

Terms

(Stolen from the Incremental Flattening paper :))

$$\begin{aligned}
 bop &::= + \mid - \mid * \mid / \mid \mathbf{i} \mid \dots \\
 op &::= \mathbf{transpose} \mid \mathbf{rearrange} (d, \dots, d) \mid \mathbf{replicate} \\
 soac &::= \mathbf{map} \mid \mathbf{reduce} \mid \mathbf{scan} \mid \mathbf{redomap} \mid \mathbf{scanomap} \\
 e &::= x \mid d \mid b \mid (e, \dots, e) \mid e[e] \mid e \mathbf{bop} e \mid op\ e \dots e \mid \mathbf{0} \\
 &\quad \mid \mathbf{loop} \ x_1 \dots x_n = e \ \mathbf{for} \ y < e \ \mathbf{do} \ e \\
 &\quad \mid \mathbf{loop} \ x_1 \dots x_n = e \ \mathbf{for} \ y = e \ \mathbf{to} \ 0 \ \mathbf{do} \ e \\
 &\quad \mid \mathbf{let} \ x_1 \dots x_n = e \ \mathbf{in} \ e \mid \mathbf{if} \ e \ \mathbf{then} \ e \ \mathbf{else} \ e \\
 &\quad \mid soac \ f \ e \dots e \\
 f &::= \lambda x_1 \dots x_n \rightarrow e \mid soac \ f \ e \dots e \mid e \mathbf{bop} \mid bop\ e
 \end{aligned}$$

The $\mathbf{0}$ expression is an array of zeros of arbitrary (and polymorphic!) shape.

Reverse-mode Rules

We define *tape maps* ($\parallel \Omega$) and *adjoint contexts* ($\Lambda \vdash$) as

$$\begin{aligned}
 \Omega &::= \varepsilon \mid \Omega, (x \mapsto x_s) \\
 \Lambda &::= \varepsilon \mid \Lambda, (x \mapsto \hat{x})
 \end{aligned}$$

The union of two maps prefers the right map in the instance of key conflicts:

$$(\Lambda_1 \cup \Lambda_2)[x] = \begin{cases} \Lambda_2[x] & \text{if } (x \mapsto \hat{x}) \in \Lambda_2 \\ \Lambda_1[x] & \text{otherwise} \end{cases}$$

Mappings of lists of variables is sugar for a list of mappings:

$$x_1 x_2 \dots x_n \mapsto \hat{x}_1 \hat{x}_2 \dots \hat{x}_n = \epsilon, (x_1 \mapsto \hat{x}_1), (x_2 \mapsto \hat{x}_2), \dots, (x_n \mapsto \hat{x}_n)$$

dom returns all keys of a map, i.e.,

$$\mathbf{dom} (\epsilon, (x_1 \mapsto \hat{x}_1), (x_2 \mapsto \hat{x}_2)) = \{x_1, x_2\}$$

im returns all elements:

$$\mathbf{im} (\epsilon, (x_1 \mapsto \hat{x}_1), (x_2 \mapsto \hat{x}_2)) = \{\hat{x}_1, \hat{x}_2\}$$

We're sloppy and overload the notation somewhat, so expressions like

$$\mathbf{let} \ \mathbf{dom} (\epsilon, (x_1 \mapsto \hat{x}_1), (x_2 \mapsto \hat{x}_2)) = e_1 \ \mathbf{in} \ e_2$$

are to be understood as

$$\mathbf{let} \ x_1 \ x_2 = e_1 \ \mathbf{in} \ e_2$$

or

$$\mathbf{let} \ (x_1, x_2) = e_1 \ \mathbf{in} \ e_2$$

depending on the context.

Forward pass (\Rightarrow_F)

$ \frac{e = \mathbf{loop} \ \bar{x} = e_0 \ \mathbf{for} \ y < e_n \ \mathbf{do} \ e_{body} \quad x_{s_0} \text{ fresh} \quad x_{s_0} = \mathbf{replicate} \ e_n \ \mathbf{0}}{e \Rightarrow_F \mathbf{loop} \ (\bar{x}, x_s) = (e_0, x_{s_0}) \ \mathbf{for} \ y < e_n \ \mathbf{do} \ (e_{body}, x_s[y] = \bar{x}) \parallel (\bar{x} \mapsto x_s)} \text{FWDLOOP} $
--

