Parallel electro-optical rule-based system for fast execution of expert systems

Ahmed Louri and Jongwhoa Na

The slow execution speed of current rule-based systems has restricted their application areas. Multiprocessor architectures have been proposed to overcome this limitation. However, as the number of processors in a multiprocessor system grows, so does the cost of communication between processors or between processor and memory units. The use of optics for a fast and parallel implementation of rule-based systems is proposed. The proposed optical system is hybrid in nature, using electronics for the user interface and optics for the rule-based inference engine. The proposed system uses two-dimensional planes as basic computational entities and is therefore able to provide concurrent rule processing. Furthermore, it provides highly efficient implementation of the basic operations needed in rule-based systems; namely, matching, selection, and rule firing. The execution speed of the proposed system is theoretically estimated and is shown to be potentially of orders of magnitude faster than current electronic systems.

Key words: Expert system, electro-optical rule-based system, inference engine, optical interconnection networks.

1. Introduction

A key feature of artificial intelligence computing systems is the large amount of irregular communications required between the large number of processing elements[1–3] (PE's). Unfortunately, current electronics technology is unable to provide adequate bandwidth and connectivity for the large number of PE's involved. Consequently, current symbolic computing tasks, including rule-based systems (RBS's) require an excessive amount of time to produce results. Optics offers the possibility of eliminating the performance bottlenecks associated with communications. Optics has the advantages of higher spatial and temporal bandwidth, no cross talk, larger fan-in and fan-out, and two-dimensional (2-D) data-processing capability.[4,5] These advantages are useful in exploiting massive parallelism and in achieving higher throughput.[6] Optical systems can provide not only adequate communication support, but also the necessary parallelism to perform the most frequently used and time-consuming operations in RBS's, namely, pattern matching and comparison operations.

The authors are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, Arizona 85721.

Received 21 October 1991.
0003-6935/93/111863-13$05.00/0.

This has resulted in major research efforts for exploiting the use of optics in symbolic artificial intelligence computing.[7–16] Warde and Kottas[7] have proposed two different hybrid optical inference machines. The matched-filter inference machine performs inferencing using a classical Vander-Lugt matched-filter system.[8] The mapped template inference machine matches a template representing the relationship between input and output data to produce related output data with the optical correlograph system of Willshaw and Longuet-Higgins.[11] Jau *et al.*[9] used a semantic network as a knowledge-representation scheme. Jau showed that a simple query can be answered by using an optical matrix-multiplication technique. Schmidt and Cathey[10] proposed an optical mathematical resolution system, which can be used as an inferencing mechanism in an expert system.

We propose a parallel electro-optical rule-based system (called EORBS), in which rules are used to represent knowledge and an inference engine is used for reasoning. To take advantage of optics properties, we represent rules in 2-D space, and the inference engine is logically transformed into an optical interconnection network that can carry out inferencing optically and hence in a highly parallel fashion.

Section 2 provides a brief background of RBS's. Section 3 describes fundamental concepts of the proposed EORBS. Section 4 discusses the implemen-

tation of EORBS. Section 5 describes the operation of EORBS, and Section 6 provides projected performance data for the EORBS. Section 7 concludes.

2. Background

An expert system can be defined as an intelligent system that can mimic some part of human intelligence. For example, MYCIN is a rule-based expert system that diagnoses bacterial infections.[17] MYCIN uses ~500 rules to diagnose approximately 100 causes of infections. An expert system is composed of (1) a RBS and a knowledge-acquisition facility, (2) an explanation facility, and (3) a user interface, as shown in Fig. 1.[18] The RBS performs inferencing by using the knowledge base and the inference engine. The knowledge-acquisition facility and the user interface act as an interface unit between the RBS and the user. The explanation facility explains to the user how the results have been obtained. The knowledge base is composed of rules representing universal knowledge and facts that represent current knowledge. A rule conjoins condition elements C_i and action elements A_i with the following format:

$$\text{if}\ \overbrace{(C_1 \wedge C_2 \wedge C_3 \cdots \wedge C_n)}^{\text{condition elements}} \Rightarrow \text{then}\ \overbrace{(A_1,\ A_2, \ldots, A_m)}^{\text{action elements}}.$$

The condition elements and action elements are represented by facts, which are the basic units of knowledge in an RBS.

The inference engine uses the modus ponens theorem as an underlying principle. The modus ponens theorem states that if there is an axiom of the form $E_1 \Rightarrow E_2$, and if there is another axiom of the form E_1, then E_2 logically follows.[19] The inference engine uses modus ponens as an underlying principle as follows:

$$\left\{ \overbrace{(A \text{ is TRUE})}^{\text{current fact}} \wedge \overbrace{(\text{if } A, \text{ then } B)}^{\text{rule}} \right\} \Rightarrow \overbrace{(B \text{ is TRUE})}^{\text{inferred fact}}$$

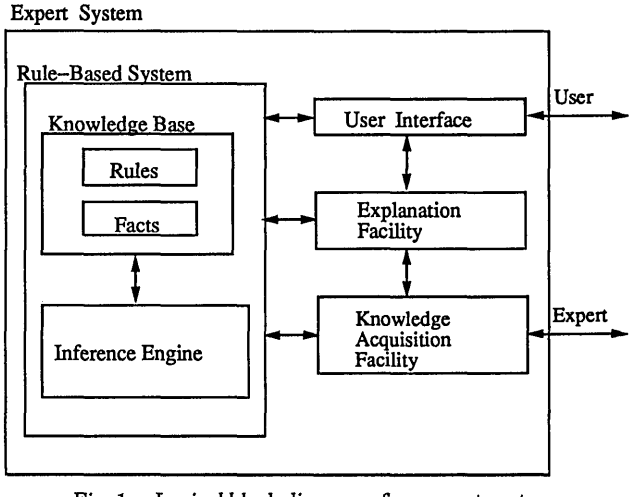


Fig. 1. Logical block diagram of an expert system.

Thus, by using known facts and rules, the inference engine can produce new facts in a forward-chaining system, a backward-chaining system, or both.[19] In the forward-chaining system the system tries to find final (goal) states by comparing the known facts, which describe the initial states, with the condition-specifying "if" parts of the rules. In the backward-chaining system the system tries to find initial hypotheses by comparing the known facts, which here describe the final (goal) states, with the action-specifying "then" parts of the rules.

The basic operations of an RBS are as follows:

(1) For the match operation (for each rule), determine whether the condition part of the rule matches the current facts. If a rule satisfies the condition part, the rule is added to the conflict set. Otherwise, the rule is discarded. A conflict set is a set of triggered rules that have satisfied condition parts.

(2) For the select operation, if the conflict set is empty, the inference engine stops inferencing and reports the failure to the user. If the conflict set is not empty and contains more than one rule, the inference engine selects one rule from the conflict set by applying one or more of the following conflict-resolution strategies.[20]

(a) For specificity ordering, if there are two rules and the condition part of one rule is more specific than the other, discard the latter.

(b) For rule ordering, if there are many rules, choose the first rule.

(c) For recency ordering, if there are many rules, choose the most (or least) recently used rule.

(d) For data ordering, if there are many rules, choose the rule that has the most important condition elements.

(3) For the act (rule-firing) operation, the inference engine fires the selected rule by executing its action. Since the current facts are changed by the fired rule, it is important to check whether the changes agree with predefined goals. If the goals are satisfied, the inference engine stops inferencing, and it reports results to the user. Otherwise, the inference engine must continue until the goal is satisfied or until there are no more matching rules.

In an RBS, facts and rules can be represented as an inference net or as an AND/OR tree. Figure 2 shows an inference net and an AND/OR tree for the following set of rules ($R1$–$R4$):

$$\text{If } (a \wedge b) \text{ then } i,$$

$$\text{If } (a \wedge c) \text{ then } j,$$

$$\text{If } (d \wedge i) \text{ then } u,$$

$$\text{If } (e \wedge j) \text{ then } v,$$

where $\wedge$ is a logical AND operator.

