REPRINTED FROM:

MICROCOMPUTERS, USAGE AND DESIGN

eleventh EUROMICRO symposium on microprocessing and microprogramming

Brussels, 3–6 September, 1985

edited by

Klaus Waldschmidt
Bjørn Myhrhaug

PREPRINTS

1985

NORTH-HOLLAND – AMSTERDAM • NEW YORK • OXFORD

A Semantic Network Language Machine

Tetsuya Higuchi, Ken'ichi Handa, Tatsumi Furuya and Akio Kokubu

Electrotechnical Laboratory
1-1-4 Umezono, Sakura-mura, Niihari-gun, Ibaraki 305, Japan

This paper describes the IX system to support the development of knowledge information systems based on the semantic network representation. The IX system provides IXL, which is a knowledge representation language based on semantic networks, and the IX machine which is the parallel hardware dedicated to semantic network processing. IXL solves the problems of existing semantic network languages: the method of procedure addition and the description capability of relations themselves and negative knowledge. The knowledge representation method and the interpretation mechanism in IXL are discussed. As an example of IXL programming, French wine knowledge base is shown. The design of the IX machine is IXL oriented to treat efficiently both the declarative and the procedural knowledge. By directly mapping the semantic network structure onto the processing elements of the IX machine, the parallelism inherent in the semantic network processing are utilized to reduce the execution time. The IX machine consists of the pyramid-shaped packet switching network and processing elements with associative memories.

1. Introduction

The semantic network model in knowledge representation has been widely used in areas of artificial intelligence [1]. However, it is difficult to develop practical applications which use large semantic networks. There are two reasons for this:

(1) a lack of semantic network based languages which can flexibly describe various kinds of applications,

(2) a lack of computers which can utilize the parallelism peculiar to semantic network processing in order to reduce the processing time caused by the large size of the semantic networks.

The authors are developing the IX system (the pronunciation of 'IX' is [iks]) in order to solve these problems.

IX system provides,
(1) a semantic network oriented language for knowledge representation -> IXL([iksl]) [2][3], and
(2) massively parallel hardware dedicated to semantic network processing ->IX machine.

The objective of IX is to support all the processes of semantic network information processing in an integrated fashion: from using IXL in the modeling and description of semantic network applications (knowledge base,natural language processing, etc), to parallel execution of the IXL program with the IX machine.

The semantic network representation is suitable especially for the description of declarative knowledge. In order to support various kinds of knowledge information processing, however,it is essential that both declarative and procedural knowledge can be described and treated efficiently. The architecture of the IX machine and IXL are designed to meet this requirement.

This paper discusses the advantages of using IXL for the development of knowledge information systems and design considerations concerning the architecture of the IX machine.

In section 2, the main features of IXL are described. A graphic subsystem for the input and the debug of semantic networks is described in section 3. In section 4, a knowledge base system for French wine is described. In section 5, the architecture of the IX machine is discussed.

2. A Knowledge Representation Language IXL

The design philosophy of IXL is to solve the problems of existing semantic network languages: the description capability of relations themselves, the method of procedure addition, and the treatment of negative knowledge.

IXL can perform, (1) modification, (2) search, and (3) description of knowledge bases according to semantic network formalism. As shown in Fig. 1, IXL is a command language which looks like logic predicate.

Seven primitive links are provided to describe semantic networks. IXL is currently implemented in DEC-10 Prolog.

The main features of IXL are:
1) Relations in IXL are classified into basic hierarchical relations (is_a, instance_of) and general relations. General relations are further classified into assertion and property.

To construct a relation:

```
assertion(R,X,Y).
property(R,X,Y).
```

To inquire about a link:

```
isa(X,Y).
instance(X,Y).
ako(X,Y).
source(R,X).
destination(R,Y).
```

To inquire about a relation:

```
asst(R,X,Y).
prop(R,X,Y).
```

To connect nodes by a link:

```
link(is_a,X,Y).
link(not_isa,X,Y)
link(instance_of,X,Y).
link(not_instaceof,X,Y)
link(a_kind_of,[X,Y,..],Z).
link(source,R,X).
link(destination,R,Y).
link(rule,X,((
  asst(R,X,Y):-....
  prop(R,X,Y):-....
  isa(X,Y):-....
  instance(X,Y):-....)).
```

Primitive links:
is_a, not_isa
instance_of, not_instanceof
a_kind_of
source, destination

Meanings of the variables:
X,Y:concept node
R: relational node

Fig.1 Command list of IXL

2) A general relation is represented not by a link but by a node in order to precisely define the meaning of the relation.
3) Procedural knowledge is described in logic-like expressions.
4) Knowledge can be checked for inconsistency by the a_kind_of link (exclusive is_a link).
5) Negative knowledge can be explicitly described.

