Elegant and Efficient Solution for Problem 4: Different Neighbour (QUANTA 2007)

Tom Verhoeff
Eindhoven University of Technology, The Netherlands
Email: `T.Verhoeff@TUE.NL` Web: `www.win.tue.nl`

17 November 2007
Lucknow, India

Given initial state: $\wedge$ and

$$a < b \ \wedge \ f(a) \neq f(b)$$

Desired final state: $\neq$ differs

$$a \leq r < b \ \wedge \ f(r) \neq f(r+1)$$

Possible pre-final state (introducing fresh variable):

$$a \leq r < s \leq b \ \wedge \ f(r) \neq f(s) \ \wedge \ s = r + 1$$

Loop invariant:

$$a \leq r < s \leq b \ \wedge \ f(r) \neq f(s)$$

Loop termination condition:

$$s = r + 1$$

Loop guard:

$$s \neq r + 1$$

Variant function for loop termination (decreases every iteration until 0):

$$s - (r + 1)$$

Initialisation of loop invariant: $:=$ replace by

$$r, s := a, b$$

Candidate statements for loop progress:

$$r := m$$

$$s := m$$

Condition on m to maintain invariant $a \le r < s \le b$ under progress:

$$r < m < s$$

To decrease $s - (r+1)$ most *under all circumstances* take the middle:

$$m := (r+s)\ \mathbf{div}\ 2$$

div integer division

Conditions to maintain inariant $f(r) \ne f(s)$ under progress:

$$(r := m)(f(r) \ne f(s))$$

$$(s := m)(f(r) \ne f(s))$$

Rewritten conditions:

$$f(m) \ne f(s) \to r := m$$
$$f(r) \ne f(m) \to s := m$$

$\to$ conditional

These conditions cover all possibilities, because if both would fail then

$$f(m) = f(s)\ \wedge\ f(r) = f(m)\ \Rightarrow\ f(r) = f(s)$$

$\Rightarrow$ implies

contradicting the invariant $f(r) \ne f(s)$

The completed abstract program:

$r, s := a, b$
; **do** $s \ne r + 1 \to$
  $m := (r+s)\ \mathbf{div}\ 2$
 ; **if** $f(m) \ne f(s) \to r := m$
  [] $f(r) \ne f(m) \to s := m$
  fi
od

In the C programming language:

```
r = a ; s = b ;
while ( s != r + 1 ) {
  m = ( r + s ) / 2 ;
  if    ( f(m) != f(s) )  r = m ;
  else /* f(r) != f(m) */ s = m ;
} /* end while */
```

Efficiency of this solution: constant memory, logarithmic time

32-bit integers go well over 10^9, and 64-bit exceeds 10^18. The difference between these two is a factor 10^9. If it takes 1 s for 10^9 then a linear program will take 10^9 more time for 10^{18}, or some 30 years. The logarithmic program merely doubles its execution time.

By the way, the linear program is best written as

```
r = a ;
while ( f(r) == f(r+1) ) {
  r = r + 1 ;
} /* end while */
```

Here is a construction simillar in style to the one given above:

Given initial state:

$$a < b \ \wedge \ f(a) \neq f(b)$$

Desired final state:

$$a \leq r < b \ \wedge \ f(r) \neq f(r+1)$$

Loop invariant:

$$a \leq r < b \ \wedge \ f(r) \neq f(b)$$

Loop termination condition:

$$f(r) = f(r+1)$$

Loop guard (note that this is well-defined when the invariant holds):

$$f(r) \neq f(r+1)$$

Variant function for loop termination (decreases every iteration until 0):

$$b - (r+1)$$

Initialisation of loop invariant:

$$r := a$$

Candidate statement for loop progress:

$$r := r + 1$$

Condition to check:

$$f(r) = f(r{+}1) \wedge a \leq r < b \wedge f(r) \neq f(b) \ \Rightarrow \ a \leq r{+}1 < b \wedge f(r{+}1) \neq f(b)$$