ICOT Technical Report: TR-300

TR-300

Current R & D Results and Their Perspective in Japanese Fifth Generation Computer Systems (FGCS)

by
T. Ichiko and T. Kurozumi

September, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191 - 5
Telex ICOT J32964

Institute for New Generation Computer Technology

Current R&D Results and Their Perspective in Japanese Fifth Generation Computer Systems(FGCS)

Takao ICHIKO

Takashi KUROZUMI

Institute for New Generation Computer Technology (ICOT)

Abstract The Fifth Generation Computer System (FGCS) is being developed to satisfy the demands of advanced information processing in the 1990s by eliminating various limitations of conventional computer technology. The new generation computer will be based on novel concepts and innovative theories and technologies to perform knowledge information processing effectively. The system will have three major functions: a problem-solving and inference function, a knowledge base management function, and an intelligent interface function. ICOT aims to develop software and hardware systems for each of these functions using logic programming and VLSI technologies. Basic software and hardware development environments have been built up during the last five years. An experimental parallel inference machine with an effective software paradigm will be developed in the near future, and a totally integrated demonstration system will be implemented in the final stage. Thus, its performance will be upgraded to several hundred times more than the value of conventional inference mechanisms, thereby realizing a feasible environment for many pragmatic uses in knowledge information processing.

Keywords : Knowledge Information Processing, Kernel Language, Knowledge Base, Parallel Inference Machine, Multi-PSI

§1 Introduction

Ten years were allotted for effective R&D of the FGCS project. The project was then divided into three stages to make the best use of research results: a three-year initial stage for basic technology development; a four-year intermediate stage for subsystem development; and a three-year final stage for total system development (Figure 1).

The intermediate stage plan (Ref. 1) based on the results from the three-year initial stage continues to focus on the following four core research areas, with substantial results :

1) Hardware system
   (1) Inference subsystem
   (2) Knowledge base subsystem
2) Basic software system
3) Development support system

1) Research and development of the hardware system in the intermediate stage will be based on the results of the initial stage R&D activities on the following items: the functional mechanism for the inference subsystem, the relational database mechanism and other mechanisms for the knowledge base machine, and the sequential inference machine. The specifications of the parallel-type Kernel Language version 1 (KL1) designed in the previous stage will be used as a guideline for research on the hardware system.

The intermediate stage work on the hardware system aims to establish a machine architecture for the inference subsystem with advanced parallel processing capabilities and a parallel machine architecture for the knowledge base subsystem. Emphasis will be placed on a study of general-purpose machine architectures suitable for inference and knowledge base processing to form the core of the software system for knowledge information processing.

(1) The inference subsystem is a machine for parallel logic programming languages and can efficiently execute KL1. Research and development of the inference subsystem in the intermediate stage aims to establish an architecture for a parallel inference machine for large-scale parallel processing consisting of about 100 processing elements. A prototype hardware for the machine to check its operation will also be developed. The processing element will perform detailed parallel processing within the unit. Dataflow, reduction, and other mechanisms studied in the initial stage will be evaluated and reorganized according to the specifications of KL1 to provide the foundations for the intermediate stage R&D (Ref. 2).

(2) Research and development of the knowledge base subsystem in the intermediate stage aims to define technology to implement a knowledge operation mechanism required by the knowledge base machine, establish the parallel architecture for the knowledge base machine, and develop prototype hardware. Mechanisms to support a distributed knowledge base will also be studied, and the technology required to implement the mechanisms will be clarified.

2) Research and development of the basic software system will be based on the results of the basic technology developed in the initial stage. This R&D aims to provide a second version of the kernel language (KL2) by enhancing KL1 through new software technologies (e.g., parallel inference control based on parallel processing), the evaluation of the component technologies developed in the initial stage, and the development of more practical prototype systems.

Fundamental studies will be conducted on meta-level inference, distributed knowledge base management, and other functions to lay the groundwork for the cooperative problem-solving system, the final target of basic software system R&D.

A major research goal for the intermediate stage is establishing the technology to implement parallel-processing-based software modules for controlling and managing the hardware system of the inference and knowledge base subsystems.

Other points to be stressed are the clarification of the implementation technology for knowledge base management functions involving many unknown factors (e.g., a knowledge acquisition function), and the establishment of techniques to implement practical intelligent interface and programing functions in logic programming languages.

Research and development of the basic software in the intermediate stage will be conducted in the following five areas:

i. Fifth Generation (5G) Kernel Language
ii. Problem-solving and inference software module
iii. Knowledge base management software module
iv. Intelligent interface software module
v. Intelligent programming software module

i. Two Kernel Language versions, KL1 and KL2, will be developed. Work on KL1 aims to improve its language specifications, speed up its language processor, implement the language, and complete its programming system in a parallel execution environment. The specifications of KL1 will be established to define detailed specifications of the parallel inference machine (PIM).

Preliminary specifications will be developed for KL2 with emphasis on the knowledge representation function. This version of KL2 and other knowledge programming languages will be used throughout intermediate-stage R&D. The experience obtained using these languages will be used to remove bottlenecks in KL2 and identify functions to be supported by hardware for making KL2 faster, mainly its knowledge representation function. The final specifications of KL2 will be determined based on this research.

ii. The target of R&D in this area is to develop the cooperative problem-solving system. Work in the intermediate stage mainly aims to clarify the basic technology for developing this system, i.e., techniques to implement problem solving in a parallel environment. Different levels of parallelism, ranging from the system programming level to application systems, will be investigated.

Parallelism will be studied in a wide variety of application fields. This work will be required to develop techniques for modeling systems capable of achieving an average parallelism

factor of about 100 and methods of hierarchically configuring the systems.

In addition, inductive inference, analogy, and other sophisticated inference functions necessary for implementing systems will be developed.

These basic technologies will be combined to develop prototypes of the cooperative problem-solving system.

Demonstration software will be prototyped for the problem-solving and inference software module. It will be used to apply the results in this area to real-world problems for demonstration and feed the evaluation results back to each software module.

iii. R&D in this area aims to establish the basis for knowledge information processing. The activities planned in the intermediate stage focus on (a) the development of a distributed knowledge base control system, an extension of the large-scale relational database management system; (b) the establishment of distributed knowledge base utilization methods to allow the distributed knowledge base control system to solve problems in parallel processing; and (c) the construction of systems to support the more sophisticated functions of knowledge acquisition. A knowledge programming language and its language processor (e.g., knowledge base editor) will provide the common base. In addition, an experimental system for distributed knowledge base utilization will be prototyped to integrate the results of these activities. This system will be able to manipulate knowledge obtained from dozens of experts.

iv. Semantic analysis will be the primary focus of natural language processing research, while a preliminary study will be conducted on context analysis. Linguistic data will be analyzed and organized. Text understanding systems are also planned capable of understanding, for example, junior-high-school science textbooks with a vocabulary of about 3000 words.

Research and development will be carried out on the construction of interactive models which permit mutual understanding between man and machine.

v. This theme aims to pursue various software engineering problems in logic programming language. Research and development will focus on the design of specification description languages to merge logic-based formal approaches with more natural approaches based on natural languages and graphics.

The major subject of software engineering is how to speed up, with computers, not the coding stage itself but the stages before and after coding; that is, design, maintenance, refinement, and debugging.

Rapid prototyping is considered the key factor for design support; a support system will be developed for this technique. Work will be conducted on (a) a technique to implement programs as parts; (b) a software base to achieve this technique; and (c) expert systems for design support by choosing appropriate application areas.

3) Research subjects include the language processor for KL1, prototyping of software for the KL1-based parallel inference mechanism and the parallel knowledge base management mechanism. A pilot model for a parallel software development machine will be developed to support these activities. This development machine will be a small-scale multiprocessor system tightly coupling several processors. The CPU of the sequential inference machine developed in the initial stage will be used as the processor. It should achieve process-by-process parallel processing at an early stage and serve as a tool for developing parallel-processing-based prototype software and evaluating the effectiveness of parallel processing. The machine is planned to be available as a tool from the middle of the intermediate stage. The software and hardware of both the sequential inference machine and the relational data base machine developed in the initial stage will be refined and enhanced so that these machines can be used as tools. The local area network (LAN) will also be expanded and refined to facilitate software development. In addition, the wide area network (WAN) will be arranged so that research groups can share data, software, and devices for more efficient research and development.

§2 Inference Subsystem

(1) Parallel inference machine architecture

An architecture for the parallel inference machine was established (Ref. 2) as follows:

i. Overall structure

For communications between parallel processes distributed among multiple PEs, models appropriate for parallel software should have no difference or only a small difference in "distance" expressed by logical communication cost. However, some difference in distance will inevitably occur between the intermediate-stage PIM hardware, consisting of about 100PEs, and the final-stage PIM, consisting of about 1000 PEs. Therefore, the entire structure of hardware for the parallel inference machine was designed hierarchically, taking into consideration the distance between PEs in hardware as shown in Figure 2. Parallel processing in this structure is performed at three levels: within a PE, within a cluster, and between clusters.

ii. Processing element (PE)

The processing element (PE) used in the intermediate-stage PIM has a KL1-oriented instruction set whose data width is around 40 bits. The PE has a parallel cache memory and a dedicated lock mechanism to permit a faster access to global memory (GM) and perform exclusive control on access. The PE must be compact, have low power consumption, and operate fast. To achieve this goal, a single PE will be installed on a PC board, and the instruction execution unit of each PE will be compact so that it can be implemented by a single LSI chip.

iii. Parallel cache memory mechanism

Data must always be consistent for all parallel cache memories. Therefore, there must be a way of informing other PEs of an address written to a cache by its PE. One key to cache memory design is to make each PE use the global bus

(GB) less frequently, because the maximum number of PEs that can be connected to GM through the GB increases as each PE uses the GB less frequently. Therefore, the PIM has a parallel cache mechanism in which the cache protocol is optimized by the parallel execution of KL1.

iv. Cluster controller and network

A cluster controller provides a single point of control over references to data coming into or out of each cluster in cluster-to-cluster message communications. Any communications request issued from a PE in a cluster to a PE in another cluster is handled exclusively by the cluster controller. The cluster controller in a cluster has a table to manage pointers between clusters; while referring to or updating the table, the controller generates a message to be sent to another cluster and sends it via the network. The controller in the destination cluster receives the message and performs the processing specified by the message or delivers the message to the processor.

(2) Detailed study of processing methods of KL1

KL1 is based on the parallel logic language, GHC, and appears to the user as a three-layered structure as shown in Figure 3.

KL1-u is a user language and can be regarded as a higher level language of GHC. A concept for incorporation in KL1-u to help programmers to write parallel programs is under investigation. KL1-c provides the framework for KL1. It is a restricted version of GHC allowing only built-in predicates in the guard part. KL1-b is a machine language for parallel inference machines. Abstract machine-language specifications (machine-independent specifications of machine languages) were investigated. KL1-p is information to be added to programs to run them efficiently on parallel machines. The specifications of KL1-b (abstract machine language), the basic machine language for parallel inference machines, were established by assuming the abstract machine shown below. In the assumed abstract machine,each processor will have an arithmetic unit and registers capable of handling tagged data. The memory will be accessible from all processors. Goal arguments to be reduced will be stored in the registers before unification. That is, instructions similar to those in the Warren Abstract Machine (WAM) will be issued for unification in the passive part.

(3) Functional specifications of processing methods of parallel inference machines

These features correspond to the functional specifications of the abstract machine-language instructions described in the previous section.

Parallel processing in parallel inference machines is conducted in units of goals. Goals are distributed among PEs in a cluster and among clusters. Within a cluster, communication via the GM are used to perform goal tree management, scheduling, synchronization control, and load distribution. Functional specifications for these operations were studied to make use of locality in processing. For inter-cluster communications, functional design was produced for, for example, goal tree management based on the message communications method, and part of the design was subjected to a detailed study.

(4) Functional specifications of hardware for parallel inference machines

The functional specifications of the intermediate-stage PIM hardware were examined in detail.

The PE for the intermediate-stage PIM mustbe designed for compactness, low power consumption, and fast operation. To achieve this goal, a single PE will be installed on a single PC board and the instruction execution unit of each PE will be made compact so that it can be implemented by a single LSI chip.

