A Type- and Control-Flow Analysis for System F

Matthew Fluet
mtf@cs.rit.edu

Computer Science Department
Rochester Institute of Technolgy

August 31, 2012
IFL'12

Introduction

Control-flow analysis

A compile-time approximation of the "flow" of functions in program: which functions might be bound to a given variable at run time.

- an enabling analysis for the compilation of functional languages
  - because control flow is not syntactically apparent
- typically formulated for dynamically- or simply-typed languages

- popular statically-typed functional languages are richly-typed
- System F (and extensions) are popular intermediate languages

Seek a control-flow analysis formulated for System F.

Exploit well-typedness to obtain more precise control-flow information.

Example 1

$f1 = \lambda x1.\ \cdots$
$f2 = \lambda x2.\ \cdots$

$id = \lambda x.\ x$

$res1 = id\ f1$
$res2 = id\ f2$

Example 1

$f1 = \lambda x1 . \ \cdots$
$f2 = \lambda x2 . \ \cdots$

$id = \lambda x . \ x$

$res1 = id\ f1$
$res2 = id\ f2$

	0CFA
$\hat{\rho}(x)$	$\{ f1 , f2 \}$
$\hat{\rho}(res1)$	$\{ f1 , f2 \}$
$\hat{\rho}(res2)$	$\{ f1 , f2 \}$

Example 1

$\mathsf{f1} : \mathsf{Int} \to \mathsf{Int} = \lambda \mathsf{x1} : \mathsf{Int} .\ \cdots$
$\mathsf{f2} : \mathsf{Bool} \to \mathsf{Bool} = \lambda \mathsf{x2} : \mathsf{Bool} .\ \cdots$

$\mathsf{id} : \forall \alpha .\ \alpha \to \alpha = \Lambda \alpha .\ \lambda \mathsf{x} : \alpha .\ \mathsf{x}$

$\mathsf{res1} : \mathsf{Int} \to \mathsf{Int} = \mathsf{id}\ [\mathsf{Int} \to \mathsf{Int}]\ \mathsf{f1}$
$\mathsf{res2} : \mathsf{Bool} \to \mathsf{Bool} = \mathsf{id}\ [\mathsf{Bool} \to \mathsf{Bool}]\ \mathsf{f2}$

	0CFA
$\hat{\rho}(\mathsf{x}^{\alpha})$	$\{ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Bool}} \}$
$\hat{\rho}(\mathsf{res1}^{\mathsf{Int}\to\mathsf{Int}})$	$\{ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Bool}} \}$
$\hat{\rho}(\mathsf{res2}^{\mathsf{Bool}\to\mathsf{Bool}})$	$\{ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Bool}} \}$

Example 1

$\mathsf{f1} : \mathsf{Int} \to \mathsf{Int} = \lambda \mathsf{x1} : \mathsf{Int} .\ \cdots$
$\mathsf{f2} : \mathsf{Bool} \to \mathsf{Bool} = \lambda \mathsf{x2} : \mathsf{Bool} .\ \cdots$

$\mathsf{id} : \forall \alpha .\ \alpha \to \alpha = \Lambda \alpha .\ \lambda \mathsf{x} : \alpha .\ \mathsf{x}$

$\mathsf{res1} : \mathsf{Int} \to \mathsf{Int} = \mathsf{id}\ [\mathsf{Int} \to \mathsf{Int}]\ \mathsf{f1}$
$\mathsf{res2} : \mathsf{Bool} \to \mathsf{Bool} = \mathsf{id}\ [\mathsf{Bool} \to \mathsf{Bool}]\ \mathsf{f2}$

	0CFA then type-filter
$\hat{\rho}(\mathsf{x}^{\alpha})$	$\{ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Bool}} \}$
$\hat{\rho}(\mathsf{res1}^{\mathsf{Int}\to\mathsf{Int}})$	$\{ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}} \}$
$\hat{\rho}(\mathsf{res2}^{\mathsf{Bool}\to\mathsf{Bool}})$	$\{ \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Bool}} \}$

