NEW HARDWARE CIRCUITS FOR THE IMPLEMENTATION OF LOGICAL RELATIONS IN INFORMATION PROCESSING—PART I AND II*
PART I*

G. N. RAMACHANDRAN
INSA Albert Einstein Professor, Mathematical Philosophy Group,
Indian Institute of Science, Bangalore 560012, India.

ABSTRACT

This report, presented in two parts, deals with some novel circuitry involving logic gates which can represent very faithfully the vector-matrix representation of Boolean algebra and statements in propositional calculus (PC), quantified predicate logic (QPL) and multivalued logic (MVL). The circuits that are presented will represent unary and binary relations involving $m \times n$ matrices in the general theory of relations, which is extended to circuits specially designed for answering complicated logical queries in information processing in general. One such shown in figure 4 can perform the simultaneous truth checking of n-relations of the form $a_j \mathbf{R} l_j$ where $\mathbf{R}$ can be one of $>$, $=$, $<$, $\geqslant$, $\leqslant$, as well as $l_1 < a < l_2$, l_1 or $l_2 \ldots\ldots$ or $l_k = a$, purely in hardware. Novel circuits that will delete the output of some of the relations $a_j \mathbf{R} l_j$ are presented. An elegant circuit for checking $>$, $=$, $<$, of two numbers given in the binary system appears to be very promising and capable of being used even in conventional computers. The circuitry given for a general relation of the type $\bigvee_{j=1 \text{ to } n} (a_j \text{ REL } b_j) = c$, and $\bigwedge_{j=1 \text{ to } n} (a_j \text{ REL } b_j) = c$ can yield the truth value of the l.h.s. by the Boolean value (1 to 0) of c.

The circuits presented will work very well for use with array processors or for parallel processing in suitably designed computers. The design of these circuits were arrived at as a consequence of the application of the Boolean algebraic representation of the theory of relations (and their reversal and complementation,) worked out by the author, and therefore a brief outline of this theory is given in Section 2.

1. INTRODUCTION

THE novel circuitry that are presented in this paper were the consequence of the studies made by the author on the vector-matrix representation of Boolean algebras (denoted by BVMF-Boolean vector-matrix formalism) and its application for solving problems in propositional calculus (PC), quantified predicate logic (QPL) and multivalued logic (MVL)[1, 2]. The application of Boolean matrices to the general theory of relations with special reference to the answering of complicated logical queries is contained in an unpublished report[3], which is being written for publication. This study led to the discovery of interesting new hardware circuits and these are presented in this paper. The most significant of these is that in figure 4, capable of application in information processing, which is quite general, and capable of answering any queries from a list of attributes. This is treated in Section 4, and some new basic circuits that can be applied for implementing the general *logical* properties of MVL are discussed in Section 5.

Although this paper deals essentially with hardware circuits, the understanding of their principle as regards the computer implementation of the theory of relations requires a knowledge of the essentials of our BVM formalism for MVL, and therefore the essential formulae of this formalism are presented in Section 2. Section 3 contains an account of the practical implementation of the BVMF—in the form of circuits in Section 3(a), with a table containing problems to which these circuits can be applied in Section 3(b).

As a result of obtaining familiarity with the type of logical circuits which became evident in this study, some two or three similar problems in multi-valued logic also were found to be very well implemented on circuits analogous to those developed for the theory of

---

* Part II of this article will appear in January 20, 1986 issue of *Current Science*.

relations. The most general of these which can deal with most of the problems containing queries in information processing procedure is described in Section 4 with special reference to the hardware.

Although these circuits have not yet been constructed and checked, simple vector-matrix equations containing two and three variables have been implemented in the form adopted by us, and there is no reason why larger matrices cannot be implemented in practice, with closely similar lines of approach in the design of the hardware circuit. It is the belief of the author that the general circuits described in Sections 4 and 5 will be of great utility in computer science. Although they require a large amount of hardware, this is over-shadowed by the great simplification of the procedural equations in programing, as well as in the large variety of problems in information processing that can be implemented in these circuits.

It should be mentioned that this paper is written in a pedagogic style and simple examples are presented when needed as illustrations, and the most general cases are considered only in Sections 3, 4 and 5. Also the paper is self-contained, and section 2 contains a brief summary of the formulation (BVMF) of MVL which we have adopted. For editorial reasons, this article is published in two parts in two consecutive issues of the journal. However, the presentation is continuous (*e.g.* in Section Nos., Equation Nos. etc.,) and a common abstract is given in the beginning of Part I, with a common list of references at the end of Part II. Part I deals essentially with BVMF and associated circuits (essentially connected with logical operations) while Part II is concerned more with hardware circuits some of which may find application even in conventional computers.

