

Reduce

Alex Aiken

April 17, 1989

1 Reduce

This document makes two points: first, we need a reduce construct to effectively optimize some programs, and second, Ed's new formulation of Bind is more powerful than my original formulation.

Consider the program `average`, which divides the sum of a sequence of numbers by the length of the sequence.

$$\text{def } \text{average} \leftarrow (\text{isnum} \rightarrow \text{seqof:isnum}) \equiv + \div \text{len}$$

With the methods now available, the body of this function could be optimized as follows.

$$\begin{aligned} & + \div \text{len} \\ \equiv & \div \circ [+ , \text{len}] \\ \equiv & \div \circ [+ , + \circ \alpha: \langle \sim 1 \rangle] \\ \equiv & \div \circ [+ , + \circ \text{for:} \langle \sim 1, \text{len} \rangle] \\ \equiv & \div \circ [/l: + , /l: + \circ \text{for:} \langle \sim 1, \text{len} \rangle] \end{aligned}$$

Note that there is an implicit assumption that list processing is going to be used in the generated code. If the argument to the function were an array, then the length would be known and there would be no need to expand `len` into a loop. Also note that `len` is optimized in terms of itself—the expression for `len` contains `len`. This is OK because the `len` in the bound function of the `for` construct should be recognized by the code generator as saying “traverse this list”.

Currently, the code generator is supposed to recognize the form

$$/l: g \circ \text{for:} \langle E, f \rangle$$

and turn out a reduction loop. However, in this example, the code generator would produce two reduction loops over the same list. This function should be optimized further so that the resulting code traverses the list only once. To do this, a new combining form “reduce” is needed. Reduce is defined in Figure 1; for comparison, a definition of `for` is also given. Some identities involving `for` and `reduce`:

```

def reduce([E.isfunc, first.isfunc, lim.isfunc]) ← isval ≡
  isnull →
    [];
    /l: E ◦ al ◦ [first, distl ◦ [id, tl ◦ intsto ◦ lim]]

/* The following selectors are used inside of reduce */
exdef  [[Tot., Cur.] ≡ id

def for([E.isfunc, lim.isfunc]) ← isval ≡
  α: E ◦ distl ◦ [id, intsto ◦ lim]

/* The following selectors are used inside of reduce and for */
exdef  [[Arg., Ind.] ≡ id

```

Figure 1: Reduce and for.

$$\begin{aligned}
\text{Safe: } (/l: g) &\equiv \text{Safe: } (\text{reduce: } \langle g \circ [\text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], s1, \text{len} \rangle) \\
\text{Safe: } (\text{for: } \langle E, f \rangle) &\equiv \text{Safe: } (\text{reduce: } \langle \text{ar} \circ [\text{Tot}, E \circ \text{Cur}], E \circ [\text{id}, \sim 1], f \rangle) \\
\text{Safe: } (/l: g \circ \text{for: } \langle E, f \rangle) &\equiv \text{Safe: } (\text{reduce: } \langle g \circ [\text{Tot}, E \circ \text{Cur}], E \circ [\text{id}, \sim 1], f \rangle) \\
\text{Safe: } ([\dots, \text{reduce: } \langle E_i, g_i, f \rangle, \dots]) &\equiv \text{Safe: } (\text{reduce: } \langle [\dots, E_i \circ [s1 \circ \text{Tot}, \text{Cur}], \dots], [\dots, g_i, \dots], f \rangle)
\end{aligned}$$

Using reduce, average can be optimized into a single loop.

