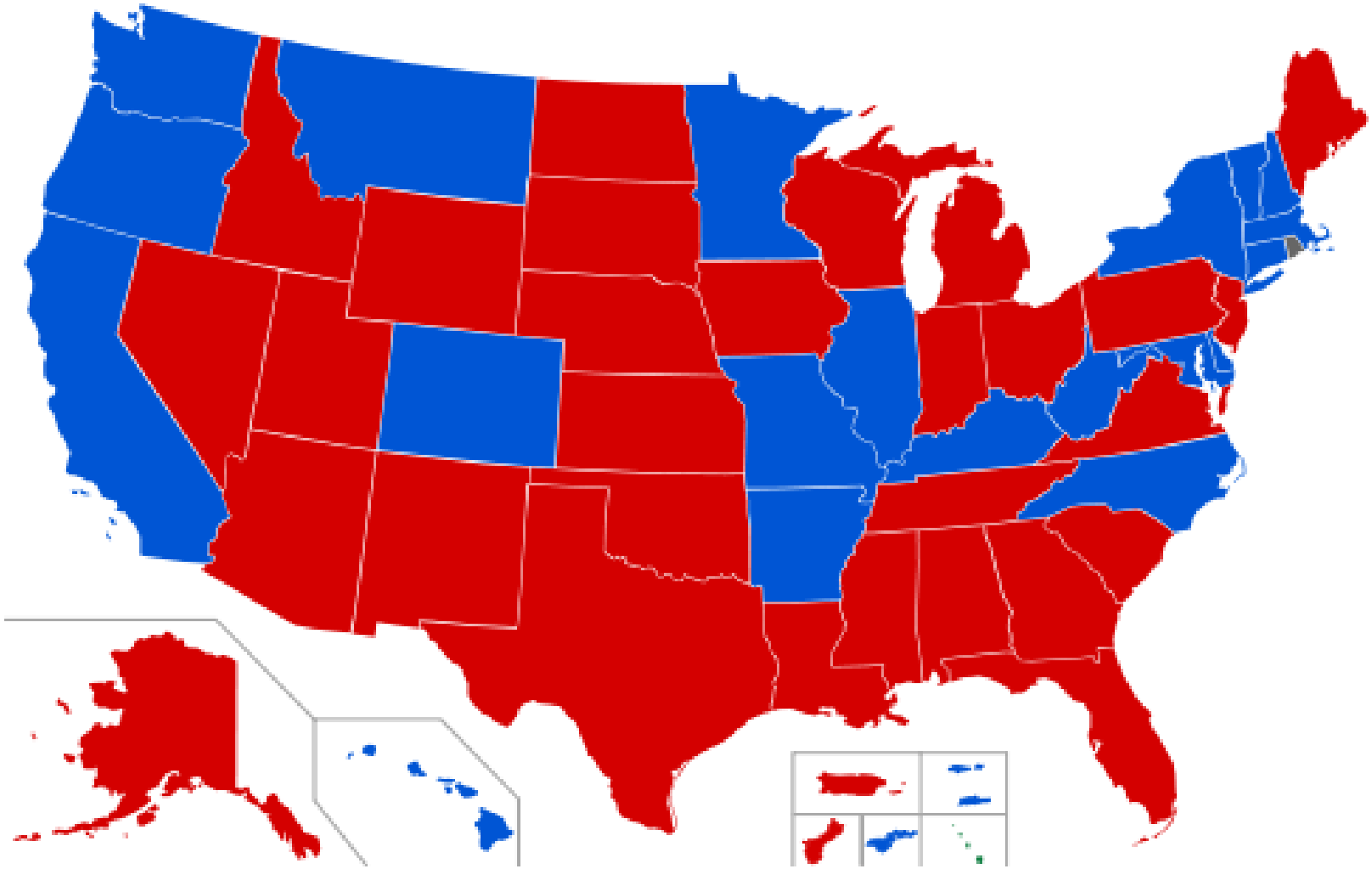
Blame for All

Amal Ahmed, Robert Bruce Findler,
Jeremy Siek, Philip Wadler

Vs.



