Polymorphic Contracts

João Belo, *Michael Greenberg*,
Atsushi Igarashi, Benjamin Pierce

ESOP 2011

$$
\begin{aligned}
\text{NAT} \;:\; \exists\alpha.\; & Z:\alpha \\
& S:\alpha \to \alpha \\
& \mathit{isZero}:\alpha \to \text{Bool} \\
& \mathit{pred}:\alpha \to \alpha \\
& \leq:\alpha \to \alpha \to \text{Bool} \\
& \mathit{sub}:\alpha \to \alpha \to \alpha
\end{aligned}
$$

$$\frac{}{Z : \mathrm{Nat}} \qquad \frac{e : \mathrm{Nat}}{S\ e : \mathrm{Nat}}$$

$$\frac{e : \mathrm{Nat}}{\mathrm{pred}\ e : \mathrm{Nat}} \qquad \frac{e_1 : \mathrm{Nat} \qquad e_2 : \mathrm{Nat}}{\mathrm{sub}\ e_1\ e_2 : \mathrm{Nat}}$$

"judgments as types"
—*Harper, Honsell, Plotkin 1993*

$$\text{pred (S Z)} \longmapsto^* \text{Z}$$

pred (S Z) $\longmapsto^{*}$ Z

pred Z $\longmapsto^{*}$ Z

$$\frac{}{Z : Nat}$$

$$\frac{e : Nat}{S\ e : Nat}$$

$$\frac{e : \{x{:}Nat \mid x \neq 0\}}{pred\ e : Nat}$$

$$\frac{e_1 : Nat \qquad e_2 : \{y{:}Nat \mid y \leq e_1\}}{sub\ e_1\ e_2 : \{z{:}\ Nat \mid z \leq e_1\}}$$

$$
\begin{aligned}
\mathrm{NAT} \;:\; \exists\alpha.\;& Z{:}\alpha \\
& S{:}\alpha \to \alpha \\
& \mathrm{isZero}{:}\alpha \to \mathrm{Bool} \\
& \mathrm{pred}{:}\{x{:}\alpha \mid \mathrm{not}\ (\mathrm{isZero}\ x)\} \to \alpha \\
& {\leq}{:}\alpha \to \alpha \to \mathrm{Bool} \\
& \mathrm{sub}{:}(x{:}\alpha) \to \{y{:}\alpha \mid y \leq x\} \to \{z{:}\alpha \mid z \leq x\}
\end{aligned}
$$

First-order contracts

First-order contracts

assert($n \geq 0$)

First-order contracts

assert($n \geq 0$)

sqrt : $\{x{:}\mathrm{Float} \mid x \geq 0\} \rightarrow \mathrm{Float}$

First-order contracts

assert($n \geq 0$)

sqrt : $\{x{:}\text{Float} \mid x \geq 0\} \rightarrow \text{Float}$

sqrt : $(x{:}\{x{:}\text{Float} \mid x \geq 0\}) \rightarrow \{y{:}\text{Float} \mid \text{abs}(y^2 - x) < \epsilon\}$

Higher-order contracts

fixed : (f:{x:Int | $x \geq 0$}→{x:Int | $x \geq 0$}) → {y:Int | y=f y}

You give a function *f* on Nats, I return a fixed point of *f*

- If you don't get a fixed point of *f*, oops—you blame me
- If *f* is called with a negative number, oops—you blame me
- If *f* returns a negative, oops—I blame you

Higher-order contracts

fixed : (f:{x:Int | $x \geq 0$}→{x:Int | $x \geq 0$}) → {y:Int | y=f y}

You give a function *f* on Nats, I return a fixed point of *f*

- If you don't get a fixed point of *f*, oops—you blame me
- If *f* is called with a negative number, oops—you blame me
- If *f* returns a negative, oops—I blame you