2. UNARY AND BINARY RELATIONS IN BVMF

(*a*) *Vectors and matrices*: Suppose that we have two sets $\mathscr{A} \equiv \{a_1, a_2, \ldots, a_m\}$ and $\mathscr{B} \equiv \{b_1, b_2, \ldots, b_n\}$ which are related by the $m \times n$ Boolean matrix $\mathbf{R} \equiv |R_{ij}|$, where R_{ij} is equal to 1 if the relation $a_i \mathbf{R} b_j$ is true and 0 otherwise. Following[2], a subset A of $\mathscr{A}$ having for example, the elements $\{a_1, a_3, \ldots, a_m\}$ is represented by a $1 \times m$ Boolean row vector (1 0 1 0 . . . 1), which is designated by either of the symbols a or $\langle a|$. More generally, the m-vector a (representing the subset A of $\mathscr{A}$) has 1's corresponding to the i-values of those a_i of the full set $\mathscr{A}$ that are present in the subset A, and 0's corresponding to the i-values of the elements of $\mathscr{A}$ which are absent in the chosen subset A, with similar conditions for the definition of the n-vector b or $\langle b|$ representing the subset B of $\mathscr{B}$. Obviously, the m-vector corresponding to the full set $\mathscr{A}$ has 1's *for all the m elements* a_i, and this is indicated by the symbol 1. Similarly the null set of $\mathscr{A}$ is denoted by 0, and it has $a_i = 0$ for all $i = 1$ to m. Since we often have to refer to a subset and also its representation as a vector, we also refer to the subset A by the vector a or $\langle a|$ instead of the symbol for the subset, namely A. So also, the standard use of the *row* vector such as $\langle a|$ is a very reasonable one from the way in which unary and binary relations are written below (details are given in ref 2,3)

With the above definitions, we can represent a unary relation in the form a **R** b, which in BVM formalism becomes

$$\langle a|\, R = \langle b|; \quad \sum_i a_i\, R_{ij} = b_j \text{ Defn. (1a,b)}$$

It has the property that it gives as output the Boolean value 1 for all the elements b_j, of B, which are related to some element of A, and 0 for all the other b_j's. It should be noted that all the sums and products in Defn. 1 are Boolean.

So also, a binary relation a **R** b = c, where c is a Boolean scalar, is given in Definition 2.

$$\langle a|R|b\rangle = c; \sum_i \sum_j (a_i\, R_{ij}\, b_j) = c;$$

$$c = 0 \text{ or } 1 \qquad \text{Defn. (2a,b)}$$

Expressed briefly, the consequence of the equation in Defn. 2 is that if $\langle a|R|b\rangle = 0$, then *no* a_i is related *via* R_{ij} to *any* b_j and vice versa. If $\langle a|R|b\rangle = 1$, then *some* a_i are related to *some* b_j (but not for none)—See Section 3 for examples of both unary and binary relations as dealt with in BVMF.

It is necessary to define two more matrices related to **R**—namely $\mathbf{R}^t$, for the reverse relation, and $\mathbf{R}^c$, for the complementary relation—as in Defns (3) and (4).

$$\text{Reverse of } \mathbf{R} \quad : (R^t)_{ij} = R^t_{ij} = R_{ij} \qquad (3)$$

$$\text{Complement of } \mathbf{R} \quad : (R^c)_{ij} = R^c_{ij} = 1 - R_{ij} \qquad (4)$$

It is readily verified that $\mathbf{R}^t$ defines the relation $\mathscr{A}\ \mathbf{R}\ \mathscr{B}$ in the reverse sense. Thus

$$\mathscr{A}\, R\, \mathscr{B} \equiv \mathscr{B}\, R^t\, \mathscr{A} \qquad \text{Defn. (5)}$$

Also, $\mathbf{R}^c$ stands for the "non-relation" as in Defn. (6): (See Table 1(a) for an example).

