Inference engine for expert system by using optical array logic

Masaya Iwata, Jun Tanida, and Yoshiki Ichioka

A method of implementing an inference engine for an expert system by using optical array logic is presented. Optical array logic is a technique for parallel neighborhood operations based on spatial coding and two-dimensional correlation. The processing capability of optical array logic is useful for the efficient execution of an inference operation in artificial intelligence problems. Inference proceeds with the help of a token propagation technique in which the knowledge is expressed as spatial patterns and processed collectively with parallel neighborhood operations in optical array logic. With several additional functions, an expert system can be constructed by using the inference technique.

Key words: Digital optical computing, optical array logic, artificial intelligence, inference engine, expert system.

1. Introduction

Optical parallel computing systems are capable of massive parallel processing and of making good use of the inherent parallelism and the high speed of light propagation. The systems are expected to solve problems in massive information processing. Therefore various studies on system architecture and computational algorithms in optical computing have been reported.[1–3]

Optical array logic (OAL) is proposed as a technique for achieving a parallel neighborhood operation for two binary images.[4,5] The advantages of OAL are its capability of parallel logic operations, its programmability with an operation kernel, and its suitability for optical parallel processing. Several optical systems have been developed to implement OAL.[6,7] On these optical systems, various single-instruction stream–multiple-data stream (SIMD) types of parallel processing can be achieved with OAL. Therefore studies of parallel processing with OAL programming are important subjects for exploring the capability of OAL processors. Image and numerical processing have been studied[4–8] on the basis of this principle.

Artificial intelligence[9] (AI) is one of the research fields in which massive parallel processing is strongly desired. The main process in the AI problem is inference. Inference is an operation that searches specific data from a knowledge base, and usually it must handle a large amount of data. In a practical system, such as an expert system,[10] this inference operation determines the total performance of the system. Thus it is important to develop an efficient searching method for AI problems.

Recently several methods have been proposed for a data search that utilize the parallelism in optics.[11–15] In some methods, knowledge data are represented in an image, and inference is achieved by SIMD types of parallel operation such as matrix operations[12] and matched filtering.[13] Since these operations are effectively executed by optical systems, a high throughput of inference can be expected. We also found that OAL is useful for parallel data search in inference. In Ref. 14 we reported an inference engine that uses a template matching technique based on OAL. However, these methods lack flexibility and, hence, cannot achieve minute operations for data. To overcome these shortcomings, there is an approach to map knowledge data directly onto an optoelectronic multiprocessor system.[15] In this method, inference is executed by marker propagation among processing elements with an optical interconnection. However, the parallelism of light is not fully utilized in this method.

In this paper we present a method of implementing an inference engine with a token propagation technique that is a useful technique in OAL programming. This method can fully extract parallelism in optical processing and can be executed by an OAL processor. The flexibility of this method is demonstrated by an

The authors are with the Department of Applied Physics, Faculty of Engineering, Osaka University, 2-1 Yamadaoka, Suita 565, Japan.

Received 20 May 1991.
0003-6935/92/265604-10$05.00/0.

expert system that uses the inference engine. In Section 2 we briefly explain OAL and a specific operation used in an inference process. In Section 3 we describe an inference process based on a semantic network as knowledge representation. In Section 4 we explain the concept of token propagation and we discuss a method of implementing a primitive inference engine by using the technique. In Section 5 we describe an inference engine for an expert system, and in Section 6 we estimate the processing efficiency of the presented inference engine.

2. Optical Array Logic and Template Matching for Two Images

OAL is a technique for executing logic operations for a pixel and their neighboring pixels on two-dimensional (2-D) binary images in parallel. The processing procedure of OAL is shown in Fig. 1. Two binary images are inputs on which all pair of pixels at the same location are converted into spatial patterns by the coding rule, which is shown in the Fig. 1. The resultant image, which is called a coded image, is correlated with an operation kernel configuring operation of OAL. The operation kernel is a set of delta functions positioned at some grid points in which the interval of the grid is equal to the cell size of the coded image. Therefore the correlation is regarded as a sequence of operations: duplicating the coded image, and then shifting and overlapping these images.

The correlated image is spatially sampled at the upper left pixels of cells on the input images. As a decoding process, the collection of the sampled pixels is dilated and inverted. By the above sequence, we can execute a product term operation that is a range of logic operations expressed by the logic product of pixel values. To extend the flexibility of the operation, multiple product term operations are executed with different operation kernels, and their results are combined by a logic sum. As a result, OAL is capable of executing an arbitrary neighborhood logic operation with a combination of operation kernels used in correlation. These operations are executed in parallel with a simple optical correlator, as shown in Fig. 2. Thus an SIMD-type of parallel neighborhood operation can be achieved by OAL.

Parallel processing, as described by OAL, can be optically executed on a general-purpose optical processing system called an OPALS (optical parallel array logic system). Figure 3 shows a block diagram of a hybrid version of the OPALS.[8] OAL is considered as software (program) for controlling such an OPALS system. The following is addressed to programming techniques for the OPALS.

Programming OAL is achieved by specifying a set of operation kernels used in correlation. For programming convenience, we have developed kernel expressions[6,7] that can denote a sequence of operation kernels. A kernel expression is an intuitive expression of logic operations for 2-D images. Thus, if a problem is given as a logic relation between spatial patterns, it can be easily described by a kernel expression. Once a kernel expression is determined, we can convert it into operation kernels used in OAL and execute on the OPALS. The notation of the kernel expression is summarized briefly in appendix A. The operation kernels designed in this study are listed in appendix B.

As an example of OAL, we explain a useful operation in data search for an inference operation. The operation is basically logic template matching, but it operates the patterns in images A and B at the same time. In this operation, specific positions in which the combination of the pixels in two images are identical to the predetermined patterns are detected.