Higher-order contracts

fixed : (f:{x:Int | $x \geq 0$}→{x:Int | $x \geq 0$}) → {y:Int | y=f y}

You give a function *f* on Nats, I return a fixed point of *f*

If you don't get a fixed point of *f*, oops—you blame me

Higher-order contracts

fixed : (f:{x:Int | $x \geq 0$}→{x:Int | $x \geq 0$}) → {y:Int | y=f y}

You give a function *f* on Nats, I return a fixed point of *f*

If you don't get a fixed point of *f*, oops—you blame me

If *f* is called with a negative number, oops—you blame me

Higher-order contracts

$$\text{fixed} : (f\!:\!\{x\!:\!\text{Int} \mid x \geq 0\} \to \{x\!:\!\text{Int} \mid x \geq 0\}) \to \{y\!:\!\text{Int} \mid y = f\ y\}$$

You give a function *f* on Nats, I return a fixed point of *f*

- If you don't get a fixed point of *f*, oops—you blame me
- If *f* is called with a negative number, oops—you blame me
- If *f* returns a negative, oops—I blame you

Our work

Combine polymorphism and contracts

Manifest contracts

Contracts = Types

Manifest contracts

Types $B ::= \mathrm{Bool} \mid \mathrm{Int} \mid \ldots$

$T ::= \{x{:}B \mid e\} \mid x{:}T_1 \to T_2$

Terms $e ::= \ldots \mid \langle T_1 \Rightarrow T_2 \rangle^{\ell} \mid \Uparrow \ell$

$\mathrm{fixed} : (f{:}(\{x{:}\mathrm{Int} \mid x \geq 0\} \to \{x{:}\mathrm{Int} \mid x \geq 0\})) \to \{y{:}\mathrm{Int} \mid y = f\ y\}$

Refinements

$$<\{x{:}\mathrm{Int} \mid \mathrm{true}\} \Rightarrow \{x{:}\mathrm{Int} \mid x \geq 0\}>^{\ell}\ 2 \longmapsto^{*} 2$$

Refinements

$\langle \{x{:}\mathrm{Int} \mid \mathrm{true}\} \Rightarrow \{x{:}\mathrm{Int} \mid x \geq 0\} \rangle^{\ell}\ 2 \longmapsto^{*} 2$

$\langle \{x{:}\mathrm{Int} \mid \mathrm{true}\} \Rightarrow \{x{:}\mathrm{Int} \mid x \geq 0\} \rangle^{\ell}\ {-5} \longmapsto^{*} \Uparrow \ell$

Functions

$<x{:}T_1 \to T_2 \Rightarrow x{:}U_1 \to U_2>^{\ell}\ f \longmapsto$

$\lambda x{:}\ U_1.\ <T_2[{}^{<U_1 \Rightarrow T_1>^{\ell}\ x}/_{x}] \Rightarrow U_2>^{\ell}\ (f\ (<U_1 \Rightarrow T_1>^{\ell}\ x))$

Unwind contravariantly; extra cast in the codomain

Motivated by typing rules; see *Greenberg, Pierce, and Weirich 2010*

Our work

Add polymorphism to a manifest calculus

Adding polymorphism

Types $B ::= \mathrm{Bool} \mid \mathrm{Int} \mid \ldots$

$T ::= \{x{:}B \mid e\} \mid x{:}T_1 \rightarrow T_2$

Terms $e ::= \ldots \mid \langle T_1 \Rightarrow T_2 \rangle^{\ell} \mid \Uparrow \ell$

Adding polymorphism

Types $B ::= \mathrm{Bool} \mid \mathrm{Int} \mid \ldots$

$T ::= \{x{:}B \mid e\} \mid x{:}T_1 \to T_2$

$\mid \alpha \mid \forall\alpha.T$

Terms $e ::= \ldots \mid \langle T_1 \Rightarrow T_2 \rangle^{\ell} \mid \Uparrow \ell$

$\mid \Lambda\alpha.\, e \mid e\, T$

Adding polymorphism

Types $B ::= \mathrm{Bool} \mid \mathrm{Int} \mid \ldots$

$T ::= \{x{:}B \mid e\} \mid x{:}T_1 \rightarrow T_2$

$\mid \alpha \mid \forall\alpha.T$

Terms $e ::= \ldots \mid \langle T_1 \Rightarrow T_2 \rangle^{\ell} \mid \Uparrow \ell$

$\mid \Lambda\alpha.\, e \mid e\, T$

No interaction:
need to put refinements on type variables!

