Comments on Futo, Gergeley
by Brian R. Gaines, Department of Computer Science, York University,
4700 Keele Street, Downsview, Ontario, Canada M3J 1P3.

INTRODUCTION

Futo, Gergely and Szeredi have put forward a very interesting approach to the first problem situation of this conference, that of remote space vehicle control. The methodology they propose is based on a computer-based problem-solving system, Prolog, that tests assertions against a database and it is noteworthy that it is also applicable to the other two problem situations. The authors presumably chose the first situation as their example because it involves a classical dynamical system which, at first sight, is an unlikely candidate for logic simulation, but where their extensions through temporal primitives in T-Prolog (Futo & Szeredi 1981) are shown to full advantage.

This paper is one in a sequence (referenced in the main paper) and a full appreciation of the methodology in a wider range of situations can be obtained from these references, particularly the T-Prolog manual and Futo, Szeredi and Szenes (1981). This commentary is concerned first of all with the wider background to the development of Prolog, its virtues and deficiencies, and then to the impact of these on the applicability to the first problem situation.

PROLOG AND EXPERT SYSTEMS

One of the most remarkable developments in computer science has been that of so-called **expert systems** (Michie 1979). These have been initially restricted to very small problem domains but through dealing with these effectively techniques have been developed which are generalizable. These techniques can be grouped into three major categories: databases for knowledge representation; acquisition schema for setting up the database generally from human experts; and inferencing systems for using the database to solve specific problems. Prolog (Clocksin & Mellish 1981) falls into this last category and is a language based on a Kowalski's (1974) notion of **logic programming** in which a program is a collection of assertions stored inside a **logic machine.**

The flavour of Prolog is best obtained through a simple example. Consider a database consisting of instances of a two-variable predicate, eats(X,Y) meaning "X eats Y", e.g.:-

```
  eats(George,sausage).
  eats(Horst,chips).
  eats(George,ham).
  eats(Horst,sausage).
```

Then the Prolog queries:-

```
  ?- eats(Horst,chips).
  yes
  ?- eats(George,chips).
  no
```

produce the expected results. Variables may be used to search the database for all items satisfying the query:-

```
  ?- eats(X,sausage).
  X=George;
  X=Horst;
  no
```

Comments on Futo, Gergeley
by Brian R. Gaines, Department of Computer Science, York University, 4700 Keele Street, Downsview, Ontario, Canada M3J 1P3.

INTRODUCTION

Futo, Gergely and Szeredi have put forward a very interesting approach to the first problem situation of this conference, that of remote space vehicle control. The methodology they propose is based on a computer-based problem-solving system, Prolog, that tests assertions against a database and it is noteworthy that it is also applicable to the other two problem situations. The authors presumably chose the first situation as their example because it involves a classical dynamical system which, at first sight, is an unlikely candidate for logic simulation, but where their extensions through temporal primitives in T-Prolog (Futo & Szeredi 1981) are shown to full advantage.

This paper is one in a sequence (referenced in the main paper) and a full appreciation of the methodology in a wider range of situations can be obtained from these references, particularly the T-Prolog manual and Futo, Szeredi and Szenes (1981). This commentary is concerned first of all with the wider background to the development of Prolog, its virtues and deficiencies, and then to the impact of these on the applicability to the first problem situation.

PROLOG AND EXPERT SYSTEMS

One of the most remarkable developments in computer science has been that of so-called **expert systems** (Michie 1979). These have been initially restricted to very small problem domains but through dealing with these effectively techniques have been developed which are generalizable. These techniques can be grouped into three major categories: databases for knowledge representation; acquisition schema for setting up the database generally from human experts; and inferencing systems for using the database to solve specific problems. Prolog (Clocksin & Mellish 1981) falls into this last category and is a language based on a Kowalski's (1974) notion of **logic programming** in which a program is a collection of assertions stored inside a **logic machine.**

The flavour of Prolog is best obtained through a simple example. Consider a database consisting of instances of a two-variable predicate, eats(X,Y) meaning "X eats Y", e.g.:-

```
  eats(George,sausage).
  eats(Horst,chips).
  eats(George,ham).
  eats(Horst,sausage).
```

Then the Prolog queries:-

```
  ?- eats(Horst,chips).
  yes
  ?- eats(George,chips).
  no
```

produce the expected results. Variables may be used to search the database for all items satisfying the query:-

```
  ?- eats(X,sausage).
  X=George;
  X=Horst;
  no
```

where we type the semicolon ";" to ask Prolog for any further instances. More complex queries may be generated through the use of shared variables, e.g. "what food do both George and Horst eat?" is:-

```
?- eats(George,X), eats(Horst,X).
X=sausage;
no
```

By using more complex predicates and queries, possibly involving arithmetic, Prolog may be used to solve a wide range of significant problems.

