Correspondence

Pascal for declaration of types

Dear Sir,

I should like to suggest a small improvement to Pascal-like languages for the declaration of types. At present in Pascal you can define an unnamed type in a VAR declaration like this:

```
VAR colour: (red, green, blue),
```

but if you want to declare another variable with the same type as **colour** you have to split the declaration into two parts as follows:

```
TYPE coltype = (red, green, blue);
.
.
VAR colour:coltype;
.
.
PROCEDURE xyz;
VAR colour2:coltype;
```

My suggestion is this. If **type-name = type-description** is allowed wherever **type-name** is allowed now, the type definition can be written

```
VAR colour:colytype = (red, green, blue);
```

As well as simplifying the definition of user-defined types, the simultaneous declaration of a variable and its type provides a way round the problem that Pascal, Modula 2, Ada, etc. all have with the declaration of pointers to records. Pascal and Modula 2 both allow a pointer to an undeclared type to be defined so that the type can contain a pointer to another element of the same type as in a linked list or binary tree. Ada requires an incomplete definition of the undeclared type to be given. Neither of these is completely satisfactory.

I suggest that pointers should be declared during the definition of a record or other compound type instead of before it like this:

```
TYPE node = RECORD
              contents:integer;
              link:nodeptr = ^node
            END {RECORD};
```

or even:

```
VAR root:tree =
    ^branch = RECORD
              key:real;
              left, right:tree
            END {RECORD};
```

The only time you would then need to define a pointer to an undeclared type is if you wanted a complicated structure with two or more different record types with pointers to each other, like a memory-resident version of a Codasyl database. In these less common situations it is not unreasonable for a computer language to insist on a pre-declaration like FORWARD procedures in Pascal so that the compiler can tell that it is dealing with an undeclared type and not one that has been misspelt or omitted by accident. I suggest the following as examples of its usage:

```
TYPE btype: FORWARD;
     bptr = ^btype;
```

or

```
TYPE bptr = ^btype = FORWARD;
```

Yours faithfully

M. H. HUBBARD
Churches Work Scheme Ltd,
Church of the Ascension,
Hulme, Manchester M15 5FQ

Legal Logic Programming

Dear Sir,

When I wrote my 'Fundamental Errors in Legal Logic Programming'[2] I was very much aware that I was putting forward a heterodox view and, as such, could expect a variety of reponses ranging from the academic to the personal. The latter type I imagined would be those which intimated that I was 'rocking the boat' or had been too forthright. I did not expect to be accused (as I have been by Professor Kowalski and Marek Sergot[1] of making personal attacks myself. ('We take', they state, 'particular objection to quite outrageous claims.') I was even told of one logic programmer who described me as that 'little Irish pedant'. I am, in fact, Scots. No doubt, though, feeling runs high; for I am as committed to my anti-logic-programming stance as logic programmers are to theirs.

However, in the space you have offered me, I wish, without qualification, to state that I have never believed nor attempted to hint that any member of Professor Kowalski's team was racist. I, in fact, take objection to that unwarranted reading of my article. What I have pointed out is that, almost without exception, legal textbooks on immigration law and citizenship have pointed to the racist nature of much of the legislation – Grant and Martin[3] even devote a whole chapter to this factor. I used this point in (as I thought) a *technical* argument to show that logic programming could not encapsulate this aspect of the legislation.

Along with some personal comment, though, I had expected some academic response to my article. This has been much more limited than I had wished, and much of it seems to be confused. For example, I am not sure whether the 'logic programming in law' project is experimental or is meant to provide working systems; take one sentence from Kowalski and Sergot's letter: 'Since there is more to legal reasoning than the literal application of the letter of the law, we did not propose that our program could be used to decide questions of British citizenship autonomously' and from the team's major paper: 'An obvious application of the formalization of the act is to determine whether in a given circumstance a particular individual is or is not a British citizen.'[4] Also note from their letter that they are interested in reasoning in law: 'It is precisely because reasoning in law demands such great flexibility that we believe it is an ideal domain in which to test the application of these developing techniques,'; and from that same paper they seem to deny this interest in legal reasoning, only evidencing an interest in mechanical rule handling: 'we have not addressed the broad and much more difficult problem of simulating legal reasoning. Rather, we have concentrated on the limited objective of implementing rules and regulations with the purpose of applying them mechanically to individual cases.'[4]

In the preparation of any paper, one must be selective about the evidence to be used to back up one's case. The selectivity is mostly brought on by the limitations of length. In the 'Fundamental Errors..' paper I concentrated upon the notion of 'clear rules'. As Kowalski and Sergot rightly point out, my argument against clear rules is as much an argument against Fortran programs as against Prolog programs; indeed I have always believed that Prolog is just another programming language. But no one is making claims for 'Fortran in law' as they are for logic programming in law.