Example 2

$$\mathsf{f1}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x1}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g1}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y1}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g1}$$
$$\mathsf{f2}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x2}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g2}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y2}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g2}$$
$$\mathsf{f3}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x3}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g3}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y3}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g3}$$

$$\mathsf{id}:\forall\alpha.\ \alpha\to\alpha = \Lambda\alpha.\ \lambda \mathsf{x}:\alpha.\ \mathsf{x}$$

$$\begin{aligned}
&\mathsf{h}:\forall\beta.\ (\beta\to\mathsf{Int}\to\mathsf{Int})\to\beta\to\mathsf{Int}\to\mathsf{Int} = \\
&\quad \Lambda\beta.\ \lambda \mathsf{f}:\beta\to\mathsf{Int}\to\mathsf{Int}.\ \lambda \mathsf{y}:\beta. \\
&\quad \texttt{let}\ \mathsf{ff}:\beta\to\mathsf{Int}\to\mathsf{Int} = \mathsf{id}\ [\beta\to\mathsf{Int}\to\mathsf{Int}]\ \mathsf{f} \\
&\qquad\ \ \mathsf{g}:\mathsf{Int}\to\mathsf{Int} = \mathsf{ff}\ \mathsf{y} \\
&\quad \texttt{in}\ \mathsf{g} \\
&\cdots
\end{aligned}$$

Example 2

$$\mathsf{f1}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x1}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g1}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y1}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g1}$$
$$\mathsf{f2}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x2}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g2}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y2}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g2}$$
$$\mathsf{f3}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x3}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g3}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y3}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g3}$$

$$\mathsf{id}:\forall\alpha.\ \alpha\to\alpha = \Lambda\alpha.\ \lambda \mathsf{x}:\alpha.\ \mathsf{x}$$

$$\begin{array}{l}\mathsf{h}:\forall\beta.\ (\beta\to\mathsf{Int}\to\mathsf{Int})\to\beta\to\mathsf{Int}\to\mathsf{Int} = \\ \quad \Lambda\beta.\ \lambda \mathsf{f}:\beta\to\mathsf{Int}\to\mathsf{Int}.\ \lambda \mathsf{y}:\beta. \\ \quad \texttt{let}\ \mathsf{ff}:\beta\to\mathsf{Int}\to\mathsf{Int} = \mathsf{id}\ [\beta\to\mathsf{Int}\to\mathsf{Int}]\ \mathsf{f} \\ \quad\quad \mathsf{g}:\mathsf{Int}\to\mathsf{Int} = \mathsf{ff}\ \mathsf{y} \\ \quad \texttt{in}\ \mathsf{g} \\ \cdots\end{array}$$

	0CFA
$\hat{\rho}(\mathsf{x}^{\alpha})$	$\{\ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}},\ \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}},\ \mathsf{f3}^{\mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}}\ \}$
$\hat{\rho}(\mathsf{f}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}},\ \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}\ \}$
$\hat{\rho}(\mathsf{ff}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\ \mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}},\ \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}},\ \mathsf{f3}^{\mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}}\ \}$
$\hat{\rho}(\mathsf{g}^{\mathsf{Int}\to\mathsf{Int}})$	$\{\ \mathsf{g1}^{\mathsf{Int}\to\mathsf{Int}},\ \mathsf{g2}^{\mathsf{Int}\to\mathsf{Int}},\ \mathsf{g3}^{\mathsf{Int}\to\mathsf{Int}}\ \}$

Example 2

$\mathsf{f1}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x1}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g1}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y1}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g1}$
$\mathsf{f2}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x2}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g2}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y2}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g2}$
$\mathsf{f3}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x3}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g3}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y3}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g3}$

$\mathsf{id}:\forall\alpha.\ \alpha\to\alpha = \Lambda\alpha.\ \lambda \mathsf{x}:\alpha.\ \mathsf{x}$

