『プログラム言語論』 宿題の解答例
(2013/05/30 出題; 2013/06/06 締め切り, 亀山)

1 関数クロージャに対応した抽象機械

L_2 に対応する CEK 機械 (関数クロージャを持つ抽象機械) で、以下の e_1 と e_2 に対する状態遷移を書きなさい。
(計算は $\langle eval, e, \texttt{[]}, \texttt{init}\rangle$ という状態からはじめる。また、ステップ数が非常に多い場合は、ある程度省略して書いてもよい。)

- e_1: $\texttt{let}\ y = 10\ \texttt{in let}\ f = \lambda x.x + y\ \texttt{in}\ f(f\ 2)$ (具体構文での記述)

$$
\begin{aligned}
&\langle eval, \texttt{let}\ y = 10\ \texttt{in let}\ f = \lambda x.x + y\ \texttt{in}\ f(f\ 2), \texttt{[]}, \texttt{init}\rangle \\
&\to \langle eval, 10, \texttt{[]}, \texttt{push}((letin, y, \texttt{let}\ f = \lambda x.x + y\ \texttt{in}\ f(f\ 2), \texttt{[]}), \texttt{init})\rangle \\
&\to \langle apply, \texttt{push}((letin, y, \texttt{let}\ f = \lambda x.x + y\ \texttt{in}\ f(f\ 2), \texttt{[]}), \texttt{init}), 10\rangle \\
&\to \langle eval, \texttt{let}\ f = \lambda x.x + y\ \texttt{in}\ f(f\ 2), \texttt{[]}[y = 10], \texttt{init}\rangle \\
&\to \langle eval, \lambda x.x + y, \texttt{[]}[y = 10], \texttt{push}((letin, f, f(f\ 2), \texttt{[]}[y = 10]), \texttt{init})\rangle \\
&\to \langle apply, \texttt{push}((letin, f, f(f\ 2), \texttt{[]}[y = 10]), \texttt{init}), \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])\rangle \\
&\to \langle eval, f(f\ 2), \texttt{[]}[y = 10][f = \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])], \texttt{init}\rangle
\end{aligned}
$$

ここで $F_1 = \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])$, $E_1 = \texttt{[]}[y = 10][f = F_1]$ と置いて計算を続ける。

$$
\begin{aligned}
&\to \langle eval, f, E_1, \texttt{push}((apply1, (f\ 2), E_1), \texttt{init})\rangle \\
&\to \langle apply, \texttt{push}((apply1, (f\ 2), E_1), \texttt{init}), F_1\rangle \\
&\to \langle eval, (f\ 2), E_1, \texttt{push}((apply2, F_1)), \texttt{init})\rangle \\
&\to \langle eval, f, E_1, \texttt{push}((apply1, 2, E_1), \texttt{push}((apply2, F_1), \texttt{init}))\rangle \\
&\to \langle apply, \texttt{push}((apply1, 2, E_1), \texttt{push}((apply2, F_1), \texttt{init})), F_1\rangle \\
&\to \langle eval, 2, E_1, \texttt{push}((apply2, F_1), \texttt{push}((apply2, F_1), \texttt{init}))\rangle \\
&\to \langle apply, \texttt{push}((apply2, F_1), \texttt{push}((apply2, F_1), \texttt{init})), 2\rangle \\
&\to \langle eval, x + y, E_1[x = 2], \texttt{push}((apply2, F_1), \texttt{init})\rangle \\
&\to^* \langle apply, \texttt{push}((apply2, F_1), \texttt{init}), 12\rangle \\
&\to^* \langle eval, x + y, E_1[x = 12], \texttt{init}\rangle \\
&\to^* \langle apply, \texttt{init}, 22\rangle \\
&\to 22
\end{aligned}
$$

[2013/6/20; 昨日までここに置いていた解答ファイルでは、最後の最後の値を「12」と書く誤植がありましたので、修正しました。Thanks to 須永君]

- e_2: $\texttt{let}\ y = 10\ \texttt{in let}\ f = \lambda x.x + y\ \texttt{in let}\ y = 20\ \texttt{in}\ f\ 30$ (具体構文での記述)

$$
\begin{aligned}
&\langle eval, \texttt{let}\ y = 10\ \texttt{in let}\ f = \lambda x.x + y\ \texttt{in let}\ y = 20\ \texttt{in}\ f\ 30, \texttt{[]}, \texttt{init}\rangle \\
&\to^* \langle eval, \texttt{let}\ f = \lambda x.x + y\ \texttt{in let}\ y = 20\ \texttt{in}\ f\ 30, \texttt{[]}[y = 10], \texttt{init}\rangle \\
&\to^* \langle eval, (f\ 30), \texttt{[]}[y = 10][f = \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])][y = 20], \texttt{init}\rangle
\end{aligned}
$$

ここで $F_1 = \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])$, $E_2 = \texttt{[]}[y = 10][f = F_1][y = 20]$ と置いて計算を続ける。

$$
\begin{aligned}
&\equiv \langle eval, (f\ 30), E_2, \texttt{init}\rangle \\
&\to \langle eval, f, E_2, \texttt{push}((apply1, 30, E_2), \texttt{init})\rangle \\
&\to \langle apply, \texttt{push}((apply1, 30, E_2), \texttt{init}), F_1\rangle \\
&\to \langle eval, 30, E_2, \texttt{push}((apply2, F_1), \texttt{init})\rangle \\
&\to \langle apply, \texttt{push}((apply2, F_1), \texttt{init}), 30\rangle \\
&\equiv \langle apply, \texttt{push}((apply2, \texttt{Closure}(x, x + y, \texttt{[]}[y = 10])), \texttt{init}), 30\rangle \\
&\to \langle eval, x + y, \texttt{[]}[y = 10][x = 30], \texttt{init}\rangle \\
&\to^* \langle apply, \texttt{init}, 40\rangle \\
&\to 40
\end{aligned}
$$

ここで $A \equiv B$ というのは、A と B がまったく同じという意味であり、上記では、F_1 や E_2 の定義をほぐすところで使っている。

上記の計算で特筆すべきは、下から 4 行目から 3 行目に行くところ (apply ルールで、スタックのトップに

$$(apply2, \texttt{Closure}(x, x + y, \texttt{[]}[y = 10]))$$

があるとき) である。

このとき、apply2 に対応する計算ルールに従って計算されるが、下から 3 行目の状態を作るとき、クロージャの中に保存しておいた環境 $\texttt{[]}[y = 10]$ が使われるのがポイントである。これは、$\lambda x.x + y$ を計算したとき (関数定義をしたとき) の環境であり、一番最近の環境 E_2 ではない。(E_2 では y の値は 20 だが、クロージャの中に保存しておいた環境では y の値は 10 である。)

この仕組みにより、関数の本体 $x + y$ を計算する時 (下から 3 行目から下から 2 行目への計算) に使われる環境が、「関数定義時」の環境 (に、$x = 30$ を追加したもの) となるため、静的束縛が実現できる。