The operation is achieved by a parallel AND operation for neighborhood pixels in the two images. For example, to detect occurrences in which pixels with 1 and 0 are arranged side by side in the corresponding position of images A and B, the following operation is used:

$$c_{i,j} = a_{i,j}\bar{a}_{i,j+1}b_{i,j}\bar{b}_{i,j+1}, \tag{1}$$

where a, b, and c mean pixel data in images A, B, and C, respectively; subscripts denote the location of the data in the image. A kernel expression of Eq. (1) is shown in appendix B. Figure 4 shows the flow of the operation.

In OAL, one input image can be assigned to control signals for the data primitives located in the other

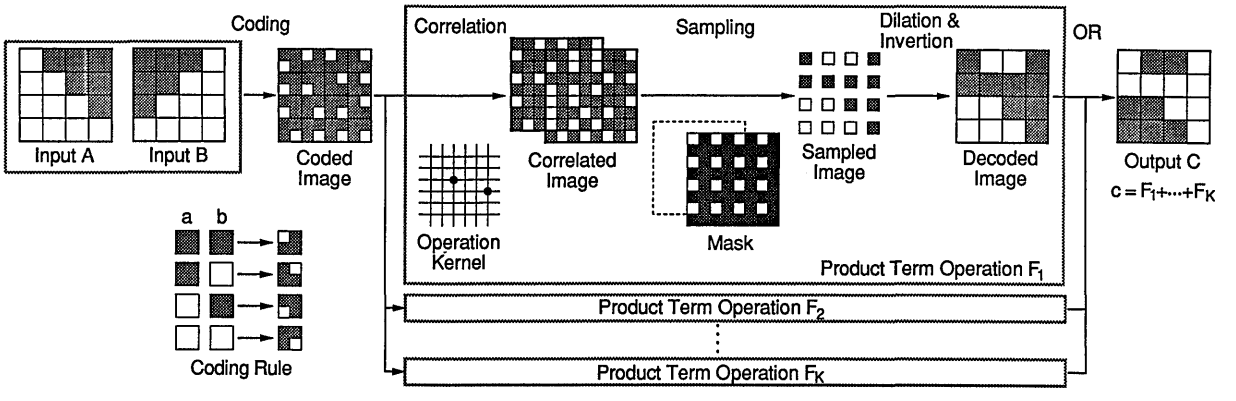

Fig. 1. Processing procedure of OAL.

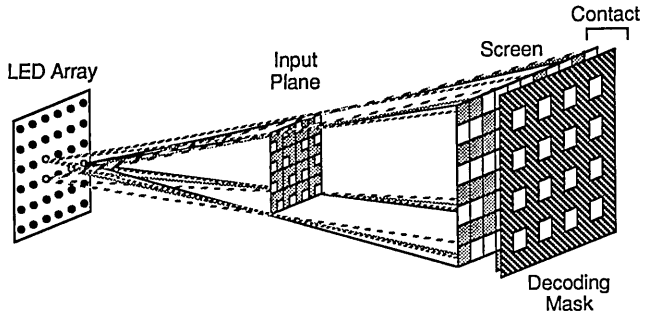

Fig. 2. Optical correlator that is using a shadow-casting system.

input image. For this technique, template matching on two images is frequently used to select specific data. This template matching plays an important role in the inference operation by OAL.

3. Inference

In a practical AI system, such as an expert system, inference is a process for answering given queries with a knowledge base that consists of data that represents various knowledge.[10] An inference engine is a mechanism for finding a solution for queries by searching data related to the queries with a knowledge base. To execute humanlike intelligent inference, the knowledge base must contain a huge amount of data. Thus searching data in inference must be executed efficiently by fast processing.

There are several methods for knowledge representation in computer science, for example, a production rule, a frame, a semantic network, and so on.[10] We use a semantic network as a knowledge representation because it is suitable for SIMD-type processing in OAL. A semantic network is a diagrammatic knowledge representation that is described by a directed graph consisting of nodes and links [Fig. 5(a)]. A node describes a concept, and a link represents a relation

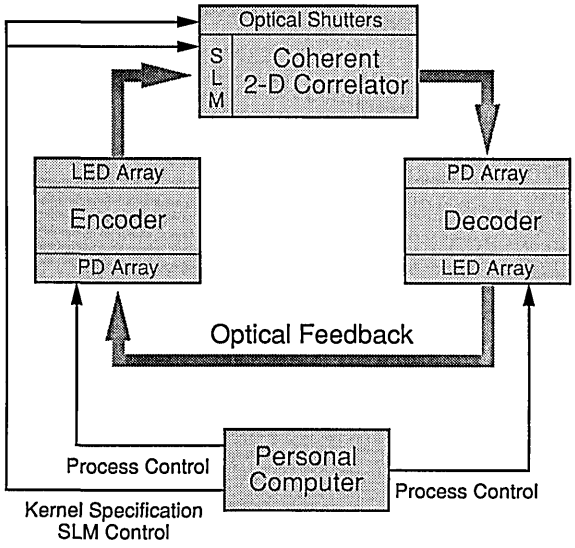

Fig. 3. Block diagram of a hybrid version of the OPALS. SLM, spatial light modulator; LED, light-emitting diode; PD, photodetector.

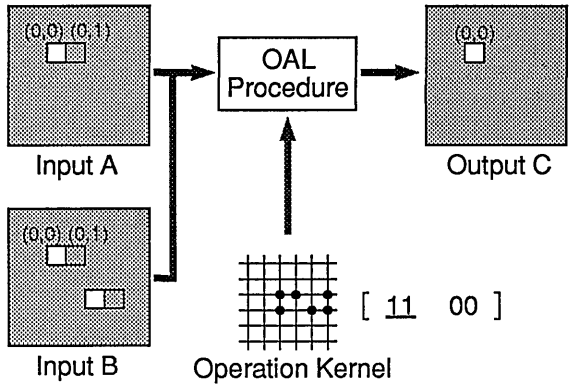

Fig. 4. Processing flow of template matching for two images.

between two concepts. Inference with a semantic network is achieved by tracking links from a node to the connected nodes. For example, when a query "What is a cat?" is given, the answer "An animal" is obtained by reversely tracking the ISA link from the Cat node to the Animal node.