Considering that the typical number of rules in an RBS is large and that each rule has several

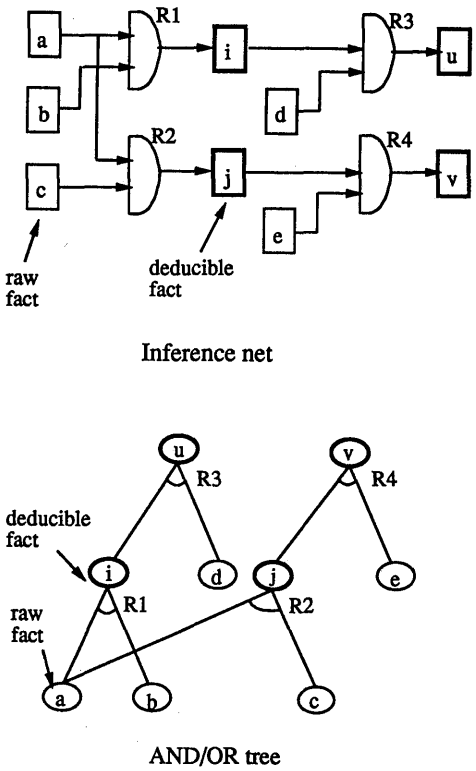


Fig. 2. Graphical interpretation of a knowledge base.

condition/action elements, the inference net can be extremely complicated. To traverse this increasing search space effectively, researchers have proposed parallel-processor systems that incorporate sophisticated parallel algorithms.[21] For example, DADO2 (Ref. 1) is a tree-structured multiprocessor system that has 1023 PE's. Each PE is composed of an 8-bit microprocessor, 64 kbytes of random-access memory, and a switch. Owing to the large number of PE's, communication time is a dominating factor in DADO2.[21] Furthermore, since each rule of the knowledge base is unique and requires a different processing time, there is a synchronization problem between rules with a small number of condition elements and rules with a large number of condition elements.

As an alternative to the electronic parallel-processing systems, optical systems such as the matched-filter inference engine (MFIE)[7] and the matrix-multiplication inference engine (MMIE)[9] have been proposed. These systems can represent a large number of related facts, called relations, owing to the multi-dimensionality of optical systems. One characteristics of these systems is that many facts within a relation can be processed concurrently (fact-level parallelism). In the MFIE a relation (set of facts) is compared with a condition element of a rule; therefore, if a problem consists of r rules, l condition elements per rule, and a condition elements per relation, the MFIE must execute rl/a matched-filtering operations to answer a query. In the MMIE a relation is represented in a 2-D matrix. Then, this relation is loaded into an optical matrix-multiplication unit so that multiple facts in a relation can be compared with a condition element of a rule. Thus, for a problem consisting of r rules, l condition elements per rule, and a condition elements per relation, the MMIE requires rl/a matrix multiplications to generate a final matrix that can be used to answer a query. Note that the MFIE and MMIE require a multiple optical matched filter or a multiple optical matrix multiplier, respectively, to exploit a condition element-level parallelism and a rule-level parallelism available in the rule-based system. In the following sections we discuss an optical RBS architecture that can exploit a condition element-level parallelism and a rule-level parallelism as well as a fact-level parallelism.

3. Electro-Optical Rule-Based System: Overview

The main objective of the EORBS is the exploitation of maximum parallelism in RBS's. This is achieved by using a 2-D representation of knowledge and multidimensional optical interconnects to speed up the match, select, and act operations of an RBS. EORBS offers a major advantage over previous proposals (including optical ones): simultaneous processing of several rules as opposed to the sequential processing of rules.

In order to process many rules in parallel, we have to ensure that multiple rule firing has no side effects.[22] EORBS guarantees this by employing a monotonic-reasoning methodology[23] and by identifying a set of mutually exclusive rules. The monotonic-reasoning methodology ensures that rule-firing operations only augment the facts without deleting or changing any facts that can be sources of side effects. Therefore, once a fact is asserted, the fact remains true forever. Thus the number of true facts keeps growing until the end of an inference operation. To apply the monotonic-reasoning scheme, we have to restrict a variable to be bound to some value during compilation time so that the data presented at the run time can be represented by propositions. Consider the following examples F_1 and F_2:

$$72 \leq 3.832,$$

$$a \geq b.$$

For F_1 the value of the expression can be evaluated directly so that the host can assign FALSE to F_1. For F_2, first, the variables a and b must be bound to some values. Then, the host can evaluate F_2 and assign TRUE or FALSE to F_2. Therefore, if we can collect all the data in advance and we can infer something out of the given knowledge, the monotonic-reasoning methodology can be applied. The application areas for the monotonic-reasoning methodology include theorem proving, problem-solving systems, diagnostics, and consultation problems.

In order to evaluate rules in parallel, the host must maintain consistency. To maintain consistency, the host must compare a fact element with a group of

related rules and facts. Thus the size of the group can be important in that the comparison time depends on the number of data in the group. One of the major characteristics of the rule-based programming environment is modularity and hierarchy.[20] These stem from the fact that the structure of human knowledge can be modeled in a modular and hierarchical structure such as a tree, in which each node represents a module that is composed of a small number of rules and facts. In this tree (from a high-level view point), checking whether an incoming fact belongs to some mutually exclusive group of facts or not implies that the incoming fact must be compared with a mutually exclusive group. Thus the fact is compared with a specific group that is small in size. Also, note that this checking operation is performed at the compilation time. Therefore the checking time for a given fact for the group should not cause any excessive overhead at run time.

Using a monotonic-reasoning scheme is advantageous in that EORBS does not require conflict resolution. Since EORBS explores all the possible paths in the search space without any side effect, it does not have to select which path to follow. An example of parallel rule firing can be found in the fuzzy RBS used in control.[24]

A. Knowledge Representation in the Host Computer

In general, to run any high-level programming language on a computer, one must translate the source code into an object code that can be run by a central processing unit. Likewise, RBS's require the rules to be in human-readable form, which is also required for the translated code that is optimized for the hardware. In the RBS the human-readable rules act as an interface between the user and the computer. These rules are used in editing and debugging the knowledge base of a given problem. As a result, RBS's keep their own version of the translated knowledge base. As an example, MYCIN uses a rule compiler for generating a decision tree that is used by the inference engine. For EORBS we developed a translation algorithm that translates the knowledge base into a 2-D array called a proposition table (PT). Figure 3(b) shows a PT that results from the application of the translation algorithm introduced below to the knowledge base of Fig. 3(a).

An entry of the PT is composed of a condition/action element and a proposition. The condition element slot can be divided further into a left-element slot, an operator slot, and a right-element slot. The left-element slot has the name of an object or variable, the right-element slot has a variable or some constant, and the operator slot contains the operation to be performed with the left-element slot and right-element slot. Also, the proposition slot can be divided further into two slots: one for the name (integer identification number) of the proposition and the other for the binary value of the proposition.

Along with the PT, the host computer generates a condition array (CA) and an action array (AA), which

R1. If (weight < 110 lb) ∧ (frame = small) ∧ (gender = female)
Then (relative-weight = normal)

R2. If (weight < 110 lb) ∧ (frame = large) ∧ (gender = male)
Then (relative-weight = under-weight)

(a) Example rules

Condition / Action element			Proposition	
Left-element	Operator	Right-element	s	Vc
weight	<	110 (lb)	F 001	T
frame	=	small	F 002	F
gender	=	female	F 003	T
relative-weight	=	normal	F 004	F
frame	=	large	F 005	F
gender	=	male	F 006	T
relative-weight	=	underweight	F 007	T
⋮	⋮	⋮	⋮	⋮

(b)

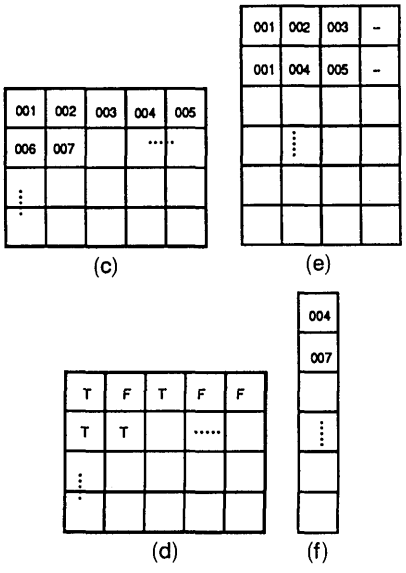


Fig. 3. Representation of the rules and facts in the host computer: (a) example rules, (b) PT, (c) RFA, (d) DFA, (e) CA, (f) AA.

are shown in Figs. 3(e) and 3(f), respectively. The CA and the AA represent the condition parts and action parts of all the rules in the knowledge base. The CA has a dimension of $r \times l$ and the AA has a dimension of $r \times m$, where r is the number of rules, l is the number of condition elements per rule, and m is the number of action elements per rule.

