Liquid Types Revisited

Mário Pereira Sandra Alves Mário Florido

University of Oporto - Department of Computer Science & LIACC

May 18, 2014

Outline

Consider the following OCaml examples:

```
let div x y = x / y ;;
val div : int → int → int = <fun>
```

```
let last_cell = function
  | [v] -> v
  | e :: l -> last_cell l
```

First one ⇝ compile-time error if $y = 0$.

The other one ⇝ if, provably, `last_cell` is never applied to an empty list ⇝ no warning about incomplete pattern matching.

Refinement types can be used to overcome such situations.

Outline

Refinement types: complex program's invariants, by augmenting type systems with **logical annotations**.

$\{\nu : B \mid \phi\}$ stands for the set of terms M of basic type B such that the formula $[M/\nu]\phi$ holds.

$\{\nu : \mathrm{int} \mid \nu \geq 0 \wedge \nu \leq 5\}$ represents the integers values from 0 to 5.

Let
$$\frac{\Gamma \vdash M : \tau \qquad \Gamma; x : \tau \vdash N : \sigma \qquad \Gamma \vdash \sigma}{\Gamma \vdash \text{let } x = M \text{ in } N : \sigma}$$

P-Var
$$\frac{\Gamma(x) = y : \sigma \to \tau}{\Gamma \vdash x : (y : \sigma \to \tau)}$$

Fun
$$\frac{\Gamma_x; x : \sigma \vdash M : \tau \qquad \Gamma_x \vdash \sigma}{\Gamma_x \vdash \lambda x.M : (x : \sigma \to \tau)}$$

App
$$\frac{\Gamma \vdash M : (x : \tau \to \sigma) \qquad \Gamma \vdash N : \tau}{\Gamma \vdash MN : [N/x]\sigma}$$

B-Var
$$\frac{\Gamma(x) = \{\nu : B \mid \phi\}}{\Gamma \vdash x : \{\nu : B \mid \nu = x\}}$$

Cst
$$\frac{}{\Gamma \vdash c : \text{ty}(c)}$$

Sub
$$\frac{\Gamma \vdash M : \tau \qquad \Gamma \vdash \tau <: \sigma \qquad \Gamma \vdash \sigma}{\Gamma \vdash M : \sigma}$$

Figure 1: Type system for refinement types

$$\frac{}{\Gamma \vdash \alpha <: \alpha}\ <:\text{-Var} \qquad \frac{\Gamma \vdash \sigma <: \tau}{\Gamma \vdash \forall\alpha.\sigma <: \forall\alpha.\tau}\ <:\text{-Poly}$$

$$\frac{\Gamma \vdash \sigma_2 <: \sigma_1 \qquad \Gamma; x : \sigma_2 \vdash \tau_1 <: \tau_2}{\Gamma \vdash x : \sigma_1 \to \tau_2 <: x : \sigma_2 \to \tau_2}\ <:\text{-Fun} \qquad \frac{\Gamma; \nu : B \vdash \phi \Rightarrow \psi}{\Gamma \vdash \{\nu : B \mid \phi\} <: \{\nu : B \mid \psi\}}\ <:\text{-Base}$$

Figure 2: Subtyping relation

Rule [<:-Base] generates **implication conditions**.

Arbitrary boolean terms for refinement predicates $\rightsquigarrow$ **undecidability**.

Outline

Liquid Types (*Logically Qualified Data Types*) $\rightsquigarrow$ Damas-Milner inference $+$ predicate abstraction:

1. refinements are conjunctions of logical qualifiers;
2. decidable notion of subtyping.

Extension to the Damas-Milner type system.

Refinement type $\rightsquigarrow$ refinement of the corresponding ML type.

Let $\mathbb{Q} = \{\nu \geq 0, x \leq \nu, y \leq \nu\}$

- $\mathbb{Q} \rightsquigarrow$ set of logical qualifiers, over **program variables**, the **value variable** ν and the **variable placeholder** $\star$.