Adding polymorphism

Types $T ::= \mathrm{Bool} \mid \mathrm{Int} \mid \ldots$

$\mid \{x{:}T \mid e\} \mid x{:}T_1 \to T_2$

$\mid \alpha \mid \forall\alpha.T$

Terms $e ::= \ldots \mid \langle T_1 \Rightarrow T_2 \rangle^{\ell} \mid \Uparrow \ell$

$\mid \Lambda\alpha.\, e \mid e\, T$

Adding polymorphism

In the paper:

Op. sem. for general refinements

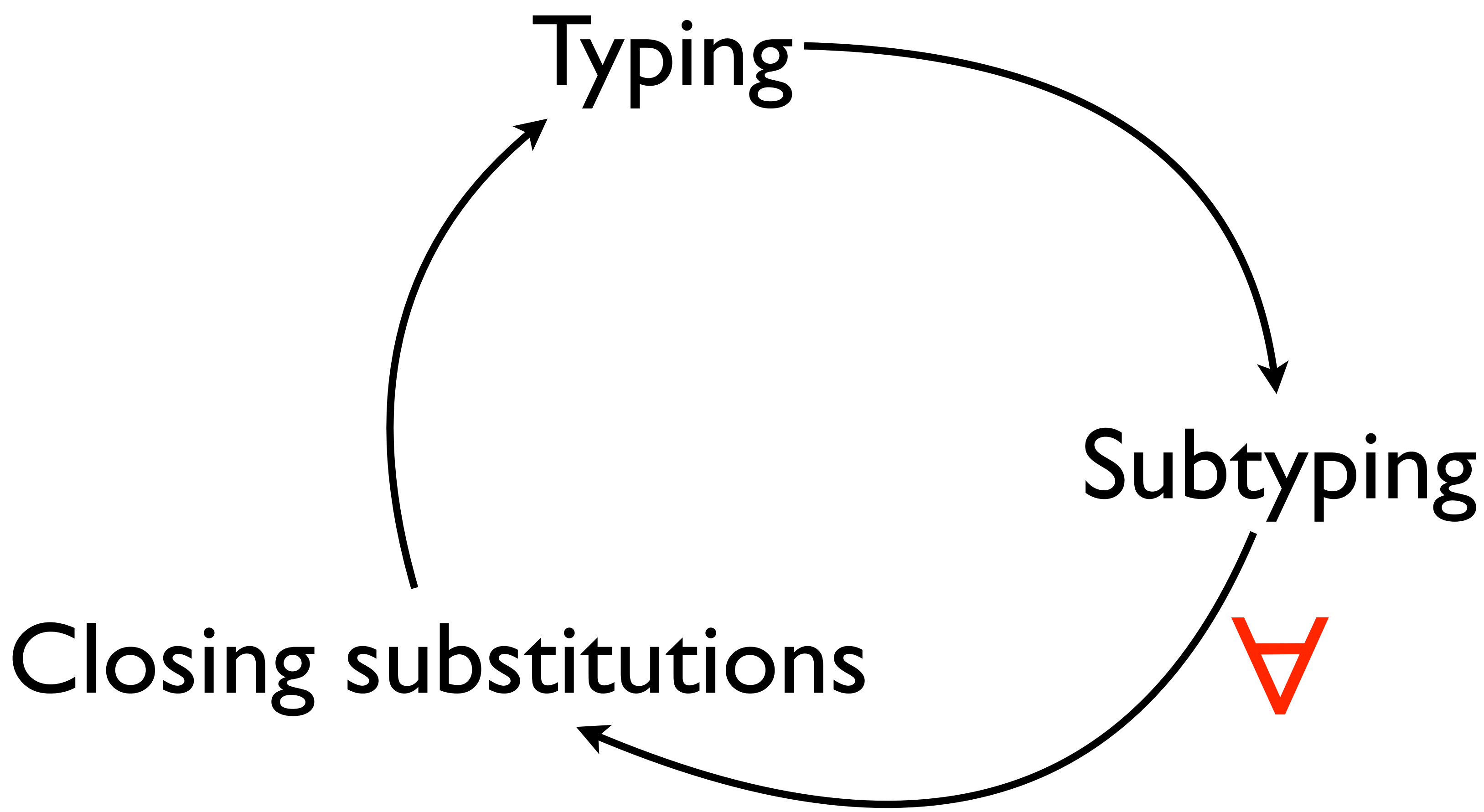
Syntactic type soundness proof

Proof of parametricity

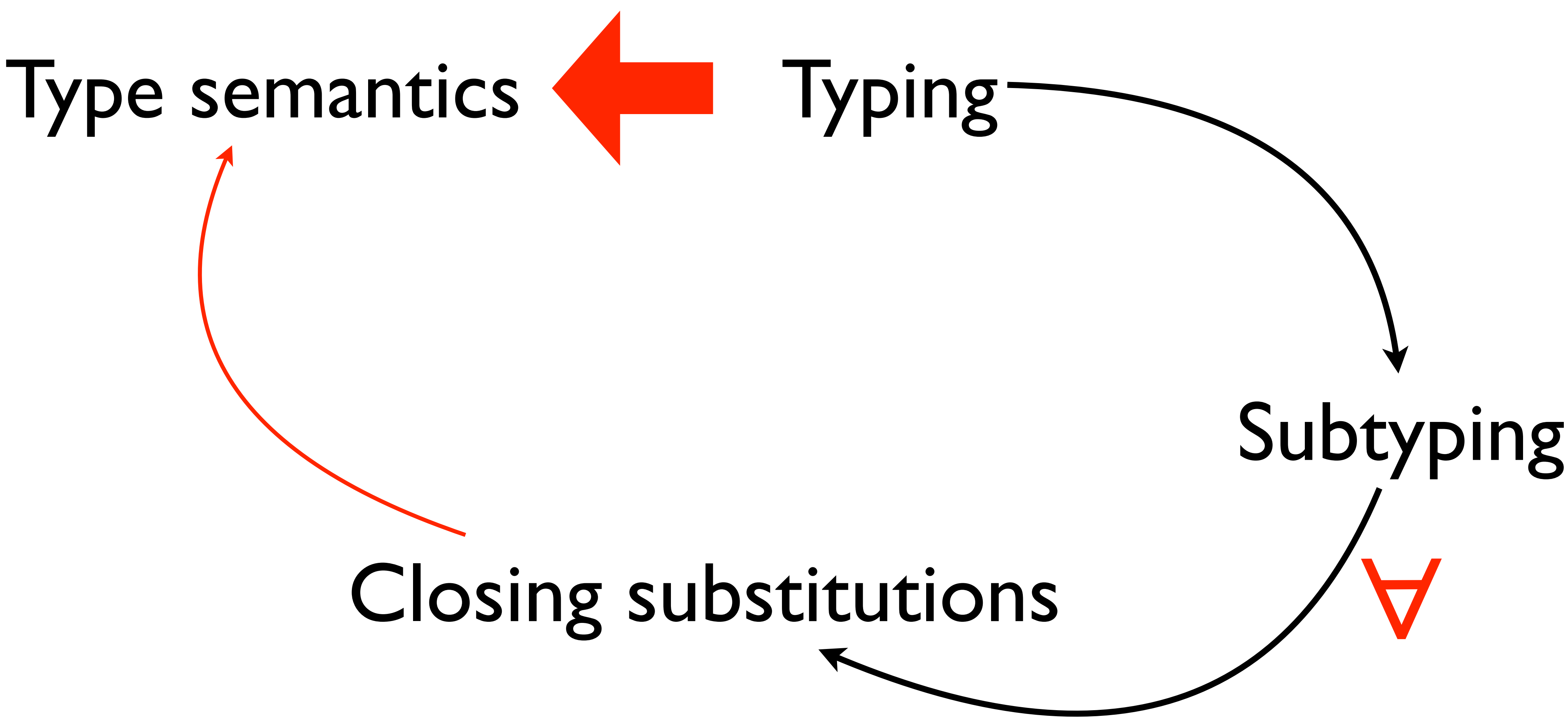
Proof of upcast elimination

See *Knowles and Flanagan 2010*

Type soundness



Type soundness



Greenberg, Pierce, Weirich 2010;
Knowles and Flanagan 2010

Type soundness, take 2

Parallel reduction Typing

Logical relation

Subtyping

Closing substitutions

Our contribution

Parametrically polymorphic manifest calculus

Same great theorems

New and improved metatheory

Outlook

Contracts + polymorphism

=

better guarantees for ADTs

=

better encodings of DSLs

$$\frac{}{c_1 : T} \qquad \frac{e : T}{c_2\ e : T}$$

