

Exploring Polymorphic Gradual Types with Phil and Peter

Jeremy G. Siek

Indiana University, Bloomington



System F programs in color

$$(\Lambda X. \lambda f: (\forall Z. Z \rightarrow Z). \lambda x: X. f \ X \ x) \ \mathbb{N} \ (\Lambda Z. \lambda z: Z. z) \ 42$$

- ▶ Uncoloured terms have no type variable in scope;
- ▶ terms have X in scope;
- ▶ terms have Z in scope.

Reduction wipes out the colors, so sad!

$$\begin{aligned} & (\Lambda X. \lambda f: (\forall Z. Z \rightarrow Z). \lambda x: X. f \ X \ x) \ \mathbb{N} \ (\Lambda Z. \lambda z: Z. z) \ 42 \\ \longrightarrow & (\lambda f: (\forall Z. Z \rightarrow Z). f \ \mathbb{N} \ x) \ (\Lambda Z. \lambda z: Z. z) \ 42 \\ \longrightarrow & (\Lambda Z. \lambda z: Z. z) \ \mathbb{N} \ 42 \\ \longrightarrow & (\lambda z: \mathbb{N}. z) \ 42 \\ \longrightarrow & 42 \end{aligned}$$

What if terms could stay the same color?

What if the type variables in scope for each subterm were invariant under reduction?

$$\begin{aligned} & (\lambda X. \lambda f: (\forall Z. Z \rightarrow Z). f X) \mathbb{N} (\lambda Z. \lambda z: Z. z) \\ \longrightarrow & (\lambda f: (\forall Z. Z \rightarrow Z). f X) (\lambda Z. \lambda z: Z. z) & X \mapsto \mathbb{N} \\ \longrightarrow & (\lambda Z. \lambda z: Z. z) X \\ \longrightarrow & \lambda z: Z. z & Z \mapsto X \mapsto \mathbb{N} \end{aligned}$$

- ▶ But what happens at the boundaries?
- ▶ What happens to the types?
- ▶ What happens to the scope?

Syntactic Type Abstraction (Grossman et al.)

- ▶ Define a multi-agent calculus that supports the *embedding* of an agent within another agent.
- ▶ The agents have differing knowledge about type equalities that link abstract types to their representations.
- ▶ System F is modeled by the body of each Λ spawning a new agent. Type application gives the knowledge (e.g. $X = A$) to the surrounding agent, not the spawned one.
- ▶ The typing of embedding says the inner and outer types must match using the knowledge of the agents involved.
- ▶ Their type system does not include well-formedness conditions to check type variable scoping.
- ▶ Type variables are added but never removed from the typing context.

$\nu[F]$, a colorful System F

$$\begin{aligned}
 & (\Lambda X. \lambda f: (\forall Z. Z \rightarrow Z). f X) \mathbb{N} (\Lambda Z. \lambda z: Z. z) \\
 \longrightarrow & \frac{\emptyset}{\nu X := \mathbb{N}} \left[\lambda f: (\forall Z. Z \rightarrow Z). f X \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z) \rightarrow (-X \rightarrow +X)} (\Lambda Z. \lambda z: Z. z) \\
 \longrightarrow & \frac{\emptyset}{\nu X := \mathbb{N}} \left[(\lambda f: (\forall Z. Z \rightarrow Z). f X) \frac{-X}{\nu \emptyset} \left[\Lambda Z. \lambda z: Z. z \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} \right]_{-X \rightarrow +X} \\
 \longrightarrow & \frac{\emptyset}{\nu X := \mathbb{N}} \left[\frac{-X}{\nu \emptyset} \left[\Lambda Z. \lambda z: Z. z \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} X \right]_{-X \rightarrow +X} \\
 \longrightarrow & \frac{\emptyset}{\nu X := \mathbb{N}} \left[\frac{-X}{\nu Z := X} \left[\lambda z: Z. z \right]_{-Z \rightarrow +Z} \right]_{-X \rightarrow +X}
 \end{aligned}$$

An example that breaks Fitch-style locking

The term $M \uparrow[X:=A]$ reveals X outward, $M \downarrow[X:=A]$ conceals it; the interior of a conceal was typed at $\Gamma \downarrow X$, the context with X *and everything bound after it* dropped.

