A Visit to ICOT

Ian Foster

Department of Computer Science
Imperial College
London, England

June 14th - July 10th, 1987

1 Introduction

I was invited to visit ICOT to discuss issues connected with systems programming in parallel logic languages and the design of programming systems to support these languages on parallel computers.

I was very pleased to accept this invitation, as I was keen to see at first hand how work on these and other topics was progressing at ICOT. The similarities of our interests suggested that fruitful collaboration was likely. In addition, I felt it to be an honour to be invited, as ICOT is widely recognized as a centre of excellence in logic programming research.

I had been invited to visit ICOT's First Research Laboratory. Until recently, this was responsible for research on the design and implementation of ICOT's 5G kernel language, KL1, and operating systems to support this language. These are essentially my research interests. Recently, however, responsibility for these topics has been moved to the Fourth Research Laboratory, which is starting to design and build systems based on KL1. The First Research Laboratory is now conducting more basic research concerned with problem solving and inference, intelligent programming and the design of the next generation of kernel language, KL2. This reorganization was explained to me by Dr Iwata, who gave me an introduction to ICOT and more fully by Dr Tanaka of the First Laboratory. After this introduction I realized that I would have much to discuss with members of both the First and Fourth Laboratories.

2 Presentations by ICOT Researchers

The first week and a half of my stay were largely taken up with presentations by members of the First and Fourth Laboratories on the current status of ICOT research. These were all extremely interesting and resulted in some stimulating discussions.

2.1 First Laboratory

Dr Furukawa described his recent work on unfolding rules for GHC. This work answers earlier criticism concerning the semantics of the transformations performed by partial evaluation. Dr Furukawa suggested that simple algorithms for the application of these rules can be developed. Mr Fujita described his self-applicable, incremental partial evaluator for Prolog, which he hopes to extend to handle GHC. I was interested by his observation that partial evaluation of programs that manipulate complex data structures such as streams requires some type inference system to avoid generating superfluous code.

Mr Matsumoto described the layered stream method which he and Mr Okumura had developed and applied to searching problems. Though producing more complex programs, the method appeared to lead to substantial increases in potential parallelism and also provides useful structural information which could permit more efficient implementations of searching algorithms. The increase in shared data appeared to limit its application to shared memory machines, however. It also makes compile-time garbage collection harder.

Dr Murakami gave a presentation of his verification method for GHC programs, based on a Hoare-like axiomatic system. This was an exciting piece of work which I hope will be further developed. I found his requirement that relations not be called with output mode arguments instantiated interesting: earlier versions of PARLOG imposed the same restriction to facilitate implementation. Violation of this condition was flagged as a run-time error. In fact, it was the example of GHC that inspired the incorporation of general output unification into PARLOG. As this feature seems to be rarely used, it may be that it can be removed from these languages, simplifying both their verification and implementation.

Dr Ueda described the current status of GHC. We discussed problems involved in compiling and executing GHC and PARLOG. The space reclamation problem is clearly important. New syntax to denote stream operations and block compilation are hopeful avenues for future research. Dr Ueda also mentioned some issues being considered in the design of KL2.

Dr Tanaka presented his work on parallel operating systems based on virtual machines and arranged for a demonstration on the multi-PSI simulator. He then described the current status of KL2.

2.2 Fourth Laboratory

Dr Taki gave an overview of the multi-PSI project and described simulation studies on the multi-PSI. I was also interested by his description of the PSI-II machine. The high performance of this machine will permit useful experimental studies of problems such as load balancing, metacontrol mechanisms and garbage collection. The relatively low performance of the current distributed implementation makes it hard to evaluate the likely cost of these mechanisms. Nevertheless, projected communication costs emphasize the need for intelligent load-balancing mechanisms.

I discussed with Mr Ichiyoshi the distributed implementation of FGHC on the multi-PSI. I was already familar with the work, having read his ICLP paper, but he was able to describe recent optimisations to their distributed unification scheme. We discussed alternative approaches to the implementation of distributed unification and possible means of optimising communication, both through specialized hardware and compilation.

Dr Goto gave a detailed presentation of the PIM machine design and in particular his work on cache simulation and design. The simulation results were very interesting. I was particularly interested in his results demonstrating the advantages of the MRB scheme. The low memory contention rates were very encouraging.