$$\frac{e_1 : T \quad e_2 : T \quad P_1(e_1, e_2)}{c_3\ e_1\ e_2 : T} \qquad \frac{e_1 : T \quad e_2 : T \quad P_2(e_1) \quad P_2(e_2)}{c_4\ e_1\ e_2 : T}$$

$P_2(c_4\ e_1\ e_2)$

$$
\begin{aligned}
T : \exists\alpha.\ & c_1 : \alpha \\
& c_2 : \alpha \to \alpha \\
& p_1 : \alpha \to \alpha \to \mathrm{Bool} \\
& c_3 : (x{:}\alpha) \to \{y{:}\alpha \mid p_1\ x\ y\} \to \alpha \\
& p_2 : \alpha \to \mathrm{Bool} \\
& c_4 : \{x{:}\alpha \mid p_2\ x\} \to \{y{:}\alpha \mid p_2\ y\} \to \{z{:}\alpha \mid p_2\ z\}
\end{aligned}
$$

Appendix

Using polymorphism

Standard encodings:

- Existentials ($\exists\alpha.T$, pack, unpack)
- Products ($T_1 \times T_2$ and $\Sigma x{:}T_1.T_2$, fst, snd)
- Sums ($T_1 + T_2$, in_L, in_R)

Using polymorphism

$$
\begin{aligned}
\mathrm{NAT} \;:\; \exists\alpha.\; & Z{:}\alpha \\
& S{:}\alpha \to \alpha \\
& \mathrm{isZero}{:}\alpha \to \mathrm{Bool} \\
& \mathrm{pred}{:}\{x{:}\alpha \mid \mathrm{not}\ (\mathrm{isZero}\ x)\} \to \alpha \\
& {\leq}{:}\alpha \to \alpha \to \mathrm{Bool} \\
