Shifting and Resetting in the Calculus of Constructions

Youyou Cong Kenichi Asai

June 11, 2018

Motivation

Dependent Types + **Control Operators**

ensure safety — increase convenience

$\Longrightarrow$ Efficient implementation of safe programs?

Challenge 1: Logical consistency

Herbelin '05: Σ + equality + `call/cc` $= \bot$

- Proofs may have different witnesses
- Need to restrict dependency to pure terms (Herbelin '12)

Challenge 2: CPS semantics

Barthe & Uustalu '02: Standard CPS fails

- CPS changes interface to values
  $e : A \rightsquigarrow v : A \quad \xrightarrow{\text{CPS}} \quad \lambda k.\, e : \neg\neg A \rightsquigarrow \, ???$
- Need tricks for value-extraction
  (Bowman et al. '18)

This work

Calculus of Constructions + shift/reset

- What restrictions do we need?
- How do we obtain a type-preserving CPS?

shift and reset

shift ($\mathcal{S}$) and reset ($\langle\ \rangle$)

$$\begin{aligned} E[(\lambda x.\, e)\ v] &\rightsquigarrow E[e\,[v/x]] \\ E[\langle F[\mathcal{S}k.\, e]\rangle] &\rightsquigarrow E[\langle e\,[\lambda x.\, \langle F[x]\rangle/k]\rangle] \end{aligned}$$

E.g., $1 + \langle 2 + \mathcal{S}k.\, k\ (k\ 3)\rangle$

$\rightsquigarrow^* 1 + \langle 2 + (2 + 3)\rangle$

$\rightsquigarrow^* 8$

Pure and impure terms

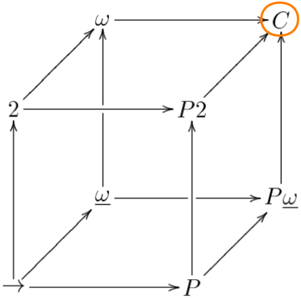
Pure	Impure
$\lambda x.\, x$	$\mathcal{S}k.\, y$
$\langle \mathcal{S}k.\, x \rangle$	$(\lambda x.\, x)\ (\mathcal{S}k.\, y)$
$(\lambda x.\, x)\ y$	$(\lambda x.\, \mathcal{S}k.\, x)\ y$

Pure and impure terms

Pure	Impure
$\lambda x.\, x$	$\mathcal{S}k.\, y$
$\langle \mathcal{S}k.\, x \rangle$	$(\lambda x.\, x)\ (\mathcal{S}k.\, y)$
$(\lambda x.\, x)\ y$	$(\lambda x.\, \mathcal{S}k.\, x)\ y$
$\Gamma \vdash_p e : A$	$\Gamma \vdash_{i(\alpha,\beta)} e : A$

$CC^{s/r}$: a shift/reset-extension of CC

The language



+ shift/reset

Key result: Need 3 restrictions on type dependency

1. Types do not depend on impure terms

$\mathbf{Vec}\ \mathbb{N}\ \langle \boldsymbol{\mathcal{S}k.\,1} \rangle$: ✓ $\mathbf{Vec}\ \mathbb{N}\ (\boldsymbol{\mathcal{S}k.\,1})$: ×

Reason: Impure indices are not informative

(What is the length of $\boldsymbol{v}$: $\mathbf{Vec}\ \mathbb{N}\ (\boldsymbol{\mathcal{S}k.\,1})$?)

2. Continuations are non-dependent functions

$$\mathcal{S}k.\, e$$

$$\uparrow$$

Must not have type $\Pi\, x : A.\, \alpha$

Reason: No closed α for $\Gamma \vdash_{i(\alpha,\beta)} \mathcal{S}k.\, e : A$

Cf. `call/cc` $: ((A \to \bot) \to A) \to A$

3. Answers do not depend on continuations

$$\mathcal{S}k.\,e$$

$$\uparrow$$

Must not depend on k

Reason: No closed β for $\Gamma \vdash_{i(\alpha,\beta)} \mathcal{S}k.\,e : A$

Cf. `call/cc` $: ((A \to \bot) \to A) \to A$

Enforcing Restriction 1

$$\frac{\Gamma \vdash_p e_0 : \Pi\, x : A.\, B \qquad \Gamma \vdash_p e_1 : A}{\Gamma \vdash_p e_0\ e_1 : B\,[e_1/x]}\ (\textsc{E-App1})$$

$$\frac{\Gamma \vdash_p e_0 : A \to B \qquad \Gamma \vdash_{i(\alpha,\beta)} e_1 : A}{\Gamma \vdash_{i(\alpha,\beta)} e_0\ e_1 : B}\ (\textsc{E-App2})$$

Enforcing Restriction 1

$$\frac{\Gamma \vdash_p e_0 : \Pi x : A.\, B \qquad \Gamma \vdash_p e_1 : A}{\Gamma \vdash_p e_0\ e_1 : B\,[e_1/x]}\ (\text{E-App1})$$

$$\frac{\Gamma \vdash_p e_0 : A \to B \qquad \Gamma \vdash_{i(\alpha,\beta)} e_1 : A}{\Gamma \vdash_{i(\alpha,\beta)} e_0\ e_1 : B}\ (\text{E-App2})$$