The host also generates two copies of the 2-D fact array [Figs. 3(c) and 3(d)]. In the first copy, called the reference fact array (RFA), each element of the array represents the name of the fact; therefore the RFA is a name template of the optical fact plane of EORBS. In the second copy, called the data fact array (DFA), each element of the array is the value of the element that is designated by the RFA; therefore the DFA is an actual data plane for EORBS. An

algorithm that interprets a human-readable knowledge base accessed by users to generate the PT, CA, AA, RFA, and DFA is described below.

(1) Initialize the PT, RFA, DFA, CA, and AA.
(2) For $i = 1$ to (the number of rules), find current rule.
(3) For $j = 1$ to (number of condition elements of the current rule), do the following:
(a) Find the current condition element.
(b) Find the symbol for the current condition element. (If the current condition element has already used the symbol, then find the used symbol, else create a new symbol.)
(4) With the symbol (current condition element), do the following:
(a) Evaluate the value of the symbol. (Fill in a row of the PT.)
(b) Calculate PT[symbol] = (current element, symbol, symbol value).
(c) Calculate CA$[i, j]$ = symbol.
(d) Calculate $j = j + 1$; go to step (3).
(5) For $k = 1$ to (number of action elements of the current rule), do the following:
(a) Find the current action element.
(b) Find the symbol for the current action element. (If the current action element has already used the symbol, then find the used symbol or create a new symbol.)
(6) With the symbol (current action element), do the following:
(a) Evaluate the value of the symbol. (Fill in a row of the PT.)
(b) Calculate PT[symbol] = (current element, symbol, symbol value).
(c) Calculate AA$[i, k]$ = symbol.
(d) Calculate $k = k + 1$; go to step (5).
(7) Calculate $i = i + 1$; go to step (2). (Find the next rule).
(8) For $l = 1$ to (number of row of the RFA), do step (9).
(9) For $m = 1$ to (number of column of the RFA), do step (10).
(10) For $n = 1$ to (number of row of the PT), do the following:
(a) Calculate RFA$[l, m]$ = symbol slot of PT$[n]$.
(b) Calculate DFA$[l, m]$ = symbol-value slot of PT$[n]$.
(c) Calculate $m = m + 1$ and $n = n + 1$; go to step (9).
(11) Calculate $l = l + 1$; go to step (8).

B. Knowledge Representation in Electro-optical Rule-Based System

In EORBS, knowledge is represented in a 2-D form. Facts are represented in a 2-D plane called the fact plane. Rules have two parts, a condition plane to represent condition elements and an action plane to represent action elements. These planes represent the basic computational entity in EORBS. Also, to increase the utilization ratio of limited 2-D space, propositional logic is used. In a propositional logic system, each fact, goal, condition, or action is represented by a pixel in the corresponding 2-D plane. Each element of the plane is expressed in a proposition that is position encoded in 2-D space, as shown in Fig. 4. The value of a fact F_i, for example, is represented by light being on/off at the specific location of the fact plane. Similar considerations take place for other planes.

In the fact plane the host allocates a small portion of the fact plane for the goal facts. These goal facts occupy the last row of the fact plane and are used at the end of the inference operation. Thus the host computer must generate the RFA such that the last row of the fact plane has goal facts. If the number of goal facts exceeds one row, the host can add rows in the RFA to accommodate a large number of goal facts. The optimal size for the number of goal facts should be determined after a thorough simulation study.

C. Description of Electro-optical Rule-Based System

The overall role of EORBS is to infer new facts by using the current facts and rules and by comparing the inferred facts to the given goals. Figure 5 shows a block diagram of EORBS. It consists of an electronic host computer, a goal-checking unit (GCU), and an inferencing unit (IU). EORBS receives the necessary facts and rules from the host computer and returns the processed data to the host. The host computer uses a 2-D fact plane to convert electronic data to optical data for optical inferencing. The fact plane is encoded as explained above. The host computes and sets the interconnection pattern between the facts and rules in the IU (explained below).

The IU performs the match and the act operations. The match operation consists of comparing the current facts with the condition elements of the rules. During the act operation, the action parts of the rules with satisfied condition elements are executed. A detailed explanation of the match and the act operations is presented in subsection 3.D.

The role of the GCU is to determine whether or not there is any assertion of goal facts during the inferencing process. If the output from the GCU indicates a match, EORBS stops inferencing, and the host starts its own operation with the processed results from EORBS; otherwise, the inference engine starts the inference operation.

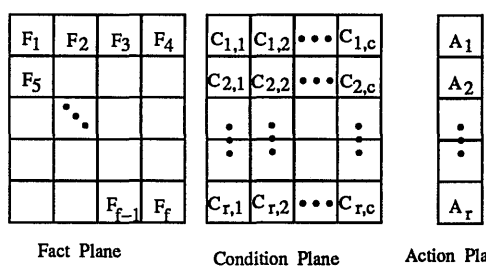


Fig. 4. Representation of the rules and facts for an optical system: F_i, fact element; C_{ij}, condition element; A_i, action element.

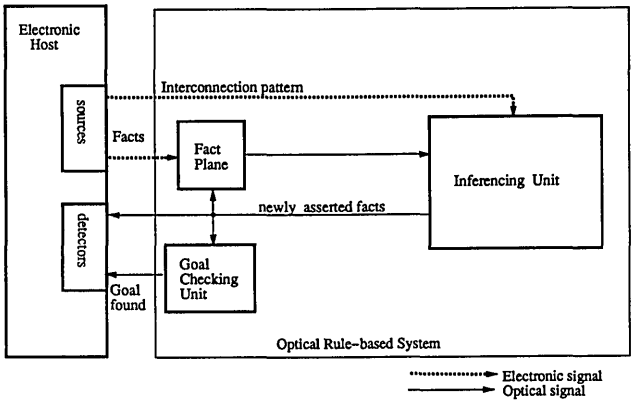


Fig. 5. EORBS.

D. Fact-to-Rule and Rule-to-Fact Mapping

Performing the match operation means that the condition elements of rules are evaluated with the current values of facts. This is usually a time-consuming operation in conventional RBS's, since it is done sequentially. Here we introduce a new method for performing rule matching that can be carried out in parallel, owing to the parallel nature of optical interconnects. The method consists of first providing a mapping called fact-to-rule (F-R) mapping from the facts to the rules. This mapping specifies the interconnection patterns required from the fact plane to the condition plane. Next, the condition templates of each rule are logically AND'ed into a single element whose value indicates whether the corresponding rule is to be selected or not. For example, consider the following two rules R_1 and R_2 and the general rule format R_i, given in sequence:

$$\text{if } (F_1 \wedge F_2), \text{ then } (F_{10});$$

$$\text{if } (F_1 \wedge F_3), \text{ then } (F_{20});$$

$$\text{if } (C_{i,1} \wedge C_{i,2} \wedge C_{i,3} \wedge C_{i,4}), \text{ then } (A_i).$$

$C_{i,j}$ represents a condition element in the ith row and jth column of the condition plane, and A_i represents an action element of rule R_i.

