Some Comments on the Aims of MIRFAC.

Dear Editor,

Recently H.J. Gawlik [1] has published an article on project MIRFAC: A Compiler Based on Standard Mathematical Notation and Plain English. Its author is aware of earlier projects along analegous lines (MADCAP and COLASL [2]). When I heard of these earlier projects I was filled with some amazement for what they aimed to seemed to me hardly a sensible thing to do. I did not raise my voice then, convinced and trusting that people would discover this for themselves in a very short time. Now, two and a half years later I am faced with the fact that the movement has not died its natural death as I had supposed it would do. This discovery has caused me some disappointment and I can only regret my earlier silence on the subject.

The justification for the project MIRFAC seems the opinion that what is right for communication from man to man should also be right for communication from man to machine. (This is the only interpretation which allows me to attach a meaning to Gawlik's statement "that a compiler should aim not merely to simplify programming, but to abolish it.") But this opinion should not pass unchallenged!

If we instruct an "intelligent" person to do something for us, we can permit ourselves all kind of sloppiness, inaccuracies, incompleteness, contradictions etc., appealing to his understanding and common sense: he is not expected to perform litterally the nonsense he is ordered to do, he is expected to do what we intended to order him. And a human servant is therefore useful by virtue of his "disobedience". This may

be of some convenience for the master who dislikes to express himself clearly; the price paid is the non-negligible risk that the servant performs, on his own account, something completely unintended.

If, however, we instruct a machine to do something we should be aware of the fact that for the first time in the history of Mankind we have a servant to our disposal who really does what he has been told to do. In man-computer communication there is not only a need to be unusually precise and unambiguous, there is, at last, also a point in being so, if at least we wish to obtain the full benefits of the powerful obedient mechanical servant. Efforts aimed to conceal this new need for preciseness -for the supposed benefit of the user- will in fact be harmful: for at the same time they will conceal the equally new possibilities of automatic computing, of having intricate processes under complete control.

I go on quoting Mr. Gawlik: "... MIRFAC has been developed to satisfy the basic criterion that its problem statements should be intelligible to non-programmers, with the double aim that the user should not be required to learn any language that he does not already know and that the problem statement can be checked for correctness by somebody who understands the problem but who may know nothing of programming."

I do not see the point of Mr. Gawlik's "basic criterion". Elsewhere (see [3]) I have warned against the "...tendency to design programming languages so that they are easily readable for a semi-professional, semi-interested reader. (Symptoms of this tendency are

languages the vocabulary of which includes a wild variety of English words to be used in a nearly normal sense, and some translators that even allow a steadily expanding list of synonyms and misspellings for these words. Particularly, languages designed under commercial pressure have suffered seriously from this tendency.) It looks so attractive: "Everybody can understand it immediately." But giving a plausible semantic interpretation to a text which one assumes to be correct and meaningful, is one thing; writing down such a text[.....]expressing exactly what one whishes to say, may be quite a different matter!" On comparable grounds, John McCarthy calls "COBOL .. a step up a blind alley on account of its orientation towards English which is not well suited to the formal description of procedures" [4].

Furthermore, to set Mr. Gawlik's double aim is fooling oneself. Standard mathematical notation has been designed to describe relations and now we have to define processes. Plain English has grown out of a need of interhuman communication, to be vague and ambigueus, te tell jokes and to sing nursery rhymes, but is obviously unfit to express what has to be expressed now. One can borrow mathematical notations, one can borrow English words, but completely new semantics must be attached to them and despite its superficial similarity one creates a new language. And I think the similarity more misleading than clarifying.

This fear is confirmed by Mr. Gawlik's second aim, viz. "that the problem statement can be checked for correctness by somebody who understands the problem but who may know nothing of programming." Of course he can check it, but the crucial point is whether he will find the errors! And of course he will not find them: for in human communication

one is constantly trained to try to understand the others intentions and not to notice the nonsense. The corrector who understands the problem but knows nothing of programming, will be mislead by the familiarity of the characters and the words and he will, in all probability, be satisfied if he recognizes the problem.

I am all in favor of clear and convenient Algorithmic Languages, but, please, let them honestly be so: to disguise them in clothes which have been tailored to other purposes can only increase the confusion.

E.W. Dijkstra
Department of Mathematics
Technological University
Postbox 513
EINDHOVEN
The Netherlands

References:

[1] Gawlik, H.J. MIRFAC: A Compiler Based on Standard Mathematical Notation and Plain English. Comm. ACM 6, 9 (Sep.1963).

[2] Wells, Mark B. MADCAP: A Scientific Compiler is a Displayed Formula Textbook Language. Comm. ACM 4, 1 (Jan.1961).

[3] Dijkstra E.W. On the Design of Machine Independent Programming Languages. In Goodman, R. (Editor): Annual Review in Automatic Programming, Vol. III Pergamon Press, 1961.

[4] McCarthy, John. A Basis for a Mathematical Theory of Computation, Preliminary Report. Western Joint Computer Conference, 1961.