Enforcing Restriction 1

$$\frac{\Gamma \vdash_p e_0 : \Pi\, x : A.\, B \qquad \Gamma \vdash_p e_1 : A}{\Gamma \vdash_p e_0\ e_1 : B\,[e_1/x]}\ (\text{E-App1})$$

$$\frac{\Gamma \vdash_p e_0 : A \to B \qquad \Gamma \vdash_{i(\alpha,\beta)} e_1 : A}{\Gamma \vdash_{i(\alpha,\beta)} e_0\ e_1 : B}\ (\text{E-App2})$$

Enforcing Restriction 1

$$\frac{\Gamma \vdash_p e_0 : \Pi\, x : A.\, B \qquad \Gamma \vdash_p e_1 : A}{\Gamma \vdash_p e_0\ e_1 : B\,[e_1/x]}\ (\text{E-App1})$$

$$\frac{\Gamma \vdash_p e_0 : A \to B \qquad \Gamma \vdash_{i(\alpha,\beta)} e_1 : A}{\Gamma \vdash_{i(\alpha,\beta)} e_0\ e_1 : B}\ (\text{E-App2})$$

Enforcing Restriction 1

$$\frac{\Gamma \vdash_p e_0 : \Pi x : A.\, B \qquad \Gamma \vdash_p e_1 : A}{\Gamma \vdash_p e_0\ e_1 : B\,[e_1/x]}\ (\textsc{E-App1})$$

$$\frac{\Gamma \vdash_p e_0 : A \to B \qquad \Gamma \vdash_{i(\alpha,\beta)} e_1 : A}{\Gamma \vdash_{i(\alpha,\beta)} e_0\ e_1 : B}\ (\textsc{E-App2})$$

This work

Calculus of Constructions + shift/reset

- What restrictions do we need? ✓
- How do we obtain a type-preserving CPS?

CPS for (pure) CC

$$\boldsymbol{e_0\ e_1 \rightsquigarrow \lambda k.\, e_0^{\div}\ (\lambda v_0.\, e_1^{\div}\ (\lambda v_1.\, v_0\ v_1\ k))}$$

CPS for (pure) CC

$$e_0\ e_1 \rightsquigarrow \lambda k.\, {e_0}^{\div}\ (\lambda v_0.\, {e_1}^{\div}\ (\lambda v_1.\, v_0\ v_1\ k))$$

Problem:

$$k : \neg(B\,[e_1/x])^{+},\ \ v_0\ v_1 : \neg\neg B^{+}\,[v_1/x]$$

CPS for (pure) CC

$$e_0\ e_1 \rightsquigarrow \lambda k.\, e_0{}^{\div}\ (\lambda v_0.\, e_1{}^{\div}\ (\lambda v_1.\, v_0\ v_1\ k))$$

Problem:

$$k : \neg(B\,[e_1/x])^{+},\ \ v_0\ v_1 : \neg\neg B^{+}\,[v_1/x]$$
$$v_1 \ = \ \text{value of } e_1{}^{\div} \ = \ e_1{}^{\div}\ \mathbf{id}$$

CPS for (pure) CC

$$e_0\ e_1 \rightsquigarrow \lambda k.\, e_0{}^{\div}\ (\lambda v_0.\, e_1{}^{\div}\ (\lambda v_1.\, v_0\ v_1\ k))$$

Problem:

$$k : \neg(B\,[e_1/x])^+,\ \ v_0\ v_1 : \neg\neg B^+\,[v_1/x]$$

$$v_1 \ = \ \text{value of } e_1{}^{\div} \ = \ e_1{}^{\div}\ \textbf{id} \quad \text{(ill-typed)}$$

CPS for (pure) CC

$$e_0\ e_1 \rightsquigarrow \lambda\, k.\, {e_0}^{\div}\ (\lambda\, v_0.\, {e_1}^{\div}\ (\lambda\, v_1.\, v_0\ v_1\ k))$$

Problem:

$$k : \neg(B\,[e_1/x])^{+},\ \ v_0\ v_1 : \neg\neg B^{+}\,[v_1/x]$$

$$v_1 \ = \ \text{value of } {e_1}^{\div} \ = \ {e_1}^{\div}\ \textbf{id} \quad \text{(ill-typed)}$$

Solution (Bowman et al. '18):

- $e^{\div} : \Pi\, \alpha : *.\, (A \to \alpha) \to \alpha$
- New equivalence/typing rules

Finding a "better" translation

Bowman et al.:

- CPS as a compiler pass
- Need make control flow explicit everywhere

Our work:

- CPS as a `shift`/`reset`-elimination
- Only need to translate impure terms

Selective CPS translation

- Impure terms into CPS
- Type preservation *for free*

$$e_0\ e_1 \rightsquigarrow \lambda k.\, e_0^{\div}\ (\lambda v_0.\, v_0\ e_1^{+}\ k)$$

impure pure

$$k : \neg_\alpha(B\,[e_1/x])^{+},\ \ v_0\ e_1^{+} : \neg_\beta\neg_\alpha B^{+}\,[e_1^{+}/x]$$

where $\neg_\alpha A \stackrel{\text{def}}{\equiv} A \to \alpha$

Takeaway

Let's make dependently typed programming more fun with shift and reset!

- 3 restrictions on type dependency
- Selective CPS translation