Part I

The bit you know from before with a twist

A simple untyped program

$$\texttt{let } \mathit{inc}^\star = \lceil \lambda x.\, x + 1 \rceil \texttt{ in}$$
$$\texttt{let } \mathit{app}^\star = \lceil \lambda f.\, \lambda x.\, f\ x \rceil \texttt{ in}$$
$$\lceil \mathit{app}^\star\ \mathit{inc}^\star\ 41 \rceil$$

$$\longrightarrow$$

$$\lceil 42 \rceil : \star$$

A simple typed program

$$
\begin{array}{l}
\texttt{let } \mathit{inc} = \lambda x{:}\texttt{Int}.\, x + 1 \texttt{ in} \\
\texttt{let } \mathit{app} = \Lambda X.\, \Lambda Y.\, \lambda f{:}X \to Y.\, \lambda x{:}X.\, f\ x \texttt{ in} \\
\mathit{app}\ \texttt{Int Int}\ \mathit{inc}\ \texttt{41} \\
\longrightarrow \\
\texttt{42} : \texttt{Int}
\end{array}
$$

Widening: casting from typed to untyped

$$
\begin{array}{l}
\texttt{let } \mathit{inc}^\star = \lceil \lambda x.\, x + 1 \rceil \texttt{ in} \\
\texttt{let } \mathit{app} = \Lambda X.\, \Lambda Y.\, \lambda f{:}X{\to}Y.\, \lambda x{:}X.\, f\ x \texttt{ in} \\
\texttt{let } \mathit{app}^\star = \mathit{app} : \forall X.\, \forall Y.\, (X{\to}Y){\to}X{\to}Y \Rightarrow^p \star \texttt{ in} \\
\lceil \mathit{app}^\star\ \mathit{inc}^\star\ 41 \rceil \\
\longrightarrow \\
\lceil 42 \rceil : \star
\end{array}
$$

Widening: casting from typed to untyped

$$
\begin{array}{l}
\texttt{let } \mathit{inc}^\star = \lceil \lambda x.\, x + 1 \rceil \texttt{ in} \\
\texttt{let } \mathit{app} = \Lambda X.\, \Lambda Y.\, \lambda f{:}X{\to}Y.\, \lambda x{:}X.\, f\ x \texttt{ in} \\
\texttt{let } \mathit{app}^\star = \mathit{app} : \forall X.\, \forall Y.\, (X{\to}Y){\to}X{\to}Y \Rightarrow^{p} \star \texttt{ in} \\
\lceil \mathit{app}^\star\ \texttt{41}\ \mathit{inc}^\star \rceil \\
\longrightarrow \\
\texttt{blame}\, \overline{p}
\end{array}
$$

A cast from more-precise to less-precise type may blame the *context containing* the cast, but never the *term contained* in the cast.

Narrowing: casting from untyped to typed

$$\texttt{let } \mathit{inc} = \lambda x{:}\texttt{Int}.\, x + 1 \texttt{ in}$$

$$\texttt{let } \mathit{app}^{\star} = \lceil \lambda f.\, \lambda x.\, f\ x \rceil \texttt{ in}$$

$$\texttt{let } \mathit{app} = \mathit{app}^{\star} : \star \Rightarrow^{p} \forall X.\, \forall Y.\, (X \to Y) \to X \to Y \texttt{ in}$$

$$\mathit{app}\ \texttt{Int}\ \texttt{Int}\ \mathit{inc}\ \texttt{41}$$

$$\longrightarrow$$

$$\texttt{42} : \texttt{Int}$$

Narrowing: casting from untyped to typed

$$
\begin{array}{l}
\texttt{let } \mathit{inc} = \lambda x{:}\texttt{Int}.\, x + 1 \texttt{ in} \\
\texttt{let } \mathit{app}^{\star} = \lceil \lambda f.\, \lambda x.\, x \rceil \texttt{ in} \\
\texttt{let } \mathit{app} = \mathit{app}^{\star} : \star \Rightarrow^{p} \forall X.\, \forall Y.\, (X \to Y) \to X \to Y \texttt{ in} \\
\mathit{app}\ \texttt{Int}\ \texttt{Int}\ \mathit{inc}\ 41 \\
\longrightarrow \\
\texttt{blame}\, p
\end{array}
$$

A cast from less-precise to more-precise type may blame the *term contained* in the cast, but never the *context containing* the cast.

