Trip report E.W.Dijkstra, Marktoberdorf 1988

J.C.S.P. van der Woude and I had arranged to drive together the 750 km to Marktoberdorf and van der Woude had agreed to collect me on Sunday 24 July at 5:00 in the morning; he made the mistake of being 40 minutes late. Alternatingly, we took the driver's seat for two-hour intervals. We arrived at the Marktoberdorf Gymnasium in the course of the afternoon; a violent thunderstorm that had travelled over Europe in the same direction reached Marktoberdorf two hours later. (From Richard Bird and Tony Hoare I heard that the storm had reached Munich Airport in all its violence ten minutes after their plane had landed.)

The schedule of the Summer School was as heavy as usual: 6+4 days with in total more than 50 lectures and 8 discussion sessions. The programme was perhaps heavier than usual: we had a set of high-powered speakers, who together had too much to tell. Their saving grace was the high level of care with which many of the visuals had been prepared and their obvious effort to be as clear as possible. (They were really grown-up: no matter how hard they worked on their lectures, they did <u>not</u> compete with each other for the favour of the audience.)

The theme of the Summer School was "Constructive Methods in Computing Science", but traditionally the theme is no more than a flag and the speakers are encouraged to talk about what is dearest to their hearts. The first days, the topics seemed to be of a bewildering variety; as the school progressed, an amazing number of (unplanned) links emerged.

Ronald Backhouse (University of Groningen) talked about "Constructive Type Theory". In an early stage, he lost his audience when he lingered too long on the propositions-as-types paradigm, but he worked very hard and re-established contact. Mathematically, I don't feel attracted to constructive type theory because of its close links to Gentzen's "Natural Deduction", which is usually so much more clumsy than equational reasoning. By showing the link with programming methodology, Backhouse convincingly argued the legitimacy of the computing scientist's interest in constructive type theory.

Richard S. Bird (University of Oxford) talked about "Constructive Functional Programming", and did so with contagious enthusiasm. His generalizations from dealing with lists to dealing with bags, sets, or trees revealed the well-educated mathematician. I doubt whether at my age I will take the trouble of getting his formalism at my fingertips; I'll know how to justify my abstinence by remarking that it is too much attention to too tiny a topic. But I am very

glad that the next generation of computing scientists has been exposed to it: it was nicely calculational and constructive, it demonstrated the importance of having theorems, and it showed how much can be achieved when notational courage and care are well-balanced.

Manfred Broy (University of Passau) spoke under the title "Towards a Design Methodology for Distributed Systems". I am afraid that he failed to reach me. For one thing, his visuals were very hard to read from the rear; moving to the front, I could read the characters, but not yet his formulae, as relative binding powers of infix operators and scopes of dummies usually had to be guessed. When he seemed to use his formalism primarily for descriptive purposes (rather than for manipulation) my interest waned.

Robert L. Constable (Cornell University) had as topic "On the Design and Construction of Problem Solving Environments". To begin with, I was — perhaps more than mildly — annoyed by his violation of the tradition that all speakers attend the complete Summer School: he showed up during the fourth day, thereby missing what Backhouse had told about a closely related topic. In his first lecture, his visuals were hardly readable; he somewhat redeemed himself by working very hard on the visuals for his later lectures. He was a conscious

lecturer and reached part of his audience; others had the uncomfortable feeling that here logic was parading as computing science. I was put off by his notational immodesty and the repeated intrusions of philosophy.

I gave this time only four lectures: I proved a conjecture of Sylvester's by deriving a program, I derived a proof of J.C.S.P. van der Woude, minimizing the number of rabbits to be pulled out of the hat and finally derived two prototype programs, building up specification or invariant as I went along. The lectures were well-received because everybody could follow them and could see that the arguments were elegant. I am afraid I failed to make clear why I had chosen to give these lectures.

C.A.R. Hoare (University of Oxford) was the only director of the Summer School that gave seven lectures; he gave them on "Approach to Category Theory for Computer Scientists". We had been given 44 pages of lecture notes – rich in examples –, at the end we received copies of 100 beautiful visuals mostly made during the Summer School. His purpose was to give "the reason why theoretical computer scientists should be interested in (and enjoy) category theory". In this respect he succeeded as far as I am concerned much better than R.M. Burstall several Summer Schools ago. He used preorders and monoids as stepping stones to cate-

gories, and spent most of his time on the stepping stones. He had abolished the traditional diagrams and worked with formula instead; all his proofs were short calculations and I fervently hope that this innovation of his is as great an improvement as I think it is. I have my doubts about his way of denoting functional composition, and his density of definitions was sometimes higher than I could cope with. His lectures were undoubtedly among the high points of the Summer School.

