PROCEEDINGS OF THE INTERNATIONAL CONFERENCE
ON FIFTH GENERATION COMPUTER SYSTEMS 1992,
edited by ICOT. © ICOT, 1992

Knowledge Information Processing in the 21st Century

Shunichi Uchida

Institute for New Generation Computer Technology
4-28, Mita 1-chome, Minato-ku, Tokyo 108, Japan
uchida@icot.or.jp

1 A New Research Platform

Here in the last decade of the 20th century, the beginning of the 21st century is close enough for us to be able to forecast the kind of changes that will happen to new computer technologies and in the market and to predict what kind of research fields will be the most important.

I would like to try to forecast what will happen to parallel processing and knowledge information processing (KIP) based on my experience in the FGCS project.

It is quite certain that the following two events will happen;

1. Large-scale parallel hardware will be used for large-scale problem solving.
2. Symbolic processing and knowledge information processing applications will be extended greatly.

However, it is not so obvious whether these two events will be effectively combined or will remain separate. The key technology is new software technology to enable us to efficiently produce large and complex parallel programs, especially for symbolic and knowledge processing applications. If this parallel software technology is provided with large-scale parallel hardware, a very large change will happen in the market in the 21st century. I think that the FGCS project has developed the kernel of this key technology and shown that these two events will surely be combined.

In the FGCS project, we proposed the hypothesis that a logic programming language family would be superior to any other language families in exploiting new software technology and applications, especially for symbolic and knowledge information processing. The first step in proving this hypothesis was to show that the above two events can be smoothly combined by logic programming. We decided to design and implement a logic language on large-scale parallel hardware.

In designing and implementing this logic language, the most important problem was to find an efficient method to realize the following two very complex mechanisms;

1. An automatic process synchronization mechanism based on a dataflow model
2. An automatic memory management mechanism including an efficient garbage collection method for distributed memories

These mechanisms greatly reduce the burden of parallel programming and are indispensable for implementing not only a parallel logic language but also any other high-level language including functional language such as a parallel version of LISP.

We have developed a parallel logic language, KL1, its language processor and programming environment, and a parallel operating systems, PIMOS. These are now implemented on parallel inference machine hardware, PIM hardware, which connects up to 512 processing elements. We have also developed a parallel DBMS called Kappa-P on the PIMOS. We call all of these software systems FGCS basic software.

Through the development of experimental parallel application systems using this basic software, we have already experienced that we can efficiently produce parallel programs which make full use of the power of parallel hardware.

This basic software is now available only on PIM hardware which has some hardware support to make KL1 programs run faster such as tag handling support or a large capacity main memory. However, recently, it has been announced that many interesting parallel hardware systems are to appear in the market as high-end supercomputers aiming at large-scale scientific calculations. Some of them have an MIMD architecture and employ a RISC type general purpose microprocessor as their processing element.

It is certain that the performance and memory capacity of these processing elements will increase in the next few years. At that stage, it will be possible to implement the FGCS basic software on this MIMD parallel hardware and obtain reasonable performance for symbolic and knowledge processing applications. If this is implemented, this parallel hardware will have a high-level parallel logic programming environment combined with a conventional programming environment.

This new environment should provide us with a powerful and widely-usable common platform to exploit knowledge information processing technology.

2 KIP R&D in the 21st Century

2.1 Knowledge representation and knowledge base management

The first step to proving the hypothesis that the logic language family is the most suitable for knowledge information processing is to obtain a new platform for further research into knowledge information processing. For this step, a low-level logic language, namely, KL1 was developed.

The second step is to show that a logic language will exploit new software technology to handle databases and natural knowledge bases. The key technology in this step will be knowledge representation and knowledge base management technology.

Using a logic language as the basis for knowledge representation, it should be a natural consequence that the knowledge representation language has the capability of performing logical deduction.

