Closure Conversion

Di Zhao

zhaodi01@mail.ustc.edu.cn

Document for the closure conversion phase.

1 Calculating free variables

The first step to perform closure conversion is to calculate the free variables inside all function definitions. This allows us to decide which values need to be included in environments (and fetched out from the environments). Only after that will we be able to do closure conversion.

The source language in this phase is the continuation-passing language (Figure 1).

$$
\begin{array}{lrcl}
\text{(terms)} & K & \to & \mathsf{letval}\ x\ =\ V\ \ \mathsf{in}\ \ K \\
 & & | & \mathsf{letcont}\ k\ x = K\ \mathsf{in}\ K' \\
 & & | & k\ x \\
 & & | & f\ k\ x \\
 & & | & \mathsf{case}\ x\ \mathsf{of}\ \overrightarrow{\mathtt{in}i_j\ x_j\ \mathtt{=>}\ K_j} \\
 & & | & \mathsf{letprim}\ x = PrimOp\ \vec{y}\ \ \mathtt{in}\ K \\
 & & | & \mathsf{if}\ x\ \mathsf{then}\ k_1\ \mathsf{else}\ k_2 \\
 & & | & \mathsf{letfix}\ f\ k\ x = K\ \mathsf{in}\ K' \\
\text{(values)} & V & \to & ()\ \ |\ \mathsf{true}\ |\ \mathsf{false} \\
 & & | & i\ \ \ |\ \ \ ''s'' \\
 & & | & (x_1, x_2, ..., x_n) \\
 & & | & \mathsf{in}i\ x \\
 & & | & \lambda k\ x.K \\
 & & | & \#i\ x \\
\text{(primitive operations)} & PrimOp & \to & +\ |\ -\ |\ * \\
 & & | & >\ |\ <\ |\ = \\
 & & | & \mathsf{andalso}\ \ |\,\mathsf{orelse}\ \ |\ \ \mathsf{not} \\
 & & | & \mathsf{print}\ |\ \mathsf{int2string}
\end{array}
$$

Figure 1: CPS syntax

A free variable of an expression is one that is not bound but is used inside the expression. Such as the y and k in $\mathsf{letval}\ x = y\ \mathsf{in}\ k\ x$. It's worth noticing here that in $\mathsf{letval}\ x = x\ \mathsf{in}\ k\ x$, the second "$x$" is also a free variable (and thus should also be included in an environment).

Figure 2 and Figure 3 demonstrate the mutual recursive function $\mathcal{F}$ and function $\mathcal{H}$ to calculate the free variables of a CPS term and CPS value respectively. Both $\mathcal{F}$ and $\mathcal{H}$ return an ordered set of variable names. Function `set` generate a set of strings from a string list.

$$
\begin{aligned}
\mathcal{F} &: \text{Cps.t} \to \text{string set} \\
\mathcal{F}(\mathsf{letval}\ x = V\ \mathsf{in}\ K) &= (\mathcal{F}(K)/x) \cup \mathcal{H}(V) \\
\mathcal{F}(\mathsf{letcont}\ k\ x = K\ \mathsf{in}\ K') &= (\mathcal{F}(K)/x) \cup (\mathcal{F}(K')/k) \\
\mathcal{F}(k\ x) &= \{k, x\} \\
\mathcal{F}(f\ k\ x) &= \{f, k, x\} \\
\mathcal{F}(\mathsf{case}\ x\ \mathsf{of}\ \overrightarrow{\mathtt{in}i_j\ x_j\ \texttt{=>}\ K_j}) &= \{x\} \cup (\mathcal{F}(K_1)/x_1) \cup ... \cup (\mathcal{F}(K_n)/x_n) \\
&\quad \text{(where } j = 1,\ ...\ , n) \\
\mathcal{F}(\mathsf{letprim}\ x = PrimOp\ \vec{y}\ \mathtt{in}\ K) &= (\mathcal{F}(K)/x) \cup \mathtt{set}(\vec{y}) \\
\mathcal{F}(\mathsf{if}\ x\ \mathsf{then}\ k_1\ \mathsf{else}\ k_2) &= \{x, k_1, k_2\} \\
\mathcal{F}(\mathsf{letfix}\ f\ k\ x = K\ \mathsf{in}\ K') &= (\mathcal{F}(K) - \{f, k, x\}) \cup (\mathcal{F}(K')/f)
\end{aligned}
$$