Part II

The fairly straightforward bit

Explicit binding

$$\frac{\Gamma, X{:=}A \vdash t : B \qquad \Gamma \vdash A \qquad X \notin \mathrm{ftv}(B)}{\Gamma \vdash \nu X{:=}A.\, t : B}$$

$$\frac{\Gamma \vdash t : B \qquad (X{:=}A) \in \Gamma}{\Gamma \vdash t : B[X{:=}A]} \qquad \frac{\Gamma \vdash t : B[X{:=}A] \qquad (X{:=}A) \in \Gamma}{\Gamma \vdash t : B}$$

$$(\Lambda X.\, v)\ A \quad \longrightarrow \quad \nu X{:=}A.\, v$$

Global store vs. Local bindings

George Neis, Derek Dreyer, and Andreas Rossberg. Non-parametric parametricity. ICFP 2009, Edinburgh.

Polymorphic blame calculus

$$\frac{\Gamma \vdash s : A \qquad A \prec B}{\Gamma \vdash (s : A \Rightarrow^{p} B) : B}$$

This is a *contract*.

The *contained term* promises to provide a B.

The *containing context* promises to require an A.

Compatibility and reductions

$$A \prec \star \qquad \star \prec B \qquad \frac{A' \prec A \quad B \prec B'}{A \to B \prec A' \to B'}$$

$$\frac{A \prec B}{A \prec \forall X.\, B}\, X \notin \mathrm{ftv}(A) \qquad \frac{A[X{:=}\star] \prec B}{\forall X.\, A \prec B}$$

$$\begin{array}{rcll}
v : A \to B \Rightarrow^{p} A' \to B' & \longrightarrow & \lambda x{:}A'.\, v\ (x : A' \Rightarrow^{\overline{p}} A) : B \Rightarrow^{p} B' & \\
v : A \Rightarrow^{p} (\forall X.\, B) & \longrightarrow & \Lambda X.\, v : A \Rightarrow^{p} B & \text{if } X \notin \mathrm{ftv} A \\
v : (\forall X.\, A) \Rightarrow^{p} B & \longrightarrow & v \star : A[X{:=}\star] \Rightarrow^{p} B & \\
v : X \Rightarrow^{p} \star \Rightarrow^{q} X & \longrightarrow & v & \\
v : X \Rightarrow^{p} \star \Rightarrow^{q} Y & \longrightarrow & \mathtt{blame}\, q & \text{if } X \neq Y
\end{array}$$

Part III

The bit with the cute name

The Jack-of-All-Trades Principle

Proposition 1 (Jack-of-All-Trades Principle)**.** *If* $\Delta \vdash v : \forall X.\, A$ *and* $A[X{:=}C] \prec B$ *(and hence* $A[X{:=}\star] \prec B$*) then*

$$(v\ C : A[X{:=}C] \Rightarrow^{p} B) \quad \sqsubseteq \quad (v\ \star : A[X{:=}\star] \Rightarrow^{p} B).$$

$$\frac{s \sqsubseteq^p t \qquad A \sqsubseteq A'}{s : A \Rightarrow^p B \sqsubseteq^p t : A' \Rightarrow^p B}$$

$$\frac{s \sqsubseteq^p t \qquad A \sqsubseteq A'}{s : B \Rightarrow^{\bar{p}} A \sqsubseteq^p t : B \Rightarrow^{\bar{p}} A'}$$

$$\mathtt{blame}\, q \sqsubseteq^p t$$

$$\frac{s \sqsubseteq^p t}{\Lambda Y.s \sqsubseteq^p t} \qquad \frac{s \sqsubseteq^p t}{s : A \Rightarrow^P B \sqsubseteq^p t}$$

