Reducing control traffic in a distributed implementation of mutual exclusion

We consider a network of machines that can send messages to each other. Each machine is in 1 of 3 states, viz.

n	for "neutrally engaged",
d	for "delayed", or
c	for "critically engaged" .

A critical engagement lasts only a finite period and is immediately followed by a neutral engagement of the machine in question. Between a neutral and the subsequent critical engagement a delay may occur in view of the requirement that at any moment at most 1 machine be critically engaged (so-called "mutual exclusion"). The implied synchronization has to be implemented in such a manner that no delay lasts forever (so-called "fairness").

We introduce a single <u>token</u>, either held by one of the machines or being sent from one machine to another. Mutual exclusion is then achieved by maintaining

a critically engaged machine holds the token .

The machines maintain this by (i) not initiating a critical engagement unless holding the token, and (ii) not sending the token to another machine while being critically engaged.

Furthermore each machine maintains

the machine holding the token is not delayed

by (i) skipping the delay upon termination of a neutral engagement while holding the token, and (ii) initiating a critical engagement upon receipt of the token while delayed. Fairness is therefore ensured when each delayed machine receives the token within a finite period of time.

The rest of this note deals with the control of the movement of the token. To this end the machines are arranged in a ring, the two circular directions in which are called "to the left" and "to the right" respectively. The token is sent to the left, so-called <u>signals</u> are sent to the right. Each link connecting two neighbouring machines in the ring is in 1 of 3 states, viz.

u	for "unused"
t	for "carrying the token to the left"
s	for "carrying a signal to the right" .

The latter two states are postulated to last only a finite period of time.

The idea is that a delayed machine can send a signal to the right to call the token to the left. In order to mark ring segments traversed to the right by signals but not yet traversed to the left by the token, machines on such segments will be <u>black</u>; the others will be <u>white</u>. We shall now make this precise by developing the possible states of the ring, and begin by describing the coding conventions we shall use for the description of these ring states.

The state of a link will be coded by u, t or s. The state of a machine will be coded by the colour b or w, followed by n, d, or c; the machine holding the token is identified by writing its colour with a capital letter. The state of the ring is described by a string of the appropriate length in which machine states and link states alternate; the direction "to the right" in the string coincides with the circular direction "to the right" in the ring.

Strings are given by so-called "regular expressions". They enable us to describe the initial state by

$$Wn\ u\ (wn\ u)^* \qquad , \qquad (0)$$

i.e. a white neutral machine holding the token, an unused link, followed by zero or more occurrences — that is what the star stands for — of a white neutral machine not holding the token followed by an unused link. The above regular expression expresses: all machines are white and neutrally engaged, all links are unused and the token resides at some machine.

My intention is to build up the invariant as the transitive closure of the successor relation, hand in hand considering, as the need arises, more (local!) successor relations and the resulting extensions of the grammar of reachable states. We shall mark the successor relations since they have all to be catered for in the final algorithm.

In order to find with the program its strongest invariant we propose to proceed as follows. To begin with we assume that machine states n and c and link states s and t last forever, assumptions that will be discarded one by one. Each time we shall extend the regular expression, i.e. weaken the invariant, so as to cover precisely the states reachable by means of the transitions introduced in the mean time. The "passive" states d and u —in contrast to the other four— are treated as not terminating "spontaneously".

With n, c, s, and t lasting forever, the initial state (0) is invariant. The first thing to do is obviously to take possible termination of n into account, and (0) tells us that two transitions have to be introduced. We have already decided on

(n.0) $\quad Wn \rightarrow Wc \quad$;

for the other we propose

(n.1) $\quad wn\ u \rightarrow wd\ s \quad$.

Their transitive closure yields —with a rotation so as to have delayed machines to the left of the token—

(1) $\quad (wd\ s\ (wn\ u)^*)^*\ (Wn | Wc)\ u\ (wn\ u)^*$

in which, as usual with regular expressions, the "|" is used to separate alternatives. (It will be given a lower binding power than the concatenation which is indicated by a blank.)

