Prive

COMMENTS ON "MERGING PROBLEMS REVISITED"

by

Maarten M. Fokkinga
Twente University of Technology,
Department of Applied Mathematics,
P.O. Box 217, 7500 AE ENSCHEDE,
The Netherlands.

August 1978

Abstract. It is shown that the invariant relation in the chapter "Merging Problems Revisited" of "A Discipline of Programming" by E.W. Dijkstra (Prentice Hall, 1976) is unnecessarily strong and complicated, and that a suitable, weaker and simpler invariant relation may be obtained more easily. This holds independently of the knowledge of the final representation of the sets.

Keywords and phrases. Invariant relation, variant function, guarded commands, programming methodology.

Contents.

1. Introduction

In the chapter "Merging Problems Revisited", Dijkstra gives a formal treatment of the development of a program for the establishment of the union of two sets of integers. He gives two pages of reasoning for the formulation of the invariant relation; included in the reasoning is an unmotivated theorem. We show in section 2 that the invariant is unnecessarily complicated and constraning the development of the algorithm. A much simpler invariant will do as well, while its formulation seems to require less invention and inspiration. Even if the ultimate representation by monotonically increasing functions is taken into account in an early stage, the above holds true as shown in section 4.

For ease of comparison we follow Dijkstra's notation So we write $+$ and $*$ for union and intersection; $z * \{e\} = \{e\}$ then means that e is a member of z, and $x * X = x$ that x is a subset of X. The tilde in $x \tilde{+} y$ demands that $x * y = \emptyset$, and in $x \simeq y$ that $x * y = y$.

2. The simpler invariant relation

On p. 124 line 10 Dijkstra starts the development of the algorithm. At this very point we propose to deviate from his approach. We try the standard technique

"what you want to be computed is
the result achieved hitherto together with
what is still to be computed"

in order to find a suitable invariant. Application and a little inspiration yield

P1: $X + Y = z + (x+y)$

where x,y are variables with initial values X,Y and z is the result variable -- with initial value $\emptyset$ -- .

Remark. The standard technique also yields the following invariants.

For summation : $t1+\cdot\cdot+tn = z + ti+\cdot\cdot+tn$,

for multipication : $t1*\cdot\cdot*tn = z * ti*\cdot\cdot*tn$,

for exponentiation: $X^Y = z * x^y$

Characteristic is the view towards the future -- what still is to be computed -- in stead of back to the history -- the way z

has been computed. Compare e.g. the above relations with

for summation : $z = t1+\cdots+ti-1$,

for multiplication: $z = t1*\cdots*ti-1$,

for exponentiation: $z = X^{Y-2^j.y}$ <u>and</u> $x = X^{2^j}$ for some j.

(End of remark.)

Remarkably Dijkstra also mentions P1 , but then immediately applies The Theorem. The invariant relations thus found essentially allow to identify z, x and y with members of a partioning of X+Y , X and Y respectively. It is this identification (and the wish for it!) which is completely absent from our reasoning, and which, we think, is caused by a too operational point of view, looking back to the computational history, as expressed on page 124 lines 10-24. (Not requiring that " x,y is a subset of X,Y " (although it turns out to be so) might be compared with not requiring that " x,y is a particular multiple or fraction of X,Y " (although it turns out to be so) as addition to the invariant $X^Y = z*x^y$ for exponentiation. Both requirements might unnecessarily restrict the massaging of x,y and z .)

The relation P1 is sufficient for the development of an algorithm. However, let us now assume that $z:=z\tilde{+}\{e\}$ is the only operation available for addition of elements e to z , and that it will pay off to avoid testing whether e already belongs to z or not. (But note that this assumption need not be valid if sets are represented by unordered lists or boolean functions.) So we may try to guarantee that such elements are <u>not</u> member of z . On account of P1 the only candidates to be added to z are the elements of x+y . Thus we take as additional invariant

P2: $z*(x+y) = \emptyset$, or equivalently

P2': $z*x = \emptyset$ <u>and</u> $z*y = \emptyset$.

Similarly to the calculations on p. 128 we may now verify that the following alternative keeps P1 <u>and</u> P2 invariant.

A1: $x*\{e\} = \{e\}$ <u>and</u> $y*\{e\} = \emptyset \rightarrow z,x:=z\tilde{+}\{e\},\ x\tilde{-}\{e\}$.