If for the sets $\mathscr{A}$ and $\mathscr{B}$, $a_i\, R_{ij}\, b_j = 0$, then

$$a_i\, R^c_{ij}\, b_j = 1 \text{ and vice versa} \qquad \text{Defn. (6)}$$

Although the matrix $|R^t_{ij}|$ represents the reverse relation from $\mathscr{B}$ to $\mathscr{A}$, the two matrices $|R|$ and $|R^t|$ *do not necessarily satisfy* the equation $|R|R^t| = |R^t|R| \equiv |E|$. Thus, $\langle a|R| = \langle b|$ yields B containing *all* b_j of $\mathscr{B}$ that are related to those a_i that are present in the set A. Then $\langle b|R^t| = \langle a'|$ gives, by a', *all* elements of $\mathscr{A}$ that are related to *every* member of b. The set A' thus obtained will surely contain *all elements of* A, but *may be a larger set—i.e.* $A' \supseteq A$. As an example, if B stands for the nephews of the uncles in A, then $\langle a|R|$ yields all the nephews, but these nephews may have uncles, other than those included in A, so that $A' \supseteq A$. (See ref 3 for a different example, explained more fully.)

The complement a^c of the vector a is defined by (7a):

$$\mathsf{a}^c_i = \text{Complement of } \mathsf{a}\text{: } a^c_i = 1 - a_i,$$

$$i = 1 \text{ to } m \qquad \text{Defn. (7a)}$$

Thus the complement vector a^c contains 1's for all a_i that are absent in the set a, and vice versa. Obviously, $(\mathsf{a}^c)^c = a$ and the set A^c is the complement of the set A in $\mathscr{A}$. Thus, the set-theoretical equations in (7b) lead to the corresponding equations in (7c) for the representative vectors for the sets concerned.

$$A^c = \sim A,\ A \cup A^c = \mathscr{A},\ A \cap A^c = \emptyset \qquad (7b)$$

$$a^c_i = (a_i)^c,\ \mathsf{a} \oplus \mathsf{a}^c = 1,\ \mathsf{a} \otimes \mathsf{a}^c = 0 \qquad (7c)$$

3. PRACTICAL IMPLEMENTATION OF THE VECTOR-MATRIX FORMALISM

(a) *Hardware*

Figures 1 and 2 represent the circuit diagrams corresponding to the implementation of Eqns (1) and (2) for unary and binary relations respectively. It is readily seen that the circuits are faithful representations by logic chips of the logical operations contained on the l.h.s. of (1) and (2). Thus, figure 1 contains the logic chip AND for products such as $a_i R_{ij}$ (which can be written in Boolean algebra as $a_i \otimes R_{ij}$) and the summation contained in Eq. (1) is represented by the logical chips OR and the Boolean algebraic symbol $\oplus$. The equation in (1), expanded, is

$$(a_i \otimes R_{ij}) \oplus (a_2 \otimes R_{2j}) \oplus \ldots \oplus$$

$$(a_m \otimes R_{mj}) = b_j,\ j = 1 \text{ to } n \qquad (8)$$

For simplicity, m and n are both taken to be equal to 3 in figures 1 and 2. All the essential principles involved in constructing the circuit are available in this example, which can be readily generalized to an m-vector $\langle a|$, an n-vector $\langle b|$, and an $m \times n$ matrix $|R|$. We shall explain the circuits with reference to the example given.

