Report to ICOT concerning my visit from
24 January 1983 to 10 February 1983

J.A.Robinson   Syracuse University, New York

U.S.A.

1.

In October 1982 I accepted the invitation of Kazuhiro Fuchi, Director of the ICOT Research Center, to spend three weeks visiting ICOT for the purpose of exchange of ideas and study. My visit began on Monday, January 24, 1983 and will end on Thursday, February 10, 1983. This report is written on Tuesday, February 8, 1983.

2.

From my point of view, the visit has been very valuable. I have had the opportunity to discuss many research topics with ICOT researchers, and in the course of these discussions I have come to know many of them quite well. Without exception I have been most impressed by the caliber of the ICOT researchers and their leaders - they are extremely intelligent, well-informed, hard-working individuals whose dedication to the ideas and goals of the ICOT project is of the highest order. By comparison with that of the best research organizations in the West, the ICOT spirit is more unified, more sharply focussed, more urgent, more serious. I am comparing ICOT here with, for example, not only the private industrial research organizations such as IBM's Yorktown Heights and San Jose Laboratories, General Electric's Schenectady Research Center, Texas Instruments' Research Center in Dallas, and the Data General Research Center in Chapel Hill, North Carolina, but also with government research organizations such as the Argonne National Laboratory, the U.S. Naval Research Laboratory, and the U.S. Air Force Rome Air Development Center. I have visited and/or worked at all of these institutions over the past twenty years, and have also worked at several research centers in Europe, notably the University of Edinburgh and the Imperial College, London, in the U.K., and the UPMAIL Laboratory in Uppsala, Sweden. In my judgment, the research atmosphere at ICOT is superior to that of any of these places.

3.

On addition to an extended interaction with ICOT itself, I was very pleased to be able to visit several important other organizations. These were : Kyoto University, on Monday, 24 January, 1983 : Tokyo University, on Tuesday, 25 January, 1983 : the Electrotechnical Laboratory, on Monday, 7 February, 1983 : and the Electrical Communication Laboratofy of NTT, on 8 February, 1983. It was also a pleasure to meet Professor Takayasu Ito of Tohoku University when he visited ICOT on 2 February 1983: we had met previously many years ago in Edinburgh and were able to renew our mutual interests. His present work to develop an in-house L.S.I. fabrication facility is very interesting. The impressions I formed of the research being conducted at these several centers are very favorable, and I shall be occupied for many weeks in reading and digesting the large number of articles and reports I collected from over 20 research projects covering the fields of natural language translation, speech understanding, program verification, inductive definition of data structures, parallel inference machines, pattern recognition, superspeed computer design, program transformation, knowledge based game playing, LISP machine design, advanced PROLOG implementation, and new LISP-like programming languages.

4.

I was surprised to discover a very sharp division between research projects in the universities, supported by the Ministry of Education, and government research projects supported by the Ministry of International Trade and Industry. It is not clear to this Western observer what purpose is served by such a rigid compartmentalization of the government's research effort - it is as though the two Ministries were separate countries instead of two departments of the governmental apparatus of one country! Might it not be a fruitful practice to have coordinated research efforts between university and government groups? This is certainly the practice in the U.S.A., the U.K., Sweden, France and Italy (to mention only the countries I have had experience of).

5.

The formal exchange of ideas at ICOT took two forms: a series of "mini- workshops" on 26, 27 and 28 January, 1983 in which ICOT researchers explained their work in several areas, and a series of lectures given by me on various topics, which I cover in more detail below. The mini-workshops were thorough, detailed accounts of work in progress and ideas for future work. It was during these presentations that I came to realise the depth and strength of ICOT's research staff. The content of the presentations will take me quite some time to analyze properly after my return to Syracuse, but I can say immediately that the general view they gave me was entirely in accord with the picture I had formed over the past year and a half, by careful study of the Fifth Genaration Report and its accompanying position papers. It was good to see the actual work progressing well along the path laid down in the overall plan.

6.

