A visit to ICOT

Maurice Nivat

Professor at the University of Paris 7
Corresponding member of the French Academy of Sciences
November 25th – December 15th 1988

1. Background

I have had many opportunities to visit Japan and to get acquainted with the Fifth Generation Computer Project and the work which is achieved in ICOT. I also visited a number of Japanese Universities including Tokyo, Kyoto, Tohoku, Osaka, Hiroshima and Kyushu and a number of public and private research laboratories including ETL, NTT, Mitsubishi, Fujitsu and NEC.

Two France-Japan seminars were arranged jointly by ICOT and INRIA and, in both cases, I was partly responsible for the program with Professor Laurent Kott, the director of IRSISA which is a branch of INRIA in Rennes and we discussed the program with Dr. Kurozumi, Dr. Furukawa. I remember very well the friendly discussion I had with Dr. Furukawa, in Pisa, when attending a TAPSOFT conference: we drafted the program of the second seminar which was held in Cannes in November 1987 and gave rise to a proceedings volume which has been very recently published by Elsevier Science Publishers in Amsterdam.

I remember also very friendly discussions I had, both in Tokyo and in Paris, with the late Professor Tohru Moto-oka at an early stage of the development of the Fifth Generation Computer Project. We were already discussing the Parallel Inference Machine issue and the real gain in efficiency that one could expect from the use of several locally connected processors for logic programming. He was a great physicist, I am, both by training and by taste, were a mathematician, and of course we could only disagree, but in the most friendly and fruitful way, each one hearing carefully to the other's argument and, at times, changing its mind when some new idea was presented by his colleague.

There remains and deep contacts with quite a number of top Japanese scientists and the experience I have been fortunate enough to get of the "Japanese-way" of doing research (a way which can be as cheerful as it is efficient) have had a strong influence on me and I have been fighting, in France, to have no build "un project à la japonaise"(Japanese-like project) with up to now no real success. Who knows about the future?

2. The present Visit

A word-to-mouth invitation from Dr. Fuchi and Dr. Furukawa given to me in Cannes in November 1987 was made formal in April 1988 by a letter of Dr. Fuchi. I accepted gratefully this invitation to spend two weeks in ICOT including the week of FGCS'88. To tell the exact truth I managed to visit the Institute of System Science of the National University of Singapore on my way to Japan (I left France on November 13th and stayed in Singapore until November 25th) and also to visit India on my way back (on December 15th I leave Japan to Dehli). The conference FGCS'88 was quite different from the previous one in 1984 which I also attended. The difference can be stated in one sentence.
"*A lot of work has been accomplished, in the world, and more specially in ICOT, and the problems to be solved appear with a much greater clarity.*"
The introducing talk of Dr. Fuchi on Monday, November 28th, was really enlightening. We know now where are the "stumbling blocks" and ICOT seems perfectly prepared to enter the "final stage" which should result in actual prototypes, meeting some of the requirement of the original project.

A large number of talks, presented both by Japanese researchers and foreign researchers were dedicated to the extremely precise problems of parallelizing logic programs. In an other room one could see demos of the "multi Psi" machine. This was an extremely tense and vivid exchange. Ueda's GHC system of "guarded Horn clauses" is an extremely interesting attempt to solve the above-mentioned problem. (One should refer to the report Gerard Huet, from INRIA, wrote after a stay in ICOT in April this year), the ANDORRA PROLOG system of D. Warren et alii is also extremely interesting as well as proposals made by the group of ECRC (in München, FRG) and the group of "Logical programming with constraints" (Lassez, IBM Yorktown Heights, Jaffar, Colmerauer in Marseilles).

Most of my time was spent listening to the talks related to this problem, which is the problem I am mostly interested in. I have also listened to several talks on "knowledge base system" or (the two expressions are synonymous for me) "deductive data bases" and talks on "natural language understanding" and "natural language processing" (including the talks on "situation semantics")

Afterwards, starting December 6th, there was a closed symposium which I did not attend faithfully. The closing round table was rather disappointing.