Users of the language will consider this capability desirable for describing knowledge fragments, such as various rules in our social systems and constraints in various machine design. The users may also want the language to have been an object-oriented modeling capability and a relational database capability, as built-in functions.

Currently, we do not have good criteria to combine and harmonize these important concepts and models to realize a language having these rich functions for knowledge representation.

The richness of these language capabilities will always impose a heavy overhead on its language processor. The language processor in this case is a higher-level inference engine built over a database management system. It is interesting to see how much the processing power of parallel hardware will compensate for this overhead.

In the FGCS project, we developed a database management system, Kappa-II based on the nested relational model. It was implemented on a sequential inference machine, PSI, for the first time. Now, its parallel version, Kappa-P written in KL1, has been built on the PIM hardware. Over Kappa-P, we have designed a knowledge representation language, Quixote and a KBMS based on the deductive and object-oriented model. Its first implementation has been completed and is now under evaluation. Quixote is one of the high-level logic languages developed over KL1. These evaluation results should provide very interesting data for forecasting database research at the beginning of the 21st century.

Another high-level logic language developed in the FGCS project is a parallel constraint logic programming language, GDCC. GDCC has a constraint solver in its language processor which can be regarded as an inference engine dedicated to algebraic problem solving.

Another kind of inference engine is a parallel theorem prover for first order logic which is called a model generation theorem prover, MGTP. This prover is now used as the kernel of a rule-based reasoner in a law expert system, also known as the legal reasoning system, Hellic-II.

These logic languages and inference engines will be further developed during this decade. They will be implemented on large-scale parallel hardware and will be used as important components to organize a new platform to build a knowledge programming environment in the first decade of the 21st century.

2.2 Knowledge programming and knowledge acquisition

The third step to proving the hypothesis is to show that a knowledge programming environment based on logic programming will efficiently work to build knowledge bases, namely, the contents of a KBMS.

Knowledge programming is a programming effort to translate knowledge fragments into internal knowledge descriptions that are kept and used in a KBMS.

This process may be regarded as a conversion or compiling process from "natural" knowledge descriptions, which exist in our society for us to work with, into "artificial" knowledge descriptions, which can be kept in the KBMS and used efficiently by application systems such as expert systems. If this process is done almost automatically by some software with a powerful inference engine and knowledge base, it is called "knowledge acquisition". Some people may call it "learning".

In human society, we have many "natural" knowledge bases such as legal rules and cases, medical care records, design rules and constraints, equipment manuals, language dictionaries, various business documents and rules and strategies for game playing. They are too abstract and too context-dependent for us to translate them into "artificial" knowledge descriptions.

In the FGCS project, we developed several experimental expert systems such as a natural language processing system, a legal reasoning system, and a Go playing system. We have learned much about the problems of how to code or program a "natural" knowledge base, how to structure knowledge fragments to be able to use them in application programs, and so on.

We have also learned that there is a big gap between the level of "natural" knowledge descriptions and that of the "artificial" knowledge descriptions which current software technology can handle. We were forced to realize again that "natural" knowledge bases have been built not for computers but for human beings. The existence of this large gap means that current computer technology is not intelligent enough to accept such knowledge bases.

It is obvious that more research effort is needed to build much more powerful inference engines that will provide us with much higher-level logical reasoning functions based on formal and informal models such as CBR, ATMS and inductive inference. In parallel with this effort, we have to find some new methods of preprocessing "natural" knowledge descriptions to obtain more well-ordered forms and structures for "artificial" knowledge bases. For example, we have to create new theories or smodeling techniques to explicitly define context-dependent information hidden behind "natural" knowledge descriptions. The situation theory will be one of these theories.

It is interesting to see how these powerful inference engines will relate to knowledge representation language and knowledge structuring methods. Another interesting question will be to what extent the power of larger-scale parallel hardware and parallel software technology will make these higher-level inference functions practical for real applications.

It is certain that research into knowledge information processing will continue to advance in the 21st century, opening many new research fields as it advances and leaving a large growing market behind it.