$$
\begin{array}{lcl}
\mathcal{S} : \texttt{SparcAsm.prog} \to \texttt{string} & & \\
\mathcal{S}((\{D_1, \ldots, D_n\}, E)) & = & \texttt{.section ".text"} \\
 & & \mathcal{S}(D_1) \\
 & & \ldots \\
 & & \mathcal{S}(D_n) \\
 & & \texttt{.global min\_caml\_start} \\
 & & \texttt{min\_caml\_start:} \\
 & & \texttt{save \%sp, -112, \%sp} \\
 & & \mathcal{S}(E, \texttt{\%g0}) \\
 & & \texttt{ret} \\
 & & \texttt{restore} \\
 & & \\
\mathcal{S} : \texttt{SparcAsm.fundef} \to \texttt{string} & & \\
\mathcal{S}(\texttt{L}_x(y_1, \ldots, y_n) = E) & = & x\texttt{:} \\
 & & \mathcal{S}(E, \texttt{R}_0) \\
 & & \texttt{retl} \\
 & & \texttt{nop} \\
 & & \\
\mathcal{S} : \texttt{SparcAsm.t} \times \texttt{Id.t} \to \texttt{string} & & \\
\mathcal{S}((x \leftarrow e; E), z_{\texttt{dest}}) & = & \mathcal{S}(e, x); \mathcal{S}(E, z_{\texttt{dest}}) \\
\mathcal{S}(e, z_{\texttt{dest}}) & = & \mathcal{S}(e, z_{\texttt{dest}})
\end{array}
$$

図 1: 単純なアセンブリ生成 $\mathcal{S}(P)$, $\mathcal{S}(D)$ および $\mathcal{S}(E, z_{\texttt{dest}})$

$$
\begin{array}{lcl}
\multicolumn{3}{l}{\mathcal{S} : \texttt{SparcAsm.exp} \times \texttt{Id.t} \to \texttt{string}} \\
\mathcal{S}(c, z_{\texttt{dest}}) & = & \texttt{set}\ c, z_{\texttt{dest}} \\
\mathcal{S}(\texttt{L}_x, z_{\texttt{dest}}) & = & \texttt{set}\ \texttt{L}_x, z_{\texttt{dest}} \\
\mathcal{S}(op(x_1, \ldots, x_n), z_{\texttt{dest}}) & = & op\ x_1, \ldots, x_n, z_{\texttt{dest}} \\
\mathcal{S}(\texttt{if}\ x = y\ \texttt{then}\ E_1\ \texttt{else}\ E_2, z_{\texttt{dest}}) & = & \texttt{cmp}\ x, y \\
 & & \texttt{bne}\ b_1 \\
 & & \texttt{nop} \\
 & & \mathcal{S}(E_1, z_{\texttt{dest}}) \\
 & & \texttt{b}\ b_2 \\
 & & \texttt{nop} \\
 & & b_1\texttt{:} \\
 & & \mathcal{S}(E_2, z_{\texttt{dest}}) \\
 & & b_2\texttt{:} \\
\mathcal{S}(\texttt{if}\ x \le y\ \texttt{then}\ E_1\ \texttt{else}\ E_2, z_{\texttt{dest}}) & = & \text{同様} \\
\mathcal{S}(x, z_{\texttt{dest}}) & = & \texttt{mov}\ x, z_{\texttt{dest}} \\
\mathcal{S}(apply\_closure(x, y_1, \ldots, y_n), z_{\texttt{dest}}) & = & shuffle((x, y_1, \ldots, y_n), (\texttt{R}_0, \texttt{R}_1, \ldots, \texttt{R}_n)) \\
 & & \texttt{st}\ \texttt{R}_{\texttt{ra}}, [\texttt{R}_{\texttt{st}} + 4\#\varepsilon] \\
 & & \texttt{ld}\ [\texttt{R}_0], \texttt{R}_{n+1} \\
 & & \texttt{call}\ \texttt{R}_{n+1} \\
 & & \texttt{add}\ \texttt{R}_{\texttt{st}}, 4(\#\varepsilon + 1), \texttt{R}_{\texttt{st}}\ \texttt{!}\ \textit{delay slot} \\
 & & \texttt{sub}\ \texttt{R}_{\texttt{st}}, 4(\#\varepsilon + 1), \texttt{R}_{\texttt{st}} \\
 & & \texttt{ld}\ [\texttt{R}_{\texttt{st}} + 4\#\varepsilon], \texttt{R}_{\texttt{ra}} \\
 & & \texttt{mov}\ \texttt{R}_0, z_{\texttt{dest}} \\
\mathcal{S}(apply\_direct(\texttt{L}_x, y_1, \ldots, y_n), z_{\texttt{dest}}) & = & shuffle((y_1, \ldots, y_n), (\texttt{R}_1, \ldots, \texttt{R}_n)) \\
 & & \texttt{st}\ \texttt{R}_{\texttt{ra}}, [\texttt{R}_{\texttt{st}} + 4\#\varepsilon] \\
 & & \texttt{call}\ x \\
 & & \texttt{add}\ \texttt{R}_{\texttt{st}}, 4(\#\varepsilon + 1), \texttt{R}_{\texttt{st}}\ \texttt{!}\ \textit{delay slot} \\
 & & \texttt{sub}\ \texttt{R}_{\texttt{st}}, 4(\#\varepsilon + 1), \texttt{R}_{\texttt{st}} \\
 & & \texttt{ld}\ [\texttt{R}_{\texttt{st}} + 4\#\varepsilon], \texttt{R}_{\texttt{ra}} \\
 & & \texttt{mov}\ \texttt{R}_0, z_{\texttt{dest}} \\
\mathcal{S}(x.(y), z_{\texttt{dest}}) & = & \texttt{ld}\ [x + y], z_{\texttt{dest}} \\
\mathcal{S}(x.(y) \leftarrow z, z_{\texttt{dest}}) & = & \texttt{st}\ z, [x + y] \\
\mathcal{S}(\texttt{save}(x, y), z_{\texttt{dest}}) & = & \text{もし}\ y \notin dom(\varepsilon)\ \text{なら}\ \varepsilon \leftarrow (\varepsilon, y \mapsto 4\#\varepsilon)\ \text{として} \\
 & & \texttt{st}\ x, [\texttt{R}_{\texttt{st}} + \varepsilon(y)] \\
\mathcal{S}(\texttt{restore}(y), z_{\texttt{dest}}) & = & \texttt{ld}\ [\texttt{R}_{\texttt{st}} + \varepsilon(y)], z_{\texttt{dest}}
\end{array}
$$