$$\begin{aligned} & ((\Lambda X. \lambda f:(\forall Z. Z \rightarrow Z). \Lambda Y. f Y) \mathbb{N} (\Lambda Z. \lambda z:Z. z)) \mathbb{B} \text{true} \\ \longrightarrow & ((\lambda f:(\forall Z. Z \rightarrow Z). \Lambda Y. f Y) \uparrow[X:=\mathbb{N}] (\Lambda Z. \lambda z:Z. z)) \mathbb{B} \text{true} \\ \longrightarrow & ((\lambda f. \Lambda Y. f Y) (\Lambda Z. \lambda z:Z. z) \downarrow[X:=\mathbb{N}]) \uparrow[X:=\mathbb{N}] \mathbb{B} \text{true} \\ \longrightarrow & ((\Lambda Y. (\Lambda Z. \lambda z:Z. z) \downarrow[X:=\mathbb{N}] Y) \uparrow[X:=\mathbb{N}]) \mathbb{B} \text{true} \\ \longrightarrow & ((\Lambda Y. ((\Lambda Z. \lambda z:Z. z) Y) \downarrow[X:=\mathbb{N}]) \uparrow[X:=\mathbb{N}]) \mathbb{B} \text{true} \end{aligned}$$

The last term is ill-typed because $(\Lambda Z. \lambda z:Z. z) Y$ refers to Y which is blocked by the $\downarrow X:=\mathbb{N}$.

The same program in $\nu[F]$

$$\begin{aligned}
 & \left(\left(\Lambda X. \lambda f. (\forall Z. Z \rightarrow Z). \Lambda Y. f Y \right) \mathbb{N} \left(\Lambda Z. \lambda z. Z. z \right) \right) \mathbb{B} \text{true} \\
 \rightarrow & \left(\left(\frac{\emptyset}{\nu X := \mathbb{N}} \left[\lambda f. (\forall Z. Z \rightarrow Z). \Lambda Y. f Y \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z) \rightarrow \forall Y. (\text{id}_Y \rightarrow \text{id}_Y)} \left(\Lambda Z. \lambda z. Z. z \right) \right) \right) \mathbb{B} \text{true} \\
 \rightarrow & \left(\left(\frac{\emptyset}{\nu X := \mathbb{N}} \left[\left(\lambda f. \Lambda Y. f Y \right) \frac{-X}{\nu \emptyset} \left[\Lambda Z. \lambda z. Z. z \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} \right]_{\forall Y. (\text{id}_Y \rightarrow \text{id}_Y)} \right) \right) \mathbb{B} \text{true} \\
 \rightarrow & \left(\left(\frac{\emptyset}{\nu X := \mathbb{N}} \left[\Lambda Y. \frac{-Y}{\nu \emptyset} \left[\frac{-X}{\nu \emptyset} \left[\Lambda Z. \lambda z. Z. z \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} Y \right]_{\forall Y. (\text{id}_Y \rightarrow \text{id}_Y)} \right) \right) \mathbb{B} \text{true} \\
 \rightarrow & \left(\left(\frac{\emptyset}{\nu X := \mathbb{N}} \left[\Lambda Y. \frac{-Y}{\nu Z' := Y} \left[\frac{-X, -Z'}{\nu \emptyset} \left[\Lambda Z. \lambda z. Z. z \right]_{\forall Z. (\text{id}_Z \rightarrow \text{id}_Z)} Z' \right]_{-Z' \rightarrow +Z'} \right]_{\forall Y. (\text{id}_Y \rightarrow \text{id}_Y)} \right) \right) \mathbb{B} \text{true} \\
 \rightarrow & \left(\left(\frac{\emptyset}{\nu X := \mathbb{N}} \left[\Lambda Y. \frac{-Y}{\nu Z' := Y} \left[\frac{-X, -Z'}{\nu Z := Z'} \left[\lambda z. Z. z \right]_{-Z \rightarrow +Z} \right]_{-Z' \rightarrow +Z'} \right]_{\forall Y. (\text{id}_Y \rightarrow \text{id}_Y)} \right) \right) \mathbb{B} \text{true} \\
 \rightarrow &^* \text{true}
 \end{aligned}$$

Terms: Binding, Anti-Binding, and Conversion

$$A, B, C, D ::= \iota \mid A \rightarrow B \mid X \mid \forall X. A$$
$$L, M, N ::= k \mid x \mid \lambda x:A. N \mid LM \mid \Lambda X. N \mid LA \mid \delta_{\nu\Theta}[M]_c$$
$$\delta ::= \emptyset \mid -X, \delta \mid +X, \delta$$
$$\Theta ::= \emptyset \mid X := A, \Theta$$
$$c, d ::= \text{id}_A \mid -X \mid +X \mid c \rightarrow d \mid \forall X. c$$

