Re: "Formal Derivation of Strongly Correct Parallel Programs" by Axel van Lamsweerde and M.Sintzoff.

This paper leaves me with very mixed feelings. I have the greatest sympathy for its goal of applying formal techniques at the stage of program construction rather than at the stage of verification of "given" programs; besides that I have not found any formal error, although I went fairly carefully through considerable portions of the text. Yet I would hate to see this article published in its current form, because in many respects it is so ugly that its publication would fire back on the goals it seeks to promote.

* * *

The paper is much too long, and as it stands no one is going to read it. It is a rambling paper in the sense that it deals with a number of very different problems. I don't feel that the "unified framework" alluded to in the summary justifies their combination into a single paper. My first recommendation to the authors is to consider to base upon this material, say, three separate articles.

Fairly isolated problems are

a) the correct and complete signalling; this is an optimization problem, and its isolation could be considered because only here the notational apparatus of ask and see and the strongest post-condition are needed

b) deadlock prevention; this is a logical problem that, in principle, admits a unique solution

c) starvation prevention; this is in general a strategic problem that admits infinitely many different solutions.

* * *

It is also a rambling paper in the sense that it is unclear in its objectives: do the authors want to present a set of workable techniques for program construction, or do they want to show that most predicates one is interested in during program construction can be viewed as extremal solutions of derivable equations? Maybe the authors' answer is "Both goals are equally dear to our hearts." Even if this is true --the article makes upon me the impression that the second goal is dearer to their hearts, and "sell" the topic by overstating the degree in which the first goal has been reached-- it is questionable whether a single article should hit at both targets, because many readers interested in the one goal don't care too much about the other, and vice versa.

* * *

The prose should be written more carefully, for as it stands it contains too many shaky expressions. I mention: "to statically abstract", "statically omitted", "temporarily delayed", "logical properties", "logical starvation" --these expressions are not made any clearer by typing "logical" in italics!-- "predicates which verify" (pg.12) instead of "satisfy", "fixed point equation" (pg.14) instead of "equation", "fixed point condition" (pg.36) instead of "condition", "fixed point solution" (pg.30) instead of "solution", "(global) deadlock-free invariant J" (pg.27) instead of "an invariant J ensuring absence of (global) deadlock".

* * *

I was very much amazed when I read (pg.55): "We must confess that our techniques [...] are not [...] based upon a formal model of parallel computation", amazed because I had been struck all the time by the extremely operational approach. (Each waiting process is "waiting on a queue": while reading I was

associating with each non-rectangular triangle a queue on which Pythagoras's Theorem could wait to become true.) After re-reading the quoted sentence I realized that I had skipped the word "formal". The quoted sentence continues "perhaps this should be done.". Personally I doubt that: I suspect that we can only make real progress provided we define the semantics independent of any underlying computational model.

In this article I find a confirmation of that suspicion. What are the "parallel programs" referred to in its title? What we are --or: should be-- talking about is programs whose implementations allow concurrency. It are not the programs themselves that are "parallel": parallelism or concurrency are operational concepts that refer not to the program, but to its execution. In this respect that paper is half-hearted: on pg.1 they write, "if all components of the parallel program can be executed an arbitrary number of times, they do what the specifications require them to do" but also "good proof techniques [...] should abstract from all these execution paths". But if you really do a good job of that last abstraction, the operational view would disappear (as would the word "parallel" from the title, a word whose presence is already now somewhat hard to defend).

* * *

Notation, terminology and formalism are often clumsy, e.g.

a) They use "I" for identification of an invariant relation: one should never do so in English texts --what about "I cannot be violated"?-- , nor in texts that also use Roman numerals, see pg.7, line 5 from below:

"according to rules based on the truth of I, see Appendix I."

b) Sections 2, 3, and 4 are done with double (quadruple) subscripts, only thereafter the authors simplify. (On page 9, line 11 from below, read: "programs of the simpler form"; this was the only typing error I saw! My compliments!)

c) What they call "local deadlock" should have been called "(danger of) collective starvation", because "deadlock" suggests something irreversible, and "local deadlock" should remain reserved for the situation --not studied in this paper-- that a true subset of the processes gets irrevocably stuck, no matter how the remaining processes are thereafter executed.

d) They often "unfold and fold again", but in all cases that I checked it, these operations were not necessary at all, e.g. in the derivation of (6.3), where the explicit mentioning of the use of the formula pg.30, line 4 from above, would have been more helpful.

* * *

From page 26 onwards, where starvation enters the picture, the treatment becomes increasingly unconvincing, because the danger of individual starvation can be exorcized in so many different ways. If we formulate the requirement operationally as a restriction defining a subset of the possible execution paths, I prefer the introduction of (perhaps ghost-) variables recording what is relevant about the path (or: "the past") so that we can formulate our restriction in terms of (original and) added variables. This being added to the relation to be kept invariant, the problem is reduced to the previous one, viz. the prevention of deadlock. That seems a more "unified" approach, which has the further advantage of making one's design freedom explicit.

Edsger W. Dijkstra