FGCS Assessment

Gilles Kahn
INRIA Sophia Antipolis, FRANCE

Introduction

Assessing the FGCS project is a hard task, not only because there is little time and space to do so, but also because the scope of this project is very wide. I doubt if many people can read equally competently the work that has been performed in so many diverse areas: computer architecture, programming language design, database design, natural language analysis and generation, genome research. I for one have very little competence on natural language work, although I hear from a variety of sources that this may indeed be one of the strongest points of your work in the last phase of the project.

The FGCS project is very broad, but everyone can see its unity of purpose. When listening to all the presentations and reading some papers, I wonder whether there has been enough time for true integration of the many components that have been developed in the seven laboratories. I suggest the following problem, to see what I have in mind. You have developed a theorem prover, MGTP. On the other hand, you have sophisticated tools to generate sentences and paragraphs in Japanese. Assume that you would like to connect these two components, so that the theorem prover produces proofs in natural language, that a Japanese mathematician would like. In particular, these proofs should not be too verbose, concentrate on the real difficulties and be socially acceptable. Is this a problem that can be solved in a matter of weeks or months with the software that you have developed, or do we need to start a new project?

I have listened carefully to all the talks that reported the work of FGCS project, and I must say that they were all very high quality presentations. The laboratory chiefs show considerable experience, maturity in their scientific fields. Answers to questions are very direct and frank and do not try to cover up difficulties when there are some. As well, the demonstrations have shown strengths and weaknesses of the software. I appreciate the considerable amount of work needed in preparing such a thorough presentation of the FGCS achievements. This has confirmed an attitude that I have witnessed in the wonderful INRIA–ICOT meetings that I have attended: a completely frank exchange of views with scientists of high caliber, who are concentrating on basic research, and building software prototypes to demonstrate the validity of their fundamental ideas.

The Central Issue

As I see it now, the FGCS project has been attacking a fundamental problem in Computing, namely concurrency; and it has chosen a line of attack, Logic Programming. This is a priori how good projects get started: with a difficult and deep problem on the one hand, with an original idea on how to solve it on the other hand. Indeed, the problem of concurrency in computing, which has been with us practically since the invention of computers, has become ever more essential in the past 10 years. Let us review for a moment the positive and negative aspects of your approach.

On the plus side, we can see many advantages: first the attack is extremely original, almost far fetched for some. The Japanese effort appears immediately as a leader, because noone else is betting on this direction of work on the same scale, even though a number of very bright individual researchers around the world have had successes. Second, your approach is a software approach, i.e. you are concerned a priori with the intellectual control of parallel hardware, with putting into hardware mechanisms that will make it easier to program. Let us go more quickly through the other

advantages: the problem seems tractable, opens many questions, leads to basic research; it focusses on fine grain parallelism, which is a priori harder, on irregular problems that occur in symbolic computing –of course–, but even in numerical computing (finite elements), in geometry or in discrete events simulation.

On the negative side, the major problem is that the approach is

very difficult:

how can one have an efficient computing system that combines parallelism with non-determinism, which is implied by a declarative approach? Parallelism means that you parcel out work to remote computing units, non-determinism means that you may discover at any time that this work is useless and should be canceled immediately. In fact, the problem is so difficult that one of these two aspects runs the risk of being short-changed. Clearly, GHC and KL1 have shown their bias in favor of parallelism. The second problem with your approach is that people have misunderstood it: they considered that you worked on Logic Programming, using parallelism to compensate for its intrinsic inefficiencies. Another difficulty, linked to the originality of your approach, is that when you started, there were very few applications based on the logic programming paradigms, so that you ended up having to mount your own effort to build applications.

Let me add two remarks regarding the difficulty of your project. First I have stated in my talk that the Logic Programming community, as a scientific group, was "weird". As there were many proeminent members of that community in the audience, I received a large number of inquiries about what that statement really meant. First, Logic Programming, in 1981, was virtually unknown in the United States. The group of scientists who had been fighting for it was necessarily a bit paranoid about that fact. Second, there was, and unfortunately there still is, in this community –usually not among the top

leaders– a slightly sectarian attitude: they have seen the Truth revealed, nothing else deserves paying attention to.