From (1) we see that termination of s is the next event to be taken into account; it confronts us with four cases, and we propose

(s.0)	s wn u	$\rightarrow$	u bn s
(s.1)	s wd	$\rightarrow$	u bd
(s.2)	s Wn	$\rightarrow$	t wn
(s.3)	s Wc	$\rightarrow$	u Bc

In order to determine the new transitive closure, we first consider (s.0), yielding

(2) $(\text{wd } (\text{u bn})^* \text{s} (\text{wn u})^*)^* \, (\text{Wn}|\text{Wc}) \, \text{u} \, (\text{wn u})^*$,

at which stage a new circumstance has been introduced under which n may terminate. To cope with it we propose the transition

(n.2) $\quad$ bn $\rightarrow$ bd ,

the transitive closure of which yields from (2)

(3) $\text{E } (\text{Wn}|\text{Wc}) \text{ u } (\text{wn u})^*$ $\quad$ with

$\text{E} = (\text{wd } (\text{u } (\text{bn}|\text{bd}))^* \text{ s } (\text{wn u})^*)^*$

thereby not introducing new circumstances for termination of s to be considered. Transition (s.1) may occur within an E of the form

$$E \;\; wd\;(u\,(bn|bd))^*s\;\; wd\;\;(u\,(bn|bd))^*\;s\;\;(wn\;u)^*\,E,$$

which (s.1) transforms into

$$E \;\; wd\;(u\,(bn|bd))^*\;u\;\;bd\;\;(u\,(bn|bd))^*\;s\;\;(wn\;u)^*\,E,$$

which is a special instance of E itself: transition (s.1) does therefore not introduce new reachable states.

Transitions (s.2) and (s.3) can occur within instances of (3) of the form

$$E\;\;wd\,(u\,(bn|bd))^*s\;(Wn\,|\,Wc)\;u\;(wn\;u)^*$$

yielding

$$E\;\;wd\,(u\,(bn|bd))^*\;(t\;\;wn\;\;u\,|\,u\;Bc\;u\,)\,(wn\;\;u)^*\quad .$$

Thanks to (n.1) and the form of E this can lead to

$$E\;\;wd\;\;(u\,(bn|bd))^*\;(t\,|\,u\;Bc\;\;u\,)\,(wn\;\;u\,)^*$$

which extends (3) to the reachable states

(4) $E\,((Wn|Wc)\,u\;|\;wd\,(u\,(bn|bd))^*(t\,|\,u\,Bc\,u))\;\;(wn\,u)^*$

which is now closed as far as the considered terminations of n and s are concerned.

Now we take termination of c into account; (4) tells us that we have two cases to consider and that the transitions

(c.0) $\qquad Wc \;\rightarrow\; Wn$

(c.1) $\qquad u\;\;Bc \;\rightarrow\; t\;\;wn$

do not lead to new reachable states.

Finally we take termination of t into account; (4) presents three cases to be considered, for which we propose the transitions

(t.0)	bd t $\rightarrow$ Bc u	
(t.1)	u bn t $\rightarrow$ t wn u	
(t.2)	wd t $\rightarrow$ Wc u	,

none of which leads to new reachable states. Hence (4) is invariant and by virtue of its construction it is the strongest invariant. It allows obviously desirable conclusions such as

the recipient of a signal is white
the recipient of the token is black <u>or</u> is delayed
the machine holding the token is not delayed
a critically engaged machine holds the token
a neutrally engaged black machine does not
hold the token.

For fairness, take a delayed machine and consider the string of links and machines to its right up to and including the link carrying or the machine holding the token. Let

k= the length of the string + the number of white machines in the string .

Obviously, $k \geq 0$. We observe
(i) -see (4)- the string contains at least 1 s, c, or t ;

(ii) – see (4) – if ending with (Wn|Wc), the string contains at least one s ;
(iii) all states s, c, or t are of finite duration;
(iv) all terminations of these states decrease k, except transition (c.0): Wc → Wn .

From the above we conclude that no delay lasts forever.