図 2: 単純なアセンブリ生成 $\mathcal{S}(e, z_{\texttt{dest}})$。$\varepsilon$ はスタック位置を記憶するグローバル変数。$\#\varepsilon$ は $\varepsilon$ の要素の個数。$shuffle((x_1, \ldots, x_n), (r_1, \ldots, r_n))$ は $x_1$, …, $x_n$ を $r_1$, …, $r_n$ に適切な順序で移動する命令。

$$
\begin{array}{lcll}
\mathcal{S} : \texttt{S.t} \to \texttt{SparcAsm.t} \times \texttt{Id.t} \to \texttt{S.t} \times \texttt{string} & & & \\
\mathcal{S}_s((x \leftarrow e; E), z_{\texttt{dest}}) & = & \mathcal{S}_s(e, x) = (s', S), & \\
 & & \mathcal{S}_{s'}(E, z_{\texttt{dest}}) = (s'', S') \text{ として} & \\
 & & (s'', SS') & \\
\mathcal{S}_s(e, z_{\texttt{dest}}) & = & \mathcal{S}_s(e, z_{\texttt{dest}}) & \\
 & & & \\
\mathcal{S} : \texttt{S.t} \to \texttt{SparcAsm.exp} \times \texttt{Id.t} \to \texttt{S.t} \times \texttt{string} & & & \\
\mathcal{S}_s(\texttt{if } x = y \texttt{ then } E_1 \texttt{ else } E_2, z_{\texttt{dest}}) & = & \mathcal{S}_s(E_1, z_{\texttt{dest}}) = (s_1, S_1), & \\
 & & \mathcal{S}_s(E_2, z_{\texttt{dest}}) = (s_2, S_2) \text{ として} & \\
 & & (s_1 \cap s_2, & \\
 & & \;\texttt{cmp } x, y & \\
 & & \;\texttt{bne } b_1 & \\
 & & \;\texttt{nop} & \\
 & & \;S_1 & \\
 & & \;\texttt{b } b_2 & \\
 & & \;\texttt{nop} & \\
 & & \;b_1\texttt{:} & \\
 & & \;S_2 & \\
 & & \;b_2\texttt{:}) & \\
\mathcal{S}_s(\texttt{if } x \le y \texttt{ then } E_1 \texttt{ else } E_2, z_{\texttt{dest}}) & = & \text{同様} & \\
\mathcal{S}_s(\texttt{save}(x, y), z_{\texttt{dest}}) & = & (s, \texttt{nop}) & y \in s \text{ の場合} \\
\mathcal{S}_s(\texttt{save}(x, y), z_{\texttt{dest}}) & = & \text{もし } y \notin \mathit{dom}(\varepsilon) \text{ なら } \varepsilon \leftarrow (\varepsilon, y \mapsto 4\#\varepsilon) \text{ として} & \\
 & & (s \cup \{y\}, \texttt{st } x, [\texttt{R}_{\texttt{st}} + \varepsilon(y)]) & y \notin s \text{ の場合} \\
\mathcal{S}_s(e, z_{\texttt{dest}}) & = & (s, \text{以前と同様}) & \text{上述以外の場合}
\end{array}
$$