But P1 and P2 leave us greater freedom; the following do as well,

A2: $x*\{e\} = \{e\}$ <u>and</u> $y*\{e\} = \{e\} \rightarrow x:=x\tilde{-}\{e\}$,

A3: $x*\{e\} = \{e\}$ <u>and</u> $y*\{e\} = \emptyset \rightarrow x,y:=x\tilde{-}\{e\},y\tilde{+}\{e\}$.

Dijkstra's invariant relations, viz.

$x*X = x$ $\quad$ $y*Y = y$

$x*(Y\tilde{-}y) = \emptyset$ $\quad$ $y*(X\tilde{-}x) = \emptyset$

$z = (X\tilde{-}x) + (Y\tilde{-}y)$,

may each be disturbed by A2 and A3 . So clearly they unnecessarily restrict the development. It should be avoided to preclude potential algorithms, if this is not motivated by assumptions about the sets involved.

Remarkably, the constraning effect of Dijkstra's invariant may be achieved approximately by the choice of a rather simple variant function. Consider for instance

T1: $card(x+y)$,

T2: $card(x) + card(y)$.

Both A2 and A3 do not decrease T1 , but ~~they do~~ A2 does decrease T2 .
We said "approximately" because a combination of A1 and A2 (and of A1 and A3) does decrease T1 :

A4: $\{e1,e2\}*x = \{e1,e2\}$ <u>and</u> $\{e1,e2\}*y = \{e2\} \rightarrow z,x := z\tilde{+}\{e1\},\ x\simeq\{e1,e2\}$.

3. Generalization to intersection and exclusive union

The generalization to the problem "establish $X\oplus Y = z$" , where $\oplus$ varies over $+$, $*$ and $\underline{+}$ (defined by $x\underline{+}y = (x+y)\simeq(x*y)$) is straightforward.
The relations and functions become:

P1($\oplus$): $X\oplus Y = z+(x\oplus y)$,

P2($\oplus$): $z*(x\oplus y) = \emptyset$,

P2'($\oplus$): $z*x = \emptyset$ <u>and</u> $z*y = \emptyset$,

T1($\oplus$): $card(x\oplus y)$,

T2($\oplus$): $card(x) + card(y)$.

(Jvd Snepsch)

There are three things to note. First, P1($\oplus$) shows the two different roles of $+$ in P1($+$) and T2($\oplus$) shows another role of + . Second, P2'($\oplus$) is stronger than P2($\oplus$) ; for instance,

A5: $x*\{e\} = \{e\}$ <u>and</u> $y*\{e\} = \emptyset \rightarrow z,y := z\tilde{+}\{e\},\ y\tilde{+}\{e\}$

does leave P1($\underline{+}$) <u>and</u> P2($\underline{+}$) invariant, but not P1($\underline{+}$) <u>and</u> P2'($\underline{+}$) .
Third, variant function T1 is less suitable for $*$ and $+$: processing a member of x which does not belong to $x\underline{+}y$ doesnot decrease T1 . Indeed, on account of P1 <u>and</u> P2 , T1 equals $card(X\oplus Y) - card(z)$, thus requiring that in each alternative z must be extended.

4. Representing sets by monotonic functions

We have already remarked twice that the development of the algorithm heavily depends on (or anticipates) the representations of the sets involved. In this section we show how the development could have read, in case we would have known in advance the representation of sets by monotonically increasing functions. For simplicity we only pursue the case for exclusive union.

The main invariant relation P1($\underline{+}$) and variant function T2($\underline{+}$) are obtained as before. Further, the representation by monotonically increasing functions eases the following operations in particular: determination, extension and removal of the least (and greatest) element. For concreteness sake we write these as f.low, f:loext(e), f:lorem (and f.high, f:hiext(e) and f.hirem) respectively. We denote by fz, fx and fy the representations of the sets z, x and y .

In order that fz be monotonically increasing and be built up by fz:hiext(e) only -- the operation fz:loext(e) would give rise to a symmetric solution -- , we try to guarantee that z is extended only with elements greater than those already contained in it. Hence we take as additional invariant
P3: $z < (x \underline{+} y)$, or slightly stronger
P3': $z < x$ and $z < y$,
where $<$ is defined by $x < y \Longleftrightarrow \underline{A} e \epsilon x, e' \epsilon y: e < e'$.
So, if we take care to apply fz:hiext(e) only with e from $x \underline{+} y$, and only if P3 holds, then is is guaranteed that fz is monotonically increasing. (In Dijkstra's development this has been verfied afterwards!)