Consider the following piece of code:

let max x y = **if** x > y **then** x **else** y

Liquid Types is able to infer

$$\texttt{max} :: x : \texttt{int} \rightarrow y : \texttt{int} \rightarrow \{\nu : \texttt{int} \mid (x \leq \nu) \wedge (y \leq \nu)\}$$

Outline

Inspiration from the work of Freeman and Pfenning, 1991 $\leadsto$ refinement type system $+$ intersection types.

We restrict our intersection to **types of the same form**: differences at refinement expressions level, only.

We use the restriction of Liquid Types: refinement expressions are **collected from a set $\mathbb{Q}$ $+$ decidable** subtyping.

The type language:

$$\begin{array}{lcl} T & ::= & \alpha \mid \{\nu : B \mid \phi\} \mid x : T_1 \to T_2 \mid T_1 \cap T_2 \\ \widetilde{\sigma} & ::= & T \mid \forall \alpha . \widetilde{\sigma} \\ \tau & ::= & B \mid \alpha \mid \tau_1 \to \tau_2 \\ \sigma & ::= & \widetilde{\sigma} :: \tau \end{array}$$

To ensure our intersections are only of types of the same form:

$$\frac{}{\alpha :: \alpha}\ {}^{::\text{-Var}} \qquad \frac{T_1 :: \tau \qquad T_2 :: \tau'}{T_1 \to T_2 :: \tau \to \tau'}\ {}^{::\text{-Fun}} \qquad \frac{}{\{\nu : B \mid \phi\} :: B}\ {}^{::\text{-Ref}}$$

$$\frac{\widetilde{\sigma} :: \tau}{\forall \alpha . \widetilde{\sigma} :: \tau}\ {}^{::\text{-}\forall} \qquad \frac{T_i :: \tau\ (\forall i.\ 1 \leq i \leq n)}{T_1 \cap \ldots \cap T_n :: \tau}\ {}^{::\text{-}\cap}$$

Our system presents the traditional rules for refinement type checking, plus the following rule:

$$\textsc{Intersect}\quad \frac{\Gamma \vdash^{\cap} M : \sigma \qquad \Gamma \vdash^{\cap} M : \sigma' \qquad \sigma \cap \sigma' :: \tau}{\Gamma \vdash^{\cap} M : \sigma \cap \sigma'}$$

Decidable notion of subtyping:

$$\prec\text{-Base}\quad \frac{\texttt{Valid}([\![\Gamma]\!] \wedge ([\![\phi_1]\!] \wedge \ldots \wedge [\![\phi_n]\!]) \Rightarrow ([\![\phi'_1]\!] \wedge \ldots \wedge [\![\phi'_m]\!]))}{\Gamma \vdash^{\cap} \{\nu : B \mid \phi_1\} \cap \ldots \cap \{\nu : B \mid \phi_n\} \prec \{\nu : B \mid \phi'_1\} \cap \ldots \cap \{\nu : B \mid \phi'_m\}}$$

$[\![\cdot]\!] \rightsquigarrow$ embedding of environments and formulas to a decidable logic of **equality**, **uninterpreted functions** and **linear arithmetic**.

Example with intersections

With our system, we are able to derive very precise types, $\Gamma = x : \{\nu \geq 0\}$:

$$\mathcal{D}_1' : \quad \dfrac{\text{Const}\ \dfrac{}{\Gamma \vdash^{\cap}_{\mathbb{Q}} - : (y : \mathrm{int} \to \{\nu = -y\})} \qquad \text{Var-B}\ \dfrac{\Gamma(x) = \{\nu \geq 0\}}{\Gamma \vdash^{\cap}_{\mathbb{Q}} x : \{\nu = x\}} \qquad \dfrac{\mathrm{Valid}(x \geq 0 \wedge \nu = x \Rightarrow \top)}{\Gamma \vdash^{\cap} \{\nu = x\} \prec \mathrm{int}}\ \prec\text{-Base}}{\Gamma \vdash^{\cap}_{\mathbb{Q}} x : \mathrm{int}}\ \text{Sub}}{\Gamma \vdash^{\cap}_{\mathbb{Q}} -x : \{\nu = -x\}}$$