To determine the required interconnection pattern, we first compare rule R_1 and R_2 with the given rule format R_i. It can be easily seen that F_1 is mapped into $C_{1,1}$ and that F_2 is mapped into $C_{1,2}$, while $C_{1,3}$ and $C_{1,4}$ are not used in R_1. Also, comparing the next rule R_2 with R_i, we find that F_1 is mapped into $C_{2,1}$ and that F_3 is mapped into $C_{2,2}$, while $C_{2,3}$ and $C_{2,4}$ are not used in R_2. Thus, to perform the match operation, the routing network routes facts F_1 to $C_{1,1}$, F_1 to $C_{2,1}$, F_2 to $C_{1,2}$ and F_3 to $C_{2,2}$, respectively. An algorithm is described below that generates automatically a fact-to-rule mapping table (FRMT) from RFA and CA inputs.

(1) For $i = 1$ to (number of rows on the CA), do step (2).

(2) For $j = 1$ to (number of columns on the CA), do step (3).

(3) If CA$[i, j] \neq$ null, do the following:

(a) Find the available slot in the row (content of CA$[i, j]$) of the FRMT (content of CA$[i, j]$ = symbol identification number in integer).

(b) Calculate FRMT(CA$[i, j]$) = (i, j) [save destination (i, j) at the FRMT].

(c) Calculate $j = j + 1$; go to (2).

(d) Calculate $i = i + 1$; go to (1).

Since rules are defined as the conjunction of condition elements, we can set the unused elements (e.g., $C_{1,3}$ and $C_{1,4}$ in R_1) to a logical 1. Next, all the elements of each row of this condition plane are logically AND'ed together to form a one-dimensional (1-D) column vector representing the selected rules for the current iteration. This column vector represents the action plane. In general, the size and the dimension of the interconnection network and the AND-gate array representing the action plane depend on the configuration of the condition plane and the number of action elements per rule. In this example, we assume that each rule has only one action, hence the action plane is one dimensional. In general, the action part of the rule may contain more than one action element. In this case the AND array and the action plane are both of dimensions $r \times m$, where r is the number of rules and m is the number of action elements per rule.

In a similar fashion, the act operation must assert the action elements of the triggered rules into the current fact plane to form a new fact plane. Again, this operation is carried out by mapping from the action plane to the fact plane. This mapping, called the rule-to-fact (R-F) mapping, asserts new facts from the newly generated action plane into the previous fact plane. Using the above two rules and the algorithm described below, we can simply find the required interconnection patterns A_1–F_{10} and A_2–F_{20} for the RFMT from the RFA and AA inputs.

(1) For $i = 1$ to (number of elements on the AA), do (2).

(2) If AA$[i] \neq$ null, do the following:

(a) Find the available slot in the row (content of AA$[i]$) of the RFMT (content of AA$[i]$ = symbol identification number in integer).

(b) Calculate RFMT(AA$[i]$) = i (save destination i at the RFMT).

(c) Calculate $i = i + 1$; go to 1.

The new fact plane contains previous facts as well as newly inferred facts. These facts are then sent to the GCU for further processing. A detailed example of this mapping is described in Section 5.

4. Architecture of Electro-Optical Rule-Based System

A detailed optical architecture for EORBS is shown in Fig. 6. As described in Section 3, the system consists of a GCU, an IU, and 2-D optical source (fact) and detector planes for input–output (I/O) interfacing with the host. A 2-D optical source such as an electrically addressed spatial light modulator (SLM)

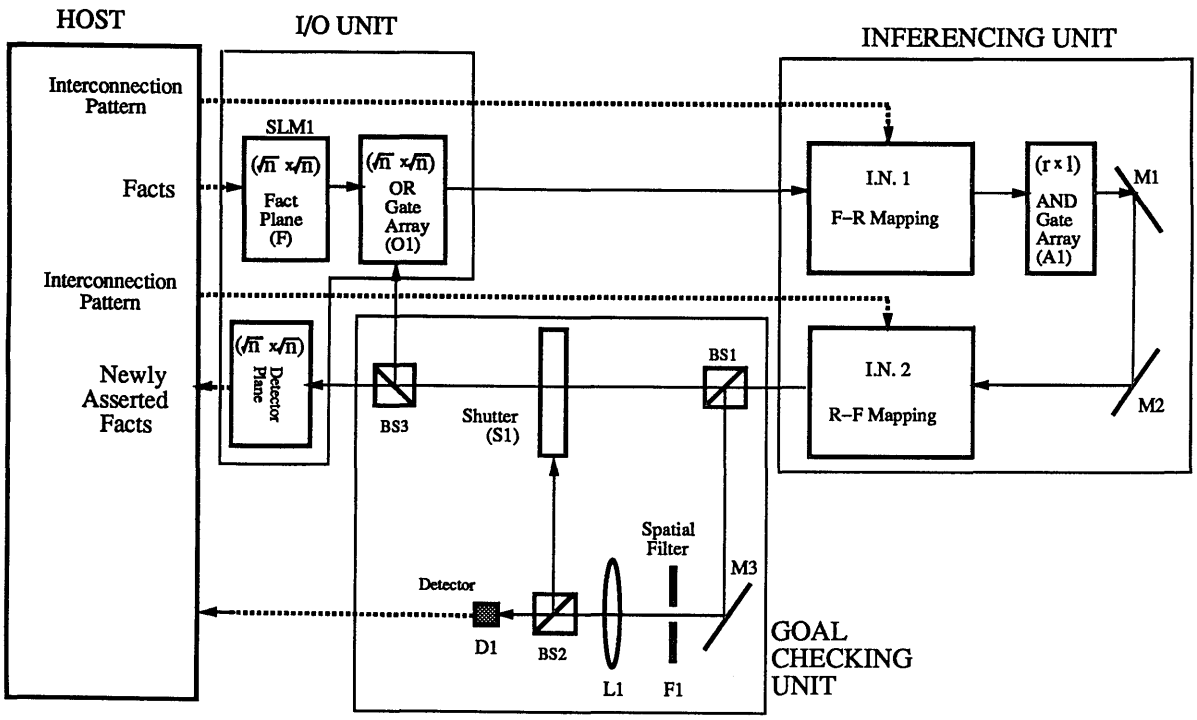


Fig. 6. Implementation of the optical RBS: n, number of facts; r, number of rules; l, number of condition elements per rule; dashed arrows, electrical signal; solid arrow, optical signal.

can be used for the fact planes.[25] For the electrically addressed SLM, a ferro-electric liquid-crystal SLM, an electrically addressed microchannel SLM, or a silicon/lead lanthanum zirconate titonate SLM can be used. Since EORBS must send results to an electronic host, an optical detector array is needed to convert optical signals into electronic signals. EORBS also needs 2-D optical logic-gate arrays to perform AND and OR logic functions. A possible candidate for the optical logic-gate array is the self-electro-optic-effect-device family of devices.[26] In what follows, we examine the optical implementation of each unit needed in EORBS.

A. Input–Output Unit

The input–output unit is the interface unit between the electronic host and the optical rule-based system (ORBS). The input–output unit consists of an SLM, an optical OR-gate array O_1, and a 2-D detector array. The SLM is used to load the input fact plane (DFA) to the ORBS. The OR-gate array O_1 receives two input planes, an initial fact plane from the host and a processed fact plane from the ORBS. Thus, by superimposing the two planes, O_1 can generate an updated input fact plane at every cycle for inference operations in the IU. The detector array is used to send the processed fact plane to the host.

B. Inferencing Unit

The IU is composed of an optical AND-gate array A_1 and two interconnection networks, IN_1 and IN_2. The interconnection network IN_1 performs the F-R mapping, and IN_2 performs the R-F mapping, as described in subsection 3.D. IN_1 uses the fact plane as an input and generates the condition plane. The AND-gate array A_1 receives the condition plane from IN_1 and produces a column vector that is used as the action plane. This vector is sent to the IN_2 for asserting the new facts into the fact plane for the next iteration and to the GCU for goal-checking purposes.

