Generalization by Abstraction, once more

0 Introduction

The well-known programming problem of the *maximal segment sum* is not difficult to solve by means of functional programming. Initially, this gives rise to a solution with quadratic time complexity, but by means of the, by now well-understood, technique of *tupling* this solution can be transformed into one with linear time complexity.

Until about half a year ago, I was convinced that tupling was the only way to obtain a linear solution for this problem. Only recently, however, it dawned upon me that the, also well-understood (so I thought), technique of *Generalization by Abstraction* might be applicable here as well. This is a record of the result of this insight.

1 The Maximal Segment Sum

The following recursive function definitions provide a solution for the problem of the maximal segment sum. Their construction is not our concern here, this has been discussed at length in Chapter 9 of "Programming by Calculation":

$$\begin{array}{lll} & f\cdot[\,] & = & 0 \\ \& & f\cdot(b \triangleright s) & = & g\cdot(b \triangleright s)\ \mathsf{max}\ f\cdot s \\ \& & g\cdot[\,] & = & 0 \\ \& & g\cdot(b \triangleright s) & = & 0\ \mathsf{max}\ (b+g\cdot s) \end{array}$$

Function g has linear time complexity and, therefore, function f has quadratic time complexity. It has been known for a very long time by now[0], that the efficiency can be improved by an application of the well-understood tupling technique. This amounts to introducing a new function h, say, the value of which is the pair of the values of f and g; that is, function h has to satisfy this specification, for all integer lists s:

$$h\cdot s \ = \ \langle\, f\cdot s\,, g\cdot s\,\rangle \ .$$

In terms of h function f can be defined by:

$$f\cdot s \ = \ h\cdot s\cdot 0 \ ,$$

which just defines $f\cdot s$ as the left element of the pair $h\cdot s$.

A direct – not involving f or g anymore – recursive definition for h can be derived in a straigthforward way, with the following result, which has linear time complexity indeed:

$$\begin{array}{lll} & h\cdot[\,] & = & \langle\, 0\,, 0\,\rangle \\ \& & h\cdot(b \triangleright s) & = & \langle\, (b+y)\ \mathsf{max}\ x\,,\ 0\ \mathsf{max}\ (b+y)\,\rangle\ \ \mathsf{whr}\ \langle x\,, y\rangle = h\cdot s\ \ \mathsf{end} \end{array}$$

[0] Almost 30 years

2 A few simple but relevant properties

Without proof – which just requires case analysis, not even Mathematical Induction – we observe that f and g satisfy, for all s:

$$0 \le g\cdot s \quad \text{and:} \quad g\cdot s \le f\cdot s \quad .$$

As a consequence, we also have, of course:

$$0 \le f\cdot s \quad .$$

Finally, we will need the algebraic property that *addition distributes over maximum*, that is, for all integer x, y, z we have:

$$x + (y \mathbin{\mathsf{max}} z) \;=\; (x+y) \mathbin{\mathsf{max}} (x+z) \quad .$$

Elementary algebraic properties like commutativity and associativity will be used without explicit mentioning.

3 Generalization

We recall that the principle of *Generalization by Abstraction* entails that one tries to identify and exploit what several similar expressions have in common. By abstracting from their differences one often obtains a useful generalization.

From the definitions of f and g we derive that f also satisfies:

$$f\cdot(b \triangleright s) \;=\; 0 \mathbin{\mathsf{max}} (b+g\cdot s) \mathbin{\mathsf{max}} f\cdot s \quad .$$

The general pattern in this formula is that it is an expression of the shape, for some x, y:

$$x \mathbin{\mathsf{max}} (y+g\cdot s) \mathbin{\mathsf{max}} f\cdot s \quad .$$

The application $f\cdot s$ itself also matches this pattern, because:

$$(0) \qquad f\cdot s \;=\; 0 \mathbin{\mathsf{max}} (0+g\cdot s) \mathbin{\mathsf{max}} f\cdot s \quad ,$$

which follows from the properties of f and g formulated in Section 2 .

