A CPS Transformation for Gradual Programs with Evidence

CPSC 539B Final Project

Joey Eremondi

Source Language

Term Syntax

THE UNIVERSITY OF BRITISH COLUMBIA

e	$::=$		
	$\mid$	x	Variables
	$\mid$	b	Booleans
	$\mid$	n	Natural Numbers
	$\mid$	$\lambda x : T.\, e$	Functions
	$\mid$	$e_1 \; e_2$	Function Application
	$\mid$	$e_1 + e_2$	Addition
	$\mid$	$e_1 \stackrel{?}{=} e_2$	Number Equality Test
	$\mid$	$\textbf{if}\, e_1 \,\textbf{then}\, e_2 \,\textbf{else}\, e_3$	Conditionals
	$\mid$	$\langle e_1, e_2 \rangle$	Tuples
	$\mid$	$\pi_1 e$	Tuple First Projection
	$\mid$	$\pi_2 e$	Tuple Second Projection
	$\mid$	$\varepsilon\, e$	Evidence Ascription
	$\mid$	$\textbf{error}$	Runtime Type Error

Type Syntax

$$
\begin{array}{llll}
T & ::= & & \text{Types} \\
 & | & \text{Nat} & \\
 & | & \text{Bool} & \\
 & | & T_1 \to T_2 & \\
 & | & T_1 \times T_2 & \\
 & | & ? & \\
 & & & \\
\varepsilon & ::= & & \\
 & | & \{T\} &
\end{array}
$$

Type Rules

$$\textsc{HastypeAscr}\quad \frac{\Gamma \vdash e : T_2 \qquad \varepsilon \vdash T_1 \cong T_2}{\Gamma \vdash \varepsilon\, e : T_1}$$

$$\textsc{ConsistentEv}\quad \frac{T_3 \sqcap T_1 = T_3 \qquad T_3 \sqcap T_2 = T_3}{\{T_3\} \vdash T_1 \cong T_2}$$

Combining Evidence

$\boxed{T_1 \sqcap T_2 = T_3}$ *(Precision Meet)*

$$\textsc{MeetDynL}\ \frac{}{? \sqcap T = T} \qquad \textsc{MeetDynR}\ \frac{}{T \sqcap ? = T} \qquad \textsc{MeetRefl}\ \frac{}{T \sqcap T = T}$$

$$\textsc{MeetFun}\ \frac{T_1 \sqcap T_1' = T_1'' \qquad T_2 \sqcap T_2' = T_2''}{T_1 \to T_2 \sqcap T_1' \to T_2' = T_1'' \to T_2''}$$

$$\textsc{MeetProd}\ \frac{T_1 \sqcap T_1' = T_1'' \qquad T_2 \sqcap T_2' = T_2''}{T_1 \times T_2 \sqcap T_1' \times T_2' = T_1'' \times T_2''}$$

Semantics

RedAscr

$$\frac{}{\varepsilon_1\,(\varepsilon_2\,e) \longrightarrow (\varepsilon_1 \sqcap \varepsilon_2)\,e}$$

RedAscrFail

$$\frac{\varepsilon_1 \sqcap \varepsilon_2 \;\mathsf{undefined}}{\varepsilon_1\,(\varepsilon_2\,e) \longrightarrow \mathsf{error}}$$

RedAppEv

$$\frac{}{(\varepsilon_1\,(\lambda x : T.\,e))\,(\varepsilon_2\,r) \longrightarrow (\mathsf{cod}\,\varepsilon_2)\,([(\mathsf{dom}\,\varepsilon_1 \sqcap \varepsilon_2)\,r/x]e)}$$

RedAppEvFail

$$\frac{\mathsf{dom}\,\varepsilon_1 \sqcap \varepsilon_2 \;\mathsf{undefined}}{(\varepsilon_1\,(\lambda x : T.\,e))\,(\varepsilon_2\,r) \longrightarrow \mathsf{error}}$$

Examples

$\{\mathsf{Nat}\}\,(\{\mathsf{Bool}\}\,\mathbf{true}) + 0$ typechecks!

$\{\mathsf{Bool}\} \vdash \mathsf{Bool} \cong\ ?$ and $\{\mathsf{Nat}\} \vdash\ ? \cong \mathsf{Bool}$

But: fails at runtime!

$\mathsf{Nat} \sqcap \mathsf{Bool}$ undefined

The Target

Simplified λ^K

$$
\begin{array}{lll}
u,\ k & ::= & \\
& \mid & x \\
& \mid & n \\
& \mid & b \\
& \mid & \mathsf{fix}\, x\, u \\
& \mid & \lambda x_1 \ldots x_i .\, t \\
& \mid & \langle u_1, u_2 \rangle
\end{array}
$$

$$
\begin{array}{lll}
d & ::= & \\
& \mid & x := u \\
& \mid & x := \pi_1 u \\
& \mid & x := \pi_2 u \\
& \mid & x := u_1 + u_2 \\
& \mid & x := u_1 \stackrel{?}{=} u_2
\end{array}
$$

$$
\begin{array}{lll}
t & ::= & \\
& \mid & v \\
& \mid & \mathsf{let}\, d\, \mathsf{in}\, t \\
& \mid & u(arg) \\
& \mid & \mathsf{if}\ u\ \mathsf{then}\ t_1\ \mathsf{else}\ t_2 \\
& \mid & \mathsf{halt}\, [u] \\
& \mid & \mathsf{error}
\end{array}
$$

$$
\begin{array}{lll}
arg & ::= & \\
& \mid & u_1, \ldots, u_i
\end{array}
$$

The Translation

THE UNIVERSITY OF BRITISH COLUMBIA

Integer constants $\mathtt{BOOL}, \mathtt{NAT}, \mathtt{ARROW}, \mathtt{PRODUCT}, \mathtt{DYN}$