In the following sections, these features are discussed in detail.

2.1 Relations in IXL

Relations in IXL are classified into basic hierarchical relations supported by the system and general relations which are defined arbitrarily by the user according to application programs.

As basic hierarchical relations, "is_a" and "instance_of" links are supported. These are sufficient to describe hierarchical relations which appear in usual applications, although there exist many kinds of hierarchical relations. This is because the other hierarchical relations such as the part_of relation can be defined using procedure addition and the two basic hierarchical relations.

"Instance_of" is a link to connect two concepts in the case that one (i.e. "X") is the instanciation of the other (i.e. "Y"). In other words, X is at a lower abstraction level than Y. We call X the "instance" in regard to Y, and Y the "class" in regard to X.

There is also a hierarchy at the same abstraction level, and "is_a" is used to connect two concepts in the case that one (i.e. "XX") is a specialization of the other (i.e. "YY"), or the latter is a generalization of the former. Every instance of XX is also an instance of YY. Concepts such as XX/YY are called subclass/superclass.

General relations are divided into assertions and properties. When the user defines a general relation, the relation has to be declared to which the relation belongs.

When a concept X has a relationship R with Y, X is called the source of this relation and Y the destination.

A relation is called an *assertion* when the relation holds concerning the source concept. For example, if the user represents the knowledge that Mr. Smith is 25 years old, the semantic network representation may be that a concept "Mr. A" and a concept "25" are connected by a relation "age". The relation "age" is an assertion because "age" holds concerning the concept "Mr. Smith" itself.

A relation is called a *property* when the relation holds concerning all the instances of the source concept. If a concept "airplane" and a concept "fly" are connected by a relation "can", then a relation "can" holds concerning all the instances of the concept "airplane". Therefore, "can" is a property in this situation. However, if "can" is connected not to "airplane" but to "Clark Kent", "can" is called an assertion because "can", in this case, holds concerning only "Clark Kent".

A difference between assertion and property also appears in inheritance rules. An assertion is never inherited unless a procedure forces it. A property is inherited through an "is_a" link, which means some properties true with one concept also hold true with its subclass. A property is inherited through an "instance of" link in a different way.

Properties which are true with one concept are inherited by its instance as assertions.

2.2 Representation of Relations

In the early works on semantic networks, a general relation is represented as a mere link which just connects between two concepts. For example, "The vintage of Chablis wine is 1976" is represented as shown in Fig. 2.

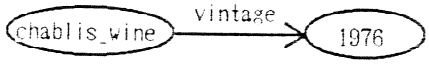

Fig.2 An example of semantic network representation

In this representation, however, the meaning of "vintage" can not be specified in detail. Therefore, in IXL, a general relation such as "vintage" is defined as a relational concept: a relation itself is also represented in terms of concepts (a network of concepts). As shown in Fig. 3, the user first defines the meaning of the relation using source and destination links. Then the knowledge "the vintage of Chablis wine is 1976" is instanciated using this definition as shown in Fig. 4. Three is_a links and the instanciation of "vintage" are newly added.

By representing a relation itself in terms of relational concepts, the meaning can be further defined in the network of relational concepts. For example, as shown in Fig. 5, the "vintage" relation can be defined as one of the wine properties.

Though the structure proposed here is similar to that of PSN [4], its meaning is more explicit in that a relation must hold concerning all the instances of a concept directed by a source link of the relation.

Fig.3 Definition of a relation in IXL

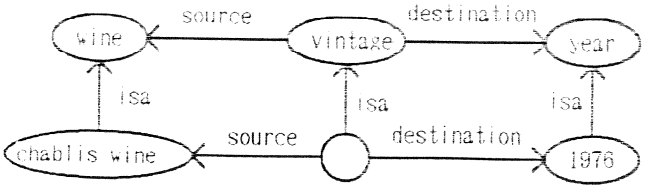

Fig.4 Instanciation of a relation in IXL

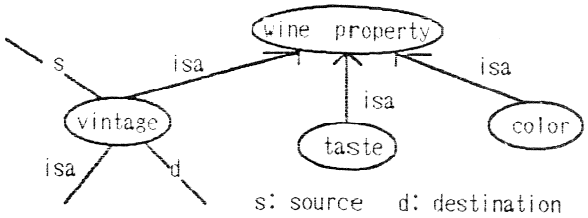

Fig.5 Representation of complicated relation

2.3 Procedure Addition

A procedure in knowledge representation is assumed to be a rule to infer the existence of a relation between some concepts. From this point of view, it is natural to express the procedure in a logic-like form. In IXL, Prolog is adopted to express procedural knowledge, but the IXL commands can also be written in the definition of procedural knowledge as well as Prolog predicates.

If procedural knowledge is described monotonously, we can not tell when to use it to infer some relation. Therefore, we added a procedure to a node which specifically concerns the relation.