I then discussed the KLI-b instruction set with Mr Kimura. I had read the paper describing this work, and had a number of criticisms: I felt it was unduly complicated compared with a Flat Parlog instruction set I had designed with Steve Taylor. However, since preparing the paper describing it a number of improvements have been made, with the result that the two instruction sets are now very similar. This is perhaps not surprising, as KL1 and Flat PARLOG are essentially equivalent languages, but it was encouraging to see similar solutions being adopted. We discussed briefly the use of decision graph compilation techniques to improve the quality of compiled code. I was able to familiarize Mr Kimura with some recent work by Steve Taylor on this subject which appears to provide an excellent basis for efficient KL1 compilation.

Dr Chikayama gave a detailed presentation of both his MRB scheme and the current status of KL1-c. I was interested to see that the KL1-c specification is very similar to the extended PARLOG language I am using for systems programming.

Dr Sato presented the current status of PIMOS design. In particular, he described their approach to managing communications between application programs and operating system so as to ensure application program behaviour could not compromise operating system safety.

Mr Nakajima gave a detailed description of the design of PSI-I and PSI-II. I was interested to see that PSI-II is primarily a Prolog machine: the design could have been somewhat simpler if only KL1 were to be supported. Nevertheless, a machine that supports both Prolog and KL1 will permit interesting experiments in mixed language processing.

Finally, Mr Ishibashi gave an overview of ESP and SIMPOS and demonstrated the PSI machine.

3 Presentations to ICOT

During my stay at ICOT I gave three presentations to ICOT researchers. Two of these were concerned with language design and implementation issues and the other with my work on parallel programming systems.

The first of my presentations, entitled "Efficient Metacontrol in Parallel Logic Languages", described some recent work concerned with the implementation of metacontrol mechanisms in parallel logic languages. I described an efficient implementation scheme for such mechanisms on uniprocessors and presented benchmark results that permitted comparison with an alternative scheme based on program transformation. I also showed how uniprocessor mechanisms could be exploited in a loosely coupled implementation to provide efficient distributed implementations of metacontrol mechanisms without complex message protocols.

The second of my presentations (chronologically the third) was entitled "Flat PARLOG: A Basis for Comparison". It described some joint work with Steve Taylor of the Weizmann Institute on the design and implementation of a flat subset of PARLOG. The implementation details were not of particular interest to ICOT researchers, as their own KLI-b instruction set is very similar to our Flat PARLOG machine. I therefore concentrated on the presentation of benchmark results permitting comparison of Flat PARLOG and another parallel logic programming language, FCP. These results are of considerable interest to ICOT researchers, as the essential equivalence of Flat PARLOG and FGHC means that the benchmark results can also be applied to FGHC. The results also indicated the likely cost of utilizing a parallel rather than sequential evaluation strategy for guard primitives. This is desirable for semantic reasons, particularly if optimizing compilers may reorder guard tests. Some discussion ensued concerning probable performance difference on parallel machines. I indicated that the benchmark results only permitted comparison of single processor performance, but agreed that FCP's more complex distributed unification algorithm would tend to result in higher communication costs.

My third talk was entitled "PARLOG Programming System", and described the operating system/programming environment for PARLOG that I have been developing for some time. PPS is an operating system intended to support parallel logic programming on parallel computers. Currently, only a prototype has been implemented: this runs on SUN workstations and exploits SUN features such as its window system to provide a friendly user environment. Nevertheless, a substantial subset of operating system functionality – such as secondary storage management, exception handling and computation management – is implemented in PARLOG.

Following my initial discussions with Dr Tanaka on KL2, he had suggested that I consider how a system such as PPS could support knowledge representation and knowledge processing in application programs. As these appear particularly important issues at this stage of PIMOS design, I gave only a brief overview of PPS structure and facilities in my talk and concentrated on how an operating system (and PPS in particular) could provide support for these activities. These issues are discussed in more detail in a later section of this report.

I also gave a demonstration of the PPS prototype on a SUN workstation.

4 Visits to other Institutions

I was fortunate to be able to visit both Fujitsu Laboratories and the Science University of Tokyo during my stay at ICOT. At both institutions I presented a survey of research in the PARLOG group at Imperial College and described my own work.

Mr Hattori introduced me to the research of his group at Fujitsu. They are involved in the development of a large FGHC application (a parallel router) and in addition are investigating the implementation of parallel hardware for ICOT's PIM. Mr Kishimoto gave a very interesting presentation on an evaluation of their parallel routing application. Their conclusion that tail recursion optimisation was not particularly effective was interesting, as a similar analysis of a Flat PARLOG compiler that I had conducted gave similar results. We had an interesting discussion concerning possible compiler optimisations of FGHC programs. I briefly described techniques that I thought could prove effective on applications with simple producer-consumer relationships, such as compilers.