$\mathsf{h}:\forall\beta.\ (\beta\to\mathsf{Int}\to\mathsf{Int})\to\beta\to\mathsf{Int}\to\mathsf{Int} =$
$\quad\Lambda\beta.\ \lambda \mathsf{f}:\beta\to\mathsf{Int}\to\mathsf{Int}.\ \lambda \mathsf{y}:\beta.$
$\quad\texttt{let}\ \mathsf{ff}:\beta\to\mathsf{Int}\to\mathsf{Int} = \mathsf{id}\ [\beta\to\mathsf{Int}\to\mathsf{Int}]\ \mathsf{f}$
$\qquad\mathsf{g}:\mathsf{Int}\to\mathsf{Int} = \mathsf{ff}\ \mathsf{y}$
$\quad\texttt{in}\ \mathsf{g}$
$\ldots$

	0CFA
$\hat{\rho}(\mathsf{x}^{\alpha})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f3}^{\mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{f}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{ff}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f3}^{\mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{g}^{\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{g1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{g2}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{g3}^{\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\phi}(\alpha)$	$\{\beta\to\mathsf{Int}\to\mathsf{Int}, \mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}\}$
$\hat{\phi}(\beta)$	$\{\mathsf{Int}, \mathsf{Bool}\}$

Example 2

$$\mathsf{f1}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x1}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g1}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y1}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g1}$$
$$\mathsf{f2}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x2}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g2}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y2}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g2}$$
$$\mathsf{f3}:\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{x3}:\mathsf{Int}.\ \texttt{let}\ \mathsf{g3}:\mathsf{Int}\to\mathsf{Int} = \lambda \mathsf{y3}:\mathsf{Int}.\ \cdots\ \texttt{in}\ \mathsf{g3}$$

$$\mathsf{id}:\forall\alpha.\ \alpha\to\alpha = \Lambda\alpha.\ \lambda \mathsf{x}:\alpha.\ \mathsf{x}$$

$$\mathsf{h}:\forall\beta.\ (\beta\to\mathsf{Int}\to\mathsf{Int})\to\beta\to\mathsf{Int}\to\mathsf{Int} =$$
$$\Lambda\beta.\ \lambda \mathsf{f}:\beta\to\mathsf{Int}\to\mathsf{Int}.\ \lambda \mathsf{y}:\beta.$$
$$\texttt{let}\ \mathsf{ff}:\beta\to\mathsf{Int}\to\mathsf{Int} = \mathsf{id}\ [\beta\to\mathsf{Int}\to\mathsf{Int}]\ \mathsf{f}$$
$$\mathsf{g}:\mathsf{Int}\to\mathsf{Int} = \mathsf{ff}\ \mathsf{y}$$
$$\texttt{in}\ \mathsf{g}$$
$$\ldots$$

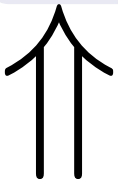
	0T&CFA
$\hat{\rho}(\mathsf{x}^{\alpha})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f3}^{\mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{f}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{ff}^{\beta\to\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{f1}^{\mathsf{Int}\to\mathsf{Int}\to\mathsf{Int}}, \mathsf{f2}^{\mathsf{Bool}\to\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\rho}(\mathsf{g}^{\mathsf{Int}\to\mathsf{Int}})$	$\{\mathsf{g1}^{\mathsf{Int}\to\mathsf{Int}}, \mathsf{g2}^{\mathsf{Int}\to\mathsf{Int}}\}$
$\hat{\phi}(\alpha)$	$\{\beta\to\mathsf{Int}\to\mathsf{Int}, \mathsf{Str}\to\mathsf{Int}\to\mathsf{Int}\}$
$\hat{\phi}(\beta)$	$\{\mathsf{Int}, \mathsf{Bool}\}$

Introduction

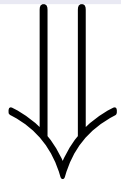
A type- and control-flow analysis for System F

Type-flow analysis

A compile-time approximation of the "flow" of types in program: which types might be bound to a given type variable at run time



determines type abstractions flowing to type applications



rejects flows incompatible with static typing

Control-flow analysis

A compile-time approximation of the "flow" of functions in program: which functions might be bound to a given variable at run time

