Project Euler, Problem 1

Find the sum of all natural multiples of 3 or 5 below 1000.

* *
*

Let the raised infix dot, ·, denote function composition. We are heading for a program of the form

$$\mathit{foldr}.f.e \cdot \mathit{unfoldr}.\mathit{step} \quad .$$

The function foldr is defined by

$$\begin{array}{lcl} \mathit{foldr} & : & (a \to b \to b) \to b \to \mathit{List}.a \to b \\ \mathit{foldr}.f.e.[\,] & = & e \\ \mathit{foldr}.f.e.(x : xs) & = & f.x.(\mathit{foldr}.f.e.xs) \end{array}$$

Regarding $\mathit{unfoldr}$, recall the standard type Maybe:

datatype Maybe($a : Set$) ::=
  $\mathit{Nothing}$: Maybe a
  Just : a → Maybe a

The definition of $\mathit{unfoldr}$ is

$$\begin{array}{lcl} \mathit{unfoldr} & : & (b \to \mathit{Maybe}(a \times b)) \to b \to \mathit{List}.a \\ \mathit{unfoldr}.\mathit{step}.x & = & \textbf{case } \mathit{step}.x \textbf{ of} \\ & & \quad \mathit{Just}(y, x') \to y : \mathit{unfoldr}.\mathit{step}.x' \\ & & \quad \mathit{Nothing} \to [\,] \end{array}$$

* *
*

Our algorithm will proceed in four phases:

0. Generate the list $[i : 0 \leq i < 1000]$.
1. Filter out the non-multiples of 3.
2. Filter out the non-multiples of 5.
3. Compute the sum of the elements of the list.

* *
*

Generating the list

We generate our list with

$$\mathit{unfoldr}.\mathit{step}.0 \qquad \text{where } \mathit{step} = \langle \lambda x \to \textbf{if } x = 1000 \textbf{ then } \mathit{Nothing} \textbf{ else } \mathit{Just}(x, x+1) \rangle$$

* *
*

Filtering multiples of 3 and 5

To remove non-multiples of 3 and 5 we use the standard function $filter$ defined by

$$\begin{array}{rcl} filter & : & (a \to Bool) \to List.a \to List.a \\ filter.p & = & foldr.g.[\,] \quad \text{where } g.x = if\ p.x\ then (x:)\ else\ id \end{array}$$

We define

$$\begin{array}{lcl} muliple3or5 & : & Int \to Bool \\ multiple3or5\ n & = & n \bmod 3 \ = \ 0 \quad \vee \quad n \bmod 5 \ = \ 0 \end{array}$$

giving us as our filter

$$filter.multiple3or5 \quad .$$

* *
*

Summing the list

To sum the list we define

$$sum = foldr.(+).0$$

* *
*

Our program is thus

$$sum \cdot filter.multiple3or5 \cdot unfoldr.step.0$$

* *
*

If we postulate that our program notation has non-strict semantics then the list need never exist as a whole. It remains advantageous, however, to fuse the phases into one in order to reduce the amount of heap space turnover.

We will first combine sum and $filter.multiple3or5$. To this end we recall the fusion theorem:

|[Context: f is strict $\wedge$ $f.a = b$ $\wedge$ $f.(g.x.y) = h.x.(f.y)$

$$f \cdot foldr.g.a = foldr.h.b$$

]|

Let us solve the equation

$$h, b \quad :: \quad sum \cdot filter.p = foldr.h.b$$

Looking at the conditions of the fusion theorem, the first is satisfied by having sum be strict, and the second by taking $0 = b$. As for the third, we observe

$$\begin{array}{ll} & sum.(g.x.xs) = h.x.(sum.xs) \\ = & \quad \{ \; g \; \} \\ & \begin{cases} sum.(x : xs) = h.x.(sum.xs) & \text{if } multiple3or5.x \\ sum.xs = h.x.(sum.xs) & \text{if } \neg multiple3or5.x \end{cases} \\ \Leftarrow & \quad \{ \text{ property of } sum\text{; Leibniz } \} \\ & \begin{cases} h.x = (x+) & \text{if } multiple3or5.x \\ h.x = id & \text{if } \neg multiple3or5.x \end{cases} \end{array}$$

* *
*

It remains to fuse $foldr.h.0$ and $unfoldr.step$ by solving the equation

$$f \quad :: \quad f = foldr.h.0 \cdot unfoldr.step$$

We observe

$$\begin{array}{ll} & f.x \\ = & \quad \{ \; f \; \} \\ & foldr.h.0.(unfoldr.step.x) \\ = & \quad \{ \; step.x = Nothing \; \} \\ & foldr.h.0.[] \\ = & \quad \{ \; foldr \; \} \\ & 0 \end{array}$$

and

$$\begin{array}{ll} & f.x \\ = & \quad \{ \; f \; \} \\ & foldr.h.0.(unfoldr.step.x) \\ = & \quad \{ \; step.x = Just.(x, x+1) \; \} \\ & foldr.h.0.(x : unfoldr.step.(x+1)) \\ = & \quad \{ \; foldr; h \; \} \\ & \begin{cases} x + foldr.h.0.(unfoldr.step.(x+1)) & \text{if } multiple3or5.x \\ foldr.h.0.(unfoldr.step.(x+1)) & \text{if } \neg multiple3or5.x \end{cases} \\ = & \quad \{ \; f \; \} \\ & \begin{cases} x + f.(x+1) & \text{if } multiple3or5.x \\ f.(x+1) & \text{if } \neg multiple3or5.x \end{cases} \end{array}$$

*   *
*

Our solution is thus f.0 or 233168.

Eric Macaulay
2008.05.19