However, these did not appear applicable to the parallel routing application. It is interesting to consider which is the more typical application.

I was invited to the Science University of Tokyo by Dr Mizoguchi. He presented the work of his laboratory and several of his students demonstrated systems. I was particularly interested by their work on qualitative reasoning and its implementation in both a constraints based language and FGHC. I also enjoyed a Prolog graphics demonstration, particularly the 'Winking Madonna'!

5 Discussions with Fourth Laboratory

Throughout my stay at ICOT I had a number of discussions with members of the Fourth Laboratory. I shall just note the more interesting points raised here.

Dr Chikayama and his colleagues are working on the design of PIMOS, the operating system for ICOT's parallel inference machines. I had several discussions with members of this group. It was interesting to see where our views on important issues coincided and diverged.

Our approaches to language design were very similar. The sho'en feature described by Dr Chikayama is essentially equivalent to the extended PARLOG metacall described in my paper at the 4th ICLP. However, Dr Chikayama and his colleagues seem to have a rather different view as to its intended implementation and use.

Our views on the degree of kernel support that should be provided by a parallel implementation differed somewhat. Dr Chikayama and his colleagues seemed keen to implement significant functionality at this level. Their sho'en would thus be responsible for resource management and control over many processors. In addition, individual processes within a sho'en could be assigned different priorities.

I on the other hand favour a more light-weight metacall, perhaps implemented initially as a uniprocessor mechanism only. A parallel implementation then only needs to implement distributed unification. I find this approach very attractive, because of its inherent elegance and simplicity. Of course, it is then necessary to program load balancing, code mapping and distributed metacontrol in a parallel logic language. However, I believe that this is desirable, as these issues are poorly understood at present and it is important to encourage experimentation. As I showed in my first talk, distributed control can exploit uniprocessor metacontrol mechanisms. My PPS architecture also incorporates code mapping and load balancing mechanisms.

Naturally, it is more efficient to implement metacontrol, load balancing and resource management mechanisms in a language kernel. However, this reduces the amount of the operating system that is implemented in a parallel logic language. This complicates the parallel implementation and discourages experimentation with alternative mechanisms.

Another issue which was discussed on several occasions was that of deadlock detection. Perhaps because the semantics of GHC as originally described do not distinguish failure and deadlock, this issue does not appear to have been considered in as much detail as other issues. However, once the concept of task is introduced via sho'en or metacall, deadlock detection becomes extremely important. It is for this reason that I support it at a very low level in my metacall implementations.

I had several interesting discussions with Mr Ichiyoshi concerning distributed unification algorithms and distributed deadlock detection. He described a termination detection algorithm that he had developed and proved correct. Unfortunately, though very elegant, the algorithm and proof of correctness appeared essentially equivalent to one due to Dijkstra. Mr Ichiyoshi pointed out a weakness in the distributed deadlock detection scheme I had presented and this led me to formulate a revised distributed unification algorithm that would permit deadlock detection within a computation. The new algorithm introduces some extra communication however.

In further discussions with Dr Goto, I described an alternative implementation technique for FGHC on shared memory machines. We discussed the applicability of this technique (the usefulness of which depends on the relative costs of lock and memory access operations). Dr Goto offered to use existing simulation data to evaluate it.

6 Collaborative Research

Following his presentation of KL2, Dr Tanaka suggested that I consider how an operating system such as PPS could provide support for the knowledge processing applications which the Fifth generation computer is intended to support. I talked about these issues in my presentation to ICOT and subsequently discussed them with Dr Tanaka and Dr Ueda. Jonas Barklund, visiting from Uppsala University, also provided some very valuable comments.

The ability of parallel logic languages to express parallelism and to support symbolic processing applications has motivated their selection by ICOT as a kernel language for their Fifth Generation computers. Techniques for expressing the AI problems that these computers are intended to solve in these languages must now be developed. Several approaches to the formulation of such problems in parallel logic languages are being investigated. The *direct* approach attempts to implement solutions to AI problems directly in parallel logic languages. The *embedded* approach attempts to reimplement existing methodologies and languages (such as object-oriented languages, or Prolog) by providing compilers to parallel logic languages. The *interface* approach aims to provide interfaces to existing knowledge representation systems, such as DBMS, KBMS and Prolog systems. ICOT's Fourth Laboratory seek to take a fourth approach, designing a new language (KL2) that provides powerful knowledge representation and problem solving capabilities and in addition is compilable to KL1. This approach has great promise, but perhaps is the most difficult avenue to pursue in the short-term.