$$\frac{v \sqsubseteq^p w}{\lambda x' {:} A'.\, v\ (x' : A' \Rightarrow^{\overline{P}} A) : B \Rightarrow^P B' \sqsubseteq^p w}$$

$$\frac{v \sqsubseteq^p w}{\Lambda X.\, (v\ X : B \Rightarrow^P B') \sqsubseteq^p w}$$

$$\frac{v \sqsubseteq^p w}{v \sqsubseteq^p w : G \Rightarrow \star} \qquad \frac{v \sqsubseteq^p w}{v \sqsubseteq^p w : Y \Rightarrow^q \star}$$

$$c \sqsubseteq^p c \qquad x \sqsubseteq^p x$$

$$\frac{s \sqsubseteq^p t}{\lambda x.s \sqsubseteq^p \lambda x.t} \qquad \frac{s_1 \sqsubseteq^p t_1 \quad s_2 \sqsubseteq^p t_2}{s_1\ s_2 \sqsubseteq^p t_1\ t_2}$$

$$\frac{s \sqsubseteq^p t}{\Lambda Y.s \sqsubseteq^p \Lambda Y.t} \qquad \frac{s \sqsubseteq^p t \quad A \sqsubseteq A'}{s\ A \sqsubseteq^p t\ A'}$$

$$\frac{s \sqsubseteq^p t}{s \,\mathtt{is}\, G \sqsubseteq^p t \,\mathtt{is}\, G} \qquad \mathtt{blame}\, q \sqsubseteq^p \mathtt{blame}\, q$$

$$\frac{s \sqsubseteq^p t}{s : A \Rightarrow^q B \sqsubseteq^p t : A \Rightarrow^q B} \qquad \frac{v \sqsubseteq^p w}{v : G \Rightarrow \star \sqsubseteq^p w : G \Rightarrow \star}$$

$$\frac{s \sqsubseteq^p t \quad A \sqsubseteq A' \quad C \sqsubseteq C'}{s : A \Rightarrow^{X:=C} B \sqsubseteq^p t : A' \Rightarrow^{X:=C'} B'}$$

$$\frac{s \sqsubseteq^p t \quad B \sqsubseteq B' \quad C \sqsubseteq C'}{s : A \Rightarrow^{\overline{X:=C}} B \sqsubseteq^p t : A' \Rightarrow^{\overline{X:=C'}} B'}$$

Part IV

The dauntingly complicated but rewarding bit

$$\prec \qquad <: \qquad <:^{+} \qquad <:^{-} \qquad <:_{n}$$

Subtyping

$$\frac{A' <: A \qquad B <: B'}{A \to B <: A' \to B'}$$

$$\frac{A <: B}{A <: \forall X.\, B}\, X \notin \mathrm{ftv}(A) \qquad \frac{A[X{:=}C] <: B}{\forall X.\, A <: B}$$

Proposition 2 (Subtyping)**.** *Let t be a program with a subterm $s : A \Rightarrow^{p} B$ where the cast is labelled by the only occurrence of p in t, and $\overline{p}$ does not appear in t.*

- *If $A <: B$, then $t \not\longmapsto^{*} \texttt{blame}\, p$ and $t \not\longmapsto^{*} \texttt{blame}\, \overline{p}$.*

Proof depends on Jack-of-All-Trades.

Positive and negative subtyping

$$A <:^+ \star \qquad \frac{A' <:^- A \quad B <:^+ B'}{A \to B <:^+ A' \to B'}$$

$$\frac{A <:^+ B}{A <:^+ \forall X.\, B}\, X \notin \mathrm{ftv}(A) \qquad \frac{A[X:=\star] <:^+ B}{\forall X.\, A <:^+ B}$$

$$\star <:^- B \qquad \frac{A' <:^+ A \quad B <:^- B'}{A \to B <:^- A' \to B'}$$

$$\frac{A <:^- B}{A <:^- \forall X.\, B}\, X \notin \mathrm{ftv}(A) \qquad \frac{A[X:=\star] <:^- B}{\forall X.\, A <:^- B}$$