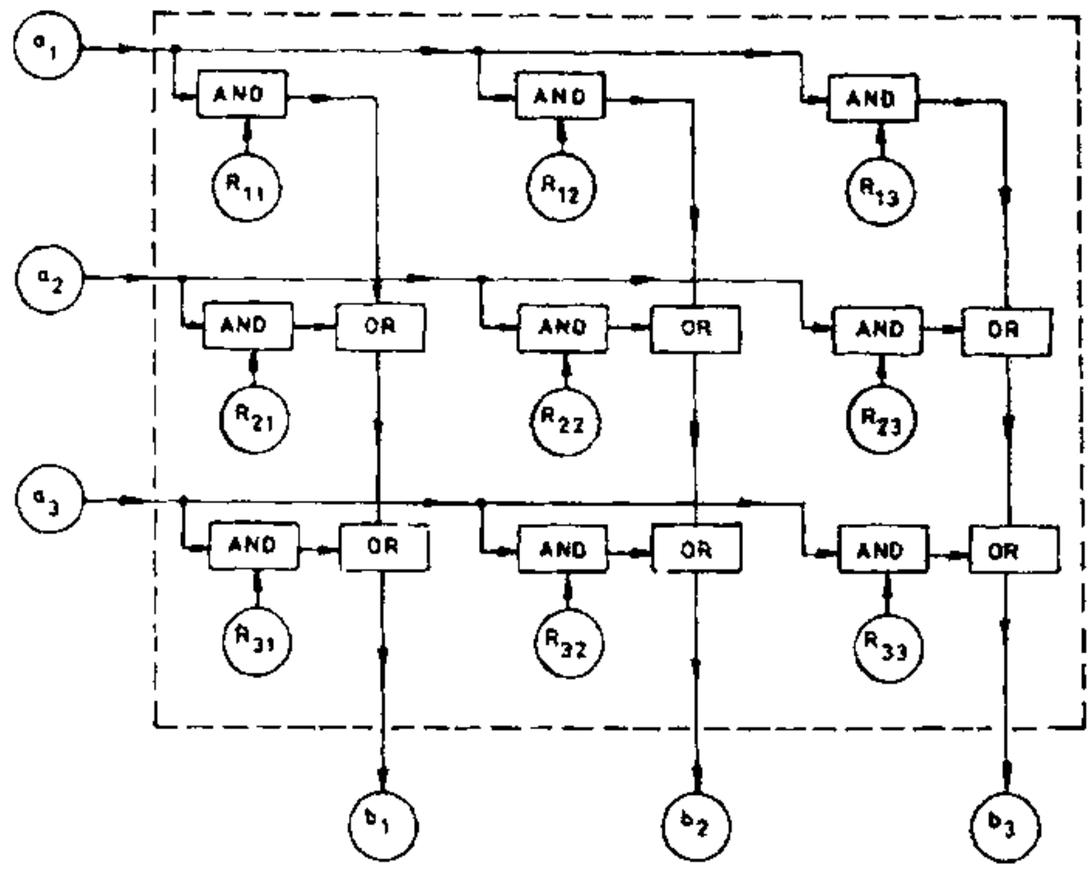

Figure 1. Lay-out of the circuitry for the unary logical relation $\sum_i a_i R_{ij} = b_j$.

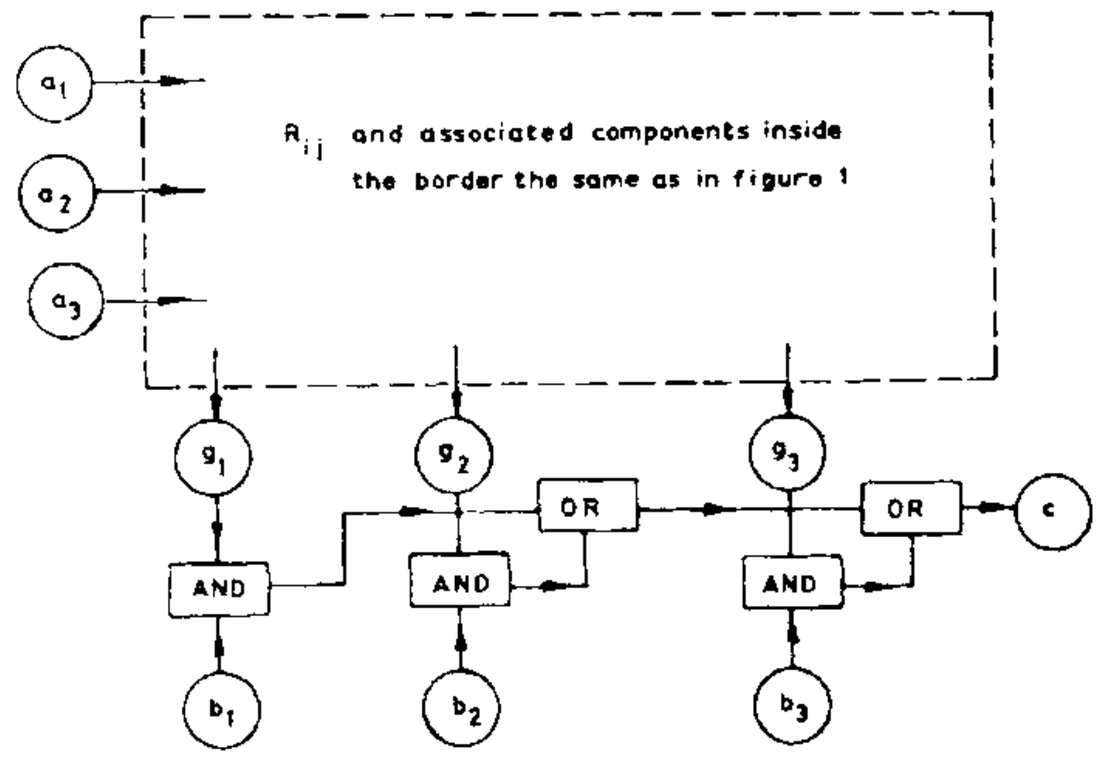

Figure 2(a). Additional circuit required to calculate $\sum_i \sum_j a_i R_{ij} b_j = c$. $\langle g|$ is the intermediate output from figure 1, and in addition, the binary relation $\langle g|E|b\rangle = c$ is implemented.