Figure 2: Calculating Free Variables in CPS terms

$$
\begin{aligned}
\mathcal{H} &: \text{Cps.v} \to \text{string set} \\
\mathcal{H}(()) &= \emptyset \\
\mathcal{H}(\mathtt{true}) &= \emptyset \\
\mathcal{H}(\mathtt{false}) &= \emptyset \\
\mathcal{H}(i) &= \emptyset \\
\mathcal{H}(\text{"}s\text{"}) &= \emptyset \\
\mathcal{H}((x_1, x_2,\ ...\ , x_n)) &= \{x_1, x_2,\ ...\ , x_n\} \\
\mathcal{H}(\mathsf{in}i\ x) &= \{x\} \\
\mathcal{H}(\texttt{\#}i\ x) &= \{x\} \\
\mathcal{H}(\lambda k\ x.K) &= \mathcal{F}(K) - \{k, x\}
\end{aligned}
$$

Figure 3: Calculating Free Variables in CPS values

2 Closure Syntax

The target language is a closure-passing language. Wherever a function is needed to be returned as a result, or passed as an argument, a corresponding closure will be transmitted instead. Aside from that, the Closure syntax (shown in Figure 4) is similar to the CPS syntax.

$$
\begin{array}{rrcl}
\text{(terms)} & K & \rightarrow & \mathsf{letval}\ x\ =\ V\ \ \mathsf{in}\ \ K \\
 & & | & \mathsf{let}\ x = \#i\ y\ \mathsf{in}\ K \\
 & & | & \mathsf{letcont}\ k\ env\ x = K\ \mathsf{in}\ K' \\
 & & | & k\ env\ x \\
 & & | & f\ env\ k\ x \\
 & & | & \mathsf{case}\ x\ \mathsf{of}\ \overrightarrow{\mathsf{in}i_j\ x_j \Rightarrow K_j} \\
 & & | & \mathsf{letprim}\ x = PrimOp\ \vec{y}\ \mathsf{in}\ K \\
 & & | & \mathsf{if}\ x\ \mathsf{then}\ K\ \mathsf{else}\ K' \\
 & & | & \mathsf{letfix}\ f\ env\ k\ x = K\ \mathsf{in}\ K' \\
\text{(values)} & V & \rightarrow & () \\
 & & | & \mathtt{true} \\
 & & | & \mathtt{false} \\
 & & | & i \\
 & & | & ''s'' \\
 & & | & (x_1, x_2,\ \dots\ , x_n) \\
 & & | & \mathtt{in}i\ x \\
 & & | & \lambda env\ k\ x.K \\
\text{(primitive} & PrimOp & \rightarrow & +\ |\ -\ |\ * \\
\text{operations)} & & | & >\ |\ <\ |\ = \\
 & & | & \mathsf{andalso}\ \ |\ \mathsf{orelse}\ \ |\ \ \mathsf{not} \\
 & & | & \mathsf{print}\ |\ \mathsf{int2string}
\end{array}
$$

Figure 4: Closure syntax

3 Closure Conversion

Updates:

15-7-3: Change the form of `case` in `closure.sig`, `closure.sml`, and changed the respective converting rule in `closure-convert.sml` to enable multiple cases .

15-9-5: Added boolean values and corresponding operations. Changed `if0` expression into `if` expression.

