ICOT Technical Memorandum: TM-0117

TM-0117

A Knowledge Base Architecture and its Experimental Hardware

by
S. Shibayama, K. Iwata and H. Sakai
(Toshiba Corp.)

June, 1985

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

A Knowledge Base Architecture and its Experimental Hardware

Shigeki Shibayama, Kazuhide Iwata and Hiroshi Sakai

Information Systems Lab., Toshiba Research and Development Center

1, Komukai-Toshiba-cho, Saiwai-ku, Kawasaki, Japan

Abstract

This paper describes a scheme for manipulating multi-paradigm knowledge representations using an object-oriented concept and an experimental hardware prototype based on this scheme.

It is useful to be able to include various knowledge representation schemes in a system. For example, some kinds of knowledge are better represented using databases, logical formulae, or production systems. Usually, knowledge is concisely represented and efficiently processed by limiting the domains.

We define an object as an individual entity representing specific domain of knowledge with associated methods for the use of the knowledge. The knowledge base architecture we propose is an environment with objects possessing their own knowledge domains. The object methods have to determine both what the object can do and how to make the methods work.

Knowledge information processing in this knowledge base architecture is performed by a controller object that broadcasts a query and collects the responses from objects.

We are planning to build an experimental prototype for a knowledge base machine. The knowledge base machine contains various specific domain objects. It has a disk bank, a hardware stream filter, and an intelligent memory with a master processor. The hardware stream filter performs coarse filtering on the large capacity knowledge in the disk bank. The intelligent memory performs complex searching that cannot be done by the high-speed hardware filter. The intelligent memory consists of a controller processor and an array of local processors. Each local processor has its local memory. This group of the local memories is regarded as the core of the intelligent memory. The local memory is used for communication areas and for a work space to internally process messages sent from the master processor. Local memories are mapped in the master processor memory space, providing the master processor with the capability to collect the responses with reduced overheads. We will build an experimental system and develop knowledge base processing algorithms using this hardware.

1. Introduction

We have participated in Japan's FGCS (Fifth Generation Computer Systems) project and developed a dedicated relational database engine as the key component of the relational database machine Delta [Shibayama 84],[Kakuta 85]. Delta was developed to provide a research tool for the integration of the knowledge base and inference mechanism into the knowledge base machine [Murakami 83]. Delta is connected to the Personal Sequential Inference machines (PSIs) through a local area network (LAN) and logically interfaced by relational-algebra-level commands.

The PSI machines store programs written in a logic programming language and execute them in their main memory as long as the rule and fact sections of the programs are small and can fit into it. However, as the applications approach more and more to real situations, the number of the rules and facts increase dramatically. The knowledge base machine should be capable of manipulating and managing those rules and facts. The relational database machine stores the fact section of knowledge in tabular form. The database machine provides services on a vast amount of facts given in relations. However, this approach requiring that the fact section be stored in the relational database has some problems. It involves very many interactions between PSI and Delta, gaps between logic and the relational model, limited representation capability of knowledge, and so on. In order to overcome these problems, we have come to the conclusion that the database interface approach should be refined. We did not abandon the relational database as a useful container for a class of fact-based knowledge. We now regard the relational database as a component of our knowledge base approach.

In this paper, an object-oriented knowledge base scheme is presented in Section 2. In Section 3 we further discuss the application of the scheme to a local area network computing environment and to the knowledge base machine experimental hardware. In Section 4, we discuss the various knowledge base machine approaches. In Section 5, we describe the knowledge base machine experimental hardware. Section 6 presents the conclusion and future research plans.

2. An object-oriented knowledge base scheme

Various models to represent real world knowledge in computer-manipulable forms has been proposed. Some of them aim at constructing knowledge-based systems for specific areas. The most famous and successful of these are expert systems. The expert systems are based on a knowledge representation scheme known as the production system. Other knowledge representation schemes such as logic, semantic network, and frames produced fruitful results and contributed to knowledge representation research.