System F (in Administrative Normal Form (ANF))

$$\mathit{Type} \ni \tau ::= \tau \to \tau \mid \alpha \mid \forall\alpha.\, \tau$$

$$\begin{aligned}
\mathit{Exp} \ni e ::= {} & x \\
\mid {} & \texttt{let } x{:}\tau_x = \mu f{:}\tau_f.\lambda z{:}\tau_z.e_b \texttt{ in } e \\
\mid {} & \texttt{let } x{:}\tau_x = \mu f{:}\tau_f.\Lambda\beta.e_b \texttt{ in } e \\
\mid {} & \texttt{let } x{:}\tau_x = x_f\ x_a \texttt{ in } e \\
\mid {} & \texttt{let } x{:}\tau_x = x_f\ [\tau_a] \texttt{ in } e
\end{aligned}$$

- Type system — standard
- Operational semantics — C_aEK abstract machine

Type- and Control-Flow Analysis

$$\begin{array}{lcll} \mathit{AType} & \ni & \hat{\pi} & ::= \quad \tau \\ \mathit{ATEnv} & \ni & \hat{\phi} & \in \quad \mathit{TyVar} \to \mathcal{P}(\mathit{AType}) \end{array} \qquad \begin{array}{lcll} \mathit{AValue} & \ni & \hat{w} & ::= \quad \mu f{:}\tau_f\,.\,\lambda z{:}\tau_z\,.\,e_b \mid \\ & & & \quad\ \mu f{:}\tau_f\,.\,\Lambda\beta\,.\,e_b \\ \mathit{AEnv} & \ni & \hat{\rho} & \in \quad \mathit{Var} \to \mathcal{P}(\mathit{AValue}) \end{array}$$

$\boxed{\hat{\phi}; \hat{\rho} \vDash e}$ "$\hat{\phi}$ and $\hat{\rho}$ are an *acceptable* type- and control-flow analysis for e"

Type- and Control-Flow Analysis

$$\begin{array}{llllll} \mathit{AType} & \ni & \hat{\pi} & ::= & \tau & \\ \mathit{ATEnv} & \ni & \hat{\phi} & \in & \mathit{TyVar} \to \mathcal{P}(\mathit{AType}) & \end{array} \qquad \begin{array}{llllll} \mathit{AValue} & \ni & \hat{w} & ::= & \mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b \mid \\ & & & & \mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b \\ \mathit{AEnv} & \ni & \hat{\rho} & \in & \mathit{Var} \to \mathcal{P}(\mathit{AValue}) \end{array}$$

$\boxed{\hat{\phi};\hat{\rho} \vDash e}$ "$\hat{\phi}$ and $\hat{\rho}$ are an *acceptable* type- and control-flow analysis for e"

$$\frac{}{\hat{\phi};\hat{\rho} \vDash x}$$

$$\frac{\hat{\rho}(f) \supseteq \{\mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b\} \qquad \hat{\phi};\hat{\rho} \vDash e_b \\ \hat{\rho}(x) \supseteq \{\mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b\} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let}\ x:\tau_x = \mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b\ \texttt{in}\ e}$$

$$\frac{\hat{\rho}(f) \supseteq \{\mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b\} \qquad \hat{\phi};\hat{\rho} \vDash e_b \\ \hat{\rho}(x) \supseteq \{\mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b\} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let}\ x:\tau_x = \mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b\ \texttt{in}\ e}$$

Type- and Control-Flow Analysis

$$\mathit{AType} \ni \hat{\pi} ::= \tau \qquad \mathit{AValue} \ni \hat{w} ::= \mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b \mid \mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b$$

$$\mathit{ATEnv} \ni \hat{\phi} \in \mathit{TyVar} \to \mathcal{P}(\mathit{AType}) \qquad \mathit{AEnv} \ni \hat{\rho} \in \mathit{Var} \to \mathcal{P}(\mathit{AValue})$$

$\boxed{\hat{\phi};\hat{\rho} \vDash e}$ "$\hat{\phi}$ and $\hat{\rho}$ are an *acceptable* type- and control-flow analysis for e"

$$\frac{\bigwedge_{\mu f:\tau_f\,.\,\lambda z:\tau_z\,.\,e_b \in \hat{\rho}(x_f)} \begin{pmatrix} \hat{\rho}(z) \supseteq \hat{\rho}(x_a) \wedge \\ \hat{\rho}(x) \supseteq \hat{\rho}(\mathsf{last}(e_b)) \end{pmatrix} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let } x:\tau_x = x_f\ x_a \texttt{ in } e}$$

$$\frac{\bigwedge_{\mu f:\tau_f\,.\,\Lambda\beta\,.\,e_b \in \hat{\rho}(x_f)} \begin{pmatrix} \hat{\phi}(\beta) \supseteq \{\tau_a\} \wedge \\ \hat{\rho}(x) \supseteq \hat{\rho}(\mathsf{last}(e_b)) \end{pmatrix} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let } x:\tau_x = x_f\ [\tau_a] \texttt{ in } e}$$

