

For the record: Batcher's Baffler

In this note we consider a special sorting routine for array  $f(i: 0 \leq i < N)$ . Predicate OK is given by

$$OK.i:j \equiv f.i \leq f.j \quad ;$$

quantifications over  $OK.i:j$  are implicitly constrained by

$$0 \leq i < j < N \quad .$$

The functional specification of Batcher's Baffler is

$\llbracket N: \text{int } \{N \geq 0\}$

$;$   $\llbracket f(i: 0 \leq i < N): \text{array of int } \{ \text{bag. } f = B \}$

$;$  Batcher's Baffler

$\{ \text{bag. } f = B \wedge (\underline{A}i :: OK.i.(i+1)) \}$

$\rrbracket$

$\rrbracket$

From now on we no longer mention the invariance of  $\text{bag. } f = B$ ; it is trivially maintained as the algorithm only manipulates array  $f$  by means of the operation  $\text{Ord}$  given by

$$\text{Ord.}i:j = \text{if } OK.i:j \rightarrow \text{skip } \rrbracket \neg OK.i:j \rightarrow \text{swap.}f.i:j \text{ fi}$$

— the swap interchanges the values of  $f.i$  and  $f.j$  — .

The  $\text{Ord}$  operation satisfies  $\{ \text{true} \} \text{Ord.}i:j \{ OK.i:j \}$ ; quantifications over it are again implicitly constrained by

$$0 \leq i < j < N.$$

There are many ways of expressing our postcondition that  $f$  is ascending, but this one is a nice starting point for the invariant

$P_0: (\underline{A}i :: OK.i.(i+t))$

which suggests for Batchier's Baffler the form

"establish  $t \geq N$ "  $\{P_0\}$   
 ; do  $t \neq 1 \rightarrow$  "decrease  $t$  under invariance of  $P_0$ " od .

Let the decrease of  $t$  under invariance of  $P_0$  involve the transition from  $t=t'$  to  $t=t''$  with  $t'' < t'$ . It would be nice if we could exploit the precondition  $(\underline{A}i :: OK.i.(i+t'))$  by keeping it invariant; we can only hope to do so provided it is implied by the postcondition  $(\underline{A}i :: OK.i.(i+t''))$ , i.e. provided  $t''$  is a factor of  $t'$ . Under that constraint, the most modest decrease of  $t$  - i.e. the one that strengthens  $P_0$  as little as possible - is halving  $t$ , and therefore we suggest to restrict  $t$  to powers of 2. (The confirmation of the wisdom of this choice will come later.)

Our next approximation of Batchier's Baffler gives explicitly the manipulations on  $t$  :

$t := 1$ ; do  $t < N \rightarrow t := 2 \cdot t$  od  $\{P_0 \wedge t \text{ is a power of } 2\}$

do  $t \neq 1 \rightarrow t := t/2$

;  $\{P_1: (\underline{A}i :: OK.i.(i+2 \cdot t))\}$

"restore  $P_0$ "

$\{P_0: (\underline{A}i :: OK.i.(i+t))\}$

od

The rest of this note is concerned with the algorithm for "restore  $P_0$ " as specified by its pre- and postconditions in the last approximation. (For this subalgorithm it is no longer relevant that  $t$  is a power of 2).

The design of Batcher's Baffler is driven by the desire of finding groups of Ord operations with disjoint argument pairs, because such Ord operations could be executed concurrently. As each Ord operation establishes the corresponding OK relation, we are invited to try to partition the OK relations in the postcondition  $P_0$  such that the argument pairs in each group are disjoint. This is achieved by writing  $P_0$  as  $P_2 \wedge P_3$  with

$$P_2: (\underline{A}i: e.i: \text{OK}.i.(i+t)) \quad \text{and}$$

$$P_3: (\underline{A}i: \neg e.i: \text{OK}.i.(i+t)) \quad ,$$

provided we can find a predicate  $e$  such that

$$e.i \equiv \neg e.(i+t) \quad .$$

There are many such predicates, all variations on the same theme. The simplest is

$$e.i \equiv (i \bmod 2 \cdot t) < t \quad .$$

Remark. It is the factor of 2 in the above formula that will justify our earlier choice to restrict  $t$  to powers of 2. (End of Remark.)