Whatever approach is taken to knowledge representation and processing, the knowledge must eventually be expressed (in some form or another) as parallel logic language programs. It is therefore very important that it be easy to experiment with changes to such programs, the inference mechanisms used to evaluate them and their structure. It should therefore be possible to describe such changes using parallel logic language programs.

One approach to providing such metaprogramming functions is to program them on top of a conventional programming system using metainterpretation techniques. This was the approach taken in Mandala. However, in practice this lead to unacceptable overheads. Also, metainterpretation, by simulating change in the language, separates the tasks of *describing* and *implementing* change. The implementation of change impacts on distributed operating system functions such as code mapping. The description and implementation of change should therefore be closely linked.

I therefore believe that it is necessary to provide support for metaprogramming functions in the operating system itself. They can then be made available to the programmer in terms of an enhanced parallel logic language. This language can be used to program tools for maintaining, structuring and modifying applications programs represented as parallel logic programs. It can also be used for writing these programs, as of course many AI-like applications require the same ability to describe and reason about change.

These considerations led me to reconsider the function and design of PPS. Comments from Mr Barklund stimulated me to redesign the system's metaprogramming facilities to provide a purer declarative semantics. I also removed the previous built-in inheritance mechanisms and provided tools (essentially reflection mechanisms) which permitted the programmer to define his own inference mechanisms (including inheritance). The result of these modifications is an operating system that supports a parallel logic language with extensions for metaprogramming. This language is named PARLOG+ to distinguish it from the PARLOG language on which it is based. Briefly, PARLOG+ permits a program to access first order representations of other programs, to construct alternative states in which modified versions of these programs apply and to specify that some modified state be made available to other programs upon successful termination. Concurrency control mechanisms ensure conflict does not occur when state-changing programs execute concurrently. PARLOG+ programs can also define non-standard inference mechanisms.

These extensions to PPS and the preliminary definition of PARLOG+ seem to me to indicate a promising direction for work on operating systems designed to support parallel logic languages. Metaprogramming functions are necessary if effective program development tools are to be built and AI applications programmed. Low-level support for these functions ensures that efficient execution is not compromised by unnecessary layers of interpretation. Also, as these mechanisms are implemented in a parallel logic language, they are not difficult to implement on parallel machines.

Another issue that I discussed with Dr Tanaka and Dr Ueda was how programs could be provided with the ability to reason about their own behaviour and situation. We did not reach any firm conclusions as how such functionality might be introduced into parallel logic languages. However, I was able to show how PARLOG+ programs could take advantage of limited inforamtion about their situation to define (for example) load balancing shells.

The facilities provided by the redesigned PPS are summarized in a draft document written by me at ICOT. This describes the PARLOG+ language and shows how it can be used for metaprogramming and to implement alternative inference mechanisms – in particular, inheritance and query-the-user evaluation mechanisms. It also illustrates the application of simple reflective programming techniques and shows how PPS can support other languages.

7 Future Collaborative Research

I plan to further investigate the question of distributed deadlock detection and will attempt to prove my scheme correct when I return to Imperial College. Experimental studies currently being performed by Mr Ichiyoshi and his colleagues will permit me to quantify the overhead introduced by the additional communication.

Dr Tanaka and I plan to continue discussion of issues connected with operating system support for metaprogramming in parallel logic languages. I will revise the draft document I wrote at ICOT, and plan to eventually release it as an ICOT Technical Report.

8 Impressions of ICOT

Several of my colleagues at Imperial College had visited ICOT in the past and had spoken very warmly of their experiences there. Their only warning to me was that the laboratories were rather crowded! However, ICOT's recent expansion meant that this was not a problem. I was very happy to be given a desk with a view over Tokyo harbour. In general, I find that open-plan environments such as that at ICOT rather noisy, but perhaps because ICOT researchers are quite polite (or perhaps because I don't generally understand Japanese!), this did not seem to be a problem. On the contrary, it appeared to encourage an open and friendly atmosphere that I am sure is good for research.

My impressions of the research being performed at ICOT were very favourable. It was very exciting to see so many able researchers working on what I believe are important problems. ICOT has already achieved significant results, and judging by progress made to date, the FGCS programme will be a success.

One of my few concerns is an apparent tendency to push functionality that might be better realized in software into hardware. The problems being tackled by ICOT researchers are new and as yet not well understood. We are still learning how to implement parallel logic languages and how to build operating systems to support programming in these languages. Clearly, if the aims of the Fifth Generation Project are to be met, it is necessary to begin building parallel implementations now, using solutions developed to date. However, implementing these solutions in hardware may discourage a search for even better solutions.