The process of tracking links is executed by a method called marker propagation.[16] A marker is an information primitive that propagates on links to help the inference process. Parallelism in inference with a semantic network is obtained mainly by the marker propagation algorithm.

For example, when a query "What animal has whiskers?" is given, as shown in Fig. 5(b), two markers (marker #1 propagating along the HAS link in the reverse direction and marker #2 propagating along the ISA link in the reverse direction) are prepared and set on Whiskers and Animal nodes, respectively. These markers propagate simultaneously along the specified links. The node to which both markers reach indicates the answer Cat [Fig. 5(c)].

The process of the inference engine with marker propagation is as follows: (1) searching nodes appearing in a given query, (2) setting markers on the searched nodes, (3) propagating the markers along the specified links to the next nodes, and (4) repeating step (3) until all markers come to the end of the propagation paths.

Fortunately the above process can be executed for all markers with a SIMD type of parallel processing. In our method, a token propagation technique is used to implement the markers. The direction and type of links are encoded into pixel patterns on an image plane. As a result the capability of parallel processing in OAL is quite useful for an inference process to use to increase processing efficiency.

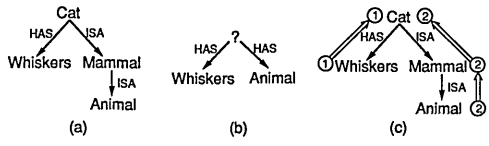

Fig. 5. Semantic network and inference based on the semantic network: (a) an example of a semantic network, (b) a query, (c) inference based on the semantic network with marker propagation.

4. Primitives for Inference Engine in Optical Array Logic

A. Encoding of a Knowledge Base

To implement an inference process by using OAL, a semantic network must be represented by pixel patterns on an image plane. A semantic network in Fig. 5(a) is converted into a spatial representation by the following procedures, as shown in Fig. 6.

First, the target semantic network is divided into multiple subnetworks with a one-to-many structure. In an individual subnetwork one node is connected to several nodes through different links [Fig. 6(a)]. The assembly of the subnetworks is coded into a table format in which the rows and the columns are assigned to each subnetwork and to an individual node, respectively [Fig. 6(b)].

In the table format, the direction and the type of link must be stored at each cell. The type of link is set at the terminating cell of the link. The direction and type of link are coded into spatial patterns, called node patterns, and set at the cells [Fig. 6(c)]. An area for setting a marker in inference is also prepared to the cell. Code patterns are assigned to each meaning, as shown in Fig. 6(d). In the explanation of the link direction pixels, a small dot indicates a node. The code patterns for the link type are the specified ones for this example. By setting the code patterns to the appropriate position in Fig. 6(b), we can obtain a spatial representation of a semantic network for OAL [Fig. 6(e)].

In addition, to identify the individual cells in the spatial representation, anchor tags are set on the other image [Fig. 6(f)]. This technique is often used in OAL to increase processing flexibility. The images containing the data and the anchor tags are called data and attribute planes of a knowledge base, respectively. Template matching for the both planes enables us to access an arbitrary part of the knowledge base.

A query is represented with a subset of a semantic network, as shown in Fig. 5(b). It can be encoded in the same way as the knowledge base. That is, the query is represented by the table format and coded into spatial patterns. Usually each condition in a query is assigned to an individual marker for inference. Thus markers #1 and #2 are set in the appropriate positions in the cells [Fig. 6(g)].

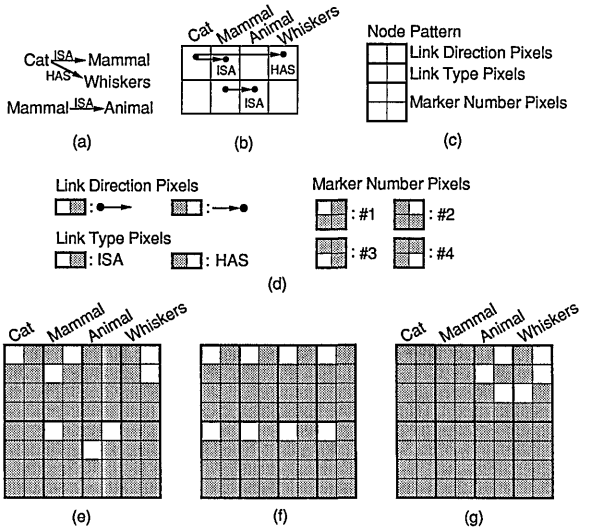

Fig. 6. Encoding of knowledge base by using a semantic network: (a) assembly of semantic subnetworks; (b) table format representing the subnetworks in (a); (c) node pattern used in the table in (b); (d) example of pixel assignment for the node pattern; (e), (f) data and attribute planes of the knowledge base, respectively, (g) a coded query.

B. Token Propagation

To implement parallel inference by using OAL, we develop a token propagation technique. The basic ideas of the token propagation are (1) setting some tokens on a 2-D image to indicate interested objects and (2) transferring the tokens concurrently to execute desire operations with parallel neighborhood logic operations in OAL. Since individual tokens are transferred independently, great flexibility and processing capability can be attained even on a SIMD processor.

To implement token propagation with OAL, pattern expansion and template matching are employed. Figure 7 indicates a procedure for propagating a token to a desired destination. First, a token to be propagated is duplicated over a specific area. This operation is called pattern expansion and is achieved by the logical sum of variables a and b at the neighborhood location on which the target pattern is to be expanded. Correlation in the procedure of OAL provides this pattern expansion. The expanded tokens are selected by template matching; the template is specified by an image called a condition image. As a result, the initial token is transferred to the destination that satisfies a given condition. This procedure is executed by SIMD operations so that OAL can be used to implement it.

