Open Closure Types

Gabriel Scherer, Jan Hoffmann

Gallium – INRIA

Background

Look at type systems that classify intentional properties of programs: resource usage, liveness analysis, information flow, etc.

Background

Look at type systems that classify intentional properties of programs: resource usage, liveness analysis, information flow, etc.

$$x{:}\texttt{List}(\texttt{int})^{1}, y{:}\texttt{bool}^{0} \vdash e : \tau$$

may mean, among others:

- e will not use y at runtime
- e will allocate memory linear in the size of x (list length)
- e will treat y as secure information requiring declassification

Lots of possible structures for the funny annotations (coeffects [POM12]).

Motivation

Existing type systems are good at first-order programs.

$$\texttt{append} : \texttt{List}(\texttt{A})^{1} * \texttt{List}(\texttt{A})^{0} \to \texttt{List}(\texttt{A})$$

[JHLH10] handles functions that take functions as parameter.

Motivation

Existing type systems are good at first-order programs.

$$\texttt{append} : \texttt{List}(\texttt{A})^{1} * \texttt{List}(\texttt{A})^{0} \to \texttt{List}(\texttt{A})$$

[JHLH10] handles functions that take functions as parameter.

This work attacks functions that **return** functions, eg. currification.

$$\texttt{append} : \texttt{List}(\texttt{A})^{?} \to (\texttt{List}(\texttt{A})^{0} \to \texttt{List}(\texttt{A}))$$

Motivation

Existing type systems are good at first-order programs.

$$\texttt{append} : \texttt{List}(\texttt{A})^{1} * \texttt{List}(\texttt{A})^{0} \to \texttt{List}(\texttt{A})$$

[JHLH10] handles functions that take functions as parameter.

This work attacks functions that **return** functions, eg. currification.

$$\texttt{append} : \texttt{List}(\texttt{A})^{?} \to (\texttt{List}(\texttt{A})^{0} \to \texttt{List}(\texttt{A}))$$

We propose **open closure types**

$$\texttt{append} : [](x : \texttt{List}(\texttt{A})^{0}) \to \Big([x{:}\texttt{List}(\texttt{A})^{1}](y{:}\texttt{List}(\texttt{A})^{0}) \to \texttt{List}(\texttt{A})\Big)$$

General Idea

Open types track how terms use variables from the context – we reason about it statically.

Work out the metatheory of the simplest interesting system:

- simply-typed lambda-calculus
- dead-simple annotations (booleans on inputs)
- call-by-value : only function types need context information

$$\frac{\Gamma^{\Phi}, x{:}\sigma^{\phi} \vdash e : \tau}{\Gamma^{0} \vdash \lambda x.e : [\Gamma^{\Phi}](x : \sigma^{\phi}) \to \tau}$$

Goal: Pave the way for more sophisticated stuff later.

Seems easy...

$$\textsc{Lam}\ \frac{\Gamma^{\Phi}, x{:}\sigma^{\phi} \vdash e : \tau}{\Gamma^{0} \vdash \lambda x.e : [\Gamma^{\Phi}](x : \sigma^{\phi}) \to \tau}$$

Seems easy...

$$\textsc{Lam}\ \frac{\Gamma^{\Phi}, x{:}\sigma^{\phi} \vdash e : \tau}{\Gamma^{0} \vdash \lambda x.e : [\Gamma^{\Phi}](x : \sigma^{\phi}) \to \tau}$$

$$\textsc{Var}\ \frac{\Gamma, x{:}\sigma, \Delta \vdash}{\Gamma^{0}, x{:}\sigma^{1}, \Delta^{0} \vdash x : \sigma}$$

Seems easy...

$$\textsc{Lam}\ \frac{\Gamma^{\Phi}, x{:}\sigma^{\phi} \vdash e : \tau}{\Gamma^{0} \vdash \lambda x.e : [\Gamma^{\Phi}](x : \sigma^{\phi}) \to \tau} \qquad \textsc{Var}\ \frac{\Gamma, x{:}\sigma, \Delta \vdash}{\Gamma^{0}, x{:}\sigma^{1}, \Delta^{0} \vdash x : \sigma}$$

$$\textsc{Let}\ \frac{\Gamma^{\Phi_{\mathtt{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\mathtt{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau}{\Gamma^{\phi.\Phi_{\mathtt{def}} + \Phi_{\mathtt{body}}} \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau}$$