Conversions are inspired by Henglein's Coercion Calculus (1994).
They represent the static casts of Ahmed et al. (2011)

Contexts and type-variable lookup

$$\beta ::= X \mid X := A$$

$$\hat{\beta} ::= \beta \mid \text{locked}(\beta)$$

$$\Delta ::= \emptyset \mid \Delta, \hat{\beta}$$

$$\text{name}(X) = X$$

$$\text{name}(X := A) = X$$

$$\text{name}(\text{locked}(\beta)) = \text{name}(\beta)$$

$$\boxed{\Delta \ni X \mapsto \hat{\beta}}$$

$$\frac{\text{name}(\hat{\beta}) = X}{\Delta, \hat{\beta} \ni X \mapsto \hat{\beta}} \quad \frac{\text{name}(\hat{\beta}') \neq X \quad \Delta \ni X \mapsto \hat{\beta}}{\Delta, \hat{\beta}' \ni X \mapsto \hat{\beta}}$$

$$\boxed{\Delta \ni X}$$

$$\frac{\Delta \ni X \mapsto X}{\Delta \ni X} \quad \frac{\Delta \ni X \mapsto X := A}{\Delta \ni X}$$

$$\boxed{\Delta \ni_{\text{lock}} X}$$

$$\frac{\Delta \ni X \mapsto \text{locked}(\beta)}{\Delta \ni_{\text{lock}} X}$$

Operations on type contexts

$$\text{lock}(X, \Delta) = \Delta'$$

$$\text{lock}(X, \emptyset) = \emptyset$$

$$\text{lock}(X, (\Delta, \beta)) = \begin{cases} \Delta, \text{locked}(\beta) & \text{if } \text{name}(\beta) = X \\ \text{lock}(X, \Delta), \beta & \text{otherwise} \end{cases}$$

$$\text{lock}(X, (\Delta, \text{locked}(\beta))) = \text{lock}(X, \Delta), \text{locked}(\beta)$$

$$\text{unlock}(X, \Delta) = \Delta'$$

$$\text{unlock}(X, \emptyset) = \emptyset$$

$$\text{unlock}(X, (\Delta, \text{locked}(\beta))) = \Delta, \beta \quad \text{if } \text{name}(\beta) = X$$

$$\text{unlock}(X, (\Delta, \hat{\beta})) = \text{unlock}(X, \Delta), \hat{\beta} \quad \text{if } X \neq \text{name}(\hat{\beta})$$

$$\delta(\Delta) = \Delta'$$

$$\delta^+(\Delta) = \Delta'$$

$$\emptyset(\Delta) = \Delta$$

$$(-X, \delta)(\Delta) = \text{lock}(X, \delta(\Delta))$$

$$(+X, \delta)(\Delta) = \text{unlock}(X, \delta(\Delta))$$

$$\emptyset^+(\Delta) = \Delta$$

$$(-X, \delta)^+(\Delta) = \delta^+(\Delta)$$

$$(+X, \delta)^+(\Delta) = \text{unlock}(X, \delta^+(\Delta))$$

Well-formed types, bindings, and anti-bindings

$$\boxed{\Delta \vdash A}$$

$$\frac{\Delta \ni X}{\Delta \vdash X} \quad \frac{}{\Delta \vdash \iota} \quad \frac{\Delta \vdash A \quad \Delta \vdash B}{\Delta \vdash A \rightarrow B} \quad \frac{\Delta, X \vdash A}{\Delta \vdash \forall X. A}$$

$$\boxed{\Delta \vdash \Theta}$$

$$\frac{}{\Delta \vdash \emptyset} \quad \frac{\Delta \vdash A \quad \Delta \vdash \Theta}{\Delta \vdash X := A, \Theta}$$

$$\boxed{\Delta \vdash \delta}$$

$$\frac{}{\Delta \vdash \emptyset} \quad \frac{\delta(\Delta) \ni X \quad \Delta \vdash \delta}{\Delta \vdash -X, \delta} \quad \frac{\delta(\Delta) \ni_{\text{lock}} X \quad \Delta \vdash \delta}{\Delta \vdash +X, \delta}$$