It is unlikely, however, that the real world knowledge can be efficiently represented, manipulated, and maintained using a sole knowledge representation scheme. There are several domains of human knowledge. Each domain of knowledge has a knowledge representation scheme that is most appropriate for that specific domain. Various knowledge-based systems have been implemented (commercially or experimentally) with their own problem domains and methodologies. We think of including various systems that have different knowledge representations into a total system.

The concept of the object-oriented computing paradigm was introduced recently and is recognized as a powerful methodology that can be applied to a wide range of problems. The object-oriented concept is mainly useful for data encapsulation and for highly modularized computing with communication between modules achieved by message-passing. The concept is applicable not only for specific language primitives but also for a wide range of system design level problems.

We adopted an object-oriented approach for a knowledge-based information manipulation scheme. The base concept we have for adopting this approach is cooperative problem solving. Cooperative problem solving requires a number of independent programs or systems which have self-contained functions. Such programs or systems are considered to be functions of the subsystems in the total system. Given a question that requires complex decomposition for a total system to answer it, first the question has to be manipulated by the many subsystems that make up the whole system.

The subsystems are considered as containers of various objects. An object is an active entity providing certain functions to other objects with an interface defined in a system-wide manner. For example, a relational database management system provides functions for accesses on its databases. It provides objects containing a database (a set of relations) and a set of associated database access functions to it (Fig. 1). Thus, each of the subsystems holds a number of objects. The objects are then hierarchically combined to form the higher-level objects. The system offers more intelligent (requiring more inferences) interfaces to the even higher-level objects.

In this scheme, we do not strictly define the languages or knowledge representations that describe the objects. Rather, the system, when implemented, will be like a distributed system possessing an integrated protocol with the capability of knowing what other subsystems can do, as well as informing those subsystems what it can do, using the protocol.

For example, if the total system is given a query such as, "What is the population density of Japan?", the query is represented in Prolog as:

```
population_density(japan, X) :-
  population(japan,XP), area(japan,XA), X is XP/XA.
```

We do not discuss the transformation of the natural language query into Prolog. It is done somewhere in a man-machine interface within the total system. The high-level object interprets the Prolog query by decomposing it into requirements for knowledge of the population and area of Japan. We do not assume an object-level Prolog interpreter here. If we did, query processing would proceed to find a

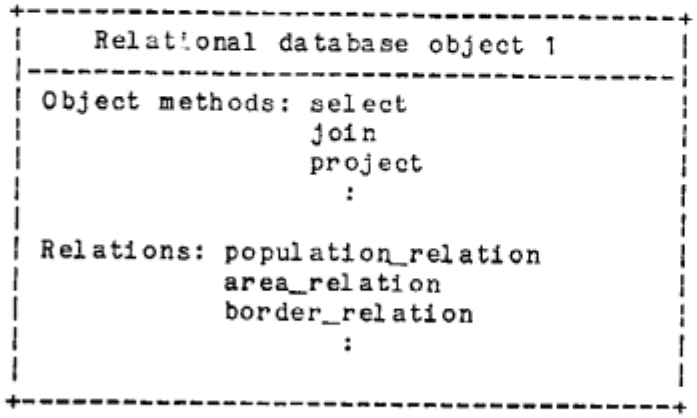

Fig. 1. A relational database object

population predicate in the processor's main memory. When it is not found, the system would simply return fail. (Closed World Assumption) If we have a Prolog/relational database interface interpreter (metalevel interpreter), the query will be interpreted to form a relational database query represented in SQL as:

```
select population
from population_relation
where country = Japan
```

and

```
select area
from area_relation
where country = Japan
```

(The relational database query could be different, by optimization or interpretation of the Prolog query.)

A system-wide object is needed which contains a directory as to what the objects in the total system provide as methods. In this case, the database management system (or a database machine) provides a relational database access interface. The top-level (query controller) object responsible for the query asks the directory object to know how to get the population and area of Japan. The directory object returns the object identifier corresponding to the relational database management object containing the information. The query controller object then sends a message to the relational database management object through the relational database access interface (Fig. 2).