Optical Implementation of Fact-to-Rule and Rule-to-Fact Mapping

The optical implementation of the F-R mapping requires a 2-D to 2-D network since we are mapping the 2-D fact plane to the 2-D condition plane. The R-F mapping requires a 1-D to 2-D network that maps an element from the 1-D action plane to some elements of the 2-D fact plane. The required interconnection patterns in the F-R/R-F mappings are any-to-any connections. This means that any element of the input plane should be able to access any element of the output plane. This can be achieved by several techniques.[27–33] A possible optical implementation that uses space-invariant holograms is proposed below.

Holographic Interconnection Network

As an example, let us consider the following facts and rules:

facts F_1, F_2, F_3, F_4;

rule 1 is, if $(F_1 \wedge F_2)$, then F_3;

rule 2 is, if $(F_1 \wedge F_3)$, then F_4;

rule 3 is, if $(F_2 \wedge F_3)$, then F_4.

Also, assume that we have a 2 × 2 fact plane and a 3 × 3 condition plane. Figures 7(a), 7(b), and 7(c) describe the RFA, CA, and AA, respectively. Then, using the FRMT algorithm, we can obtain the FRMT shown in Fig. 7(d). The FRMT indicates the following interconnection patterns:

$$(F_1\text{–}C_{1,1})(F_1\text{–}C_{2,1}),$$

$$(F_2\text{–}C_{2,1})(F_2\text{–}C_{3,1}),$$

$$(F_3\text{–}C_{2,2})(F_3\text{–}C_{3,2}).$$

The holographic-interconnection-network implementation of this mapping is shown in Fig. 8. The major components of this interconnection network are hologram H_1, an SLM, and demagnifying lens L_{i1}. The hologram is space invariant, such that each facet acts as a one-to-nine beam splitter. Since all the connections are not necessary, an SLM is used after H_1 to block the unnecessary connections. Lens L_{i1} is used for imaging and demagnification purposes. It reduces the 6 × 6 intermediate plane to a 3 × 3 output plane conforming to the size of the condition plane. Figure 8 shows the permitted and the blocked connections as transparent and dark pixels, respectively, on the SLM. In general, the space–bandwidth product (SBWP) of H_1 and L_{i1} depend on the given facts and rules. With a $\sqrt{n} \times \sqrt{n}$ fact plane and a $r \times l$ condition plane, the SBWP of H_1 is $O(rl)$ and that of L_{i1} is $O(nrl)$, where n is the number of facts, r is the number of rules, and l is the number of condition elements.

As described in subsection 3.D, the R-F mapping must compare the action elements with the facts. Using the same example as above and the RFMT algorithm, we generate the RFMT shown in Fig. 7(e). The RFMT indicates the following interconnection patterns between the action plane and the fact plane:

$$(A_1\text{–}F_3),\ (A_2\text{–}F_4),\ (A_3\text{–}F_4).$$

Figure 9 depicts a holographic-interconnection-network implementation of IN_2 that provides such a mapping. IN_2 is implemented by a hologram H_2, a lens L_{i2}, and an SLM. Hologram H_2 in IN_2 is also a multifacet hologram and has the same size as the action plane. Therefore H_2 contains three facets, and each facet acts as a one-to-four beam splitter, as shown in Fig. 9.

As in IN_1, an SLM is needed to block unwanted connections. Lens L_{i2} is used to demagnify the image generated by H_2 and the SLM. The SBWP of hologram H_2 is $O(rm)$ and that of L_{i2} is $O(nrm)$, where m is the number of action elements per rule. Note that there are many different ways of achieving these mappings optically.

C. Goal-Checking Unit

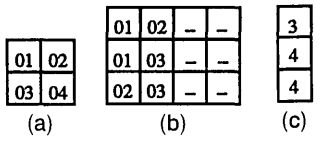
The GCU consists of lens L_1, spatial filter F_1, shutter S_1, three beam splitters (BS_1, BS_2, and BS_3), and detector D_1. At the output stage of the interconnection network IN_2, the fact plane is divided into two fact planes by beam splitter BS_1. One is sent to shutter S_1 and the other to mirror M_3. The fact plane routed to M_3 is filtered spatially by F_1 so that only the goal fact portion of the fact plane arrives at lens element L_1. Then the goal fact plane is collimated by L_1 upon detector D_1 and by the control input to shutter S_1. When the input to D_1 is high, D_1 signals to the host computer that the goal has been found. At the same time, the high control signal disables the IU from further processing by disabling S_1. On a low signal, the low control signal to D_1 enables S_1, which in turn permits further inferencing in the IU.

5. Detailed Operation of Electro-optical Rule-Based System

For an illustration of the operation of EORBS, consider the example of the family tree of Fig. 10(a). The current facts consist of two relationships: is-married-to and is-father-of, as shown in Fig. 10(b). The rules for this system are shown in Fig. 10(c). The goal of this example is to find who is the grandmother of whom (e.g., is-grandmother-of).

Using the rules shown in Fig. 10(c) and the translation algorithm presented in Subsection 3.A, the host computer generates the PT, RFA, CA, and AA [Figs. 10(e) and 10(f)]. Next, the DFA is generated using the RFA of Fig. 10(f). If a fact is currently known, then its corresponding location in the fact plane is set to 1, while others are set to 0. Note that, among the elements of the DFA, the host reserves the last row for the goal facts. The goal of this example is to find a grandmother. Therefore the host assigns the goal facts (F_{18} and F_{19}) in the last row of the RFA. The resulting DFA is shown in the table of Fig. 11(a).

Then, to provide the means for the F-R mapping and the R-F mapping described in Subsection 3.D, the



Source (Fact Element)	Destination (Condition Element)		
01	C(1,1)	C(2,1)	–
02	C(2,1)	C(3,1)	–
03	C(2,2)	C(3,2)	–

(d)

Source (Action Element)	Destination (Fact Element)
01	3
02	4
03	4

(e)

Fig. 7. Example of the (a) RFA, (b) CA, (c) AA, (d) FRMT, and (e) RFMT.

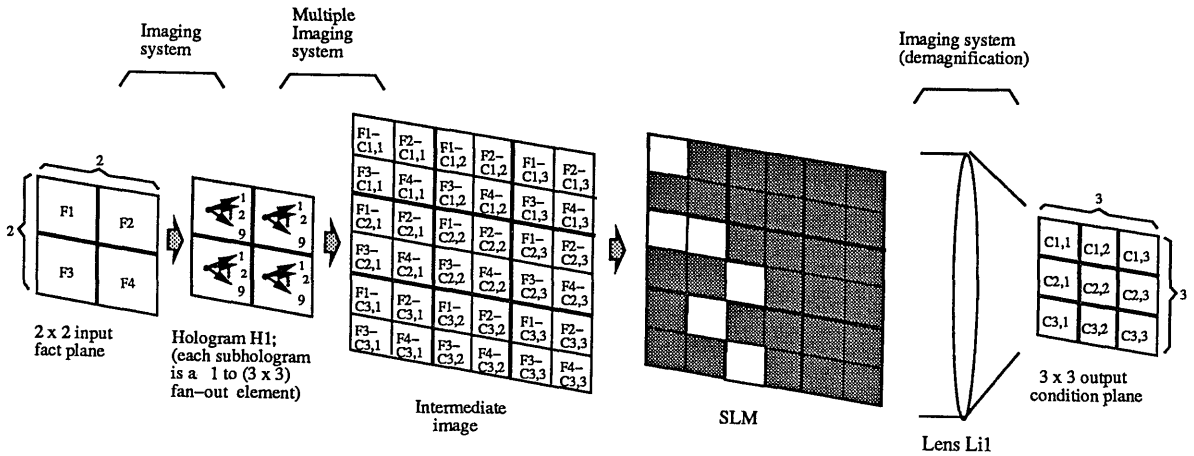


Fig. 8. Holographic interconnection network for F-R mapping.

host computes the F-R/R-F mapping and sends it to the IU. Using the RFA and the CA of Fig. 10(f) as inputs, the FRMT algorithm computes the required interconnection patterns. For example, fact F_1 is used as $C_{1,1}$ and $C_{2,1}$ of the condition plane. Thus EORBS must be able to connect F_1 to $C_{1,1}$ and $C_{2,1}$. The complete F-R mapping is shown in Fig. 12(a).

