Elicited by Roland C. Backhouse

In [0], Roland Backhouse describes a design of a very nice algorithm for distributed sorting. We quote the problem specification.

> "Given is a sequence of bags B_i $(0 \leq i < N)$, and the problem is to design an array of processes, each of which stores one of the bags and which communicate by message-passing, to sort the bags in such a way that on termination
> (0) $\underline{A}(i: 0 < i < N: \max(B_{i-1}) \leq \min(B_i))$
> and such that the size of the bags remains constant throughout the computation."

This note is written a) to forward our comments to Roland Backhouse b) to experiment with yet another example in which a mathematical design can be driven quite completely by investigating the shape rather than an imagined interpretation of the emerging formulae c) because Backhouse's algorithm can be presented in a somewhat more disentangled fashion. It begins with a description of the design and it ends with a brief discussion of those parts of Backhouse's text that brought about this little study.

*

We consider a finite nonempty sequence of machines. Each pair of neighbouring machines is connected by a channel along which the machines can communicate. For each channel a boolean function is defined and it depends

on the states of the two machines connected by the channel, and on nothing else. We aim at a computation of the network that upon termination has established relation

(0) $(\underline{A}k :: d.k)$;

here k ranges over the channels and $d.k$ is the boolean function for channel k.

Inspired by the shape of (0) we wish to view a computation of the network as a sequence of what we shall call channel activities. For a channel k such an activity comprises the evaluation of $d.k$ and, in case $d.k$ is false, a step towards termination. More precisely, using Chandy and Misra's notation, we choose

$$(\,[\!]\, k :: \neg d.k \rightarrow \text{step towards termination})$$

as our first approximation for a program that establishes (0).

In the case we want to save on superfluous channel activities, i.e. activities in which $d.k$ evaluates to true, we wish to have more explicit control on when to trigger channels. For that purpose we introduce a boolean variable $g.k$ for each channel k, and consider as our next approximation the program

$$(\,[\!]\, k :: g.k \rightarrow \text{step towards termination}\,),$$

establishing

(1) $\quad (\underline{A}k :: \neg g.k)$.

The question then is under what additional condition (0) still holds on termination. For such a condition we can choose any solution of the equation X: (1) $\wedge X \Rightarrow$ (0) . The weakest solution is (1) $\Rightarrow$ (0) but we prefer a solution that has the shape of (0) and (1), expressed in local assertions on the channels. Therefore we propose $(\underline{A}k :: \neg g.k \Rightarrow d.k)$, or --equivalently--

(2) $\quad (\underline{A}k :: g.k \vee d.k)$.

We will see to the validity of (2) upon termination by choosing it to be an invariant of the computation. It can be established by

(3) $\quad (\Box k :: g.k := \text{true})$.

Next we investigate a step towards termination. In view of (1) the simplest possibility for making progress is, for channel k,

$$g.k := \text{false} .$$

Since $g.k$ is a precondition of the step it decreases the value of function v defined by

$$v = (\underline{N}k :: g.k) .$$

For it to maintain the invariant (2), its precon-

dition should satisfy d.k . Hence for the step of channel k we choose a program of the form

(4)
```
if d.k  →  g.k := false
[] ¬d.k →  S.k
fi .
```

In order to guarantee progress towards termination for the second alternative as well, we shall see to it that

(5) for each k, S.k decreases a function u that is bounded from below and that is not increased by the first alternative,

so that each step lexically decreases the pair (u, v) (, which is bounded from below).

In the above program, statement S.k is the only statement that enables us to influence the value of d . Next we investigate, for the sake of maintaining invariant (2), which d-values it may influence. In our ultimate solution S.k will be a combined activity of the two machines adjacent to channel k. Because the d-value of a channel depends on the states of its adjacent machines, this combined activity may influence the d-value of each channel that is incident to one of the two machines. Because our network is a linear arrangement of channels and machines, S.k can only influence the d-values of the three successive channels i, k, and j . Therefore, for the sake of maintaing the invariant, S.k

has to maintain the conjunction of

(6i) $g.i \vee d.i$
(6k) $g.k \vee d.k$
(6j) $g.j \vee d.j$.

Remark The above conjunction can always be established by

(7) $g.i := true \;[\!]\; g.k := true \;[\!]\; g.j := true$,

and in case we do not want to exploit any additional knowledge about d, there is hardly any other choice.
(End of Remark .)

+

Next we do justice to the desire that a channel activity be distributed over the two adjacent machines. More specifically, this means that we have to distribute (3), (4), and (7) over these machines. They comprise an inspection of the value of $d.k$ and some assignments to some g's. Because thus far the discussion has been symmetric in the two sides of channel k we shall see to it that the distribution does not destroy that symmetry.

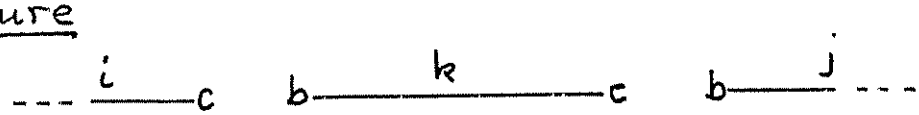
We investigate the partitioning of the auxiliary booleans g first. The simplest symmetric partitioning is obtained by representing g as $g \equiv (b \,?\, c)$ for some symmetric boolean operator $?$, with the understanding that b is a boolean that can be manipulated and inspected

only at the one side of the channel -- we call it the b-side -- and c is a boolean that can be manipulated and inspected only at the other side of the channel -- we call it the c-side --. From the thus far optional (7) we conclude that the machines engaged in the activity of channel k should be enabled to effectuate an assignment g := true for an adjacent channel. More specifically, ? has to be such that g := true can be effectuated independently at either side of a channel. This does not leave much room for choice:

(8) $\quad (\underline{A}k :: g.k \equiv b.k \vee c.k)$

Next we decide that, for the sake of symmetry of the machines, each machine is at the b-side of one of its adjacent channels and at the c-side of the other.

