Not only computing – also art

JOHN LANSDOWN

On not being precisely certain

Since the last issue, fuzziness and uncertainty has played an even more significant part in my life than usual. For two months or so, I was unable to see clearly after an operation to fix a defect in my only usable eye. Like the other one, it has a retina which has a disturbing tendency to fall off and float around aimlessly. However, the resulting fuzziness was better than not being able to see at all. Because of it, computer graphics has temporarily figured somewhat sparsely in my day-to-day working. This year there was no Siggraph nor Eurographics conferences for me and this means that, for this issue, only one illustration is available.

I did, though, attend a number of conferences and meetings on knowledge-based systems. In all these, I was struck by the fact that another form of fuzziness affected some speakers: they were apparently unable to distinguish between imprecision and uncertaintly. This is odd because these things are by no means the same and, if we are to represent more of our knowledge in computers, it will be necessary for us to be absolutely clear about the differences between fuzzy, imprecise concepts on the one hand and uncertain, probabilistic ones on the other.

We can make fuzzy statements without any appeal to probability - and *vice versa*. Thus, coming in out of the rain, I can say with certainty that I am wet. I might even say that I am very wet. The words 'very wet' in this context are imprecise because we don't have an accurate measure of wetness when talking about being drenched with rain. We simply understand that 'very wet' is wetter than just 'wet'. I can also remark 'It is going to rain tomorrow'. This is not a fuzzy statement. It is a precise statement of my belief about the weather. It is though, probablistic in the sense that no one will be surprised if my prediction turned out to be untrue. In a more numerate society than we have, I might rephrase my statement into something like, 'There is a 75 per cent chance of rain tomorrow'. People would then see that this is a precise expression of a probability. Confusion arises when we mix fuzzy and uncertain concepts together - something that we often do - like when we say, 'It's very likely to rain this afternoon'.

For about 200 years we have had the ability to deal with uncertainty by means of the mathematics of probability. It's only in the last 20 years, due to the seminal work of Lofti Zadeh and others, that we have been able to cater for imprecision by mathematical means. In both cases, scales of values are set in the range 0 to 1. Every schoolboy knows that something that cannot happen has a probability of 0; something that is certain to happen has a probability of 1. Anything that *might* happen has a probability lying somewhere between these limits: the more likely, the bigger the number. Furthermore, the probability of two unconnected uncertain things happening together, like for example, my winning the pools and it raining tomorrow, is given by the product of the individual probabilities. The probability that one or the other will happen is given by the sum of the individual probabilities.

Zadeh was the first to show that we could handle fuzziness by analogous methods and that our everyday expressions like 'very', 'not quite', 'almost', 'slightly' could be treated as if they were linguistic variables on a scale of 0 to 1. Thus, we might say that someone was 'experienced' or 'very experienced', 'naïve', 'completely experienced' and so on. These are obviously fuzzy concepts but we can manipulate them like uncertain concepts if we assume that 'naïve' has the fuzziness of 0 and 'completely experienced' has the value 1. Other degrees of experience can then have values of fuzziness somewhere between these extremes. He demonstrated that useful results could arise if, when we want to assess the fuzziness of two unrelated expressions taken together, we take the minimum of their individual values and, when we want to assess the fuzziness of one thing or another, we take the maximum of their individual values. This means that, given the fuzziness of 'slightly experienced in Prolog' as 0.3 and of 'very hardworking' as 0.8, we can assess the fuzziness of 'both very hardworking and slightly experienced in Prolog' as MIN (0.3, 0.8) = 0.3. Alternatively, we can say that 'either very hardworking or slightly experienced in Prolog' has a fuzziness of MAX (0.3, 0.8) = 0.8.

The mathematics of fuzziness is still being worked out but, already, it has found many useful applications in all sorts of areas. Like probability theory, it has many pitfalls for the unwary and is perhaps best used for assessing membership of classes or sets. Classical set theory works on the assumption that something either belongs to a particular set or it does not. Fuzzy set theory assumes that things are not so clear cut and that something can belong to a set with fuzziness of, say, 0.6 or 0.2. This idea seems to accord more with our view of reality than does classical set theory. We are very hard put to distinguish things in terms of their everyday characteristics and constantly meet borderline cases which are difficult to classify. Fuzzy set theory can certainly help here. In particular, I see it as a potential tool for introducing commonsense reasoning into our computer systems.

A clearing in the forest

A few years ago (*Computer Bulletin* September 1978), I commented that I was experimenting with a parametric method of storing details about the graphic outlines of botanical trees to go with my tree selection program. Rather than store an enormous number or graphical data on each tree in various configurations, the idea was simply to store a set of parameters which would generate the configurations on request.

Paul Brown, current Editor of PAGE: *The Bulletin of the Computer Arts Society*, did some work on this, using a fractal approach but lots of things intervened to prevent my continuing with the idea. Recently, some Japanese workers published a paper showing that it can be done and that fairly realistic pictures of particular trees can, in fact, be generated from quite limited data. Those interested will find a comprehensive description of their methods in the May 1984 issue of *IEEE Computer Graphics and Applications*. The article is by M. Aono and T. L. Kunii and is called, 'Botanical Tree Image Generation'.