$\boxed{[\![\varepsilon]\!] = u}$ *(CPS Representation of Runtime Evidence)*

$$\textsc{EvTransformBool}\quad \frac{}{[\![\{\mathrm{Bool}\}]\!] = \langle \mathtt{BOOL}, 0\rangle}$$

$$\textsc{EvTransformNat}\quad \frac{}{[\![\{\mathrm{Nat}\}]\!] = \langle \mathtt{NAT}, 0\rangle}$$

$$\textsc{EvTransformDyn}\quad \frac{}{[\![\{?\}]\!] = \langle \mathtt{DYN}, 0\rangle}$$

$$\textsc{EvTransformArr}\quad \frac{}{[\![\{T_1 \to T_2\}]\!] = \langle \mathtt{ARROW}, \langle [\![\{T_1\}]\!], [\![\{T_2\}]\!]\rangle\rangle}$$

$$\textsc{EvTransformProd}\quad \frac{}{[\![\{T_1 \to T_2\}]\!] = \langle \mathtt{PRODUCT}, \langle [\![\{T_1\}]\!], [\![\{T_2\}]\!]\rangle\rangle}$$

Evidence Operations

$\texttt{MEET}(u_1, u_2, k)$

Combines evidence representation u_1 and u_2, gives result to k

Passes control **error** continuation if meet undefined

Similar for `DOM`, `COD` to decompose function types

$\boxed{[\![v]\!] = u}$ *(CPS Translation of Closed Values)*

$$\textsc{ValTransformBool}\quad \frac{}{[\![b]\!] = \langle \texttt{DYN}, b\rangle}$$

$$\textsc{ValTransformNum}\quad \frac{}{[\![n]\!] = \langle \texttt{DYN}, n\rangle}$$

$$\textsc{ValTransformFun}\quad \frac{[\![e]\!]c = t}{[\![\lambda x : T.\, e]\!] = \langle \texttt{DYN}, \lambda x\, c.\, t\rangle}$$

$$\textsc{ValTransformPair}\quad \frac{}{[\![\langle v_1, v_2\rangle]\!] = \langle \texttt{DYN}, \langle [\![v_1]\!], [\![v_2]\!]\rangle\rangle}$$

$$\textsc{ValTransformEv}\quad \frac{[\![r]\!] = \langle \texttt{DYN}, u\rangle}{[\![\varepsilon\, r]\!] = \langle [\![\varepsilon]\!], u\rangle}$$

Translating Applications

$$
\textsc{TransformApp}\;
\frac{
\begin{array}{c}
k_1 := (\lambda x_2.\, \mathsf{let}\, y_1 := \pi_1 x_1 \,\mathsf{in}\, \mathsf{let}\, z_1 := \pi_2 x_1 \,\mathsf{in}\, \mathsf{let}\, y_2 := \pi_1 x_2 \,\mathsf{in}\, \mathsf{let}\, z_2 := \pi_2 x_2 \,\mathsf{in}\, t_1) \\
t_1 := \mathtt{DOM}(y_1, \lambda y_1'.\, \mathtt{COD}(y_1, \lambda y_1''.\, \mathtt{MEET}(y_1', y_2, (\lambda y_3.\, z_1(\langle y_3, z_2 \rangle, (\lambda z_3.\, t_2)))))) \\
t_2 := \mathsf{let}\, z_3' := \pi_1 z_3 \,\mathsf{in}\, \mathsf{let}\, z_3'' := \pi_2 z_3 \,\mathsf{in}\, \mathtt{MEET}(y_1'', z_3', (\lambda z_4.\, k(\langle z_4, z_3'' \rangle))) \\
[\![e_2]\!] k_1 = t' \\
[\![e_1]\!] (\lambda x_1.\, t') = t
\end{array}
}{
[\![e_1\ e_2]\!] k = t
}
$$

Correctness

Key Lemmas

THE UNIVERSITY OF BRITISH COLUMBIA

$\texttt{MEET}(\llbracket\varepsilon_1\rrbracket, \llbracket\varepsilon_2\rrbracket, k) \longrightarrow^* k(\llbracket\varepsilon_1 \sqcap \varepsilon_2\rrbracket)$

If $\varepsilon_1 \sqcap \varepsilon_2$ undefined, then $\texttt{MEET}(\llbracket\varepsilon_1\rrbracket, \llbracket\varepsilon_2\rrbracket, k) \longrightarrow^* \mathsf{error}$

$\llbracket v\rrbracket k \longrightarrow^* k(\llbracket v\rrbracket)$

$\llbracket [v/x]e\rrbracket k \longrightarrow^* [\llbracket v\rrbracket/x]\llbracket e\rrbracket k$

Whole Program Correctness

If $e_1 \longrightarrow e_2$, then for any k, $[\![e_1]\!]k \equiv [\![e_2]\!]k$

Defined in terms of equivalence $\equiv$, symmetric closure of $\longrightarrow^*$

Simulation proved by induction on derivation of $e_1 \longrightarrow e_2$

Corollary: if $eval(e_1) = v$, then $eval([\![e]\!](\lambda x.\, \textbf{halt}\, [x])) = \textbf{halt}\, [[\![v]\!]]$.

Incorrectness

Type Preservation

No notion of consistency in target language

$(\lambda x : ?.\, x\, x)$ typeable in source

Translation has no type in target

Possibly solved by combination of sum and recursive types

$(\lambda x : \mathrm{Nat}.\, x + 0) \approx (\lambda x : \mathrm{Nat}.\, (\{\mathrm{Nat}\}\, x) + 0)$

Distinguish by target context $(\square \, \langle n, 0 \rangle)$ where $n \neq \mathtt{DYN}, n \neq \mathtt{NAT}$

Only causes **error** in second case