The example shown here is simple, but the level of one method call can be arbitrarily high. It will reach the level of an integrated subquery, a decomposed section of the original query. If an expert system forms an object, a subquery that needs the expert system is sent to the object using the predefined object method call as interface.

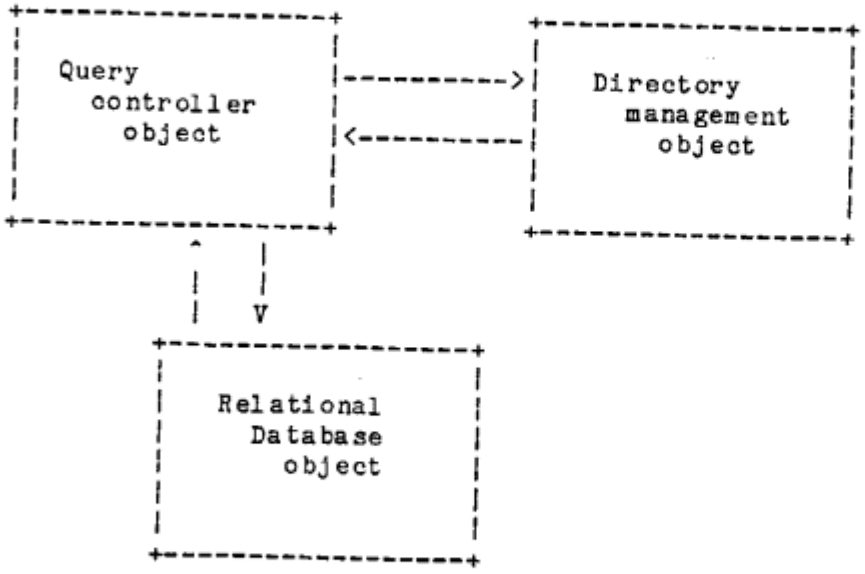

Fig. 2. Query processing by objects

In the above example, we assumed a single directory object for storing the system-wide directory. It can be easily imagined that centralizing the directory functions will cause a performance bottleneck. We can divide the directory to each subsystem. When each directory object encounters a query it cannot answer, it has to ask other directory objects distributed in other machines. The system-wide directory management is very much like a distributed database management. We can use distributed database techniques to manage the distributed directories.

3. Application of object-oriented knowledge base scheme to a distributed computing environment

When this scheme is applied to a network-oriented distributed computing environment, each subsystem is regarded as a container of objects providing some specific functions through a set of interfaces to the other subsystems. This helps to construct a unified network computing scheme. For example, there are many subsystems in ICOT's network-connected research environment. A number of PSIs and a relational database machine (Delta) have been developed and connected via a local area network. There are plans to add the HPM (High performance Prolog Machine) and PIM (Parallel Inference Machine) [Onai 85],[Ito 85] hardware components to the network environment. A global scheme to integrate them to form an experimental Knowledge Information Processing System is required to make them work in an organized way. In other words, a distributed operating system based on a network architecture that controls and manages the system components is needed. If each of the subsystems work independently, it is unlikely that the integrated system can handle wider ranging problems, however powerful each machine may be. Each subsystem is designed and (being) implemented to fit the scope of usage appropriate for them. The database machine is, for example, suited for handling (relational) database operations on data stored in relation form using stream crunching engines. The PSI is designed to provide a personal computing facility with a comfortable man-machine interface based on a bit-map display.

This knowledge base scheme is also related to the knowledge base machine architecture. In our concept, the knowledge base machine will store, process, and maintain data and relations from many areas of knowledge (Fig. 3). Each field of knowledge is better represented using knowledge representation schemes that most fit their efficient and natural description. If the knowledge is of a tabular form, it may well be stored in relations. Prolog facts are considered to be an example of this. However, it is meaningless to simply store all the facts in relation form. One of the advantages of storing facts in relation form is that thereby some operations on the stored facts can be reduced to relational database operations. When applying relational database operations on the stored facts, various hardware mechanisms and optimization methods treating the facts as relational tuples, can be used.