The method for procedure addition is shown below. A "rule" link from a concept node indicates a procedural node which consists of several Prolog Horn clauses such as:

```
asst(R,X,Y):- ...
prop(R,X,Y):- ...
isa(X,Y):- ...
instance(X,Y):- ...
```

R is the relational node, X is the source of the relation and Y is the destination of the relation. Each Horn clause is activated when the corresponding query occurs.

Thus, procedure addition is effective in defining the meaning of a relation in detail because the logic-like form is suitable for describing inference rules. In addition, it is advantageous from the viewpoint of execution time of IXL programs that IXL commands can be used in the body of Horn clauses in order to define procedural knowledge, because IXL commands are executed in parallel by the IX hardware system. This means that both procedural and declarative knowledge can be efficiently executed by the IX hardware system.

2.4 Checking inconsistency of knowledge

Checking knowledge newly added to the base for inconsistency is essential to the development of knowledge bases. IXL provides two kinds of programming constructs for this: a_kind_of link and negation links (i.e. not_isa link, not_instanceof link).

The a_kind_of link is used to state that some concepts are exclusive of each other

concerning their superclass. For example, as shown in Fig. 6, "man" and "woman" are subclasses of "human" and they are exclusive of each other. If two concepts are exclusive, their subclasses and instances are exclusive of each other. Therefore, "Mr. A", which is an instance of "man", can not be an instance of "woman". In IXL, the user writes the following command to state this example:

link(a_kind_of, [man, woman], human).

If the user attempts to state that "Mr. A" is an instance of "woman", the system detects the inconsistency because of the presence of the a_kind_of link.

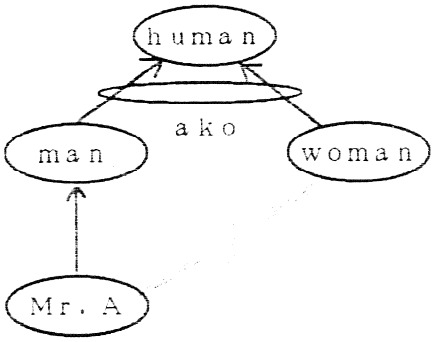

Fig. 6 Example of a_kind_of link

IXL provides "not isa" and "not_instanceof" to treat negative relations, and these are used to describe exceptions or constraints. For example, in Fig. 7, three knowledge statements are declared. The first knowledge states that a bird can fly. The second states that a penguin is a bird. Then it is deduced that a penguin can fly by these two statements, so, when the third statement "penguin can not fly" is entered, the IXL interpreter indicates an inconsistency. When this relation is forced, the third statement is represented in the semantic network by the not_isa link.

3. Graphic Subsystem

IX includes a graphic subsystem to facilitate the development of the knowledge base. In addition to the textual description by IXL, the user can define a knowledge base by drawing and debugging semantic networks directly on the graphic display.

Fig. 8 shows the contents of the screen. The command menu, working area, and built-in links are displayed. Semantic networks are drawn on the working area using a mouse. The networks drawn are converted into IXL format and transferred to the IXL interpreter.

The working area on the screen is a window that can be used to display any part of the large overall virtual working area. Therefore, the window can be moved, expanded, or reduced to view any necessary part of the semantic network.

This graphic system is also useful in debugging a knowledge base. When the textual representation in IXL treats an intricately tangled hierarchy, the graphic representation is especially helpful in grasping the interrelationships.

In addition, any part of the semantic network which the user may want to check in detail can be displayed using a color different from the rest of the network. For example, the nodes which belong to the same is_a hierarchy can be displayed in a different color to make them prominent.

```
ixl>property(can,bird,fly).
yes

ixl>link(is_a,penguin,bird).
yes

ixl>property(can,penguin,fly,
false).
It's inconsistent because.
  [isa(penguin,bird).
   prop(can,bird,fly,true)] ;
accepted
```