Other arguments I could have put forward might have dealt with the differing natures of rules and their application (for example, as 'legal rule' differs markedly from an 'administrative rule', as can be seen by the DHSS's internal rules, written legislation, tribunal interpretation of both these rules etc.). What this argument would have had in common with my clear rule argument is the belief that rules are much more complex conceptual objects than one at first imagines – that they cannot simply be slotted into a memory location and 'logically analysed'. I am perfectly prepared to further argue this, and indeed hope that Professor Kowalski and Marek Sergot will substantiate their statement: 'We would argue that [Leith's attack] is also based on a mistaken impression of the jurisprudential material which he cites', for I would be pleased to see the Imperial College team start to take studies of the nature of law more seriously than they have in the past. I would particularly like to see them more fully explicate this notion of 'logical analysis; and how a 'rigorous derivation of its logical consequences can only make [the BNA] racist character more apparent.' I would have thought that reading a textbook on immigation law would have offered more clarity than such a static analysis. The notion of 'logical analysis', it seems to me, is simply a slogan of logic programming; it is a supposed technique which has never been described.

My opposition to logic programming exists on two fronts. First, in the general, I feel that the belief in logic as a means of encapsulating all information ('There is only one language suitable for expressing information – whether declarative or procedural – and that is first-order predicate logic. There is only one intelligent way to process information – and that is by applying deductive inference methods')[5] is simply a reversion to the logical positivism and logical atomism of the turn of the century. It can do computer science little good to be taking up, now, something that was dropped by philosophy 50 years ago (or even 350 years ago).[6]

Second, my opposition to logic programming in law in particular is that the legal area seems to have been chosen more as a test

bed for logic programming that with any real desire to handle real law. And yet it has been considered suitable for incorporating within, for example, the DHSS demonstrator. The team have brought logic-tinted spectacles to their research, and have been much more interested in adjusting the spectacles than in seeing what lies beyond that tint of logic.

I do have hopes that Professor Kowalski and Marek Sergot will come around to my view of the utility of logic as means of representing information (i.e. that it offers little utility). For even Carnap gave up logical positivism.[7] and who would ever – at the height of that movement – have expected that? Until then, I shall simply cling to my heterodoxy and ignore the suggestions that my expressed views have negative 'personal overtones'.

Yours sincerely,

PHILIP LEITH
Faculty of Law
The Queen's University of Belfast
Belfast BT7 1NN, Northern Ireland

References

1. R. Kowalski and M. Sergot, Letter to the Editor, *The Computer Journal* **30**, (3), 285 (1987).
2. P. Leith, 'Fundamental errors in legal logic programming', *The Computer Journal* **29**, No (6), 545 (1986).
3. L. Grant and I. Martin, *Immigration Law and Practice* London (1982).
4. M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond and H. T. Cory, The British Nationality Act as a logic program. *Comm.ACM* **29** (5) (1986).
5. R. Kowalski, *Reply to Questionnaire, Sigart Newsletter no.* 70 Special Issue on Knowledge Representation (February 1980).
6. P. Leith, Involvement, detachment and programming: the belief in Prolog. In *The Question of AI*, edited B. Bloomfield. Croom Helm, London (forthcoming).
7. R. Carnap (1956). Reprinted in *The Methodological Character of Theoretical Concepts*, Minnesota Studies in the Philosophy of Science, **1**, 38–39.

Announcements

23 FEBRUARY 1988

B.C.S. Parallel Processing Group and Occam User Group, One-day workshop on **Parallel Architectures for Artificial Intelligence**, at Birkbeck College, Malet Street, London WC1.