For example, hardware can be used both to reduce the cost of frequent context switching and to reduce memory usage. But it is also important to investigate compilation techniques that perform compile-time garbage collection and reduce communication and context switching by increasing granularity. Both techniques will require global analysis and thus appear difficult; nevertheless, such techniques are likely to be cheaper than hardware in the long run.

Similarly, though distributed resource management functions are important, it is also important to consider what facilities an operating system must provide to support logic programming. The full benefits of logic programming may not be realized if the operating system forces the programmer to think in too imperative terms. PPS might suggest alternative ways of structuring the operating system.

Finally, I note a discrepancy between the hardware being built to support KL1 and applications built in higher-level languages. I was surprised to see that several of the applications groups rely heavily on languages incorporating enhanced unification mechanisms: such languages are unlikely to be efficiently compilable to KL1. As design and implementation of KL1 machines begins, it is clearly essential that techniques for implementing AI applications on these machines be developed. The work of the First Laboratory thus seems very important to the success of the Fifth Generation project.

9 Conclusions

This was my first visit to both ICOT and Japan, and I found both wonderful experiences. I consider myself very fortunate to have been able to spend four weeks in the midst of the Fifth Generation Project. I gained a lot from this experience. I hope that some of our discussions and my work on PPS will prove useful to ICOT people.

I think it is important for both logic programming and computer science as a whole that ICOT succeed in achieving its goals. I was thus very happy to find how much progress is being made at ICOT. I was also impressed by the broad range of research topics being pursued. Many of these topics are of intense interest to me personally, so I look forward to hearing of the future results of the project.

10 Acknowledgements

I would first of all like to thank Dr Fuchi for inviting me to visit ICOT. It was a very stimulating experience and will, I hope, lead to fruitful future collaboration.

I am also grateful to Dr Tanaka for arranging my visit and organizing a very interesting schedule, and to Dr Itoh for making me welcome in his Laboratory.

It was an unexpected pleasure to meet Ken Sato at ICOT on my first day: I had last seen him at Imperial College, when he visited two years ago. He and Dr Tanaka took very good care of me during my stay. Ken Sato was also kind enough to teach me the 5 essential phrases which, I am ashamed to say, still constitute the bulk of my Japanese vocabulary at the end of my stay.

I found everyone at ICOT very friendly, and always ready to help in whatever way required. I particularly appreciated the efforts of Mr Ichiyoshi, Miss Nakamura, Mr Rokusawa and Mr Sato, who arranged some very enjoyable visits outside Tokyo, enabling me to see a little of the rest of Japan.

I must also thank Mr Takagi for permitting me to use the Sun for demonstrations and granting my excessive demands for disk space!

Finally, I would like to thank Dr Iwata and Miss Nakamura of the research planning department for their many efforts on my behalf, both before and during my stay.

Curriculum Vitae

Name:	Ian Foster
Nationality:	New Zealand
Date of Birth:	1st January, 1959
Present Position: (1985-1987)	Research Associate, Department of Computing, Imperial College, London.
Research Areas:	Implementation and application of parallel logic languages, with particular emphasis on the use of such language for systems programming.

Qualifications:

B.Sc. Honours (First Class) in Computer Science, University of Canterbury, New Zealand. [1978-1980]. (A four year degree, completed in three).

Visiting Positions:

February-March, 1987: Visiting Scientist, Department of Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel.

April, 1986: Visiting Scientist, Argonne National Laboratory, Illinois.

October-December, 1985: Visiting Professor, University of Rome and Consiglio Nationale della Richerche, Rome, Italy.

Refereed Publications:

``Logic Operating Systems: Design Issues". To appear: The 1987 International Conference on Logic Programming.

``A Programming Environment for Concurrent Logic Programming". To appear.

``Flat Parlog: A Basis for Comparison". Submitted for publication.

``An Abstract Machine for the Implementation of Parlog on Uniprocessors". Submitted for publication.

``A Sequential Implementation of Parlog". In: The 1986 International Conference on Logic Programming.

``A Declarative Treatment of Secondary Storage". In: The 1986 International Symposium on Logic Programming.

Invited Papers, etc

``A Declarative Environment for Concurrent Logic Programming". In: TAPSOFT '87.

Panelist, ``Logic Programming for Systems Programming". 1986 International Symposium on Logic Programming.

Industrial Experience:

1983-1984: Scientist, BRS Europe. Research and development into advanced information storage and retrieval systems.

Awards:

Dux, Wellington College, 1977
Senior Scholarship, University of Canterbury, 1980