To generate the R-F mapping, we use the AA and the RFA of Fig. 10(f) as inputs to the RFMT algorithm. For example, the first action element A_1 of the action plane is used as the element F_{10} of the fact plane in Fig. 10(e). Hence we need a connection for the first action element A_1 to the fact element F_{10}. In a similar fashion, the complete R-F mapping is generated [Fig. 12(b)].

Once the DFA and the R-F/F-R mapping information is ready, this complete information is loaded into the optical fact plane and the interconnection networks IN_1 and IN_2. Then, EORBS starts its operation.

Interferencing unit (F-R mapping). Initially, the fact plane (DFA) is supplied by the host. However, in subsequent iteractions, the fact plane includes newly asserted facts from previous iterations by using the optical OR-gate array O_1. In this example, each element of the DFA is transferred into a predetermined location of the condition plane by using the FRMT described in Fig. 12(a). Figure 11(b) shows the resulting condition plane. The rightmost two unused columns are set to a logical 1. IN_1 produces the column vector of Fig. 11(c). In the column vector, a logical 1 in the ith location represents a triggered rule.

Inferencing unit (R-F mapping). If there are rules to be fired, the IU executes the action part of the triggered rules by using the RFMT. The triggered rules are mapped onto the fact plane with IN_2. In this example, each triggered rule represented in Fig. 11(c) is mapped onto the new fact plane of Fig. 11(d) with the RFMT shown in Fig. 12(b).

Goal checking. The GCU receives the DFA from the IU. Upon receiving the DFA [see Fig. 11(d)], the goal fact portion of the DFA (shaded pixels of the DFA) is routed to detector D_1 and shutter S_1 by lens L_1. A high signal to D_1 and S_1 implies that a goal has been found, while a low signal implies that no goal has been found at this time. In this example, the control signal to D_1 and S_1 is low, which means that no goal fact has been asserted. This ends the first iteration of EORBS.

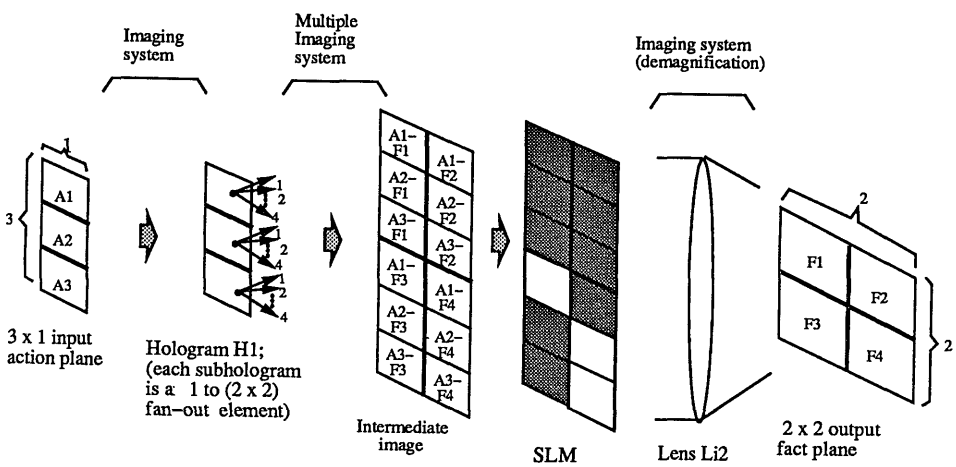
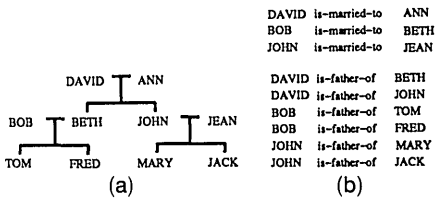


Fig. 9. Holographic interconnection network for R-F mapping.



R1 : IF (DAVID is-married-to ANN) AND (DAVID is-father-of BETH) THEN (ANN is-mother-of BETH).
R2 : IF (DAVID is-married-to ANN) AND (DAVID is-father-of JOHN) THEN (ANN is-mother-of JOHN).
R3 : IF (BOB is-married-to BETH) AND (BOB is-father-of TOM) THEN (BETH is-mother-of TOM).
R4 : IF (BOB is-married-to BETH) AND (BOB is-father-of FRED) THEN (BETH is-mother-of FRED).
R5 : IF (JOHN is-married-to JEAN) AND (JOHN is-father-of MARY) THEN (JEAN is-mother-of MARY).
R6 : IF (JOHN is-married-to JEAN) AND (JOHN is-father-of JACK) THEN (JEAN is-mother-of JACK).
R7 : IF (DAVID is-father-of JOHN) AND (JOHN is-father-of MARY) THEN (DAVID is-grandfather-of MARY).
R8 : IF (DAVID is-father-of JOHN) AND (JOHN is-father-of JACK) THEN (DAVID is-grandfather-of JACK).
R9 : IF (ANN is-mother-of BETH) AND (BETH is-mother-of TOM) THEN (ANN is-grandmother-of TOM).
R10 : IF (ANN is-mother-of BETH) AND (BETH is-mother-of FRED) THEN (ANN is-grandmother-of FRED).

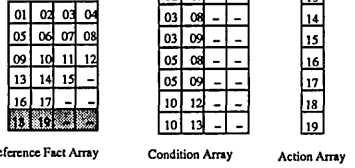
(c)

R1 : IF A1 AND B1 THEN C1.
R2 : IF A1 AND B2 THEN C2.
R3 : IF A2 AND B3 THEN C3.
R4 : IF A2 AND B4 THEN C4.
R5 : IF A3 AND B5 THEN C5.
R6 : IF A3 AND B6 THEN C6.
R7 : IF B2 AND B5 THEN D1.
R8 : IF B2 AND B6 THEN D2.
R9 : IF C1 AND C3 THEN E1.
R10 : IF C1 AND C4 THEN E2.

(d)

Condition elements			Propositions	
Lc	Oc	Rc	s	Vc
DAVID	is-married-to	ANN	01	T
BOB	is-married-to	BETH	02	T
JOHN	is-married-to	JEAN	03	T
DAVID	is-father-of	BETH	04	T
DAVID	is-father-of	JOHN	05	T
BOB	is-father-of	TOM	06	T
BOB	is-father-of	FRED	07	T
JOHN	is-father-of	MARY	08	T
JOHN	is-father-of	JACK	09	T
ANN	is-mother-of	BETH	10	F
ANN	is-mother-of	JOHN	11	F
BETH	is-mother-of	TOM	12	F
BETH	is-mother-of	FRED	13	F
JEAN	is-mother-of	MARY	14	F
JEAN	is-mother-of	JACK	15	F
DAVID	is-grandfather-of	MARY	16	F
DAVID	is-grandfather-of	JACK	17	F
ANN	is-grandmother-of	TOM	18	F
ANN	is-grandmother-of	FRED	19	F

(e)

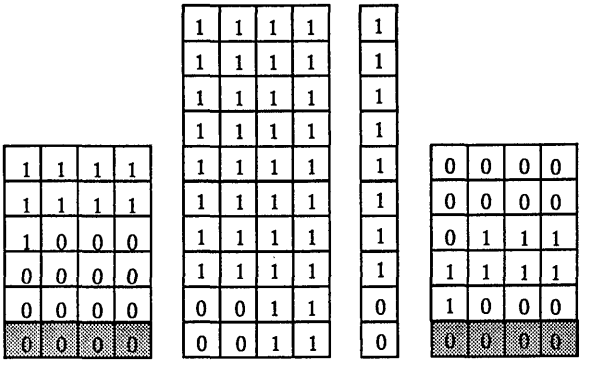


(f)

Fig. 10. Rules and facts for the family tree of David and Ann: (a) diagram of family-tree example, (b) facts known to users, (c) rules from users, (d) translated rules, (e) PT, (f) rules and facts in table form.