In token propagation, the area of pattern expansion and the condition image are used to specify the contents of the processing. As shown in Fig. 8, if tokens are expanded in the horizontal (or vertical) direction, multiple tokens can be transferred indepen-

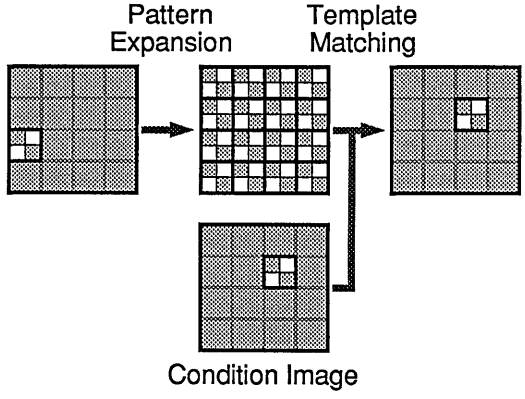

Fig. 7. Procedure for propagating a token to a desired destination.

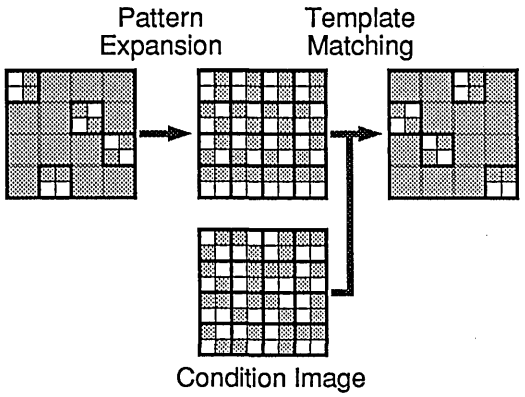

Fig. 8. Procedure for propagating multiple tokens to desired destinations.

dently. In addition, appropriate patterns are set on the condition image so that the behavior of each token can be controlled. Although the destination of tokens is limited by the area of pattern expansion, multiple processes are handled in OAL with this technique.

C. Inference with Token Propagation

With the token propagation technique on a spatially encoded knowledge base, inference processing is executed efficiently by OAL. For example, a problem in Fig. 5(b), or a query, "What animal has whiskers?," is solved by OAL. As described in Section 3, markers propagating from the Whiskers and the Animal nodes are used for this problem. In our method, tokens on an image are assigned to the markers and propagated by OAL operations.

The basic procedure for executing the inference is (1) searching subnetworks corresponding to given queries, which are assigned to the rows in the knowledge base; (2) setting tokens at the searched cells; (3) propagating the tokens in the same rows to track reversely the specified links; and (4) repeating steps (1)–(3) until all tokens reach an identical node or the same column.

Figures 9–11 show practical processing for the inference. Steps (1) and (2) from above are executed by vertical token propagation, shown in Fig. 9, in which the condition image is created from the knowledge base to select the terminating nodes of the ISA and HAS links. In this case, a pattern consisting of 4×2 pixels for a cell is assigned as a token. To identify the region of the individual cell, anchor tags in the attribute plane of the knowledge base are used. As a preparation of step (3), the link direction pixels on the tokens are changed to indicate an initiating node, as shown in Fig. 10. The modified tokens are propagated horizontally to execute step (3), as shown in Fig. 11. The transferred token is again changed to indicate an initiating node. The obtained token has the same format of the query, so that the above processing is repeated until the final result is obtained, as shown in Fig. 12. The operation kernels used in the operations are shown in Appendix B.

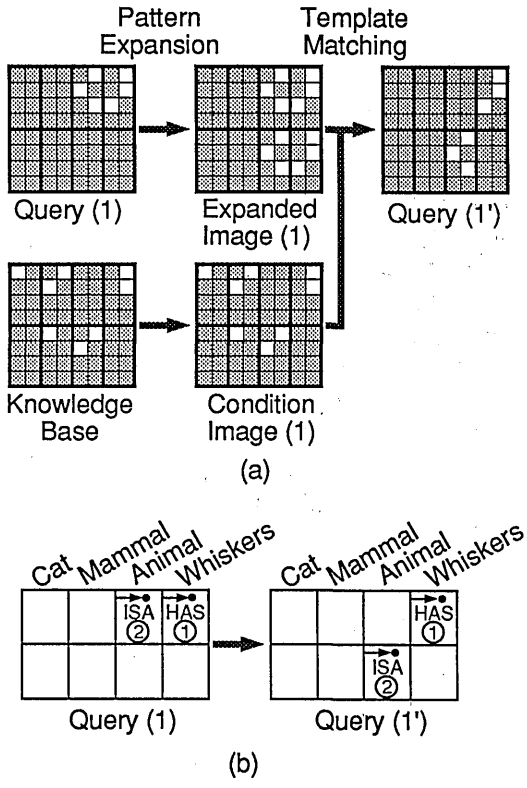

Fig. 9. Processing sequence of vertical token propagation: (a) images, (b) semantic representation.

5. Inference Engine for an Expert System by Using OAL

A. Required Functions for an Inference Engine

An expert system stores the knowledge of experts of a specific field in a knowledge base and efficiently offers the knowledge for nonexperts. To extend a primitive inference engine to an expert system, three functions must be added: (1) inheritance, (2) the detection of propagated markers, and (3) the detection of the termination of marker propagation.

We explain function (1) with an example in Fig. 13. We assume a query "What has whiskers and skin?" [Fig. 13(b)] on the semantic network shown in Fig.

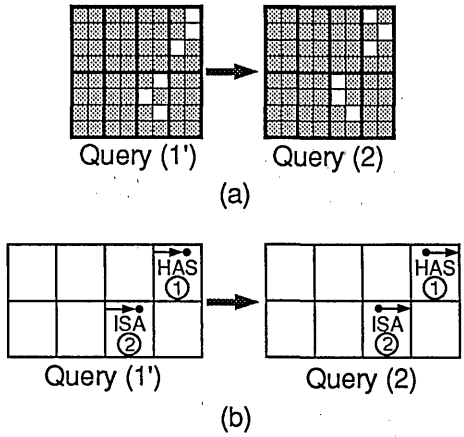

Fig. 10. Changing of link direction pixels on the tokens: (a) images, (b) semantic representation.

