Vth ICOT

VISITING RESEARCHER'S REPORT

Mark Grundy,

Center for Information Science Research,

The Australian National University

4 May — 7 June

1992

Introduction

This report describes my visit to ICOT, which took place from 4th May to 7th June inclusive, 1992. It was timed to coincide with the visit of Dr. John Slaney, also of CISR, who visited ICOT for the first ten days of my stay. My travel and living expenses for this stay were kindly funded by ICOT, and I am very grateful to ICOT for their generous support and warm hospitality during my visit.

Goals

There were several purposes for my visit to ICOT. They are listed below, in no particular order of importance:

- To become better acquainted with MGTP and related ICOT theorem-proving software.
- To gain expertise in the KL/1 programming language, the SIMPOS development environment, and related new ICOT technology, in a situation where close consultation was possible with the experts who produced it.
- To collaborate with Dr. Slaney and ICOT's Fifth Laboratory in work on the development of a *semantically constrained* theorem prover, to be adapted from ICOT's *ground* and *non-ground* MGTP inference engines, and based upon a model of the theorem prover SCOTT developed under the ANU/Argonne National Laboratory (USA) collaboration agreement.
- To develop demonstration software for the 1992 Conference on Fifth Generation Computing Systems (FGCS-92), held in Tokyo from June 1 through to June 5.
- To present a demonstration at FGCS outlining ICOT's collaboration with the Australian National University.
- To briefly visit the Numazu laboratories of Fujitsu Ltd, with whom the ANU has a collaboration, research and development agreement.

Report Structure

The remainder of this report deals with the details of my visit, and in accordance with the agreement by which my visit was funded, offers constructive criticisms based on my experiences at ICOT.

The rest of this report is arranged into the following major sections:

Schedule contains details of how my time was spent at ICOT.

Comments contains my opinions and suggestions, and is divided into the following sub-sections:

> *SC/MGTP* deals with the development of the semantically constrained theorem prover SC/MGTP, and related software. It also includes comments on the MGTP provers.
>
> *SIMPOS* contains details of my experiences with the SIMPOS operating system, the PSI-3 personal workstations and other related experiences.
>
> *KL/1* addresses my experiences with this language and includes comments and suggestions for future use and development of KL/1.

Conclusions contains my final comments.

Schedule

This table below outlines the schedule of my time at ICOT:

Week 1: Discussions were held with Drs. Hasegawa and Slaney, Mr. Fujita and Mr. Koshimura regarding the design principles of a semantic constraint theorem prover. During these discussions Dr. Slaney described the semantic constraint prover SCOTT, which was developed in as a joint collaboration of ANU and Argonne National Laboratories. Dr. Hasegawa described the operation of non-ground MGTP, and Mr. Fujita spoke about the operation of ground MGTP.

At this time, a preliminary strategy was formulated for the development of a semantic constraint prover using both ground and non-ground MGTP software. To this we would add KL/1 interface software designed and implemented by myself, and we would study the system with a graphical performance analysis program which I was to design and develop at ICOT under X11/Unix™.

We targeted the initial test problem domain of the semantic constraint prover as the Lukesiewicz Pure Implication calculus, with which SCOTT had already had notable success.

During this week Dr. Slaney and I became acquainted with the PSI-3 workstations, and used Internet to import development support software from ANU. Mr. Fujita introduced me to SIMPOS, and I began writing initial KL/1 code for the semantic constraint interface. This entailed considerable study of SIMPOS and KL/1 documentation,

including learning to use the powerful PEMACS editing system.

On the weekend, Mr. Fujita kindly showed Dr. Slaney and myself the sights of Kamakura and Yokohama, including visits to the temple of Daibatsu Buddha, and to a Shinto shrine where many beautiful Japanese art treasures were on display.

Week 2: In this week development with the semantic constraint prover (SC/MGTP) continued. Regrettably, Dr. Slaney had to return to Australia to prepare for CADE-11, but we maintained contact with him by electronic mail. More discussions were held during which we discussed MGTP operation in more detail, and refined our notions for the development of SC/MGTP. We also discussed possibilities for demonstration of this software at FGCS-92, and assigned various development tasks to the researchers involved in the collaboration.