forcing this relation

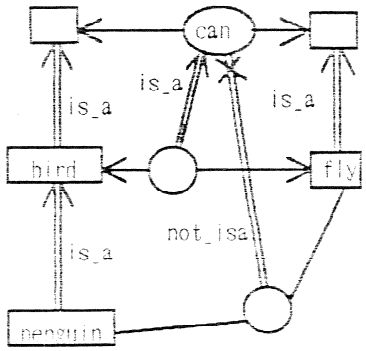

Fig. 7 Structure of property and negation

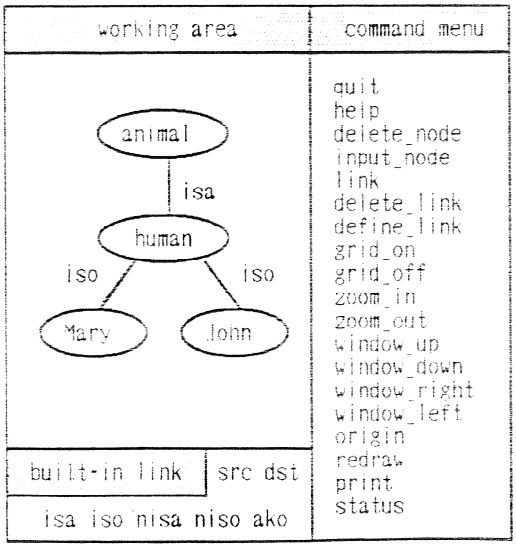

Fig.8. Screen image of graphic subsystem

4. Implementation of French wine knowledge base system

A knowledge base system was written in IXL to verify its descriptive capabilities. The system stores knowledge about French wines. This subject was chosen because such knowledge as types and production areas can be described naturally in a taxonomical manner, and so this subject is suitable for experimental representation in semantic network.

This knowledge base treats wine types, vintages, rankings, taste, colors, etc. Some procedural knowledge, such as the suitability for meals, is also included. The knowledge base consists of about 350 nodes which include 40 brands of wines in the Bourgogne area.

The query put to the knowledge base is translated into IXL search commands (i.e., asst(age,Mr.A,X), prop(vintage,chablis_wine, X), etc.).

The search strategy in the execution of IXL search commands is as follows:

(S1) The IXL interpreter searches for the answers which are explicitly described in the knowledge base. This process is the same as in usual database systems.

(S2) If S1 fails, the interpreter attempts to deduce the answers by examining the superclasses and by using inheritance rules.

(S3) If both S1 and S2 fail, the interpreter deduces the answers using the procedural knowledge written in Prolog and IXL commands.

The dialog in Fig. 9 is an example of the (S2) search. The query "Please describe the color of chablis wine" is translated into an IXL command, prop(has_color,chablis_wine,X). The command is executed assuming the semantic network in Fig.10.

```
| ?- question.
#Question > Please describe the color of
           chablis_wine.
#Answer > greeny_gold
yes
| ?- logging.
# inference rules
(ako,[chablis_grand_cru_wine,
      chablis_premier_cru_wine,
      petit_chablis_wine,
      chablis_wine],
     chablis_area_wine)
(has_color,chablis_area_wine,greeny_gold)
```

Fig. 9 Dialogue(1) with wine database

The execution process is as follows. The (S1) search fails because the fact about color is not described explicitly concerning the "chablis_wine" node in Fig.10. Therefore, the (S2) search is activated. The interpreter examines the superclass of the "chablis_wine" node and finds the "chablis_area_wine" node whose color is explicitly described as greeny gold. Therefore, the color of "chablis_wine" is deduced as greeny gold according to the inheritance rule. The rules used for the inference can be displayed for the user's convenience.

The dialog in Fig.11 shows an example of a search using procedural knowledge (i.e. type (S3)). The query "Does petit chablis wine go with oysters?" is translated into the IXL command prop(go_with, petit_chablis_wine, oyster), and the interpreter examines whether this command holds true or not.

The (S1) search fails along with the (S2) search, although it can be deduced using inheritance that petit chablis wine goes with food. Next, the search using procedural knowledge is activated. Procedural knowledge directed by a rule link from a "go_with" node is as follows:

```
prop(go_with,X,Y):-prop(go_with,X,W),
                    isa(Y,W).
```

The two predicates in the body are IXL commands. According to the query, X is "petit_chablis_wine" and Y "oyster". The first IXL command, prop(go_with, petit_chablis_wine, W), examines whether or not an instance of a relational node ("go_with") exists between a node (represented by variable W) and any node which is the superclass of "petit_chablis_wine".

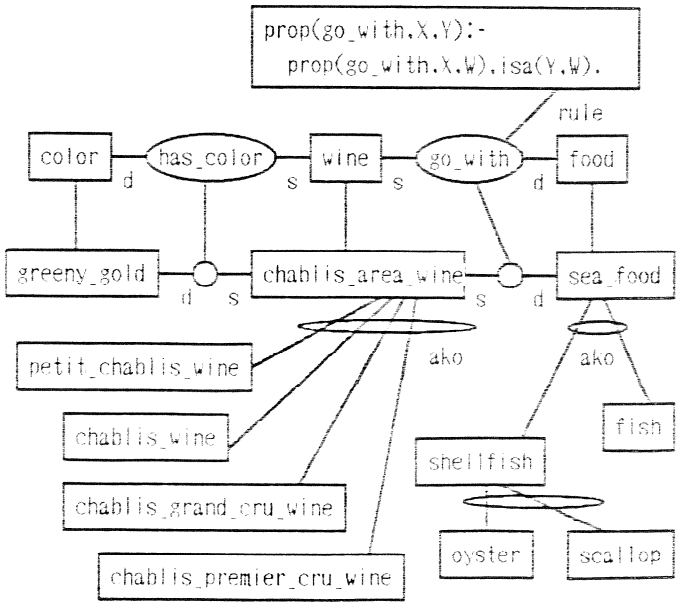

Fig. 10 An example of wine database

```
|?-question.
$Question > Does petit_chablis_wine go with
            oysters ?
$Answer >
yes
```