I also had numerous informal exchanges with individual researchers, notably with Kuniaki Mukai, whose beautiful unification algorithm I studied hard with a view to helping simplify (if possible) its intricate proof of termination, and to developing (if possible) a suitable analysis of its computational complexity. I hope to continue studying this work after I return to Syracuse: and Mukai and I plan to stay in touch about it. I was also delighted to find that our Syracuse LOGLISP System was up and running at ICOT on the DEC-2060. I was helped frequently and generously by Yasukawa-San of the Third Research Laboratory, whose expert knowledge of the DEC-20 operating system compensated for my own DEC-10-based strangeness. I hope that ICOT researchers will experiment with the different kind of logic programming environment provided by LOGLISP. I also had good discussion with Rikio Onai about reduction machines for logic programming.

7.

I gave four private ICOT lectures, on 25, 27 February 1983 and 3,4 February 1983. I will gave a public ICOT lecture on 10 February 1983, and another lecture at ECL this afternoon (8 February 1983). Two of my private ICOT lectures concerned the ideas which I and my student Kevin Greene have been thinking about concerning parallelism in logic programming. The transparencies from these lectures are attached as Appendix A to this report. The general idea is to exploit both OR-parallelism and AND-parallelism, but to confine the latter entirely to the internal archtechture and functioning of the several parallel unifiers in the system. Each parallel unifier computes the environment:

$$\mathcal{E}' = (\text{unify } \alpha\ \beta\ \mathcal{E})$$

which extends a given environment ε in the most general possible manner so as to make the expressions α and β become identical. In general α and β can be lists

$$\alpha = (\alpha_1\ \alpha_2\ \ldots\ \alpha_n)$$
$$\beta = (\beta_1\ \beta_2\ \ldots\ \beta_m)$$

of expressions, and n cam be quite large. The parallel unifier regards such lists in the LISP manner as pairs

$$\alpha = (\alpha_1\ .\ (\alpha_2\ \ldots\ \alpha_n))$$
$$\beta = (\beta_1\ .\ (\beta_2\ \ldots\ \beta_n))$$

and upon encounterring them, creates two new processes,

(unify $\alpha_1\ \beta_1$)
(unify $(\alpha_2 \ldots \alpha_n)(\beta_2 \ldots \beta_m)$)

to be pursued in (AND-) parallel, and so on. The environment ε' is built up by a process which is in concurrent communication with all of the unifying subprocesses. The details are given on the transparencies. Such a parallel unifier is activated (in OR-parallel) by each inference step by which one goal-statement:

$$(\gamma, \varepsilon)$$

(γ a list of goals: $\gamma = (\gamma_1\ \ldots\ \gamma_k)$, and an environment of bindings) is succeeded by another, (γ', ε'). In ordinary (serial, mono-processing) logic programming, (γ', ε') comes from (γ, ε) by a single-LUSH resolution step involving some clause

$$\alpha_i <- \beta_i$$

from the knowledge base

$$\alpha_1 <- \beta_1$$
$$\vdots$$
$$\alpha_r <- \beta_r$$

Thus, if

$$(\gamma, \varepsilon) = ((\gamma_1\ \ldots\ \gamma_k), \varepsilon)$$

the "single-LUSH" successors of (γ, ε) have the form:

$$(\beta_i * (\gamma_2\ \ldots\ \gamma_k).(\text{unify}\ \alpha_i\ \gamma_1\ \varepsilon)),$$

using a single clause $\alpha_i <- \beta_i$; while the "multi-LUSH" successors of (γ, ε) have the form:

$$(\beta_{i_1} * \ldots * \beta_{i_k},\ (\text{unify}\ (\alpha_{i_1}\ \ldots\ \alpha_{i_k})(\gamma_1 \ldots \gamma_k)\ \varepsilon))$$

for some k clauses

$$\alpha_{i_1} <- \beta_{i_1}\ .\ \ldots\ .\ \alpha_{i_k} <- \beta_{i_k}\ .$$

It is clear that a mult-LUSH deduction tree consists of fewer, but "larger" nodes than a single-LUSH deduction tree. All multi-LUSH successors of a given goal-statement (γ, ε) can be obtained simultaneously in OR-parallel. Each one involves the activation of an independent unifier which operates with internal AND-parallelism. The "combinatorial explosion" involved in detecting all k-tuples $(i_1, \ldots, i_k)$ of indices for which (unify $(\alpha_{i_1} \ldots \alpha_{i_k})(\gamma_1 \ldots \gamma_k)$ ε) is not "impossible" is not as bad as it may seem at first glance. If we define

$$I_j = \{p;\ (\text{unify } \alpha_p\ \gamma_j\ \varepsilon) = \text{"impossible"}\}$$

then it suffices to consider only those $(i_1, \ldots, i_k)$ satisfying:

$$(i_1, \ldots, i_k) \text{ in } \quad I_1 \times I_2 \times \cdots \times I_k = I$$

