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
2. The simpler invariant
3. Generalization to intersection and exclusive union
4. Representing sets by monotonic functions
5. Final remark
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 constraining 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; $z \in \{e\} = \{e\}$ 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 $x \tilde{=} y = \emptyset$ and in $x \approx y$ that $x \approx y = y$.

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 : $t1+\ldots+tn = z + t1+\ldots+tn$ ,
for multiplication : $t1*\ldots*tn = z * t1*\ldots*tn$ ,
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:  \( z = t_1 + \cdots + t_{i-1} \),
for multiplication:  \( z = t_i \cdots t_{i-1} \),
for exponentiation:  \( z = X^{2^j} \cdot y \) and \( x = X^{2^j} \) 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 partitioning 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 \( X^Y = z^x \) 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 \( z := z^x(e) \) is the only operator 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 not 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:  \( z^x(x+y) = \emptyset \) , or equivalently
P2': \( z^x = \emptyset \) and \( z^y = \emptyset \).

Similarly to the calculations on p. 128 we may now verify that the
following alternative keeps P1 and P2 invariant.
A1: \( x^e(e) = \{ e \} \) and \( y^e(e) = \emptyset \) + \( z, x := z^x(e) \), \( x^z(e) \).
But P1 and P2 leave us greater freedom; the following do as well,
A2: \( x^e(e) = \{ e \} \) and \( y^e(e) = \{ e \} \) + \( x := x^z(e) \),
A3: \( x^e(e) = \{ e \} \) and \( y^e(e) = \emptyset \) + \( x, y := x^z(e), y^z(e) \).
Dijkstra's invariant relations, viz.
\[
\begin{align*}
x^x = x & \quad y^y = y \\quad x^y(y^x) = \emptyset & \quad y^x(x^y) = \emptyset \\
z = (x^z) + (y^z) & ,
\end{align*}
\]
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 constraining effect of Dijkstra's invariant may be achieved approximately by the choice of a rather simple variant function. Consider for instance

\[
\begin{align*}
T1 &: \text{card}(x+y) , \\
T2 &: \text{card}(x) + \text{card}(y) .
\end{align*}
\]

Both \( A2 \) and \( A3 \) do not decrease \( T1 \), but they do decrease \( T2 \).

We said "approximately" because a combination of \( A1 \) and \( A2 \) (and of \( A1 \) and \( A3 \)) does decrease \( T1 \):

\[
\begin{align*}
A4 &: \{e1,e2\} \times x = \{e1,e2\} \text{ and } \{e1,e2\} \times y = \{e2\} + z , x:=z\{e1\} , x=\{e1,e2\} .
\end{align*}
\]

3. Generalization to intersection and exclusive union

The generalization to the problem "establish \( x \otimes y = z \)" , where \( \otimes \) varies over \(+, *, +\) (defined by \( x\times y = (x+y)=x\times y \)) is straightforward.

The relations and functions become:

\[
\begin{align*}
P1(\otimes) &: x \otimes y = z+(x\otimes y) , \\
P2(\otimes) &: x\times(y\otimes y) = \emptyset , \\
P2'(\otimes) &: x\times y = \emptyset \text{ and } z\times y = \emptyset , \\
T1(\otimes) &: \text{card}(x\otimes y) , \\
T2(\otimes) &: \text{card}(x) + \text{card}(y) .
\end{align*}
\]

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

\[
\begin{align*}
A5 &: x\times(e) = \{e\} \text{ and } y\times(e) = \emptyset + z , y:=z\{e\} , y\{e\}
\end{align*}
\]

does leave \( P1(+) \) and \( P2(+) \) invariant, but not \( P1(+) \) and \( P2'(+) \).

Third, variant function \( T1 \) is less suitable for \(*\) and \(+\): processing a member of \( x \) which does not belong to \( x\times y \) does not decrease \( T1 \). Indeed, on account of \( P1 \) and \( P2 \), \( T1 \) equals \( \text{card}(x\otimes y) - \text{card}(z) \), 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 $P_1(\cdot)$ and variant function $T_2(\cdot)$ 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 $f:\mathrm{loext}(e)$, $f:\mathrm{lorem}$ (and $f:\mathrm{hiext}(e)$ and $f:\mathrm{hirem}$) 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 $fz:\mathrm{hiext}(e)$ only -- the operation $fz:\mathrm{loext}(e)$ 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

$P_3$: $z \prec (x+y)$, or slightly stronger $P_3'$: $z \prec x$ and $z \prec y$,

where $\prec$ is defined by $x \prec y \iff \exists e \in x, e' \in y: e < e'$.