& \mathrm{sub}{:}(x{:}\alpha) \to \{y{:}\alpha \mid y \leq x\} \to \{z{:}\alpha \mid z \leq x\}
\end{aligned}
$$

Using polymorphism

NAT = < Z=...,

S=...,

isZero=λn:Nat. ...,

pred=...,

≤=λm:Nat. λn:Nat. ...,

sub=...> pack as ∃α. ...

Interfaces

sub : $(x{:}\ \mathbb{N}) \rightarrow \{y{:}\ \mathbb{N} \mid y \leq x\} \rightarrow \{z{:}\ \mathbb{N} \mid z \leq x\}$

Interfaces

$\text{sub} : (x{:}\, N) \to \{y{:}\, N \mid y \leq x\} \to \{z{:}\, N \mid z \leq x\}$

$\text{sub} = \langle N \to N \to N \Rightarrow (x{:}N) \to \{y{:}N \mid y \leq x\} \to \{z{:}N \mid z \leq x\} \rangle^{\ell}\ \text{sub}'$

$\text{sub}' = \lambda m{:}N.\ \lambda n{:}N.\ \ldots$

Obligations

$$\text{sub}:(x:\alpha) \rightarrow \{y:\alpha \mid y \leq x\} \rightarrow \{z:\alpha \mid z \leq x\}$$

Obligations

$$sub{:}(x{:}\alpha) \to \{y{:}\alpha \mid y \leq x\} \to \{z{:}\alpha \mid z \leq x\}$$

Positive positions—server's responsibility
Negative positions—client's responsibility

cf. Findler and Felleisen 2002