From the above it follows that the following four atomic actions within each machine implement our mutual exclusion algorithm.

```
upon completion of a neutral engagement:
if holding the token → {n.0} initiate critical engagement
▯ ¬ holding the token ∧ white → {n.1} send signal to
                  the right and initiate delay
▯ ¬ holding the token ∧ black → {n.2} initiate delay
fi

upon receipt of a signal:
if ¬ holding the token ∧ neutrally engaged → {s.0} be-
                  come black and send signal to the right
▯ holding the token ∧ neutrally engaged → {s.2}
                  send the token to the left
▯ ¬ neutrally engaged → {s.1, s.3} become black
fi

upon completion of a critical engagement:
if white → {c.0} initiate neutral engagement
▯ black → {c.1} become white, send token to the left
                  and initiate neutral engagement
fi
```

```
upon receipt of the token
if delayed → {t.0, t.2} initiate critical engagement
▯ neutrally engaged → become white and send the
                       token to the left
fi .
```

Remark. Since the above code for the four events contains nothing new, my decision to include it in this text is questionable. (End of Remark.)

In retrospect

The above presentation does not reflect at all the way in which I discoverd this algorithm, the possibility of which I saw in a flash. The flash must have been helped by the circumstance, which I remembered only a few days later, that Alain J. Martin [1] had shown me the same algorithm —be it coded in a form of Hoare's Communicating Sequential Processes— five years ago.

The first time I wrote it down I had not bothered about formulating the invariant and —as was to be expected— my program contained an error. The second version explored the idea of developing the invariant formulated by means of regular expressions; I "developed" the transitive closure without a clear picture of the proof obligations and —as was to be expected— my "invariant" was wrong. In this version I somewhat belatedly included the fairness argument; my first effort, which I failed to check carefully, was wrong.

Just giving invariant (4), listing the transitions, showing the completeness of that list and demonstrating, one by one, that the transitions maintain (4) could have led to a much shorter text; it would, however, fail to argue that (4) is the strongest invariant, i.e. that all ring states it permits can indeed occur and each transition catered for by the algorithm may indeed be required.

The fact that the same set of ring states could be characterized by many different regular expressions annoyed me at first because, for lack of experience, I had little guidance what to do with that freedom. I suspect, however, such freedom to be an essential ingredient of any notation on which a calculus can be based: it allows one to massage an expression into different forms suitable for different substitutions.

My current estimation is that regular expressions are a suitable tool for the problem at hand. Note, for instance, that there was no need to number the machines from 0 through N-1: we could do without all the subscripts that quantification would have required. Note also that it was a fortunate decision to introduce in the characterizing string separate elements for machines and for links; it gave us more freedom —(0) could equivalently have been written as $u\ Wn\ (u\ wn)^*$ — but without the freedom to pair a link either with the preceding or with the following machine, our invariant would have been much more clumsy to formulate.

There was another reason why I developed the algorithm in the way I did, i.e. by taking possible events into account one at a time, each time proposing an action and, if necessary, an adjustment of the invariant. I used this technique last year for the development of a new distributed algorithm for the detection of termination and the technique could be more generally applicable.

Acknowledgements

I express my gratitude to K. Mani Chandy [0] for drawing my attention to the problem, my respect for Alain J. Martin [1] for having found this solution 5 years ago, and to the members of the Tuesday Afternoon Club for their comments, in particular to F.E.J. Kruseman Aretz for reminding me of my proof obligations.

References

[0] Chandy, K.M., Private Communication (1 March 1983)

[1] Martin, A.J., Private Communication (Spring 1978)

[2] Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System" Comm. ACM, Vol. 21, No. 7 (July 1978) 558-565

[3] Ricart, G. and Agrawala, A.K., "An Optimal Algorithm for Mutual Exclusion in Computer Networks", Comm. ACM, Vol. 24, No. 1 (Jan 1981) 9-17

[4] Carvalho, O.S.F. and Roucairol, G. "On Mutual Exclusion in Computer Networks", Comm. ACM, Vol. 26, No. 2 (Feb. 1983) 146-148

Plataanstraat 5
5671 AL NUENEN
The Netherlands

25 March 1983
prof. dr. Edsger W. Dijkstra
Burroughs Research Fellow