they do serve to undermine my confidence in the catalogue as a whole. Since I know that some of the entries on topics I understand are flawed, how can I trust those on topics about which I know nothing?

Perhaps most disconcerting of all as far as this discouraging line of thought goes is the entry for Fortran (81). I had to read it several times before I realized its mirthful intent, though others, I am sure, will catch on as soon as they read "*Fortran is the programming language considered by many to be the natural successor to Lisp and Prolog for AI research*" and be splitting their sides by the time they reach the contributor, Aloysius Hacker. What worries me is that now I know that some of the descriptions are booby-trapped in this way, how can I ever trust the entries for such outlandish sounding techniques as viewer-centered co-ordinates (252)?

To be fair, most of these quibbles are largely due to the fact that the entries are unsolicited contributions which have been arbitrarily altered by the reviewers, so the problems with them cannot be laid directly at anyone's door. This style is inevitable in a document that attempts, as this does, to present an ever-changing picture of the state of AI with readers urged to write in with improvements and updates. I believe, however, that it would improve the catalogue if some kind of quality control was applied to the entries to ensure a more even standard even if this means slowing down the turnover of editions. It would also make the catalogue more useful and a more worthwhile purchase. Who is going to buy a reference book that cannot be fully trusted and which they are expected to make redundant by picking holes in, unless of course a discount on the next edition is offered to contributors?

Computational logic by JW Lloyd (Ed.), Springer-Verlag, 1990, pp 211, DM46.

Reviewed by: Simon Parsons, Department of Electronic Engineering, Queen Mary and Westfield College, Mile End Road, London E1 4NS, UK.

This book consists of the proceedings of the "Symposium on Computational Logic" held during the 7th ESPRIT Conference Week in November, 1990, and according to the foreword, inspired by the work of Esprit Basic Research Action 3012 (COMPULOG). The book seems to be intended as the first of a series with the admirable aim of disseminating results obtained by basic research actions which might otherwise be hidden away in stacks of unpublished deliverables. Unfortunately, in their eagerness to attract international names, no doubt to boost the profile of their work, the organizers of the symposium have to some extent marginalized those supported by the basic research action.

There are two sections to the book. The first consists of ten research papers detailing the work of invited luminaries, both from COMPULOG and North America, including such well known figures as Bob Kowalski, Alan Bundy, John McCarthy and Dana Scott. These cover topics as diverse as the study of open defaults and the teaching of symbolic computation. The second section contains five position papers for a panel discussion on "Programming in 2010: The Role of Computational Logic". This seems, at first, a formidable line up, but its impact is considerably reduced when one takes into account the fact that seven of the papers, two research papers (those of Scott and McCarthy) and all five of the panel papers, are only available as abstracts. There is, nevertheless, sufficient material in the remaining papers to make the book both interesting and useful. However, the diversity of the work described suggests that no individual is likely to be interested in more than a couple of the contributions, making this a book for the library rather than essential desktop reference.

The volume opens with papers by Bob Kowalski and Alain Colmerauer on developments in computational logic. This is a term which, as Kowalski notes, has no universally agreed definition although that used by COMPULOG, namely "*the use of logic for all aspects of computation*", seems reasonable enough. Kowalski's paper discusses the ways in which logic programming may be extended. He shows how fault diagnosis and default reasoning may be tackled by abduction, and discusses how metareasoning may be used to implement knowledge assimilation, reflection, and

the handling of "*knowledge and belief*". This enables him to conclude that the extension of logic programming techniques to encompass full first order logic and modal logic is unnecessary, whilst the inclusion of abduction and metareasoning, along with integrity constraints, are essential. Alain Colmerauer describes the Prolog III family of programming languages which are based on the redefinition of the unification procedure at the heart of the Prolog language. The original unification procedure, suggested by Robinson in 1965, is integrated with mechanisms for manipulating trees, including infinite trees, and lists, Boolean algebra, mathematical operations and relations. This, it is claimed, replaces "the very concept of unification by the concept of constraint solving in a chosen mathematical structure". There is a full description of the functionality of the new languages, with full examples, and some details of the implementation.

The next papers, by Raymond Reiter and Vladimir Lifschitz present very different work, both being complex technical papers on knowledge representation. Lifschitz discusses some problems inherent in Reiter's (1980) Default Logic, caused by the interpretation of free variables. He proposes an alternative treatment which alters the formalism so that it may be viewed as a syntactic transformation of sentences, and is thus similar to circumscription (McCarthy, 1980). Reiter starts from the assumption that a database is best viewed as a set of first order sentences, and continues by extending Levesque's (1984) work on the modal query language KFOPCE discovering that it is suitable for reasoning about queries and constraints. He also provides a sound Prolog-like evaluator for a well defined set of queries.