図 3: 無駄な save を省略するアセンブリ生成 $\mathcal{S}_s(E, z_{\texttt{dest}})$ および $\mathcal{S}_s(e, z_{\texttt{dest}})$。$s$ はすでに save された変数の名前の集合。以前の $\mathcal{S}(E, z_{\texttt{dest}})$ は $\mathcal{S}_\emptyset(E, z_{\texttt{dest}}) = (s, S)$ として $S$ の略記とする。

$$
\begin{array}{lcl}
\mathcal{S} : \texttt{SparcAsm.fundef} \to \texttt{string} & & \\
\mathcal{S}(\texttt{L}_x(y_1, \ldots, y_n) = E) & = & \mathcal{S}_\emptyset(E, \texttt{tail}) = (s, S) \text{ として} \\
 & & x\texttt{:} \\
 & & S \\
 & & \\
\mathcal{S} : \texttt{S.t} \to \texttt{SparcAsm.exp} \times \texttt{Id.t} \to \texttt{S.t} \times \texttt{string} & & \\
\mathcal{S}_s(\texttt{if } x = y \texttt{ then } E_1 \texttt{ else } E_2, \texttt{tail}) & = & \mathcal{S}_s(E_1, \texttt{tail}) = (s_1, S_1), \\
 & & \mathcal{S}_s(E_2, \texttt{tail}) = (s_2, S_2) \text{ として} \\
 & & (\emptyset, \\
 & & \;\texttt{cmp } x, y \\
 & & \;\texttt{bne } b \\
 & & \;\texttt{nop} \\
 & & \;S_1 \\
 & & \;b\texttt{:} \\
 & & \;S_2) \\
\mathcal{S}_s(\texttt{if } x \leq y \texttt{ then } E_1 \texttt{ else } E_2, \texttt{tail}) & = & \text{同様} \\
\mathcal{S}_s(\mathit{apply\_closure}(x, y_1, \ldots, y_n), \texttt{tail}) & = & (\emptyset, \\
 & & \;\mathit{shuffle}((x, y_1, \ldots, y_n), (\texttt{R}_0, \texttt{R}_1, \ldots, \texttt{R}_n)) \\
 & & \;\texttt{ld } [\texttt{R}_0], \texttt{R}_{n+1} \\
 & & \;\texttt{jmp } \texttt{R}_{n+1} \\
 & & \;\texttt{nop}) \\
\mathcal{S}_s(\mathit{apply\_direct}(\texttt{L}_x, y_1, \ldots, y_n), \texttt{tail}) & = & (\emptyset, \\
 & & \;\mathit{shuffle}((y_1, \ldots, y_n), (\texttt{R}_1, \ldots, \texttt{R}_n)) \\
 & & \;\texttt{b } x \\
 & & \;\texttt{nop}) \\
\mathcal{S}_s(e, \texttt{tail}) & = & \mathcal{S}_s(e, \texttt{R}_0) = (s', S) \text{ として} \\
 & & (\emptyset, \\
 & & \;S \\
 & & \;\texttt{retl} \\
 & & \;\texttt{nop}) \qquad\qquad \text{上述以外の場合}
\end{array}
$$