The impact of parallel hardware and software on problems in the field of Artificial Intelligence has attracted considerable interest from varied disciplines. Computational models such as Logic, Functional, Dataflow and Communicating Sequential Processes have strongly influenced the design of computers, but these must match the requirements of applications such as Natural Language, Pattern Recognition, Mathematical Reasoning and Decision Support Systems. The workshop will bring together researchers working in these fields and provide a forum for discussion.

If you would like further information, please contact:

Mr A. Gupta, STC plc, Department 32372, Oakleigh Road South, New Southgate, London N11 1HB (phone 01 368 1234, ext. 2955) *or* Mr Steve E. Zenith, INMOS Ltd, 1000 Aztec West, Almondsbury, Bristol BS12 4SQ (phone 0454 616616).

20–22 APRIL 1988

Euteco 88 European Teleinformatics Conference, organised by the European Action in Teleinformatics COST 11ter, Technical University of Vienna, Austria.

Purpose

The objective of EUTECO 88 is to disseminate the scientific information generated by the COST 11ter Action, to compare it with other research work in the teleinformatics field in Europe and to discuss future directions of research in this area.

Background

The COST (Coopération Scientifique et Technique) Action 11ter, a concerted research action project in the field of teleinformatics, was established in 1985 as a successor to the COST 11bis Action. The project is providing a framework for European cooperative basic research within a specified scientific range of activities in teleinformatics. The COST 11ter Action is part of the European Community Mulitannual Programme (1984–6) in the field of data processing and paired by following other European countries: Austria, Finland, Norway, Spain, Sweden, Switzerland and Yugoslavia.

In its two-year operation, the COST 11ter Action has established projects and conducted working groups on specific topics bringing together national research projects in a European framework. At present more than 90 research teams – mainly from universities and public research institutions but also to a smaller extent from industry and PTT – participate in the activities, involving about 200 researchers.

The COST 11ter Action is executed by the Commission of the European Communities (Directorate General XIII, Telecommunications, Information Industries and Innovation) with the assistance of the COST 11ter Concertation Committee composed of representatives from all participating countries and the Commission.

In 1983 the COST 11bis Action finished with a conference, called EUTECO (European Teleinformatics Conference) held in Varese/Italy, the *Proceedings* of which have been published by North-Holland.

Topics

- Computer-mediated human communication.
- Management of distributed systems; OSI and distributed operating systems' aspects.
- Architectural issues in distributed systems.
- Security issues in distributed systems.
- Human factors in telematics.
- Distribution aspects in generalised databases, including issues of knowledge distribution.
- Formal description techniques for distributed systems.
- High-speed WAN communication: applications in teleinformatics.

EUTECO Programme structure

In **plenary sessions** with a number of distinguished invited speakers from academic and public research institutions as well as from industry, directions and trends in teleinformatics research will be presented. Current work and perspectives of standardisation will also be discussed.

Parallel sessions on specific topical areas will be conducted to present research work from European laboratories.

Panels will discuss future directions in teleinformatics research.

Contact address

The address for general enquiries is:
EUTECO 88, R. Speth, CEC, DG XIII, COST 11ter, 200 rue de la Loi, B-1049 Brussels (Tel. 0032 2 236.04.16, telex 21877 COMEU B; EMail COM/Stockholm and EUROKOM/Dublin, ROLF SPETH).

Please write to this address if you wish to receive the full programme

28–30 JUNE 1988

Third International Conference on Data and Knowledge Bases, Improving Usability and Responsiveness, Jerusalem, Israel.
(Sponsored by the Information Processing Association of Israel (IPA) in cooperation with ACM.)

The conference follows the two previous conferences on databases held in Israel in 1978 and 1982. Its goal is to provide a forum for researchers and practitioners, to present and discuss advances and new directions in the database and related areas. The emphasis of the conference will be on significant new concepts and ideas, leading to a better understanding of the structure of database systems, or their relationship to related fields, such as Artificial Intelligence, Programming Languages, and Distributed Computing, and of their use in new application domains. Both theoretical and systems-oriented papers will be accepted.

Topics of Interest

Topics of interest include, but are not limited to, the following.

- *Data and knowledge.* Logic Programming languages and interfaces for databases, intelligent (deductive) database systems, data-