BASIC SOFTWARE SYSTEM

Koichi Furukawa and Toshio Yokoi

ICOT Research Center
Institute for New Generation Computer Technology
Tokyo, Japan

ABSTRACT

The basic software system is a core software for the Fifth Generation Computer Systems. A bridge to fill the gap between a highly parallel computer architecture and knowledge information processing is strongly needed in order to build a highly parallel super computer for knowledge information processing. In this project, "logic programming" was selected as a conceptual bridge. The basic software system is supposed to play the role of an actual bridge. For the initial stage of the project, it has been assigned to design the entire system and develop the underlying technology necessary for implementing its subsystems. Although this project has been in operation for only two and a half year, and achieved only a small portion of the entire research plan, we are convinced our approach based on logic programming is very promising. This paper describes the current status on our research and development with the basic software system.

1 INTRODUCTION

The target of the Fifth Generation Computer Systems (FGCS) Project is to build a highly parallel super computer for knowledge information processing. In this project, "logic programming" was selected as a bridge to fill the gap between a highly parallel computer architecture and knowledge information processing, as shown in Fig. 1.

To implement an actual bridge, a sophisticated software system which we will call the basic software system is required. It is supposed to be composed of the following five modules, as shown in Fig. 2:

(1) Kernel language / Knowledge programming language

(2) Problem-solving and inference software module

(3) Knowledge base management software module

(4) Intelligent interface software module

(5) Intelligent programming software module

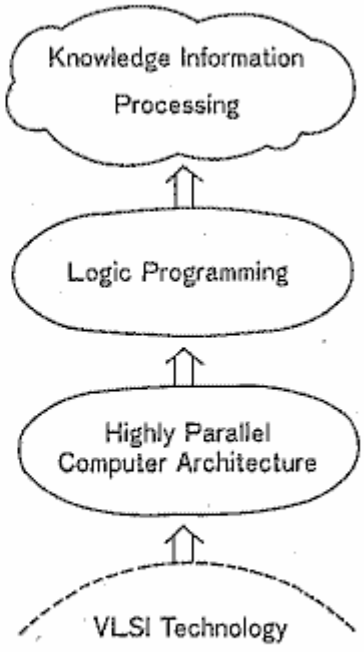

Fig. 1 The Fifth Generation Project

The initial stage of the project will be the development of the underlying technology necessary for implementing each module. The modules will be developed independently and will then be integrated to form a total basic software system.

The integration, which is a crucial problem in developing the Fifth Generation Computer Systems, will be achieved by using a common programming language in their development. In this project, logic programming has been selected for this purpose. The development of a series of kernel languages, KL0, KL1 and KL2 is planned. These are logic programming languages which define an abstract interface between the hardware and the software. As a first step in 1982, the machine language KL0 for the Sequential Inference Machine (SIM) and its user language ESP (Extended Self-contained Prolog) were designed as the common languages.

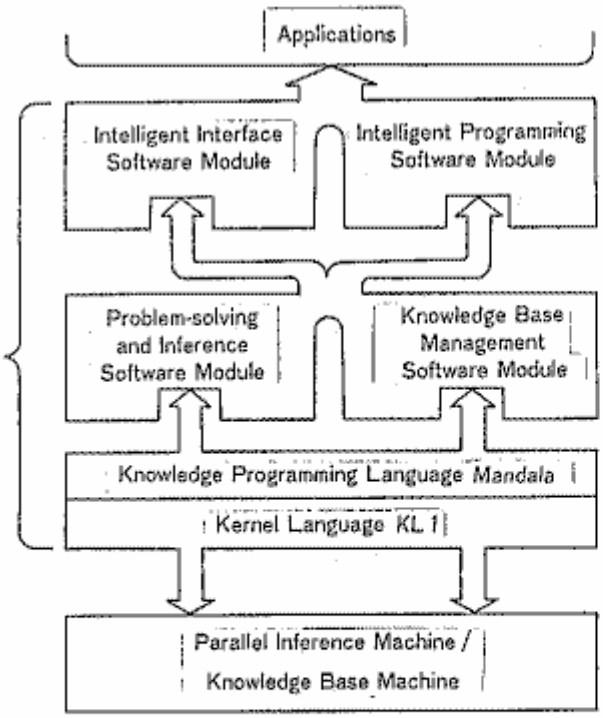

Fig. 2 The basic software system

The language pair KL0 and ESP is not sufficient for the development of a final basic software system which satisfies the target shown in Fig. 1. KL1 as well as its user language Mandala are now being designed for the purpose of developing a firm base to achieve the target. The essential extention of KL1 into Prolog is a stream-and-parallel function which will be used to describe the behavior of multiple objects acting in parallel and will be executed in parallel on a Parallel Inference Machine (PIM). The user language Mandala (Furukawa, *et al.* 1983c) for KL1 is also intended to be a knowledge programming language for building various application systems related to knowledge information processing. This will enable extraction of a large amount of parallelism. It is therefore important that each function in Mandala are implemented with the most straightforward method possible.

The integration of the problem-solving and inference software module and the knowledge base management software module into a single framework is planned, and these two modules will be combined using Mandala to establish a powerful knowledge representation system.

As far as the problem-solving and inference function is concerned, the greatest challenge for the next stage is to incorporate parallelism. Cooperative problem solving will be the key issue in achieving this goal. We investigated to implement a few parallel inference engines in Mandala. Concurrent Prolog (CP) interpreter written in CP is one example. Another example is a pure Prolog interpreter in CP. The implementation of a first order theorem prover is planned as well as a term rewriting system in KL1.

In dealing with knowledge base management functions, the areas requiring special attention are (1) how to deal with very large

knowledge bases, (2) the investigation of the knowledge acquisition problem and (3) the design of a knowledge representation system.

With regard to (1), the newly developed Prolog machine PSI (Personal Sequential Inference machine) and relational database machine Delta are going to be connected via local area network INI (Internal Network in ICOT) as a first step toward realizing a very large knowledge base.

For (2), knowledge acquisition problems have been approached from a logical point of view. Formulation of consistency checking, rule induction from a set of facts and truth maintenance in a logical framework were implemented in Prolog using the meta programming feature.

For (3), these two systems will be rebuilt in KL1 and will be combined using Mandala to realize an ultimate knowledge representation system in the next stage.

Intelligent man-machine interface functions and intelligent programming functions have also been studied within the basic software system's research. This research plays a double role in the FGCS project: the establishment of techniques for building parts of the basic software system and the evaluation and increase of the usefulness of the logic programming approach.

For the intelligent man-machine interface function, we concentrated on natural language understanding research and pursued three subjects: (1) the development of a powerful parsing system, (2) the investigation of the discourse understanding problem, and (3) the design of a set of machine readable dictionaries.

For the intelligent programming function, many rather independent research activities were being performed to determine the key issues required for realization of the function. They include software specification, program understanding, program verification, program transformation and automatic programming.

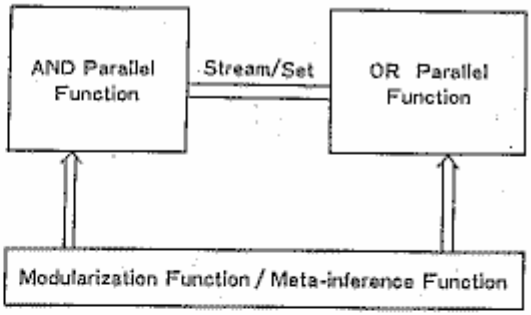

Fig. 3 Conceptual configuration of KL1

In the following, more detailed descriptions will be given for the above five modules of the basic software system.