It was decided to introduce the idea of both reduced instruction set computers (RISC) and complicated instruction set computers (CISC) in the processor design to decrease the amount of hardware logic and to adopt low-power CMOS VLSI chips.

The performance of the PE must be sufficiently enhanced to increase the impact of parallel inference machines on software development. The targetted performance of a single PE is more than 250 KLIPS on average and more than 500 KLIPS for simple bench-mark programs like APPEND. The single PE can be made faster in two ways: (a) use of the pipeline scheme, which divides an instruction into several cycles to reduce the clock cycle of the machine, and (b) addition of a hardware mechanism to support the execution of KL1 programs. This hardware mechanism uses tag architecture and KL1-specific instructions, and enables conditional interruption.

Faster access to the global memory (GM) and faster inter-PE communications and synchronization are crucial for intra-cluster parallel processing. This led to the design of the PIM cluster with the emphasis on the parallel cache and lock mechanisms of the GM. The parallel cache was introduced to make access to the GM faster. It was optimized to fit the parallel execution characteristics of KL1. Lock operation, particularly one-word lock, frequently appears in KL1 processing. For faster execution of one-word lock, it was decided to perform distributed control by adding a hardware lock mechanism to each PE.

Address-by-address interleaving and the reduction of the bus cycle time were investigated for the global bus (GB) to prevent the throughput of the GB from degrading the performance of the parallel inference machine. Assuming that each word of the GM consists of a 8-bit tag and a 32-bit data section, it was decided to establish an address transformation mechanism based on 32-bit word addressing and a reference recording mechanism to support virtual storage for address transformation. An interprocessor interrupt bus will be built among PEs separately from the GB to synchronize communications among PEs.

(5) Control software

The prototype control software described below was developed to investigate KL1 processing methods and architectures for the parallel inference machine.

The development of the prototype aims to:

- Verify processing algorithms of KL1 processing methods for the parallel inference machine
- Establish a basis for comparing processing methods
- Investigate the behavior of language processors
- Collect data to design pilot machines

The prototype control software developed consists of a compiler, assembler, linker, and experimental simulator for parallel processing with a cluster.

The compiler reads a program written in KL1 and generates an abstract machine language program. The abstract machine language program is converted by the assembler into a relocatable machine language program, which is then converted by the linker into an executable machine language program. The compiler, assembler, and linker are written in Prolog and C.

The experimental simulator is a pseudo-parallel processing system to emulate a cluster in the intermediate-stage PIM. It was developed with the SIMASM, a multi-process simulator support tool developed. The system simulates 4 to 16 PEs. It can operate as a parallel processing system and provide various statistical data and memory access information for hardware design. When not providing statistical and memory access information, the simulator can execute the 8-queen program in 27 seconds for four PEs.

The SIMASM was equipped with a utility to describe queue simulation and other tasks.

(6) Parallel inference machine model and component module

R&D of the parallel inference machine component modules in the intermediate stage aims to obtain information on parallel execution methods and mechanisms and load-distributing mechanisms in parallel inference machines. So far, functional design was produced for the data-flow, reduction, sequential, and goal-rewriting modules. The experimental machine for the parallel inference machine model prototyped and enhanced was used in evaluation experiments to investigate the parallel inference machine component modules in detail.

§3 Knowledge Base Subsystem

(1) Special-purpose unit for knowledge base operation

i Development of a conceptual model of knowledge base machine pilot system

The conditions required for the knowledge base machine and methods of implementing the pilot system were determined, and the requirements for the knowledge base operation engine were defined.

1) The prototype of the relational data base machine Delta in the first stage showed the similarity between Prolog and relational data bases. The research in the preceding years confirmed that the parallel relational data base engine (RE) implemented in LSI permits rapid execution of ordering and relational algebra operations, such as join and restriction, while data flows in stream form. (Ref. 3).

2) Investigation disclosed that the knowledge base machine must be able to :

(a) Efficiently process structured variable-length data ;
(b) Perform secondary-storage-to-main-storage on-the-fly processing ;
(c) Efficiently process simple unification which handles a large number of facts ;
(d) Perform computations for a set of knowledge.

3) Figure 4 shows the conceptual configuration of the knowledge base machine pilot system with the necessary capabilities described above.

(a) Standalone knowledge base machine ;
(b) Prolog has built-in KBM predicates for access to knowledge bases ;
(c) Supports the knowledge base management system (KBMS) ;
(d) Uses the knowledge base operation engine to perform rapid ordering and relational algebra operations on variable-length data ;
(e) Adopts the on-the-fly method;
(f) Application-specific user interface and deductive processing.

ii. Development of prototype knowledge base operation engine

Figure 5 is an outline of the hardware structure of the knowledge base operation engine (KE), a component of the knowledge base machine pilot system. The hardware of the pilot system is built around a general-purpose computer and has a KE. The KE receives an operation instruction from the CPU and reads the data necessary for the operation from the secondary storage linked to the KE. The operation, such as ordering or a relational algebra operation, is performed by the engine core in the KE, and the result is sent to main storage via the channel.

Research into the knowledge base machine pilot system will be carried out using a more advanced model than in the initial stage. The initial-stage model loosely coupling the inference engine and data base will be replaced with a model where the inference function and knowledge base is tightly coupled over a shorter distance. This will allow the pilot system to handle a wider variety of more complicated knowledge data in a general representation format.

R&D activities consisted of the examination of the work-evaluation of the initial-stage engine and establishment of the functions of the knowledge base machine pilot model - and the development of the conceptual model. In addition, the knowledge base engine was implemented in hardware. The hardware engine was directly connected to the internal bus of the host computer to form a tightly coupled model. The model was then used to develop a prototype for a basic knowledge base operation engine which will provide the basis for research on smaller, faster systems built using advanced component technology in the future.

The knowledge base machine pilot system was studied as a basic experimental system for the knowledge base machine parallel control mechanism to be integrated into the hardware system of the FGCS. The results described above will be incorporated into research on the knowledge base machine parallel control mechanisms.

(2) Control mechanism for parallel access and processing

Research on this subject aims to develop a knowledge base machine which will perform a rapid search on a large amount of knowledge stored in secondary storage and be used by multiple users via host machines.

The results are outlined below.

i. Theoretical study of relational knowledge base model and retrieval-by- unification operation

The relational knowledge base and retrieval-by-unification (RBU) operations were theoretically defined, followed by a study of the characteristics of the RBU operations.

A relational knowledge base model was formulated, as a candidate for a common knowledge base model capable of responding to various users and applications. The model is a conceptual model for the knowledge base and consists of simple structures and operations capable of dealing with various knowledge representations presented by users.

A term, a certain type of structure including variables, was used as a basic component to represent knowledge, and a relation was used as a basic container to hold terms. The RBU operation, an integration of relational algebra and unification, was formulated for term relations.

If unification-join, an RBU operation, is performed simply, the machine must perform huge amounts of processing, because it must check all the term relations to see whether the specified tuples in one attribute can be unified with any of the possible combinations of the other term relations. To make unification-join more efficient, a method to give an ordering to terms was defined, and the number of combinations reduced by ordering terms with the characteristics of the method.

ii. Design of unification engine

The unification engine is special-purpose hardware to perform unification-join and other operations on term relation data stored in the multi-port page memory (MPPM) in the knowledge base machine.

Major R&D activities included a detailed study of the hardware structure proposed, incorporation of new processing schemes, development of detailed software to simulate the hardware, and evaluation of the engine with the hardware.

The unification engine consists of about 10 modules. A detailed study was conducted on the functions and processing schemes for each of the modules proposed. The unification engine is a dedicated unit for unification in the knowledge base machine. Software simulation was performed to evaluate the performance and features of the unification engine, by measuring two factors concerning data transfer within a sort cell and among sort cells in the engine; the amount of data transferred and transfer time. The simulation was conducted prior to implementing the unification engine in hardware. The major components of the unification engine are the sorter, pair generator, and unification unit. These are structured to permit pipeline processing. The simulation results showed that the amount of processing in the unification unit almost entirely agreed with that of the entire unification engine for uniformly distributed comprehensive data. The processing in the unification unit depends on the amount of output from the pair generator, while the output amount of the pair generator is determined by the generator's function to eliminate term pairs on which unification must fail. Because this function works less effectively for comprehensive data, the amount of processing required to order terms in the sorter and generate pairs is smaller than that of unification processing. The simulation revealed that the performance of the unification engine depends on the elimination ratio of the pair generator and the unification processing capability of the unification unit. Additional investigation of the operation of the unification unit and pair generator led to the discovery of various ways to reduce the processing time. The knowledge base machine system (Figure 6) is a computer system devised to process relational knowledge. The core components of the system are the unification engine and MPPM described in the previous section. Software simulation was conducted and a detailed design for the system configuration and control method proposed was produced. These activities were helpful in understanding the behavior of the system when running specific problems.

Further investigation of the system configuration and control method revealed that the performance of the architecture depends on the type of problem. The relation between the architecture and the characteristics of specific problems was partly clarified. The knowledge base machine is a back-end knowledge base machine linked to its host, the inference machine. It provides a knowledge processing function simultaneously for multiple inference machines. It has basic functions to store relational knowledge representations and perform operations on them. The operations include unification-restriction and unification-join in addition to the conventional relational algebraic operations. When the host machine sends a request to perform processing on relational knowledge to the knowledge base machine, the control processor analyzes the request and translates it into operations on relational knowledge. The knowledge base machine can divide an operation or group several operations page by page. The machine simultaneously processes a set of problems on multiple unification engines. To investigate control schemes and other problems of the parallel processing on the knowledge base machine, software simulation was performed by developing a model for the knowledge base machine. The simulator was planned so that the number of unification engines, page size of the MPPM, method to divide a problem, method to assign sub-problems to unification engines, and other factors could be specified by parameters. The simulation showed that the page size does not much influence the processing time, that the effective utilization ratio of the page decreases as the number of unification engines increases, and that the scheme for restoring data in pages planned for other systems is not necessary, because the effective urilization ratio exceeds 50% for up to 1K byte.

(3) Distributed knowledge base control mechanisms

Research into this subject aims to develop technologies to integrate multiple knowledge base machines, interconnected

via high-speed buses, local area networks, and other links,to form a distributed knowledge base system which will appear to the user as a logically single knowledge base machine. The development of the distributed knowledge base system, Predicate Logic Based Hierarchical Knowledge Management System (PHI), began as part of the intermediate-stage R&D to provide a means to develop and verify the various control and management mechanisms. Figure 7 shows the configuration of the PHI system. Host machines, inference machine PSIs, are linked to knowledge base machine PHIs via the ICOT-LAN. The knowledge base system in the PHI is hierarchically structured; the knowledge bases are managed by dividing them into an intensional data base (IDB) section and an extensional data base (EDB) section. This approach allows the construction of a system which provides the features of both the knowledge representation function of logic language processors and the management function of relational data base systems.

The relational data base management system (RDBMS) forms the nucleus of the knowledge base management unit, supporting the functions of the relational data base machine, Delta, developed in the initial stage. The knowledge management system in the knowledge base management unit stores and searches for knowledge via the RDBMS. Prototype software capable of standalone operation was developed as the first step of the RDBMS development. The RDBMS was installed on one of the PSIs interconnected via a LAN. The knowledge base engine (KBE), a hardware option for the PHI, is an operation unit which performs comparison-based processing such as search very quickly. Figure 8 shows the structure of the KBE. The KBE receives an operation command from a PHI, executes it, and returns the results of the operation to the PHI. The KBE repeats this sequence of processing. The KBE uses the superimposed code method, an enhanced superimposed-code-based search technique, to rapidly perform not only search but also relational operations used in knowledge processing.

(4) Knowledge representation processing methods for knowledge base machines

A knowledge base machine must have a mechanism which allows the users and manager of a knowledge base to store, manage, and use knowledge easily.

Here, the users of the knowledge base are assumed to use logic-programming-based knowledge representation suitable for various problem domains. It is also assumed that the knowledge base is shared among more than one user.

If the knowledge base has information on knowledge representation to be used by users, the knowledge base manager can understand how the users obtain access to the knowledge base, and therefore can take measures against dangerous access which may spoil the uniformity and consistency of the knowledge base.