In this situation, a knowledge base architecture in a high-level sense controls an environment which contains many objects such as relational database objects as well as other objects with more self-contained computation units. The object methods in the knowledge base machine are supported using the hardware resources in the relational database section and knowledge base section. To support various object methods within the knowledge base machine, flexible system design utilizing specialized hardware and control software is

required.

4. Approaches to the knowledge base machine

The Prolog machines assume that logic programs are kept in main memory. A knowledge base machine is responsible for providing efficient and responsive storage of the rules and facts in external storage. Let us take a look at some possible approaches and discuss their characteristics.

4.1 Relational database approach

The idea of this approach is the combination of a relational database section and a Prolog section to construct a knowledge base machine. The relational database access sections are extracted from application programs and commands are issued to the relational database management system or relational database machines. Interface efficiency between the logic programming section and database section have to be fully taken into account in any designs using this approach. If the interface is slow, like the local area network connection, the command issue count should be kept low to avoid communication overhead associated with every interaction between the two sections. There is a technique called the compiled approach that collects the relational database accesses from Prolog programs [Yokota 84a]. It is a mechanism to reduce the interactions. In this approach, it is up to the interpreter (in a Prolog machine) to discriminate database accesses. It also has to control issuing commands, receive the results, and modify the obtained result from the database section as the answer to the application programs. There are major drawbacks in this approach. One is that if the communication speed between the sections is slow, processing speed is slow. Second, as the result from the database section is in relation form, that is, a flat collection of tuples, the read routine must transform this result with the Prolog machine's internal representation. This is partly because the relational database section is separated at the fact interface. The knowledge base and inference sections are better separated at the point where each of them completes meaningful computation units, not where facts and rules are discriminated syntactically. Third, although some operations are efficiently performed in the database section, they must be performed in the inference section because the database section only provides limited relational-algebra-based interface. Fourth, this approach does not utilize the benefits associated with the incorporation of relational databases. Efficient manipulation of sets is one of the features of the relational database. The Delta and Prolog combination, however, does not positively use the set manipulation concept. For these

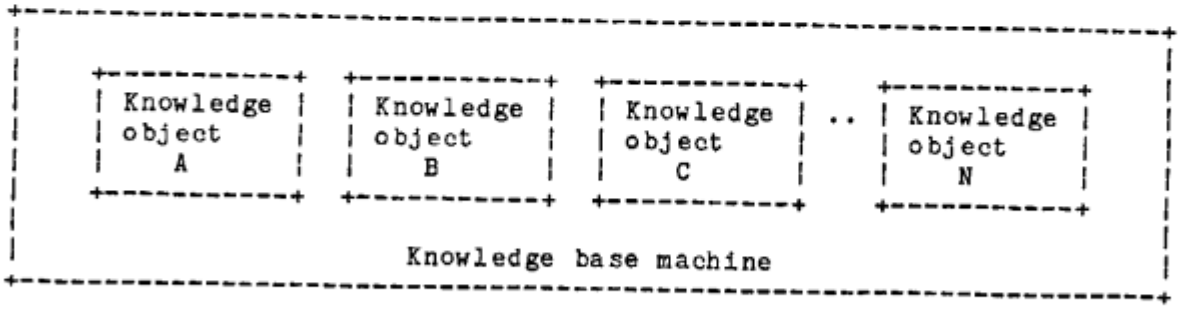

Fig. 3. A knowledge base machine model

reasons, it is unlikely that using simple Prolog with a relational database will be fruitful in knowledge base machine research.

4.2 Extended Relational Database Approach

The natural extension of the first approach is to enhance the interface (adding new commands to the database section) to increase the class of operations in the database section. This is considered to be a modification of the relational database system to match the logic programming environment efficiently and naturally. In other words, an inference mechanism is incorporated into the database section by the addition of certain commands. For example, a unification command can be added to the database [Yokota 84b]. This requires modification of relational database schemas to handle tags, store variables, and so forth. Yet these are not very drastic modifications and involve relatively little effort. In this approach, like the first approach, the interface is at the command level and it is still necessary for the host computer to generate the commands and transform the result.