On Wednesday, December 7th, I gave a 1h 1/2 talk on "Fairness of Finite Transition Systems", with the impression that people are less interested by the automata-theoretic approach to the semantics of distributed systems in ICOT than in many European laboratories (in UK, FRG, Italy, The Netherlands and France). Maybe I would have liked to give a series of lectures rather than just one. Though apparently simple this automata theoretic approach which was initiated by Robin Milner when designing CCS is still full of surprise and tricky problems: despite the fact that all the existing systems to verify properties of concurrent, parallel and distributed

system there are still many non answered questions especially as concerns the notion of fairness (several definitions have been proposed) and the equivalence of non deterministic finite automata (extensively studied in Edinburgh, Pisa, INRIA). I had on this question a very interesting discussion with Robin Milner himself who attended FGCS'88).

On Thursday December 8th I was invited to an excellent lunch by Dr Hiroshige, the executive director of ICOT, the lunch was also attended by Dr. Fuchi, Dr. Furukawa and Dr. Iwata. We discussed several issues in parallel computing and also the change in European Research on Computer Technology which is due to the extension and the success of the ESPRIT program of the European communities.

An extremely cheerful welcome party in the evening of the same day gave me a new proof of the deep sense of hospitality of ICOT researchers towards their foreign guests and also of the excellent links which exists between ICOT researchers regardless of rank position and age. Last, but not least, I have been able to discover the incredible beauties of the Mt. Fuji -Hakone National Park during two sunny days with a very brisk and pure late Autumn atmosphere.

3. Some remarks of a theoretical computer scientist

I must say that I am a doubtfull fellow and recall that I am mathematically minded and trained.

It is obvious from all the talks on the subject which were presented at FGCS'88 by ICOT researchers, other Japanese researchers and foreign researchers that we badly lack a methodology of parallel programming. Sequential programming has made huge progresses in the past 15 years. There exist excellent, fast and cheap programming environments. This is due to the skill of a huge number of system designers, to the tremendous increase in the computation power and speed and in the memory capacity, but this is also due to a fundamental basic knowledge of what is a sequential program, how it can be specified by formal methods which allow also to define implementations and verify that they are at least piecewise correct, how one can define the environment in which a program component can be reused. In the same vein a lot of work has been achieved as program transformation, partial evaluation and measurement of performances. This has been achieved for the simplest while (do-loops) programs (with appropriate semantics such as Hore's logic) for recursive programs (and fixpoint semantics), for LISP and computations as formal objects, for logic programming including PROLOG and alternatives to PROLOG and in an even more general framework for general systems of rewrite rules and algebraic data types (I was a bit surprised that this last topic was running in FGCS'88). For the special purpose of knowledge and data bases, similar work is being achieved for computation in relational algebras and, in an interesting way, for the computation of recursively defined relations over a data base, providing a good theoretical basis for the handling of "deductive data bases" (which are, for me, the prototype of knowledge base in which one should be able to use not only the basic knowledge which has been stated but all the derived knowledge which is implied by this basic knowledge and simple inference rules). I repeat it is obvious from all the presentations that we

do not have the same knowledge about parallel or concurrent or distributed programs but in one case which is the SIMD architecture and "vector machines" such as the CRAY or the NEC supercomputers, mainly intended for fast numerical computations.

For computers composed of a number of loosely coupled processors (be they 6, 12, 24, 128 or 32,000) exchanging messages or sharing a memory we are very for to being able of programming them in the same way. The parallelization of a given algorithm is a problem in itself, usually quite difficult, and if we slightly change either the algorithm or the architecture of the machine we wish to adapt the algorithm to we are facing a new problem and little knowledge from a previous work can be used. Just now it seems unlikely that ordinary users can be left with the problem of assigning tasks to a number of loosely coupled processors, of defining the messages they should exchange (or the values in some shared memory architecture) and the synchronization mechanisms so that the whole process will be correct and efficient. (What I am writing here is very close to the introduction of the paper by D.Warren et al).