Assuming that the interface functions for individual knowledge representation have some part in common, a system was devised as a means to implement these interface functions. It will compile knowledge representations into intermediate code in some form and store the code in the knowledge base machine to efficiently process requests to the knowledge base, such as retrieve and updating, as operations on the intermediate code.

Research into knowledge representation processing methods for knowledge base machines investigated the problems involved when such interface functions of the knowledge base system were implemented as a knowledge compiler.

The effect of the implementation was also studied.

i. Knowledge compiler

1) Knowledge compilation control model

Figure 9 shows the execution process of knowledge compilation as the total image of the knowledge compilation control model.

2) Investigation was performed on program transformation, partial computation, input to partial computation, and partial computation in Prolog. The results led future focus on the knowledge compiler and the interpreter for intermediate representation as follows:

(a) Knowledge compiler

A study will be conducted of methods to compile the following into Prolog programs:(a) knowledge representations shown by object knowledge, and (b) meta-knowledge and a meta-interpreter to execute and interprete the knowledge representations.

(b) Intermediate-representation interpreter

Prolog will be used to describe the intermediate representation, and Prolog-based representation methods will be studied. An investigation will also be conducted on RBU for clause search and an interface between Prolog and RBU.

ii. Study of knowledge representation

In the knowledge representation for planning, the initial state, operators to change the initial state, and the final expected condition are represented, and the represented results are used to infer a series of operators beginning with the initial state and ending with the expected condition. Typical applications of this knowledge representation include design-related tasks, such as logic design and layout, and plan-related work, such as planning of an operation sequence of robots and process control. Production systems are often used for the knowledge representation for planning. The existing production systems, including production rule representation, were surveyed and production systems suitable to the knowledge representation for planning were proposed. Future ways to research the knowledge compiler, production systems, modality-symbol-based control, and other problems were also investigated.

iii. Study of intermediate representation

Knowledge representation explicitly and declaratively expresses human thought. To process knowledge representation efficiently on computers, the idea of introducing intermediate representation with the emphasis on fast processing on computers and using the knowledge compiler to convert knowledge representation into intermediate representation was devised, as described above. the intermediate representation was examined in terms of the functions required and the architecture of computers.

Knowledge processing differs considerably from

conventional information processing in that it mainly consists of pattern matching between pieces of knowledge expressed in symbols with some structure. Unification of Prolog is a powerful pattern matching function.

The knowledge base system is also characterized by the large knowledge bases involved. To manage large amounts of data efficiently, the system uses the schema model of the relational data base, which is based on handling several data items as a set.

The architecturally preferred approach is to store a large-scale knowledge base in a disk unit and pass only the necessary data from the disk unit to main memory. Few successful examples, however, have been reported, even for data base machines in which a filter was introduced to a disk controller or disk unit to extract necessary data. The functions of such filters have yet to be defined for knowledge bases. Unification appears to be an effective filter function, providing an approach to a new architecture.

The knowledge base processing described above can be conceptually illustrated as in Figure 10.

Processing in the knowledge representation system appears to consist of a repetition of pattern matching between temporary states in the working memory and the knowledge base as shown in the figure. That is, a repetition of retrieval can be considered as inference processing. Some inference processing mainly changes the state. Retrieval-based processing mechanisms do not always increase efficiency. The filter mechanism described above seems effective for knowledge base processing focusing on shallow and wide inference.

§4 Basic Software System

(1) Fifth Generation Kernel Language

i. Detailed study of GHC specifications

GHC (Guarded Horn Clauses) (Ref. 4,8) is a parallel logic programming language developed. The grammar rules of the language were refined and enhanced to provide the foundation for KL1 in the basic software system. The semantics of GHC were defined from the following two aspects: (1) parallel input resolution neglecting the difference between guards and bodies, and (2) restriction on the parallel input resolution to execution in a single environment.

ii. Development and evaluation of a prototype KL1 language system

1) Implementation on a general-purpose machine

A basic study was conducted on a GHC-based parallel programming environment to give it wider acceptance*. The results of the study were used to develop a GHC system on a general-purpose machine which runs programs efficiently. The system is based on FGHC (flat GHC: a subset of GHC which balances the existing compiling technology with the descriptive power of the language). It executes a sequence of guard goals sequentially and a sequence of body goals on a pseudo-parallel basis.

*GHC is also regarded as the effective way to merge message passing - a standard parallel - programming construct with the declarative style, which makes more portable and easier to debug, and deductive capabilities of logic programming.

2) Sequential implementation

KL1 consists of four components is hierarchically structured as shown in Figure 3 : KL1-c, KL1-p, KL1-u, and KL1-b. Two software modules had already been developed for the sequential processing system : a compiler which converts KL1-u and KL1-c into KL1-b, and an abstract machine emulator which directly executes KL1-b. The emulator is implemented in ESP on the PSI. The system was further enhanced by the addition of the following debugging support tools: a style checker which uses the grammar and characteristics of the language to perform static analysis as debugging prior to program execution, a tracer with which to debug a program while running it, and an algorithmic debugger with which to debug a program based on its specifications after execution. These additional functions proved effective in parallel programs where debugging is far more difficult than in conventional sequential programs.

3) Pragma evaluation system

KL1-c forms the core of KL1. KL1-p is the pragma language which indicates goal distribution. These two components work together to extract the distribution processing capability inherent in a parallel machine. A pragma evaluation system was developed as an experimental environment (support tool) to advance the tentative specifications of the pragma to practical specifications applicable to real-world problems. The system is a support tool to examine the pragma specifications, and directly compiles a KL1-c program with the pragma into Prolog, a sequential logic language.

4) Distributed implementation

A distributed implementation of KL1 was developed on the Multi-PSI according to the basic design of the components of KL1 (tentative specifications were used for KL1-p). The specifications of KL1-u in the distributed implementation are based on the KL1-u specifications on the PSI established, and are added with a pragma which allows programs to specify load distribution among processing elements (PEs). For KL1-b, the abstract machine language of the system, the syntax of body goals was expanded by introducing the pragma, to permit, for example, display of the PEs which should execute goals and the priority of goal execution.

5) Execution environment of programs and enhancement of program transformation technique

A GHC execution environment program has a hierarchical structure in which an abstract machine layer is sandwiched between the hardware layer and the user layer. The abstract machine layer contains differently shaped multiple abstract machines. The execution control program for user processes is installed on an abstract

most suitable for solving a goal and executes the goal on that machine.

The GHC execution environment program consists of an abstract machine unit, an execution control program unit, and a supervisor unit to coordinate the two units.

The supervisor/abstract machine layer has a supervisor and differently shaped multiple abstract machines. The execution control program layer is an execution control program to execute user processes in the user layer on abstract machines. It consists of a user interface, a shell, a server for user processes, and a device server. In the GHC execution environment program, the PSI was used as the hardware, and the pseudo Multi-PSI, a simulator for the Multi-PSI, was used as the distributed processing system.

iii. KL2

Research into this subject was continued to investigate Kernel Language Version 2 (KL2) basically and accumulate experience at an application level. The results are given below.

KL2 is an extension of KL1 and will also have a hierarchical structure ranging from a user language to a core language including a pragma, or to a base language. KL2 is being studied in the following areas to improve programmability and implement functions not offered by KL1:

1. Knowledge representation (Ref.9)
2. Parallel problem solving
3. Support for increased software productivity
4. Description of real-time systems
5. Integration with knowledge bases

These areas can be positioned in the structure shown in Figure 11.

The core language should be essentially an extension of GHC. In GHC, however, all programs have a guard, making program transformation (unfolding) more difficult. Improvement on GHC and design of a new language are under investigation for easier program transformation and partial evaluation.

The pragma language does not describe logical execution of programs. It mainly describes utilization and control of program execution. Three areas must be investigated for the pragma language: implementability of high-degree parallelism, control on search, and unified handling of meta.

(2) Problem-solving and inference software module

Major research subjects in this area are parallel inference software, basic meta-inference software, basic coordinative problem-solving software, and demonstration software.

i. Parallel inference software

An experiment on methods to describe various parallel applications and parallel algorithms was conducted for evaluation. It had been planned that the experiment would feed the results back to give the execution environment and specifications of KL1. A detailed design was developed for PIMOS, an operating system for parallel inference machines. PIMOS is designed as parallel inference control software to provide users with an environment of parallel languages. Application-based evaluation of the parallel inference software was conducted in two areas: (a) languages to describe problems concerning a parallel operation system and methods to compile the languages into KL1 (FGHC), and (b) applications of KL1 such as LSI layout design and syntactic analysis for natural language processing.

1) Parallel inference control software

As with conventional sequential computers, parallel inference machines cannot control program execution smoothly without an operating system, the basic software to manage program execution.

PIMOS is designed as the operating system for the Multi-PSI version 2, intermediate-stage PIM consisting of 100 PIMs, and final-stage PIM, or fifth-generation machine, consisting of 1000 PIMs. A revised version of the original PIMOS will be installed in the final-stage PIM.

2) Application-based evaluation of parallel inference

It is almost confirmed that the committed choice type parallel logic language, the base for KL1, has sufficient descriptive power for clear algorithms. Prolog is easy to use for general problem solving. It is not clear that the committed choice type parallel logic language can be characterized likewise. The application capability of the language was evaluated, aiming at solving problems in a parallel operation system where multiple components run in parallel. Problems on problem solving for a parallel operation system were defined, the committed choice type parallel logic language was shown to be too primitive as a language to represent problems, and an approach based on a problem-oriented language was proposed. The approach consists of a language simply representing problems and an inference mechanism including search and simulation based on the representation. The language contains both AND and OR parallelism. The parallel operation system description language, ANDOR-II, is a parallel logic language with an all-solution search function. The function is implemented by adding OR parallelism to FGHC, which has AND parallelism. A given problem can be described declaratively and easily using AND parallelism to represent parallel events and OR parallelism to represent objects with multiple possibilities (possible world).

3) Programming of search problems

An effective search technique must be established for problems in which the search space is very large and the object to be searched for has a complex structure.

LSI layout is one such problem. The search space is very large because a vast number of components must be laid out, and problem solving is complicated because the layout problem is on the border between logic design and device design and is therefore restricted by both areas. This makes it necessary to search for solutions quickly and intelligently. LSI layout differs considerably between design methods (e.g., custom LSI, silicon compiler, standard cell, gate array). This research focused on the standard cell LSI, because its design is automated to some degree by existing CAD systems and can therefore be evaluated relatively easily, and because fairly large scale LSI can be designed.

The results of the investigation described above suggest

that interactive search in which a designer and a system cooperate to solve problems is possible for the building block layout, whereas a parallel search for increased speed is effective in cell layout. CIL (complex indeterminate as language element), which can describe constraints positively, and IQ (incremental query system), which provides an interactive execution environment, have a function to execute a sequence of Prolog goals step by step and partially change goal sequences. CIL and IQ were used to develop a prototype building block layout system, which was then evaluated. A prototype cell layout system for evaluation using GHC and capable of describing parallelism positively was also built.

4) Parallel syntactic analysis

A parallel syntactic analysis technique suitable for implementation in parallel logic languages was proposed. The technique exploits the features of committed-choice language, a parallel logic language which describes given natural-language grammar categories (non-terminal symbols) as processes able to run in parallel. The technique can handle in a unified manner all of the logic-language-based grammar description formats proposed so far, such as definite clause grammars (DCG), extraposition grammars (XG), and gapping grammars (GG). It also works as an extremely efficient parsing algorithm for these logic grammars even in sequential execution environments, because it does not repeat the same processing during analysis and a program is completely compiled into a logic language.