Type- and Control-Flow Analysis

$$\mathit{AType} \ni \hat{\pi} ::= \tau \qquad \mathit{AValue} \ni \hat{w} ::= \mu f:\tau_f.\lambda z:\tau_z.e_b \mid \mu f:\tau_f.\Lambda\beta.e_b$$

$$\mathit{ATEnv} \ni \hat{\phi} \in \mathit{TyVar} \to \mathcal{P}(\mathit{AType}) \qquad \mathit{AEnv} \ni \hat{\rho} \in \mathit{Var} \to \mathcal{P}(\mathit{AValue})$$

$\boxed{\hat{\phi};\hat{\rho} \vDash e}$ "$\hat{\phi}$ and $\hat{\rho}$ are an *acceptable* type- and control-flow analysis for e"

$$\frac{\bigwedge_{\mu f:\tau_f.\lambda z:\tau_z.e_b \in \hat{\rho}(x_f)} \begin{pmatrix} \hat{\rho}(z) \supseteq \{\hat{w}_a \in \hat{\rho}(x_a) \mid \vdash \hat{\phi} \Rrightarrow \hat{w}_a :\approx \tau_z\} \wedge \\ \hat{\rho}(x) \supseteq \{\hat{w}_b \in \hat{\rho}(\mathsf{last}(e_b)) \mid \vdash \hat{\phi} \Rrightarrow \hat{w}_b :\approx \tau_x\} \end{pmatrix} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let } x:\tau_x = x_f\ x_a \texttt{ in } e}$$

$$\frac{\bigwedge_{\mu f:\tau_f.\Lambda\beta.e_b \in \hat{\rho}(x_f)} \begin{pmatrix} \hat{\phi}(\beta) \supseteq \{\tau_a\} \wedge \\ \hat{\rho}(x) \supseteq \{\hat{w}_b \in \hat{\rho}(\mathsf{last}(e_b)) \mid \vdash \hat{\phi} \Rrightarrow \hat{w}_b :\approx \tau_x\} \end{pmatrix} \qquad \hat{\phi};\hat{\rho} \vDash e}{\hat{\phi};\hat{\rho} \vDash \texttt{let } x:\tau_x = x_f\ [\tau_a] \texttt{ in } e}$$

Type Compatibility

$\boxed{\vdash \hat{\phi} \twoheadrightarrow \hat{w} :\approx \hat{\pi}}$ "the type of $\hat{w}$ is *compatible* with $\hat{\pi}$ under $\hat{\phi}$"

$$\frac{\emptyset \vdash \hat{\phi} \twoheadrightarrow \tau_f \approx \hat{\pi}}{\vdash \hat{\phi} \twoheadrightarrow \mu f : \tau_f \,.\, \lambda z : \tau_z \,.\, e_b :\approx \hat{\pi}} \qquad \frac{\emptyset \vdash \hat{\phi} \twoheadrightarrow \tau_f \approx \hat{\pi}}{\vdash \hat{\phi} \twoheadrightarrow \mu f : \tau_f \,.\, \Lambda \beta \,.\, e_b :\approx \hat{\pi}}$$