A natural conclusion is that parallelism should be used at implementation level, the user ignoring completely how it works and writing plain programs which could be as well run on a sequential machine. The implementors' job is to use the parallel facilities of the machine in such as clever way that the end users can continue ignoring these facilities and only sees that the speed in higher and the cost lower when such a parallel mechanism is used. This means that, restricting ourselves to PROLOG programs, all the analysis of the structure of a program and the definition of the task, which may be carried in parallel ("and" or "or") has to be ensured by the system: may I say that, from all my knowledge, we are extremely far from being able to achieve that?

If one wishes to have several processes cooperate to explore the proof free corresponding to the resolution of a PROLOG program then these processes should all know what has been done by itself and the other processes. And this requirement leads to a shared memory link between all the processes in several papers presented at FGCS'88. But none of these papers given an evaluation of the speed which will be obtained using such a mechanism. It is not even proved in any way that several processes sharing a memory and cooperating to run a PROLOG program will do the job faster than a single one, or with a gain in speed which does not justify the use of several processes.

My attitude towards this precise problem would be to try to understand better what is really happening in the resolution of a PROLOG program (be it a pure PROLOG program or a program with "cuts" or a program with "constraints"). For example, the so-called "lazy evaluation" mechanism for ordinary sequential recursive programs has been studied extensively and proved to be the optimal way of computing the result of such a program. Lazy evaluation has also been studied for lambda-calculus and other "functional" programming language. And it has been successfully implemented in some interpreters of LISP-like languages. I cannot see, just now, a clear definition of "parallel lazy evaluation" which implies that the interpreter knows

at each instant of the computation which subexpression should be evaluated next (or in logic programming which subgoal has to be tried next).

Then it is a real puzzlement to me to listen to ICOT directors say that we are at the end of the "intermediate stage" (which I understand as an experimental phase of the project in which several possible machines and system architectures were experimented) and that a 3 years period of "final" stage can be launched aiming at a substantial progress in "industrial" computing using parallel multi-processors systems. My own feeling is that it will take many more years before we understand well enough the phenomena linked with parallel computation and design a general purpose efficient and easy to use parallel machine and its built-in software.

As a matter of fact, one can also remark that the time lag between the birth of an idea and its use in some widely spread industrial product is much longer than what several computer manufacturers say: over ten years where needed to define completely high level imperative languages and learn how to compile them (roughly the period 1958-1968), ten years also elapsed between the first ideas which lead to the UNIX system and its being really used on an industrial scale (roughly 1972 - 1982).

We can guess that the rather theoretical ideas about parallel programming in a PROLOG like language will take the same amount of time before they are really implemented. And, in this precise area, the situation is worse than in others. Since PROLOG like languages are in current use in only a small part of the community of uses of computers (2 %?) The efficiency issue which was raised above and is a difficult one may be a less important issue than the training of a substantial proportion of users to this type of programming. In fact the two issues of efficiency and effective use are completely linked: one has to show that logic programming will bring an improvement to solve the problems for which most people use computers. These problems are, as everybody knows, problems of large data manipulations, i.e. problems in collecting, storing, sorting, searching, restructuring large amounts of data. Up to know logic programming failed to demonstrate that it is a more efficient way to perform all these actions than a classical programming language like FORTRAN or PASCAL. The 8-queen problem and more generally the n-queen problem, as challenging as they may be far a theoretically-minded researchers, do not reflect the difficulties of real data handling: PROLOG was invented and proposed as a good way of describing and implementing "parsing algorithms". It is still the best way to describe and efficiently implement such algorithms in which the depth of recursion nesting is low, and the amount of data which is used at each step of the computation is also strictly bounded.

I am completely conscious that I am questioning 2 basic options of the research which is achieved in ICOT.

- why parallel machines rather than sequential ones?
- why logic programming and PROLOG rather than plain old sequential imperative programming languages.?

Since I dedicated a large amount of my time, as a researcher and as a professor, during the past 10 years, to work on parallel processing I hope ICOT researchers and the directors of this

Institute will understand what I mean: there is no need that an appealing idea prove eventually to be a good one.