During this week my development of the KL/1 interface code (CHECKER) continued, and at the same time I also began working on the graphical performance software XSCGRAPH. We wanted XSCGRAPH to provide a performance analysis tool for future research, but also to offer an interesting, informative display suitable for demonstration at FGCS-92.

Week 3: A formal specification for the interface program CHECKER was written and distributed to the researchers involved in development of SC/MGTP. An informal description of the semantic constraint method, as applied to MGTP, was also written and distributed. Mr. Koshimura began adapting non-ground MGTP to use the interface specification, and Mr. Fujita commenced similar adaptations to ground MGTP.

Work continued with both CHECKER and XSCGRAPH software, and we produced the first alpha versions of the software in this week.

In this week I was also able to visit the Fujitsu laboratories at Numazu, where I presented a seminar entitled: *Toward an Effective Tableau Metaprover*, which describes my design for the system PYTHIA, a model generation metaprover system. There I met with Mr. Minami and Mr. Sawamura, and we discussed the rôle of automated reasoning in Japan, with particular regard to its relationship with industry.

My wife had the opportunity to visit Japan briefly, and met with Dr. Hasegawa, Mr. Koshimura and Mr. Fujita. She was very impressed with both Tokyo and with Mr. Fujita's tireless quest to show us many interesting Japanese entertainments.

I met Dr. Chikayama in this week, and discussed details of KL/1 implementation with him. Summaries of these and subsequent discussions with Dr. Chikayama are included in the section KL/1 of this report.

Professor McRobbie also visited ICOT during this week, and we spoke briefly about the design for SC/MGTP, and about demonstration opportunities at FGCS-92.

Week 4: Testing for the alpha versions of SC/MGTP and XSCGRAPH took place during this week. As a result of insights produced by the testing we subsequently revised the formal interface description, and implemented the changes during this week.

Further testing continued, and final preparations were made for demonstrations at FGCS–92.

During this week I had the opportunity to speak with Dr. Furukawa about underlying semantics of KL/1, and about the development of coding disciplines specifically suited to the language.

Week 5: This week was primarily focussed around FGCS-92. There, we presented demonstrations of G/MGTP, N/MGTP, as well as ANU's work with SCOTT and our preliminary findings on SC/MGTP.

At the same time, research into development and improvement for SC/MGTP continued, although much slowed by FGCS-92 commitments. In spite of very heavy work-loads from FGCS-92, Dr. Hasegawa, Mr. Fujita and Mr. Koshimura continued to be very active in the refinement and development of SC/MGTP, with the result that regular observers at the SC/MGTP demonstration were able to see daily improvements in the developments of our research.

Comments

SC/MGTP

Status

At the time of writing this report, the SC/MGTP code is in place, is debugged to beta level, and is ready for extensive testing on a range of preselected problems.

XSCGRAPH is written and fully debugged, and is ready for use with SC/MGTP and SCOTT. Additionally, it is expected to prove useful for analysing the performance of N/MGTP, G/MGTP, OTTER and other provers.

Preliminary results with SCOTT demonstrate efficiency increases of 2—4 times with Lukesiewicz Pure Implication problems, and I feel that similar results will be likely for SC/MGTP in the immediate future.

Impressions

It is clear from my experiences with MGTP in both ground and non-ground forms that this software is among the most sophisticated and impressive at ICOT. Great effort has been made to produce automated reasoning systems that are at once efficient, highly parallelisable and with useful potential for both academic and industrial applications, and I feel that these goals have certainly been achieved by the MGTP systems. During my attendance at FGCS-92, I formed the impression that many other leading foreign researchers in automated reasoning feel as I do.

Overall, I feel that to have entered the world arena of theorem proving at such a high level in such a short time, and to have done so using an experimental new programming language, and on prototype hardware, is a tribute to the vision that inspired ICOT, and a credit to both the management and to the researchers involved. I personally feel very privileged to have collaborated with such a group, and look forward eagerly to further opportunities to extend this collaboration.