The next remark has to do with something that has unfolded during

the project in the commercial world: artificial intelligence, as a business, has not exploded as expected. Progressively – to the dismay of the many gold seekers in Silicon Valley, but not to the surprise of true scientists – it has turned out that the limiting factor in the development of AI is not hardware, not even software, it is the capacity of human beings to model satisfactorily a larger and larger number of problems that were not previously solved on the computer. So AI, and expert systems have developed and matured, but not at the rythm of electronic circuit technology. As a result, many companies have dropped out of the field entirely and a company like Thinking Machines Corporation has fundamentally redirected its marketing efforts away from AI. So it is certainly the case that some AI applications are compute bound –and you have worked on them–, but the pressure to solve these problems is not drastically different of that of solving a number of other scientific problems.

In view of the remarks above, my assessment is as follows. The FGCS project has accumulated considerable experience on MIMD computing, in terms of programming and architecture as well. This experience is probably unique. All methods that would be applicable only for a small number of processors have been rejected off hand. This is a very sound approach for basic research. In terms of software, you have designed and implemented bold and elegant ideas. I believe that many of these ideas will be useful, and used, outside the ideological context of FGCS. The basic research that you have performed has been published in the open literature, it is deep and durable, it has earned you the esteem of many scientists around the world. Globally, my opinion is very positive.

Now, I will turn to the aspect of your work that is closest to my personal area of research.

Language Issues

A priori, all of the work of FGCS revolves around one language, KL1. KL1 is an original construction. Aspects of KL1 are described in many papers. The implementation of KL1 must be fairly similar on all of your hardware platforms, otherwise you would have a difficulty porting PIMOS, and applications. It would also be difficult to train new users. Yet, I see no single report which is "the definition of KL1", that I could read at leisure to form a precise opinion of the language. I think one of you wrote that KL1 = FGHC + meta - control + convenient-things. That leaves a bit too much room for my imagination. In fact, I am not totally certain, given my previous understanding of ESP, that the logic programming aspect of KL1 is so important in comparison with the Object-Oriented Methodology of using KL1, that Chikayama-san alluded to in his presentation.

In any case, even if KL1 is very well designed, it is not the only language that you have designed. I have heard of A'UM, AYA, MENDELS ZONE, GDCC, cu-Prolog. The dream of having one single language to implement everything, no matter whether your are a systems or an application programmer is long gone. In fact, facing a multiplicity of languages is unavoidable. But luckily, we know now that this diversity is tractable, thanks to the advent of distributed computing. So KL1, like all programming languages, needs

a detailed evaluation of its features: what is used by systems programmers; what is important for programs that generate KL1 code; what should be in libraries rather than as a primitive of the language; what are the protocols that should be used to interface to other languages, because in the past ten years, we have all learnt that there are very few "purely AI" applications. In the process of this analysis, you may also reexamine where hardware language support was essential. This was impossible to assume in 1981, but now we know that microprocessors supporting 64 bits of address space are here.

Basic research must elucidate, by analysis and experimentation, what the basic mechanisms and the basic protocols are. I feel that part of this remains to be done for KL1, although the really costly part, building an implementation and accumulating experience in building operational software with it, has been done thoroughly in the project.

Technical questions

I would like to list here a number of technical questions that have come to my mind during your presentations. The fact that I ask such questions show that I take extremely seriously the work of the project, and that I feel that it is necessary to understand your design decisions, to appreciate them fully. First, I would like to understand with greater precision the innovations of PIMOS, in comparison to other distributed operating systems such as AMOEBA, GUIDE or CHORUS. I am convinced that PIMOS's ideas are very general and quite unconnected to Logic Programming, frankly. My second question concerns memory management. As a Prolog user, I rarely have problems with speed, but I keep fighting with the memory management schemes of the various Prolog systems. I wonder whether you have looked at the remarkable work of Bekkers and his colleagues at IRISA, in France. On a similar line, I understand that you were focussed primarily on parallelism, but some schemes for extending the applicability of logic programming have appeared in the last ten years. In particular, given your interest in Theorem Proving, I would have thought that Lambda-Prolog, an extension of Prolog that includes terms with binders, should be of interest for you.

Several groups have designed languages at ICOT. Were certain principles of language design systematically used, did you design or use general tools for this task? Certainly the technology that you have developed can be useful there.

To conclude this paragraph, I believe that having unity of purpose is extremely useful; it gives everyone a sense of a global objective, a way of measuring progress. But basic research has its own logic as well. If you pass near an important scientific problem and do not treat it because it is not squarely in the direction of your project, this will be unfortunate. Because we are in 1992, I cannot avoid the banale remark that Columbus did not look for America.