The parallel syntactic analysis method for logic-language-based grammar description languages is an extension of the parallel parsing technique for DCG which was designed to basically handle the structure of context free grammar (CFG). The structure of the grammar rule to be processed was extended so that it could also handle context sensitive grammar (CSG) and 0-type grammar with a higher-level descriptive power and XG and GG, which can describe a relation between non-adjacent elements. A method to convert a grammar written in these grammar description formats into committed-choice parallel languages such as GHC, Prolog, and Concurrent Prolog was given. In the converted program, all grammar components, such as words and grammar categories, are expressed as predicates of a logic language, each moving as a process able to run in parallel. Each process corresponds to a partial tree of a parsing tree, making it unnecessary to store an interim result during analysis as a side effect. These features permit parallel syntactic analysis to be used as an extremely efficient parsing system even in sequential execution in Prolog. The technique enhanced seems to work as an effective parsing system for GG and other grammar description formats for which efficient algorithms have not been found. The algorithm of the technique is described in the Prolog-based meta-program; however, it can also be easily implemented in parallel logic languages such as GHC. Parallel parsing seems effective for various grammar description formats, particularly GG, because even no efficient sequential algorithms have been found for the syntactic analysis of GG. It also provides a practical parsing algorithm for CSG, because it ensures termination unless cyclic rules are included.

ii. Basic meta-inference software

Research in this area seeks to implement the sophisticated inference ability possessed by human beings in software and to develop an environment and tools to support the implementation.

1) Automated partial evaluation of Prolog programs

The partial evaluation of Prolog programs serves as an effective optimization technique in meta-programming but it suffers from poor execution efficiency. PEVAL, a goal-directed partial evaluation program, was implemented to increase the efficiency of the partial evaluation. The goal for the present was to implement all layers of partial evaluation programs (Peval, Peval', Peval") with the same basic material. As the first step toward this goal, the aim was to develop an automated powerful partial evaluation program in Prolog. The most crucial problem in this research is to what extent recursively defined predicates must be expanded. An automated program was successfully developed on a multi-phase-structure basis. This was accomplished by statically analyzing the partial evaluation program to introduce conditions which prevent recursively defined predicates from being expanded infinitely in the program.

2) Proof support system

The first step to construct a logic model is to establish a logic system suitable for the characteristics of the domain involved. Then a model for the domain is developed by establishing an axiom system in the logic system. Finally, the characteristics of the defined model are verified through the configuration of proof within the model. The investigation was conducted extensively from various aspects, such as interface and description language, by introducing actual and virtual keyboards and a logic expression input editor, a structure editor corresponding to the tree structure of a logic expression.

3) Computer algebra system

Research into this subject was conducted using the algorithm approach. The survey and investigation were carried out on algorithms for computation of polynomial greatest common divisors (GCDs) and factorization, canonical simplification with respect to algebraic relations, and integration in finite terms, and prototypes were developed for arbitrary precision arithmetic to be used as the basic parts of the computer algebra system. A further survey was conducted on factorization and integration. The polynomial factorization and computation of GCDs were studied. A new concept called p-factor and p-decomposition was introduced for the factorization of uni-variate polynomials over finite fields. The concept of separability was also introduced with respect to the roots of a Zassenhaus' minimal polynomial to distinguish the irreducibility of a factor obtained from the roots. A theoretical study was also made of the lattice algorithm, a factoring algorithm with polynomial time complexity, and it was demonstrated that lattice algorithms can generally be applied to the computation of GCDs and

factors of elements in a Euclidean ring which satisfy given conditions. (In lattice algorithms, computation can be performed in polynomial time with respect to the degree of the polynomial to be factored.) Particularly, for uni-variate polynomials over integers, experimental factoring algorithms are programmed according to the above results. In R&D on the computer algebra system, survey and research were conducted to achieve a system capable of advance integration of elementary functions in finite terms with system implementation. Programs required as basic parts are being prototyped one by one from the bottom.

iii. Basic coordinative problem-solving software

Basic coordinative problem-solving software is an important application of parallel problem solving. A language useful for describing coordinative problems was designed, and R&D was continued on programming methods for coordinative problem solving and applications using parallel kernel language KL1. Mandala, an object-oriented knowledge representation language, had been developed as a descriptive language for coordinative problem solving, but the object-oriented part of Mandala had not been completely implemented in KL1.

Afterwards, an object-oriented language implemented in KL1 alone was proposed and a language was designed which can execute procedures in an object in parallel, with a few restrictions as possible imposed on its parallelism (Ref. 8).

A software reutilization system for parallel coordinative problem solving has been under study. Program reutilization is a program development method which saves existing software as a part so that it can be reused in developing similar software. Mandala has been selected as the language to develop a prototype system.

Research into a KL1-based application system in the logic device CAD field aimed to define a procedure for implementing the application system in a parallel programming environment through program development. Investigation was conducted on specification representations suitable for the style of KL1-based programming, and the results were used to develop a prototype system.

(3) Knowledge base management software module

i. Knowledge representation utilization system

Research in this area aims to investigate knowledge representation and inference engines for representing a dynamic system and conducting qualitative reasoning on the system. The investigation will be necessary for evaluating the knowledge representation capability of Mandala and the knowledge base management system. The investigations are outlined below. Functional design of the knowledge base management system in Mandala is discussed from the standpoint of functional design in the section on the cooperative problem-solving system and other related sections.

The first step of the research was investigating a reasoning system to make a computer understand the world of physics. With knowledge similar to laws in physics, the system will infer temporal transition of the state of a physical system from the given initial information. It was assumed that the knowledge is based on physical laws in textbooks of physics.

The qualitative physical reasoning system (Qupras) uses the physical laws to infer:

1. Relation between the objects forming a physical system
2. Temporal transition of the state of the system; i.e., what will happen to the entire system at the next moment

Qupras infers the relation between objects and transition of a state while performing qualitative reasoning using knowledge of physical laws and objects.

ii. Basic knowledge acquisitions system

The basic knowledge acquisition system has been studied as a basic software technology for knowledge base management. The results of the activities in this area are given below.

(a) Hypothesis generation and verification system

Conventional deduction-based reasoning systems start with existing general knowledge and proceed to specific knowledge.

Therefore, these systems cannot draw an effective inference for facts not included in the general knowledge. This problem can be solved with a system capable of performing non-deductive inference, which requires a sort of logical leap such as induction, analogy, or common reasoning. An inference result is essentially a hypothesis in such a system; the system can generate hypotheses. Investigation on such inference has just been started. In the investigations, hypotheses were given as a set of possible hypotheses in a system; the system did not actually reach a solution for generating hypotheses.

A theoretical study was conducted to gain perspective on the implementation of systems with such inference capability. The results of the study were used to propose a new theory called Ascription, a general basic theory on inference involving logical leaps.

(b) Analogy system implemented in meta-program

The general meaning of analogy is to find a similarity between two or more events and draw some inference based on the similarity. This discussion assumes that analogy is an inference category for obtaining unknown facts which cannot be obtained by conventional deduction.

Inference rules were implemented in meta-programs using logic programs and incomplete logic data bases as examples. An investigation was conducted on ways to increase the efficiency of similarity-based prediction of facts by performing partial evaluation on a given similarity. Future research subjects include the establishment of techniques to discover a similarity between attributes and the interrelationship between similarities.

(c) Increased efficiency of logic program by program transformation

Programs declaratively written are easy to understand. In general, however, they run inefficiently under the standard depth-first search strategy for Prolog in which goals are executed from left to right. Research into this subject aims to improve the execution efficiency of declaratively written easy-to-understand programs by transforming and

synthesizing them into equivalent but more efficient programs.

A program transformation technique for typical programs in the generate-and-test type was studied. This study has some features not found in research based on conventional program transformation.

(d) Knowledge base management by parallel logic language

To endow computers with problem-solving capabilities possessed by professionals, knowledge used in solving problems must be stored and managed in data bases. A knowledge base management system provides functions to represent experts' knowledge in a format that can be understood by computers, and acquire knowledge and save it in data bases systematically. A knowledge data base was devised to accumulate and manage knowledge expressed in a set of Horn clauses and ways were studied to implement the knowledge data base in GHC. The possibility of constructing a knowledge base management system which will process rules and facts was checked, and it was confirmed that data bases consisting of rules and facts, system control programs, and knowledge base management programs can be handled with a single framework of logic programming. Three methods for parallel control over GHC-based relational operation processes were also investigated. The investigation showed that parallelism differs between the types of commands when the processing granularity is made smaller by dividing data.

The problems had logically good characteristics. Actual knowledge base management systems involve a number of more complicated problems.

(e) Learning function in algebraic manipulation

Mathematicians solve problems using expert knowledge in mathematics. They choose an appropriate method which seems to help solve problems and sometimes try to reach a solution on a trial-and-error basis. The expert knowledge they use includes object knowledge which directly transforms a problem and meta-knowledge which represents knowhow to select object knowledge. Research into this subject aims to study the learning process of mathematicians by focusing on learning from examples.

Learning can be roughly divided into two types: one obtains knowledge on a domain (in this example, the domain of mathematics), the other obtains knowledge on how to learn. The following three levels of learning in the domain of algebraic manipulation were investigated and part of the detailed design developed.

- Rule learning: This type of learning in the algebraic manipulation draws general terms from a series (mathematical progression). It corresponds to a problem in conventional inductive inference.
  Concept learning: This type classifies rules in the algebraic manipulation.
- Strategy learning: This type learns heuristics to apply rules in the algebraic manipulation.

iii. Basic distributed knowledge base management software

Research into this theme aims to provide a dictionary data base for natural language processing application systems and a proof data base function for theorem-proving systems. Two subsystems are currently under study and development: the knowledge base management subsystem and the knowledge base manipulation subsystem.

(a) Distributed knowledge base management subsystem (Kappa)

The distributed knowledge base management subsystem, Kappa, uses the non-first normal form (NF^2) model instead of the relational model as the model of the lowest layer to efficiently handle knowledge with a complicated structure. The NF^2 model can be considered an extension of the relational model. Models for knowledge representation capable of handling terms and taxonomy are positioned. A prototype system was partly developed to confirm whether Kappa can be implemented.

(b) Basic distributed knowledge base manipulation software (SIGMA)

SIGMA is a knowledge base management and manipulation system built on Kappa. It is designed as a support tool for expert system development. SIGMA will use KSI, a knowledge representation language based on the concept of sets, to represent knowledge and to manage knowledge bases, perform inference, explain an inference process, and keep knowledge bases consistent. A prototype of SIGMA was partly developed to confirm the feasibility of the software. Investigation was also conducted on the interconnection between SIGMA and Kappa which will be implemented.

(4) Intelligent interface software module

i. Pilot model for discourse understanding system

A basic study of the pilot model was made and its prototype partly developed. The results of these activities were evaluated, and detailed theoretical and technological studies were conducted on anaphora processing, a crucial technology for context processing.

The discourse understanding system, DUALS (Discourse Understanding Aimed at Logic-based Systems), is an experimental system to build a logic-programming-based computational model for discourse understanding. To construct such a model, it is necessary to review problems, such as identification of ellipsis and anaphora, speech actions, and planning, from a computational standpoint, and to formulate inference. DUALS version 1 is the first attempt to implement the situation semantics theory as a foundation for discourse understanding models. It has been developed in Prolog. Prolog's term description alone, however, is not always sufficient to express complicated structures such as semantic structures of sentences, making it difficult to gain perspective on the entire system. This problem is also a stumbling block in problem domains, such as context understanding, where a number of unknown factors are involved.

DUALS version 2 was developed based on this experience. Design of the second version aimed to further clarify the structure of the system and provide a basis for future experiments on various discourse understanding. CIL was used to provide a unified description of the entire system . CIL's functions to represent and unify partial terms were used to

clarify the system structure effectively. The latest work in situation semantics was also introduced to represent semantic structures of sentences and expand the framework of situation semantics.

DUALS version 2 consists of the six modules shown in Figure 12: text analysis, object identification, discourse analysis, problem solving, text generation, and I/O routine.

The text analysis unit performs morphological, syntactic, and semantic analyses on a sentence made up of kanji (Chinese characters) and kana (Japanese syllabary), and generates an intermediate representation (event type) of the sentence. The unit uses a variant of Watanabe grammar and implements rapid parsing using the SAX analysis algorithm. The object identification unit matches objects (generally noun phrases) which appear in the intermediate representation of a sentence to objects in the real world. It also identifies anaphora and ellipsis. The discourse analysis unit interprets speech actions and stress discourse structures. The current unit is very primitive; it only determines whether the input sentence is a query or not, and stores the description situation of each sentence in the list structure. The problem solving unit obtains a solution to the query by performing a search or inference. The text generation unit generates a Japanese sentence from the intermediate representation (descriptive situation) of the input sentence. When the sentence to be generated is a solution to a question issued by the user, the sentence generation unit first produces the intermediate representation of the output sentence from the intermediate representation of the question and the solution from the problem solving unit, then generates a surface sentence. The I/O routine provides a user interface. It uses the Pmacs editor on SIMPOS to permit user-friendly data entry. DUALS version 2 is written in CIL and ESP, and installed on the PSI. As in the first version, sample sentences were extracted from tests designed to measure the reading skill of Japanese students in elementary schools.