The I_j can be computed in OR-parallel, given (γ, ε), before attacking the full computation of the set:

$$\begin{aligned}\{(\gamma', \varepsilon');\ & (i_1, \ldots, i_k) \in I \text{ and } \gamma' = \beta i_1 * \ldots * \beta i_k \\ & \text{and } \varepsilon' = (\text{unify } (\alpha_{i_1} \ldots \alpha_{i_k})(\gamma_1 \ldots \gamma_k)\ \varepsilon) \\ & \text{and } \varepsilon' \neq \text{impossible"}\}\end{aligned}$$

of all multi-LUSH successors (γ', ε') of (γ, ε).

8.

A further ICOT lecture I gave concerned the "sequent" treatment of resolution in Chapter 6 of my book Logic: Form and Function". As I try to explain in that book, the sequent concept (due originally to Gerhard Gentzen) permits a far better development of the resolution concept than the usual approach.

9.

I also gave an introductory lecture to the LOGLISP system.

10.

A short version of the parallelism lecture is to be given to ECL during my visit there on 8 February 1983.

11.

Finally, my public lecture on 10 February 1983 is entitled "Logic Programming, past, present and future". Copies of the transparencies for this lecture are attached to this report as Appendix B. The lecture covers the history of the various ideas involved in logic programming, starting with the invention of the predicate calculus by Gottlob Frege in 1879. It concludes with some attempts to glimpse the future, two or three decades from now, which the logic programming idea will help to bring about.

12.

In concluding this report I would like to express my thanks to Director Fuchi for inviting me to make this visit, and to him and all other ICOT members for the warmth of their hospitality, the openness of their friendship, and the stimulus of their intellectual example.

JOHN ALAN ROBINSON

Curriculum Vitae as of 1 January 1983

PRESENT POSITION

Distinguished University Professor of
Logic and Computer Science

Logic Programming Research Group
School of Computer and Information Science
Syracuse University
Syracuse, New York 13210

Telephone (315)-423-3159

PERSONAL DATA

Born 9 March 1930, Halifax, England.
British citizen.
U.S. resident since 1952.

EDUCATION

Rishworth School, Halifax, England, 1941 - 1948.

Cambridge University, Cambridge, England, 1949 - 1952. B.A. (with Honours) in Classics, 1952.

Cambridge University, England. M.A., 1957.

University of Oregon, 1952 - 1953. M.A. (with Honors) in Philosophy, 1953.

Princeton University, 1953 - 1956. M.A. in Philosophy, 1955. Ph.D. in Philosophy, 1956. Dissertation: Causality, Probability and Testimony. Thesis advisor: Hilary Putnam.

EXPERIENCE

Operations Research Engineer, E.I. du Pont de Nemours and Co., Wilmington, Delaware. 1956 - 1960.

Postdoctoral Research Fellow, University of Pittsburgh, 1960-61.

JOHN ALAN ROBINSON

Research Associate, Applied Mathematics Division, Argonne National Laboratory. Summers of 1961 to 1964.

Consultant, Applied Mathematics Division, Argonne National Laboratory, 1961 - 1969.

Member, Computation Group, Stanford Linear Acceleration Center. Summers of 1965 and 1966.

Lecturer, University of Michigan Engineering Summer Conference. 1964 and 1966.

Member, Faculty of Rice University, 1961 - 1967, in Philosophy and Computer Science. Full Professor 1964 - 1967.

Member, Faculty of Syracuse University, 1967 - present. Distinguished University Professor of Logic and Computer Science, 1967 - present.

Guggenheim Foundation Fellow, 1967 - 1968.

Honorary Research Fellow, University of Edinburgh, 1967 - 1968.

Visiting Research Fellow, University of Edinburgh, 1968 - present.

Consultant, Stanford Linear Accelerator Center, 1966-1970.