Future Developments

Clearly, the applications that are already being made of MGTP show that it is already very sensitive to real-world problem solving issues, and I think that this is a great virtue. Although research into MGTP is still continuing, I feel that now is an opportune time to consolidate the work that has been done to date, and to make it more readily available to the theorem proving community outside Japan. This is consistent with the philosophy of ensuring that MGTP remains a real-world prover as well

as a tool for future research investigations, and is in keeping with ICOT's stated goal of providing free software to the international community.

To put my comments into proper context, it may prove useful to make a comparison with another well-known prover, Argonne's OTTER. OTTER is an internationally recognised standard against which other provers are measured. I feel that this is because it is fast, can be run on a variety of computing systems, and is well documented at every level from the theory down to the fine details of implementation. I think that these features make it very attractive for researchers outside of Argonne to experiment with OTTER's algorithms, and such external interest is also very useful to OTTER's development.

Because of its speed and generality, I think that MGTP is also a candidate for becoming a standard against which other provers are compared. But I believe that in order to achieve this aim, two further goals should be set:

Firstly, I feel that MGTP needs to become available for execution on general computing architectures, and in particular, on a variety of common parallel computers. Such execution can only be possible if KL/1 is written for a broader group of target architectures than just PIM computers. I deal with this matter in greater detail in the *KL/1* section.

Secondly but more immediately, other researchers can become familiar with MGTP through collaboration at ICOT. Already there is sufficient documentation for visiting researchers to investigate the basic algorithms of MGTP software. All that remains is to make the implementation itself easier to understand by documenting it thoroughly. To document and clarify MGTP implementation is relatively short work, and I believe that it will be well worth the effort in terms of the benefits it brings to future collaborations.

With regard to future collaborations with ANU, I feel that the opportunities in this area are very rich. Although the performance of SC/MGTP is not yet quantified, I feel that there is plenty of scope for improving its performance in the short term. For the medium and longer terms, discussions have already taken place with a view to making the semantic models of SC/MGTP more concise and also more powerful. It may be that research along these lines will prove very rewarding. Professor McRobbie has already expressed interest in this research, and I am personally very attracted to the idea of pursuing research along these lines.

SIMPOS

Impressions

I am not a specialist in operating systems development, and so I offer these comments as a naive, casual user of SIMPOS. In particular, I don't know how easily some of my suggested improvements could be implemented, or at what cost to system performance, but I offer them here anyway.

I feel that SIMPOS offers a potent environment for developing KL/1 applications. I am particularly impressed by the support libraries of KL/1 graphics output routines that are available under SIMPOS. Although I haven't used these libraries myself, I saw their use at many FGCS demonstrations, and I felt that they were of high quality in terms of both performance and flexibility.

During my visit to ICOT I was surprised to learn that on a PSI-3, SIMPOS executes within a single X11 window, so that each SIMPOS window is maintained directly by SIMPOS software rather than by X11 software. This fact caused me some confusion, since I expected that SIMPOS applications would either write directly to the PSI screen (and hence be incompatible with X11), or else would interface with X11 on a window—for—window basis. I don't fully understand the reasons for the present design, but I imagine that one reason might have been the enormous amount of work needed to provide a full KL1/X11 interface on a function—for—function basis.

In any case, there seems to be some difficulty with the present approach. I noticed there was some bad timing for cursor motions and mouse clicks, and that SIMPOS would sometimes freeze when I typed ahead quickly during a window refresh. At such times, the only solution I could find was to reset SIMPOS completely. I feel that the most likely reason for this is just the enormous amount of interfacing required to translate a SIMPOS event into the final screen update.

I don't know whether making each SIMPOS window an X11 window would provide a solution to this problem, or whether such a change is even feasible. However, I believe that such a change would be useful for many other reasons, and I invite the SIMPOS designers to please consider it.

I noticed another difficulty that emerged as a result of my heavy use of virtual ttys (i.e, the Unix devices "/dev/ttyp0[0-9]"). Due to my need to access five or six different computers during my software development, I often used every virtual tty device for Internet connections and local windows. When I did this with SIMPOS running, the system would sometimes freeze, and would only respond to pairs of keystroke+mouse motion events. My solution in this case was to close one or more windows to free up some virtual ttys, and then the system immediately became fast again. I don't know whether the response problem came

from the fact that *all* the virtual ttys were used, or whether it was because *too many* virtual ttys were used. In the first case, increasing the number of virtual ttys may provide a simple solution; in the second case, the problem might be correctable through system fine-tuning.