Fig.11 Dialog(2) with wine database

In this case, an instance of "go_with" exists at the "chablis_area_wine" node which is the superclass of "petit_chablis_wine". Therefore, assigning "sea food" to W, the first IXL command holds true.

The second IXL command, isa(cyster,W), holds true if nodes exist which are superclasses of "oyster". In this case, both "shellfish" and "sea food" can be W.

When both the first and the second IXL commands hold true, prop(go_with, petit_chablis_wine, cyster), which corresponds to the query, is proved to be true. Therefore, when "sea food" is assigned to W, the answer of the query becomes 'yes'.

The dialog in Fig. 12 is another example of type (S3). The query "Is Lyon to the south of Beaujolais ?" is put to the semantic network in Fig. 13. The IXL command translated from the query is:

asst(south_of,lyon,beaujolais).

Searches (S1) and (S2) are attempted in turn, but both of them fail. Then, the procedural knowledge attached to the "south_of" node is attempted. However, because there is no rule link from this node, the superclass of "south_of", which is the "relation" node, is examined. There, the following procedural knowledge exists:

```
asst(R,X,Y):- prop(inverse_of,R,RR),
              asst(RR,Y,X).
```

In this case, R is "south_of", X lyon, Y beaujolais. If the two IXL commands in the body hold true, the answer of the query becomes 'yes'.

The first command instanciated as prop(inverse_of,south_of,RR) holds true if there exists a relational node (i.e. RR) which is the inverse of "south_of". In order to hold true, the "south_of" node must be directed by the source link from the instance node (of the "inverse_of" node), and node (RR) has to be directed by the destination link.

```
|?-question.
$Question > Is Lyon to the south of Beaujolais ?
$Answer >
yes
```

Fig.12 Dialog(3) with wine database

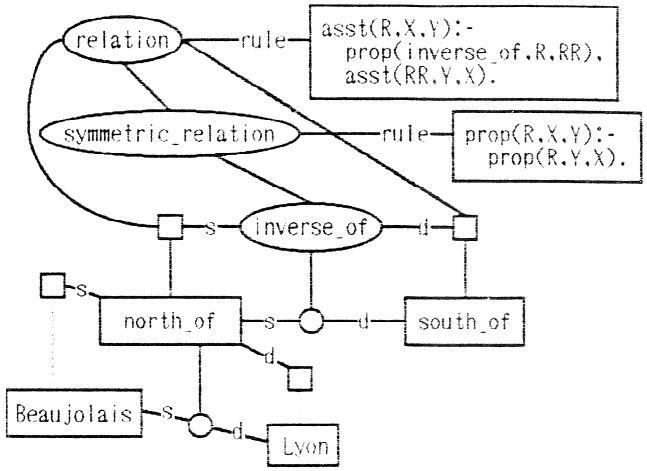

Fig. 13 Procedural knowledge in wine database

Actually, however, although "north_of" is described to be the inverse of "south_of" in Fig. 13, the "south_of" node is directed not by the source link but by the destination link. Therefore, prop(inverse_of, south_of, north_of) can not hold by (S1).

However, the procedural knowledge exists which is attached to the superclass of the "inverse_of" node. This procedural knowledge is "symmetric relation" which means that prop(inverse_of, south_of,north_of) holds true if "inverse_of" is a symmetric relation and prop(inverse_of, north_of ,south_of) holds true. Therefore, using this knowledge, prop(inverse_of, south_of, north_of) holds true because prop(inverse_of, south_of, north_of) is explicitly described in Fig. 13. Thus, the first IXL command in the body of knowledge at "relation" holds true, thus assigning "north_of" to a variable RR. Then, the second IXL command is attempted as asst(north_of,beaujolais,lyon). This command holds true because it is explicitly described in Fig. 13.

At this point, since the first and second IXL commands have been found true, asst(south_of, lyon, beaujolais) holds true and 'yes' is displayed.

5. IX hardware system design

5.1 Design considerations on IX machine

Von Neumann architecture is not suitable for the processing of a large semantic network because the parallelism peculiar to semantic network processing (such as marker propagation) can not be utilized.

However, approaches which attempt to map the semantic networks directly onto the collection of processing elements (PE's) have good promise [5][6]. Execution may be performed with the instruction cycles proportional to the depth of the network even in the case of large semantic networks. In the machines of this approach,

declarative knowledge is treated efficiently by utilizing the parallelism.