So, this brings us to the idea to introduce a new function h, say, with this specification, for all integer x, y and integer list s:

$$(1) \qquad h\cdot x\cdot y\cdot s \;=\; x \mathbin{\mathsf{max}} (y+g\cdot s) \mathbin{\mathsf{max}} f\cdot s \quad .$$

As we will see, parameter x will actually assume natural values only, so we can (and do) add $0 \le x$ as a precondition to this specification. This is not essential but it simplifies matters a little.

Because of property (0) function f can now be defined in terms of h:

$$f\cdot s \;=\; h\cdot 0\cdot 0\cdot s \quad .$$

For function h we derive:

$$
\begin{array}{ll}
 & h\cdot x\cdot y\cdot[\,] \\
= & \quad \{ \text{ specification of } h \ \} \\
 & x \ \mathsf{max}\ (y+g\cdot[\,])\ \mathsf{max}\ f\cdot[\,] \\
= & \quad \{ \text{ definitions of } f \text{ and } g \ \} \\
 & x \ \mathsf{max}\ (y+0)\ \mathsf{max}\ 0 \\
= & \quad \{ \ y+0=y \text{ and } 0\le x \ \} \\
 & x \ \mathsf{max}\ y \quad ,
\end{array}
$$

and:

$$
\begin{array}{ll}
 & h\cdot x\cdot y\cdot(b\triangleright s) \\
= & \quad \{ \text{ specification of } h \ \} \\
 & x \ \mathsf{max}\ (y+g\cdot(b\triangleright s))\ \mathsf{max}\ f\cdot(b\triangleright s) \\
= & \quad \{ \text{ definition of } f \text{ (from Section 1) } \} \\
 & x \ \mathsf{max}\ (y+g\cdot(b\triangleright s))\ \mathsf{max}\ g\cdot(b\triangleright s)\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \ 0 \text{ is the identity of } + \ \} \\
 & x \ \mathsf{max}\ (y+g\cdot(b\triangleright s))\ \mathsf{max}\ (0+g\cdot(b\triangleright s))\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \ + \text{ distributes over } \mathsf{max} \ \} \\
 & x \ \mathsf{max}\ ((y\,\mathsf{max}\,0)+g\cdot(b\triangleright s))\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \text{ definition of } g \ \} \\
 & x \ \mathsf{max}\ ((y\,\mathsf{max}\,0)+(0\,\mathsf{max}\,(b+g\cdot s)))\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \ + \text{ distributes over } \mathsf{max} \ ; \ 0 \text{ is the identity of } + \ \} \\
 & x \ \mathsf{max}\ y\ \mathsf{max}\ 0\ \mathsf{max}\ ((y\,\mathsf{max}\,0)+b+g\cdot s)\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \ 0\le x \ \} \\
 & x \ \mathsf{max}\ y\ \mathsf{max}\ ((y\,\mathsf{max}\,0)+b+g\cdot s)\ \mathsf{max}\ f\cdot s \\
= & \quad \{ \text{ specification of } h \text{, by Induction Hypothesis } \} \\
 & h\cdot(x\,\mathsf{max}\,y)\cdot((y\,\mathsf{max}\,0)+b)\cdot s \quad .
\end{array}
$$

Thus we obtain the following definition for function h:

$$
\begin{array}{lll}
 & h\cdot x\cdot y\cdot[\,] & = \ x \ \mathsf{max}\ y \\
\& & h\cdot x\cdot y\cdot(b\triangleright s) & = \ h\cdot(x\,\mathsf{max}\,y)\cdot((y\,\mathsf{max}\,0)+b)\cdot s
\end{array}
$$

Not only does this definition have linear time complexity, it also is tail-recursive. Hence, it can be encoded directly as an iterative program in a sequential programming language.

4 Epilogue

It is somewhat amazing that it took me almost 30 years to come up with this solution. Apparently, wonders will never cease...

The nice thing about this story is that the above solution is not difficult at all. The only hurdle to be taken is that one must have the insight, and the courage, to propose specification (1) for function h. From this, the rest follows by systematic calculation. For the latter, one must be aware of the relevant algebraic properties –such as that addition distributes over maximum–, but that is always the case with the calculational style of programming.

Eindhoven, 7 august 2013

Rob R. Hoogerwoord
department of mathematics and computing science
Eindhoven University of Technology
postbus 513
5600 MB Eindhoven