For the record: Batcher's Baffler

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

$$OK.i.j \equiv f.i \le f.j \quad ;$$

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

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

The functional specification of Batcher's Baffler is

|[N: int {N ≥ 0}
; |[f(i: 0 ≤ i < N): _array of_ int {bag.f = B}
; Batcher's Baffler
{bag.f = B ∧ (_A_ i :: OK.i.(i+1))}
]|
]| .

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

$$Ord.i.j = \underline{if}\ OK.i.j \rightarrow skip\ [\!]\ \neg OK.i.j \rightarrow swap.f.i.j\ \underline{fi}$$

— the swap interchanges the values of $f.i$ and $f.j$ — . The Ord operation satisfies {true} Ord.i.j {OK.i.j} ; quantifications over it are again implicitly constrained by

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

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

P0: $(\underline{A}i :: OK.i.(i+t))$

which suggests for Batcher's Baffler the form

"establish $t \geq N$" {P0}
; do $t \neq 1 \rightarrow$ "decrease t under invariance of P0" od .

Let the decrease of t under invariance of P0 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 P0 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 Batcher's Baffler gives explicitly the manipulations on t :

$t := 1$; do $t < N \rightarrow t := 2 \cdot t$ od {P0 $\wedge$ t is a power of 2}
do $t \neq 1 \rightarrow t := t/2$
; {P1: $(\underline{A}i :: OK.i.(i+2 \cdot t))$}
"restore P0"
{P0: $(\underline{A}i :: OK.i.(i+t))$}
od

The rest of this note is concerned with the algorithm for "restore P0" 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 P0 such that the argument pairs in each group are disjoint. This is achieved by writing P0 as $P2 \wedge P3$ with

P2: $(\underline{A}i: e.i: OK.i.(i+t))$ and

P3: $(\underline{A}i: \neg e.i: OK.i.(i+t))$,

provided we can find a predicate e such that

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

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

$$e.i \equiv (i \ \underline{mod}\ 2 \cdot t) < t \qquad .$$

<u>Remark</u>. 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 $\|$ to denote the potentially concurrent combination, we have indeed

$$\{true\}\ (\| i: e.i: Ord.i.(i+t))\ \{P2\} \qquad (0)$$

$$\{true\}\ (\|i : \neg e.i :\ Ord.i.(i+t))\ \{P3\} \qquad (1)$$

But we cannot achieve $P2 \wedge P3$ – i.e. $P0$ – 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 $P2$ by means of (0) and then establishing $P3$ with a repetition for which $P2$ is an invariant, i.e. we may expect that repetition to establish $P2 \wedge P3$ under invariance of $P2 \wedge P4$ where $P4$ is some generalization of $P3$.

<u>Remark</u> 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 $P3$ first. (End of Remark.)

For our analysis we rewrite – because it is slightly more convenient – $P3$ as

$$P3: \qquad (\underline{A}i : e.i :\ OK.(i+t).(i+2\cdot t))$$

and (1) as

$$(\|i : e.i :\ Ord.(i+t).(i+2\cdot t)) \qquad (1)$$

and ask ourselves under what further conditions the generalized operation

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

maintains $P2$. (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 P2 with no argument involved in an Ord operation of (2) are maintained on account of Lemma 0 .

(ii) Of the OK relations of P2 with one argument involved in an Ord operation of (2) we have two types: with e.i

$$\begin{tikzcd} i \arrow[d] & i+v\cdot t \\ i+t \arrow[ur, dashed] & \end{tikzcd} \qquad \begin{tikzcd} & i+v\cdot t \arrow[d] \\ i+t \arrow[ur, dashed] & i+v\cdot t+t \end{tikzcd}$$

neither of which, however, is a lemma. But

$$\begin{tikzcd} i \arrow[r] \arrow[d] & i+v\cdot t \\ i+t \arrow[ur, dashed] & \end{tikzcd} \qquad \begin{tikzcd} & i+v\cdot t \arrow[d] \\ i+t \arrow[ur, dashed] \arrow[r] & i+v\cdot t+t \end{tikzcd}$$

represent Lemma 3. Note that, v being even, the two OK relations added are implied by P1. (Here we are getting our first glimpse of how to exploit precondition P1.)

(iii) Finally we investigate the OK relations of P2 with both arguments involved in an Ord operation of (2): with e.i

$$\begin{tikzcd} & i+v\cdot t \arrow[d] & i+2\cdot v\cdot t \\ i+t \arrow[ur, dashed] & i+v\cdot t+t \arrow[ur, dashed] & \end{tikzcd}$$

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

$$\begin{tikzcd} & i+v\cdot t \arrow[r] \arrow[d] & i+2\cdot v\cdot t \\ i+t \arrow[ur, dashed] \arrow[r] \arrow[urr] & i+v\cdot t+t \arrow[ur, dashed] & \end{tikzcd} .$$

Two of the added OK relations are again implied by P1 . The third one

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

tells us that (2), which eo ipso establishes

P4: $(\underline{A}i: e.i: OK.(i+t).(i+v\cdot t))$,

when preceded by $v:=v/2$ maintains -besides P2- P4 as well. As $P4 \wedge v=2 \Rightarrow P3$,

$$P4 \wedge v \geq 2 \wedge v \text{ is a power of } 2$$

is the proper invariant under which repeated execution of (2) can establish P3 without falsifying P2 ; it can be initialized by establishing $v\cdot t \geq N$. Under the as yet unverified assumption of P1's validity in the precondition of (2) we would get the following text for Batcher's Baffler:

```
|[ 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 ∧ P4(v := 2·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}
]| {(Ai:: OK.i.(i+1))}    .
```

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

P1: $(\underline{A} i :: \mathrm{OK}.i.(i+2\cdot t)$.

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

$$(\| i :\ p.i :\ \mathrm{Ord}.i.(i+m\cdot t)) \qquad (3)$$

with $p.i \equiv e.i$ or $p.i \equiv \neg e.i$ and odd m . We shall verify the invariance of P1 under (3) .

(i) The OK relations of P1 with no argument involved in an Ord operation of (3) are maintained on account of Lemma 0 .

(ii) Of the OK relations of P1 with one argument involved in an Ord operation of (3) we have two types:

$$\begin{tikzcd} i \arrow[r] \arrow[dr, dashed] & i+2\cdot t \\ & i+m\cdot t \end{tikzcd} \qquad\qquad \begin{tikzcd} & i \arrow[r] & i+2\cdot t \\ i+(2-m)\cdot t \arrow[urr, dashed] & & \end{tikzcd}$$

$(i+(2+m)\cdot t > N)$ $\qquad\qquad$ $(i-m\cdot t < 0)$.

Fortunately they represent Lemma 1b and 1a respectively.

(iii) OK relations of P1 with both arguments involved in an Ord-relation of (3) are of the types:

$$\begin{tikzcd} i \arrow[r] \arrow[dr, dashed] & i+2\cdot t \arrow[dr, dashed] & \\ & i+m\cdot t \arrow[r] & i+(2+m)\cdot t \end{tikzcd}$$.

This, fortunately, is Lemma 2, and thus P1's invariance under (3) has been demonstrated.

And this completes my treatment of Batcher's Baffler.

* * *

This algorithm was invented by K.E. Batcher 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 Batcher'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 <u>implicitly</u> constraining the arguments of OK and similarly of Ord.

I am endebted 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