2:(\mathbf{bool}\to\mathbf{bool})\to\mathbf{bool}\to\mathbf{bool}}\quad\dfrac{\dfrac{}{x_3:\mathbf{bool}\vdash x_3:\mathbf{bool}}}{\vdash\lambda x_3.x_3:\mathbf{bool}\to\mathbf{bool}}}{\vdash(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3):\mathbf{bool}\to\mathbf{bool}}\quad\dfrac{}{\vdash\mathbf{true}:\mathbf{bool}}}{\vdash(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ \mathbf{true}:\mathbf{bool}}$$

Inference Algorithm W (Damas–Hindley–Milner)

Control-flow analysis

Problem Software analysis tools often require access to a *control flow graph*:

- For *first-order languages* it can be extracted syntactically.
- *Higher-order languages* require semantic methods.

Analysis Solve the *dynamic dispatch problem*. I.e., for a given program e, determine for all variables x_i in e to which abstractions they can be bound at run-time.

Example 1

Example

$$(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ \textbf{true}$$

Solution

$$\begin{aligned} x_1 &\mapsto \{x_3\} \\ x_2 &\mapsto \emptyset \\ x_3 &\mapsto \emptyset \end{aligned}$$

Example 2

Example

$$(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ (\lambda x_4.x_4)$$

Solution

$$\begin{aligned} x_1 &\mapsto \{x_3\} \\ x_2 &\mapsto \{x_4\} \\ x_3 &\mapsto \{x_4\} \\ x_4 &\mapsto \emptyset \end{aligned}$$

Annotated types

Annotations

$$\varphi \in \mathcal{P}(\mathbf{Var})$$

Types

$$\widehat{\tau} ::= \widehat{\tau}_1 \xrightarrow{\varphi} \widehat{\tau}_2 \mid \widehat{\tau}_1 \times \widehat{\tau}_2 \mid \mathbf{bool}$$

Type system (1st attempt)

$$\frac{}{\Gamma, x : \widehat{\tau} \vdash x : \widehat{\tau}} \text{ [CF-Var]} \qquad \frac{\Gamma, x : \widehat{\tau}_1 \vdash e : \widehat{\tau}_2}{\Gamma \vdash \lambda x.e : \widehat{\tau}_1 \xrightarrow{\{x\}} \widehat{\tau}_2} \text{ [CF-Abs]}$$

$$\frac{\Gamma \vdash e_1 : \widehat{\tau}_2 \xrightarrow{\varphi} \widehat{\tau}_1 \quad \Gamma \vdash e_2 : \widehat{\tau}_2}{\Gamma \vdash e_1\ e_2 : \widehat{\tau}_1} \text{ [CF-App]}$$

$$\frac{\Gamma \vdash e_1 : \widehat{\tau}_1 \quad \Gamma \vdash e_2 : \widehat{\tau}_2}{\Gamma \vdash (e_1, e_2) : \widehat{\tau}_1 \times \widehat{\tau}_2} \text{ [CF-Pair]}$$

$$\frac{}{\Gamma \vdash \textbf{true} : \textbf{bool}} \text{ [CF-True]} \qquad \frac{}{\Gamma \vdash \textbf{false} : \textbf{bool}} \text{ [CF-False]}$$

$$\frac{\Gamma \vdash e_1 : \textbf{bool} \quad \Gamma \vdash e_2 : \widehat{\tau} \quad \Gamma \vdash e_3 : \widehat{\tau}}{\Gamma \vdash \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 : \widehat{\tau}} \text{ [CF-If]}$$

Typing derivation

Example

$$(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ (\lambda x_4.x_4)$$

Derivation

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdots}{x_1:\left(\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\right)\xrightarrow{x_3}\widehat{\tau}\xrightarrow{x_4}\widehat{\tau},x_2:\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\vdash x_1\ x_2:\widehat{\tau}}}{x_1:\left(\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\right)\xrightarrow{x_3}\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\vdash\lambda x_2.x_1\ x_2:\widehat{\tau}\to\widehat{\tau}}}{\vdash\lambda x_1.\lambda x_2.x_1\ x_2:\ldots} \quad \dfrac{\dfrac{}{x_3:\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\vdash x_3:\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}}}{\vdash\lambda x_3.x_3:\left(\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\right)\xrightarrow{x_3}\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}}}{\vdash(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3):\left(\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}\right)\xrightarrow{x_3}\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}} \quad \dfrac{\dfrac{}{x_4:\widehat{\tau}\vdash x_4:\widehat{\tau}}}{\vdash\lambda x_4.x_4:\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}}}{\vdash(\lambda x_1.\lambda x_2.x_1\ x_2)\ (\lambda x_3.x_3)\ (\lambda x_4.x_4):\widehat{\tau}\xrightarrow{x_4}\widehat{\tau}}$$

Inference

- Algorithm W with unification modulo UCAI
- Two-phase constraint-based type inference

Conservative extension

- This system fails to be a *conservative extension*:

$$\textbf{if true then } (\lambda x_1.x_1) \textbf{ else } (\lambda x_2.x_2)$$

Conservative extension

- This system fails to be a *conservative extension*:

$$\textbf{if true then } (\lambda x_1.x_1) \textbf{ else } (\lambda x_2.x_2)$$

- Introduce *subeffecting*:

$$\frac{\Gamma, x : \widehat{\tau}_1 \vdash e : \widehat{\tau}_2}{\Gamma \vdash \lambda x.e : \widehat{\tau}_1 \xrightarrow{\{x\} \cup \varphi} \widehat{\tau}_2} \text{ [CF-Abs]}$$