In order to describe and execute application programs of practical use, however, it is essential that procedural knowledge be also treated as efficiently as declarative knowledge. If only declarative knowledge is executed in parallel, the procedural part will become a processing bottleneck.

The main objective of IX hardware design is to process both types of knowledge efficiently. This can be realized by designing the architecture to execute IXL commands in parallel, because, in IXL, procedural knowledge is described easily in terms of IXL commands, which are also useful in the description of declarative knowledge. Therefore, the architecture design is IXL oriented in order to execute IXL commands fast.

The basic operations in semantic network processing for declarative knowledge are,

- association,
- set operations, and
- marker propagations.

The association and the set operation can be performed in SIMD mode on the assumption that semantic networks are directly mapped on the hardware.

The marker propagation is used to mark all the nodes in parallel which meet a particular condition. For example, the nodes which belong to an is_a hierarchy are marked in parallel by the marker propagation. As semantic networks often take on the shape of a tree, the marking starts at the root node of the tree and propagates downward in a radial manner. The necessary steps to mark all the nodes may be proportional to the depth of the tree. Thus, the marker propagation contains the parallelism peculiar to the semantic network processing.

In addition to the above basic three operations, the following operations are required in order to process the procedural knowledge:

- arithmetic operations,
- checking inconsistency of solutions, and
- creation and deletion of links.

Procedure addition in IXL permits the description of arithmetic expressions. For example, the following procedural knowledge contains an arithmetic expression in order to know the wine, X, of more than 300 Franc which goes with the food ,Y.

```
prop(go_with,X,Y):-
        prop(go_with,X,W),isa(Y,W),
        prop(price,X,T), T>300.
```

We have implemented the IXL interpreter in Prolog on DEC 2060. It determines a solution of an IXL command one at a time. On the other hand, the IXL interpreter written in IXL machine instructions, which will run on the IX machine, can determine a lot of solutions of an IXL command simultaneously, because semantic networks are directly mapped onto the IX hardware. In this viewpoint, the IX machine may be regarded as a parallel logic machine.

However, the solutions of the IXL commands which appear in a clause must be consistent. For example, the solutions of X of the first IXL command (i.e. prop(go_with,X,W)), should be the same as the solutions of X in the third IXL command. Thus, the checking of the inconsistency of the solutions must be made in the processing of the procedural knowledge.

After the solutions are determined, new links are established between the nodes. In the example above, new links are established between the nodes of X and the nodes of Y.

IX machine architecture is designed to process above six operations effectively.

5.2 Overview of the IX machine

Fig. 14 shows the overview of the IX machine. Its main components are the pyramid-shaped packet switching network and the processing elements with associative memories. Fig. 15 is shown as seen from above the machine.

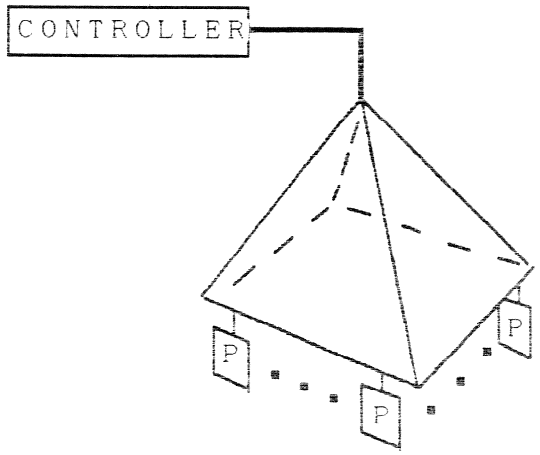

Fig. 14 Overview of the IX machine

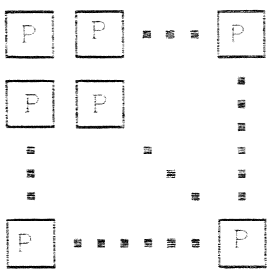

Fig. 15 Arrangement of PEs (ground plan)