Both of the above two approach uses command-based interfaces. The command-based interface is inefficient in most cases for the following reasons.

One is that the knowledge base machine is operated by the command sequence from the host machine. It is efficient to give the knowledge base machine some degree of freedom to control the order of fact or rule base access. Apparently, even if the compiled approach discriminates the database access section from Prolog programs, it will cause frequent accesses to the loosely-connected database machine. Another problem is that an integrated database management system (or a database machine) provides for the maintenance of database such as transaction control, security control, and concurrency control. Each time a command (sequence) is sent to the database machine, overhead accumulates and reduces system performance. Thus, it would be a good idea to give a macroscopic query representing a meaningful unit to the database machine and let the database machine to take care of the query, including optimization of the ordering of such as database access and command execution.

4.3 Prolog Interface Approach

The third approach is to set up the interface as a subset of Prolog. This is a departure from the command-oriented interface idea. Internally, of course, a knowledge base machine needs to manipulate on a very large fact or rule base. So, in spite of the change of interface, this approach does not imply any departure from database manipulation. There should be internal mechanisms to handle large amount of facts and rule bases. Providing predefined predicate calls of the Prolog subset is considered as the defining characteristic of object methods for knowledge base machine. This approach provides flexibility to the interface. When an application requiring knowledge base handling is found, it is not obvious where to draw a line between the knowledge base machine and inference machine. In other words, there are up to various different levels the knowledge base machine section should perform.

For example, the list manipulation can be a method for a knowledge base manipulation. For example, if there are two long lists in Prolog, call them a_long_list and another_long_list, we can write a Prolog program long_append for appending the two long lists. However, it will be time- and resource-consuming to execute it in a Prolog processor using backtracking. The semantics of list-appending is obvious if only one-way appending is needed. In this case we do not have to use a non-deterministic calculation. For example, we can represent the list as an array in the knowledge base machine, then the long_append is easily done by forming a new appended array. In other words, the knowledge base machine could be considered as a machine which features fast execution of extended builtin predicates. The builtin predicate set must be determined around various knowledge base manipulations to give a well-separated interface with the inference machine section. It provides powerful functions utilizing efficient storage strategy to handle data for storage in the knowledge base section on account of its large volume.

The simple long_append example will give the impression that the operation is too simple to call it a knowledge base machine. However, the level of the interface can be varied to answer a full subquery as long as it is self-contained within the knowledge base machine.

5. An experimental knowledge base machine

Knowledge stored in a knowledge base machine has two characteristics. It quickly becomes so large in quantity that it does not fit in the main memory. This means that the number of items holding pieces of knowledge becomes large. Secondary memory is needed to store the knowledge physically. Also, the knowledge items are interpreted to generate the result. This is considered as broadening the meaning of the inference process. Examples of the inference process are rule matches, pattern matches, and reductions. We think that two-layered knowledge base processing is appropriate to make the architecture match the two different types of knowledge processing.

For example, if the knowledge base contains a large number of Prolog clauses, preunification is effective for the first-level selection of clauses. Full unification on the candidates and the inference process that follows are second-level processing. In the case of intelligent text retrieval, the candidate text is first selected from the knowledge base using the keyword match. The candidate texts undergo a further examination by the more intelligent match against the criterion.

For our experience of the implementation of the relational database engine in the initial stage of the FGCS project, we believe that stream processing is a strong hardware computing methodology for crunching large amount of data in data streams. The data stream is a concept referring to a continuous physical chain of data flowing out from a buffer (possibly a disk cache), processed in pipeline fashion and returned to another (or the same) buffer. This is not efficient for processing queries that have a fixed pattern. In this case, some fine grain indices can be applied. However, to deal with general classes of queries, some hardware mechanism must be incorporated to perform brute-force data crunching. Stream processing can be applied to first-level filtering of knowledge items. Stream processing hardware is required to prevent bottlenecks in the data stream flow, because the speed of the stream flow is the key factor in this filtering stage. Hardwired logic is therefore a natural solution for

the filter.