The results of the design and installation of DUALS version 2 showed that the semantic representation based on situation semantics and the processing modules will provide a basic system in discourse understanding experiments in the future.

The model proposed by Barwise was introduced as the framework for identification on anaphora relations in the anaphora treatment. In this model, indexing which represents anaphora relations is performed on a subset of English, and a model-theory-like interpretation is given by considering it a logical expression. Unlike interpretations in ordinary predicate logic, this interpretation is regarded as procedural, and assignment of a value to the logical expression is considered a partial function. Therefore, this framework seems suitable for implementation in computer programs; especially, it appears to provide excellent compatibility with CIL, which can represent a partial value assignment as an associative list. It was shown that Barwiseś model based on situation semantics can be easily installed in CIL. An example of detailed design of the model was demonstrated.

The model, of course, does not provide a framework capable of treating all anaphora phenomena. Particularly, the interpretation rule shown above is directly established based on the syntactic structure of English (L(SS)). This seems quite different from the framework of Kamp which is very closely related to the model. That is, intermediate structures such as Kampś DRS (discourse representation structure) are removed from Barwiseś model. Neither framework, however, can be used directly as a model for Japanese, because its syntactic structure differs from that of English. Still, the idea of partial assignment and dynamic interpretation shown in Barwiseś model provides a basic approach to anaphora treatment in Japanese.

ii. Basic sentence analysis and synthesis software

Research in this area seeks to develop Japanese phrase structure grammar (JPSG), a grammar system to describe basic Japanese structure and its language processor.

A basic framework to describe grammar was designed and used to describe one of the most basic structures. The framework was enhanced by expanding the area of linguistic phenomena involved; new features and principles were introduced according to the expansion. It was demonstrated that the framework can also describe and formulate Japanese morphology. The sentence level concept was introduced, and the priority of the elements forming a sentence, such as tense, aspect, and modality, was established. A study was performed on whether the structure of a sentence consisting of these elements could be governed by any rule. The same investigation was performed on complicated sentences involving these elements and conjunctive postpositional particles. The resulting framework was then used to experimentally describe sentences extracted from Japanese readers for sixth-grade students in elementary schools.

A prototype parser based on the idea of JPSG has been under development to demonstrate the feasibility of applying JPSG to language understanding. An attempt was made to install some of the functions of JPSG for evaluation using two schemes: conditional unification and CIL-based GALOP. A grammar description language was also designed to provide an interface between linguists and the parser, and experimentally used. The results of the evaluation are summarized below.

The conditional unification technique allows most of the grammar descriptions of JPSG to be converted easily into programs. This facilitates the development of an analysis system to verify JPSG, a goal set up for this research subject. The technique does not depend on an analysis algorithm.

The processing efficiency of the parser was not very high, because the conditional unification was implemented as Prolog programs. The parser implemented in C-Prolog on the VAX 11/8600 took about 20 seconds to analyze the sentence: "(Ken was made to read a book.)". The processing speed is relatively fast, considering that the conditional unification simultaneously searches all possible solutions.

(5) Intelligent programming software modules

Under the theme of intelligent programming software modules, research and development was conducted on the experimental program transformation, proof, and synthesis system, software knowledge management system, and experimental program transformation, analysis and verification system. Demonstration software was developed for

these systems.

1. Experimental program transformation, proof, and synthesis system

1) Theorem proving system

The theorem proving system must have functions to process a huge amount of mathematical knowledge. The system must also have utility functions for proving theorems such as proof editing, two-dimensional I/O, and formula manipulation functions.

This system is used primarily as a notebook by mathematicians. It may also be used as a CAI system for mathematics education, an expert system, or a problem-solving and inference system. It will be particularly useful for programming support; thus, this application of the system will be an important theme in the future.

When development of the theorem proving system started, the following areas were chosen to be covered by the system:

- Linear algebra (LA)
- QJ
- Synthetic differential geometry (SDG)

A prototype of the system, named the CAP-LA system, was developed using the DEC10-PROLOG machine, and its first version was developed. Based on these experiences, the second version of the CAP-LA system was completed.

The CAP-LA system deals with linear algebraic theories appearing in textbooks for first-year university students. Tsuchiyaś textbook was used to design and evaluate the proof description language (PDL) and system. As a checker of proofs written in PDL, the CAP-LA system supports evolution of linear algebraic theories. Development of the second version of the CAP-LA system emphasized improvement of the following features:

- Proof speed
- Inter-line interpolation function
- Verification process display functions for system developers and experts
- Structure editing function and syntax guidance function
- Representation of proofs in a natural language

The objectives were achieved satisfactorily.

2) Term rewriting system

The heaviest mathematical work load on researchers is formula manipulation. Generally, the work requires complicated and profound inference concerning equal signs.

Therefore, the term rewriting system is an important subsystem for the theorem proving system. Having both proving and computing functions, this system can be used alone or can support many other systems.

As a theorem proving subsystem, the term rewriting system is typically applied to equational logic. It is also useful for proving theorems based on predicate logic. The latter is a typical application of a SDG theorem proving system. Other applications of the system may be program synthesis and verification.

When its computing functions are brought to the fore, the term rewriting system is a programming language and its executor. Although term rewriting is regarded as a functional programming technique, the system can also handle logic. Thus, it is a candidate logical language making up for its weak points in representing mathematical functions. The kernel technique for realizing a logical as well as a functional programming language may be term rewriting.

An experimental system called Metis was developed. It has the following functions based on the term rewriting technique:

- Determining directions of equations and converting equations to rewriting rules
- Removing ambiguity from rewriting
- Executing rewriting

Moreover, Metis was improved as described below. The biggest problem for Metis is processing equations such as the commutative law because, once this law is converted to a rewriting rule, the possibility of terminating the rewriting cannot be guaranteed. To solve this problem, an AC-unification algorithm and an AC-completion procedure into which associative and commutative laws had been incorporated were first implemented. An unfailing completion procedure that does not necessarily convert equations to rewriting rules was then implemented.

Another problem is how to treat not-equal signs, i.e., unequal relationships. However, this was completely solved by implementing Jieh Hsiang's S-strategy.

3) Program construction system (PCS)

There is a lot of similarity between proving mathematical theorems and computer programming. This fact is recognized from mathematical and other perspectives. The realizability interpretation given by Kleene in his theory of constructive mathematics is a principle of extracting programs from proofs. The concept of programming constructive proving was researched. Since a number of systems based on this concept had already been reported, their features were investigated.

Sample programs were then studied to seek techniques for developing such a system. For this work, Satohś QJ was used as the logic system and Quty as the programming language. Partly modified PDL was used as a specification and proof description language in CAP-LA and its affinity with the theorem proving system was checked.

Figure 13 shows the image of PCS. The proof editor and system manage, proof parser, and proof verifier have the same functions as the CAP system. The module knowledge base corresponds to the proof data base in the CAP system and is used to store and retrieve created modules. The proof compiler is the module extracting programs by analyzing verified proofs.

2. Program design, development, and maintenance system

The R&D results of software design support system using a knowledge base are outlined here.

Fundamental measures covering the whole software life cycle are now needed to improve software productivity and

maintainability. Reusing software resources and accumulating and transferring software development techniques are becoming particularly important to cope with the shortage of software engineers. The focus is on developing a system with a data base , supporting software transfer techniques and analysis/verification of designs. The techniques to be transferred are closely associated with knowledge information processing techniques such as knowledge representation and application techniques.

Models of design activities were investigated to clarify the role of a knowledge base in designing software. The concept of a three-source software design model was then implemented. In this concept, a software design model is a set of interactions between three systems: the processing system (designers); the resource system (knowhow, design information, and program parts); and the system being processed (specifications and programs to be designed). According to the concept, a software design activity is defined as "an activity of the processing system to obtain necessary information from the resource system and use it to convert the processed system from specifications to refined specifications to a program".

3. Software knowledge management system

The R&D goal of SIMPOS is to provide a better environment for logic programming. The development placed importance on the following items:

- Unifying interfaces between subsystems
- Management of multiple PSI machines connected by a network
- Flexible handling of abnormal conditions (errors)

The internal algorithms and execution efficiency of modules were also improved.

The logic language, ESP, having object-oriented functions, was used to describe the system.

As a result, the following objectives were achieved:

- Compatibility between the system and application programs through the logic language
- Accessibility to the system through an object-oriented paradigm

§5 Development Support System

(1) Parallel software development machine

i. Software

The software system consists of the connection control software, which runs on the parallel software development machine, Multi-PSI, and part of the SIMPOS programming and operating system installed on the PSI, the processing element of the Multi-PSI (Ref.5).

The connection control software links and coordinates multiple PSIs. It provides a function to control a dedicated network and a user-interface function to enable a host PSI to control execution smoothly on the networked PSIs.

SIMPOS running on the PSI is vital for smooth operation of the Multi-PSI, system. The PSI, serving as the PE of the Multi-PSI, is also designed to work as a standalone sequential inference machine. Therefore, the PSI offers a user-friendly interface for the Multi-PSI or true parallel inference machine systems in the future, because it can be used as a front-end processor or a work bench to provide a cross compiler for program development. Basic development and improvement were completed for SIMPOS. The subsystems of SIMPOS were improved and enhanced so that the PSI could be used more easily as the processing element or front-end processor of the Multi-PSI. The PSIs have become widely used in many kinds of research areas with the improvement of available software tools.

ii. Hardware

The Multi-PSI system is being developed in two stages ; the first version and the second version (Figures 14 and 15). The first version has a structure which can be completed in a short time to enable research on parallel software to be conducted as soon as possible. It can act as an experimental system for the design of a larger, more advanced version of Multi-PSI, Multi-PSI version 2. The first version is a small-scale multi-processor system made up of six to eight PSIs linked via the connection network hardware developed.

In the development of the Multi-PSI version 2, the PSI itself is being improved and reduced in size, because the second version should have more CPUs and higher performance. The connection network is also being refined to link up to 64 PSIs. The entire part of the KL1 processor will be implemented in firmware to achieve high performance. An improved, smaller processing element is also being configured as a separate standalone system to be used as the front-end processor for the Multi-PSI version 2 or as a simulator. This system, a smaller version of the PSI, is called the PSI-II. A prototype of the Multi-PSI version 1 was developed for evaluation. The connection network hardware and processing element hardware of the Multi-PSI version 2 were designed in detail, and a prototype PE was developed. The PE hardware of the Multi-PSI version 2 was used to conduct the partial prototyping of the PSI-II.

The R&D results of the hardware system for the parallel software development machine are outlined below separately for the connection network hardware and PE. Various items disclosed in the evaluation and the results of the detailed investigation will be used to develop prototypes of the Multi-PSI version 2 in the future. In this development, prototypes will be characterized more as small-scale parallel inference machines than as parallel software development machines. This shift in approach is expected to facilitate R&D of parallel software considerably, including parallel inference control software PIMOS, and R&D of the intermediate-stage and final-stage PIMs.

Detailed designs were produced for the hardware and firmware of the PE, and their prototypes were partially developed. Detailed designs were also developed for the PSI-II hardware using the PE hardware and for its firmware. This was followed by partial prototyping. The PE hardware used as the engine of the parallel software development machine is designed to run fast and have a large-capacity memory. The design also stressed compactness and low power consumption so that a system consisting of 16 to 64 PEs could be configured relatively easily.

(2) Improvement and enhancement of sequential inference machine (high-speed processor module)

The high-speed processor module is an extension of the sequential inference machine. It is a dedicated processor capable of rapidly processing programs in a Prolog-based inference language and has a large-capacity memory. R&D on the module consisted of designing improved and enhanced software and producing functional designs of smaller hardware. Three types of software were developed to make full use of the power of the smaller version of the module, called the smaller CHI (Co-operative High Performance Sequential Inference Machine): programming system software, basic software for the CHI, and an ESP translator to provide functional compatibility with the basic part of the SIM. A detailed design of the smaller hardware and its prototype were developed. The firmware of the high-speed processor module was also designed in detail and coded to complete the R&D of the module.