Bindings are dual to anti-bindings

$$\Theta^{\text{op}} = \delta$$

$$\emptyset^{\text{op}} = \emptyset$$

$$(X := A, \Theta)^{\text{op}} = -X, \Theta^{\text{op}}$$

$$\delta^{\text{op}} = \delta$$

$$\emptyset^{\text{op}} = \emptyset$$

$$(+X, \delta)^{\text{op}} = \delta^{\text{op}}, -X$$

$$(-X, \delta)^{\text{op}} = \delta^{\text{op}}, +X$$

Conversion typing

$$\boxed{\Delta \vdash c : A \Rightarrow B}$$

$$\frac{}{\Delta \vdash \text{id}_\iota : \iota \Rightarrow \iota} \quad \frac{\Delta \ni X}{\Delta \vdash \text{id}_X : X \Rightarrow X}$$

$$\frac{\Delta \ni X := A}{\Delta \vdash +X : X \Rightarrow A}$$

$$\frac{\Delta \ni X := A}{\Delta \vdash -X : A \Rightarrow X}$$

$$\frac{\Delta \vdash c : A' \Rightarrow A \quad \Delta \vdash d : B \Rightarrow B'}{\Delta \vdash c \rightarrow d : (A \rightarrow B) \Rightarrow (A' \rightarrow B')}$$

$$\frac{\Delta, X \vdash c : A \Rightarrow B}{\Delta \vdash \forall X. c : (\forall X. A) \Rightarrow (\forall X. B)}$$

Term typing

$$\boxed{\Delta \mid \Gamma \vdash M : A}$$

$$\frac{x : A \in \Gamma}{\Delta \mid \Gamma \vdash x : A} \quad \frac{}{\Delta \mid \Gamma \vdash k : \iota}$$

$$\frac{\Delta \vdash A \quad \Delta \mid \Gamma, x : A \vdash N : B}{\Delta \mid \Gamma \vdash \lambda x : A. N : A \rightarrow B} \quad \frac{\Delta \mid \Gamma \vdash L : A \rightarrow B \quad \Delta \mid \Gamma \vdash M : A}{\Delta \mid \Gamma \vdash LM : B}$$

$$\frac{\Delta, X \mid \Gamma \vdash N : B}{\Delta \mid \Gamma \vdash \Lambda X. N : \forall X. B} \quad \frac{\Delta \mid \Gamma \vdash L : \forall X. B \quad \Delta \vdash A}{\Delta \mid \Gamma \vdash LA : B[X \mapsto A]}$$

$$\Delta \vdash \delta \quad \delta(\Delta), \text{rev}(\Theta) \mid \emptyset \vdash M : A$$

$$\frac{\delta^+(\Delta) \vdash \Theta \quad \delta^+(\Delta), \text{rev}(\Theta) \vdash c : A \Rightarrow B \quad \Delta \vdash B}{\Delta \mid \Gamma \vdash \delta_{\nu\Theta}^+[M]_c : B}$$

Values, Inert and Active Conversions

$$\begin{array}{lll} \text{values} & V, W & ::= k \mid \lambda x:A. N \mid \Lambda X. V \mid \delta_{\nu\Theta}^{\delta}[V]_{c^i} \\ \text{inert} & c^i & ::= \text{id}_X \mid -X \mid c \rightarrow d \mid \forall X. c \\ \text{active} & c^a & ::= \text{id}_\ell \mid +X \end{array}$$

A boundary is a value exactly when its interior is a value
and its conversion has inert form c^i .

Term substitution

$$M[x \mapsto V : A]$$

$$x[x \mapsto V : A] = V$$

$$y[x \mapsto V : A] = y \quad \text{if } y \neq x$$

$$(\lambda y:B. N)[x \mapsto V : A] = \lambda y:B. N[x \mapsto V : A]$$

$$(LM)[x \mapsto V : A] = L[x \mapsto V : A] M[x \mapsto V : A]$$

$$\delta_{\nu\Theta}[M]_c[x \mapsto V : A] = \delta_{\nu\Theta}[M]_c$$

$$(\Lambda Y. N)[x \mapsto V : A] = \Lambda Y. N[x \mapsto \bar{\nu}^Y_{\emptyset}[V]_{\text{Id}(A)} : A]$$

