Typed Closure Conversion

Def. [Closure conversion]
makes explicit in pgm text
the closures ((code x environment) pairs)

Analogy. [CPS conversion]
makes explicit in pgm text
the continuations (things to do with the current result)

e.g.)

$(\lambda x.x+y)\,1 \Rightarrow$

let
$f = \langle \lambda\langle r, x\rangle.\ x + r.1, \langle y\rangle\rangle$
in
$f.1\ \langle f.2, 1\rangle$

CPS Conversion

Source

$$e \to x \mid \lambda x.e \mid e\,e \mid 1$$

Target

$$e \to x \mid \lambda x.e \mid e\,e \mid 1$$

Closure Conversion

Source

$$e \to x \mid \lambda x.e \mid e\,e \mid 1$$

Target

$$e \to x \mid \lambda\langle x,y\rangle.e \mid e\,e \mid \langle e,\cdots,e\rangle \mid e.n \mid 1$$

Typed Target

$$\mid \text{pack}\,(e,\tau)\ \text{as}\ \exists\alpha.\tau$$

$$\mid \text{let}\ x,\alpha = \text{unpack}\ e\ \text{in}\ e$$

Problem in Typed CC

$$
\begin{array}{l}
\text{if } e \text{ then } \left(\begin{array}{l}\text{let } y=1\\ \text{in } \lambda x.\, x+y\end{array}\right)\ 1\\
\quad\text{else } \left(\begin{array}{l}\text{let } y=\text{true}\\ \text{in } \lambda x.\, \text{if } y \text{ then } x \text{ else } x+1\end{array}\right)\ 1
\end{array}
$$

$$\Downarrow$$

$$
\begin{array}{l}
\text{if } \underline{e} \text{ then } \left(\begin{array}{l}\text{let } y=1\\ \text{in } \langle \lambda\langle r,x\rangle.\, x+r.1,\ \underline{\langle y\rangle}_A\rangle\end{array}\right)\ 1\\
\quad\text{else } \left(\begin{array}{l}\text{let } y=\text{true}\\ \text{in } \langle \lambda\langle r,x\rangle.\, \text{if } y\cdots,\ \underline{\langle y\rangle}_B\rangle\end{array}\right)\ 1
\end{array}
$$

fails to type-check;

the explicit environments $\underline{\langle y\rangle}_A$ & $\underline{\langle y\rangle}_B$ have conflicting types.

Existential Type System

Polymorphic Types = Universally Quantified Types.
Generic Types

$\forall \alpha . \tau$

(Module Types) = Existentially Quantified Types
Abstract Data Types
Package Types

$\exists \alpha . \tau$

$\langle 3, \text{true} \rangle : \exists \alpha . \alpha$

$: \exists \alpha . \alpha \times \text{bool}$

$: \exists \alpha . \text{int} \times \text{bool}$

$: \exists \alpha, \beta . \alpha \times \beta$

Existentially typed objects are useful if the types are sufficiently structured.

e.g.) $x : \exists\alpha.\ \alpha \times (\alpha \to \text{int})$

Then the type gives us an efficient information to allow us to compute with x.

We can write

$(x.2)\,(x.1)$

to compute an integer value.

Existential quantification is an useful typing of an object whose information is not completely uncovered.

i.e. existential quantification ~ information hiding

$$
\begin{aligned}
e \rightarrow\ & 1 \\
\mid\ & x \\
\mid\ & \lambda x.e \\
\mid\ & e\ e \\
\mid\ & \langle e, e \rangle \mid e.1 \mid e.2 \\
\mid\ & \text{pack}\,(e, \tau)\ \text{as}\ \exists \alpha.\tau \\
\mid\ & \text{let}\ x, \alpha = \text{unpack}\ e\ \text{in}\ e
\end{aligned}
$$

$$
\begin{aligned}
\tau \rightarrow\ & \alpha \\
\mid\ & \iota \\
\mid\ & \tau \rightarrow \tau \\
\mid\ & \tau \times \tau \\
\mid\ & \exists \alpha.\tau
\end{aligned}
$$

$$\frac{\Gamma \vdash e : \{\tau_1/\alpha\}\tau_2}{\Gamma \vdash \text{pack}\,(e, \tau_1)\ \text{as}\ \exists\alpha.\tau_2 : \exists\alpha.\tau_2}$$

$$\frac{\Gamma \vdash e_1 : \exists\alpha.\tau \qquad \Gamma, x:\tau \vdash e_2 : \tau'}{\Gamma \vdash \text{let}\ x, \alpha = \text{unpack}\ e_1\ \text{in}\ e_2 : \tau'}$$

e.g.) If e_1 is

$$\text{pack}\,(\langle 1, \lambda x.x+1\rangle, \text{int})\ \text{as}\ \exists\alpha.\ \alpha \times (\alpha \to \text{int})$$

then

$$\text{let}\ x, \alpha = \text{unpack}\ e_1\ \text{in}\ (x.2)\,(x.1)$$

has type int.

Typed CC : The Rules

(monomorphic cases)

$$\frac{\Gamma x = \tau}{\Gamma \vdash x : \tau \rightsquigarrow x} \qquad \frac{}{\Gamma \vdash 1 : \iota \rightsquigarrow 1}$$

$$\frac{\Gamma, x:\tau \vdash e : \tau' \rightsquigarrow e'}{\Gamma \vdash \lambda x:\tau.\, e : \tau \to \tau' \rightsquigarrow \text{pack } (\langle \text{code}, \text{env} \rangle, \tau_n) \text{ as } \underline{\tau \to \tau'}}$$

where

$\Gamma = \{x_1:\tau_1, \cdots, x_n:\tau_n\}$

$\tau_n = \underline{\tau_1} \times \cdots \times \underline{\tau_n}$

$\text{code} = \lambda \langle r, x \rangle : \tau_n \times \underline{\tau}.$

let $x_1 = r.1$

$\vdots$

$x_n = r.n$

in

e'

$\text{env} = \langle x_1, \cdots, x_n \rangle$

$$\frac{\Gamma \vdash e_1 : \tau_2 \to \tau \rightsquigarrow e_1' \qquad \Gamma \vdash e_2 : \tau_2 \rightsquigarrow e_2'}{\Gamma \vdash e_1\, e_2 : \tau \rightsquigarrow \text{let } x, d = \text{unpack } e_1' \text{ in } (x.1)\langle x.2, e_2' \rangle}$$

$$\underline{\bot} = \iota$$

$$\underline{\tau_1 \to \tau_2} = \exists \alpha.\, (\alpha \times \underline{\tau_1} \to \underline{\tau_2}) \times \alpha$$