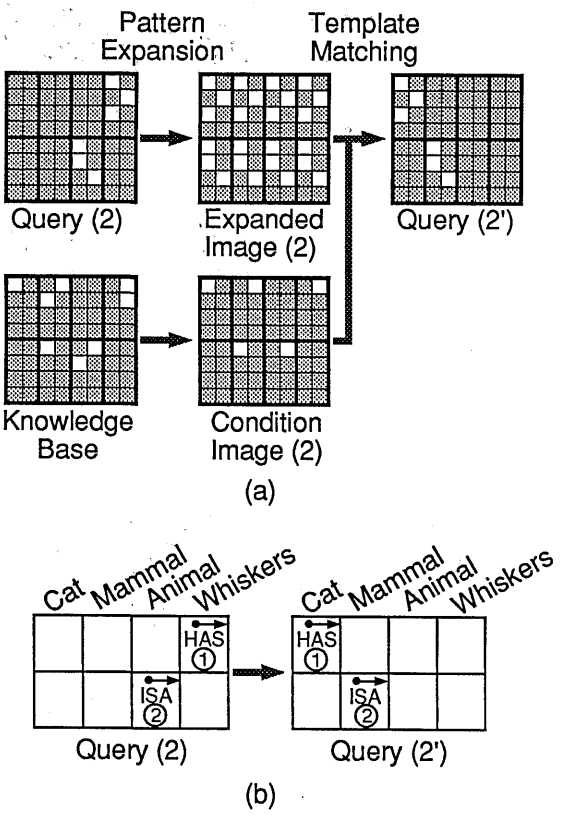

Fig. 11. Processing sequence of horizontal token propagation: (a) images, (b) semantic representation.

13(a). As shown in the semantic network, Cat is a member of Animal. Since characteristics of Animal must be held by its member Cat, the property Has Skin should be inherited from the Animal to the Cat node. The function is achieved by propagating markers along both a specified link (in this example, the HAS link) and along the ISA link [Fig. 13(c)].

To execute function (2), markers must be recorded on all the propagated nodes. After propagation, we can find the node that satisfies specific conditions by detecting the corresponding recorded markers. In the example of Fig. 13, by detecting a marker pair of #1 and #2 the Cat node is derived as the answer.

Before detection of the nodes with the desired markers, the marker propagation process must be terminated [function (3)]. This function is achieved by detecting whether any marker is propagated from the current node. Since these functions can be executed for all data at a time, parallelism in OAL is utilized.

B. Inference Engine for Expert System by Using OAL

The processing flow of inference for an expert system is shown in Fig. 14. The main part of the processing is identical to the marker propagation process described in Section 4. Thus we explain additional functions for an expert system.

To simplify the sequence, the inheritance process is assumed to be executed only by tracking reversely. In this case, all links in the given query, except the

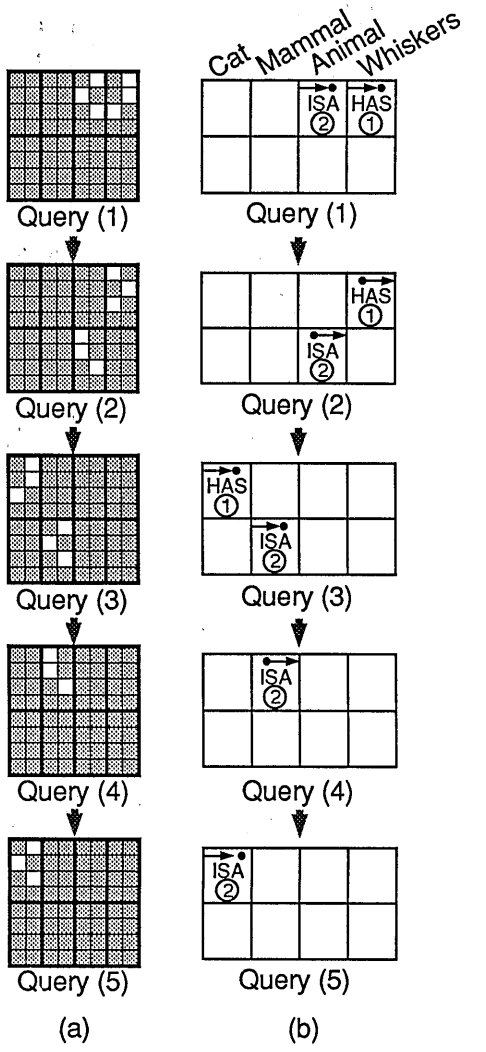

Fig. 12. Processing sequence of token propagation in query image: (a) images, (b) semantic representation.

ISA links, are reversely tracked and then the same operation is executed for the ISA links. As a result, inheritance can be utilized in a kind of inference, such as "What has skin?" Although this restricts the functions of an expert system, a more complicated sequence by OAL enables us to extend the function.

Recording of the propagated markers is achieved by copying the marker number pixels in the query to the corresponding pixels on the knowledge base. Figure 15 shows an example of the knowledge base before

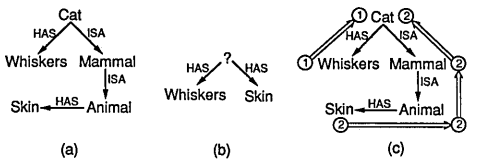

Fig. 13. Implementation of inheritance based on a semantic network: (a) an example of a semantic network, (b) a query, (c) inheritance based on the semantic network with marker propagation.

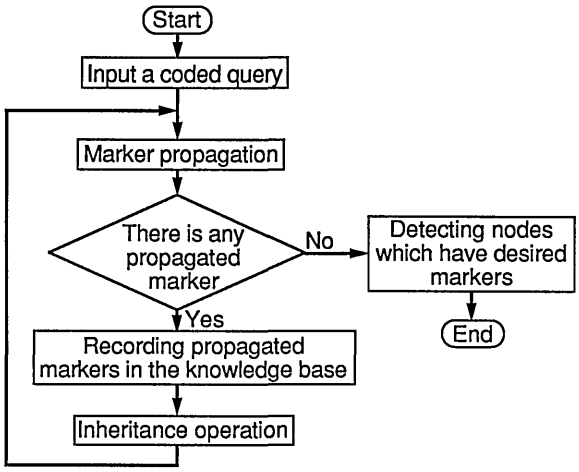

Fig. 14. Processing flow of the inference engine for an expert system with marker propagation.

and after recording markers. This operation can be executed by simple logic operations.