$$\begin{aligned}
&+ \div \text{len} & (1) \\
\equiv &\div \circ [+ , \text{len}] & (2) \\
\equiv &\div \circ [/l: + , /l: + \circ \alpha: (\sim 1)] & (3) \\
\equiv &\div \circ [/l: + , /l: + \circ \text{for: } \langle \sim 1, \text{length} \rangle] & (4) \\
\equiv &\div \circ [\text{reduce: } \langle + \circ [\text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], s1, \text{len} \rangle, & (5) \\
&\quad \text{reduce: } \langle + \circ [\text{Tot}, \sim 1 \circ \text{Cur}], \sim 1 \circ [\text{id}, \sim 1], \text{len} \rangle \\
\equiv &\div \circ [\text{reduce: } \langle + \circ [\text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], s1, \text{len} \rangle, \text{reduce: } \langle + \circ [\text{Tot}, \sim 1], \sim 1, \text{len} \rangle] & (6) \\
\equiv &\div \circ \text{reduce: } \langle [+ \circ [\text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}] \circ [s1 \circ \text{Tot}, \text{Cur}], & (7) \\
&\quad + \circ [\text{Tot}, \sim 1] \circ [s2 \circ \text{Tot}, \text{Cur}], [s1, \sim 1], \text{len} \rangle \\
\equiv &\div \circ \text{reduce: } \langle [+ \circ [\text{Tot} \circ [s1 \circ \text{Tot}, \text{Cur}], \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur} \circ [s1 \circ \text{Tot}, \text{Cur}], & (8) \\
&\quad + \circ [\text{Tot}, \sim 1] \circ [s2 \circ \text{Tot}, \text{Cur}], [s1, \sim 1], \text{len} \rangle \\
\equiv &\div \circ \text{reduce: } \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [\text{Tot}, \sim 1] \circ [s2 \circ \text{Tot}, \text{Cur}], & (9) \\
&\quad [s1, \sim 1], \text{len} \rangle \\
\equiv &\div \circ \text{reduce: } \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [\text{Tot} \circ [s2 \circ \text{Tot}, \text{Cur}], & (10) \\
&\quad \sim 1 \circ [s2 \circ \text{Tot}, \text{Cur}]], [s1, \sim 1], \text{len} \rangle \\
\equiv &\div \circ \text{reduce: } \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [s2 \circ \text{Tot}, \sim 1], [s1, \sim 1], \text{len} \rangle & (11)
\end{aligned}$$

As another example, consider the construction of two for constructs with the same bound function. Using reduce, this can be transformed into a single loop. The analog in compilers for imperative languages is “loop jamming”.

$$[\text{for: } \langle E_1, f \rangle, \text{for: } \langle E_2, f \rangle] \quad (12)$$

$$\equiv [\text{for: } \langle E_1, f \rangle, \text{for: } \langle E_2, f \rangle] \quad (13)$$

$$\equiv [\text{reduce: } \langle \text{ar} \circ [\text{Tot}, E_1 \circ \text{Cur}], E_1 \circ [\text{id}, \sim 1], f \rangle, \quad (14)$$

$$\text{reduce: } \langle \text{ar} \circ [\text{Tot}, E_2 \circ \text{Cur}], E_2 \circ [\text{id}, \sim 1], f \rangle]$$

$$\equiv \text{reduce: } \langle [\text{ar} \circ [\text{Tot}, E_1 \circ \text{Cur}] \circ [s1 \circ \text{Tot}, \text{Cur}], \text{ar} \circ [\text{Tot}, E_2 \circ \text{Cur}] \circ [s2 \circ \text{Tot}, \text{Cur}], \quad (15)$$

$$[E_1 \circ [\text{id}, \sim 1], E_2 \circ [\text{id}, \sim 1]], f \rangle$$

$$\equiv \text{reduce: } \langle [\text{ar} \circ [s1 \circ \text{Tot}, E_1 \circ \text{Cur}], \text{ar} \circ [s2 \circ \text{Tot}, E_2 \circ \text{Cur}]], [E_1, E_2] \circ [\text{id}, \sim 1], f \rangle \quad (16)$$

2 Reduce and Bind

Consider the last line of the optimization of average:

$$\div \circ \text{reduce: } \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [s2 \circ \text{Tot}, \sim 1], [s1, \sim 1], \text{len} \rangle$$