Reverse pass (\Leftarrow)

$$\begin{array}{c}
e_{body} = \mathbf{let} \ \overline{rs} = e'_{body} \ \mathbf{in} \ \overline{rs} \quad e_{loop} = \mathbf{let} \ \overline{lres} = \mathbf{loop} \ \overline{x} = e_0 \ \mathbf{for} \ y < e_n \ \mathbf{do} \ e_{body} \ \mathbf{in} \ \overline{lres} \\
e_{loop} \Rightarrow_F e'_{loop} \parallel \Omega \quad \overline{fv} = FV(e_{body}) \setminus \overline{x} \quad \overline{\hat{x}}, \overline{\hat{fv}}, \overline{\hat{fv}'}, \overline{\hat{fv}''}, \overline{rs}, \overline{rs'} \text{ fresh} \\
\overline{reset} = \mathbf{map} (\lambda _ . 0) \ \overline{\hat{x}} \quad \Lambda'_1 = \Lambda_1, \ \overline{x} \mapsto \overline{\hat{x}}, \ \overline{fv} \mapsto \overline{\hat{fv}}, \ \overline{rs} \mapsto \overline{rs} \\
\hat{e}_{body} = \mathbf{let} \ \overline{\hat{z}} = \hat{e}'_{body} \ \mathbf{in} \ \overline{\hat{z}} \quad (\Lambda'_1 \vdash e_{body}) \Leftarrow (\Lambda_2 \vdash \hat{e}_{body}) \\
\Lambda_{2,fv} = \{v \mapsto \hat{v} \mid v \in \overline{fv}, (v \mapsto \hat{v}) \in \Lambda_2 \setminus \Lambda'_1\} \quad \Lambda_{2,rs} = \{r \mapsto \hat{r} \mid r \in \overline{rs}, (r \mapsto \hat{r}) \in \Lambda_2\} \\
\hat{e}''_{body} = \mathbf{let} \ \overline{rs} = \Omega[y] \ \mathbf{in} \ (\mathbf{let} \ \overline{\hat{z}'} = \hat{e}'_{body} \ \mathbf{in} \ (\overline{reset}, im(\Lambda_{2,rs}), im(\Lambda_{2,fv}))) \\
\widehat{init} = (\overline{reset}, \Lambda_1[\overline{lres}], \Lambda_1[dom(\Lambda_{2,fv})]) \\
\hat{e}_{loop} = \mathbf{loop} \ (\overline{\hat{x}}, \overline{\hat{rs}}, \overline{\hat{fv}}) = \widehat{init} \ \mathbf{for} \ y = e_n - 1 \ \mathbf{to} \ 0 \ \mathbf{do} \ \hat{e}''_{body} \\
\Lambda_3 = \Lambda_1 \cup \left(dom(\Lambda_{2,fv}) \mapsto \overline{\hat{fv}''} \right) \\
\hline
\Lambda_1 \vdash e_{loop} \Leftarrow \left(\Lambda_3 \vdash \mathbf{let} \ (_, \overline{\hat{rs}'}, \overline{\hat{fv}'}) = \hat{e}_{loop} \ \mathbf{in} \ (\mathbf{let} \ \overline{\hat{fv}''} = (\mathbf{map} \ (+) \ \overline{\hat{fv}'} \ \overline{\hat{rs}'}) \ \mathbf{in} \ \overline{\hat{fv}''}) \right) \quad \text{REVLOOP} \\
\\
\Lambda \vdash e_t \Leftarrow \Lambda_t \vdash \mathbf{let} \ \overline{\hat{fv}_t} = \hat{e}_t \ \mathbf{in} \ \overline{\hat{fv}_t} \\
\Lambda \vdash e_f \Leftarrow \Lambda_f \vdash \mathbf{let} \ \overline{\hat{fv}_f} = \hat{e}_f \ \mathbf{in} \ \overline{\hat{fv}_f} \quad \Lambda_{\Delta_t} = \Lambda \setminus \Lambda_t \quad \Lambda_{\Delta_f} = \Lambda \setminus \Lambda_f \\
\overline{\hat{fv}} \text{ fresh} \quad \hat{e}'_t = \mathbf{let} \ \overline{\hat{fv}} = \text{sort}(\overline{\hat{fv}_t} \ ++ \ im(\Lambda_{\Delta_f} - \Lambda_{\Delta_t})) \ \mathbf{in} \ (\mathbf{let} \ \overline{\hat{fv}_t} = \hat{e}_t \ \mathbf{in} \ \overline{\hat{fv}_t}) \\
\hat{e}'_f = \mathbf{let} \ \overline{\hat{fv}} = \text{sort}(\overline{\hat{fv}_f} \ ++ \ im(\Lambda_{\Delta_t} - \Lambda_{\Delta_f})) \ \mathbf{in} \ (\mathbf{let} \ \overline{\hat{fv}_f} = \hat{e}_f \ \mathbf{in} \ \overline{\hat{fv}_f}) \\
\Lambda' = \Lambda, \ (dom \Lambda_{\Delta_t} \cup (\Lambda_{\Delta_f}) \mapsto \overline{\hat{fv}}) \\
\hline
\Lambda \vdash \mathbf{if} \ e_p \ \mathbf{then} \ e_t \ \mathbf{else} \ e_f \Leftarrow \Lambda' \vdash \mathbf{if} \ e_p \ \mathbf{then} \ \hat{e}'_t \ \mathbf{else} \ \hat{e}'_f \quad \text{REVIF}
\end{array}$$