$$\mathcal{D}_1 : \quad \dfrac{\dfrac{\mathcal{D}_1' \qquad \dfrac{\mathrm{Valid}(x \geq 0 \wedge \nu = -x \Rightarrow \nu \leq 0)}{\Gamma \vdash^{\cap} \{\nu = -x\} \prec \{\nu \leq 0\}}\ \prec\text{-Base}}{\Gamma \vdash^{\cap}_{\mathbb{Q}} -x : \{\nu \leq 0\}}\ \text{Sub}}{\vdash^{\cap}_{\mathbb{Q}} \lambda x. -x : (x : \{\nu \geq 0\} \to \{\nu \leq 0\})}\ \text{Fun}$$

Naming the derivation for $\vdash \lambda x. -x : (x : \{\nu \leq 0\} \to \{\nu \geq 0\})$ as $\mathcal{D}_2$ we have:

$$\dfrac{\mathcal{D}_1 \qquad \mathcal{D}_2}{\vdash^{\cap}_{\mathbb{Q}} \lambda x. -x : (x : \{\nu \geq 0\} \to \{\nu \leq 0\}) \cap (x : \{\nu \leq 0\} \to \{\nu \geq 0\})}\ \text{Intersect}$$

Inference for Liquid Intersection Types:

1. Damas-Milner type inference;
2. constraint generation:
   - well-formedness;
   - subtyping.
3. constraint solving: iterative removal of intersections in types.

$$
\begin{array}{lcl}
\texttt{Infer}(\Gamma, x, \mathbb{Q}) & = & \text{if } \mathcal{W}(\texttt{Shape}(\Gamma), x) = B \text{ then } \{v : B \mid v = x\} \\
 & & \text{else } \Gamma(x) \\
\texttt{Infer}(\Gamma, c, \mathbb{Q}) & = & ty(c) \\
\texttt{Infer}(\Gamma, \lambda x.M, \mathbb{Q}) & = & \text{let } (x : \sigma_1 \to \sigma_1') \cap \ldots \cap (x : \sigma_n \to \sigma_n') = \texttt{Fresh}(\mathcal{W}(\texttt{Shape}(\Gamma), \lambda x.M), \mathbb{Q}) \text{ in} \\
 & & \text{let } \sigma_i'' = \texttt{Infer}(\Gamma; x : \sigma_i, M, \mathbb{Q}) \text{ in} \\
 & & \text{let } \mathcal{A} = \bigcap \left\{ (\mathbf{x : \sigma_j \to \sigma_j'}) \mid \Gamma \vdash^{\cap} (\mathbf{x : \sigma_1 \to \sigma_1'}) \cap \ldots \cap (\mathbf{x : \sigma_n \to \sigma_n'}) \right\} \text{ in} \\
 & & \bigcap \left\{ (x : \sigma_k \to \sigma_k') \mid x : \sigma_k \to \sigma_k' \in \mathcal{A}, \Gamma; x : \sigma_k \vdash_{\mathbb{Q}}^{\cap} \sigma_k'' \prec \sigma_k' \right\} \\
\texttt{Infer}(\Gamma, MN, \mathbb{Q}) & = & \text{let } (x : \sigma_1 \to \sigma_1') \cap \ldots \cap (x : \sigma_n \to \sigma_n') = \texttt{Infer}(\Gamma, M, \mathbb{Q}) \text{ in} \\
 & & \text{let } \sigma = \texttt{Infer}(\Gamma, N, \mathbb{Q}) \text{ in} \\
 & & \bigcap[\mathbf{N/x}] \left\{ \sigma_i' \mid \Gamma \vdash_{\mathbb{Q}}^{\cap} \sigma \prec \sigma_i \right\} \\
\texttt{Infer}(\Gamma, \text{let } x = M \text{ in } N, \mathbb{Q}) & = & \texttt{Fresh}(\mathcal{W}(\texttt{Shape}(\Gamma), \text{let } x = M \text{ in } N), \mathbb{Q}) \text{ in} \\
 & & \text{let } \sigma_1 = \texttt{Infer}(\Gamma, M, \mathbb{Q}) \text{ in} \\
 & & \text{let } \sigma_2 = \texttt{Infer}(\Gamma; x : \sigma_1, N, \mathbb{Q}) \text{ in} \\
 & & \text{let } \mathcal{A} = \bigcap \{ \sigma_i' \mid \Gamma \vdash^{\cap} \sigma \} \text{ in} \\
 & & \bigcap \left\{ \sigma'' \mid \sigma'' \in \mathcal{A}, \Gamma; x : \sigma_1 \vdash_{\mathbb{Q}}^{\cap} \sigma_2 \prec \sigma'' \right\} \\
\texttt{Infer}(\Gamma, [\Lambda\alpha]M, \mathbb{Q}) & = & \text{let } \sigma = \texttt{Infer}(\Gamma, M, \mathbb{Q}) \text{ in} \\
 & & \forall\alpha.\sigma \\
\texttt{Infer}(\Gamma, [\tau]M, \mathbb{Q}) & = & \text{let } \tau' = \texttt{Fresh}(\tau, \mathbb{Q}) \text{ in} \\
 & & \text{let } \forall\alpha.\sigma = \texttt{Infer}(\Gamma, M, \mathbb{Q}) \text{ in} \\
 & & \texttt{let } \mathcal{A} = \bigcap \{ \tau_i' \mid \Gamma \vdash^{\cap} \tau' \} \text{ in} \\
 & & \sigma[\mathcal{A}/\alpha]
\end{array}
$$