2 KERNEL LANGUAGE

The kernel language for the Fifth Generation Computer Systems will evolve from KL0, the machine language of SIM developed in the initial stage of this project. It will be further developed through KL1, which is being developed for use in the intermediate stage and beyond with primary emphasis placed on the extension of the parallel execution function, and will become Kernel Language version 2 (KL2), which will have knowledge representation support function and will be developed in the intermediate stage and used in the final stage.

This paper will concentrate on KL1. KL1 has four major functions as shown in Fig. 3. They are the and-parallel function, the or-parallel function, the modularization function (Furukawa, *et al.* 1983b) and the meta-inference support function. In the following, each function will be described briefly.

(1) And-parallel function

The and-parallel function is introduced

to describe the behavior of objects in a problem domain and thereby to support object-oriented programming (Shapiro 1983a). There are several logic programming languages with and-parallelism; Relational Language (Clark, *et al.* 1981), PARLOG (Clark, *et al.* 1984) and Concurrent Prolog (Shapiro 1982b, 1983b, 1983c, Takeuchi 1982, 1983, 1984). The detailed design of KL1 is not finished yet, but these languages will greatly affect the design.

(2) Or-parallel function

A very important element of KL1 is the or-parallel function. Among the many applications of knowledge information processing, those intended to achieve what might be considered as artificial intelligence often deal with problems that can be solved only by searching for solutions among a number of possibilities. The or-parallel function handles such search problems. The relational database machine developed in the initial stage of this project can be regarded as a kind of search-all-solutions machine. Pure Prolog, which is obtained by eliminating sequential control from standard Prolog, is a language for representing the search for all solutions.

These two constituent elements of KL1 — and-parallel function and the or-parallel function — are connected by the stream-set interface. In other words, all solutions are obtained as a set in the or-parallel subsystem. This set is then handled in the and-parallel subsystem as a stream of data (Hirakawa, *et al.* 1983b, 1984a, Yokomori 1984).

(3) Modularization support function

One major requirement of KL1 is the efficient implementation of Mandala, an object-oriented knowledge programming language based on KL1. A function to support modularization and a meta-inference function are necessary to meet this requirement. These two functions are closely related. The modularization function has two objectives: one is to improve software development (program generation, management, etc.) at the system program level, and the other is to support the structuring of knowledge into hierarchies and multiple worlds. Basic functions are to be implemented in KL1 to attain these goals. These include localized predicates used exclusively in modules, the external reference function required to parameterize modules, and the hierarchical structuring of modules.

(4) Meta-inference support function

Although it is extremely difficult to realize efficient implementation of meta-inference function, such realization is not only important for the implementation of Mandala, as described above, but also greatly influences the implementation of an intelligent editor and debugger, the development of interfaces among machines operating in parallel, and applications of cooperative problem solving. The meta-inference functions are based on the modularization support function, which includes operations for solving given goal by specifying a compiled program (Kunifuji, *et al.* 1984a, 1984b). Using this function as the core, it is necessary that control methods be parameterized to facilitate more flexible control.

The preliminary specification of KL1 was published in 1983 (Furukawa, *et al.* 1984b) and language details are now being designed. The previous specification was reviewed in terms of efficiency by actually developing a prototype of the language processor (Miyazaki 1984, Ueda, *et al.* 1983, 1984). The expressive power of KL1 was also investigated by writing a simulation program of a small electronic circuit in Mandala, the user language of KL1. It turned out that the efficient implementa-

tion of the modularization function and the meta-inference function is the key issue in achieving high efficiency in such complex programs as those used in real application.

3 PROBLEM-SOLVING AND INFERENCE ISSUE

The problem-solving and inference software module and knowledge base management software module are the two central parts of the knowledge information processing system. One of the most significant research items planned for the intermediate stage of the project is the incorporation of parallelism with those modules. Therefore, it is necessary that the problem-solving and inference software module as well as the knowledge base management software module are designed so that they can be integrated into a parallel execution environment.

There are two important factors in designing a module for a parallel execution environment. One is to achieve parallelism of the problem-solving function itself. The other is to control problem-solvers operating in parallel. In addition, in designing submodules, consideration was given to high-level inference functions for solving more fundamental questions.

The goals for the initial stage were (1) extraction of the key functions / components necessary for developing the problem-solving and inference software module in the next stage, and (2) the design of each function / component by prototyping. Extracted functions / components are [1] parallel inference function, [2] meta inference function, and [3] powerful inference engines. In the following, each function / component will be described briefly.

(1) Parallel inference function

The aims of introducing the parallel inference function are not only to speed up the inference process but also to enhance inference capability to realize distributed problem solving. This function is deeply related to the problem of parallel execution of logic programming languages. An or-parallel Pure Prolog interpreter called POPS (Hirakawa, *et al.* 1983b) was developed in CP. It performs or-parallel computation by converting it into stream-and-parallelism in CP. With POPS, a lazy computation was successfully realized as well as eager one (Hirakawa, *et al.* 1984a).

As far as distributed problem solving is concerned, Shogi (a chess-like game in Japan) was selected as an example domain and the human problem solving at the end game of a Shogi game was investigated. It turned out that distributed problem solving gives a global framework for formulating it and a new model called knowledge architecture was proposed (Kondou 1984), which consists of field (field functions as communication media), cognition-type knowledge, memory-type knowledge, control-type knowledge and object model (object model corresponds to the game board in Shogi).

(2) Meta-inference function

Inference processes may be controlled by directly attaching rules to be applied after to each rule or by constraining the use of rules by more general (e.g., grammatical) rules. The meta-inference function controls inference using such control information (Kunifuji, *et al.* 1984a, 1984b, Nakashima, *et al.* 1984).

The demo predicate is well known as a tool for implementing the meta-inference function in a sequential execution environment. The demo predicate has the function of demonstrating that the given goal statements can be proved in a set of axioms, using the given control information. We have developed a predicate called simulate, which

performs the same function in a parallel execution environment. The simulate predicate is also used to implement the Mandala interpreter (Kunifuji, *et al.* 1984a, 1984b).

(3) Powerful inference engines

Since the built-in inference engines in KL1 are Horn clause deduction engines for Pure Prolog and Concurrent Prolog (or similar stream-and-parallel language), more powerful inference engines are needed for many purposes such as natural language understanding, formula manipulation, game playing, program synthesis and so on.

Two inference engines were investigated for that purpose: a first order theorem prover and a term rewriting system. Two types of first order theorem provers were designed and implemented based on linear resolution (Mukai, *et al.* 1984) and lock resolution. In addition, it was expected that another possible inference mechanism would be found which could be applied to build PIM (a Parallel Inference Machine). It turned out that lock resolution is a suitable strategy to incorporate parallelism in theorem proving.

To extract an actual requirement from a real world application, an automatic layout problem in electronic circuits design was investigated (Koseki 1984, Mitsumoto, *et al.* 1984b, Mori, *et al.* 1984a, 1984b). The research revealed that we need specialized inference components for dealing with lower level processing such as performing a fast systematic algorithm in wiring and displaying an obtained layout on a graphic terminal. It may be a temporal situation that such extra components other than logic programming are needed. However, at least currently, the connection between Prolog and Fortran turned out to be quite useful in developing real world application systems (Goto 1984, Mitsumoto, *et al.* 1984a). Two-dimensional programming was also investigated aiming for providing a basic technique to deal with spatial problems (Furukawa, *et al.* 1983a).

4 KNOWLEDGE BASE MANAGEMENT ISSUE

The knowledge base management software module constitutes a central part of the basic software module as well as the problem-solving and inference software module. This issue was approached by studying four topics: (1) realization of a very large knowledge base, (2) formulation of knowledge acquisition in logic, (3) design of a knowledge representation system, and (4) development of expert systems. These studies will be reflected to build the knowledge base management software module in the next stage. The following are descriptions for each topic.