$$
\begin{aligned}
\Theta \;&:\; \text{Cps.t} \to \text{Closure.t} \\
\Theta(\textsf{let val } x = \texttt{\#}i\ y \textsf{ in } K) &= \textsf{let } x = \texttt{\#}i\ y \textsf{ in } \Theta(K) \\
\Theta(\textsf{letcont } k\ x = K \textsf{ in } K') &= \textsf{letcont } k_{code}\ env\ x = \\
&\qquad \texttt{let}\ \ y_1 = \texttt{\#1}\ env\ \texttt{in} \\
&\qquad\quad \texttt{let}\ \ y_2 = \texttt{\#2}\ env\ \texttt{in} \\
&\qquad\quad \ldots \\
&\qquad\qquad \texttt{let}\ \ y_m = \texttt{\#}m\ env\ \texttt{in}\ \Theta(K) \\
&\quad \textsf{in letval } env' = (y_1, y_2,\ \ldots\ , y_m)\ \texttt{ in} \\
&\qquad\qquad \textsf{letval } k = (k_{code}, env')\ \texttt{ in }\ \Theta(K') \\
&\quad (where\ \{y_1,\ \ldots\ , y_m\} = \mathcal{F}(K) - \{x\}) \\
\Theta(k\ x) &= \texttt{let}\ \ k_{code} = \texttt{\#1}\ k\ \texttt{in} \\
&\qquad \texttt{let}\ \ env = \texttt{\#2}\ k\ \texttt{in} \\
&\qquad\quad k_{code}\ env\ x \\
\Theta(f\ k\ x) &= \texttt{let}\ \ f_{code} = \texttt{\#1}\ f\ \texttt{in} \\
&\qquad \texttt{let}\ \ env = \texttt{\#2}\ f\ \texttt{in} \\
&\qquad\quad f_{code}\ env\ k\ x \\
\Theta(\textsf{case } x \textsf{ of } \overrightarrow{\texttt{in}i_j\ x_j\ \texttt{=>}\ K_j}) &= \textsf{case } x \textsf{ of } \overrightarrow{\texttt{in}i_j\ x_j \Rightarrow \Theta(K_j)} \\
\Theta(\textsf{letprim } x = PrimOp\ \vec{y}\ \texttt{ in } K) &= \textsf{letprim } x = PrimOp\ \vec{y}\ \texttt{ in } \Theta(K) \\
\Theta(\textsf{if } x \textsf{ then } k_1 \textsf{ else } k_2) &= \textsf{letval } x_1 = () \textsf{ in} \\
&\qquad \textsf{if } x \textsf{ then } \Theta(k_1\ x_1) \textsf{ else } \Theta(k_2\ x_1) \\
\Theta(\textsf{letfix } f\ k\ x = K \textsf{ in } K') &= \textsf{letfix } f_{code}\ env\ k\ x = \\
&\qquad \texttt{letval}\ \ f = (f_{code}, env)\ \texttt{in} \\
&\qquad\quad \texttt{let}\ \ y_1 = \texttt{\#1}\ env\ \texttt{in} \\
&\qquad\qquad \texttt{let}\ \ y_2 = \texttt{\#2}\ env\ \texttt{in} \\
&\qquad\qquad \ldots \\
&\qquad\qquad\quad \texttt{let}\ \ y_m = \texttt{\#}m\ env\ \texttt{in}\ \Theta(K) \\
&\quad \textsf{in letval } env' = (y_1, y_2,\ \ldots\ , y_m)\ \texttt{ in} \\
&\qquad \textsf{letval } f = (f_{code}, env')\ \texttt{ in }\ \Theta(K') \\
&\quad (where\ \{y_1,\ \ldots\ , y_m\} = \mathcal{F}(K) - \{f, k, x\})
\end{aligned}
$$

Figure 5: Closure Conversion for CPS terms

$$
\begin{aligned}
\Theta\ &:\ \text{Cps.t} \to \text{Closure.t}\\
\Theta(\mathsf{letval}\ x = ()\ \mathsf{in}\ K) &= \mathsf{letval}\ x = ()\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = \mathtt{true}\ \mathsf{in}\ K) &= \mathsf{letval}\ x = \mathtt{true}\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = \mathtt{false}\ \mathsf{in}\ K) &= \mathsf{letval}\ x = \mathtt{false}\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = i\ \mathsf{in}\ K) &= \mathsf{letval}\ x = i\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = "s"\ \mathsf{in}\ K) &= \mathsf{letval}\ x = "s"\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = (x_1, x_2,\ \ldots\ , x_n)\ \mathsf{in}\ K) &= \mathsf{letval}\ x = (x_1, x_2,\ \ldots\ , x_n)\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = \mathsf{in}_i\ y\ \mathsf{in}\ K) &= \mathsf{letval}\ x = \mathsf{in}_i\ y\ \mathsf{in}\ \Theta(K)\\
\Theta(\mathsf{letval}\ x = \lambda k\ z.K\ \mathsf{in}\ K') &= \mathsf{letval}\ x_{code} = \lambda env\ k\ z.\\
&\qquad \mathtt{let}\ \ y_1 = \mathtt{\#}1\ env\ \mathtt{in}\\
&\qquad\quad \mathtt{let}\ \ y_2 = \mathtt{\#}2\ env\ \mathtt{in}\\
&\qquad\quad \ldots\\
&\qquad\qquad \mathtt{let}\ \ y_m = \mathtt{\#}m\ env\ \mathtt{in}\ \Theta(K)\\
&\quad\ \mathsf{in}\ \mathsf{letval}\ env' = (y_1, y_2,\ \ldots\ , y_m)\ \ \mathtt{in}\\
&\qquad\qquad \mathsf{letval}\ x = (x_{code}, env')\ \ \mathtt{in}\ \ \Theta(K')\\
&(where\ \{y_1,\ \ldots\ , y_m\} = \mathcal{F}(K) - \{k, z\})
\end{aligned}
$$

Figure 6: Closure Conversion for CPS terms (continued)