図 4: 末尾呼び出し最適化をするアセンブリ生成 $\mathcal{S}_s(D)$ および $\mathcal{S}_s(e, z_{\texttt{dest}})$。$z_{\texttt{dest}} = \texttt{tail}$ の場合が末尾。

$$
\begin{array}{l}
e \ ::= \\
\quad c \\
\quad op(x_1, \ldots, x_n) \\
\quad \texttt{if}\ x = y\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \\
\quad \texttt{if}\ x \le y\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \\
\quad \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \\
\quad x \\
\quad \texttt{let rec}\ x\ y_1\ \ldots\ y_n = e_1\ \texttt{in}\ e_2 \\
\quad x\ y_1\ \ldots\ y_n \\
\quad (x_1, \ldots, x_n) \\
\quad \texttt{let}\ (x_1, \ldots, x_n) = y\ \texttt{in}\ e \\
\quad x.(y) \\
\quad x.(y) \leftarrow z
\end{array}
$$

図 5: MinCaml の K 正規形（外部配列・外部関数適用は省略）

$\mathcal{C} : \texttt{Id.t} \to \texttt{KNormal.t} \to \texttt{KNormal.t}$

$$
\begin{array}{lcl}
\mathcal{C}_k(\texttt{if}\ x \le y\ \texttt{then}\ e_1\ \texttt{else}\ e_2) & = & \texttt{if}\ x \le y\ \texttt{then}\ \mathcal{C}_k(e_1)\ \texttt{else}\ \mathcal{C}_k(e_2) \\
\mathcal{C}_k(\texttt{if}\ x = y\ \texttt{then}\ e_1\ \texttt{else}\ e_2) & = & \texttt{if}\ x = y\ \texttt{then}\ \mathcal{C}_k(e_1)\ \texttt{else}\ \mathcal{C}_k(e_2) \\
\mathcal{C}_k(\texttt{let}\ (x_1, \ldots, x_n) = y\ \texttt{in}\ e) & = & \texttt{let}\ (x_1, \ldots, x_n) = y\ \texttt{in}\ \mathcal{C}_k(e) \\
\\
\mathcal{C}_k(\texttt{let}\ x = (\texttt{if}\ y \le z\ \texttt{then}\ e_1\ \texttt{else}\ e_2)\ \texttt{in}\ e_3) & = & \texttt{let rec}\ k'\ x = \mathcal{C}_k(e_3)\ \texttt{in} \\
 & & \texttt{if}\ y \le z\ \texttt{then}\ \mathcal{C}_{k'}(e_1)\ \texttt{else}\ \mathcal{C}_{k'}(e_2) \quad (k'\ \text{は fresh}) \\
\mathcal{C}_k(\texttt{let}\ x = (\texttt{if}\ y = z\ \texttt{then}\ e_1\ \texttt{else}\ e_2)\ \texttt{in}\ e_3) & = & \texttt{let rec}\ k'\ x = \mathcal{C}_k(e_3)\ \texttt{in} \\
 & & \texttt{if}\ y = z\ \texttt{then}\ \mathcal{C}_{k'}(e_1)\ \texttt{else}\ \mathcal{C}_{k'}(e_2) \quad (k'\ \text{は fresh}) \\
\mathcal{C}_k(\texttt{let}\ x = y\ z_1\ \ldots\ z_n\ \texttt{in}\ e) & = & \texttt{let rec}\ k'\ x = \mathcal{C}_k(e)\ \texttt{in}\ y\ k'\ z_1\ \ldots\ z_n \\
 & & \hfill (k'\ \text{は fresh}) \\
\mathcal{C}_k(\texttt{let}\ x = e_1\ \texttt{in}\ e_2) & = & \texttt{let}\ x = e_1\ \texttt{in}\ \mathcal{C}_k(e_2) \quad \text{（上述以外の場合）} \\
\\
\mathcal{C}_k(\texttt{let rec}\ f\ x_1\ \ldots\ x_n = e_1\ \texttt{in}\ e_2) & = & \texttt{let rec}\ f\ c\ x_1\ \ldots\ x_n = \mathcal{C}_c(e_1)\ \texttt{in}\ \mathcal{C}_k(e_2) \\
 & & \hfill (c\ \text{は fresh}) \\
\mathcal{C}_k(x\ y_1\ \ldots\ y_n) & = & x\ k\ y_1\ \ldots\ y_n \\
\\
\mathcal{C}_k(e) & = & \texttt{let}\ x = e\ \texttt{in}\ k\ x \quad \text{（上述以外の場合。}x\ \text{は fresh）}
\end{array}
$$

図 6: [参考] $\alpha$ 変換および A 正規化の完了した K 正規形に対する CPS 変換 $\mathcal{C}_k(e)$。ただし $k$ は $e$ の継続。継続がないときは、$\texttt{let rec}\ k\ x = x\ \texttt{in}\ \mathcal{C}_k(e)$ のように恒等関数とする（メインルーチンなど）