4.1 Very Large Knowledge Base

As a first step toward constructing a very large knowledge base, combining a Horn clause deduction system with a relational database in both the hardware level and the software level has been attempted (Kitakami, *et al.* 1984e).

For the hardware level connection, a method for combining the newly developed Prolog machine PSI with the very large relational database machine Delta via the local area network INI is being investigated. At the same time, the mechanism for directly connecting those two machines is also investigated.

For the software level connection (Kitakami, *et al.* 1984c), a new method for linking Horn clause deduction with relational algebra was proposed (Kunifuji, *et al.* 1982, Yokota, *et al.* 1983a, 1983b). It is based on a so-called compiled approach and it generates a sequence of relational algebra formulas to handle queries on recursively defined relations.

An experimental database management system supporting the actual combination of PSI with Delta has been designed and is being implemented on PSI. It is called KAISER (Knowledge Acquisition-oriented Information SuppliER) and it also includes such functions as user friendly interface and knowledge acquisition support.

The knowledge acquisition issue will be discussed later in more detail. KAISER also provides a prototype of a distributed knowledge base system since it manages a local database of PSI as well as the global database in Delta. A relational database system on PSI's file system was also designed and is being implemented to realize the local database.

As far as the user friendly interface is concerned, the main emphasis was put on conversation control such as recognizing the change of topics and guessing elliptical input. The target is the construction of a rather domain independent conversation control system which will be used in many different actual situations (Miyachi 1984c).

4.2 Knowledge Acquisition

The knowledge acquisition problems have been approached from a logical point of view. As a consequence of the epistemological analysis (Kunifuji, *et al.* 1984c, 1984d), it was found that the knowledge acquisition process consisted of the knowledge assimilation process (Miyachi, *et al.* 1983a, 1983b, Kunifuji, *et al.* 1983a), the knowledge accommodation process and the knowledge equilibration process. Therefore, KAISER has the knowledge acquisition functions (Kitakami, *et al.* 1983c, 1983e, Kunifuji,*et al.* 1984c) which support the above-mentioned processes. Using a meta-inference function, all of them have been systematically implemented in Prolog. The conceptual diagram of these functions is shown in Fig. 4 (Kitakami, *et al.* 1983e, 1984a). The diagram consists of several functions such as, the meta-inference function, the knowledge assimilation function, the knowledge accommodation function, and the knowledge equilibration function.

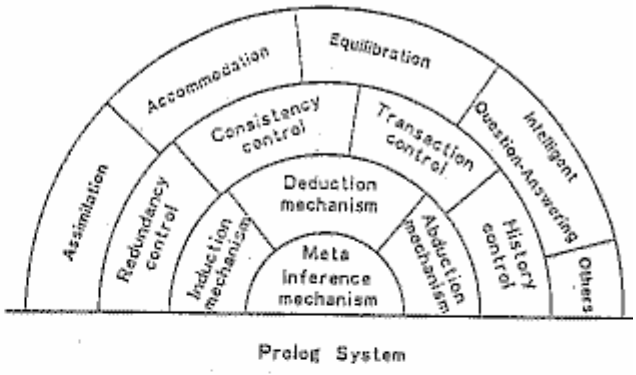

Fig. 4 The knowledge acquisition module

The meta-inference function controls (monitors) information about how to use object knowledge. The meta-inference mechanism (Kunifuji, *et al.* 1983b, Kitakami, *et al.* 1983e) is essential for the implementation of the knowledge acquisition process. In short, the "demo" predicate suggested in (Bowen, *et al.* 1982) is used as the primitive in this function, and is implemented by extending a Prolog interpreter written in Prolog.

The knowledge assimilation function (Miyachi, *et al.* 1984a, 1984b, Kitakami, *et al.* 1983a, 1983b) systematically assimilates knowledge from the external world that does not contradict the requirements of the integrity constraints (Kitakami, *et al.* 1983d) specified in the knowledge base. At the same time, the function is optionally provided with a mechanism to remove redundancy (Miyachi, *et al.* 1984b).

In the knowledge accommodation process, an erroneous piece of knowledge in the current knowledge base is corrected to avoid inconsistency assuming that new facts are true. An experimental knowledge accom-

modation system (Kitakami, *et al.* 1983d, 1983e, 1984a) was developed by enhancing Shapiro's Model Inference System (Shapiro 1982a) using the idea of integrity constraints which describe general conditions that a class of models has to satisfy. By this enhancement, we succeeded in reducing the amount of negative examples given to the system for locating bugs.

The knowledge equilibration function (Kitakami, *et al.* 1984c, 1984d, 1984e) has a function to adjust consistency between premise-type knowledge and assume-type knowledge (belief) (Kitakami, *et al.* 1984b). It provides a function to revise a belief in a truth maintenance system (Doyle 1979, Goodwin 1982, Martins 1983). Belief revision is realized by constructing a proof-tree through a meta-inference process and replacing the most uncertain piece of knowledge in the proof-tree by the alternative given by the user, if any, or the negation of the original one.

In KAISER, these functions were realized elegantly by enhancing the "demo" predicate refered to above (Kunifuji, *et al.* 1984b). This demonstrates the advantage of the choice of logic programming. In the future, an integrated knowledge acquisition system on KL1 will be developed to increase efficiency by making it to run in parallel, and also to combine with Mandala.

4.3 Knowledge Representation

Knowledge representation is a major challenge for research in artificial intelligence. Since it is an extremely difficult problem to invent a single formalism to represent all kinds of knowledge, a knowledge programming language which can be used to build specialized knowledge representation systems was designed and implemented. At Working Group 4 (chairman: Prof. Mizoguchi, F., a committee in charge of investigating consultation systems), a preliminary study was made for designing such a language (Mizoguchi, *et al.* 1984). The language is called Mandala (Furukawa, *et al.* 1983c, 1983d, 1984a) and a detailed description is given in (Furukawa, *et al.* 1984c). Mandala is expected to play an important role in the Fifth Generation Computer System, as shown in Fig. 2. The features are summarized as follows:

1. It is not only a knowledge programming language but also a basis for a knowledge base management system. The nature of this duality comes directly from the capability of double interpretations of Horn clauses: procedural interpretations and declarative interpretations.

2. It incorporates parallelism both in problem description due to its ability to support object-oriented programming, and in execution due to the property of its base language KL1.

3. It provides a variety of programming styles, thus becoming a powerful tool for describing knowledge information systems. Particularly, it enables dynamic behavioral description as well as static property description.

Strong programming support is necessary to allow the user to utilize Mandala as a knowledge programming system. The knowledge base editor plays the role of such a support system. As indicated by the dual nature of Mandala, the knowledge base editor is analoguous to a support environment for normal programming functions, such as tracer, debugger, and editor. Parallelism has become more important as the base programming language evolves from KL0 to KL1. An approach utilizing the knowledge programming system may be more successful in implementing these features. In Mandala, knowledge is formalized in predicate logic, which facilitates knowledge base manage-

ment functions, such as consistency check.

The knowledge base editor edits programs representing knowledge written in Mandala. Here, editing not only means the editing of static programs but also editing of worlds manipulated by programs after these worlds have been changed. Therefore, the knowledge base manager and the user interface, which are both required at the time of program execution, are also regarded as constituents of the knowledge base editor. The unit-world and the instance editor edit basic elements of Mandala – unit worlds (fragments of KL1 programs, modules) and instances (running processes of KL1 programs), respectively. A prototype of a knowledge representation system for natural language understanding research was implemented in Prolog (Sugiyama, *et al.* 1983). It was utilized to design the knowledge base editor for Mandala as well as the one for ESP.