Second-level of knowledge base processing includes operations involving more complex data handling than those which stream-oriented hardware can perform. In particular, the complex search problem (like production rule matching) requires several program steps to run. This is not simple pattern matching to be easily implemented in hardware but a computation that requires sophisticated algorithms.

We plan to build a hardware system for experimental knowledge base algorithms incorporating mechanisms for the two important aspects. The basic idea is to use different hardware component for the two aspects. The stream crunching task is to be performed using a stream-oriented engine and the complex tasks, such as flexible matching, in the intelligent memory section.

The hardware configuration is shown in Fig. 4. There are four major components. The intelligent memory is the section where second-level knowledge base manipulation is performed. This is explained in detail later. The relational database section is responsible for the relational database operations. We use the relational database engine and a large semiconductor memory to exploit the relational database operation processing technology adopted in Delta. The file management processor is responsible for managing the file system used in the machine. There are relational files for the relational database section and non-relational files for other knowledge representations. The disk bank stores the relational and non-relational files. A filter is associated with each disk in the bank. The filter is responsible for stream-oriented processing. It performs operations that are possible within a scan of the stream.

The intelligent memory has several features. The core of the

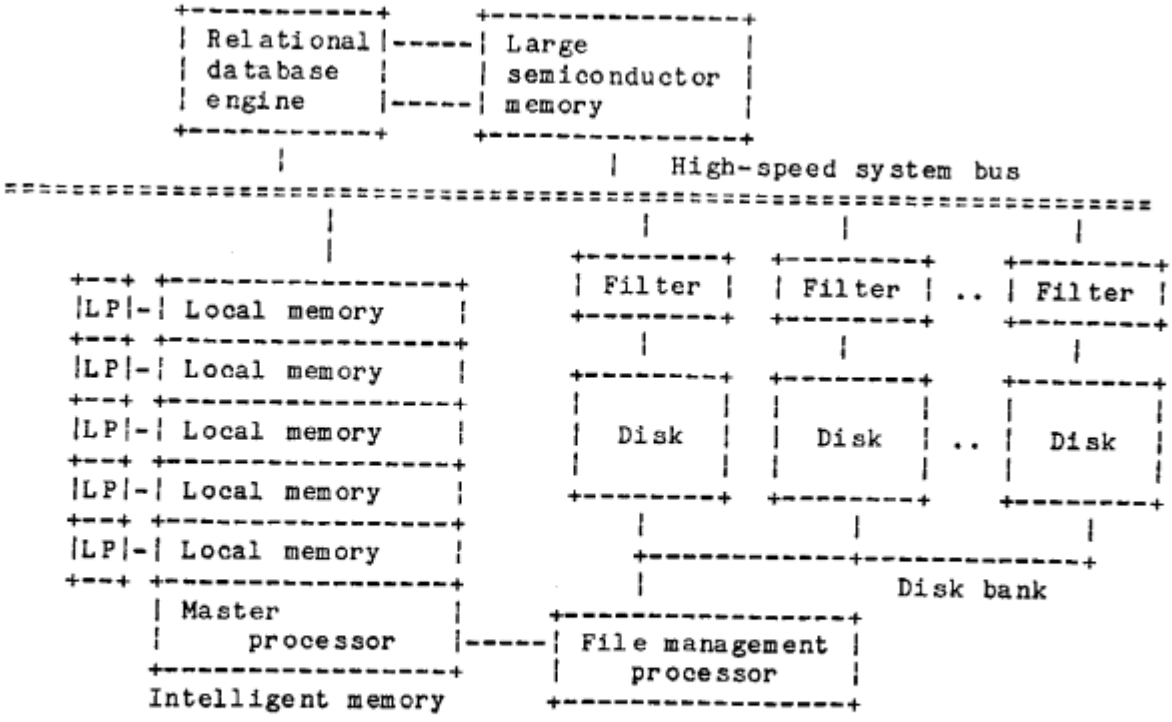

Fig. 4. Experimental knowledge base machine hardware configuration

hardware consists of an array of microprocessors (local processors). Each of the processors has its local memory of the order of a few megabyte. The two ends of each local memory are shared by two neighboring local processors (Fig. 5). The shared section is a "memory window" used for communication between neighboring local processors without data transfer. The memory window is derived from the register window idea of the RISC architecture [Patterson 