Figure



(End of Figure.)

The assignments to g in terms of b and c are

g.k := true	(in (3)) :	b.k := true ‖ c.k := true
g.i := true	(in (7)) :	c.i := true
g.k := true	(in (7)) :	b.k := true ‖ c.k := true
g.j := true	(in (7)) :	b.j := true
g.k := false	(in (4)) :	b.k := false ‖ c.k := false

So much for the partitioning of g.

Now for the partitioning of d. Because, in general, d is an arbitrary function of the states of two adjacent machines, we propose that prior to the execution of (4) these machines engage in a communication revealing the value of d. So much for the partitioning of d.

(Apart from distribution of the "data" and the operations on them, we will probably also have to distribute the "sequencing", such as the beginning and the end of a channel activity. This being a separable concern, we have chosen not to elaborate on it in this treatment of the algorithm.)

This concludes our treatment of a program that establishes (0).

*

Now we turn our attention to the specificities of Backhouse's algorithm. Our remaining obligations are to see to it that

a) the value of d.k is communicated
b) S.k is chosen such that it satisfies (5)
c) S.k is chosen such that it maintains (6).

In Backhouse's scenario each machine owns a finite nonempty bag of integers. For each bag two functions p and q are defined that each select an element from the bag. (One may think of functions like max, min, median.) With

machine $\mathcal{B}$ at the b-side of channel k, and machine C at the c-side. Backhouse's choice for $d.k$ is

(9) $\quad d.k \equiv p.\mathcal{B} \leq q.C$.

Next we address the remaining obligations a), b), and c) in turn.

Ad a For the communication of the value of $d.k$ we introduce two fresh variables $h.\mathcal{B}$ and $h.C$. The communication establishes

(10) $\quad h.\mathcal{B} = q.C \;\wedge\; h.C = p.\mathcal{B}$.

As a result $d.k$ can be expressed in the local nomenclature of machine $\mathcal{B}$ as $p.\mathcal{B} \leq h.\mathcal{B}$ and in machine C as $h.C \leq q.C$.
(End of Ad a.)

Ad b A precondition of $S.k$ is {see (4)} $\neg d.k$, i.e. $p.\mathcal{B} > q.C$. Therefore the bag adjustments

$$bag.\mathcal{B} := bag.\mathcal{B} + [q.C] - [p.\mathcal{B}]$$
$$, bag.C := bag.C + [p.\mathcal{B}] - [q.C]$$

decrease the number of inversions. (Here we use that p and q select bag elements and we also use that the network is a sequence of machines for which the function "number of inversions" is properly defined. We do not treat this in more detail.) With this choice (5) is satisfied because in the first alternative of (4) no bag manipulations occur.
Again, by (10), the bag adjustments can be expressed in the local terminology of each machine.
(End of Ad b.)

Ad c Backhouse chooses to maintain (6) by establishing (6i) and (6j) in the "easy way", viz. by $g.i := true \;[]\; g.j := true$, and (6k) in a more subtle way by exploiting specificities of d, with which we deal now. Condition (6k) is {see (8) and (9)}

(11) $\quad b.k \vee c.k \vee p.B \leq q.C$.

which is to be established by $S.k$. A precondition of $S.k$ is {see (4)} $\neg d.k$, i.e. {see (9)} $p.B > q.C$. I.e. {see (10)}

(12) $\quad h.C > h.B$ is a precondition of $S.k$.

The bag adjustments in $S.k$ may truthify the third disjunct of (11). Therefore we are interested in preferably strong -- stronger than true -- expressions EB and EC satisfying

(13) $\quad EB \vee EC \vee p.B \leq q.C$.

because then $S.k$ can establish (11) by

$$b.k := EB \;[]\; c.k := EC$$

(, provided, of course, that EB and EC are expressible in the local nomenclatures of machines B and C respectively). In our search for EB and EC, (13) leads us to investigating $p.B > q.C$: we have

$$\begin{array}{ll} & p.B > q.C \\ \Rightarrow & \quad \{\text{transitivity}\} \end{array}$$

$$\begin{aligned} & p.B > h.B \;\vee\; h.B > h.C \;\vee\; h.C > q.C \\ = & \quad \{ \text{by (12) the middle disjunct is false} \} \\ & p.B > h.B \;\vee\; h.C > q.C \end{aligned}$$

Hence, adequate choices for EB and EC are $p.B > h.B$ and $h.C > q.C$ respectively. (The expressions found can indeed be stronger than true: if, for instance, B owns a one-element bag, p.B and h.B, both being bag elements, are equal.)

(End of Ad c.)

This concludes our treatment of what we believe to be Backhouse's distributed sorting algorithm.

*

Backhouse, in his design, is guided by more or less operational concepts, such as "estimates" and their "inaccuracies", "uncertainty", and processes that "have complete information". In our opinion such concepts have too little technical precision to assist in the design of an algorithm. Therefore we wanted to investigate how much of the work could be done by just regarding the shape of the emerging formulae.

Another reason for us to undertake this little study was our impression that in [0] the symmetry between the two sides of a channel is not displayed in a sufficiently satisfactory way.

A third reason was our observation that in many a part of Backhouse's treatise the internal

structure of formulae was irrelevant for the manipulations they were subjected to. This suggested that there could be room for a more disentangled presentation.

Eindhoven, April 1987

A.J.M. van Gasteren
&
W.H.J. Feijen

[0] Distributed Sorting,
Roland C. Backhouse
(Private Communication)