4.4 Expert Systems

In this section, the experimental knowledge utilization system is described. The aim is to clarify what function is needed for actual use, and therefore, what ability is required for the knowledge utilizing computer system.

Starting from preparatory investigations in fiscal 1982, two experimental systems, a Japanese proofreader system and a logic design support system were selected. Now, both experimental systems are under implementation.

4.4.1 Japanese proofreader system

The Japanese proofreader system is a knowledge utilization system which finds errors in Japanese text and corrects them when possible. This system is assumed to form a part of a computerized Japanese documents preparation system.

First, proofreading techniques were studied and computerization capabilities were considered using about 4,000 lines of data obtained from a Japanese newspaper company. Results showed that 56% of the corrections needed in-depth sentence or text understanding to computerize. This is left as a future problem.

On the other hand, 44% of the corrections were relatively easy to computerize under present conditions, using knowledge in dictionaries and a stylebook, and by consulting directories (Ishii 1983b, 1984). Thus, the system was designed to handle such knowledge effectively. The knowledge structure is an important problem and has been studied carefully (Ishii 1983a).

4.4.2 Logic design support system

The logic design support system is a knowledge utilization system that processes the logic design of computer hardware. This system converts design specifications provided as hardware operation algorithms into connections among existing circuit components.

Using only mechanical conversion processes, redundant design would be obtained, rendering the system useless. Conversely, human beings use various types of know-how accumulated through experience to get a good design.

The logic design support system aims at incorporating such know-how in Prolog and, thus, at achieving greater sophistication in logic design by computer (Maruyama, *et al.* 1984). It turned out that such know-how is well described in Prolog. Use of a structured knowledge representation system will also be a future problem.

5 INTELLIGENT MAN-MACHINE INTERFACE

5.1 Parsing System

The purpose of this research is to develop a parsing system which provides users a high level grammar description language and an efficient syntactic analysis facility.

Our approach is based on logic programming language. As already pointed out, logic programming language and the Context Free Grammar (CFG) have a close relation, i.e. a Horn clause can be seen as providing the procedural interpretation of a CFG rule. It is natural to take the CFG as the basis of our syntactic analysis. Our main concerns are the development of an efficient analysis method for CFG on logic programming language and the design of a flexible and powerful grammar description system. There are two major parts in the research , i.e. a syntactic analysis method and a grammar description system. Following is a short report on their current status.

5.1.1 Syntactic analysis method

We developed an augmented CFG parsing system, called the BUP system, in cooperation with ETL (Electrotechnical Laboratory) and TIT (Tokyo Institute of Technology) (Matsumoto, *et al.* 1983, Tanaka, *et al.* 1984a, Yokoi, *et al.* 1982a, 1982b). The parser employes a bottom-up depth-first algorithm and is naturally embedded in Prolog. This feature makes the parser efficient. Furthermore, the BUP system employes the optimization method by recording the intermediate results of a parsing process. It also comprises several tools for grammar development; the BUP tracer, the BUP translator, the epsilon reducer, and so on.

Besides the development of the BUP system, we developed a method for applying parallel execution mechanisms to syntactic analysis. The parallel parsing system is based on the chart parsing algorithm, and implemented in Concurrent Prolog (Hirakawa 1983a, Hirakawa *et al.* 1983b, 1984b). The parsing model comprises the multiple processes and message transfers between them.

In connection with a parsing system, a morphological analysis system has been developed for analyzing Japanese sentences. Since it is an agglutinate language, the morphological processing of Japanese is more complicated than English. The morphological rules include word-inflection information and word-connection conditions. The morphological rules are represented by DCG descriptions (Miyoshi, *et al.* 1983).

5.1.2 Grammar description system

The main concern is the research on the formal system for representing grammatical relations. The system is indispensable in checking the grammaticality of a sentence and in maintaining a large-scale grammar.

In practice, the new grammatical theories in current linguistics, LFG and GPSG, are adopted as candidates for our system. They provide universal accounts for linguistic phenomena and a powerful grammar description system with a highly understandable form. Hence they are useful in syntactic analysis. The LFG system is implemented in DEC-10 Prolog (Yasukawa 1983, 1984). The computational mechanism of LFG can be realized by introducing the equality on f-structures into Prolog. The grammar rules of LFG can be translated into DCG rules. Both top-down and bottom-up parsing strategies are applicable to them. The implementation method of the GPSG system is currently being discussed by ICOT's Working Group 3 (chairman Prof. Tanaka, H.). The research topics are an efficient parsing algorithm for the GPSG framework and development of a basic Japanese grammar in GPSG.

From the view of the refinement and extension of DCG, the GDL0 (Grammar

Description Language 0) is designed and implemented in DEC-10 Prolog (Morishita, *et al.* 1984). The GDL0 introduces the data structures for grammar category definition and macro facilities for improving the readability and modifiability of grammar. Also, a grammar description system based on the concept of an object-oriented programming language is developed (Miyoshi 1984). This system introduces the notion of class in defining grammatical relations such as Head feature Convention and Control Agreement Principle.

In the future, we will continue the research and development concerning the grammar description system. It is necessary to refine and extend the system, including the introduction of a parallel execution mechanism.

5.2 Discourse Understanding System

To establish a computational model for discourse understanding is an important and challenging problem in natural language understanding. The model has tight connection to the problem of building a flexible and user friendly natural language interface (Joshi, *et al.* 1984) into FGCS. In this subsection, we will describe a brief outline of our approach to discourse understanding and then describe our current research activities.

5.2.1 Basic ideas

There are at least four important factors in a discourse model. They are called speech act, coherency, presupposition, and mutual beliefs. The speaker and the hearer compute these factors by using so called discourse maxims, i.e., the principles of consistency and optimality for communication. This implies that the system must have a powerful inferencing ability.

We adopted the following views on discourse understanding:

(1) discourse understanding is a dynamic process where the computational principles are consistency and optimality for communication,

(2) the goal of the process is to construct a world model for the (written) utterances and mutual beliefs between the participants,

(3) the process repeats cycles of generating hypotheses about a partial model of the world and testing them using knowledge about the world or the discourse partner,

(4) the real world consists of events and objects which stand in various relationships with each other, and

(5) the world model under construction is fed back to the knowledge component of the system,

(6) the theory of a mental model and action have a crucial role in discourse understanding.

Our basic tool is situation semantics (Barwise, *et al.* 1983) developed at Stanford University by J. Barwise & J. Perry as a new paradigm for model theoretical semantics of natural languages. The theory seems to be well fitted to the above views. During the first stage, efforts will be directed toward analyzing discourse in Japanese.

5.2.2 Current activities

In the current state, we are designing a prototype for a discourse understanding system influenced by situation semantics. The system will read a story written in Japanese, construct situations described in it, and then answer to various types of questions about them.

As a case study, we chose a portion of a story text in Japanese. It describes a dangerous situation in a flying air craft with

many passengers which has engine trouble. In the story , there are several occurrences of action including speech acts between the captain and a stewardess of the air craft who are trying to do their best in the difficult situation.

We have been trying to analyze the meaning of the sentences for nearly one year. We found that there were many difficulties in giving complete meanings to all the sentences in the text.

The difficulties are in the following areas:

(1) the meanings of quantified noun phrase, adverb phrase, auxiliary verb, and speech act verb,

(2) sentence or phrase ellipsis,

(3) the representation and use of various types of knowledge to control sentence interpretation, and

(4) the dynamics of the focusing process.

We think situation semantics is surely a promising theory for discourse computational models. To solve these difficulties, we are trying to recapture the framework of language understanding in the light of situation semantics. This is necessary because it is one thing to define the meaning of a natural language sentence and another to understand its meaning.