Using  $\parallel$  to denote the potentially concurrent combination, we have indeed

$$\{\text{true}\} (\parallel i: e.i: \text{Ord}.i.(i+t)) \{P_2\} \quad (o)$$

$$\{\text{true}\} (\|i: \neg e.i: \text{Ord}.i.(i+t)) \{P_3\} \quad (1)$$

But we cannot achieve  $P_2 \wedge P_3$  - i.e.  $P_0$  - by performing (0) and (1) consecutively (in some order), for in general the second one will destroy what the first one has accomplished. So we have to proceed more carefully, e.g. first establishing  $P_2$  by means of (0) and then establishing  $P_3$  with a repetition for which  $P_2$  is an invariant, i.e. we may expect that repetition to establish  $P_2 \wedge P_3$  under invariance of  $P_2 \wedge P_4$  where  $P_4$  is some generalization of  $P_3$ .

Remark I did not fully check it, but my impression is that the choice just made is irrelevant, and that we could equally well have established  $P_3$  first. (End of Remark.)

For our analysis we rewrite - because it is slightly more convenient -  $P_3$  as

$$P_3: \quad (\underline{A}i: e.i: \text{OK}.(i+t).(i+2 \cdot t))$$

and (1) as

$$(\|i: e.i: \text{Ord}.(i+t).(i+2 \cdot t)) \quad (1)$$

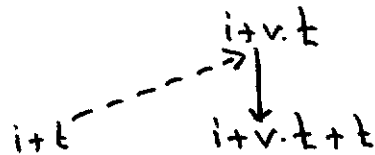
and ask ourselves under what further conditions the generalized operation

$$(\|i: e.i: \text{Ord}.(i+t).(i+v \cdot t)) \quad \text{with even } v \quad (2)$$

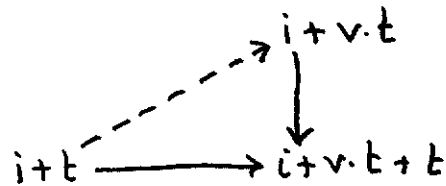
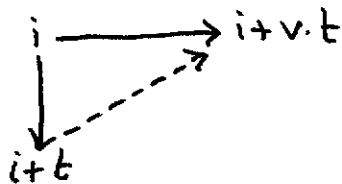
maintains  $P_2$ . (Parameter  $v$  is constrained to even values so as to ensure that the argument pairs are disjoint.) In our analysis we shall use the lemmata and -grudgingly- the notation of EWD 932 b.

(i) The OK relations of  $P_2$  with no argument involved in an Ord operation of (2) are maintained on account of Lemma 0.

(ii) Of the OK relations of  $P_2$  with one argument involved in an Ord operation of (2) we have two types: with e.i

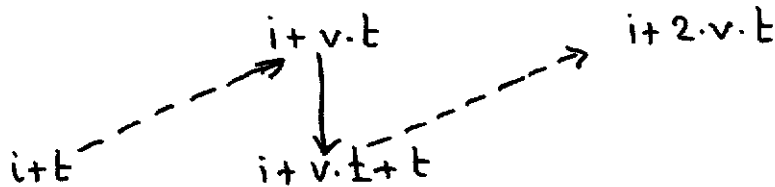


neither of which, however, is a lemma. But

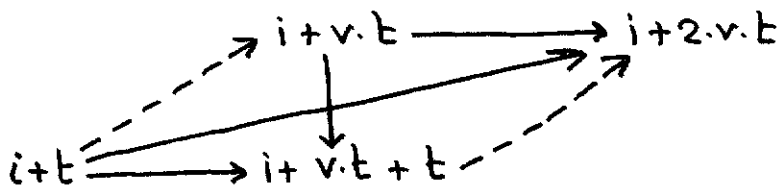


represent Lemma 3. Note that,  $v$  being even, the two OK relations added are implied by  $P_1$ . (Here we are getting our first glimpse of how to exploit precondition  $P_1$ .)