The alphabetically next speaker was Butler W. Lampson (Digital Equipment Corporation), whose topic was "Specifying Distributed Systems". He was the other speaker that arrived four days late. In doing so, he had missed my introduction in which I had asked the speakers to be constantly aware of the linguistical barriers and to speak therefore slowly and clearly. I blame myself for not having repeated the warning to Lampson, for now his first talk was somewhat of a disaster; moreover the lettering on his visuals was too small. He enlarged his visuals somewhat and spoke in later talks somewhat more slowly. In order to be "understood" back home, he had adapted his style of specification and his notational conventions to the tastes and expectations of his American colleagues. The resulting texts hardly seemed to lend themselves to manipulation; when too many errors showed up, my fears were confirmed and I lost

interest.

Jayadev Misra (University of Texas at Austin) had as title "Parallel Program Design: A Foundation". He spoke on the principles of UNITY, which he illustrated by developing quite a number of ambitious parallel programs. His lectures had been prepared with very great care, and were received accordingly. The information flow, however, was very dense, and it might have been kinder to the audience if he had shown fewer examples at a somewhat more leasurely pace; but the audience did not seem to mind. During one of the discussions, he was attacked by W.P. de Roever, who, having failed to catch the distinction between "ensures" and "leads to", had rushed to the conclusion that what Misra had done was all wrong; given the floor, de Roever completely discredited himself by making -pictorially and notationally- un unbelievable mess of his visual. I was somewhat shocked by the technical incompetence with which he tried to make his point; Misra dealt with him politely and effectively.

J Strother Moore (University of Texas at Austin) described under the title "Computational Logic" the recent inclusion of quantification in the Boyer/Moore theorem prover, and he did so very well. For most of the audience, this was the first more detailed exposure to automatic theorem proving;

moreover, the majority had no active experience in using LISP. Moore successfully overcame both handicaps. His lectures were most instructive and gave the audience a clear impression of the power of the theorem prover and of the constraints it imposes on its users. He made no secret of his unwillingness "to abandon the fruits of 18 years of investment". The extent to which the existing theorem prover might be a millstone around their necks was not so clear; the alternative of designing a completely new theorem prover seemed to have been rather rejected than seriously considered.

The last speaker was Jan L.A. van de Snepscheut (University of Groningen) on "Design of Concurrent Programs". He devoted two lectures on concurrent programs synchronized by means of semaphores and two on systolic computations. His visuals were done very well and contained precisely the information needed, and his arguments were terse and nicely calculational, but as speaker he was not his usual, relaxed, self. He sometimes seemed unhappy, as if he felt a mismatch between the examples he had chosen and the interests of the audience. In his first lecture, when introducing semaphores, he made the tactical mistake - I trust that he will never make it again - of calling the defining postulates "axioms". This

is something you can do in front of a Dutch audience, but, in front of an international audience, the mere use of the term "axiom" immediately triggers all the pollution of Platonic philosophy. To his utter amazement, people started complaining that his axioms were not "self-evident". He was taken aback, and I don't think he fully recovered.

* * *

The Summer School provided overwhelming evidence for the profound similarity between the design of a (proven) program and a (proven) theorem, and in doing so it did a great job. This emphasis, however, has also been experienced as misrepresentation, and not without justification: the practical differences remained underemphasized. On the one hand we should not forget that, viewed as a theorem, many a program is rather boring; on the other hand we should not forget that programs need to meet a greater scala of quality requirements than theorems: they need to be efficient and informal proofs rarely suffice for them. (The other day, a very recent Computing Science Master's Thesis at the Technological University Eindhoven turned out to be seriously flawed; his supervisor – a mathematician still in the process of being retreaded as computing scientist – had erroneously assumed that his traditional standards of rigour would

suffice.)

* * *

Regrettably - for he had very much looked forward to it - one of speakers who had been planned for four lectures, viz. David Gries, was prevented from attending. Fortunately, even without David Gries the scientific program was rather over- than underloaded. In this respect the Summer School was "as usual": too much of everything, except sleep.

Nuenen, 23 August 1988

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