Seems easy...

$$\textsc{Lam}\ \frac{\Gamma^{\Phi}, x{:}\sigma^{\phi} \vdash e : \tau}{\Gamma^{0} \vdash \lambda x.e : [\Gamma^{\Phi}](x : \sigma^{\phi}) \to \tau} \qquad \textsc{Var}\ \frac{\Gamma, x{:}\sigma, \Delta \vdash}{\Gamma^{0}, x{:}\sigma^{1}, \Delta^{0} \vdash x : \sigma}$$

$$\textsc{Let}\ \frac{\Gamma^{\Phi_{\mathtt{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\mathtt{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau}{\Gamma^{\phi.\Phi_{\mathtt{def}} + \Phi_{\mathtt{body}}} \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau}$$

$$\textsc{App}\ \frac{(\Gamma_0, \Gamma_1)^{\Phi_{\mathtt{fun}}} \vdash t : [\Gamma_0^{\Phi_{\mathtt{clos}}}](x{:}\sigma^{\phi}) \to \tau \qquad (\Gamma_0, \Gamma_1)^{\Phi_{\mathtt{arg}}} \vdash u : \sigma}{(\Gamma_0, \Gamma_1)^{\Phi_{\mathtt{fun}} + \Phi_{\mathtt{clos}} + \phi.\Phi_{\mathtt{arg}}} \vdash t\ u : \tau}$$

Easy to get wrong!

$$\dfrac{\vdash 57 : \mathtt{int} \qquad \dfrac{}{x{:}\mathtt{int} \vdash \lambda y.x :}}{\vdash (\mathtt{let}\, x = 57\, \mathtt{in}\, \lambda y.x) :}$$

Easy to get wrong!

$$\dfrac{\vdash 57 : \mathtt{int} \qquad \dfrac{x{:}\mathtt{int}^{1}, y{:}A^{0} \vdash x : \mathtt{int}}{x{:}\mathtt{int} \vdash \lambda y.x :}}{\vdash (\mathtt{let}\, x = 57\, \mathtt{in}\, \lambda y.x) :}$$

Easy to get wrong!

$$\dfrac{\vdash 57 : \mathtt{int} \qquad \dfrac{x{:}\mathtt{int}^1, y{:}A^0 \vdash x : \mathtt{int}}{x{:}\mathtt{int} \vdash \lambda y.x : [x{:}\mathtt{int}^1](y{:}A^0) \to \mathtt{int}}}{\vdash (\mathtt{let}\, x = 57\, \mathtt{in}\, \lambda y.x) :}$$

Easy to get wrong!

$$\dfrac{\vdash 57 : \mathtt{int} \qquad \dfrac{x{:}\mathtt{int}^1, y{:}A^0 \vdash x : \mathtt{int}}{x{:}\mathtt{int} \vdash \lambda y.x : [x{:}\mathtt{int}^1](y{:}A^0) \to \mathtt{int}}}{\vdash (\mathtt{let}\, x = 57\, \mathtt{in}\, \lambda y.x) : [x{:}\mathtt{int}^1](y{:}A^0) \to \mathtt{int}}$$

Easy to get wrong!

$$\dfrac{\vdash 57 : \mathtt{int} \qquad \dfrac{x{:}\mathtt{int}^1, y{:}A^0 \vdash x : \mathtt{int}}{x{:}\mathtt{int} \vdash \lambda y.x : [x{:}\mathtt{int}^1](y{:}A^0) \to \mathtt{int}}}{\vdash (\mathtt{let}\, x = 57\, \mathtt{in}\, \lambda y.x) : [x{:}\mathtt{int}^1](y{:}A^0) \to \mathtt{int}}$$

$[x{:}\mathtt{int}^1](y{:}\ldots) \to \ldots$ does not make sense in a closed context.

The context in the types must be kept synchronized with the environment.
Crucial difference with contextual type theory.