This loop can be further improved. On each iteration, a pair is created and then immediately used on the following iteration. Using Ed’s formulation of bind, this inefficiency can be eliminated. The following abbreviations are used:

def $\sigma \equiv \text{fetch}$

def $\pi \equiv \text{bind}$

Some laws involving fetch, bind, and reduce:

$$[f_1, \dots, f_n] \circ \sigma: x \equiv [\sigma: y_1, \dots, \sigma: y_n] \circ \pi: \langle y_n, f_n \circ \sigma: x \rangle \circ \dots \circ \pi: \langle y_1, f_1 \circ \sigma: x \rangle$$

where y_i does not appear in $f_1 \dots f_i$

$$\text{reduce: } \langle E, f, g \rangle \circ \sigma: x \equiv \sigma: t \circ \text{reduce: } \langle \pi: \langle t, E \circ [\sigma: t \circ \text{Tot}, [\sigma: x \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle, \quad (15)$$

$$\pi: \langle t, f \circ \sigma: x \rangle, g \circ \sigma: x \rangle$$

where t does not appear in E, f

$$\pi: \langle x, f \circ g \rangle \equiv \pi: \langle x, f \rangle \circ g$$

$$\text{reduce: } \langle \pi: \langle x, h \rangle \circ E, \pi: \langle x, h \rangle \circ f, g \rangle \equiv \pi: \langle x, h \rangle \circ \text{reduce: } \langle E, f, g \rangle$$

if $\sigma: x$ not in E

$$\sigma: x \circ \pi: \langle x, f \rangle \equiv f$$

There is one other law that has an unusual qualification:

$$s1 \circ \sigma: t \equiv \sigma: f_i \text{ in } \text{reduce: } \langle \pi: \langle t, B \rangle \circ E, \pi: \langle t, B \rangle \circ h, g \rangle$$

where $B = [\dots, \sigma: f_i, \dots]$

Using these laws, average can be transformed into another program that looks like machine code. In particular, the only primitive functions are machine primitives, every primitive is applied to arguments in the store (i.e., to a fetch or construction of fetches or references to a loop index), and every construct is either an assignment (a π) or a control construct (a reduce). This optimization cannot be implemented with the original formulation of Bind. For clarity, the name “a” is used for the original argument to the function rather than Ed’s “Arg”.

$$\div \circ \text{reduce}: \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [s2 \circ \text{Tot}, \sim 1]], [s1, \sim 1], \text{len} \rangle (17)$$

$$\equiv \div \circ \text{reduce}: \langle [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [s2 \circ \text{Tot}, \sim 1]], [s1, \sim 1], \text{len} \rangle (18)$$

$$\circ \sigma: a \circ \text{Setup}$$

$$\equiv \div \circ \sigma: t \circ \text{reduce}: \langle \pi: \langle t, [+ \circ [s1 \circ \text{Tot}, \text{sel} \circ [\text{Arg}, \text{Ind}] \circ \text{Cur}], + \circ [s2 \circ \text{Tot}, \sim 1]] \rangle (19)$$

$$\circ [\sigma: t \circ \text{Tot}, [\sigma: a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle,$$

$$\pi: \langle t, [s1, \sim 1] \circ \sigma: a \rangle,$$

$$\text{len} \circ \sigma: a \circ \text{Setup}$$

$$\equiv \div \circ \sigma: t \circ \text{reduce}: \langle \pi: \langle t, [+ \circ [s1 \circ \sigma: t \circ \text{Tot}, \text{sel} \circ [\sigma: a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]], (20)$$

$$+ \circ [s2 \circ \sigma: t \circ \text{Tot}, \sim 1]] \rangle,$$

$$\pi: \langle t, [s1, \sim 1] \circ \sigma: a \rangle,$$

$$\text{len} \circ \sigma: a \circ \text{Setup}$$

$$\equiv \div \circ \sigma: t \circ \text{reduce}: \langle \pi: \langle t, [\sigma: x, \sigma: y] \circ (21)$$

$$\pi: \langle x, + \circ [s1 \circ \sigma: t \circ \text{Tot}, \text{sel} \circ [\sigma: a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle$$

$$\circ \pi: \langle y, + \circ [s2 \circ \sigma: t \circ \text{Tot}, \sim 1]] \rangle,$$

