In reply to comments

Dear Colleagues,

since your comments are not disjoint, allow me to address you in principle collectively.

Of course there is more to digital system design than the formal derivation of programs from their equally formal specifications; accordingly I expect a full-blown CS curriculum to comprise more than "an introductory programming course for freshmen". Please remember that my modest proposal only pertained to the latter.

The choice of functional specifications – and of the notation to write them down in – may be far from obvious, but their rôle is clear: it is to act as a logical firewall between two different concerns. The one is the "pleasantness problem", i.e., the question of whether an engine meeting the specification is the engine we would like to have; the other one is the "correctness problem", i.e., the question of how to design an engine meeting the specification. I firmly believe that whenever we succeed in erecting such a firewall, the effort will pay off handsomely. The reason for this belief of mine is that the two concerns deserve separation because the two problems are most effectively tackled by totally different techniques.

(They are currently psychology and experimentation for the pleasantness problem and symbol manipulation for the correctness problem.)

The above, I think, is noncontroversial. The issue seems rather to be whether the above observation justifies the proposal to focus an introductory programming course on formal program derivation, while in the software industry the presupposed separation of concerns is rarely carried through (and perhaps never will be.)

Before rushing to a conclusion, I would like you to be aware of two facts, one obvious and one shocking. The obvious fact is that, after sufficient analysis, even the software industry is full of programming problems that don't defy functional specification at all. The shocking fact is that the majority of computing professionals —be they academic or industrial— cannot be trusted to program things as simple as a binary search. (To which I should add that they are usually thrilled when you show them how crisp an argument can maintain complete control over the derivation. But to have it sink in takes more than a single lecture: after a lecture, it is a common experience to receive from a fellow professor who was in the audience for one of the programs I derived a suggested "improvement", justified in

the usual handwaving fashion, but invariably in error.) Since formal derivation is by far the simplest and most effective way of constructing such programs and the techniques involved are eminently teachable (and fun to apply) I don't think we should deny them to our students. And I trust the better ones to apply such techniques on a more grandiose scale as they see fit.

While writing the above I observe that I am unable to consider the last conclusion under dispute: no responsible scientist that I know of guesses formulae of any sophistication, he derives them instead. If we agree - if necessary only for the sake of the argument - that the techniques of program derivation should be part of the computing professional's intellectual bagage, the only remaining question is: should it be covered in the first course on programming?

My answer is "Yes" because in my perception the hardest part of the educational process is not the acquiring of new habits but the getting rid of the old, inadequate ones: our past is our inalienable treasure or burden. I therefore see no justification in not teaching the most effective techniques available right from the start.

Several of you have voiced a wider concern, viz. serious doubts about the viability of a much more formalized mathematics. I can understand your concern because, for many years, I shared your doubts. Over the last decade, however, those doubts have largely evaporated, to the extent that I expect, say fifty years from now, the mathematical informality of today definitely to be a thing of the past. (I realize that this is a somewhat gratuitous statement because, if my expectation will not be fulfilled, I won't be there to be confronted with my mistake. But I cannot help having expectations beyond my lifetime.) The expectations are based on the experiences and observations collected since I started to explore the extent to which the experience gathered in programming methodology could be transferred to mathematical methodology in general. I shall indicate them briefly.

(i) The presumed dogma that calculational proofs are an order of magnitude too long to be practical has not been confirmed by my experience. On the contrary, well-chosen formalisms provide a shorthand with which no verbal rendering can compete.

(ii) Through most of this century, mathematical logic has primarily been used for soul-searching.

As a tool for daily, practical proof design it has hardly been given a fair chance. To realize its potential, it seems essential to view the purpose of logic not as mimicking human reasoning but as providing a calculational alternative to it.

(iii) In proof design, strong heuristic guidance can be extracted from a syntactic analysis of the theorem and from (baby) proof theory. Both possibilities require formalization of the proof structure for their exploitation.

(iv) While informal mathematics, with its ties to elusive entities such as "intuition" and "the human mind", is a hard topic to teach explicitly, symbol manipulation is tangible. The effective techniques of symbol manipulation are well within the teachable domain. Hence my estimate of only fifty years.

The possible future of mathematics that I envisage is very different from what the dyed-in-the-wool member of the mathematical guild of today is used and probably attached to: my dream could very well be his nightmare. So be it. I cannot consider such discrepancy my fault. If you have a technical argument why my expectation of the future of mathematics cannot come true, please let me know, for it would save me a lot of work and a lot of animosity.

Finally, allow me to express my appreciation for the care with which you phrased your comments.

With my greetings and best wishes,
yours ever,

Edsger W. Dijkstra

prof. dr. Edsger W. Dijkstra
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712-1188
USA