During my visit to ICOT I had opportunity to read several volumes of SIMPOS documentation. Although some sections of the documentation are still in draft form, I would like to say that I found the English versions to be very clear and well written. The two manuals that I used most often were the PIMOS Programming and Operations Manuals (Version 6.0.), although these documents were supplemented with several paper preprints and technical reports. Using these manuals, I was quickly able to learn to use the SIMPOS file manipulation application, the compilation and debugging software and a variety of other useful SIMPOS tools. I am particularly grateful for my painless introduction to SIMPOS emacs — an editor that I have hidden from for many years! I found Pemacs very powerful and easy to use, and the documentation was of great help in learning to use the program.

My only regrets in this area are two: I was unable to obtain English documentation for KL1/SIMPOS system calls, which I would have liked to read out of interest and for future use, and from time to time I regretted the absence of an Index in the PIMOS manuals. However, I am very pleased to have learned emacs so painlessly; I hope to take a copy of the pemacs commands summary back to Australia with me, so that I do not forget!

KL/1

I cannot adequately express my delight with this programming language. To me it suggests many possibilities for new approaches in my own field, and I consider its development to be potentially as significant for knowledge processing as the development of Prolog was to logic programming one decade ago.

My main reason for making such a strong statement is very simple: there are many searching problems applications for which I believe that no effective depth-first strategy exists. Often in my own field, I have noticed that while the best search path may be *short*, the search-space is so *broad* that the short path is difficult to find using depth-first heuristics. This is because we can seldom know ahead of time which depth-first search path is best to begin with, and because there are many such paths. Later, while we are exploring a single search path, it is difficult to discover if we are making effective progress toward a solution because we have no way of gauging how difficult other search paths may be.

I feel that using an effective breadth-first language, we may find a solution to this problem. By developing many partial solutions in a breadth-first fashion, we may be able to develop comparison procedures to determine which paths are proceeding the best. If effective, such a comparison would permit us to concentrate our resources on a few of the best paths. Although we cannot effectively identify a good approach by considering just a single path, I believe that by considering *many* paths we may attain some insight into the solution process by using the context of *many* solution attempts to guide us.

I do not claim that KL/1 is perfect for this application, but I am very excited by the potential that it offers. It seems that many other researchers at ICOT feel the same way. Often, I have seen researchers at ICOT telling Dr. Chikayama what they think should or should not go into the next revision of KL/1, and I myself have already taken several opportunities to "bend his ear" with my own views. So, in my report, I would like to bend the ear again...

I think that KL/1 offers a very broad base for a variety of applications. At present it is a relatively pure language and the semantics are still very clean. Experience with other pure, clean languages such as LISP and Prolog shows that they rarely remain pure and clean once people begin to use them. But with KL/1 I feel that there is great potential to write a series of "front-end" syntactic translators that would provide many different language facilities for essentially the same semantic base. Using this scheme, it may be that KL/1 can remain pure for a little longer, while at the same time pleasing a diverse community of potential users...

Of the suggestions I outline below, several can be implemented as syntactic variations of current KL/1, and require no change to the compiler itself. Such alterations always maintain upwards compatibility with existing KL/1 programs. Others may require more extensive changes and hence need more detail than this document can offer.

Data Typing

I feel that as a software engineering language, KL/1 is somewhat too generous in its freedom of data access. For instance, although different data types exist, most type-checking is done at run-time, using special KL/1 predicates inserted into the guard. I feel that some type-checking would be more effectively performed at compile time, and so I would like to see more explicit type declarations in the KL/1 language. A sample syntax might look like this:

```
goal(A: atom, V: vector(Size), L: list, D) :— ...
```

where V and L have explicit types, while D can have any data type. For such an extension, no real change to the language semantics is necessary, and hence no change to the compiler is required. The language extension can be made entirely through a syntactic preprocessor. I feel that this syntax produces code that is freer from bugs, and easier to understand.