$\boxed{\Delta \vdash \hat{\phi} \twoheadrightarrow \hat{\pi}_1 \approx \hat{\pi}_2}$ "$\hat{\pi}_1$ and $\hat{\pi}_2$ are *compatible* under $\hat{\phi}$"

$$\frac{\hat{\pi}_1' \in \hat{\phi}(\alpha_1) \qquad \mathsf{FTV}(\hat{\pi}_2) \cap \mathsf{dom}(\Delta) = \emptyset \qquad \emptyset \vdash \hat{\phi} \twoheadrightarrow \hat{\pi}_1' \approx \hat{\pi}_2}{\Delta \vdash \hat{\phi} \twoheadrightarrow \alpha_1 \approx \hat{\pi}_2} \qquad \frac{\hat{\pi}_2' \in \hat{\phi}(\alpha_2) \qquad \mathsf{FTV}(\hat{\pi}_1) \cap \mathsf{dom}(\Delta) = \emptyset \qquad \emptyset \vdash \hat{\phi} \twoheadrightarrow \hat{\pi}_1 \approx \hat{\pi}_2'}{\Delta \vdash \hat{\phi} \twoheadrightarrow \hat{\pi}_1 \approx \alpha_2}$$

$$\frac{\Delta \vdash \hat{\phi} \twoheadrightarrow \tau_{z1} \approx \tau_{z2} \qquad \Delta \vdash \hat{\phi} \twoheadrightarrow \tau_{b1} \approx \tau_{b2}}{\Delta \vdash \hat{\phi} \twoheadrightarrow \tau_{z1} \to \tau_{b1} \approx \tau_{z2} \to \tau_{b2}}$$

$$\frac{\Delta(\alpha) = \star}{\Delta \vdash \hat{\phi} \twoheadrightarrow \alpha \approx \alpha} \qquad \frac{\Delta, \alpha{:}\star \vdash \hat{\phi} \twoheadrightarrow \tau_{b1} \approx \tau_{b2}}{\Delta \vdash \hat{\phi} \twoheadrightarrow \forall \alpha.\ \tau_{b1} \approx \forall \alpha.\ \tau_{b2}}$$

Flow Soundness

Acceptable abstract environments approximate concrete environments

(for well-typed programs)

Flow Soundness

Acceptable abstract environments approximate concrete environments
(for well-typed programs)

Theorem (Flow Soundness)

If $\emptyset;\emptyset \vdash e : \tau$, $\hat{\phi};\hat{\rho} \vDash e$, *and* $\langle e;\emptyset;\emptyset;\bullet\rangle \longrightarrow^* \langle e';\phi';\rho';\kappa'\rangle$,
then $|\phi'| \sqsubseteq \hat{\phi}$ *and* $|\rho'| \sqsubseteq \hat{\rho}$.

- syntactic preservation proof (with $\vdash \langle e;\phi;\rho;\kappa\rangle : \tau$ and $\hat{\phi};\hat{\rho} \vDash \langle e;\phi;\rho;\kappa\rangle$)

Lemma (Expansion Type Equality implies Analysis-Time Type Compatibility)

If $\vdash \langle \tau_1;\phi_1\rangle \Rrightarrow \tau$, $\vdash \langle \tau_2;\phi_2\rangle \Rrightarrow \tau$, $\hat{\phi} \vDash \phi_1$, *and* $\hat{\phi} \vDash \phi_2$, *then* $\emptyset \vdash \hat{\phi} \Rrightarrow \tau_1 \approx\!\!\!\!\approx \tau_2$.

- $\vdash \langle \tau_1;\phi_1\rangle \Rrightarrow \tau$ and $\vdash \langle \tau_2;\phi_2\rangle \Rrightarrow \tau$ from well-typedness

Existence of Minimum, Finite Flows

There are “best” acceptable abstract environments

Existence of Minimum, Finite Flows

There are "best" acceptable abstract environments

Theorem (Minimum Flows Exist)

For all programs e,
there exist minimum abstract environments $\hat{\phi}_{\mathsf{min}}$ *and* $\hat{\rho}_{\mathsf{min}}$
such that $\hat{\phi}_{\mathsf{min}}; \hat{\rho}_{\mathsf{min}} \vDash e$.

- acceptable abstract environments are preserved by meets

Existence of Minimum, Finite Flows

There are "best" acceptable abstract environments

Theorem (Finite Flows Exist)

For all programs e, $\hat{\phi}^e_\top; \hat{\rho}^e_\top \vDash e$.