Proposition 3 (Blame)**.** *Let t be a program with a subterm $s : A \Rightarrow^p B$ where the cast is labelled by the only occurrence of p in t, and $\overline{p}$ does not appear in t.*

- *If $A <:^+ B$, then $t \not\longmapsto^* \mathtt{blame}\, p$.*
- *If $A <:^- B$, then $t \not\longmapsto^* \mathtt{blame}\, \overline{p}$.*

Naive subtyping

$$A <:_n \star \qquad \frac{A <:_n A' \qquad B <:_n B'}{A \to B <:_n A' \to B'}$$

$$\frac{A <:_n B}{A <:_n \forall X.\, B}\, X \notin \mathrm{ftv}(A) \qquad \frac{A[X:=\star] <:_n B}{\forall X.\, A <:_n B}$$

Proposition 4 (Factoring). *$A <:_n B$ iff $A <:^+ B$ and $B <:^- A$.*

Proposition 5 (Blame). *Let t be a program with a subterm $s : A \Rightarrow^p B$ where the cast is labelled by the only occurrence of p in t, and $\overline{p}$ does not appear in t.*

- *If $A <:_n B$, then $t \mapsto\!\!\!\!\!\!\rightarrow^* \texttt{blame}\, p$.*
- *If $B <:_n A$, then $t \mapsto\!\!\!\!\!\!\rightarrow^* \texttt{blame}\, \overline{p}$.*

Part V

Abstract types are existential types

Pair and existential types

$$
\begin{aligned}
A \times B &= \forall Z.\,(A \to B \to Z) \to Z \\
(v, w)^{A \times B} &= \Lambda Z.\,\lambda k{:}A \to B \to Z.\,k\ v\ w \\
\texttt{fst}^{A \times B}\ u &= u\ A\ (\lambda x{:}A.\,\lambda y{:}B.\,x) \\
\texttt{snd}^{A \times B}\ u &= u\ B\ (\lambda x{:}A.\,\lambda y{:}B.\,y)
\end{aligned}
$$

$$
\begin{aligned}
\exists X.\,B &= \forall Z.\,(\forall X.\,B \to Z) \to Z \\
\texttt{pack}\ (A, v)\ \texttt{as}\ \exists X.\,B &= \Lambda Z.\,\lambda k{:}(\forall X.\,B \to Z).\,k\ A\ v \\
\texttt{unpack}\ (X, y) = v\ \texttt{in}\ t : C &= v\ C\ (\Lambda X.\,\lambda y{:}B.\,t)
\end{aligned}
$$

Type abstraction

$$
\begin{array}{rcl}
\mathit{Sem} & = & \exists X.\, X \times (X \to X) \times (X \to \texttt{Bool}) \\
\mathit{impl}_1 & = & \texttt{pack}\,(\texttt{Bool},\, (\texttt{true},\, \lambda x{:}\texttt{Bool}.\, \neg x,\, \lambda x{:}\texttt{Bool}.\, x))\ \texttt{as}\ \mathit{Sem} \\
\mathit{impl}_2 & = & \texttt{pack}\,(\texttt{Int},\, (1,\, \lambda x{:}\texttt{Int}.\, 1 - x,\, \lambda x{:}\texttt{Int}.\, x \neq 0))\ \texttt{as}\ \mathit{Sem}
\end{array}
$$

$$\texttt{unpack}\ (X, (\mathit{sem}, \mathit{toggle}, \mathit{poll})) = \mathit{impl}_i\ \texttt{in}\ (\mathit{poll}\ \mathit{sem},\ \mathit{poll}\ (\mathit{toggle}\ \mathit{sem}))$$

$\longrightarrow$

$$(\texttt{true},\, \texttt{false})$$

$$\texttt{unpack}\ (X, (\mathit{sem}, \mathit{toggle}, \mathit{poll})) = \mathit{impl}_i\ \texttt{in}\ (\mathit{poll}\ 666,\ \mathit{poll}\ (\mathit{toggle}\ 666))$$