Second iteration. In the second iteration, the DFA [Fig. 11(d)] is combined with the original facts [Fig. 11(a)], which are given by the host. As in the first iteration, the new DFA is routed to the IU to perform F-R mapping. Then, the results of the F-R mapping of Fig. 13(b) are AND'ed row-wise. The result is shown in the column vector of Fig. 13(c). Next, using the AA, the IU performs an R-F mapping and produces the DFA shown in Fig. 13(d).

The GCU routes the goal facts (shaded pixels) of the DFA [Fig. 13(d)] to detector D_1. Since there is one pixel (F_{18}) that has a value of 1, the GCU sends a high



(a) (b) (c) (d)

Fig. 11. First run of EORBS: (a) fact plane (DFA with initial facts), (b) output of IN_1 (CA), (c) output of AND-gate array of the IU (AA), (d) new fact plane (DFA at end of first iteration).

Source (Fact Element)	Destination (Condition Element)		
01	C(1,1)	C(2,1)	–
02	C(3,1)	C(4,1)	–
03	C(5,1)	C(6,1)	–
04	C(1,2)	–	–
05	C(2,2)	C(7,1)	C(8,1)
06	C(3,2)	–	–
07	C(4,2)	–	–
08	C(5,2)	C(7,2)	–
09	C(6,2)	C(8,2)	–
10	C(9,1)	C(10,1)	–
11	C(9,2)	–	–
12	C(10,2)	–	–

(a)

Source (Action Element)	Destination (Fact Element)
01	10
02	11
03	12
04	13
05	14
06	15
07	16
08	17
09	18
10	19

(b)

Fig. 12. F-R mapping and R-F mapping: (a) FRMT and (b) RFMT for family tree of David and Ann.

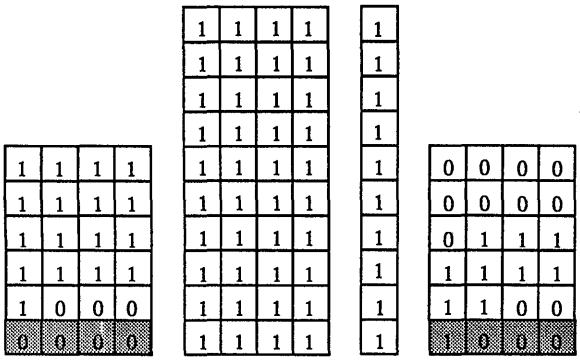


Fig. 13. Same as Fig. 11 but for second run.

control signal to the host and terminates the operation of EORBS by disabling shutter S_1. As soon as the host receives the high control signal (goal-found signal), it reads in the last fact plane and converts it to electronic signals by using the RFA shown in Fig. 10(f) to answer the query.

6. Performance Analysis of Electro-optical Rule-Based System

Here we estimate the theoretical performance of the proposed architecture by estimating its execution speed and by comparing it with that of conventional RBS's. In what follows we use the following terms and assumptions for estimating the execution speed:

(1) We assume that the response time of optical gate arrays used for logic functions (e.g., optical AND, OR, and optical shutter) is T_r.
(2) The setup time of the SLM of size n is T_s.
(3) The processing time of transferring elements of one DFA to another is T_p.
(4) The access time of the detector plane of the I/O unit is T_d.
(5) The propagation delay of optical passive devices such as lenses, mirrors, beam splitters, and holograms is negligible.
(6) We assume that the number of elements in the fact plane is n and that the number of condition elements per rule is l.
(7) We ignore the reconfiguration time of IN_1 and IN_2, since these are reconfigured only at the start of each query. While solving a query, the interconnection networks can be considered fixed.

A. Execution Speed

The execution speed of EORBS can be thought of as being divided into three parts: initialization time T_{init}, execution time T_{ex}, and output fetch time T_{fetch}. In order to estimate the execution time, we consider the longest signal path through EORBS.

To complete one iteration, the optical signal must go through the optical OR-gate array (O_1), the IU (one AND-gate array and two SLM's), and one shutter S_1. Such a path contains five active devices that require switching; therefore the total time required for one iteration is $5T_r$.

The initialization time includes the setup times for the fact plane and the SLM's in the interconnection networks IN_1 and IN_2. However, these units can be accessed simultaneously, and hence T_{init} becomes T_s (the setup time for a single SLM). The output fetch time includes the time to read in the detector of the GCU. Thus T_{fetch} is equal to T_d.

The data for the input and output planes of EORBS can be formatted in a 2-D array data structure during the knowledge-base compilation time. Since the compiled rules and facts are used throughout the query processing, the sum of T_{init} and T_{fetch} becomes the actual time for I/O operations required between the host computer and EORBS. Therefore the overall response time of EORBS, T_{response}, becomes

$$T_{\text{response}} = T_{\text{init}} + T_{\text{ex}} + T_{\text{fetch}} = T_s + x \times 5T_r x + T_d, \tag{1}$$

where x represents the number of iterations required to solve a given query.

Here we evaluate the number of rules that can be processed in EORBS. Given the number of condition elements l of a rule, the number of rules that can be represented in EORBS becomes n/l. Since the cycle time of EORBS is $5T_r$, the maximum number of rules R that can be processed per second is

$$R = \frac{n}{5lT_r}. \tag{2}$$

However, if the size of the problem (the number of rules) is greater than the size of the condition plane of EORBS, we must partition the problem into a set of subproblems whose size fits into the condition plane of EORBS. In this case the cycle time must include the time taken to read in the DFA from EORBS (detector access time T_d), the time taken to relocate some elements of the DFA into a new DFA' (host processing time T_p), and the time to load the input DFA' to the I/O unit of EORBS (SLM setup time T_s).

In this study, we assume that the SBWP of the space-invariant hologram of the interconnection network IN_1 is $O(rl)$. With this hologram, any fact element in a DFA can access any location in the condition plane. However, there is a high price to pay for such flexibility, namely, a large fan-out factor. This flexibility can be compromised without degrading the performance of EORBS. One method is limiting the fan-out rl to a small integer α. This is possible because a single fact element is not used as the condition element of all rules in practical RBS's. As an example, the sample RBS used here requires a maximum fan-out of only 3. Even if there exists an RBS that has a fact F_i that is greater than the permitted fan-out α, we can easily duplicate F_i into as many copies as necessary to cover the fan-out requirement.

Limiting the fan-out implies that the number of pixels used of the SLM of the interconnection networks, IN_1 and IN_2, can be reduced. That is, given n,

the number of facts in the fact plane, and given rl, the number of condition elements in the condition plane, we must use an SLM with a SBWP of nrl in IN_1. Also, note that the maximum number of rules and facts that can be represented in EORBS is limited by the product nrl. Therefore, if we can reduce the required SBWP of the SLM, we can increase the number of rules and facts that can be represented in EORBS by using the unused parts of the SLM.

To evaluate R, we assume y represents the ratio of the area that the hologram with the reduced fan-out can cover to the area of the SLM of the interconnection networks. Then, the number of rules that can be represented on EORBS becomes n/yl. Considering this factor, if the size of the problem is greater than the size of EORBS, we see that the maximum number of rules that can be processed per second becomes

$$R = \frac{n}{yl(5T_r + T_s + T_p + T_d)}. \quad (3)$$

As an example, if we assume $n = 10^3$, $l = 5$, $y = 0.1$, $T_r = 10^{-6}$ s, and $T_s = T_p = T_d = 10^{-3}$ s, EORBS can execute $\sim 6.65 \times 10^5$ rules. In order to better appreciate this speed advantage of EORBS, we compare the execution speed of EORBS with the electronic systems. The DADO2 multiprocessor system is estimated to process ~ 70 rules/s.[3] The NON-VON is a multiprocessor system that is composed of 32 powerful large processing elements and 16,000 small processing elements, and it is estimated to process 850 rules/s.[34] With processing capability of 6.65×10^5 rules/s, EORBS is expected to achieve a system throughput far better than any electronic RBS can achieve.