Now we consider a decrease of the variant function*) : let e be fx:low , then fx.lorem removes e from x . Two cases arise. First, if $\{e\}*(x \underline{+} y) = \{e\}$ then $\{e\}*y = \emptyset$. So, extension of z with e leaves P1 invariant, and in order that P3 remains invariant, $\{e\}<y$ should hold. Thus we find -- and may formally verify -- the alternative A6 and analogously A7 :
A6: $x \neq \emptyset$ $\underline{\text{cand}}$ {fx.low}<y → fz:hiext(fx.low); fx:lorem ,
A7: $y \neq \emptyset$ $\underline{\text{cand}}$ {fy.low}<x → fz:hiext(fy.low); fy:lorem .
Second, if $\{e\}*(x \underline{+} y) = \emptyset$ then $\{e\}*y = \{e\}$. So, deletion of e from y

---

*) Note that we do not have the inspiration of processing the least element of $x \underline{+} y$.

leaves P1 invariant, and in order that this may be performed by fy:lorem , e = fy.low should hold. Thus we find -- and may formally verify --

```
A8:  x≠∅  and  y≠∅  and  fx.low = fy.low → fx:lorem; fy:lorem.
```

Fortunately, A6 - A8 are sufficient: after

```
S:  do  A6 □ A7 □ A8  od
```

the relation $x=\emptyset$ and $y=\emptyset$, hence $X \underline{+} Y = z$ holds! And the guards are easily represented; e.g. $\{fx.low\}<y$ is equivalent to $y=\emptyset$ cor $fx.low<fy.low$. Moreover, if we make the same assumption about the additional value $inf = fx.high = fy.high$ as Dijkstra does, then the alternatives may take the following concrete form.

```
A6': fx.low<fy.low → fz:hiext(fx.low); fx:lorem ,
A7': fy.low<fx.low → fz.hiext(fy.low); fy:lorem ,
A8': inf≠fx.low = fy.low → fx:lorem; fy:lorem .
```

Thus we obtain a more efficient program than Dijkstra does, because the test for $fx.low \neq inf$ or $fy.low \neq inf$ is missing in do A6' □ A7' □ A8' od .

Note that again the relation P1 and P3 , although stronger than P1 and P2 , is weaker than Dijkstra's invariant. E.g. the following alternative leaves P1 and P3 invariant, but disturbs Dijkstra's (and doesn't decrease T2).

```
A9: x≠∅  cand  {fx.low}<y → fy:loext(fx.low); fx:lorem .
```

Remark. For completeness sake we describe the next phase in the development of the above program S . We will make implicit determinacy explicit, so that unnecessary evaluation of guards is avoided. Indeed, if $x=\emptyset$ then only the guard of A7 may hold, and similarly if $y=\emptyset$ then only the guard of A6 . Hence we place all alternatives under the (joint) guard $x\neq\emptyset$ and $y\neq\emptyset$ (and simplify their guards); after the repetitive construct either A7 or A6 should be repeated. Thus

```
S':  do  x≠∅  and  y≠∅ →
          if  fx.low<fy.low → fz:hiext(fx.low);fx:lorem
           □  fy.low<fx.low → fz:hiext(fy.low);fy:lorem
           □  fx.low=fy.low → fx:lorem;fy:lorem
          fi
     od;
     if  x=∅ → do  y≠∅ → fz:hiext(fy.low); fy:lorem  od
      □  y=∅ → do  x≠∅ → fz:hiext(fx.low); fx:lorem  od
     fi .
```

The last statement <u>if</u> $x=\emptyset \rightarrow DO1 \;\square\; y=\emptyset \rightarrow DO2$ <u>fi</u> may even be written DO1; DO2 ; this only shortens the text. (End of remark.)

5. Final remark

May be some other tacitly assumed goals have led to Dijkstra's invariant. For example, knowing that X and Y are represented by monotonically increasing functions fX and fY , we may wish to represent the sets x and y merely by a cut (that is, an index) in the representation of X and Y ; say <u>var</u> ix, iy:integer . The operations fx.low and fx.lorem should then be translated into fX(ix) and ix:=ix+1 respectively. This wish indeed leads to

P4: $x*X=x$ <u>and</u> $y*Y=y$ <u>and</u> $X \simeq x<x$ <u>and</u> $Y \simeq y<y$

as additional invariant -- in both our and Dijkstra's approach -- .

But still the term $x*(Y \simeq y)=\emptyset$ <u>and</u> $y*(X \simeq x)=\emptyset$ seems unnecessary!