The development results are outlined below for the software and hardware systems.

i) Software system

The R&D activities consisted of 1) improvement and enhancement of the programming system software, 2) improvement and enhancement as well as design and prototyping of the basic software for the CHI, 3) basic design and prototyping of the ESP translator, and 4) development of a firmware simulator for the CHI, a development tool for the smaller CHI firmware. These activities provided the expected results.

ii) Hardware system

A detailed design and circuit design were produced for the smaller CHI. A prototype was then developed and checked for performance. The firmware of the smaller CHI was also designed in detail and coded. The development activities for the smaller CHI consisted of the following: 1) research, development, and manufacture of two types CMOS gate arrays, 2) research and development, circuit design, and manufacture of processor modules. These were required to replace CMI. devices used in the CHI developed with CMOS and TTL devices, thereby making the hardware more compact. 3) Circuit design and manufacture of the main memory module and memory interface module; these were conducted to introduce higher-density memory devices (1 Mbit) for smaller hardware. 4) Circuit design and manufacture of the host interface module; these were conducted because the I/O control module was changed to make the system smaller.

(3) Improvement and enhancement of network system

Development of the network system in the FGCS project is being conducted in two areas: a SIM network system and an international network system.

i) SIM network system

The SIM network system is designed to facilitate R&D of the FGCS project. It connects the sequential inference machine PSIs installed at ICOT and its related research institutions via Ethernet-type LAN and DDX. The network functions designed and partially prototyped were improved and enhanced to build a network system. The system is currently in operation, providing communications services, such as electronic mail between 18 sites including ICOT.

The structure and functions of the system are outlined below.

1) Network system structure

Figure 16 shows the entire configuration of the SIM network.

(a) LAN interface adaptor (LIA)

This device performs interface conversion and protocol conversion to connect a host to a LAN.

(b) LAN interface board (LIB)

This board provides on IEEE488 interface between a host and a LIA. The board is installed in the host.

(c) NCLIA + network console

This pair provides the network manager with functions such as management on diagnostic information and network structure information and collection of log information. Network structure and other information is sent to hosts by communications between the LIA and NCLIA.

(d) Bridge (BRG)

Unit LANs are connected to each other by the BRG, which allows inter-LIA communication across the different unit LAN.

(e) Gateway (GW)

The GW is a public network connection unit which uses the DDX-packet network and allows LIAs linked to a composite network to communicate with LIAs linked to another composite network. A composite network consists of one or more unit LANs interlinked with a BRG. Generally, one site consists of one composite network.

(f) Transceiver (TC)

This unit is connected to a coaxial cable, the transmission line of the LAN. It sends and receives signals via the cable, and provides electrical insulation.

(g) H/SH

"H" represents general host computers, such as the PSI and general-purpose computers, while "SH" means server hosts. In principle, a server host is used in each composite network and works as a mail server, print server, and file server for general hosts.

2) Network functions

The network functions of the SIM network system are divided into basic and application functions. Application functions use the basic functions to provide users with various useful services.

(a) Basic functions

The basic functions provide the network-based interprocess communications function, offered by the LIA, to programs running on the PSI. The functions are supported by the network control program (NCP).

A new NCP was developed based on the logic design produced previously. The new NCP has the following

features:

1. Provides a 1:n communications function
2. Provides commitment mode
3. Provides datagram mode
4. Provides value-added mode enhanced in function and reliability in 1:1 communication
5. Allows a heap and string to be used as a data transfer buffer
6. Enhances simplified communications interface (n line)
7. Coexistence with user-specific NCP

(b) Application functions

The following application functions of the SIM network provide users with a variety of useful services:

1. Remote object

This function provides a basic mechanism which allows the user of a PSI to perform object generation or deletion, method calling, and other operations for another PSI. The operations, however, are restricted. It was decided to develop various systems forming the application functions based on remote objects.

2. File server system

This system provides a remote file and directory access function and a manipulator function. The remote file and directory access function permits the user program to access files and directories of all PSIs and general-purpose machines connected to the SIM network. The manipulator function uses the remote file and directory access function and the functions of other server systems. It allows the user of a host to copy, print, and perform other operations on files and directories of another host. This system was developed based on remote objects. It is provided to users as part of the file system function.

3. Mail server system

This system provides a mail function and an electronic bulletin board function. The mail function permits users to exchange electronic messages. It has various optional functions such as forwarding mail and producing printout. The bulletin board is something like a blackboard that can be shared by several users. Users can exchange their ideas via the bulletin board. Mail and bulletin boards are stored in the mailbox of the server PSI. An auxiliary server function was added to the system . This function stores mail and other information when the server PSI is disconnected from the network. It allows users to send mail when the server PSI malfunctions or is disconnected from the network.

4. Print server system

This system provides a remote print function and a manipulator function. The remote print function allows the user program to use the printers linked to all PSIs on the SIM network. The manipulator function enables the user at a terminal on the network to cancel remote print, display output conditions, and perform other operations. This system was developed based on remote object. It is offered to the user as part of the print system function.

5. Talk system

This system provides a full-duplex, real-time inter-user communications function via a bitmap display (BMD). The system has an additional function. When a user wants to communicate with another user on the network, this function lists the names of users currently using the PSIs on the network and allows the user to select the appropriate name. This function has substantially increased the ease of use of the network.

ii) International network system

The computer network plan of the FGCS project has been made around the PSI. At present, communications between PSIs and general-purpose computers are limited ; the only service supported is file transfer with VAX/VMS via Ethernet. Both overseas and domestic research institutions have long been requesting electronic-mail-based communications with ICOT over a computer network. The FGCS project therefore earmarked the problem of networking general-purpose machines as an urgent task. The SUN workstation was introduced to link computers at ICOT with other institutions. It has been used as a gateway to connect with computers at universities and other research groups. Connection to overseas sites was considerably enhanced by, for example, becoming a regular member of CSnet. At present, ICOT is officially linked to ARPA Internet via CSnet. A stable connection has also been established via the UUCP network with MIT and the following gateways: UKC in Britain, INRIA in France, and ENEA in Sweden. Connection to other countries will be implemented as the need arises in the future. ICOT is currently connected to domestic research institutions through JUNET and DECnet. Both networks support an increasing number of research groups. JUNET also supports electronic-mail-based communications with countries which have no direct link with ICOT. Development and enhancement of these networks have given ICOT sophisticated electronic-mail link comparable to that of leading research institutions anywhere in the world.

Networking still involves a number of technical problems. Services available on the international network are currently restricted to electronic mail. It is imperative to continue to enhance the network to provide the same level of reliability and service matching as the state-of-the-art ARPA Internet.

§6 Concluding Remarks

Current R&D activities at ICOT are also being used to pursue more sophisticated possibilities and to realize them so that the project can move on the the final stage smoothly. Some advanced subjects previously untouched will be handled. It was decided to start with these subjects as important issues to be studied.

Under these circumstances, the guidelines for the R&D activities were set as follows:

1. R&D will be conducted to develop basic research from the beginning and to achieve effective integration of individual research areas.
2. The R&D system established on the basis of the component and overall system technologies will be more

solidly organized in gradual readiness for research in the final stage.

Since ICOT is involved in the development of state-of-the-art technology, it is considered imperative to exchange information and research results with scientists working in related fields both at home and abroad. A research infrastructure in cooperation with universities and industry, which gives ICOT useful advice and supplements its activities, has been built up for basic research advancement.

Some researchers in advanced research areas have the cross-sectional view that the generation analogy will not hold very well in advanced information processing systems of the 1990s and that, because of the diversified research involved, the research base is seen more as branching rather than as a movement from one generation to the next. ICOT always listens to the pros and cons of the matter (Ref.6,7). From the beginning, ICOT has tried to verify the hypothesis that parallel architecture based on logic programming will be a next generation computer oriented toward advanced information processing in the 1990s. The main focus is to research computer science based on logic programming and to develop symbol crunching super parallel computer and knowledge information processing technology within this framework.

Another national project in Japan, less widely publicized, is reported to be challenging the U.S. lead in numeric supercomputers. As a result, it can also be said that both those qualities offer an excellent foundation on which to raise AI applications. Finally, an observer says, "While initiatives world-wide all have problems, they have also led to progress." He is quite right. We are looking forward to a good solution in the FGCS.

References

[1] "Fifth Generation Computer Systems Project", pp1~30, ICOT Report, Oct. (1986).

[2] S.Uchida, A.Goto,et al, "KL1 Execution Model for PIM Cluster with Shared Memory," pp338~355, Vol.1, Proc. of the 4th ICLP (1987).

[3] H.Itoh, "Research and Development on Knowledge Base Systems at ICOT", Proc. of the 12th VLDB Conf. (1986).

[4] K. Ueda, "Guarded Horn Clause," Logic Programming '85, E.Wada (Ed.). Lecture Notes in Computer Science 221, Springer-Verlag (1986).

[5] K.Taki, "The Parallel Software Research and Development Tool : Multi-PSI System," pp365~381, Proc. of France-Japan Artificial Intelligence and Computer Science Symposium '86, Oct. (1986).

[6] K.Hwang, et al, "Computer Architectures for Artificial Intelligence Processing," pp19~27, Vol.20,No.1, IEEE Computer, Jan. (1987).

[7] E.Shapiro, "Concurrent Prolog: A Progress Report" pp44~58, Vol.19, No.8, IEEE Computer, Aug. (1986).

[8] K.Furukawa, M.Ohki, et al, "An Object-oriented Programming Language based on A Parallel Logic Programming Language KL1", pp894~909, Vol.2, Proc. of the 4th ICLP (1987).

[9] Y.Matsumoto, "Knowledge Representation – In Favour of Logical Approaches", pp915~923, Vol.27, No.8, IPSJ Jr., Aug. (1986).

Acknowledgement

We would like to thank the Director, Dr.K.Fuchi, the Assistant Directors, Dr.T.Yokoi and Dr.K.Furukawa, the Research Laboratory Managers, Mr.R.Hasegawa and Mr. Y. Matsumoto, Dr. S. Uchida and Mr. S. Yoshikawa, Dr. H. Itoh, Mr.Y.Fujii and Mr. Y. Ogawa, the Managing Researchers, Mr.S.Taba, Mr.H.Amano and the previous Managing Researcher of ICOT, Mr.K.Takei, and researchers in the FGCS R&D for their useful advice, suggestions and co-operation during this research.