To detect whether any marker is propagated from the current node, we introduce conditional variables as an extension of OAL. Conditional variables indicate conditions of the processed image, like conditional flags in a microprocessor. For this case, a ZERO variable is prepared, whose value is 1 if all pixels in the output image are 0 and is 0 for the other case. Although the same function is implemented by a sequence of logic operations in OAL, conditional variables can greatly improve processing efficiency through the use of specific optical hardware. For example, a ZERO variable is simply implemented with a condenser lens and a photodetector, as shown in Fig. 16. As a result, a conditional test for termination is achieved by referring to the ZERO variable.

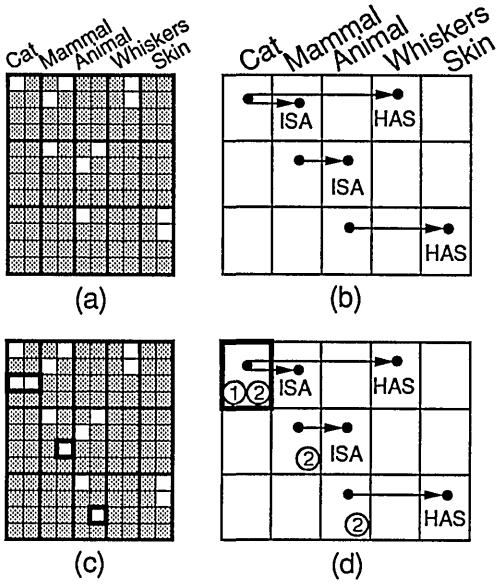

Fig. 15. Recording of the propagated markers on the knowledge base: (a) and (c), knowledge base before and after recording markers, respectively; (b) and (d), locations of the semantic subnetworks and nodes of (a) and (c), respectively.

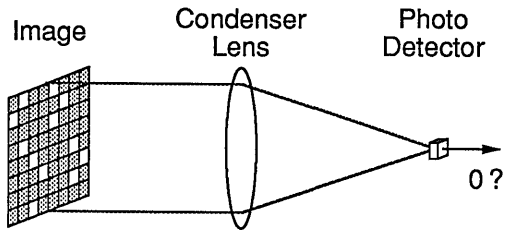

Fig. 16. Optical system for the conditional variable ZERO.

The nodes with the desired markers are detected by template matching. In Fig. 15, the marker pair of #1 and #2 is detected by testing the third column of each cell. As a result, the upper left cell is detected, and Cat is derived as the answer.

C. Simulation Result

With the above process, a simple expert system is implemented for a knowledge base, as shown in Fig. 17(a). On the expert system, a query in Fig. 17(b) is asked as an example: "What cat has short hair and no tail?" Figure 18 shows a simulated result of the expert system. Figures 18(a) and 18(b) are the attribute and the data planes of the knowledge base, respectively, and Fig. 18(c) indicates the query image. Figure 18(d) is the data plane of the knowledge base on which markers are recorded after the marker propagation process. Figure 18(e) shows the detected result of the node on which all used markers are recorded. The node is the answer of the query. The vertical and the horizontal scales around the images indicate the borders of the subnetworks and the individual nodes, respectively. The horizontal scale numbers correspond to the identifiers of the

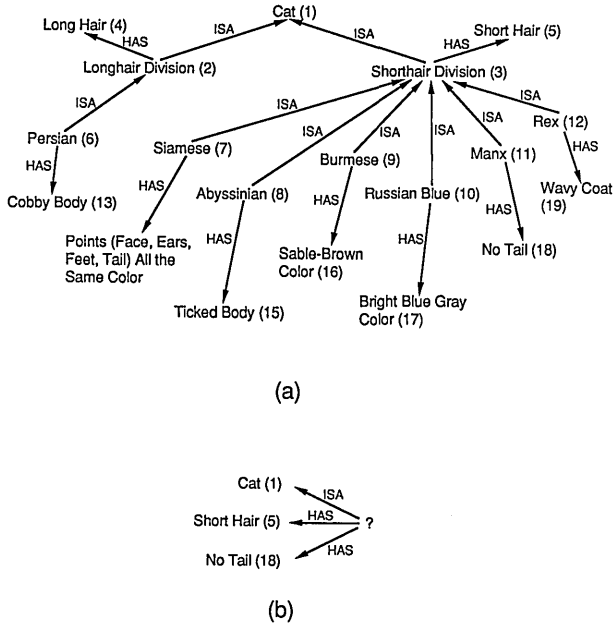

Fig. 17. Semantic network for an expert system: (a) an example of a knowledge base, (b) a query. The numbers in parentheses are the identifiers of the nodes.