Following are a few related technical memos (Mukai 1984d, 1984, Suzuki 1984). Case analyses of real discourse texts is a current topic at our working group WG3.

5.3 Machine Readable Dictionary

For the Fifth Generation Computer Project, it is extremely important to develop a sufficient amount of language data, above all – machine readable dictionaries which can be utilized freely. They are the fundamental database for the research on natural language processing and on the knowledge base, etc. These machine readable dictionaries will be transformed in the future into a number of application-oriented dictionaries, e.g. a transfer dictionary for machine translation and a semantic dictionary for text understanding.

The Research Group on Machine Readable Dictionary (chairman: Prof. Ishiwata, T.) has planned the dictionary development. The development process is divided into two stages. In the first stage (three years from 1984), we will develop four general purpose master dictionaries as follows (Ishiwata, *et al.* 1984) :

- Japanese dictionary
- English-Japanese dictionary
- Japanese-English dictionary
- English dictionary

Each master dictionary will contain the grammatical information described in standard published dictionaries and also that in the one oriented for computer processing.

In the second stage (two years from 1987), information about semantics and deep cases will be added and transformed into various application dictionaries such as the one for machine translation.

One of the purposes of the machine readable dictionary development is to provide the tools for deep semantic analysis. In Working Group 3, as an approach to achieve high quality natural language translation, the features of an actual translation by a professional translator have been investigated (Tanaka 1984b). As a preparatory experimental dictionary system for semantic analysis, a small scale dictionary is being developed with an entry number of 5000. It will contain a thesaurus and deep case information. This will be used for the text understanding system. In order to investigate the context where a specific word is used, an on-line KWIC (Key Word In Context) sys-

tem has been developed. This is a useful tool which provides a user with a lot of linguistic information.

6 INTELLIGENT PROGRAMMING SYSTEM

6.1 Programming Environments for Logic Programming

An advanced programming system on *SIM* is now under development. Its details are reported in another paper (Takagi *et al.* 1984). This activity seeks to get an actual base and experience for the study of an intelligent programming system.

A brief sketch of the *SIM* programming system follows:

It is designed as a collection of expert processes. A special expert *Coordinator* manages these expert processes. Typical experts are *Debugger/Interpreter*, *Editor*, and *Library*. *Debugger/Interpreter* interprets programs and provides debugging information concerning the control flow of the programs on the Procedure and Clause Box Control Flow Model. *Editor* (named Edips) is a structure-editor designed to be especially convenient for editing ESP programs. *Library* supervises class registration, loading program files, compiling, and building class objects by inheritance analysis.

Currently, these expert processes are not very intelligent. However, their intelligence will become gradually higher by adding problem-solving and knowledge-base functions to their internal mechanisms.

6.2 Program Specification, Verification and Transformation

In the intelligent programming research field, there are three major research components:

- program specification
- program verification
- program transformation

6.2.1 Program specification

One of the most difficult parts of Software Engineering Research is the program specification system. (Staunstrup 1981) In general, we are interested in both the informal (easy to understand) specification and the formal (logically established) specification.

- Natural language specification: For informal specification, we are interested in natural languages, especially in Japanese. In this regard, we are cooperating with the TELL project of Tokyo Institute of Technology, led by Prof. Enomoto, H., where both English and Japanese are considered in a limited syntactical form (Enomoto *et al.* 1984a).

There are two approaches for formal specification:

- First-order predicate calculus: This is a reasonable approach as logic programming has been adopted in our research base. The verification system is developed in this approach.
- Temporal logic (Enomoto *et al.* 1984b): The predicate calculus approach has the problem that it cannot describe the dynamic behavior of the system. The temporal logic framework is being investigated to fill the gap.

There are also other approaches, for example,

- Programming language itself: Prolog itself can be regarded as a workable specification

On the other hand, a support system for specification should be developed. In this line, several attempts were made. e.g. (Enomoto 1983, Sugimoto 1984a, 1984b)

6.2.2 Program verification

A verification system for Prolog programs is now under development. It is expected to be a foundational work for research of "Intelligent Programming Systems". In the design of our verification system, we tried to clarify the possibility the paradigm of logic programming provides us and to take advantage of Prolog characteristics as far as possible. It is especially useful that Prolog execution is a kind of inference and that Prolog semantics is formulated very simply and succinctly in the first order semantics.

First order inference in our verification system takes an extension of execution style. The main inferences are done like the execution of positive goals or like the "Negation as Failure" for negative goals. These rules plus the case splitting and the simplification rule are powerful enough to prove many properties of Prolog programs (Kanamori 1984a).

The application of computational induction to Prolog is much simpler than that to functional programs, because it can be done staying within the simple first order semantics that Prolog is based on. Induction formulas generated is also simple. This makes the verification efficient and it complements the overhead of the first order inference. (Kanamori 1984b).

Many BMTP(Boyer Moore Theorem Prover)-like heuristics controlling the applications of inferences and lemmas are integrated into the system. Especially a newly developed type inference method for Prolog is used effectively. (Kanamori 1984c).

6.2.3 Program transformation

Program transformation is a promising methodology for producing a correct program from a given (valid) specification.

From the theoretical viewpoint, it is necessary to verify the "correctness" of the transformation steps. Tamaki provides the framework for this equivalence preserving transformation (Tamaki 1983, 1984a, 1984b). Fuchi exemplifies the transformation technique to generate the pure Prolog interpreter (Fuchi 1984).

6.3 Specific subprojects, CAP and ACT^{-1}

Working Group 5 (chairman: Prof. Hirose, K., a committee in charge of fundamental theory) has been investigating the current status and future trend of the theoretical aspects of computer science. On the basis of the results obtained from the investigation, WG5 set two specific research projects. They were named CAP (Computer Aided Proof) and ACT^{-1} (Theoretical Computer Architecture).

6.3.1 CAP

The CAP project is an attempt to create proof checkers for some concrete theories in order to investigate artificial intelligence for solving mathematical problems and the ideal man-machine interface in such activities. We selected the following three target theories for the CAP project.

- Linear algebra
- Symbolic arithmetic
- Synthetic differential geometry

The first, linear algebra, is the most familiar of the three and need not be explained. The first target of proof checking is a textbook for a freshman course.

The second, symbolic arithmetic, is originated from a series of works by Sato (Sato, *et al.* 1983a, 1983b, 1984a, 1984b, 1984c). Programming language *Qute*, which itself is based on this theory, will be used for the implementation of the system. The appealing point of this theory is that the theory allows self reference as with the

Peano arithmetic. Therefore, the final goal will be the proof of the incompleteness theorem of the theory.

The last, synthetic differential geometry, is a very new area of mathematics originated by Kock (Kock 1981). This theory intends to study differential geometry in a synthetic manner by introducing infinitesimal objects. The project is still in the stage of basic theoretical survey and actual implementation will start next year.

6.3.2 ACT^{-1}

New programming paradigms such as logic programming, functional programming and object oriented programming are providing impetus to the researches of ACT^{-1}. Currently, the above three paradigms provide the separate conceptual framework within which we reason about algorithms. Moreover, each has an underlying computational model; e.g. first order predicate logic for logic programming, lambda calculus for functional programming and actor model for object oriented programming.

Our research hence consists in:

- identifying the differences and similarities of the three paradigms at the level of the computational model;
- seeking, if possible, the methods of unifying the above three paradigms at the computation level, or at a lower implementation level;
- exploring the means for exploiting parallelism (either implicit or explicit) in the above paradigms;
- evaluating programming techniques developed in each programming paradigms, e.g. lazy evaluation and stream communication using d-list;
- analyzing the complexities involved in the several implementations of the computation models;
- reviewing the current technology (of both hardware and software) for the realization of the computation models;
- relating the computation models (in the sense of points 1 and 2 above) to the more specific implementation oriented computation models such as the data flow model and the packet reduction model.