Figure 3: Type inference algorithm

Let $\mathbb{Q} = \{\nu \geq 0, \nu \leq 0, y = 5\}$. Inference for the term $\lambda x. -x$ with $\Gamma = \emptyset$:

1. Algorithm first generates the type:

$$
\begin{array}{l}
(x : \{\nu \geq 0\} \to \{\nu \geq 0\}) \cap \\
(x : \{\nu \geq 0\} \to \{\nu \leq 0\}) \cap \\
(x : \{\nu \leq 0\} \to \{\nu \geq 0\}) \cap \\
(x : \{\nu \leq 0\} \to \{\nu \leq 0\}) \cap \\
(x : \{\nu \geq 0\} \to \{y = 5\}) \cap \\
(x : \{\nu \leq 0\} \to \{y = 5\}) \cap \\
(x : \{y = 5\} \to \{\nu \geq 0\}) \cap \\
(x : \{y = 5\} \to \{\nu \leq 0\}) \cap \\
(x : \{y = 5\} \to \{y = 5\})
\end{array}
$$

2. With well-formedness constraints (no variable y is in scope):

$$
\begin{array}{l}
(x : \{\nu \geq 0\} \to \{\nu \geq 0\}) \cap \\
(x : \{\nu \geq 0\} \to \{\nu \leq 0\}) \cap \\
(x : \{\nu \leq 0\} \to \{\nu \geq 0\}) \cap \\
(x : \{\nu \leq 0\} \to \{\nu \leq 0\})
\end{array}
$$

3. Finally, because of subtyping relations:

$$
\begin{array}{l}
(x : \{\nu \geq 0\} \to \{\nu \leq 0\}) \cap \\
(x : \{\nu \leq 0\} \to \{\nu \geq 0\})
\end{array}
$$

Outline

Theorem (Subject reduction)

If $\Gamma \vdash^{\cap} M : \sigma$ *and* $M \rightsquigarrow N$ *then* $\Gamma \vdash N : \sigma$.

Theorem (Soundness of inference)

If $\texttt{Infer}(\Gamma, M, \mathbb{Q}) = \sigma$ *then* $\Gamma \vdash M : \sigma$.

Completeness of inference.

Design of a refinement type system for rank-2 intersection types.

Thank you for your attention