$\longrightarrow$

(static type error)

Pair and existential types, untyped

$$
\begin{array}{rcl}
(v, w) & = & \lceil \lambda k.\, k\ v\ w \rceil \\
\texttt{fst}\ t & = & \lceil t\ (\lambda x.\, \lambda y.\, x) \rceil \\
\texttt{snd}\ t & = & \lceil t\ (\lambda x.\, \lambda y.\, y) \rceil
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{pack}^p\ v\ \texttt{as}\ \exists X.\, B & = & \lceil \lambda k.\, k\ v \rceil : \star \Rightarrow^p \forall Z.\, (\forall X.\, B \to Z) \to Z \Rightarrow^p \star \\
\texttt{unpack}\ y = v\ \texttt{in}\ t & = & \lceil v\ (\lambda y.\, t) \rceil
\end{array}
$$

Type abstraction, untyped

$$\begin{array}{rcl} \mathit{Sem} & = & \exists X.\, X \times (X \to X) \times (X \to \texttt{Bool}) \\ \mathit{impl}_1^\star & = & \texttt{pack}^p\ (\texttt{Bool},\ (\texttt{true},\ \lambda x{:}\texttt{Bool}.\, \neg x,\ \lambda x{:}\texttt{Bool}.\, x))\ \texttt{as}\ \mathit{Sem} \\ \mathit{impl}_2^\star & = & \texttt{pack}^p\ (\texttt{Int},\ (1,\ \lambda x{:}\texttt{Int}.\, 1 - x,\ \lambda x{:}\texttt{Int}.\, x \neq 0))\ \texttt{as}\ \mathit{Sem} \end{array}$$

$$\texttt{unpack}\ (X, (\mathit{sem}, \mathit{toggle}, \mathit{poll})) = \mathit{impl}_i^\star\ \texttt{in}\ (\mathit{poll}\ \mathit{sem},\ \mathit{poll}\ (\mathit{toggle}\ \mathit{sem}))$$

$\longrightarrow$

$$(\texttt{true},\ \texttt{false})$$

$$\texttt{unpack}\ (X, (\mathit{sem}, \mathit{toggle}, \mathit{poll})) = \mathit{impl}_i^\star\ \texttt{in}\ (\mathit{poll}\ 666,\ \mathit{poll}\ (\mathit{toggle}\ 666))$$

$\longrightarrow$

$$\texttt{blame}\ \overline{p}$$

Part VI

The surprise ending

Static

Polymorphic lambda calculus with static casts

$$\frac{\Gamma \vdash t : B \qquad (X{:=}A) \in \Gamma}{\Gamma \vdash (t : B \Rightarrow^{X} B[X{:=}A]) : B[X{:=}A]} \qquad \frac{\Gamma \vdash t : B[X{:=}A] \qquad (X{:=}A) \in \Gamma}{\Gamma \vdash (t : B[X{:=}A] \Rightarrow^{\overline{X}} B) : B}$$