This leads me to the last remark, I wish to make, and these remarks concern knowledge bases systems. My remark will concern only the "knowledge-bases" intended to help either a "computer-aided transformation system" or an "automatic natural language understanding system". Clearly all the rules of the form "in context γ the word Y can be either translated in Z or understood as W" can be written in Situation Semantics. In fact, Barwise's Situation Semantics was designed precisely to express rules of this kind. The problem is then, not to find an expressive-enough system, but to write down all the rules which are needed. Let me call this set of rules a "dictionary" which will incorporable linguistic informations as the following
Je suis tombé sur l'herbe is translated in English by "I fell on grass" (you need know that in French you say "*je suis tombé*" and not "*j'ai tombé*" the auxialiary verb "avoir" which is used with most verbs to build the past tense being replaced for the verb "tomber" by the auxiliary verb "être". One should note also that the substantive "grass" which translates the French "herbe" is preceded by no article). "*Je suis tombé sur un vieil ami*" is translated in English by "I came across on old friend" (the meaning of the same verb "tomber" with the same preposition "sur" is entirely different).
"*Je suis tombé sur la tête*" can be, roughly, translated as "I become mad": it is what my friend and colleague M. Gross calls a "fixed-expression", i.e., an expression whose meaning cannot be determined from the meaning of the components.

We have no precise idea of how many rules of that kind are needed either to handle a language in other one or to understand the meaning of a sentence. Likely the number of such rules (in every natural language) is 5 or 6 or 7 hundred thousands. Immediately an efficiency problem is raised: how shall we handle such a huge number of rules? But there is an other problem which is even worse: who will write down such a set of rules? Computer scientists will not write these rules for the excellent reason that they do not know enough about linguistics. And linguistists are unlikely to do it for they do not know enough about computers and also because it would be a tremendous and not very rewarding job. Maybe also a hopeless job. Consider the French sentence
"*Je suis tombé sur Pierre*"
It can be translated into English either as "I fell on Peter" or as "I came across Peter" and it is perfectly, semantically ambiguous (no semantics, situationists or not, can distinguish these two possible meanings of this extremely simple French sentence).
Hence I can see a lack of reflexion, not only in the Japanese project, but indeed in all the projects which tend, all over the world , to either translate one natural language into another or to understand some natural language. Fortunately, the problems of translation and natural language understanding seem too complicated to be solved by a computer. I would hate a computer understanding me!

4. Conclusion

ICOT is an excellent laboratory and I am really surprised that it could, in such a short period of time, train so many very high quality young researchers. Now, ICOT is a place where one can find people to talk, about, more or less, every real issue in computer science. All these people have also brought to light many ideas which seem to me very important and promising for the future of computer science, and the future of computer Industry.

But ICOT, as all other laboratories in the world working on similar subjects is facing very hard problems: some will be solved, maybe, in the next 10 or 20 years, and some may not be solvable in any predictable future. Trying to go faster may be dangerous for the future of ICOT and for the future of Computer science, as a whole.

Maurice Nivat

CURRICULUM VITAE OF

Maurice Nivat
born in Clermont-Ferrand (France) - Dec. 21st 1937
Citizen of France, married 3 children.

Education

Ecole Normale Supérieure de la rue d'Ulm (entrance in 1956)
Master degree and Agregation in Mathematics (1961)
Doctor degree in Computer Science (1967)

Occupation

Professor since 1969 at the University Paris 7
First director (1975-1985) of the LITP (Laboratory for Theoretical Computer Science and Programming) associated with CNRS (National Center for Scientific Research)
Still head of the research team "Languages, concurrency and parallelism" and head of the scientific council of this laboratory.
Advisor to the President of INRIA (National Institute for Computer Science and Automation)
Editor in chief of Theoretical Computer Science (North Holland Publishing Cy, Amsterdam, the Netherlands)

Honors

Corresponding member of the French Academy of Sciences
Knight of Légion d'Honneur and Ordre National du Mérite.

Research activity

Over 100 papers on the following subjects

- formal language theory, especially context-free languages, rational transductions, congruences on words

- algebraic semantics of programming languages, free algebras, tree grammars, computational rules

- infinite and biinfinite words, infinite trees and infinite computations

- finite automata models of distributed systems

- scattered combinatorial and algorithm problems : shuffle, tree codes, tilings of the plane.

Several reports to the French Government on Training and Research in Computer Science.