- $\hat{\phi}^e_\top$ maps every Λ-bound type variable in e (a finite set) to the (finite) set of all type expressions in e
- $\hat{\rho}^e_\top$ maps every `let`-, μ-, and λ-bound variable in e (a finite set) to the (finite) set of all Λ- and λ-expressions in e

Computability/Decidability of Flows

Are the "best" acceptable abstract environments computable?

Computability/Decidability of Flows

Are the "best" acceptable abstract environments computable?

Is the acceptability of abstract environments decidable?

- To show $\hat{\phi}; \hat{\rho} \vDash e$,
  must show inclusions of the form $\hat{\rho}(z) \supseteq \{\hat{w} \in \hat{\rho}(x) \mid \vdash \hat{\phi} \twoheadrightarrow \hat{w} :\approx \tau_z\}$
  - For each $\hat{w} \in \hat{\rho}(x)$ but $\hat{w} \notin \hat{\rho}(z)$,
    must show $\vdash \hat{\phi} \twoheadrightarrow \hat{w} :\approx \tau_z$ is not derivable.

Computability/Decidability of Flows

Are the "best" acceptable abstract environments computable?

Is the acceptability of abstract environments decidable?

- To show $\hat{\phi}; \hat{\rho} \vDash e$,
  must show inclusions of the form $\hat{\rho}(z) \supseteq \{\hat{w} \in \hat{\rho}(x) \mid \vdash \hat{\phi} \Rrightarrow \hat{w} :\approx \tau_z\}$
  - For each $\hat{w} \in \hat{\rho}(x)$ but $\hat{w} \notin \hat{\rho}(z)$,
    must show $\vdash \hat{\phi} \Rrightarrow \hat{w} :\approx \tau_z$ is not derivable.

Is type compatibility decidable?

Decidability of Type Compatibility

If $\hat{\phi}^{\ddagger}(\alpha) = \{\, \mathsf{int} \to \mathsf{int}\,,\ \mathsf{int} \to \alpha \,\}$ and $\hat{\phi}^{\ddagger}(\beta) = \{\, \mathsf{int} \to \mathsf{bool}\,,\ \mathsf{int} \to \beta \,\}$,
then is $\emptyset \vdash \hat{\phi}^{\ddagger} \Rrightarrow \mathsf{int} \to \beta \approxeq \alpha$ derivable?

Decidability of Type Compatibility

If $\hat{\phi}^{\ddagger}(\alpha) = \{\, \mathsf{int} \to \mathsf{int}\,,\ \mathsf{int} \to \alpha \,\}$ and $\hat{\phi}^{\ddagger}(\beta) = \{\, \mathsf{int} \to \mathsf{bool}\,,\ \mathsf{int} \to \beta \,\}$,
then is $\emptyset \vdash \hat{\phi}^{\ddagger} \Rrightarrow \mathsf{int} \to \beta \approxeq \alpha$ derivable?

"Recursion" in abstract type environment foils exhaustive search.

Decidability of Type Compatibility

If $\hat{\phi}^{\ddagger}(\alpha) = \{\, \mathsf{int} \to \mathsf{int}\,,\ \mathsf{int} \to \alpha \,\}$ and $\hat{\phi}^{\ddagger}(\beta) = \{\, \mathsf{int} \to \mathsf{bool}\,,\ \mathsf{int} \to \beta \,\}$, then is $\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \alpha$ derivable?

"Recursion" in abstract type environment foils exhaustive search.

$$\dfrac{\mathsf{int} \to \alpha \in \hat{\phi}^{\ddagger}(\alpha) \qquad \dfrac{\dfrac{}{\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \mathsf{int} \approx\!\!\!\!\approx \mathsf{int}} \qquad \dfrac{\mathsf{int} \to \beta \in \hat{\phi}^{\ddagger}(\beta) \qquad \dfrac{\vdots}{\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \alpha}}{\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \beta \approx\!\!\!\!\approx \alpha}}{\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \mathsf{int} \to \alpha}}{\emptyset \vdash \hat{\phi}^{\ddagger} \twoheadrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \alpha}$$

Decidability of Type Compatibility

If $\hat{\phi}^{\ddagger}(\alpha) = \{\, \mathsf{int} \to \mathsf{int}\,,\ \mathsf{int} \to \alpha \,\}$ and $\hat{\phi}^{\ddagger}(\beta) = \{\, \mathsf{int} \to \mathsf{bool}\,,\ \mathsf{int} \to \beta \,\}$,
then is $\emptyset \vdash \hat{\phi}^{\ddagger} \Rrightarrow \mathsf{int} \to \beta \approxeq \alpha$ derivable?