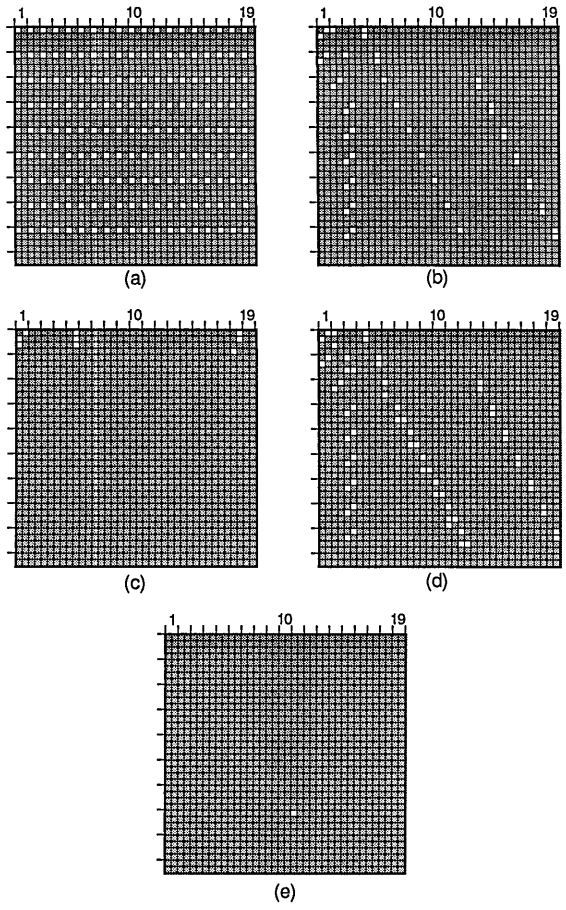

Fig. 18. Simulation result of the expert system with OAL: (a) and (b), data and attribute planes of the knowledge base, respectively; (c) the query image; (d) the data plane of the knowledge base that is recording marker numbers after marker propagation process; (e) detected result of the nodes that are recording all marker numbers. The vertical and the horizontal scales around the images indicate the borders of the subnetworks and the individual nodes, respectively. The horizontal scale numbers correspond to the identifiers of the nodes shown in Fig. 17.

nodes shown in Fig. 17. Consequently the final result, Manx, is obtained.

6. Performance Estimation

The processing performance of the programs described in OAL can be estimated by three factors: the total number of product terms required in the processing, the spatial size of the operation kernels used in correlation, and the spatial size of the images to be processed. The first determines the system cycles, and the others state necessary hardware specifications for the processing. The total number of product terms corresponds to the total number of system cycles required in processing. Actual processing time is obtained by the product of the total number of product terms and the processing time for one system cycle, which depends on the response time of the devices used, such as spatial light modulators.

Table 1 tabulates these values of the inference engine and the expert system by using the token propagation. As Table 1 shows, these values are

Table 1. Performance Estimation of the Inference Engine and Expert System by Using Token Propagation[a]

Number of Product Terms	Inference Engine $12i + 1$	Expert System $21i + 17 + 2MH$
Size of operation kernel (unit number)		
x	$(2S - 1)V$	$(2S - 1)V$
y	$(2N - 1)H$	$(2N - 1)H$
Size of Images		
x	SV	SV
y	NH	NH

[a] N, number of nodes; S, number of subnetworks; V, vertical pixel number of a node pattern; H, horizontal pixel number of a node pattern; M, vertical pixel number of marker number pixels; i, cycle number of inference sequence.

greatly affected by the size of the semantic network. For a practical expert system, at least a few hundred nodes are required. If 500 nodes are required, images that consist of 1000×1000 pixels and operation kernels with 2000×2000 units must be handled in an optical system.

The processing speed is estimated by the total number of product terms in the processing. Note that this value is independent of the size of the semantic network in one cycle of the inference sequence. Consequently, if a large size of images can be processed, the inference engine with OAL has the capability of executing one cycle of the inference sequence in a constant time, regardless of the size of the semantic network.

However, to accomplish an inference process, one step of inference described here must be iterated by the number of the maximum depth of a hierarchy in a semantic network. If the average of the division number of node information is m, the maximum depth of hierarchy is considered to be $\sim \log_m n$, where n is the number of nodes. This value depends on the character of a semantic network.

If an image representing a semantic network is larger than that handled by a system, the image must be separated and processed page by page. However, this method needs additional tasks for managing continuity of adjacent pages. Since this process reduces the performance, an efficient processing method of the overlapped pages should be considered.

7. Conclusion

We have presented an implementation method of an inference engine that utilizes the parallelism of OAL. The token propagation technique that was proposed as an OAL programming technique is useful for parallel inference. By using this technique, we have developed OAL programs for an inference engine and an expert system. As a result of the performance estimation, we have found that if a large size of images is handled on an optical system, the inference engine by OAL has the capability of executing one

step of the inference sequence in a constant time regardless of the size of the semantic network.

Appendix A

An operation kernel explicitly corresponds to logical meanings. Thus it can be expressed by symbolic notation for intuitive expression. We call the native notation for OAL kernel expression. The essential components of the kernel expression are as follows:

Product Term Block: PTB

Since an operation kernel specifies a product term operation, it is reasonable to introduce a functional block that corresponds to an operation kernel. The functional block is called the product term block, or PTB. It is indicated by a set of brackets.

Function Symbols

Elements in a PTB are two-letter symbols that indicate any one of two-variable binary logic functions. The symbols are tabulated in Table 2. The first and second letters assign functions for inputs A and B, respectively. To handle both the logic product and the sum, two series of letter sets are used. 1, 0, and . are for products, whereas P and N are for sums; symbols UU, EE, and DD are special symbols for specific functions.

Neighborhood Mapping and the Origin Marker

Operands of the target logic neighborhood operation are specified by the location of function symbols set in a PTB. That is, a PTB directly corresponds to a neighborhood area. The origin of a PTB is indicated by an underscore. If a PTB has only one symbol, the underscore can be omitted. The axes or the neighborhood area are set vertically downward and horizontally rightward, respectively.

Shift Suffix

There is a case in which operands of the target operation localize in the neighborhood area. For convenient notation, a PTB may have shift suffix indicating the amount of the origin offset. The shift suffix can be regarded as the address of the underscored position.

Sum and Product of PTB's

In OAL, multiple results of product term operations are logically summed. This procedure is expressed by the sum of the PTB's that correspond to the product term operations. The product of PTB's is identical to a PTB whose elements are composed of the result of the logic product of individual functions at the corresponding position in the PTB's. The product of PTB's can be used for separating operands in a PTB.