(iii) Finally we investigate the OK relations of  $P_2$  with both arguments involved in an Ord operation of (2): with e.i



This, again, is not a lemma, but we can recognize the sequence  $\dashrightarrow \rightarrow \dashrightarrow$  in Lemma 4



Two of the added OK relations are again implied by  $P_1$ . The third one

$$(\underline{A}i: e.i: \text{OK}.(i+t).(i+2 \cdot v \cdot t))$$

tells us that (2), which eo ipso establishes

$$P_4: (\underline{A}i: e.i: \text{OK}.(i+t).(i+v \cdot t)) ,$$

when preceded by  $v := v/2$  maintains -besides  $P_2$ - $P_4$  as well. As  $P_4 \wedge v=2 \Rightarrow P_3$ ,

$$P_4 \wedge v \geq 2 \wedge v \text{ is a power of } 2$$

is the proper invariant under which repeated execution of (2) can establish  $P_3$  without falsifying  $P_2$ ; it can be initialized by establishing  $v \cdot t \geq N$ . Under the as yet unverified assumption of  $P_1$ 's validity in the precondition of (2) we would get the following text for Batchner's Bafter:

```

]] [t, v0: int; t:=1; do t < N → t := 2 * t od; v0 := 1 { P0 ∧ t · v0 ≥ N }
; do t ≠ 1 → t, v0 := t/2, 2 * v0 { P1 ∧ t · v0 ≥ N }
; (]] i: e.i: Ord.i.(i+t)) { P1 ∧ P2 ∧ t · v0 ≥ N }
; ]] [v: int; v := v0 { P1 ∧ P2 ∧ P4 }
; do v ≠ 2 → v := v/2 { P1 ∧ P2 ∧ P42·v }
; (]] i: e.i: Ord.(i+t).(i + v · t)) { P1 ∧ P2 ∧ P4 }
od { P2 ∧ P4 ∧ v = 2 }
]] { P0 ∧ t · v0 ≥ N }
od { P0 ∧ t = 1 }
]] { (]] i: e.i: OK.i.(i+1)) } .

```

We are left with the obligation of verifying that neither (0) nor (2) falsifies

$$P_1: (\forall i :: OK.i.(i+2.t))$$

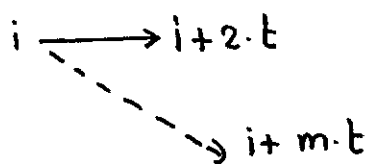
Operations (0) and (2) are of the form

$$(\exists i :: p.i: Ord.i.(i+m.t)) \quad (3)$$

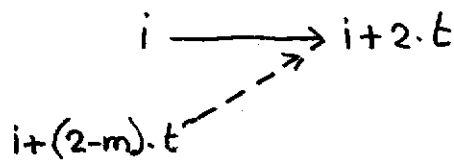
with  $p.i \equiv e.i$  or  $p.i \equiv \neg e.i$  and odd  $m$ . We shall verify the invariance of  $P_1$  under (3).

(i) The OK relations of  $P_1$  with no argument involved in an Ord operation of (3) are maintained on account of Lemma 0.

(ii) Of the OK relations of  $P_1$  with one argument involved in an Ord operation of (3) we have two types:



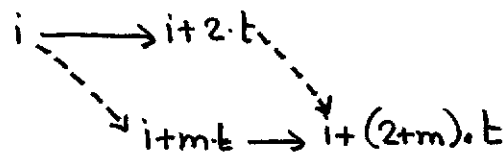
$$(i+(2+m).t > N)$$



$$(i-m.t < 0)$$

Fortunately they represent Lemma 1b and 1a respectively.

(iii) OK relations of  $P_1$  with both arguments involved in an Ord-operation of (3) are of the types:



This, fortunately, is Lemma 2, and thus  $P_1$ 's invariance under (3) has been demonstrated.

And this completes my treatment of Batcher's Baffler.

\* \* \*

This algorithm was invented by K.E. Batchier in 1968 and brought to my attention by David Gries, who invented its name and proved its correctness, in essence along the same lines but in a very different presentation. Besides considerable notational differences the main distinctions of this presentation are

- (i) the isolation of the Lemmata as in EWD932 b,
- (ii) the amount of heuristics included.

It is worth noting that Lemmata 0 through 4 of EWD932 b have all been used. Lemma 5 - which is more or less an extra - is followed in EWD932 b by precisely the sorting process that Batchier's Baffler would generate for  $N=4$ .

Finally I would like to draw attention to the considerable benefit we derived from the convention - introduced right at the beginning - of implicitly constraining the arguments of OK and similarly of Ord.

I am indebted to David Gries and to the Tuesday Afternoon Clubs of Eindhoven and of Austin.

Austin 18 September 1985

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