Reduction: elimination rules

$$\boxed{\Delta \vdash M \longrightarrow N}$$

$$\Delta \vdash (\lambda x:A. N) W \longrightarrow N[x \mapsto W : A]$$

$$\Delta \vdash \delta_{\nu\Theta} \left[\boxed{V} \right]_{c \rightarrow d} W \longrightarrow \delta_{\nu\Theta} \left[V \stackrel{\delta^{\text{op}}, \Theta^{\text{op}}}{\nu\emptyset} \left[\boxed{W} \right]_c \right]_d$$

$$\Delta \vdash (\Lambda X. \boxed{V_B}) A \longrightarrow \emptyset_{\nu X:=A} \left[\boxed{V} \right]_{X \uparrow B}$$

$$\Delta \vdash \delta_{\nu\Theta} \left[\Lambda X. \boxed{V} \right]_{\forall X. c} A \longrightarrow \delta_{\nu X:=A, \Theta} \left[\boxed{V} \right]_{X \uparrow c}$$

$$\Delta \vdash \delta_{\nu\Theta} \left[\delta'_{\nu\Theta'} \left[\boxed{W} \right]_{\forall X. d} \right]_{\forall Y. c} A \longrightarrow \delta_{\nu Y:=A, \Theta} \left[\delta', -^Y_{\nu\Theta'} \left[\boxed{W} \right]_{\forall X. d} Y \right]_{Y \uparrow c}$$

Reduction: active conversions and congruence

$$\boxed{\Delta \vdash M \longrightarrow N}$$

$$\Delta \vdash \left[\begin{array}{c} \delta \\ \nu\Theta \end{array} \left[\begin{array}{c} k \end{array} \right]_{\text{id}_\ell} \right] \longrightarrow k$$

$$\Delta \vdash \left[\begin{array}{c} \delta_2 \\ \nu\Theta_2 \end{array} \left[\begin{array}{c} \delta_1 \\ \nu\Theta_1 \end{array} \left[\begin{array}{c} V \end{array} \right]_{-X} \right]_{+X} \right] \longrightarrow \left[\begin{array}{c} \delta_2^{\text{op}}, \delta_2 \\ \nu\Theta_2 \end{array} \left[\begin{array}{c} \delta_1, \delta_2 \\ \nu\Theta_1 \end{array} \left[\begin{array}{c} V \end{array} \right]_{\text{Id}(A)} \right]_{\text{Id}(A)} \right]$$

if $\Delta \ni X := A$

$$\Delta \vdash \left[\begin{array}{c} \delta_2 \\ \nu\Theta_2 \end{array} \left[\begin{array}{c} \delta_1 \\ \nu\Theta_1 \end{array} \left[\begin{array}{c} V \end{array} \right]_{\text{id}_X} \right]_{+X} \right] \longrightarrow \left[\begin{array}{c} \delta_2^{\text{op}}, \delta_2 \\ \nu\Theta_2 \end{array} \left[\begin{array}{c} \delta_1, \delta_2 \\ \nu\Theta_1 \end{array} \left[\begin{array}{c} V \end{array} \right]_{+X} \right]_{\text{Id}(A)} \right]$$

if $\Delta \ni X := A$

Reduction: congruence rules

$$\boxed{\Delta \vdash M \longrightarrow N}$$

$$\Delta \vdash LM \longrightarrow L' M \quad \text{if } \Delta \vdash L \longrightarrow L'$$

$$\Delta \vdash VM \longrightarrow VM' \quad \text{if } \Delta \vdash M \longrightarrow M'$$

$$\Delta \vdash LA \longrightarrow L' A \quad \text{if } \Delta \vdash L \longrightarrow L'$$

$$\Delta \vdash \Lambda X. N \longrightarrow \Lambda X. N' \quad \text{if } \Delta, X \vdash N \longrightarrow N'$$

$$\Delta \vdash \delta_{\nu\Theta}[M]_c \longrightarrow \delta_{\nu\Theta}[M']_c \quad \text{if } \delta(\Delta), \text{rev}(\Theta) \vdash M \longrightarrow M'$$

Progress

Every closed, well-typed term is a value or can take a reduction step.

If $\Delta \mid \emptyset \vdash M : A$
then M is a value or $\exists M'. \Delta \vdash M \longrightarrow M'$.

Preservation

A reduction step preserves the type of a closed term.

If $\Delta \mid \emptyset \vdash M : A$ and $\Delta \vdash M \longrightarrow M'$
then $\Delta \mid \emptyset \vdash M' : A$.

Determinism

Every term has at most one immediate reduct.

If $\Delta \vdash M \longrightarrow M_1$ and $\Delta \vdash M \longrightarrow M_2$
then $M_1 = M_2$.

Other Properties?

What about the thing I promised at the beginning!

The set of type variables in scope at every subterm (not include boundaries) is invariant under reduction.

(Sorry, forgot to prove that! Will do next week.)

Collaborating with Phil, Peter, and Claude