For example, the following kernel expression specifies a logic expression, which is indicated by Eq. (A2).

$$[10 \quad \underline{0.}] + \left[\begin{matrix}\underline{1.}\\ .0\end{matrix}\right]_{0,1}, \tag{A1}$$

$$c_{i,j} = a_{i,j-1}\bar{b}_{i,j-1}\bar{a}_{i,j} + a_{i,j+1}\bar{b}_{i+1,j+1}. \tag{A2}$$

Table 2. Symbols for Specifying Kernel Units

Function	Symbol	Function	Symbol
1	..	$a + b$	PP
$\bar{a} + \bar{b}$	NN	$a \oplus b$	UU
$\bar{a} + b$	NP	b	.1
$\bar{a}$	0.	$\bar{a}b$	01
$a + \bar{b}$	PN	a	1.
$\bar{b}$	.0	$a\bar{b}$	10
$\overline{a \oplus b}$	EE	ab	11
$\bar{a}\bar{b}$	00	0	DD

Appendix B

A kernel expression of template matching for two images (Section 2) is

$$[\underline{11} \quad 00]. \tag{B1}$$

Kernel expressions of inference with token propagation (Subsection 4.C) are as follows:

1. Horizontal marker propagation (see Fig. 9).

Step (1) Vertical pattern expansion (input A, query (1); input B, don't care):

$$\prod_{i=-S+1}^{S-1} [1.]_{4i,0}; \quad [0.]. \tag{B2}$$

Step (2) Creation of the condition image from the knowledge base (input A, knowledge base; input B, attribute plane of knowledge base):

$$[11] + [.1 \quad \underline{10}] + \left[\begin{matrix}\underline{.1}\\ 10\end{matrix}\right] + \left[\begin{matrix}.1 & ..\\ .. & \underline{10}\end{matrix}\right]. \tag{B3}$$

Step (3) Template matching [input A, expanded image (1); input B, condition image (1)]:

$$\sum_{i=0}^{3} \left(\left[\begin{matrix}\underline{00} & 11\\ EE & EE\end{matrix}\right]_{0,-i} + \left[\begin{matrix}00 & \underline{11}\\ EE & EE\end{matrix}\right]_{0,-i} \right). \tag{B4}$$

2. Change in the link direction pixels on the tokens [see Fig. 10; input A, query (1'); input B, don't care]:

$$[\underline{11} \quad 00]. \tag{B5}$$

3. Horizontal token propagation (see Fig. 11).

Step (1) Horizontal pattern expansion [input A, query (2); input B, don't care]:

$$\prod_{i=-N+1}^{N-1} [1.]_{0,2i}; \quad [0.]. \tag{B6}$$

Step (2) Creation of the condition image from the knowledge base (input A, knowledge base; input B, attribute plane of knowledge base):

$$[11] + [.1 \quad \underline{10}]. \tag{B7}$$

Step (3) Template matching [input A, expanded image (2); input B, condition image (2)]:

$$\sum_{i=0}^{3} ([\underline{11}\quad 00]_{0,-i} + [11\quad \underline{00}]_{0,-i}). \tag{B8}$$

4. Change in the link direction pixels on the tokens [input A, query (2′); input B, attribute plane of knowledge base):

$$[01\quad \underline{10}]. \tag{B9}$$

In the above expressions and equations, Σ and Π denote sum and product operations in kernel expressions, respectively; S is the number of subnetworks, and N is the number of nodes. The input images A and B in expressions (B2)–(B9) are represented by the image names in Figs. 9–12. The attribute plane of the knowledge base is shown in Fig. 6(f). In expressions (B2) and (B6), the symbol ; means that the following operation is executed after the previous operation has been done.

References

1. D. G. Feitelson, *Optical Computing. A Survey for Computer Scientists* (MIT, Cambridge, Mass., 1988).
2. B. S. Wherrett and F. A. P. Tooley, eds., *Optical Computing* (Scottish Universities Summer School in Physics, Edinburgh, Scotland, 1989).
3. R. Arrathoon, ed., *Optical Computing: Digital and Symbolic* (Dekker, New York, 1989).
4. J. Tanida and Y. Ichioka, "Programming of optical array logic. 1: image data processing," Appl. Opt. **27,** 2926–2930 (1988).
5. J. Tanida, M. Fukui, and Y. Ichioka, "Programming of optical array logic. 2: numerical data processing based on pattern logic," Appl. Opt. **27,** 2931–2939 (1988).
6. J. Tanida, J. Nakagawa, E. Yagyu, M. Fukui, and Y. Ichioka, "Experimental verification of parallel processing on a hybrid optical parallel array logic system," Appl. Opt. **29,** 2510–2521 (1990).
7. S. Kakizaki, D. Miyazaki, Eiji Yoshikawa, J. Tanida, and Y. Ichioka, "Hybrid optical array logic system," in *Optics for Computers: Architectures and Technologies,* G. J. Lebreton, ed., Proc. Soc. Photo-Opt. Instrum. Eng. **1505,** 199–205 (1991).
8. M. Fukui, J. Tanida, and Y. Ichioka, "Flexible-structured computation based on optical array logic," Appl. Opt. **29,** 1604–1609 (1990).
9. N. J. Nilsson, *Principles of Artificial Intelligence* (Tioga, Palo Alto, Calif., 1980).
10. K. Bauer, I. Büttel, L. Eberhard, M. Hälker, H. Lehner, K. Micholka, and R. Paulsburg, in *Expert Systems,* D. Nebendahl, ed. (Wiley, Chichester, England, 1988).
11. Feature issue on optical artificial intelligence, Appl. Opt. **26,** 1827–1958 (1987).
12. G. Eichmann and H. J. Caulfield, "Optical learning (inference) machines," Appl. Opt. **24,** 2051–2054 (1985).
13. C. Warde and J. Kottas, "Hybrid optical inference machines: architectural considerations," Appl. Opt. **25,** 940–947 (1986).
14. M. Iwata, J. Tanida, and Y. Ichioka, "Inference engine using optical array logic," Jpn. J. Appl. Phys. **29,** L1259–L1261 (1990).
15. F. Kiamilev and S. Esener, "Implementation of NETL knowledge-base system with programmable opto-electronic multiprocessor architecture," in *Optical Computing,* Vol. 9 of 1989 OSA Technical Digest Series (Optical Society of America, Washington, D.C., 1989), pp. 303–306.
16. S. E. Fahlman, *NETL: A System for Representing and Using Real-World Knowledge* (MIT, Cambridge, Mass., 1979).