At the present research stage we are concentrating our effort on the theoretical aspects of computer architecture rather than the implementational aspects.

6.4 Transfer to Existing Technology

For research on intelligent programming, it is very important to keep tight contact with existing software technology. Therefore, a special Working Group (chairman: Prof. Saito, N.) was organized to study opinions, expectations and requests by the software industry to the FGCS project. This group investigates such subjects as how to apply new technologies to improve software development environments.

As another activity for this purpose, development of a concrete system for applying results of this project to problems in actual software development is being tried. Study and development of the system for reusing Cobol programs is an example of this activity.

7 CONCLUSION AND FUTURE DIRECTION

Although this project has been in operation for only two and a half year, and achieved only a small portion of the entire research plan, we are convinced our approach based on logic programming is very promising.

The fruitful results obtained so far are due to the concentration on logic programming. The basic research strategy is to retry difficult problems in terms of logic program-

ming. This approach may not give the best solution for each problem, but it will provide a mechanism for integrating all the results into a single system for the basic common programming framework.

The target of the next four-year stage of the project is to build each subsystem by putting ideas and/or components so far developed together. Although there is much work to be done before proceeding to the next subsystem building stage, we have been able to describe a global picture for each subsystem properly and thus to proceed further.

The main research issue in the next stage is incorporation of parallelism. This issue is a very difficult problem, which is the reason efforts have been concentrated on the design of KL1, a logic programming language with inherent parallelism.

System integration is also a difficult and challenging issue which will arise in the next stage.

ACKNOWLEDGMENTS

This research was carried out by the second and third laboratories of ICOT Research Center in very tight cooperation with eight manufactures. Many fruitful discussions were done at the meetings of Working Groups 2, 3, 4, 5 and 6.

REFERENCES

Barwise, J., Perry, J., *Situations and Attitudes.*, MIT Press, 1983.

Bowen, K. A., Kowalski, R. A., *Amalgamating Language and Meta-language.*, In *Logic Programming.*, Clark, K. L., Taernlund, S. -A., (Eds.), Academic Press, pp. 153-172, 1982.

Clark, K., Gregory, S., *A Relational Language for Parallel Programming.*, In Proc. Conf. on Functional Programming Languages and Computer Architecture, ACM, 1981.

Clark, K., Gregory, S., *PARLOG: Parallel Programming in Logic.*, Research Report DOC84/4, Imperial College, 1984.

Doyle, J., *Truth Maintenance System.*, Artificial Intelligence, Vol. 12, No. 3, 1979.

Enomoto, H., Yonezaki, N., Saeki, M., Kunifuji, S., *Paradigms of Knowledge Based Software System and Its Service Image.*, ICOT TR-030, 1983.

Enomoto, H., Yonezaki, N., Saeki, M., Chiba, K., Takizuka, T., *Natural Language Based System Development System TELL.*, ICOT TR-067, 1984a.

Enomoto, H. Yonezaki, N. Saeki, M., *Formal Specification and Verification for Concurrent System by TELL.*, ICOT TR-068, 1984b.

Fuchi, K., *Logical Derivation of Prolog Interpreter.*, To appear in FGCS'84, 1984.

Fujita, H., Horiuchi, K., *Automated Verification System of Programs.*, ICOT TM-0049, 1984.

Furukawa K., Kondou, H., *Two Dimensional Programming in Prolog.*, Proc. of the LPC'83 (in Japanese), 1983a.

Furukawa, K., Nakajima, R., Yonezawa, A., *Modularization and Abstraction in Logic Programming.*, ICOT TR-022, 1983b. (Also in New Generation Computing, Vol. 1, No. 2, 1983)

Furukawa, K., Takeuchi, A., Kunifuji, S., *Mandala: A Knowledge Programming Language on Concurrent Prolog.*, ICOT TM-0028 (in Japanese), 1983c.

Furukawa, K., Takeuchi, A., Kunifuji, S., *Mandala: A Concurrent Prolog Based Knowledge Programming Language/System.*, ICOT TR-029, 1983d.

Furukawa, K., Takeuchi, A., Kunifuji, S., *Mandala: A Knowledge Programming System on a Logic Programming Language.*, ICOT TR-043 (in Japanese) 1984a.

Furukawa, K., Kunifuji, S., Takeuchi, A., Ueda, K., *The Conceptual Specification of the Kernel Language Version 1.*, ICOT TR-054, 1984b.

Furukawa, K., Takeuchi, A., Kunifuji, S., Yasukawa, H., Ohki, M., Ueda, K., *Mandala: A Logic Based Knowledge Programming System.*, 1984c. (Also to appear in FGCS'84)

Goodwin, J. W., *An Improved Algorithm for Non-monotonic Dependency Net Update.*, Linköping Institute of Technology, Technical Report, 1982.

Goto, S., *A Language for Building Expert Systems in CAD —CADLOG—.*, PIXEL, Vol. 24, pp. 134-138 (in Japanese), 1984. (Also in ICOT TM-0067)

Hikita, T., *Average Size of Turner's Translation to Combinator Program.*, ICOT TR-017, 1983.

Hirakawa, H., *Chart Parsing in Concurrent Prolog.*, ICOT TR-008, 1983a.

Hirakawa, H., Furukawa, K., Onai, R., *Implementing an OR-Parallel Optimising Prolog System(POPS) in Concurrent Prolog.*, ICOT TR-020, 1983b. (Also to appear in "Proceedings of RIMS Symposium on Software Science and Engineering", 1984, Springer-Verlag)

Hirakawa, H., Chikayama, T., Furukawa, K., *Eager and Lazy Enumerations in Concurrent Prolog.*, ICOT TM-0036, 1984a. (Also in Proc. of Second International Logic Programming Conference held at Uppsala)

Hirakawa, H., Furukawa, K., *Syntactic Parsing with POPS —Its Parsing Time Order and the Comparison with Other Systems—.*, ICOT TM-0073, 1984b.

ICOT Study Group on New Language Dissemination, *FGCS and Software.*, ICOT TM-015 (in Japanese), 1983.

ICOT WG5, *Several Aspect of Unification.*, ICOT TM-0046, 1984.

Ida, T., Kurokawa, T., Sakai, K., Sato, M., Toyama, Y., Hagiya, M., Hayashi, S., Hikita, T., Futatsugi, K., Matsuda, T., *Higher Order: Its Implications to Programming Languages and Computation Models.*, ICOT TM-0029, 1983.

Ida, T., Konagaya, A., *Comparison of Closure Reduction and Combinatory Reduction Schemes.*, ICOT TR-072, 1984.

Ishii, S., *Handling a Bit-Table in Prolog.*, ICOT TM-0013 (in Japanese), 1983a. (The abstract is also in "Proceedings of the 27th IPSJ National Conference" (in Japanese), 1983)

Ishii, S., *Study of Proofreading Techniques Used at a Japanese Newspaper.*, ICOT TR-039 (in Japanese), 1983b. (The abstract is also in "Proceedings of the 28th IPSJ National Conference", 1984)

Ishii, S., *Study of a Newspaper's Treatment of Proper Names.*, ICOT TM-0062 (in Japanese) 1984. (The abstract is also in "Proceedings of the 29th IPSJ National Conference", 1984)

Ishiwata, T., Tanaka, H., Miyoshi, H., Tanaka, Y., Amano, S., Uchida, H., Ogino, T., Yokoi, T., *The Basic Specification of Machine Readable Dictionaries.*, ICOT TM-0072 (in Japanese), 1984.