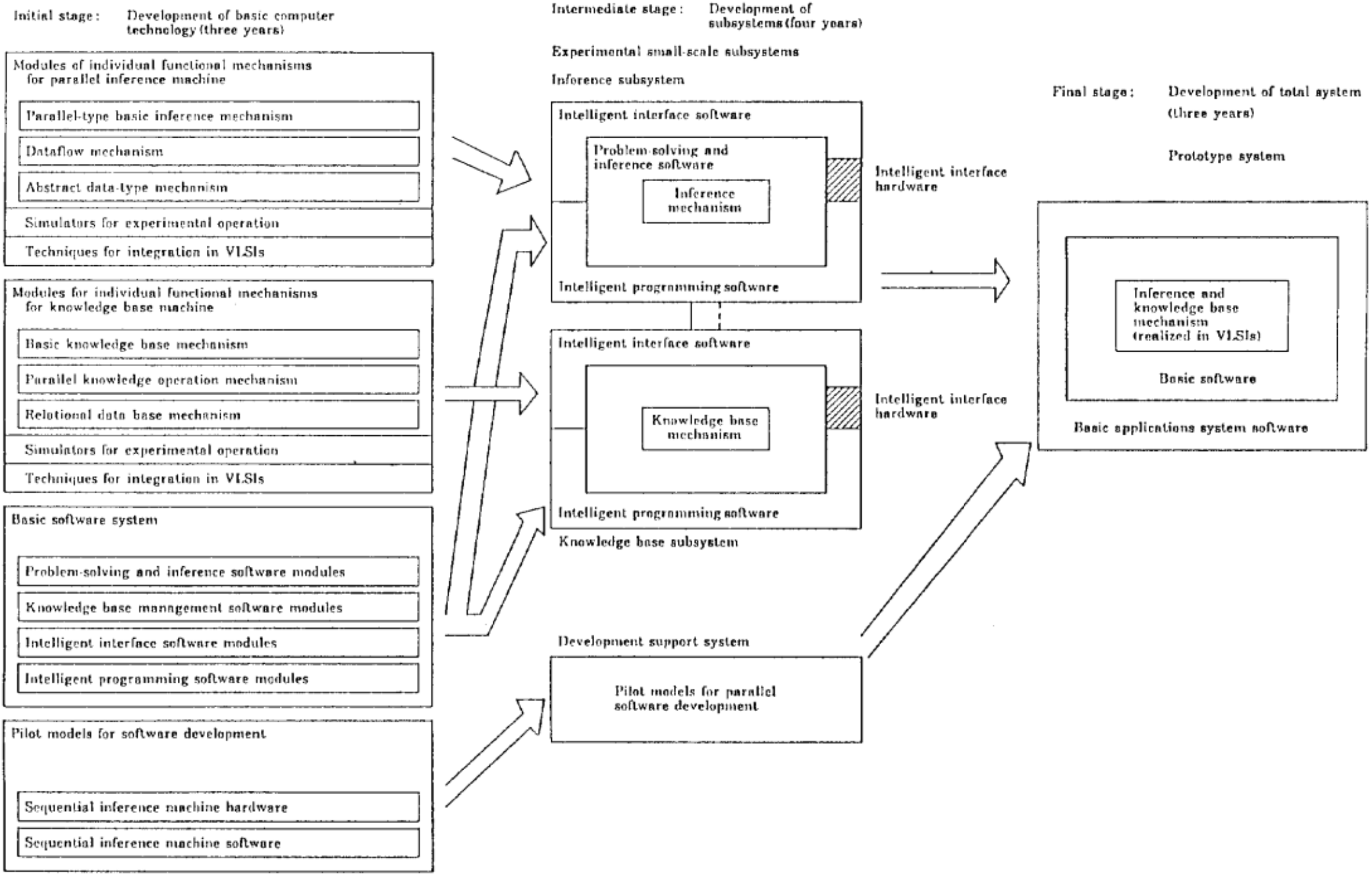
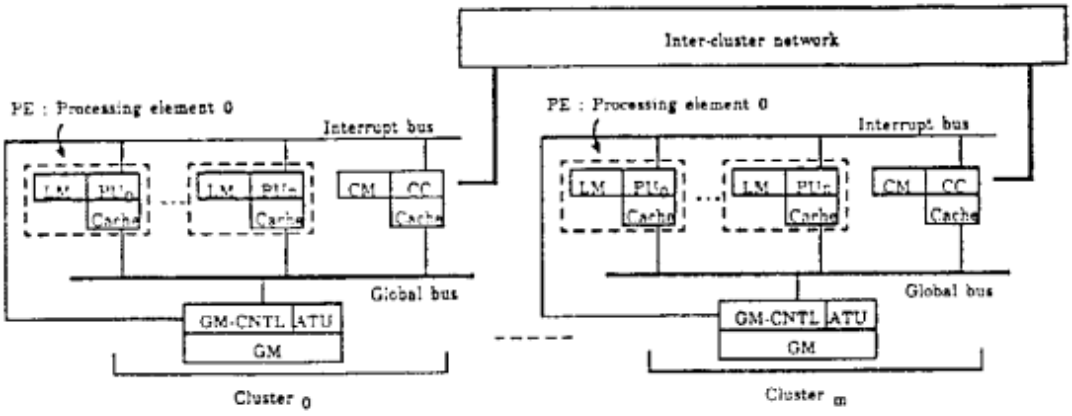


Figure 1 Stages of Fifth Generation Computer Research and Development



PU : Processor unit
CC : Cluster controller
LM : Local memory
CM : Cluster memory
GM : Global memory
ATU : Address transformation unit
GM-CNTL : GM controller

