Threading Fibonacci & Ackermann

Mayer Goldberg

January 6, 2012

1 The Problem

We would like to define the function $f(a, b, c)$ to thread both Fibonacci and Ackermann, returning 0 or 1, depending on which terminates first:

$$f(a,b,c) = \begin{cases} 0 & \text{If the } a\text{-th Fibonacci number} \\ & \text{takes less steps to evaluate than} \\ & \text{Ackermann of } b \text{ and } c. \\ 1 & \text{Otherwise} \end{cases}$$

The goal is to thread the two computations so that we terminate as soon as *either* Fibonacci or Ackermann terminate, not waiting for *both* to terminate. This is, in fact, a very simple problem, since Fibonacci will terminate sooner than Ackermann for all but a finite set of arguments. We could have just managed with a simple lookup, but this would be spoiling the fun, won't it?

2 CPS + Defunctionalisation

We begin by defunctionalising the CPS versions of both the Fibonacci and Ackermann functions:

$$\begin{cases} \text{Fib}(0,k) &=& R(0,k) \\ \text{Fib}(1,k) &=& R(1,k) \\ \text{Fib}(n,k) &=& \text{Fib}(n-1,\langle 2,n,k\rangle) \end{cases}$$

$$\begin{cases} \text{Ack}(0,b,k) &=& R(b+1,k) \\ \text{Ack}(a,0,k) &=& \text{Ack}(a-1,1,k) \\ \text{Ack}(a,b,k) &=& \text{Ack}(a,b-1,\langle 4,a,k\rangle) \end{cases}$$

$$\begin{cases} R(x,\langle 0\rangle) &=& x \\ R(x,\langle 1\rangle) &=& x \\ R(x,\langle 2,n,k\rangle) &=& \text{Fib}(n-2,\langle 3,x,k\rangle) \\ R(x,\langle 3,y,k\rangle) &=& R(x+y,k) \\ R(x,\langle 4,a,k\rangle) &=& \text{Ack}(a-1,x,k) \end{cases}$$

We intentionally have two *initial* continuations, specified by $\langle 0\rangle, \langle 1\rangle$. The idea is to have Fib() and Ack() exit from two different places. This will simplify the identification of computation later on.

3 Identifying The Computation

The point is, though, we're totally uninterested in the actual answer of either the Fibonacci or the Ackermann functions. We just want to know which one terminated first. We accomplish this by changing the behaviour of the initial continuations, to return the continuation *tag*, rather than the result of the the original function:

$$\left\{\begin{array}{rcl} \mathrm{Fib}(0,k) & = & R(0,k) \\ \mathrm{Fib}(1,k) & = & R(1,k) \\ \mathrm{Fib}(n,k) & = & \mathrm{Fib}(n-1,\langle 2,n,k\rangle) \end{array}\right.$$

$$\left\{\begin{array}{rcl} \mathrm{Ack}(0,b,k) & = & R(b+1,k) \\ \mathrm{Ack}(a,0,k) & = & \mathrm{Ack}(a-1,1,k) \\ \mathrm{Ack}(a,b,k) & = & \mathrm{Ack}(a,b-1,\langle 4,a,k\rangle) \end{array}\right.$$

$$\left\{\begin{array}{rcl} R(x,\langle 0\rangle) & = & 0 \\ R(x,\langle 1\rangle) & = & 1 \\ R(x,\langle 2,n,k\rangle) & = & \mathrm{Fib}(n-2,\langle 3,x,k\rangle) \\ R(x,\langle 3,y,k\rangle) & = & R(x+y,k) \\ R(x,\langle 4,a,k\rangle) & = & \mathrm{Ack}(a-1,x,k) \end{array}\right.$$

4 The Thread Switching Function

The final step is coding the thread-switching function G. You can think of this as a GOTO-statement on anabolic steroids. Since we are writing mathematical functions rather than computer programs, we cannot use side effects to do the thread-switching, so we thread the entire computation (no pun intended!) with an additional structure — a thread. Note that the higher-order functions (or methods) you used in coding the thread behaviour need to be defunctionalised themselves, and hence the first tuple in the thread structure — the mathematical version of a line number:

$$\left\{\begin{array}{rcl} \mathrm{Fib}(0,k,t) & = & G(t,\langle 0,0,k\rangle) \\ \mathrm{Fib}(1,k,t) & = & G(t,\langle 1,1,k\rangle) \\ \mathrm{Fib}(n,k,t) & = & G(t,\langle 2,n,k\rangle) \end{array}\right.$$

$$\left\{\begin{array}{rcl} \mathrm{Ack}(0,b,k,t) & = & G(t,\langle 5,b,k\rangle) \\ \mathrm{Ack}(a,0,k,t) & = & G(t,\langle 6,a,k\rangle) \\ \mathrm{Ack}(a,b,k,t) & = & G(t,\langle 7,a,b,k\rangle) \end{array}\right.$$

$$
\begin{cases}
R(x, \langle 0 \rangle) &=& 0 \\
R(x, \langle 1 \rangle) &=& 1 \\
R(x, \langle 2, n, k \rangle, t) &=& G(t, \langle 3, x, n, k \rangle) \\
R(x, \langle 3, y, k \rangle, t) &=& G(t, \langle 4, x, y, k \rangle) \\
R(x, \langle 4, a, k \rangle, t) &=& G(t, \langle 8, a, x, k \rangle)
\end{cases}
$$

$$
\begin{cases}
G(\langle 0, 0, k \rangle, t) &=& R(0, k, t) \\
G(\langle 1, 1, k \rangle, t) &=& R(1, k, t) \\
G(\langle 2, n, k \rangle, t) &=& \mathrm{Fib}(n-1, \langle 2, n, k \rangle, t) \\
G(\langle 3, x, n, k \rangle, t) &=& \mathrm{Fib}(n-2, \langle 3, x, k \rangle, t) \\
G(\langle 4, x, y, k \rangle, t) &=& R(x+y, k, t) \\
G(\langle 5, b, k \rangle, t) &=& R(b+1, k, t) \\
G(\langle 6, a, k \rangle, t) &=& \mathrm{Ack}(a-1, 1, k, t) \\
G(\langle 7, a, b, k \rangle, t) &=& \mathrm{Ack}(a, b-1, \langle 4, a, k \rangle, t) \\
G(\langle 8, a, x, k \rangle, t) &=& \mathrm{Ack}(a-1, x, k, t)
\end{cases}
$$

We can now define $f(a, b, c)$ as an interface to the above functions:

$$
\begin{cases}
f(a, b, c) &=& \mathrm{Fib}(a, \langle 0 \rangle, \langle 9, b, c \rangle) \\
G(\langle 9, b, c \rangle, t) &=& \mathrm{Ack}(b, c, \langle 1 \rangle, t)
\end{cases}
$$