Prolog's deficiencies lie in several different areas. It is not a universal theorem prover and is dependent on its sequence of search in finding a solution so that some solutions may be missed. This problem is exacerbated when speed-up techniques such as the "cut" for terminating a search are mis-applied. The need for speed-up techniques highlights another problem, that data-base searching may be very time-consuming both intrinsically and through poor implementation or poor problem-representation. There are deeper problems also. Prolog's answer of 'no' to a query like ?- eats(Horst,bread) does not mean "no" but rather "I don't know", i.e. the relevant information is not in the database. This shows up particularly in queries involving negation (Clark 1978, Dahl 1982), and is an example of a general weakness in database systems that are based on two-valued logic (Gaines 1981). Belnap (1976) has proposed a very interesting 4-valued logic solution to this problems that also takes into account the possibility of contradictory information in the database.

These are the types of difficulties that, although significant, may be seen as: part of the problem rather than the methodology -- we can pose problems where the exploration of the solution space leads to a combinatorial explosion; or as part of the problem representation rather than the problem solving -- many real-world problems involve vague or contradictory information and yet are still soluble using that information. Other difficulties that are again not ones of Prolog itself are those of obtaining the database itself, either by elicitation from an expert or by induction from examples (Michalski & Chilausky 1981). On the positive side Prolog is remarkably easy to implement, will run on quite small microcomputers, and its operation is based on a small, uniform set of algorithms that are logically analysable and comprehensible, unlike the heuristics of early AI research that were florid and meaningful only for their originators.

T-PROLOG, SIMULATION AND CONTROL

The database driving Prolog can be set up to handle dynamic simulation problems such as those of mechanics (Bundy, Byrd, Luger, Mellish & Palmer 1979). T-Prolog is an implementation of Prolog by the authors of the main paper which adds additional primitives to the language to enable such problems to be set up with the minimum of effort. These enable it to deal with predicates involving time, communication, synchronization and resource allocation. The detailed example they give of the first problem situation posed in T-Prolog shows the power of this approach. What one has is a statement of the basic components of the system and the constraints interrelating them. From this it is easy to check that the situation has been properly described, and to examine variants of that situation if one wishes.

The significance of this is best appreciated by noting the ease with which additional features can be added to the problem, e.g. an intermediate control station with certain capabilities. If the problem were expressed as a set of procedural computer programs rather than a set of assertions then the difficulties of comprehending it, modifying it and adding to it would be very much greater. The remarkable feature of the Prolog expression of the problem is the lack of

computational algorithms. It is the Prolog system that generates these in response to specific queries.

Thus, the method is elegant and powerful in problem expression. What we cannot tell from the example given is how it will cope with particular queries such as determining a safe trajectory from one point to another whilst coping with problems that arise en route. Will it arrive at the best solution and how long will it take to do it? These problems will also be made worse in a realistic situation by the incorporation of uncertainties in the problem situation such as a probability density for meteorites. We are then not looking for a well-defined solution but for the possible solution with the highest probability of success and this greatly increases the search space.

CONCLUSIONS

In summary one can say: firstly, that the statement of a problem situation through a set of assertions is very natural, communicable and effective; secondly, that the solution of problems directly from the assertions without specific programming is very attractive and desirable; but thirdly, it is not clear yet how many real-world problems are amenable to such an approach and how many need problem-specific algorithm generation for their efficient solution. The paper by Futo, Gergely and Szeredi and the programme of research they outline is a significant contribution to the future resolution of this question.

REFERENCES

Belnap,N.D. (1976). How a computer should think. In Ryle,G. (Ed.), **Contemporary Aspects of Philosophy.** Oriel Press, Stockfield. 30-56.

Bundy,A., Byrd,L., Luger,G., Mellish,C. & Palmer,M. (1979). Solving mechanics problems using meta-level inference. In Michie,D. (Ed.), **Expert Systems in the Micro Electronic Age.** Edinburgh University Press, Edinburgh, UK. 50-64.

Clark,K.S. (1978). Negation as failure. In Gallaire,H. & Minker,J. (Eds.), **Logic and Databases.** Plenum, New York.

Clocksin,W.F. & Mellish,C.S. (1981). **Programming in Prolog.** Springer-Verlag, Berlin.

Dahl,V. (1982). On database systems development through logic. **ACM Transactions on Database Systems,** 7(1), 102-123.

Futo,I., Szeredi,J. & Szenes,K. (1981). A modelling tool based on mathematical logic: T-PROLOG. **Acta Cybernetica,** 5(3), 363-375.

Futo,I. & Szeredi,J. (1981). T-PROLOG: a very high level simulation system. **User Manual,** Szamitastechnikai Koordinacios Intezet, 1011, Budapest I. Iskola Utca 8, Hungary.

Gaines,B.R. (1981). Logical foundations of database systems. In Mamdani,E.H. & Gaines,B.R. (Eds.), **Fuzzy Reasoning and its Applications.** Academic Press, London. 289-308.

Kowalski,R.A. (1979). **Logic for Problem Solving.** North Holland, New York.

Michalski,R.S. & Chilausky,R.L. (1981). Knowledge acquisition by encoding expert rules versus computer induction from examples: a case study involving soybean pathology. In Mamdani,E.H. & Gaines,B.R. (Eds.), **Fuzzy Reasoning and its Applications.** Academic Press, London. 247-271.

Michie,D. (Ed.) (1979). **Expert Systems in the Micro Electronic Age.** Edinburgh University Press, Edinburgh, UK.