Context managment

New substitution judgment $\Gamma, y{:}\tau, \Delta \vdash \tau \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \tau'$

Context managment

New substitution judgment $\Gamma, y{:}\tau, \Delta \vdash \tau \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \tau'$

LET

$$\frac{\Gamma^{\Phi_{\mathtt{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\mathtt{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau}{\Gamma^{\phi.\Phi_{\mathtt{def}} + \Phi_{\mathtt{body}}} \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau}$$

Context managment

New substitution judgment $\Gamma, y{:}\tau, \Delta \vdash \tau \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \tau'$

LET

$$\frac{\Gamma^{\Phi_{\mathtt{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\mathtt{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau \qquad \Gamma, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\mathtt{def}}}{\rightsquigarrow} \Gamma \vdash \tau'}{\Gamma^{\phi.\Phi_{\mathtt{def}} + \Phi_{\mathtt{body}}} \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau'}$$

Context managment

New substitution judgment $\Gamma, y{:}\tau, \Delta \vdash \tau \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \tau'$

LET

$$\frac{\Gamma^{\Phi_{\text{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\text{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau \qquad \Gamma, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\text{def}}}{\rightsquigarrow} \Gamma \vdash \tau'}{\Gamma^{\phi.\Phi_{\text{def}} + \Phi_{\text{body}}} \vdash \texttt{let}\, x = e_1\, \texttt{in}\, e_2 : \tau'}$$

APP

$$\frac{(\Gamma_0, \Gamma_1)^{\Phi_{\text{fun}}} \vdash t : [\Gamma_0^{\Phi_{\text{clos}}}](x{:}\sigma^{\phi}) \to \tau \qquad (\Gamma_0, \Gamma_1)^{\Phi_{\text{arg}}} \vdash u : \sigma \qquad \Gamma_0, \Gamma_1, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\text{arg}}}{\rightsquigarrow} \Gamma_0, \Gamma_1 \vdash \tau'}{(\Gamma_0, \Gamma_1)^{\Phi_{\text{fun}} + \Phi_{\text{clos}} + \phi.\Phi_{\text{arg}}} \vdash t\ u : \tau'}$$

Context managment

New substitution judgment $\Gamma, y{:}\tau, \Delta \vdash \tau \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \tau'$

LET

$$\frac{\Gamma^{\Phi_{\text{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\text{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau \qquad \Gamma, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\text{def}}}{\rightsquigarrow} \Gamma \vdash \tau'}{\Gamma^{\phi.\Phi_{\text{def}} + \Phi_{\text{body}}} \vdash \texttt{let}\, x = e_1\, \texttt{in}\, e_2 : \tau'}$$

APP

$$\frac{(\Gamma_0, \Gamma_1)^{\Phi_{\text{fun}}} \vdash t : [\Gamma_0^{\Phi_{\text{clos}}}](x{:}\sigma^{\phi}) \to \tau \qquad (\Gamma_0, \Gamma_1)^{\Phi_{\text{arg}}} \vdash u : \sigma \qquad \Gamma_0, \Gamma_1, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\text{arg}}}{\rightsquigarrow} \Gamma_0, \Gamma_1 \vdash \tau'}{(\Gamma_0, \Gamma_1)^{\Phi_{\text{fun}} + \Phi_{\text{clos}} + \phi.\Phi_{\text{arg}}} \vdash t\ u : \tau'}$$

SUBST-CLOSURE

$$\frac{\Gamma, y{:}\rho, \Delta, \Gamma_1 \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta', \Gamma_1' \qquad \Gamma, y{:}\rho, \Delta \vdash \sigma_1 \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta' \vdash \sigma_1 \qquad \Gamma, y{:}\rho, \Delta, x{:}\sigma_1 \vdash \sigma_2 \overset{y \setminus \Psi}{\rightsquigarrow} \Gamma, \Delta', x{:}\sigma_1 \vdash \tau_2}{\begin{array}{c}\Gamma, y{:}\rho, \Delta, \Gamma_1 \vdash [\Gamma^{\Phi_1}, y{:}\rho^{\chi}, \Delta^{\Phi_2}](x{:}\sigma_1^{\phi}) \to \sigma_2 \\ \Gamma, \Delta', \Gamma_1' \vdash [\Gamma^{\Phi_1 + \chi.\Psi}, \Delta'^{\Phi_2}](x{:}\sigma_1^{\phi}) \to \tau_2\end{array} \overset{y \setminus \Psi}{\rightsquigarrow}}$$

Validation?

Static property of **well-scopedness** preserved by type substitutions.