Poisoning

- Subeffecting can cause *poisoning*:

$$\begin{array}{ll} \textbf{let} & id = \lambda x_1.x_1 \\ \textbf{in} & (\ \textbf{if true then } id \textbf{ else } (\lambda x_2.x_2)\,, \\ & \quad \textbf{if true then } id \textbf{ else } (\lambda x_3.x_3)\ ) \end{array}$$

- Need to assign to *id* the type:

$$id : \widehat{\tau} \xrightarrow{\{x_1,x_2,x_3\}} \widehat{\tau}$$

- Analyzed type of the whole program is:

$$\left(\widehat{\tau} \xrightarrow{\{x_1,x_2,x_3\}} \widehat{\tau}\right) \times \left(\widehat{\tau} \xrightarrow{\{x_1,x_2,x_3\}} \widehat{\tau}\right)$$

  instead of:

$$\left(\widehat{\tau} \xrightarrow{\{x_1,x_2\}} \widehat{\tau}\right) \times \left(\widehat{\tau} \xrightarrow{\{x_1,x_3\}} \widehat{\tau}\right)$$

Context-sensitivity

- Introduce *subtyping*:

$$\frac{\Gamma \vdash e_1 : \mathbf{bool} \quad \Gamma \vdash e_2 : \widehat{\tau}_2 \quad \Gamma \vdash e_3 : \widehat{\tau}_3 \quad \widehat{\tau}_2 \leq \widehat{\tau} \quad \widehat{\tau}_3 \leq \widehat{\tau}}{\Gamma \vdash \mathbf{if}\ e_1\ \mathbf{then}\ e_2\ \mathbf{else}\ e_3 : \widehat{\tau}} \text{ [CF-If]}$$

$$\frac{\widehat{\tau}_1' \leq \widehat{\tau}_1 \quad \widehat{\tau}_2 \leq \widehat{\tau}_2' \quad \varphi \subseteq \varphi'}{\widehat{\tau}_1 \xrightarrow{\varphi} \widehat{\tau}_2 \leq \widehat{\tau}_1' \xrightarrow{\varphi'} \widehat{\tau}_2'} \text{ [S-Arrow]}$$

Context-sensitivity

- Introduce *subtyping*:

$$\frac{\Gamma \vdash e_1 : \textbf{bool} \quad \Gamma \vdash e_2 : \widehat{\tau}_2 \quad \Gamma \vdash e_3 : \widehat{\tau}_3 \quad \widehat{\tau}_2 \leq \widehat{\tau} \quad \widehat{\tau}_3 \leq \widehat{\tau}}{\Gamma \vdash \textbf{if } e_1 \textbf{ then } e_2 \textbf{ else } e_3 : \widehat{\tau}} \text{ [CF-If]}$$

$$\frac{\widehat{\tau}_1' \leq \widehat{\tau}_1 \quad \widehat{\tau}_2 \leq \widehat{\tau}_2' \quad \varphi \subseteq \varphi'}{\widehat{\tau}_1 \xrightarrow{\varphi} \widehat{\tau}_2 \leq \widehat{\tau}_1' \xrightarrow{\varphi'} \widehat{\tau}_2'} \text{ [S-Arrow]}$$

- Introduce *polyvariance*:

$$id : \forall \varphi : \widehat{\tau} \xrightarrow{\{x_1\} \cup \varphi} \widehat{\tau}$$

Exception analysis

- Accurately approximate exceptions that may be thrown at run-time
- In particular: pattern-match failures
  - Requires analyzing data flow

Example

Program

```
risers :  Ord a => [a] -> [[a]]
risers []                    = []
risers [x]                   = [[x]]
risers (x_1 :: x_2 :: xs) = if x_1 <= x_2
                               then (x_1 :: y) :: ys
                               else [x_1] :: (y :: ys)
    where (y :: ys) = risers (x_2 :: xs)
```

REPL

```
risers [1,4,7,3,6,2]
> [[1,4,7],[3,6],[2]]
```

Type system (informally)

- The three branches can be assigned the types:

$$\mathit{risers}_1 : \forall\alpha.\mathit{Ord}\ \alpha \Rightarrow [\alpha]^{\mathbf{N}} \to [[\alpha]^{\mathbf{N}\sqcup\mathbf{C}}]^{\mathbf{N}}$$
$$\mathit{risers}_2 : \forall\alpha.\mathit{Ord}\ \alpha \Rightarrow [\alpha]^{\mathbf{C}} \to [[\alpha]^{\mathbf{N}\sqcup\mathbf{C}}]^{\mathbf{C}}$$
$$\mathit{risers}_3 : \forall\alpha.\mathit{Ord}\ \alpha \Rightarrow [\alpha]^{\mathbf{C}} \to [[\alpha]^{\mathbf{N}\sqcup\mathbf{C}}]^{\mathbf{C}}$$

- The whole function then gets type:

$$\mathit{risers} : \forall\alpha.\mathit{Ord}\ \alpha \Rightarrow [\alpha]^{\mathbf{N}\sqcup\mathbf{C}} \to [[\alpha]^{\mathbf{N}\sqcup\mathbf{C}}]^{\mathbf{N}\sqcup\mathbf{C}}$$

- This is not what we want!

Polyvariant recursion

- It seems polyvariance might save us:

$$\mathit{risers} : \forall\alpha\beta.\mathit{Ord}\ \alpha \Rightarrow [\alpha]^{\beta} \to [[\alpha]^{\mathbf{N}\sqcup\mathbf{C}}]^{\beta}$$

- But in Hindley–Milner **fix** is always monomorphic
- We need Milner–Mycroft's polymorphic **fix**

Aap

Aap