Joshi, A. K., Webber, B. L., Weischdel, R., *Living Up to Expectations: Computing Expert Responses.*, Proc. of AAAI'84, 1984.

Kanamori, T., Seki, H., *Verification of Prolog Programs Using an Extension of Execution.*, ICOT TR, 1984a.

Kanamori, T., Fujita, H., *Formulation of Induction Formulas in Verification of Prolog Programs.*, ICOT TR, 1984b.

Kanamori, T., Horiuchi, K., *Type Inference in Prolog and Its Applications.*, ICOT TR, 1984c.

Kitakami, H., Asou, M., Kunifuji, S., Miyachi, T., Furukawa, K., *A Method of Realizing a Knowledge Assimilation Mechanism.*, ICOT TR-010 (in Japanese), 1983a.

Kitakami, H., Miyachi, T., Kunifuji, S., Furukawa, K., *A Method of Realizing a Knowledge Acquisition System.*, ICOT TM-0017 (in Japanese), 1983b.

Kitakami, H., Kunifuji, S., Miyachi, T., Furukawa, K., *A Methodology for Implementation of a Knowledge Acquisition System.*, ICOT TM-0024, 1983c.

Kitakami, H., Hirakawa, H., Furukawa, K., *Knowledge Acquisition and DCG.*, ICOT TM-0032, 1983d.

Kitakami, H., Kunifuji, S., Miyachi, T., Furukawa, K., *A Methodology for Implementation of a Knowledge Acquisition System.*, ICOT TR-037, 1983e. (Also in "Proceedings of International Symposium on Logic Programming", Atlantic City, U.S.A., 1984, IEEE Computer Society Press)

Kitakami, H., Furukawa, K., *A Consideration on Debugging of Logic Program.*, ICOT TM-0033 (in Japanese), 1984a.

Kitakami, H., Kunifuji, S., Furukawa, K., Miyachi, T., *A Method of Implementing a Belief System in Prolog.*, ICOT TM-0052 (in Japanese), 1984b.

Kitakami, H., Miyachi, T., Furukawa, K., Kunifuji, S., *An Architecture of KAISER/KIM.*, ICOT TM-0068 (in Japanese), 1984c.

Kitakami, H., *A Knowledge Acquisition System which Supports a Knowledge Adaptation Process.*, ICOT TM-0064 (in Japanese), 1984d.

Kitakami, H., Kunifuji, S., Miyachi, T., Furukawa, K., *An Architecture of a Large-Scale Knowledge Base Management System.*, ICOT TM-0070 (in Japanese), 1984e.

Kock, A., *Synthetic Differential Geometry.*, London Math. Soc. Lecture Notes Series 51, Cambridge University Press, 1981.

Kondou, H., *Plan for Constructing Knowledge Architecture.*, Preprints of 36th WGAI Meeting of IPSJ (in Japanese), 1984.

Koseki, Y., *Prolog in CAD Applications — Case Study in PLA Design—.*, The 28th IPSJ National Conference, pp. 1479-1480 (in Japanese), 1984. (Also in ICOT TM-0037)

Kunifuji, S., Yokota, H., *Prolog and Relational Data Bases for Fifth Generation Computer Systems.*, ICOT TR-002, 1982. (Also in "Proceedings of CERT Workshop on Logical Bases for Databases", Toulouse, France, 1982)

Kunifuji, S., Takeuchi, A., Miyachi, T., Kitakami, H., Yasukawa, H., Furukawa, K., *Amalgamation of Object Knowledge and Meta Knowledge and Its Application.*, ICOT TR-009 (in Japanese), 1983a.

Kunifuji, S., Miyachi, T., Kitakami, H., Furukawa, K., *Knowledge Acquisition and Meta Inference.*, ICOT TM-0016 (in Japanese), 1983b.

Kunifuji, S., Takeuchi, A., Furukawa, K., Ueda, K., *Meta-inference and Its Applications —a Meta Predicate Simulate for Parallel Meta-inference—.*, ICOT TM-0039 (in Japanese), 1984a.

Kunifuji, S., Kitakami, H., Miyachi, T., Takeuchi, A., Yokota, H., Furukawa, K., Ueda, K., *Meta-inference and Its Applications Based on a Logic Programming Language.*, ICOT TR-049 (in Japanese), 1984b.

Kunifuji, S., Kitakami, H., Miyachi, T., Furukawa, K., *A Consideration on Knowledge Base Management Based on a Logic Programming Language Prolog.*, ICOT TM-0060 (in Japanese), 1984c.

Kunifuji, S., Kitakami, H., Miyachi, T., Furukawa, K., *Toward a Mechanization of Deductive, Inductive and Abductive Inference Functions.*, ICOT TM-0068 (in Japanese), 1984d.

Martins, J. P., *Reasoning in Multiple Belief Spaces.*, State University of New York at Buffalo, Technical Report No. 203, 1983.

Maruyama, F., Mano, T., Hayashi, K., Kakuda, T., Kawato, N., Uehara, T., *Prolog-based Expert System for Logic Design.*, ICOT TR-058, 1984. (Also to appear in FGCS'84)

Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H., Yasukawa, H., *BUP: A Bottom-Up Parser Embedded in Prolog., New Generation Computing.*, Vol. 1, No.2, OHMSHA-Spiringer-Verlag, 1983.

Mitsumoto, K., Mori, H., Fujita, T., Goto, S., *CADLOG: Prolog Interpreter for CAD Applications.*, Proc. of the LPC'84 (in Japanese), 1984a. (Also in ICOT TM-0038)

Mitsumoto, K., Mori, H., Fujita, T., Goto, S., *AI Approach to VLSI Routing Problem.*, IEEE Proc. of ISCAS, pp. 449-452, 1984b. (Also in ICOT TM-0045).

Miyachi, T., Kunifuji, S., Kitakami, H., Furukawa, K., Takeuchi, A., Yokota, H., *A Proposed Knowledge Assimilation Method for Logic Database.*, ICOT TM-0004 (in Japanese), 1983a.

Miyachi, T., Kunifuji, S., Kitakami, H., Takeuchi, A., Furukawa, K., *A Knowledge Assimilation Method for Logic Databases.*, ICOT TR-025, 1983b. (Also in "Proceedings of International Symposium on Logic Programming", Atlantic City, U.S.A., 1984, IEEE Computer Society Press)

Miyachi, T., Kunifuji, S., Kitakami, H., Furukawa, K., *A Consideration on Integrity Constraint Managing in Knowledge Acquisition Systems.*, ICOT TM-0040 (in Japanese), 1984a.

Miyachi, T., Kunifuji, S., Furukawa, K., Kitakami, H., *A Constraint Based Dynamic Semantic Model for Logic Databases.*, ICOT TM-0056, 1984b.

Miyachi, T., *The Conceptual Specification of Knowledge Conversation Module of KAISER.*, ICOT TM, 1984c.

Miyazaki, T., *Sequential Implementation of Concurrent Prolog Interpreter.*, ICOT TM-0071 (in Japanese), 1984.

Miyoshi, H., Mukai, K., Hirakawa, H., Yasukawa, H., Furukawa, K., *Generation of DCG Rules for Analysis of Japanese Bunsetsu.*, ICOT TM-0010 (in Japanese), 1983.

Miyoshi, H., *Object-Oriented Parsing in the Logic Programming Language ESP.*, ICOT TM-0053, 1984.

Mizoguchi, F., Furukawa, K. (eds.), *Proceedings of WG4 Workshop '83 on Knowledge Representation (in Japanese).*, ICOT TR-070, 1984.