Figure 2 Hardware Structure of Intermediate-stage PIM

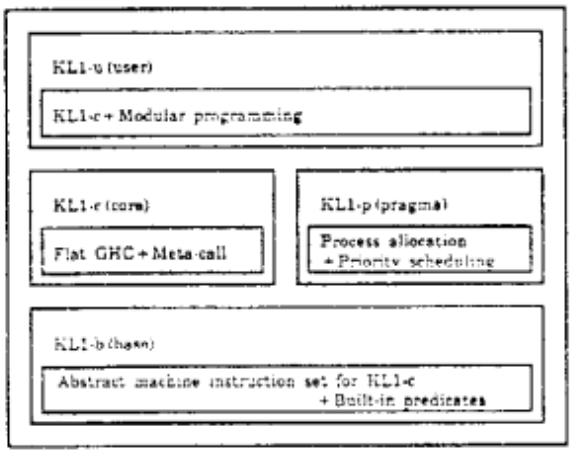


Figure 3 Hierarchical Structure of KL1

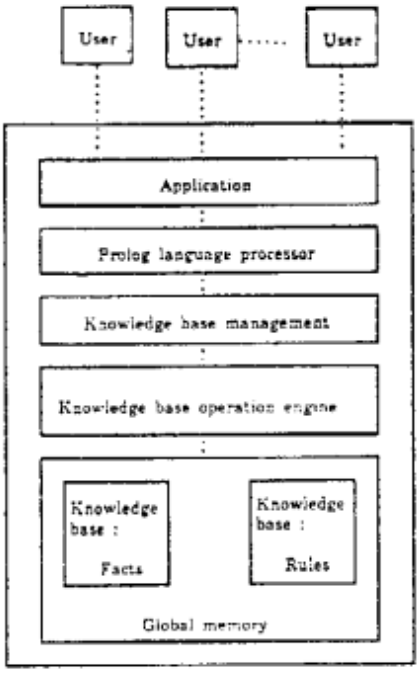


Figure 4 Conceptual Configuration of Pilot System

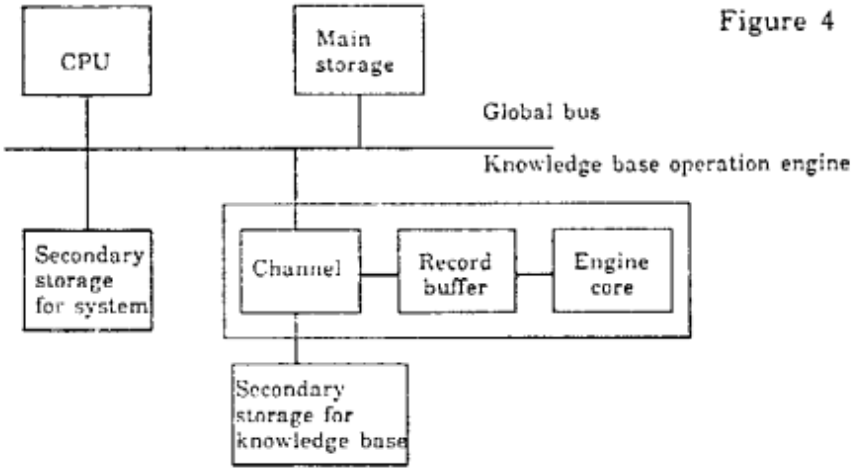


Figure 5 Outline of Hardware Structure

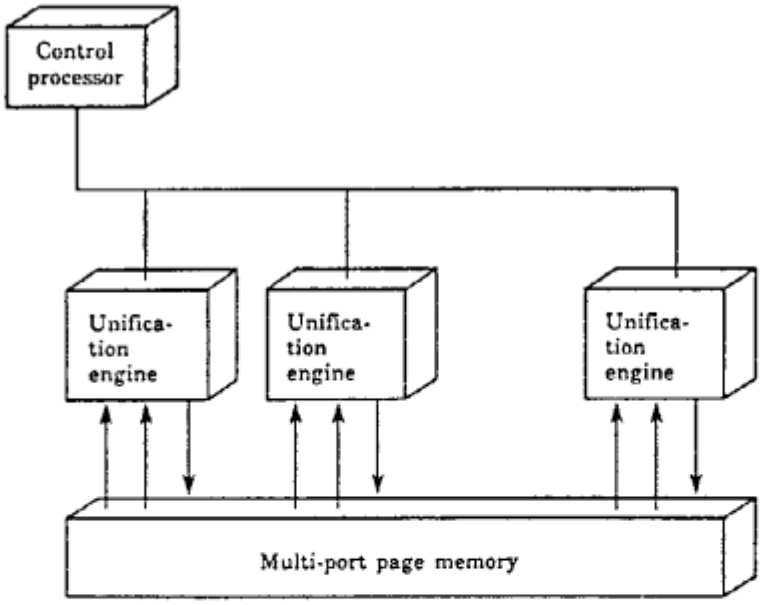


Figure 6 Configuration of Large-Scale Knowledge Base Machine

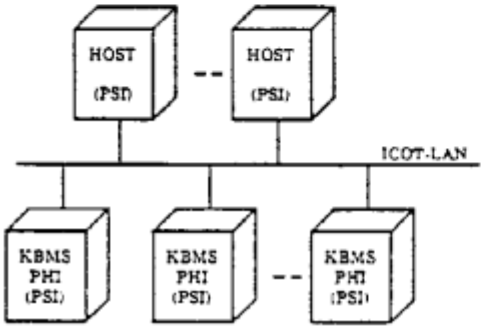


Figure 7 Configuration of PHI System

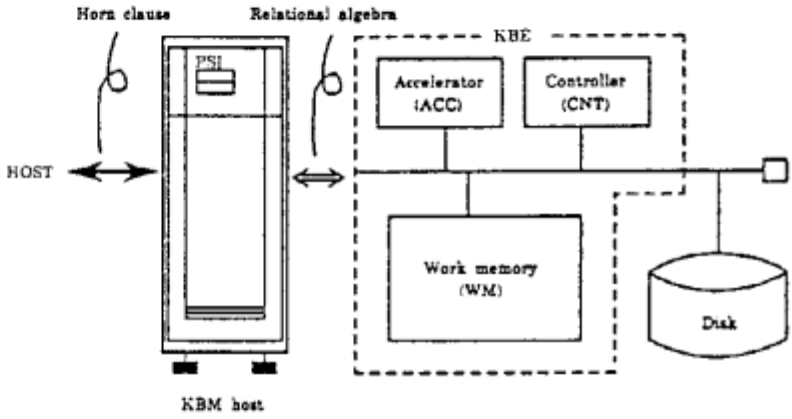


Figure 8 Structure of KBE

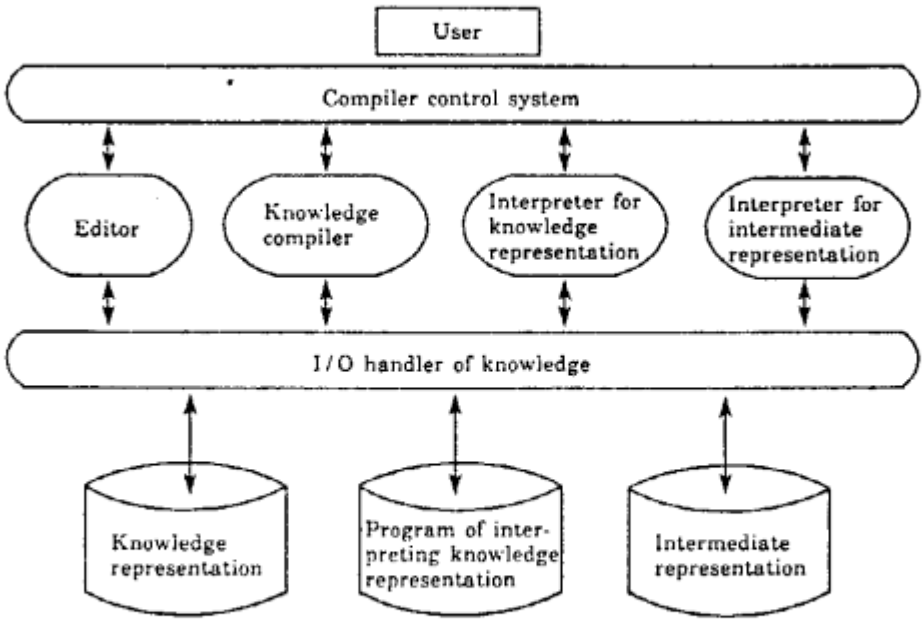


Figure 9 Total Image of Knowledge Compilation Control Model

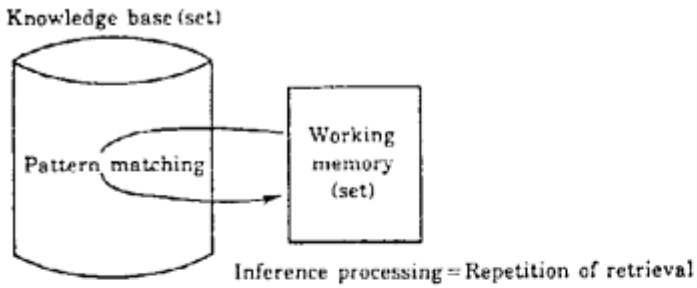


Figure 10 Knowledge Base Processing as a Set

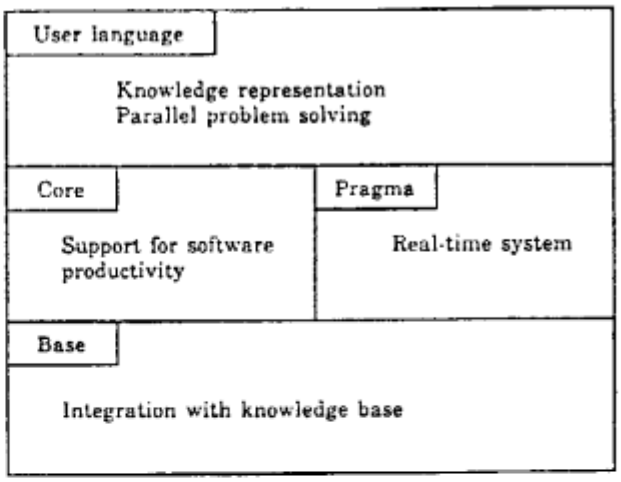


Figure 11 Various Aspects of KL2

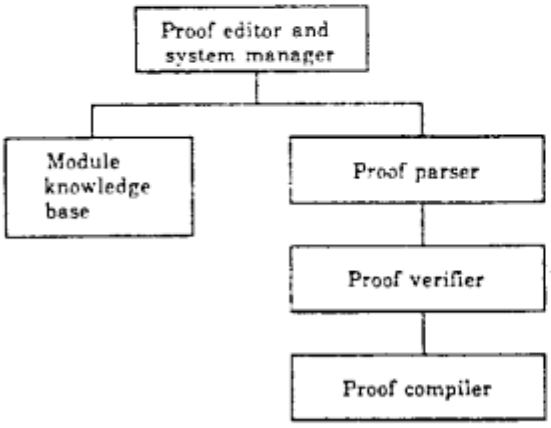


Figure 13 Outline of PCS

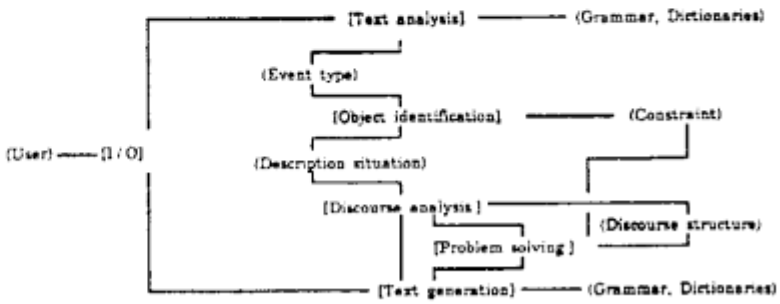


Figure 12 Structure of DUALS Version 2

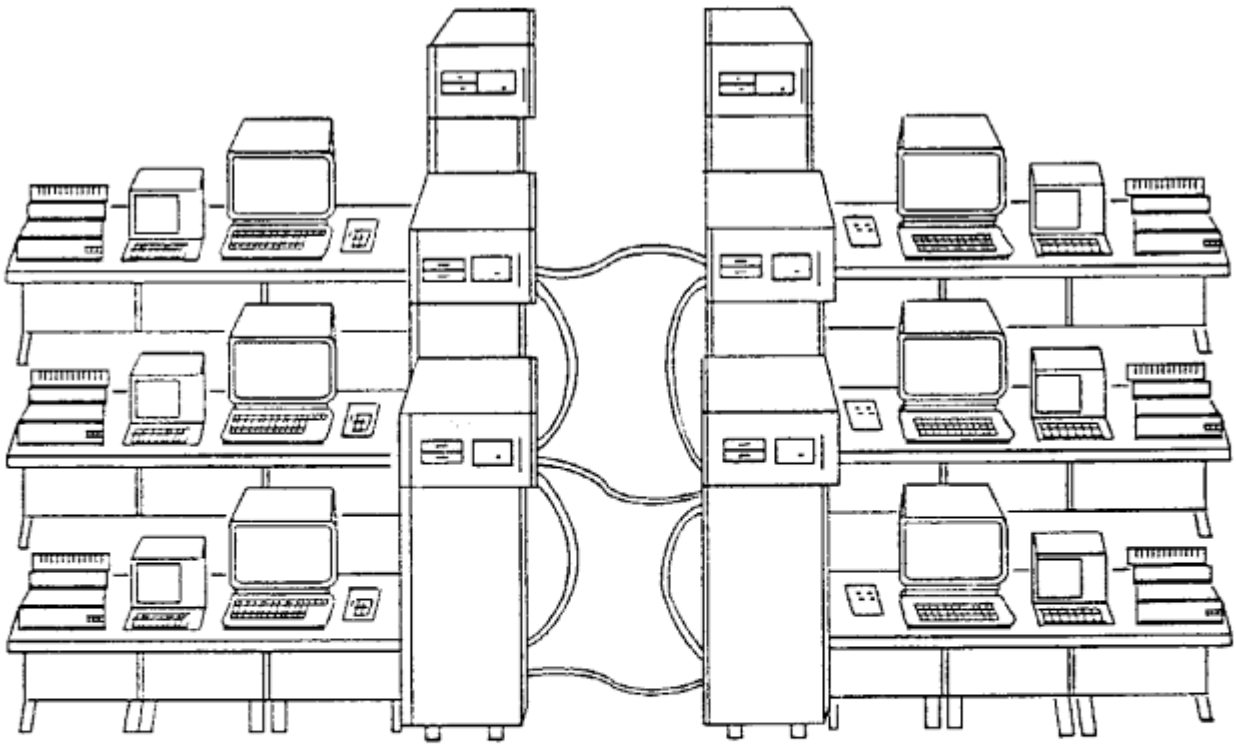


Figure 14 Configuration of Multi-PSI Vesion 1 (6 PEs)

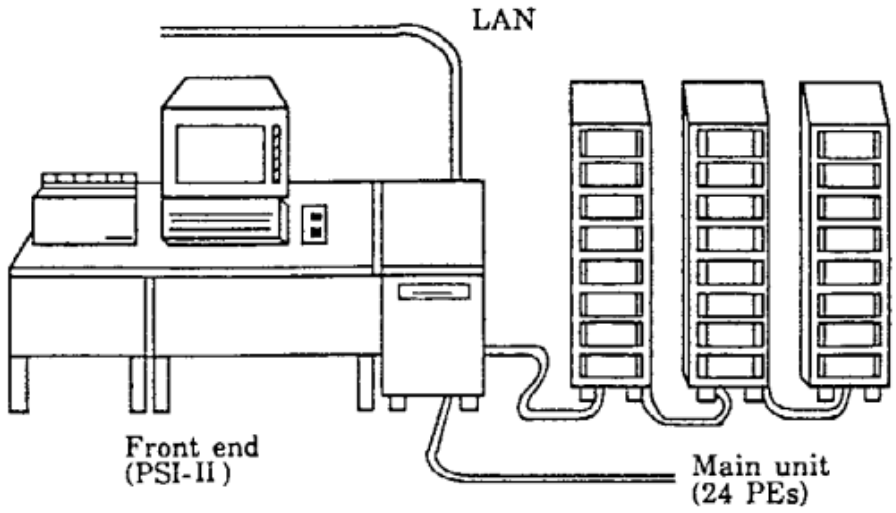


Figure 15 Configuration of Multi-PSI Version 2 (24 PEs)

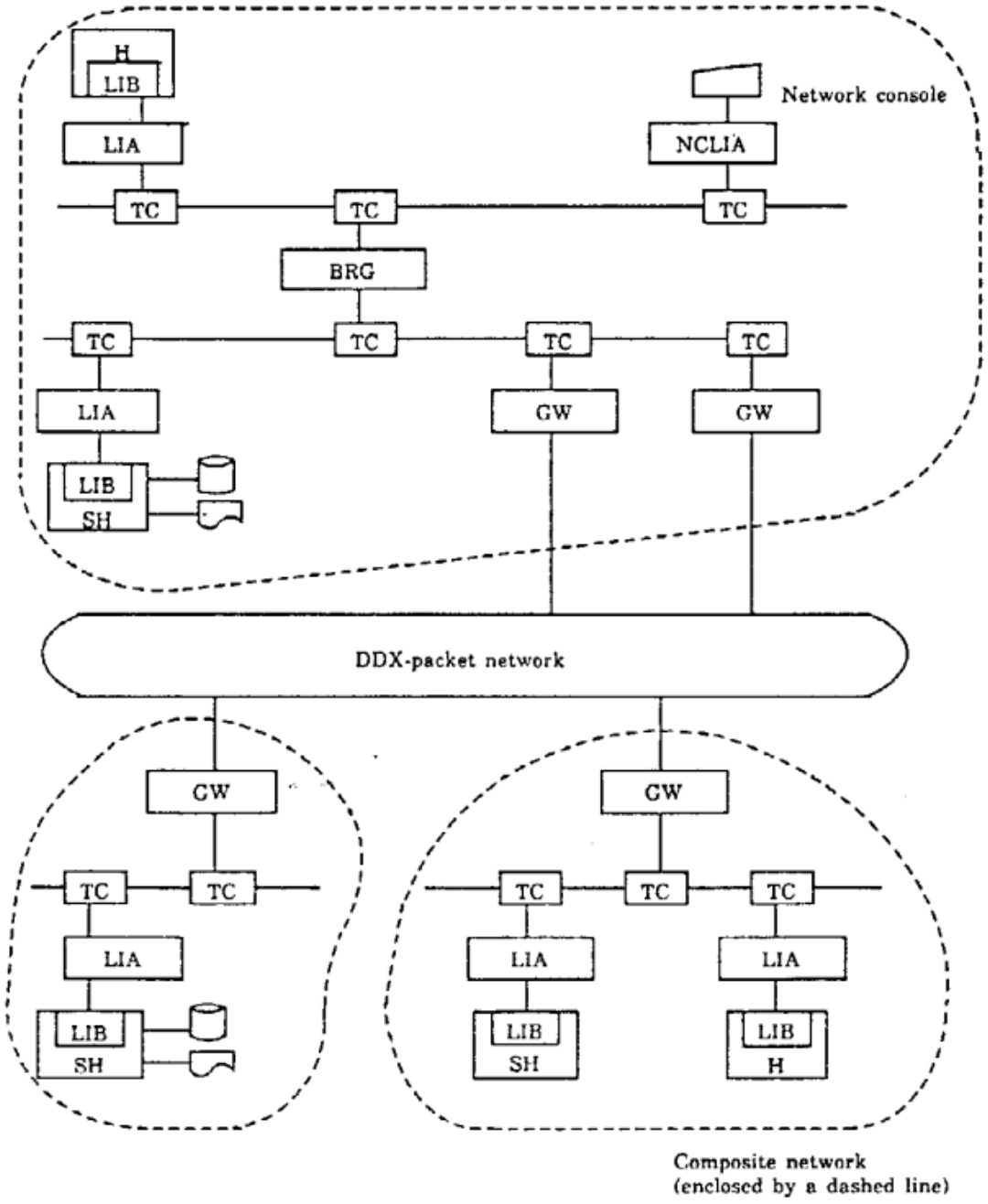


Figure 16 Entire Structure of SIM Network System