So, if we take care to apply $fz:\mathrm{hiext}(e)$ only with $e$ from $x+y$, and only if $P_3$ holds, then is is guaranteed that $fz$ is monotonically increasing. (In Dijkstra's development this has been verified afterwards!)

Now we consider a decrease of the variant function $^*): let e$ be $fx:\mathrm{low}$, then $fx:\mathrm{lorem}$ removes $e$ from $x$. Two cases arise. First, if $(e)\ast(x+y) = \{e\}$ then $(e)\ast y = \emptyset$. So, extension of $z$ with $e$ leaves $P_1$ invariant, and in order that $P_3$ remains invariant, $(e)\ast y$ should hold. Thus we find -- and may formally verify -- the alternative $A_6$ and analogously $A_7$:

$A_6$: $x \neq \emptyset \land (fx:\mathrm{low}) \prec y \Rightarrow fz:\mathrm{hiext}(fx:\mathrm{low}) = fx:\mathrm{lorem}$ ,

$A_7$: $y \neq \emptyset \land (fy:\mathrm{low}) \prec x \Rightarrow fz:\mathrm{hiext}(fy:\mathrm{low}) = fy:\mathrm{lorem}$ .

Second, if $(e)\ast(x+y) = \emptyset$ then $(e)\ast y = \{e\}$. So, deletion of $e$ from $y$

$^*$) Note that we do not have the inspiration of processing the least element of $x+y$. 
leaves $P_1$ invariant, and in order that this may be performed by
fy:lorem, $e = fy\.low$ should hold. Thus we find -- and may formally
verify --

$$A_8: x \neq \emptyset \text{ and } y \neq \emptyset \text{ and } fx\.low = fy\.low + fx\.lorem; fy\.lorem.$$ 

Fortunately, $A_6 - A_8$ are sufficient: after

$$S:\ \text{do } A_6 \square A_7 \square A_8 \text{ od}$$

the relation $x = \emptyset$ and $y = \emptyset$, hence $x + y = z$ holds! And the guards are
easily represented; e.g. $\{fx\.low\} < y$ is equivalent to $y = \emptyset$ \text{ 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.

$$A'_6: fx\.low < fy\.low + fz\.hiext(fx\.low); fx\.lorem,$$
$$A'_7: fy\.low < fx\.low + fz\.hiext(fy\.low); fy\.lorem,$$
$$A'_8: \text{inf} = fx\.low = fy\.low + fx\.lorem; fy\.lorem.$$ 

Thus we obtain a more efficient program than Dijkstra does, because the test
for $fx\.low \neq \text{inf}$ or $fy\.low \neq \text{inf}$ is missing in $\text{do } A'_6 \square A'_7 \square A'_8 \text{ od}$. 

Note that again the relation $P_1$ and $P_3$, although stronger than
$P_1$ and $P_2$, is weaker than Dijkstra's invariant. E.g. the following
alternative leaves $P_1$ and $P_3$ invariant, but disturbs Dijkstra's
(and doesn't decrease $T_2$).

$$A_9: x \neq \emptyset \text{ cand } \{fx\.low\} < y + fy\.loext(fx\.low); fx\.lorem.$$ 

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 $A_7$ may hold, and similarly if $y = \emptyset$ then
only the guard of $A_6$. 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 $A_7$ or $A_6$ should be repeated. Thus

$$S': \ \text{do } x \neq \emptyset \text{ and } y \neq \emptyset +$$
$$\text{if } fx\.low < fy\.low + fz\.hiext(fx\.low); fx\.lorem$$
$$\text{ od; if } x = \emptyset + \text{ do } y \neq \emptyset + fz\.hiext(fy\.low); fy\.lorem \text{ od}$$
$$\text{ od; if } y = \emptyset + \text{ do } x \neq \emptyset + fz\.hiext(fx\.low); fx\.lorem \text{ od}$$
$$\text{ fi }.$$
The last statement \texttt{if } x=\emptyset \rightarrow D01 \land y=\emptyset \rightarrow D02 \texttt{fi} may even be written D01; D02; 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 \texttt{var} 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

\textbf{P4:} $x^*X=x$ and $y^*Y=y$ and $X=x^<x$ and $Y=y^<y$

as additional invariant -- in both our and Dijkstra’s approach -- . But still the term $x^*(Y=y)=\emptyset$ and $y^*(X=x)=\emptyset$ seems unnecessary!