Mori, H., Mitsumoto, K., Fujita, T., Goto, S., *LSI Wiring Design Expert System.*, The 28th IPSJ National Conference, pp. 1072-1073 (in Japanese), 1984a.

Mori, H., Mitsumoto, K., Fujita, T., Goto, S., *Knowledge-based VLSI Routing System — WIREX—.*, To appear in FGCS'84, 1984b.

Morishita, T., Hirakawa, H., *GDL0: Grammar Description Language based on DCG.*, ICOT TR, 1984.

Mukai, K., *A Unification Algorithm of Infinite Trees.*, ICOT TM-009 (in Japanese), 1983a. (Also in Proc. of IJCAI '83)

Mukai, K., *Prolog Interpreter Accepting Infinite Trees Implemented in DEC-10 Prolog.*, ICOT TM-011 (in Japanese), 1983b.

Mukai, K., Furukawa, K., *An Ordered Linear Resolution Theorem Proving Program in Prolog.*, ICOT TM-0027, 1983c.

Mukai, K., *An Experiment of Agent-based on Modal Propositional Logic*, ICOT TM-0030 (in Japanese), 1983d.

Mukai, K., Yasukawa, H., Miyoshi, H., Hirakawa, H., *Toward a Computational Model of Situation Semantics and Its Implementation in Prolog.*, ICOT TM-0051 (in Japanese), 1984.

Nagai, Y., Chigira, H., Kobayashi, M., *Problems in Developing an Experimental System Able to Reuse Existing Programs.*, ICOT TM-0059, 1984.

Nakashima, H., Ueda, K., Tomura, S., *What is a Variable in Prolog?*, To appear in FGCS'84, 1984.

Sakai, K., Miyachi T., *Incorporating Naive Negation into Prolog.*, ICOT TR-028, 1983.

Sakai, K., *An Ordering Method for Term Rewriting Systems.*, ICOT TR-062, 1984.

Sato, M., *Theory of Symbolic Expressions, I.*, Theoretical Computer Science 22, 1983a.

Sato, M., Sakurai, T., *Qute: A Prolog/Lisp type Language for Logic Programming.*, ICOT TR-016, 1983b.

Sato, M., Sakurai, T., *Qute: A Functional Language Based on Unification.*, To appear in FGCS'84, 1984a.

Sato, M., *Theory of Symbolic Expressions, II.*, to appear, 1984b.

Sato, M., *Theory of Symbolic Expressions, III.*, in preparation, 1984c.

Shapiro, E. Y., *Algorithmic Program Debugging.*, The MIT press, 1982a.

Shapiro, E. Y., *A Subset of Concurrent Prolog and Its Interpreter.*, ICOT TR-003, 1982b, revised in 1983.

Shapiro, E. Y., Takeuchi, A., *Object Oriented Programming in Concurrent Prolog.*, ICOT TR-004, 1983a. (Also in New Generation Computing, Springer-Verlag Vol. 1 No. 1, 1983)

Shapiro, E. Y., *Systems Programming in Concurrent Prolog.*, ICOT TR-034, 1983b. (Also in "Proceedings of 11th ACM POPL Symposium", 1984)

Shapiro, E. Y., *Lecture Notes on the Bagel: A Systolic Concurrent Prolog Machine.*, ICOT TM-0031, 1983c.

Staunstrup, J. (ed), *Program Specification — Proceedings of a Workshop , Aarhus, Denmark, August 1981.*, Lecture Notes for Computer Science Vol. 134, Springer-Verlag, 1981.

Sugimoto, M., *Software Development Support System.*, ICOT TR-051, 1984a.

Sugimoto, M., Kato, H. Yoshida, H., *Design Concept for a Software Development Consultation System.*, ICOT TR-071, 1984b.

Sugiyama, K., Kameda, M., Akiyama, K., Makinouchi, A., *A Knowledge Representation System in Prolog.*, ICOT TR-024, 1983.

Suzuki, H., *MAID: a MAn-machine Interface for Domestic affairs.*, ICOT TM-0058, 1984.

Takagi, S., Chikayama, T., Hattori, T., Tsuji, J., Yokoi, T., Uchida, S., Kurokawa, T., Sakai, K., *Overall Design of SIMPOS.*, Proceedings of Second International Logic Programming Conference, 1984. (Also in ICOT TR-057)

Takeuchi, A., *Let's Talk Concurrent Prolog.*, ICOT TM-0003 (in Japanese), 1982.

Takeuchi, A., Furukawa, K., *Interprocess Communication in Concurrent Prolog.*, ICOT TR-006, 1983. (Also in "Proceedings of Logic Programming Workshop '83", Portugal)

Takeuchi, A., *Logic Based Parallel Programming Language Concurrent Prolog.*, Computer Software, Vol. 1, No. 2, pp. 25-37 (in Japanese), 1984. (Also in ICOT TM-0043)

Tamaki, H., *A Transformation System for Logic Programs which Preserves Equivalence.*, ICOT TR-018, 1983.

Tamaki,H., Sato, T., *Unfold/Fold Transformation of Logic Programs.*, Proc. 2nd International Logic Programming Conference, 1984a.

Tamaki, H., *Semantics of a Logic Programming Language with Reducibility Predicate.*, in Proc. 1984 International Symposium on Logic Programming, pp.259-264, IEEE, 1984b.

Tanaka, H., Takakura, S., Konno, M., *Development of English Grammar on BUP System and Its Performance.*, Proceedings of the Logic Programming Conference '84 (in Japanese), 12-2, 1984a.

Tanaka, H., Tsujii, J., Yokoyama, S., Yasukawa, H., Suzuki, K., Isahara, H., Strauss, M., *An Approach to High Quality Natural Language Translation.*, ICOT TR-073 (in Japanese), 1984b.

Ueda, K., Takeuchi, A., Kunifuji, S., Furukawa, K., *String Manipulation in Concurrent Prolog.*, ICOT TR-036 (in Japanese), 1983.

Ueda, K., Chikayama, T., *Efficient Stream/Array Processing in Logic Programming Languages.*, ICOT TR-065, 1984. (Also to appear in FGCS'84)

Yasukawa, H., *LFG in Prolog —Toward a Formal System for Representing Grammatical Relations—*, ICOT TR-019, 1983.

Yasukawa, H., *LFG System in Prolog.*, *Proc. of COLING'84.*, 1984.

Yasuura, H., *On the Parallel Computational Complexity of Unification.*, ICOT TR-027, 1983.

Yokoi, T., Mukai, K., Hirakawa, H., Yasukawa, H., Miyoshi, H., *Advanced High-Performance Parsing System.*, ICOT TM-005 (in Japanese), 1982a.

Yokoi, T., Mukai, K., Hirakawa, H., Yasukawa,

H., Miyoshi, H., *Bottom-up DCG Parser Manual.*, ICOT TM-006 (in Japanese), 1982b.

Yokomori, T., *A Note on the Set Abstraction in Logic Programming Language.*, ICOT TR-060, 1984. (Also to appear in FGCS '84)

Yokota, H., Kunifuji, S., Kakuta, T., Miyazaki, N., Shibayama, S., Murakami, K., *An Enhanced Inference Mechanism for Generating Relational Algebra Queries.*, ICOT TR-026, 1983a. (Also to appear in "Proceedings of Third ACM SIGACT-SIGMOD Symp. on Principles of Database Systems", Waterloo, Canada, 1984)

Yokota, H., Kunifuji, S., Shibayama, S., Miyazaki, N., Kakuta, T., Murakami, K., *How Can We Combine a Relational Database and a Prolog-based Inference Mechanism?*, ICOT TR-031 (in Japanese), 1983b.