Aggregate Data Definitions

A possible extension on the extension above is to offer optional special types for vectors, with named elements, eg:

```
record(Myrecord, {Name: atom, Address: string, Family: List}).

identical(V1: Myrecord, V2: Myrecord, Value) :—
        V1.Name = V2.Name  ->   Value = true;
        otherwise;
        true               ->   Value = false.
```

Here, `record` is a special pre-defined goal. Since vectors are used as both homogeneous and heterogeneous data aggregates, we can use `record` to create heterogeneous data structures. Such code could translate syntactically to something like the following:

```
identical(V1, V2) :—
        V1 = {Name1, Address1, Family1},
        V2 = {Name2, Address2, Family2} |
        Name1 = Name2      ->   Value = true;
        otherwise;
        true               ->   Value = false.
```

I feel that such a simple change offers a very structured language with great potential for software engineering — and it uses no more preprocessing functionality than is already provided for in *oldnew* implicit arguments.

Optional Term Matching

Another observation I offer is that KL/1 data flow is always achieved by unification, which by definition is always bidirectional. However, very commonly the KL/1 programmer only requires one direction of data-flow, and so only term matching is needed, as in the following example:

```
List—Tail = Stream          % Break down a stream.
```

To me it seems worthwhile to use the knowledge that data-flow is intended in only one direction so that the compiler can generate code for *matching*, rather than full unification code. Such code is likely to be

more efficient in execution, but perhaps more importantly, I feel that the program will be easier to read, and is less likely to contain bugs arising from unintended unifications of aggregate data. I envisage two possible implementations of this extension:

1) All variables may be declared as read-only, write-only or read-write, for the scope of each goal. Thus, the question of unification or term matching can be decided by considering the variables in each unification expression. Eg:

    goal(<<V>>: Vector, >>L<<: List, D) :— ...

    where "<<X>>" is used to denote input (i.e. a read-only argument), while ">>X<<" is used to denote output (i.e. a write-only argument); absence of a data-flow quotation indicates both input and output (read/write), and this maintains upwards compatibility with current programs.

2) All unifications may be written as either one-directional (i.e. term-matches), or as bidirectional (full unification), so that the use of each variable depends on the set of statements in which it occurs. Eg:

    L <<= [V | Tail],

    Using this approach, each variable is read/write, but the data assignments themselves determine variable usage. Again, this extension is upwards compatible with existing KL/1 code.

Both of these suggested extensions require a revision to the compiler and to the language semantics, but I think that the benefits here are strong ones, and well worth the effort of revision. More efficient code translation opportunities are likely, but far more importantly, it is much easier to understand the intended sense of the code.

Test-and-Set Goal

With regard to parallel data access, it seems that the only possible data protection mechanism at present is by master/slave communication. I feel that a greater variety of options for parallel data access might be available through the introduction of an atomic *test-and-set* reduction. Such a system-defined reduction would take a variable argument, and try to unify it with some function of the current processor index and the calling goal's execution index. At the end of the unification attempt, the test-and-set goal would return the value of the variable, and the caller could check whether the value was its own value, or the value of another process. This mechanism would permit many sibling processes to access data safely without the need for requests to a master process.

Such a mechanism brings many new strategies to breadth-first searching, and I warmly encourage KL/1 development in this area.

KL/1 for Other Architectures

Finally, I earnestly suggest to the researchers at ICOT that a non—PIM KL/1 implementation is of the greatest importance to the international acceptance and usage of most of ICOT's free software. History shows us that such an implementation need not be efficient initially in order to gain acceptance. Early experiences with Prolog and Lisp, and more recently with powerful new meta languages such as SML, show that people are first drawn to such languages for their expressive power, rather than for their efficiency in execution. Commonly it is only later, as many researchers begin to accept the new language, that the language becomes efficient to execute as well as to write. By that stage, the popular adherence is already strong.

For many institutions, code development time is the most expensive resource of all. I have little doubt that a language which offers a radically new paradigm will find many adherents very quickly, provided that they can use the language on their own workstations rather than just reading about it in a scientific journal. Again, the question of high-quality documentation seems central to the use of new technology. Therefore, I warmly recommend that as a matter of priority, either ICOT produce a KL/1 implementation for a non PIM architecture and support it with thorough documentation, or else that ICOT publishes a set of KL/1 semantics broad enough to interest implementors from outside ICOT.