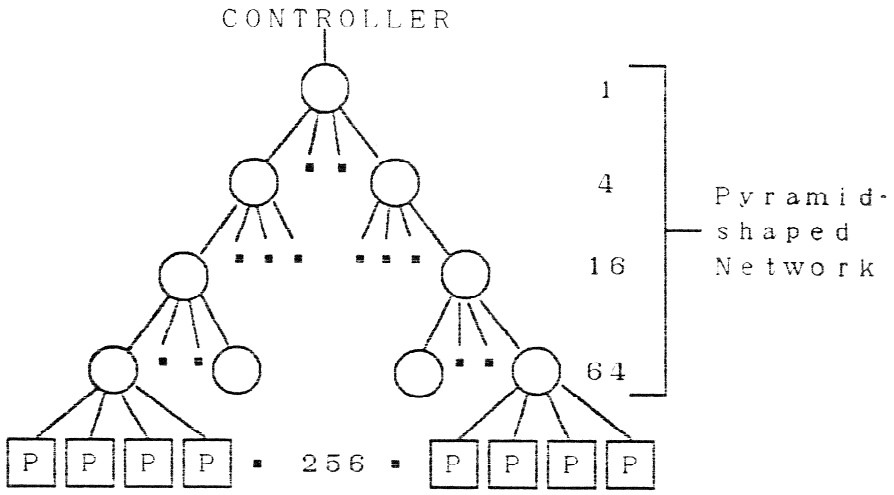

Fig. 16 The cross section of the IX machine

The cross section of the IX machine is shown in Fig. 16. The node processor of the bottom layer of the pyramid-shaped network connects four processing elements. Each upper node processor of the pyramid-shaped network is connected to four lower node processors of the network. Therefore, if the processing elements are arranged in 16 X 16, then the bottom layer of the network consists of 64 node processors. The second layer from the bottom is 16 node processors. The top node processor is connected to the host controller.

The execution of the IXL command proceeds as follows (Fig. 17). At first, an IXL command is broadcast from the controller. The command is executed in each processing element in SIMD mode.

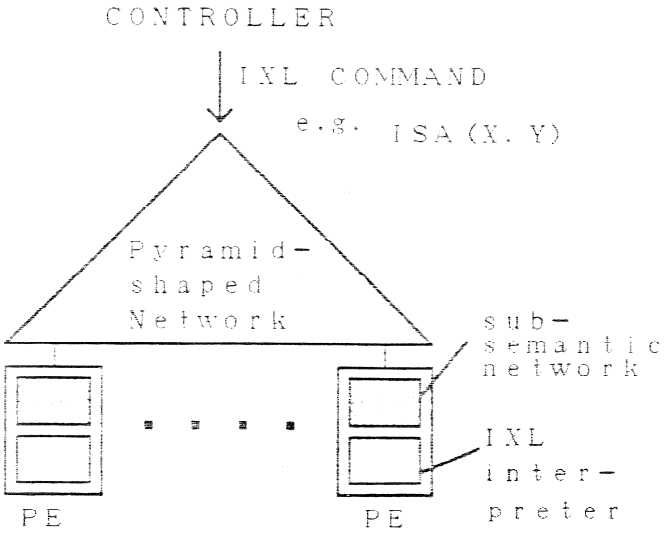

Fig. 17 Execution of anIXL command

A large semantic network is divided into sub-networks and distributed among processing elements.

The IXL interpreter, that is a collection of subroutines written in the IX machine instruction set, is also located in each processing element.

5.3 Processing element

The processing element consists of three main components:

1. the semantic network associative memory (SNAM)
2. the instruction associative memory (IRAM)
3. the control processor (CP).

SNAM is the storage for the divided semantic network (sub-network). The basic semantic network operations such as association and set operations are executed in parallel utilizing this associative memory.

Fig. 18 shows the organization of SNAM and how the sub-networks are stored in SNAMs. The semantic network in the figure is divided into two sub-networks and stored in two processing elements. Notice that a link in the semantic network occupies two words of SNAM. Therefore, we only have to add two words in order to establish a new link. The overhead is very little, compared with the circuit switching architecture.

IRAM is the storage of the IX machine instructions for the IXL interpreter. The reason why IRAM adopts an associative memory is that some of the IX machine instructions are asynchrounous. For example, the execution of the instructions for marker propagations is initiated whenever the input marker arrives at the processing elements. The associative memory is used to fetch the executable instructions.

CP treats the procedure execution which is enevitable for practical applications. Arithmetic expressions are processed by CP.

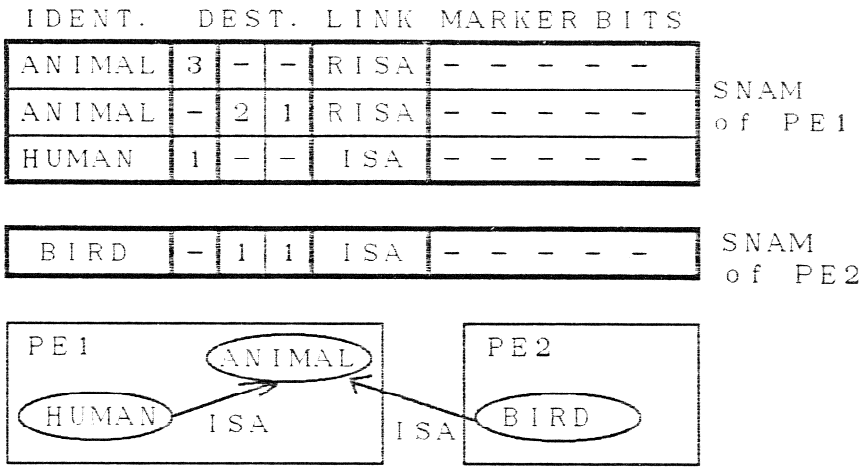

Fig. 18 The division of the semantic network and the assignment onto SNAMs.