Conclusions

In basic research, ten years is NOT a very long time. As I mentioned earlier, research on parallel processing has been with us at least since the mid-fifties. Many many subjects of Computer Science have taken longer than that to mature. In the case of an extremely original project like FGCS, everyone could –and did– predict that if you were serious about basic research, then it would take more than ten years to reach your objectives. It may be necessary to arrange the pursuit of your goals differently, but it is in the nature of good basic research to create constantly new and unforeseen problems.

In terms of technical achievements, the FGCS has produced many technical papers, it has accumulated considerable competence on software for MIMD machines, on how to harness the power of a large number of processors. It has defined and followed what I believe are fundamentally good strategic

directions of research: software for parallel processing, theorem proving, Object-oriented Operating systems, computing problems connected with the law or the understanding of the human genome.

In terms of social achievements, the success is truely remarkable. The project has developed basic research in computing on an unprecedented scale in Japan, supporting a number of activities outside ICOT as well. The Journal that it has fostered, New Generation Computing is a good scientific journal. ICOT has developed friendly and fruitful international contacts with many countries and scientific institutions. This cannot be overemphasized. For many of us, the project was absolutely crucial in opening and maintaining contact with Japan. Finally, the project has maintained faith in Artificial Intelligence as a fundamental research topic with a scientific basis.

I cannot imagine, at this stage, that ICOT will stop brutally. I have understood that you plan to release software to the world, in a novel policy of your government. On the basis of the little experience I have in this area, I would say that few researchers in the world will want to use software if they know that no one is there to maintain, develop and improve it, as well as basing its own research on it. Software that has not changed for 18 months is considered dead. Of course, free software is not maintained like commercial software, but researchers must have the feeling that the authors of the software still care for it before they use it.

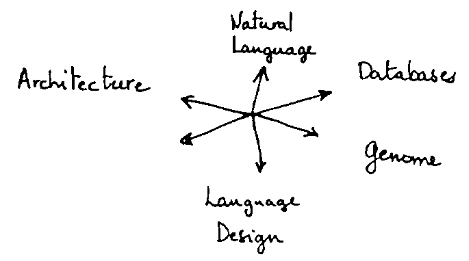
Tokyo, June 4 1992

G.KAHN

FGCS
assessment

—

Scope:



Broad, with unity of purpose

Question?: "integration" yes KLI, but..

Example: Build NL output for theorem provers

Reporting: **Very high quality talks**

Laboratory chiefs show experience, maturity, openness

Demos show strong & weak points

International Seminars: wonderful

Small objection: presentations too "politically correct"

A problem:

Concurrency

A line of attack

Logic Languages

+

- original
- software view
- tractable
- open
- basic research
- long range
- fine grain //
- irregular
- TMC

−

- difficult ***
- misleading
- applications
- weird community
- TMC
- world changes

$\boxed{\text{*** difficulty: N.D + //ism !}}$

Assessment: MIMD / many processors } positive

Language issues

In principle :

KL1

? what is it really :

FGHC + x + y

+ Methodology of OO programming

ONE language , but :

A'UM , AYA , MENDELS ZONE , GDCC , cu-Prolog.

Facing mutiplicity of languages is <u>unavoidable</u>!
and <u>tractable</u>

Hardware language support:

Is it truely necessary (History!)

Research must elucidate : basic mechanisms , protocols

Technical questions :

PIMOS versus other distributed OS

say AMOEBA , GUIDE , etc..

Memory management
was the remarkable work of BEKKERS et al.
considered

L.P. Extensions
what about λ-Prolog ? Given accent
on Theorem Porving , should be taken seriously

Language Design
Principles , common tools ?

Z Single purpose is very useful as driving force but Basic Research has its own logic too!

(C. Colombus was not looking for America)

Conclusions

1. I disagree with Furukawa-san :

   10 years is NOT a very long time

2. Achievements of FGCS (technical)

   - many
   - Competence in software for MIMD
     large # of processors
   - good strategic directions

   software , theorem proving , OOOS , genome

3. Achievements of FGCS (social)

   - Basic Research , Journal
   - International Contacts = Friends
   - Faith in AI

4. CANNOT STOP BRUTALLY

   But beware ICOT (Free Software)
   not (ICOT Free) Software