B. Discussion

The performance of EORBS is primary limited by the availability of optical devices with large SBWP and fast response time. The large SBWP permits the optical RBS to represent an increased number of rules, and the fast response time decreases the time taken for a single iteration in the optical RBS. The limited fan-out concept is introduced to increase the number of rules that EORBS can represent with a given SBWP of an SLM. Along with these requirements, the optical power is also an important problem owing to the fan-out hologram used in the interconnection network. Because of the beam-spreading operation of the fan-out hologram, the fan-out hologram is the major power-consumption factor. However, the limited fan-out concept can also reduce the problem incurred by the beam-spreading operation by controlling the number of necessary output destinations. The limited fan-out can also relieve the device requirements. For example, if we reduce the required target area of a beam, the beam-uniformity condition is greatly relieved. Also, since a small number of pixels of the SLM are routed to a pixel of the condition plane (IN_1), the contrast ratio of the SLM can be decreased accordingly.

7. Conclusions

While rule-based programming and RBS's are popular and advantageous for solving problems in the artificial intelligence domain, particularly in expert systems, their slow execution speed has limited their extensive use. Conventional electronic parallel machines contain fast processors, but have limited communication bandwidth, which is critical in parallel processing. Optics can provide the large communication bandwidths and faster execution speeds required for RBS's.

We have explored an electro-optical rule-based system architecture. The architecture exploits the natural parallelism of optics and the advantages of optical interconnects. A distinctive feature of EORBS is the concurrent rule execution, a feature absent in electronic RBS's. In addition, EORBS provides an efficient implementation of the basic operations required for a RBS; namely, selection, matching, and rule firing. The concurrent rule execution nature of EORBS, combined with the speed with which the basic operations are implemented in EORBS, can potentially result in a hybrid RBS with significant performance improvements over any existing RBS's.

This research was supported by National Science Foundation grant MIP-9113688.

References

1. S. J. Stolfo and D. P. Miranker, "DADO: a parallel processor for expert systems," in *Proceedings of the International Conference on Parallel Processing*, R. M. Keller, ed. (Institute of Electrical and Electronics Engineers, New York, 1984), pp. 74–82.
2. S. J. Stolfo, "Initial performance of the DADO2 prototype," Computer **20,** 75–83 (1987).
3. A. Gupta, "Implementing OPS5 production system on DADO," in *Proceedings of the International Conference on Parallel Processing*, R. M. Keller, ed. (Institute of Electrical and Electronics Engineers, New York, 1984), pp. 83–91.
4. A. Louri, "3-D optical architecture and data-parallel algorithms for massively parallel computing," IEEE Micro **11,** 25–84 (1991).
5. P. B. Berra and A. Ghafoor, "The impact of optics on data and knowledge base systems," IEEE Trans. Knowl. Data Eng. **1,** 111–131 (1989).
6. A. Guha and M. W. Derstine, "Designing massively parallel optical computers: a case study," Appl. Opt. **29,** 2187–2200 (1990).
7. C. Warde and J. Kottas, "Hybrid optical inference machine: architectural considerations," Appl. Opt. **25,** 940–947 (1986).
8. A. B. VanderLugt, "Signal detection by complex spatial filtering," IEEE Trans. Inf. Theory **IT-10,** 2 (1964).
9. J. Y. Jau, F. Kiamilev, Y. Fainman and S. H. Lee, "Optical expert system based on matrix-algebra formulation," Appl. Opt. **27,** 5170–5175 (1988).
10. R. A. Schmidt and T. Cathey, "Optical implementations of mathematical resolution," Appl. Opt. **26,** 1852–1858 (1987).
11. D. J. Willshaw and H. C. Longuet-Higgins, "Associative memory models," Machine Intell. **5,** 351 (1970).
12. G. Eichmann and H. J. Caulfield, "Optical learning (inference) machines," Appl. Opt. **24,** 2051–2054 (1985).
13. H. J. Caulfield, "Optical inference machines," Opt. Commun. **55,** 259–260 (1985).

14. H. H. Szu and H. J. Caulfield, "Optical expert systems," Appl. Opt. **26,** 1943–1947 (1987).
15. A. D. McAuley, "Real-time optical expert systems," Appl. Opt. **26,** 1927–1934 (1987).
16. D. Casasent and E. Botha, "Optical correlator production system neural net," Appl. Opt. **31,** 1030–1040 (1992).
17. F. Hayes-Roth, "The knowledge-based expert system: a tutorial," Computer **17,** 11–28 (1984).
18. P. Harmon and D. King, *Expert Systems: Artificial Intelligence in Business* (Wiley, New York, 1985), Sec. 1.
19. P. H. Winston, *Artificial Intelligence* (Addison-Wesley, Reading, Mass., 1984), Chap. 6.
20. L. Brownston, R. Farrell, E. Kant, and N. Martin, *Programming Expert Systems in OPS5* (Addison-Wesley, Reading, Mass., 1985), Chap. 1.
21. K. Hwang and D. Degroot, *Parallel Processing for Supercomputers and Artificial Intelligence* (McGraw-Hill, New York, 1989), Chap. 7.
22. G. E. Blelloch, "CIS: a massively concurrent rule-based system," in *Fifth National Conference on Artificial Intelligence,* T. Kehler and S. Rosenschein, eds. (American Association for Artificial Intelligence, Philadelphia, Pa., 1986), pp. 735–741.
23. E. Rich, *Artificial Intelligence* (McGraw-Hill, New York, 1983), Chap. 3.
24. L. J. Huang and M. Tomizuka, "A self-paced fuzzy tracking controller for two-dimensional motion control," IEEE Trans. System. Man and Cybern. **20,** 1115–1124 (1990).
25. J. A. Neff, R. A. Athale, and S. H. Lee, "Two-dimensional spatial light modulators: a tutorial," Proc. IEEE **78,** 826–855 (1990).
26. A. L. Lentine, H. S. Hinton, D. A. B. Miller, J. E. Henry, J. E. Cunningham, and L. M. F. Chirovsky, "Symmetric self-electrooptic effect device: optical set-reset latch differential logic gate, and differential modulator/detector," IEEE J. Quantum Electron. **25,** 1928–1936 (1989).
27. B. K. Jenkins, P. Chavel, R. Forchheimer, A. A. Sawchuk, and T. C. Strand, "Architectural implications of a digital optical processor," Appl. Opt. **23,** 3465–3474 (1984).
28. J. Taboury, J. M. Wang, P. Chavel, F. Devos, and P. Garda, "Optical cellular processor architecture. 1: Principles," Appl. Opt. **27,** 1642–1650 (1990).
29. P. Yeh, A. E. T. Chiou, and J. Hong, "Optical interconnection using photorefractive dynamic holograms," Appl. Opt. **27,** 2093–2095 (1988).
30. S. D. Smith, A. C. Walker, M. R. Taghizadeh, I. R. Redmond, and B. Robertson, "The optical wiring of photonic digital processing array for optical computing," in *Optical Computing '88,* P. Chavel, J. W. Goodman, and G. Roblin, eds., Proc. Soc. Photo-Opt. Instrum. Eng. **963,** 414–418 (1988).
31. A. Hartmann and S. Redfield, "Design sketches for optical crossbar switches intended for large-scale parallel processing applications," Opt. Eng. **28,** 315–327 (1989).
32. A. A. Sawchuk, B. K. Jenkins, C. S. Raghavendra, and A. Varma, "Optical crossbar networks," Computer **20,** 50–60 (1987).
33. A. A. Sawchuk and B. K. Jenkins, "Dynamic optical interconnections for parallel processors," in *Optical Computing,* J. A. Neff, ed., Proc. Soc. Photo-Opt. Instrum. Eng. **625,** 143–153 (1986).
34. B. W. Wah and G. J. Li, "Design issues of multiprocessors for artificial intelligence," in *Parallel Processing for Supercomputers,* K. Hwang and D. Degroot, eds. (McGraw-Hill, New York, 1989), Chap. 4, pp. 107–159.
35. S. K. Case, P. R. Haugen, and O. J. Lokberg, "Multifacet holographic optical elements for wave front transformations," Appl. Opt. **20,** 2670–2675 (1981).