82]. The register window is useful for expensive register saving/restoration and data communication in subroutine calls or process switching. We adopted this idea for the local memory for rapid data communication between a cascade of processors. We will also use these memory windows for experimenting with various knowledge base processing algorithms in a stream-processing-oriented way. In this case, the timing restriction to form a pipeline is looser than that of the disk filter.

The other feature of the configuration is that the local processors can work in parallel. This is useful for distributing problems which are parallel in nature and require more flexibility than strict stream processing. In production systems, for example, most of the performance bottlenecks are in the matching phase of the production rules. There have been proposals for executing production systems in dedicated hardware [Miranker 84]. Our machine aims at more general-purpose applications, but we think this architecture can perform productions with good performance. In our machine, we assume that the working memory is temporally redundant, that is, there are few updates to the working memory in a production cycle. So, we can broadcast copies of the working memory to all of the local processors and distribute production rules to the local processors. Each local processor has a section of production rules and a copy of working memory. In the match cycle of production system execution, all the processors run in parallel to discover matching rules. The controller microprocessor carries out selection serially by examining the flags raised by the local processors. Since the microprocessor performs the flag check by referring only to its main memory (another aspect of the local memory), sequential selection is not so time consuming. This phase would, however, be the bottleneck comparing this architecture with a production system machine. The act phase is done by

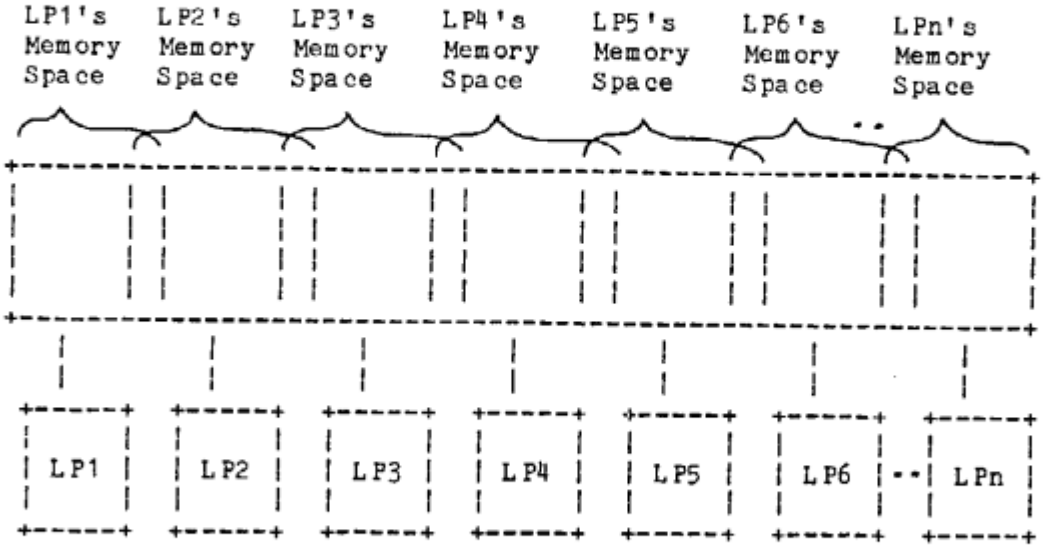

Fig. 5. Memory window

broadcasting the message to all the local processors to rewrite its working memory copies as specified. This is done in parallel, using the broadcasting function of the memory. So if working memory updates are not as frequent as we assumed, the act phase will not be time consuming.