Soundness: If $\Gamma^{\Phi} \vdash t : \sigma$, $V : \Gamma \vdash$ and $V \vdash t \implies v$ then $\Gamma \vdash v : \sigma$.

Validation?

Static property of **well-scopedness** preserved by type substitutions.

Soundness: If $\Gamma^{\Phi} \vdash t : \sigma$, $V : \Gamma \vdash$ and $V \vdash t \implies v$ then $\Gamma \vdash v : \sigma$.

Implementation of the type system.

— the next best thing after a mechanized proof?

Validation?

Static property of **well-scopedness** preserved by type substitutions.

Soundness: If $\Gamma^{\Phi} \vdash t : \sigma$, $V : \Gamma \vdash$ and $V \vdash t \implies v$ then $\Gamma \vdash v : \sigma$.

Implementation of the type system.

— the next best thing after a mechanized proof?

Dynamic correctness property: soundness of intentional information.
In our simple system, a simple **non-interference** property.

A word on soundness

Detour through an exotic (bigstep) semantics that closely mirrors the static structure.

LET

$$\frac{\Gamma^{\Phi_{\mathtt{def}}} \vdash e_1 : \sigma \qquad \Gamma^{\Phi_{\mathtt{body}}}, x{:}\sigma^{\phi} \vdash e_2 : \tau \qquad \Gamma, x{:}\sigma \vdash \tau \overset{x \setminus \Phi_{\mathtt{def}}}{\rightsquigarrow} \Gamma \vdash \tau'}{\Gamma^{\phi \cdot \Phi_{\mathtt{def}} + \Phi_{\mathtt{body}}} \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 : \tau'}$$

RED-LET

$$\frac{V \vdash e_1 \implies v_1 \qquad V, (x \mapsto v_1) \vdash e_2 \implies v_2 \qquad v_2 \overset{x \setminus v_1}{\rightsquigarrow} v_2'}{V \vdash \mathtt{let}\, x = e_1\, \mathtt{in}\, e_2 \implies v_2'}$$

(details next slide)

RED-LAM

$$V \vdash \lambda x.t \implies (\mathsf{dom}\, V, \emptyset, \lambda x.t)$$

SUBST-VALUE-CLOSURE

$$([x_{j_1}, \ldots, x_{j_n}, y], (x_i \mapsto w_i)_{i \in I}, t) \overset{y \setminus v}{\rightsquigarrow} ([x_{j_1}, \ldots, x_{j_n}], (y \mapsto v)(x_i \mapsto w_i)_i, t)$$

RED-LAM

$$V \vdash \lambda x.t \implies (\mathsf{dom}\, V, \emptyset, \lambda x.t)$$

SUBST-VALUE-CLOSURE

$$([x_{j_1}, \ldots, x_{j_n}, y], (x_i \mapsto w_i)_{i \in I}, t) \overset{y \setminus v}{\rightsquigarrow} ([x_{j_1}, \ldots, x_{j_n}], (y \mapsto v)(x_i \mapsto w_i)_i, t)$$

RED-APP

$$\frac{V, V_1 \vdash t \implies (\mathsf{dom}\, V, V_2, \lambda y.t') \qquad V, V_1 \vdash u \implies v_{\mathtt{arg}} \qquad V, V_1, V_2, y \mapsto v_{\mathtt{arg}} \vdash t' \implies w \qquad w \overset{y \setminus v_{\mathtt{arg}}}{\rightsquigarrow} w' \overset{V_2}{\rightsquigarrow} w''}{V, V_1 \vdash t\ u \implies w''}$$

Conclusion

The basic idea (Jan's idea) is simple.

The technical details are trickier than we thought.
It is certainly not the first obvious thing that worked.

Keeping a simple system made the first step possible.
But the shiny things are in the future.

Thanks. Any question?

Steffen Jost, Kevin Hammond, Hans-Wolfgang Loidl, and Martin Hofmann.

Static Determination of Quantitative Resource Usage for Higher-Order Programs.

In *37th Symp. on Principles of Programming Languages (POPL'10)*, pages 223–236, 2010.

Tomas Petricek, Dominic Orchard, and Alan Mycroft.

Coeffects: The Essence of Context Dependence.

2012.

URL: `http://www.cl.cam.ac.uk/~tp322/drafts/coeffects.html`.