Interpret abstract type environment as a regular-tree grammar.

Decidability of Type Compatibility

If $\hat{\phi}^{\ddagger}(\alpha) = \{\, \mathsf{int} \to \mathsf{int}\,,\ \mathsf{int} \to \alpha \,\}$ and $\hat{\phi}^{\ddagger}(\beta) = \{\, \mathsf{int} \to \mathsf{bool}\,,\ \mathsf{int} \to \beta \,\}$,
then is $\emptyset \vdash \hat{\phi}^{\ddagger} \Rrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \alpha$ derivable?

Interpret abstract type environment as a regular-tree grammar.

$$\begin{aligned} \mathcal{L}_{\hat{\phi}^{\ddagger}}(\mathsf{int} \to \beta) &= \{\mathsf{int} \to \mathsf{int} \to \mathsf{bool}, \mathsf{int} \to \mathsf{int} \to \mathsf{int} \to \mathsf{bool}, \ldots\} \\ \mathcal{L}_{\hat{\phi}^{\ddagger}}(\alpha) &= \{\mathsf{int} \to \mathsf{int}, \mathsf{int} \to \mathsf{int} \to \mathsf{int}, \mathsf{int} \to \mathsf{int} \to \mathsf{int} \to \mathsf{int}, \ldots\} \end{aligned}$$

$$\mathcal{L}_{\hat{\phi}^{\ddagger}}(\mathsf{int} \to \beta) \cap \mathcal{L}_{\hat{\phi}^{\ddagger}}(\alpha) = \emptyset$$

Intuitively, $\emptyset \vdash \hat{\phi}^{\ddagger} \Rrightarrow \mathsf{int} \to \beta \approx\!\!\!\!\approx \alpha$ is not derivable because there is no closed type generated by $\hat{\phi}^{\ddagger}$ from both $\mathsf{int} \to \beta$ and α.

Decidability of Type Compatibility

Type compatibility is decidable

Decidability of Type Compatibility

Type compatibility is decidable

Theorem (Analysis-Time Type Compatibility iff Languages Intersect)

$\emptyset \vdash \hat{\phi} \Rrightarrow \hat{\pi}_1 \approxeq \hat{\pi}_2$ *if and only if* $\mathcal{L}_{\hat{\phi}}(\hat{\pi}_1) \cap \mathcal{L}_{\hat{\phi}}(\hat{\pi}_2) \neq \emptyset$.

- regular-tree grammars are closed under intersection; $O(n^2)$
- emptiness of regular-tree grammars is decidable; $O(n^2)$

Computability of Flows

The “best” acceptable abstract environments are computable

Computability of Flows

The "best" acceptable abstract environments are computable

- Read analysis as a monotone function
  from abstract environments to abstract environments
- Least fixed point is the "best" acceptable abstract environments
- Least fixed point is computable using iteration

Related Work

- Need to distinguish between
  - flow analyses expressed as sophisticated type systems (many)
  - flow analyses of languages with sophisticated type systems (few)

Related Work

- Need to distinguish between
  - flow analyses expressed as sophisticated type systems (many)
  - flow analyses of languages with sophisticated type systems (few)

- Type-Directed Flow Analysis for Typed Intermediate Languages; Jagannathan, Weeks, & Wright (SAS'97)
  - limited to predicative System F with recursion
  - polyvariant analysis
  - diverges on programs using polymorphic recursion

- Type-sensitive Control-Flow Analysis; Reppy (ML'06)
  - Suggests mapping polymorphism to $\top$

Future Work

- Efficient computation of type- and control-flow analysis
- Extend to a polyvariant type- and control-flow analysis
- Extend to richer type systems: System F + GADTs; System F_ω

A Type- and Control-Flow Analysis for System F

Questions?