$$\pi: \langle t, [s1, \sim 1] \circ \sigma: a \rangle,$$

$$\text{len} \circ \sigma: a \circ \text{Setup}$$

$$\equiv \div \circ \sigma: t \circ \text{reduce}: \langle \pi: \langle t, [\sigma: x, \sigma: y] \circ (22)$$

$$\pi: \langle x, + \circ [s1 \circ \sigma: t \circ \text{Tot}, \text{sel} \circ [\sigma: a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle$$

$$\circ \pi: \langle y, + \circ [s2 \circ \sigma: t \circ \text{Tot}, \sim 1]] \rangle,$$

$$\pi: \langle t, [\sigma: x, \sigma: y] \circ \pi: \langle x, s1 \circ \sigma: a \rangle \circ \pi: \langle \sigma: y, \sim 1 \rangle \rangle,$$

$$\text{len} \circ \sigma: a \circ \text{Setup}$$

$$\equiv \div \circ \sigma: t \circ \text{reduce}: \langle \pi: \langle t, [\sigma: x, \sigma: y] \circ (23)$$

$$\pi: \langle x, + \circ [s1 \circ \sigma: t \circ \text{Tot}, \text{sel} \circ [\sigma: a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle$$

$$\circ \pi: \langle y, + \circ [s2 \circ \sigma: t \circ \text{Tot}, \sim 1]] \rangle,$$

$$\pi: \langle t, [\sigma: x, \sigma: y] \rangle \circ \pi: \langle x, s1 \circ \sigma: a \rangle \circ \pi: \langle \sigma: y, \sim 1 \rangle,$$

$$\text{len} \circ \sigma: a \circ \text{Setup}$$

$$\begin{aligned}
\equiv \quad & \div \circ \sigma : t \circ \text{reduce} : \langle \pi : \langle t, [\sigma : x, \sigma : y] \rangle \circ \\
& \pi : \langle x, + \circ [\sigma : x \circ \text{Tot}, \text{sel} \circ [\sigma : a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle \\
& \circ \pi : \langle y, + \circ [\sigma : y \circ \text{Tot}, \sim 1] \rangle, \\
& \pi : \langle t, [\sigma : x, \sigma : y] \rangle \circ \pi : \langle x, \text{sl} \circ \sigma : a \rangle \circ \pi : \langle \sigma : y, \sim 1 \rangle, \\
& \text{len} \circ \sigma : a \rangle \circ \text{Setup}
\end{aligned} \tag{24}$$

$$\begin{aligned}
\equiv \quad & \div \circ \sigma : t \circ \pi : \langle t, [\sigma : x, \sigma : y] \rangle \circ \text{reduce} : \langle \\
& \pi : \langle x, + \circ [\sigma : x \circ \text{Tot}, \text{sel} \circ [\sigma : a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle \\
& \circ \pi : \langle y, + \circ [\sigma : y \circ \text{Tot}, \sim 1] \rangle, \\
& \pi : \langle x, \text{sl} \circ \sigma : a \rangle \circ \pi : \langle \sigma : y, \sim 1 \rangle, \\
& \text{len} \circ \sigma : a \rangle \circ \text{Setup}
\end{aligned} \tag{25}$$

$$\begin{aligned}
\equiv \quad & \div \circ [\sigma : x, \sigma : y] \circ \text{reduce} : \langle \\
& \pi : \langle x, + \circ [\sigma : x \circ \text{Tot}, \text{sel} \circ [\sigma : a \circ \text{Tot}, \text{Ind} \circ \text{Cur}]] \rangle \\
& \circ \pi : \langle y, + \circ [\sigma : y \circ \text{Tot}, \sim 1] \rangle, \\
& \pi : \langle x, \text{sl} \circ \sigma : a \rangle \circ \pi : \langle \sigma : y, \sim 1 \rangle, \\
& \text{len} \circ \sigma : a \rangle \circ \text{Setup}
\end{aligned} \tag{26}$$