From the expertise that I have seen in my stay at ICOT, I have no doubt that if such an aim is attempted, it will swiftly meet with great success. I know too that I am not alone in offering ICOT the warmest of wishes for its future endeavours.

Conclusions

In the Introduction to this document I outlined the goals for my visit to ICOT, and now I would like to present a summary of my experiences here in the context of those goals:

- In my attempt to become better acquainted with MGTP and related software, I have been exposed to what I believe is some of the most promising research in automated reasoning today. I now feel that I have an understanding of the basic principles of MGTP software. I have studied it in operation and have been able to use it successfully in collaborative research. I know also that my knowledge of MGTP implementation is incomplete, and I welcome the opportunity to read any documentation that ICOT produces on the MGTP implementations.

- In my aim to gain expertise of the KL/1 programming language, I have discovered a language that I feel could become a basic paradigm for many areas of knowledge processing. I understand too though, that in such a new field, the opinions about what constitutes a good knowledge processing paradigm are many and diverse. I feel that in this field it is impossible to please everybody, and yet from the basic semantics of KL/1 I am encouraged to believe that there is a broad enough semantic base to allow many knowledge processing researchers to configure their own language extensions. It may be that in this difficult and often controversial field, such an approach is the best possible one.

- I am very pleased with my collaboration with Dr. Slaney and ICOT's Fifth Laboratory. Research is seldom conducted so quickly and decisively as it was conducted during my stay at ICOT. I attribute such rapid research progress to two main factors: the strong development environment which ICOT offered me, and the tireless work and aid of my Japanese colleagues — Dr. Hasegawa, Mr. Fujita and Mr. Koshimura are all deserving of praise. Of course, while semantically constrained MGTP is still being tested, no definitive statement can be made about the success of the research. However, whether such success comes now or later, this collaboration is still young, and I am sure that many more successes will also come if this collaboration is nurtured.

- In the production and presentation of demonstration software for FGCS-92, I feel that I am not qualified to judge the success of the attempt. Such a judgement is best made by those who *attended* the demonstration, not by those who *presented* it. However, I am personally satisfied with the audience responses I received during the demonstrations, and I am particularly satisfied at having helped produce two new research tools in the space of only one month. If my demonstration was successful, I believe that it stood in the good company of many other successful ICOT demonstrations. If it was not successful, then I feel that the quality of the collaboration alone has made my visit well worthwhile.

- In my visit to Fujitsu's laboratories I had the opportunity to witness research in a purely industrial environment, and to exchange ideas with researchers there. Our discussions often brought to my mind the obligation of researchers in knowledge processing technologies to recall the increasing demands of industry for useful and effective knowledge processing tools. It is clear to me that rapid progress in this difficult field can only be achieved through international focussing of our research goals, and through global communication of our insights and techniques.

I feel that the importance of such international and academic/industrial collaborations as the ANU/ICOT and ANU/Fujitsu collaborations cannot be overstated. Only through the exchange of ideas and opinions from both applied and theoretical spheres can we gain enough expertise to understand the depth of the questions at hand; only through frank comparison and consolidation of our many diverse approaches can we hope to achieve significant progress toward their solution.

It is clear to me that ICOT's policy of open collaboration can only be described as a great success, and I am in no doubt that I shall always remember my first visit to ICOT as one of the most formative influences of my professional life.

Curriculum Vita

Personal

Name	Mark Denis Grundy
Address	5/72 Canberra Ave, Griffith, ACT 2603, Australia
Phone	+61 6 249 0159
Home	+61 6 295 9943

Date of Birth	22/05/63
Marital Status	Married
Nationality	Australian
Occupation	Postdoctoral Research Fellow, Center for Information Science Research, ANU

Education and Awards

BSc. (Hons I)	awarded at University of Sydney, 1985.
Ph.D.	awarded at University of Sydney, 1992.
ARC Postdoctoral Research Fellowship	awarded at ANU, 1990.

Active Projects

The design and realisation of a language for programming cameo--based prover systems.

The incorporation of model--based reasoning into syntactic proof procedures.