Passing over the papers by McCarthy and Scott, which were doubtless interesting but are only included in abstract form, we come to a group of papers concentrating on technical results in logic programming, all from groups associated with COMPULOG. Turini *et al.* introduce some basic metalevel operators for logic programs, they define the union and intersection of sets of definite Horn clauses, and provide model theoretic semantics for $P \cup Q$ and $P \cap Q$ in terms of the models related to P and Q. A metainterpreter is defined for the operators, and an operator that retracts all the clauses defining a predicate P from a theory Q is given. Bundy *et al.* start with the assertion that the holy grail pursued by adherents of logic programming is to allow computer users to write down their requirements in predicate logic which is interpreted by the machine freeing the users from having to think in procedural terms. The authors then tackle some of the problems with adopting this approach by adapting the "proofs as programs" paradigm for creating functional programs from logical specifications so that it generates logic programs.

Apt and Pedreschi provide a theoretical basis for studying the termination of logic programs with the Prolog termination rule. The study concentrates on "left terminating programs", those which terminate for all ground inputs. Siekmann and colleagues address the problem of representing generic concepts and taxonomic hierarchies of given domains, especially the combination of the fast taxonomic reasoning algorithms that support knowledge representation languages and reasoning in first order logic. The interface is achieved by constrained resolution where the computation of a unifying substitution is replaced by a unifiability test. Instead of instantiating the resolvent with the substitution, a constraint consisting of the relevant term equations is added to it.

As this summary shows, the symposium covered ground of interest to researchers in artificial intelligence, logic programming and software engineering, and the papers included in the book reflect this diversity. However, the very diversity makes it difficult to see to which of these groups is likely to get its money's worth from the material, especially given the high quality binding reflecting, I would imagine, an equally high quality price tag. The fact that nearly half of the papers are abstracts less than two pages long reduces the appeal of the volume still further. I for one hope that the contributors to further symposia organized by such basic research actions are not permitted to escape so lightly, and that consequently future collections deliver as much as they promise on the contents page. In addition, it would be nice to see later volumes give more space to those actually working on the project, at the expense of the big names. If the projects are as strong and as productive as COMPULOG, there seems to be little to be gained, and perhaps much to be lost, by showcasing international figures at the expense of home grown talent.

References

Levesque, HL, 1984, "Foundations of a functional approach to knowledge representation" *Artificial Intelligence* **23** 155–212.
McCarthy, J, 1980, "Circumscription—a form of non-monotonic reasoning" *Artificial Intelligence* **13** 27–39.
Reiter, R. 1980, "A logic for default reasoning" *Artificial Intelligence* **13** 81–132.

Handbook of genetic algorithms by Lawrence Davis (Ed.), Chapman & Hall, London, 1991, pp 385, £32.50.

Reviewed by: GV Conroy, Computation Department, UMIST, Manchester, UK.

This text is the second edited by Lawrence Davis dealing with the topic of genetic algorithms. The first *Genetic Algorithms and Simulated Annealing* (Pitman, 1987), was probably the first to introduce the subject matter to the general public, or rather to the general academic. The papers presented in the early text were very much concerned with the theory of the subject, and were largely academic in nature with practical applications only remotely hinted at. With the latest text one can say that the subject has "come of age". A minor point is that the subject of the text is strictly genetic algorithms applied to optimization problems; however, the editor points this out very clearly early in his exposition, and since the problem domain is a real and difficult one this is not intended to be a criticism of the book.

Since to date the subject matter has remained rather firmly within academic circles it is worthwhile outlining briefly just what the genetic algorithm is all about. John Holland, the inventor of the subject, was intrigued with the way that natural selection amongst biological species caused such species to evolve in a relatively short time to complex beings well adapted to their environment. He therefore carried out a program of research into how one could mimic the evolutionary behaviour of natural species with a formal computer algorithm to solve difficult problems. Simplistically speaking, evolution is a process operating on the chromosomes of living beings. The chromosomes are biological mechanisms for encoding the structure of those beings, and are taken to consist of chains of genes which provide the inherited characteristics of living organisms. Successful members of their species are believed to reproduce more often than the weaker members, thus spreading the advantageous genes throughout the population. It is within the reproductive process that evolution is believed to take place, whereby the chromosomes of the parent organisms exchange genetic material. The new chromosomes may be more successful than the parent organisms, will therefore take part in reproduction more often, and eventually dominate the population they belong to. Since the genetic material may possibly converge to a fixed set of genes in some positions in the chromosomal chain, a process of mutation is also believed to operate, albeit rarely, thus allowing the introduction of new genetic material into a stagnant population.

The genetic algorithm mimics this process in the manner described below. Firstly, an attempt at a solution to a problem in the relevant domain has to be encoded as a bit-string vector of features. Secondly, operators have to be found to manipulate these bit-strings in a manner similar to biological reproduction, that is, operators have to be found that will exchange bits between parent vectors in a meaningful manner for the domain at hand, to produce child vectors. Without going into details, two operators are typically used here, crossover and inversion. Thirdly, mutation is introduced whereby bits may be randomly altered to values not possessed by either parent. The children are then assessed for fitness as solution vectors to the particular problem being considered, and the fittest are then allowed to reproduce more than the weakest. The hope is that the population of vectors will eventually converge to a good solution to the problem under consideration.

Each of the above stages in the genetic algorithm process presents its own difficulties. The first stage, that of finding a suitable bit-string vector representation of a possible solution vector, is not a trivial task for demanding problem domains. The choice of the population size to be used is a trade off between the convergence rate of the population and the loss of good genetic material if kept too