Consultant, Young and Rubicam, 1969.

Consultant, General Electric Company, 1970.

Consultant, I.B.M. Research Center, Yorktown Heights, 1977.

Consultant, I.B.M. Systems Research Institute, New York, 1981, 1982.

Co-Principal Investigator, Rice University Computer Project, U.S. Atomic Energy Commission Contract AT- (40-1)-2572, 1966-1967.

Principal Investigator, National Science Foundation Grant GP 2466, for research on theorem proving by computer. 1964 - 1966.

Principal Investigator, Advanced Research Projects Agency Contract DAHC04-72-C-0003, for research on computational logic, 1972 - 1974 .

JOHN ALAN ROBINSON

Principal Investigator, National Science Foundation Grant MCS77-20780, for research on logical computation, 1978 -198(

Principal Investigator, under RADC Contract F30602-77-C-023! of LOGLISP development project, 1977 - 1980.

Principal Investigator, GREATER LOGLISP development project, RADC Contract F30602-81-C-0024, 1981 - 1983.

External Examiner, University of Edinburgh, 1967, 1969, 1970, 1972, 1973.

External Examiner, University of London, 1980.

Trustee, The A.M.Turing Trust, Edinburgh, Scotland. 1969 - present.

Scientific Advisor, Swedish Board for Technical Development 1982 - .

RESEARCH INTERESTS

Logic Programming: theory and applications
Mechanical theorem proving
Mathematical logic and non-numerical computation
Abstract programming theory
Artificial intelligence
Foundations of mathematics
Philosophy of mathematics

LISTED IN

WHO'S WHO IN AMERICA
WHO'S WHO IN THE WORLD
AMERICAN MEN OF SCIENCE

CHRONOLOGICAL LIST OF PUBLICATIONS

GAMMA I, a general theorem-proving program for the IBM 704. Argonne National Laboratory Report 6447, 1961.

Hume's two definitions of cause. Philosophical-Quarterly 1962 (pp. 162-171).

above article, reprinted in Hume, a Selection of Critical

TOWARDS A HIGH PERFORMANCE PROLOG PROCESSOR

David Warren, February 1983

ABSTRACT

In this note we consider how one might design a high performance Prolog processor, by exploiting low-level parallelism in Prolog execution. We focus chiefly on potential parallelism in the implementation of unification.

N.B. The ideas expressed here are very tentative, and this account of them is very sketchy.

INTRODUCTION

Many tasks for which Prolog seems highly suited are not really practical on current machines because they can be implemented more efficiently in lower-level languages; examples include text editors, document formatters, and key parts of operating systems. This limitation of Prolog would completely disappear, however, if one logical inference (i.e. resolution) took the same order of time as a conventional machine instruction. Is it possible to design special Prolog hardware that can perform resolutions as fast as a conventional processor executes instructions, and that has a cost comparable with a conventional processor? To put it more concretely, can we build a one megalips (i.e. one million logical inferences per second) Prolog machine for the same kind of cost as a one mips VAX, for example? This note presents some initial ideas towards this goal.

We wish to exploit certain low-level parallelism in Prolog, by analogy with the way a conventional ALU exploits parallelism in arithmetic operations. By "low-level" parallelism, we mean parallelism that is invisible to the Prolog programmer. Of course logic programs seem to offer great potential for large-scale parallelism, but this kind of parallelism cannot be exploited without the programmer being concerned. It will require a fundamentally new approach to logic programming. Prolog's control mechanisms (including cut) and use of side effects will have to be replaced by something new. Since exploiting large-scale parallelism requires major advances in both programming methodology and computer architectures, it is clearly a subject for longer term research. What we propose here is less radical, and hopefully can be realised sooner.

Our goal is to take a conventional Prolog system on a von Neumann machine, and speed up its innermost mechanisms. Briefly, this means replacing the central processor, but leaving the organisation of main memory essentially unchanged.

If we look at a typical resolution, such as the 'concatenate' loop, we see that it requires of the order of 100 basic operations. On the DEC-10, for instance, the 'concatenate' loop compiles into 50 machine instructions (see Appendix II), and runs at around 30,000 lips on a KL-10 processor. For the Psi machine being designed at ICOT, the performance is expected to be similar, in the region of