CP also controls the marker propagations between nodes.

5.4 Pyramid-shaped network

The pyramid-shaped network plays three roles. The first one is the packet communications between processing elements. It is used mainly for the marker propagation. The second one is the broadcasting of IXL commands and their arguments from the host controller. The third one is the equivalence node function.

The equivalence node function have been introduced to treat with the nodes which have too many connection links. As mentioned earlier, a link in the semantic network occupies two words of SNAM. Therefore, if a node has too many links, many words of SNAM are required and the number of necessary words may exceed the size of SNAM. So, we chose a method where such words are distributed among processing elements.

However, the words distributed among processing elements represent one identity. So we call them equivalence nodes. If a packet such as marker propagation is delivered to one of the equivalence nodes, the packet must be copied and sent to the other members of the equivalence nodes. The role of the pyramid-shaped network is the duplication of the packet and the transfer of them.

For example, in Fig. 19, the hatched processing elements (i.e. B,C, and D) contain equivalence nodes. They contain words of SNAMs whose identities are all the same. If a packet comes from the processing element A and the destination of the packet is the processing element B, the packet is copied by the node processors of the pyramid-shaped network and sent to the other members of the equivalence nodes such as C and D.

In order to realize these functions, each node processor of the pyramid-shaped network contains an associative memory and the routing control logic. The associative memory is used to look up the members of quivalence nodes quickly.

6. Conclusion

This paper described an overview of the IX system which supports the semantic network based applications from the description to the parallel execution.

The knowledge representation language IXL is effective in describing the meaning of the relationship itself between entities, procedural knowledge and negative knowledge. Through the experience of French wine knowledge base, it is found that the facility of procedure addition is very useful in the description of practical applications. For example, complicated

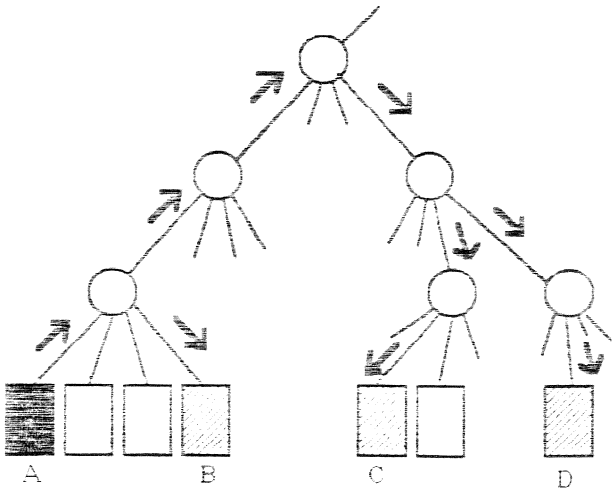

Fig.19 Packet transfer using equivalence node

inference rules or exception handling can be flexibly described.

In order to execute IXL programs efficiently, the IX machine is being designed IXL oriented. The most outstanding feature of the IX machine is that the IX machine can execute in parallel not only declarative knowledge but also procedural knowledge. The distinctive features of the architecture are the pyramid-shaped network and the processing element with associative memories. The pyramid-shaped network can treat equivalence nodes which are often required in the practical applications using large semantic networks. The processing element adopts associative memories to establish new connections easily and to perform basic semantic network operations in parallel. We are currently designing the details of this architecture.

Problems still exist with IXL such as how to treat sets, quantification, and grouping of knowledge. These are problems not only with IXL but with any other knowledge representation languages. Further research is required on these subjects.

References:

[1] M.R. Quillian: "Semantic Memory", Report AFCRL-66-189,BBN, Cambridge,Mass.,1966.

[2] K. Handa, T. Higuchi, T. Furuya, and A.Kokubu: "Semantic Memory System IX - Knowledge Representation in IXL", WGAI Report of IPSJ 39-7,pp49-56, Mar. 1985 (in Japanese).

[3] K. Handa, T. Higuchi, A. Kokubu, and T.Furuya : "IXL: A Flexible Semantic Network Language", (submitted to Journal of Information Processing).

[4] H.J. Levesque: " A procedural approach to semantic networks", TR-105,Univ. of Tronto,1977.

[5] S.E. Fahlman : " Design sketch for a million NETL machine", Proc. of First Annual National Conf. on AI,1980.

[6] G.E. Hillis: "Connection machine", TR-646,MIT,1981.