The software structure of the master processor is shown in Fig. 6. The interface is a subset of Prolog. The details of the language is still under investigation. We intend to include set operation primitives as in the relational database interface and several builtin methods represented in predicate calls. The relational database management section is responsible for handling knowledge represented in relation form. The relational database engine developed in the initial stage is managed in this section. The other main section of the software is for parallel knowledge processing using the local processors. The parallelism is applied to application-oriented knowledge base manipulation, data structure transformation, and so on. The relational database management section is supported by a realtime monitor for responsive database access control. The general-purpose operating system provides the usual support mainly for software development tool.

6. Conclusion and future plans

We claimed that a global knowledge information processing system integration concept is necessary for a distributed computing environment and an object-oriented scheme is effective to loosely define the global architecture. From a knowledge base machine development viewpoint, a logic programming based interface provides more flexible and natural separation between the inference section and the knowledge base section than a command-based interface. Finally, we have described the experimental hardware and its software structure for use in the research of a knowledge base machine.

The detailed hardware architecture and interface language are under investigation. We plan to build the hardware after we complete those designs.

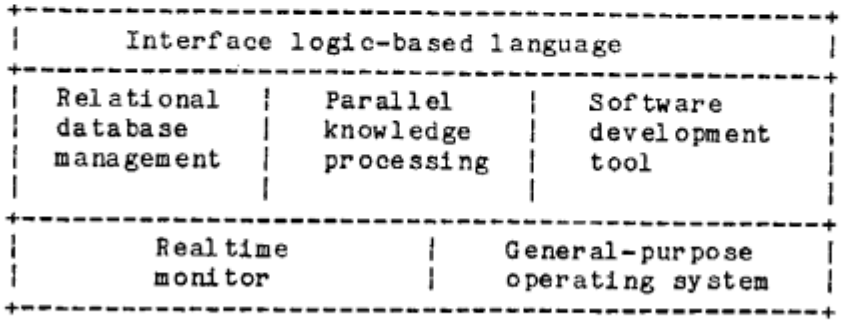

Fig. 6. Master processor software configuration

References

[Ito 85] Ito, N., Shimizu, H., Kishi, M., Kuno, E., Rokusawa, K., "Data-flow Based Execution Mechanism of Parallel and Concurrent Prolog", New Generation Computing, Vol. 3, No. 1, 1985.

[Kakuta 85] Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H., Murakami, K., "The Design and Implementation of Relational Database Machine Delta", in Proc. of 4th International Workshop on Database Machines, March, 1985.

[Miranker 84] Miranker, D. P., "Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE", in Proc. of the International Conference on Fifth Generation Computer Systems '84, Nov., 1984.

[Murakami 83] Murakami, K. et al., "A Relational Database Machine: First Step towards a Knowledge Base Machine, ICOT Technical Report TR-012, and also in Proc. of 10th International Symposium on Computer Architecture, 1983.

[Onai 85] Onai, R., Aso, M., Shimizu, H., Masuda, K., Matsumoto, A., "Architecture of a Reduction-Based Parallel Inference Machine: PIM-R", New Generation Computing, Vol. 3, No. 2, 1985.
[Patterson 82] Patterson, D. A., Sequin, C. H., "A VLSI RISC", Computer 15 (9), Sept., 1982.

[Shibayama 84] Shibayama, S., Kakuta, T., Miyazaki, N., Yokota, H., Murakami, K., "A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor", ICOT Technical Report TR-055 and also in New Generation Computing, Vol. 2, No. 2, 1984.

[Yokota 84a] Yokota, H., Kunifuji, S., Kakuta, T., Miyazaki, N., Shibayama, S., Murakami, K., "An Enhanced Inference Mechanism for Generating Relational Algebra Queries", Proc. 3rd ACM SIGACT-SIGMOD symposium on Principles of Database Systems, pp. 229 - 238, April, 1984.

[Yokota 84b] Yokota, H., Shibayama, S., Miyazaki, N., Kakuta, T., Murakami, K., "Unification in a Knowledge Base Machine", in Proc. 29th National Conference of Information Processing Society of Japan, 3F-2, 1984.