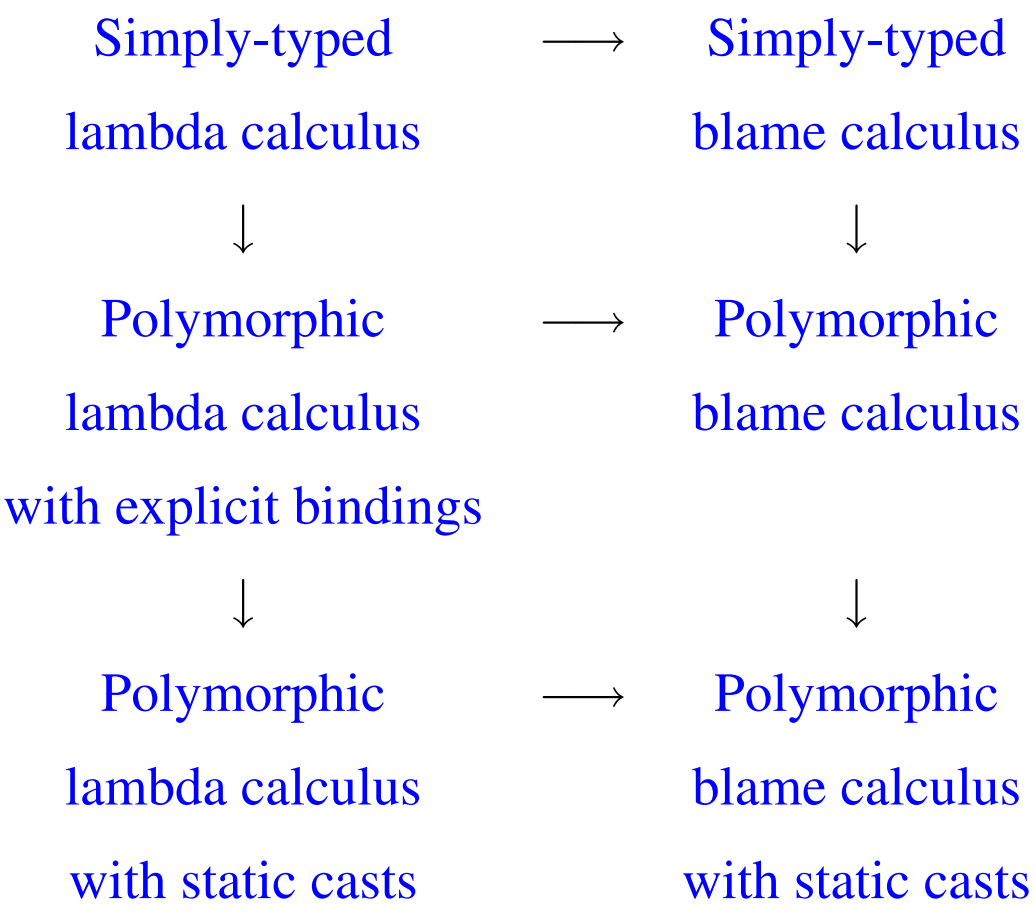
$$(\Lambda X.\, v)\ A \quad \longrightarrow \quad \nu X{:=}A.\, (v : B \Rightarrow^{X} B[X{:=}A]) \quad \text{when } \Lambda X.\, v : \forall X.\, B$$

Proposition 6 (Canonical forms)**.** *If* $\Delta \vdash v : C$ *then either*

- $v = c$ *and* $C = \iota$ *for some* c *and* ι*, or*
- $v = \lambda x{:}A.\, t$ *and* $C = A{\to}B$ *for some* x*,* t*,* A*, and* B*, or*
- $v = \Lambda X.\, w$ *and* $C = \forall X.\, A$ *for some* w*,* X*, and* A*.*
- $v = w : A \Rightarrow^{\overline{X}} X$ *and* $C = X$ *for some* w*,* X*, and* A*.*

Dan Grossman, Greg Morrisett, and Steve Zdancewic. Principals in programming languages: a syntactic proof technique. ICFP 1999, Paris.

Roadmap


Simply-typed
lambda calculus
Simply-typed
blame calculus
Polymorphic
lambda calculus
with explicit bindings
Polymorphic
blame calculus
Polymorphic
lambda calculus
with static casts
Polymorphic
blame calculus
with static casts

Notation

It took us four years to find the right notation!

$$\langle A \rhd B \rangle^p \; s$$

$$\langle B \lhd A \rangle^p \; s$$

$$s : A \Rightarrow^p B$$

We want composition to be easy to read:

$$\langle B \rhd C \rangle^q \; \langle A \rhd B \rangle^p \; s$$

$$\langle C \lhd B \rangle^q \; \langle B \lhd A \rangle^p \; s$$

$$s : A \Rightarrow^p B : B \Rightarrow^q C$$

And there is a convenient abbreviation:

$$s : A \Rightarrow^p B \Rightarrow^q C$$

Appendix

All the reduction rules

Simply-typed lambda calculus

$$(\lambda x{:}A.\, t)\; v \quad \longrightarrow \quad t[x{:=}v]$$

$$op(\vec{v}\,) \quad \longrightarrow \quad \delta(op, \vec{v}\,)$$

$$\frac{s \longrightarrow t}{E[s] \longmapsto E[t]}$$

Simply-typed blame calculus

$$
\begin{array}{rcll}
v : A{\to}B \Rightarrow^p A'{\to}B' & \longrightarrow & \lambda x{:}A'.\,(v\ (x : A' \Rightarrow^{\overline{p}} A) : B \Rightarrow^p B') & \\
v : G \Rightarrow^p G & \longrightarrow & v & \text{if } G \neq \star{\to}\star \\
v : A \Rightarrow^p \star & \longrightarrow & v : A \Rightarrow^p G \Rightarrow \star & \text{if } A \prec G \text{ and } A \neq \star \\
v : G \Rightarrow \star \Rightarrow^p A & \longrightarrow & v : G \Rightarrow^p A & \text{if } G \prec A \\
v : G \Rightarrow \star \Rightarrow^p A & \longrightarrow & \texttt{blame}\, p & \text{if } G \not\prec A
\end{array}
$$

Polymorphic lambda calculus with type binding

$$
\begin{aligned}
(\Lambda X.\, v)\ A &\longrightarrow \nu X{:=}A.\, v \\
\nu X{:=}A.\, c &\longrightarrow c \\
\nu X{:=}A.\, (\lambda y{:}B.\, t) &\longrightarrow \lambda y{:}B[X{:=}A].\, (\nu X{:=}A.\, t) \\
\nu X{:=}A.\, (\Lambda Y.\, v) &\longrightarrow \Lambda Y.\, (\nu X{:=}A.\, v)
\end{aligned}
$$

Polymorphic blame calculus

$$\nu X{:=}A.\,(v : G \Rightarrow \star) \longrightarrow (\nu X{:=}A.\,v) : G \Rightarrow \star \qquad \text{if } G \neq X$$

$$\nu X{:=}A.\,(v : X \Rightarrow \star) \longrightarrow \texttt{blame}\, p_\nu$$

$$v : A \Rightarrow^p (\forall X.\,B) \longrightarrow \Lambda X.\,(v : A \Rightarrow^p B) \qquad \text{if } X \notin \mathrm{ftv} A$$

$$v : (\forall X.\,A) \Rightarrow^p B \longrightarrow (v\ \star) : A[X{:=}\star] \Rightarrow^p B$$

$$\text{if } B \neq \star \text{ and } B \neq \forall X'.\,B' \text{ for any } X', B'$$

Polymorphic lambda calculus with static casts

$$(\Lambda X.\, v)\ A \longrightarrow \nu X{:=}A.\, (v : B \Rightarrow^{X} B[X{:=}A])$$

when $\Lambda X.\, v : \forall X.\, B$

$$\nu X{:=}A.\, (v : B \Rightarrow^{\overline{Y}} Y) \longrightarrow (\nu X{:=}A.\, v) : B \Rightarrow^{\overline{Y}} Y$$

$$v : \iota \Rightarrow^{P} \iota \longrightarrow v$$

$$(\lambda x : A.\, t) : A{\to}B \Rightarrow^{P} A'{\to}B' \longrightarrow \lambda x{:}A'.\, (t[x{:=}(x : A' \Rightarrow^{\overline{P}} A)] : B \Rightarrow^{P} B')$$

$$(\Lambda X.\, v) : \forall X.\, B \Rightarrow^{P} \forall X.\, B' \longrightarrow \Lambda X.\, (v : B \Rightarrow^{P} B') \text{ if } X \neq P \text{ and } X \neq \overline{P}$$

$$v : X \Rightarrow^{P} X \longrightarrow v \qquad \text{if } X \neq P \text{ and } X \neq \overline{P}$$

$$v : A \Rightarrow^{\overline{X}} X \Rightarrow^{X} A \longrightarrow v$$

Polymorphic blame calculus with static casts

$$v : \star \Rightarrow^{P} \star \quad \longrightarrow \quad v$$