Decomposing an integer as sum of two squares

(I had not planned to write this EWD, but, after I had given this problem as homework, it turned out that the majority of my students had problems with it. My personal difficulty is, of course, that I now have to forget all the old-fashioned ways in which I have solved this problem in distant pasts.)

It is requested to print for $N \geq 0$ all natural solutions of

(0) $\quad u, v: (u^2 + v^2 = N \ \wedge \ u \leq v)$.

The solutions have to be printed in <u>some</u> order. Since we are looking for zeros of $u^2 + v^2 - N$, an <u>increasing</u> function of both u and v , the order of increasing u is at the same time the order of decreasing v . Let us take that order in one of its directions. Arbitrarily - this is a choice we may regret - I propose as one of the conjuncts of the invariant

P0 all solutions with $u < x$ have been printed

(Note that I have chosen $<$ to characterize the upper bound: there may be a solution with $u = 0$ and I wish to characterize the empty set of printed solutions at initialization with natural x) .

Initialization of x is no problem: $x:=0$. The analysis of the question of for which value of x we can guarantee that all solutions of (0) have been printed is postponed because it depends intimately on the precise shape of (0). We should first investigate what we can do with the fact that we are looking for zeros of an $f.(u,v)$ which is increasing in both arguments. We conclude

- each u is combined with at most one v and vice versa
- if values for u are tried in increasing order, those for v should be tried in decreasing order.

(The first conclusion is based on f being increasing; for the second conclusion, f being ascending suffices. If we have the knowledge, this is the moment to be inspired by the Welfare Crook with 2 sequences instead of with 3, or by the Saddleback Search.)

How can we conclude that x is a u-value to be combined with no v-value? From the existence of a y such that —remember that f is ascending in its second argument—

$$f.(x,y)<0 \ \wedge \ f.(x,y+1)>0 \quad .$$

How can we conclude that x is a u-value for which there is a v-value? From

$$f.(x,y)=0 \quad ;$$

in that case y is the corresponding v-value we were looking for and, because f is increasing in its second argument, we then conclude that

$$f.(x,y+1) > 0 \quad .$$

Combining the two observations, we propose to add

$$\text{P1:} \qquad x^2 + (y+1)^2 > N$$

as conjunct to the invariant.

Now we return to the postponed question of an appropriate final value of x. All solutions have been printed if $((u,v) \text{ solves } (0)) \Rightarrow u < x$. We observe for any (u,v) that solves (0):

$$
\begin{array}{ll}
 & u < x \\
= & \{u \text{ and } x \text{ natural}\} \\
 & 2\cdot u^2 < 2\cdot x^2 \\
\Leftarrow & \{\text{arithmetic}\} \\
 & 2\cdot u^2 \leq u^2 + v^2 \ \wedge \ u^2 + v^2 < 2\cdot x^2 \\
= & \{(u,v) \text{ solves } (0)\} \\
 & N < 2\cdot x^2 \\
\Leftarrow & \{\text{arithmetic}\} \\
 & N < x^2 + (y+1)^2 \ \wedge \ x^2 + (y+1)^2 \leq 2\cdot x^2 \\
= & \{\text{P1}\} \\
 & (y+1)^2 \leq x^2 \\
= & \{x \text{ and } y \text{ natural}\} \\
 & y+1 \leq x \\
= & \{\text{arithmetic}\}
\end{array}
$$

$x > y$,

which suggest the guard $x \le y$ and the further — not so interesting — conjunct of the invariant

P2: $0 \le x \quad \wedge \quad x \le y+1$.

In the above seven-step derivation — probably the price for the chosen direction of solution generation — we have been fortunate. In the path from $u < x$ to $x > y$ we have strengthened the predicate twice; is $x > y$ perhaps stronger than necessary? Or: is the corresponding guard $x \le y$ perhaps weaker than necessary? The answer is no, because (0) might have a solution with $u = v$.

With invariant

P: $P0 \wedge P1 \wedge P2$

a correct solution is now

```
|[ var x,y: int;  x,y:= 0,0  {P0 ∧ P2}
 ; do x²+y² ≤ N → y:= y+1 od
   {x²+y² > N, hence P1, hence P}
 ; do x ≤ y →
        if x²+y² > N → y:=y-1
        [] x²+y² < N → x:= x+1
        [] x²+y² = N → print (x,y); x:= x+1
        fi
   od
]| .
```

Our last step —under the assumption that squaring is an expensive operation— is a coordinate transformation. We shall do it in two steps, introducing new variables in the first one and deleting old variables in the last one.

Let us add the conjunct

P3: $c = x^2 + y^2 - N \;\wedge\; p = 2\cdot x + 1 \;\wedge\; q = 2\cdot y + 1$.

```
|[ var x,y,c,p,q: int; x,y,c,p,q := 0,0,-N,1,1 {P3}
 ; do c ≤ 0 → c:= c+q; y,q:= y+1, q+2 {P3} od {P3}
 ; do x ≤ y →
        if c > 0 → y,q := y-1, q-2; c:= c-q {P3}
        [] c < 0 → c:=c+p; x,p:= x+1, p+2 {P3}
        [] c = 0 → print (x,y)
                   ; c:= c+p; x,p := x+1, p+2 {P3}
        fi
   od
]|
```

Thanks to P3, $x \leq y \equiv p \leq q$ and $(x,y) = ((p-1)/2, (q-1)/2)$. Making these substitutions, we can eliminate x and y:

```
|[ var c,p,q: int; c,p,q := -N,1,1
 ; do c ≤ 0 → c:= c+q; q:= q+2 od
 ; do p ≤ q → if c>0 → q:= q-2; c:= c-q
              [] c<0 → c:= c+p; p:= p+2
              [] c=0 → print ((p-1)/2, (q-1)/2)
              fi
   od
]| .
```

<u>Note</u> Once we have established that $y := y+1$ translates under invariance of P3 into

$$\{P3\}\ c := c+q;\ \ y, q := y+1, q+2\ \{P3\}\quad ,$$

the translation of the inverse statement $y := y-1$ is mechanical: the inverse statements of the translation in inverse order. (End of Note.)

* * *

The transition from the x,y-program to the c,p,q-program has only been added to show in one of my handouts an example of a coordinate transformation, complete with its justification by means of an invariant describing the relation between old and new coordinates.

Comparing my derivation of the x,y-program with the corresponding chapter in "A discipline of programming", I can only conclude that my memory is too good —or, equivalently, that I am not too good at getting rid of my past—. The current derivation displays a more successful disentanglement by isolating the consequences of the fact that we are dealing with a function that is increasing in both of its arguments. Also, today's solution is a bit more elegant, and in its derivation the stepping stone of a program with two nested repetitions has been avoided.

One of ALGOL60's major mistakes – faithfully copied from FORTRAN – was the introduction of the "controlled variable" that was an intrinsic part of each repetitive construct. The notion of the controlled variable was killed in Euclid's algorithm

$\{0 \le X \wedge 0 \le Y\}$ $x, y := X, Y$
; **do** $x > y \rightarrow x := x - y$
  $[]$ $y > x \rightarrow y := y - x$
od $\{x = X \ \underline{\text{gcd}}\ Y\}$

and I very well remember my excitement when I saw this code for the first time: it identified the notion of the controlled variable as a confusing artefact. (The above program was the first program I wrote with a repetition with more than 1 guarded command; it strongly suggested that repetitions with guarded command sets might be not a bad idea.) Nevertheless, old habits die hard, and several years later I still approached this problem by nested repetitions: the outer one for x, the inner one for y. A sobering thought.

Austin, 13 October 1988

prof.dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
United States of America