ICOT Trip Report

J.W. Lloyd
University of Bristol

December 14, 1988

The Fifth Generation Computer project has come to an interesting stage with just three years to go before the end of the 10 year project. In the following, I will present my impressions of the project gained from this trip, concentrating on those aspects of the project closest to my own research interests, which are mainly concerned with applications languages.

1 Discussion

It is evident to me that considerable progress has been made on the project since my last visit in 1986. Furthermore, it is heartening to see a consensus emerging amongst the logic programming community, including ICOT, of the likely architecture for fifth generation computers. Surprisingly, they are likely to be much more like general-purpose computers than most people expected when the project first started. What has emerged is that highly optimised special-purpose logic machines are not likely to be significantly faster for running logic programming languages than more general-purpose machines. I think that, if it is confirmed in the next 3 years, this is a highly significant result and also a good one. It will mean a smoother introduction of logic machines, as users will be more easily able to integrate their logic programming applications with their existing applications.

Next I would like to comment on the applications side of ICOT research. I still feel, as I did in 1986, that this is the weakest part of the project. First, there is the technical problem of getting applications languages, which mostly exploit OR-parallelism, to run very efficiently on machines for which GHC is the kernel language and which mostly exploit AND-parallelism. Some progress has been made on this problem, but there is certainly much more to do. This is a crucial problem, since ultimately the success of the project depends upon the usefulness of the applications which come out of it.

The next point concerns the applications languages themselves. I believe

that the majority of applications languages which we have available today, especially those based on Prolog, are far from satisfactory. I will not go into the reasons for this in detail here, since they have already been explained in a series of seminars which I have given at ICOT and FGCS'88 and also in a series of papers which I have left at ICOT. But, in essence, the problem is that Prolog and its derivatives have considerable semantic difficulties, especially with their declarative semantics. These problems show up in many Prolog constructs, such as *var*, *assert*, *retract*, cut and also in the lack of the occur check. This is not just a matter of concern to theoreticians, because these semantic problems greatly affect our chances to verify, debug, partially evaluate, etc. logic programs and also greatly decrease the potential for exploiting parallelism in these programs. In the period since Prolog was designed 15 years ago, we have achieved considerable understanding of the semantic basis of logic programming languages. This, combined with our experience in implementing logic programming languages and their application in a wide variety of domains, strongly suggests that the time is ripe to redesign the applications languages with a view to providing them with greatly improved declarative semantics and at least the current level of expressive power. I strongly believe that any effort put in this direction will pay great dividends in the long run.

Another area which I believe ICOT needs to look at is knowledge assimilation. There was some early work on this at ICOT which was reported at FGCS'84, but it seems not to be studied at the moment. A knowledge base system consists of four different kinds of theories: knowledge bases, interpreters, integrity constraint theories and assimilators. When it is necessary to update a knowledge base, an assimilator must interact with the knowledge base and the appropriate integrity constraint theory to achieve the required update. Typically, an update request (in the following case, a deletion) takes the form: the following formula (often just an atom) is currently a logical consequence of the knowledge base, so modify the knowledge base in order that it no longer be a logical consequence. This is an important problem, which has not received the attention from the logic programming community it deserves. In particular, I do not know of any work at all on this problem for knowledge bases allowing negation, which is the usual situation.

Incidentally, understanding the way assimilators interact with knowledge bases requires an understanding of meta-programming. There are many other parts of the project which also require a good understanding of meta-programming. I agree entirely with ICOT's current emphasis on this topic. As pointed out above, this is an area which Prolog and its derivatives do

badly and any work towards putting meta-programming on a better basis will be an important contribution to the success of the project.

Finally, let me say once again I am impressed with the overall progress of the project and I look forward to inspecting the prototype fifth generation computer which will be unveiled at FGCS'91!

2 Acknowledgements

First I would like to thank Dr. Fuchi for his kind invitation to visit ICOT for the second time. Dr. Iwata was very helpful in taking care of the non-technical details of the trip. For stimulating technical discussions, I would like to thank Dr. Furukawa, Dr. Hasegawa, Dr. Ueda, Dr. Chikiyama, Mr. Fujita, Dr. Murakami, and Mr. Seki. As during my first visit, Mr. Seki once again was very kind in looking after me. Finally, I would like to thank everyone at ICOT for their considerable hospitality during my visit.

RESEARCH RESUME

Name John Wylie LLOYD

Current Appointment

Senior Lecturer in Computer Science at Melbourne University.

Research Interests

My research interests are in Logic Programming and its impact on Database Systems and Artificial Intelligence.

Logic Programming is currently attracting substantial interest, much of which is due to the wide publicity given to the Japanese 5th Generation Computer Project. However, there are very strong technical reasons why Logic Programming will play an increasingly important role in Computer Science. The most fundamental reason is due to the fact that logical inference can be regarded as the fundamental unit of computation. From this perspective, many diverse areas of Computer Science can be unified and simplified. For example, PROLOG is now commonly used as the core of database systems, as a systems programming language and as an Artificial Intelligence programming language.

The following is a list of the specific research topics in which I am interested.

(a) *Theoretical Aspects of Logic Programming*

There are many important unresolved theoretical issues in Logic Programming. My interests here are mainly concerned with negation (esp. negation as failure), completeness, control and concurrency.

(b) *Theoretical Aspects of Deductive Database Systems*

A firm theoretical foundation for deductive database systems is just emerging. My interests here concern soundness and completeness of query evaluation, logic as a query language, integrity constraints and various update issues.

(c) *Implementation of Deductive Database Systems*

PROLOG provides a satisfactory query evaluator for deductive database systems. However, PROLOG needs to be augmented with indexing schemes, query languages and so on, before it can properly be regarded as a deductive database system. My interests here concern clause indexing, query optimization, integrity constraint implementation and query language design and implementation.

(d) *Programming Environments*

One of the weaknesses of current PROLOG systems is their programming environments. This is a major reason why many programmers still prefer LISP to PROLOG. I have recently been investigating declarative error diagnosis, starting from the work of Shapiro and Ferrand. I have written a debugging system which finds errors in extended syntax programs by asking questions of the programmer. Typically, it asks if an atom is satisfiable (in the intended interpretation) and sometimes asks for values of variables. The programmer does not need to have any understanding whatsoever of the computational behaviour of the PROLOG system. I believe such debugging systems are crucial for the future development of PROLOG because they offer the potential for writing and debugging programs at a very high and purely declarative level.

Machine Intelligence Project

I am one of the principal investigators for the Machine Intelligence Project at Melbourne University. This project is being funded by the Department of Science. One major task of this project is the development of a deductive database system based on MU-PROLOG, whose performance will be competitive with state-of-the-art relational database systems. The other major task is the development of a compiler version of MU-PROLOG.

Book on Foundations of Logic Programming

I have written this book to provide the first account of the mathematical foundations of Logic Programming. The purpose of the book is to collect, in a unified and comprehensive manner, the basic results of Logic Programming previously available only in widely scattered research papers.

Associations with Journals and Conferences

Editorial Advisor for The Journal of Logic Programming

Member of Editorial Board for Information Technology: Research & Development

Member of Program Committee for Third International Logic Programming Conference to be held in London in 1986.