Computer art and graphics

Computer graphics is such an inviting subject that it is not surprising that there is currently an enormous spate of books being published on its various aspects. One such is *Computer Art and Graphics: how to program with personal computers* by Axel Brück (Paul Petzold Ltd, London 1984 £14.95). This is an interesting work which introduces readers with a knowledge of Basic to the intricacies of making drawings by means of computers. Unfortunately, however, many of the most striking illustrations it

Book reviews

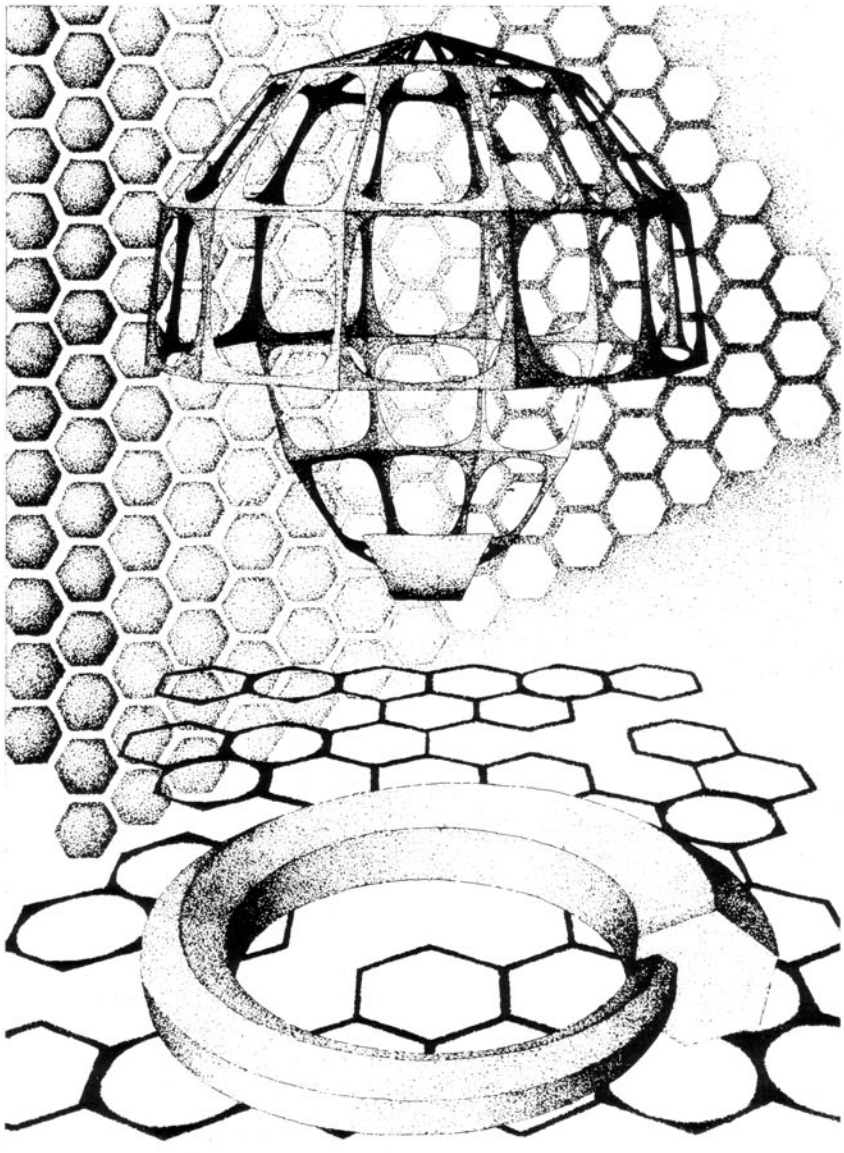

Figure 1

contains (such as Figure 1) can only be partially created by means of the programs given. It is clear that these drawings have been set up and plotted from the techniques described but then worked on by conventional manual means. The author does, in fact, justify this approach in his first chapter when he says, '... we will use the computer to generate those parts of the drawing which it can handle more quickly, more precisely, or in some other way more proficiently, than we could ourselves'. This is a perfectly justifiable approach, but casual bookshop visitors might buy the book before they realise that the artist has played such a large part in the creation of the illustrations.

It is hard to explain computer graphics programming in a structured way (and even harder for the reader to understand) when examples are given only in Basic. This is particularly so when, as in this book, the programs are set out from 40 column listings and only two-letter variable names are used. However, despite some infelicities, the author does a creditable job which could serve to encourage novices to try their hand at creative computing.

Digital Systems with Algorithm Implementation
By M. Davio, J-P. Deschamps and A. Thayse, 1983. Wiley, £14.95

This is an advanced book, intended for the expert in the implementation of digital control systems. In this respect, it is not for the computer user with a bias towards software or data processing applications, but very much for the hardware and electronics user. It is comprehensive in its covering of the hardware process control field, and necessarily presupposes much initial knowledge and ability. Even in one with considerable experience of such systems, it was not an easy matter to comprehend much of its contents, and this probably reflects accurately both its advanced status and its proper worth. Its price makes it a most attractive buy for any suitable user who is not deterred by a sound theoretical treatment of an advanced applied topic.
J. H. CHEETHAM

Ada for Programmers
By Eric W. Olsen and Stephen B. Whitehill, 1984; 310 pages. Prentice-Hall, £17.05

This book is intended as a guide to the US DoD programming language Ada for experienced programmers. In particular, useful comparisons are made between the features of Pascal and Ada. It has the advantage that it describes the ANSI standard form of Ada rather than an earlier provisional form (of which many Ada books are guilty!). However, it fails to state the LRM (language reference manual) used. Thus, the reader has to deduce which version of Ada is the subject of the book.

Unfortunately, it suffers from several faults which prevent it from being recommended (especially in view of some of the alternatives available). The book makes no mention of interrupt handling and the only description of features such as representation specifications, input/output and pragmas is in the form of copies from the LRM. These copies are exactly as found in the official LRM including section numbering and references (which are meaningless in the context of this book). The programming examples provided in the main body of the text do not conform to the style used in the LRM, which is especially noticeable since parts of the LRM are included in the book. Tasking is inadequately treated.

In conclusion, I cannot recommend this book to its target audience, *ie* programmers. However, if the book were cheaper, it could be of some use as an 'overview' for technical managers.
P. REID *Macclesfield*