More on A.J.Martin's design. (A sequel to EWD668)

This is a sequel to EWD668 "On the correctness of a design by Alain J.Martin" in the sense that we use his problem, and modify his solution, in a reconsideration of the role of fair daemons for the purpose of the prevention of individual starvation. We use here the same configuration of customer mosquitoes M , each coupled to its private service mosquito m , and the service mosquitoes again arranged in a ring, in which each refers to its neighbours as L and R . Omitting queries and shrieks, each customer mosquito can be written as before:

M: do true → noncritical section; m; m; critical section; m od

By way of experiment we write our service mosquitoes in Hehner's style as a (nonterminating) semi-recursion of two refinements "giv" and "rec", with the understanding that only one service mosquito is initialized with "giv" and all others are initialized with "rec". Let us first consider solutions in which, when none of the customer mosquitoes is interested in entering its critical section, the service mosquitoes continue frantically transmitting the privilege to their left-hand neighbours:

giv: if L → rec [] M → M; M; L; rec fi (1)
rec: R → giv

Without assumption of a fair daemon, this "solution" is full of the danger of individual starvation: in the state giv , when a service mosquito is ready to admit its customer to the critical section, it is also free not to do so, and to transmit the privilege to its left-hand neighbour. How much better than (1) is (2) in this respect?

giv: L; rec (2)
rec: if R → giv [] M → R; M; M; giv fi

At first sight this solution is much better: as long as an m-mosquito's service is not requested by its M-mosquito, it is "almost always" in the rec-state, ready to honour the call for attention from its M-mosquito. From a practical point of view (2) might be acceptable, in general (2) is almost as unsatisfactory as (1) in the sense that the absence of individual starvation relies on a guarantee of local progress of m-mosqui-

toes. If an m-mosquito is allowed to go to sleep in state rec , and is only woken up when its right-hand neighbour is ready to communicate with it and that communication is unfairly chosen each time, individual starvation may occur as before. The only way of exorcizing the danger of individual starvation without the assumption of fair daemons requires that an m-mosquito may not only react on the presence, but also on the absence of a request from its M-mosquito. Then we could write

```
giv:  L; rec                                                    (3)
rec:  R; if M → M; M [] non M → skip fi; giv
```

with the understanding that the alternative construct never gives rise to delay or abortion: either the M-mosquito is ready to communicate or it is not, and only in the latter case the second alternative will be chosen.

The next question is: is it possible, after this linguistic extension, to change this solution (without re-introducing the danger of individual starvation) in such a way that, when no M-mosquito requires service, the ring of m-mosquitoes comes to rest? (This is the usual problem of avoiding the busy form of waiting, which is of interest if the m-mosquitoes are programs sharing a processor with others.) I could do it in the following manner, in which --in order to prevent one M-mosquito with an "empty" noncritical section from monopolizing the access right-- an m-mosquito that has served its customer transmits the privilege to its left-hand neighbour, whether it has asked for it or not. Solution (4) is a modification of A.J.Martin's solution discussed in EWD668. The commands between the m-mosquitoes are of two kinds, labelled ".a" and ".b" respectively.

```
giv:  if L.a → test and transmit                                (4)
       [] M → serve and transmit
      fi
rec:  if L.a → R.a; R.a; test and transmit
       [] M → R.a; R.a; serve and transmit
       [] R.b → giv
      fi

test and transmit:   if M → M; M [] non M → skip fi; L.a; rec
serve and transmit:  M; M; if L.a → L.a [] L.b → skip fi; rec
```

Note that, because R.b and L.b occur both in guard positions, the addition of queries and shrieks would make a shriek in a "guarding communication command" unavoidable.

* * *

In a way I liked the solution discussed in EWD668, and I was pleased to discover, how the assumption of a fair daemon --which is a rather operational concept-- could be translated in a clear pattern for a proof obligation. On the other hand I am still a little bit afraid of nondeterminacy with an unknown, but finite bound, because I foresee problems when we try to define the semantics more explicitly than just by a set of proof obligations. Hence my desire to explore, whether or not I could modify A.J. Martin's solution so as to make it independent of a daemon's fairness. I was pleased with the discovery that it could be done, although I think the price of two linguistic extensions --both of which seem necessary-- rather high, as in particular the last one may have unpleasant properties. (If I remember it correctly, both extensions fall outside the scope of the synchronization facilities considered in the GREEN Language for the DoD.)

The above has been written partly in preparation for tomorrow's meeting of the Tuesday Afternoon Club, to which --and in particular to Joseph M.Morris-- I am indebted for prompting me to this exploration.

* * *

The day after the above had been written, Alain J.Martin saw the text and remarked that the above "both of which seem necessary" --8 lines above-- was unjustified, as our second language extension, i.e. the introduction of shrieks in guarding communication commands, could have been avoided after the first extension had been accepted. In a repetitive construct of the form:

do M? → S1 [] L? → S2 od

we relied in EWD668 on a fair daemon for the absence of the danger of individual starvation. With the possibility of explicit reaction on the absence of a request for communication, we could have coded instead:

```
do M? → S1; if L? → S2 [] non L? → skip fi
 [] L? → S2; if M? → S1 [] non M? → skip fi
od
```

This transformation, applied to A.J.Martin's solution as described in EWD668 would have made it starvation-free without relying on a daemon's fairness.

If we so desire we can regard the first version as an abbreviation of the second one or a similar one, which allows a bounded amount of "unfairness", such as

```
do M? → S1; if M? → S1 [] non M? → skip fi;
        if L? → S2 [] non L? → skip fi
 [] L? → S2; if L? → S2 [] non L? → skip fi;
        if M? → S1 [] non M? → skip fi
od
```

Both the argument in EWD668 and the above transformations are for me incentives to be less afraid of fair daemons than I used to be. I have now good hope that "daemons of bounded unfairness" turn out to be mathematically manageable without the need of committing ourselves to a value of the bound on their unfairness. (The conclusion, as drawn in EWD673, that well-founded sets don't seem to be really necessary points in the same direction.)

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
BURROUGHS Research Fellow