Strachey's functional pearl, forty years on

Mike Spivey, July 2006

In a 1966 paper, Christopher Strachey gave a number of examples of programs written in the programming language CPL using a functional style, and among them is a remarkable definition of a Cartesian product function cprod on lists of lists that uses (the function we now know as) foldr instead of recursion. Strachey presents this definition without comment or derivation. This note gives a derivation from a specification in the form of a list comprehension.

The function *cprod* takes a list of lists and computes its Cartesian product:

$$\begin{aligned} &\mathit{cprod}\ [[1,2,3],[4],[5,6]] \\ &\quad = [[1,4,5],[1,4,6],[2,4,5],[2,4,6],[3,4,5],[3,4,6]] \end{aligned}$$

We can specify this function using recursion and list comprehension:

$$\begin{aligned} &\mathit{cprod}\ [\,] = [[\,]] \\ &\mathit{cprod}\ (\mathit{xs} : \mathit{zss}) = [\,x : \mathit{ys} \mid x \leftarrow \mathit{xs}, \mathit{ys} \leftarrow \mathit{cprod}\ \mathit{zss}\,] \end{aligned}$$

Now what we want is a version of *cprod* that uses no recursion, but is written instead in terms of *foldr* and elementary operations on lists.

The main principle to which we shall appeal is this: that $p = \mathit{foldr}\ f\ a$ is the unique solution for p of the equations,

$$\begin{aligned} &p\ [\,] = a \\ &p\ (x : \mathit{xs}) = f\ x\ (p\ \mathit{xs}) \end{aligned}$$

For example, if we take $p\ \mathit{xs} = \mathit{concat}\ (\mathit{map}\ f_0\ \mathit{xs})$, then $p\ [\,] = [\,]$ and $p\ (x : \mathit{xs}) = f_0\ x +\!\!+ p\ \mathit{xs}$, so we can deduce that $p = \mathit{foldr}\ f\ [\,]$, where $f\ x\ \mathit{ys} = f_0\ x +\!\!+ \mathit{ys}$.

It is plain that *cprod* itself can be written as a *foldr*, because $\mathit{cprod}\ (\mathit{xs} : \mathit{zss})$ depends on *zss* only through $\mathit{cprod}\ \mathit{zss}$. Thus:

$$\mathit{cprod} = \mathit{foldr}\ f\ [[\,]]$$

where $f\ \mathit{xs}\ \mathit{yss} = [\,x : \mathit{ys} \mid x \leftarrow \mathit{xs}, \mathit{ys} \leftarrow \mathit{yss}\,]$. It remains to express this function also in terms of *foldr*. The meaning of the two-generator list comprehension for f can be expressed using nested comprehensions and *concat*:

$$\begin{aligned} &f\ \mathit{xs}\ \mathit{yss} = \mathit{concat}\ [\,[\,x : \mathit{ys} \mid \mathit{ys} \leftarrow \mathit{yss}\,] \mid x \leftarrow \mathit{xs}\,] \\ &\quad = \mathit{concat}\ (\mathit{map}\ (g_0\ \mathit{yss})\ \mathit{xs}) \end{aligned}$$

where $g_0\ yss\ x = [\ x : ys \mid ys \leftarrow yss\ ] = map\ (x:)\ yss$. (The nesting of the *ys* comprehension inside the *x* compreshension reflects the fact that *ys* "varies faster than" *x*.) Now we can use the result about *concat* and *map* mentioned above: this gives

$$f\ xs\ yss = foldr\ (g\ yss)\ [\ ]\ xs$$

where $g\ yss\ x\ zss = g_0\ yss\ x +\!\!+ zss = map\ (x:)\ yss +\!\!+ zss$.

In its turn, we look for a definition of *g* in terms of *foldr*. Clearly,

$$g\ [\ ]\ x\ zss = zss,$$
$$g\ (ys : yss)\ x\ zss = map\ (x:)\ (ys : yss) +\!\!+ zss$$
$$= (x : ys) : (map\ (x:)\ yss +\!\!+ zss) = (x : ys) : (g\ yss\ x\ zss),$$

so $g\ yss\ x\ zss = foldr\ (h\ x)\ zss\ yss$ where $h\ x\ ys\ uss = (x : ys) : uss$.

Putting it all together, we obtain

```
cprod = foldr f [[ ]]
  where
    f xs yss = foldr (g yss) [ ] xs
    g yss x zss = foldr (h x) zss yss
    h x ys uss = (x : ys) : uss
```

To smooth the derivation I have made the free variables of each auxiliary function body explicit as extra arguments, but they can be eliminated in favour of nested **where** clauses to obtain essentially the program that Strachey wrote:

```
cprod = foldr f [[ ]]
  where f xs yss = foldr g [ ] xs
    where g x zss = foldr h zss yss
      where h ys uss = (x : ys) : uss
```

Olivier Danvy suggested that Strachey's program could be seen as the earliest functional pearl, and I had already been using it as the first exercise in my course on programming languages at Oxford. This note is an expanded version of the model solution of that problem. Strachey's program had already been recalled to attention in recent times in Mike Gordon's reminiscence in a special issue of the journal Higher-Order and Symbolic Computation *published in 2000 to commemorate the 25th anniversary of Strachey's death.*