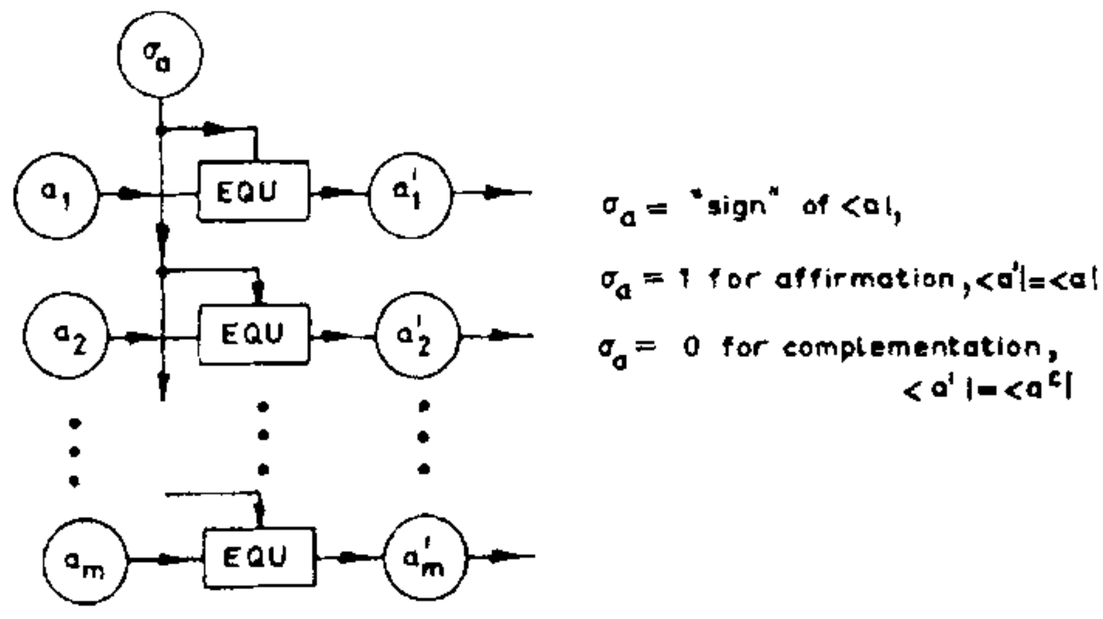

Figure 2(b). Circuit for complementing the m-vector a of figure 2(a) when needed, by setting the sign switch σ_a.

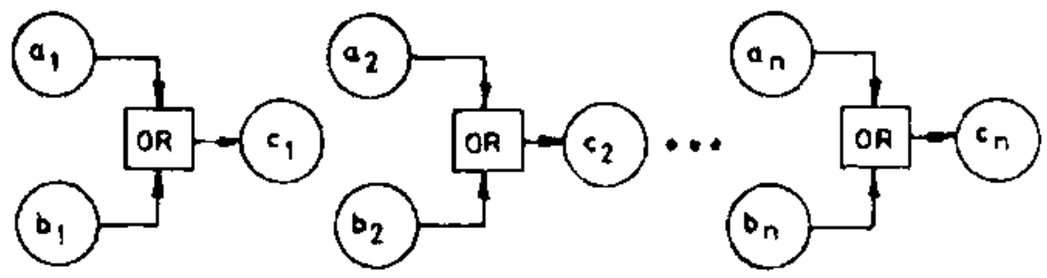

Figure 2(c). Circuit for the union of two sets which implements Eq. (10a).

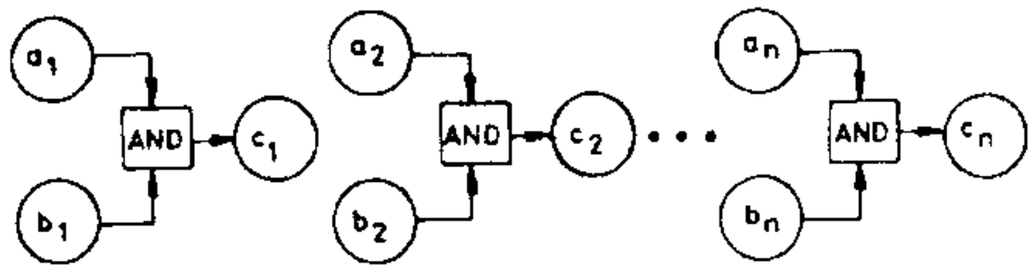

Figure 2(d). Circuit for implementing Eq. (10b) for the intersection of two sets.

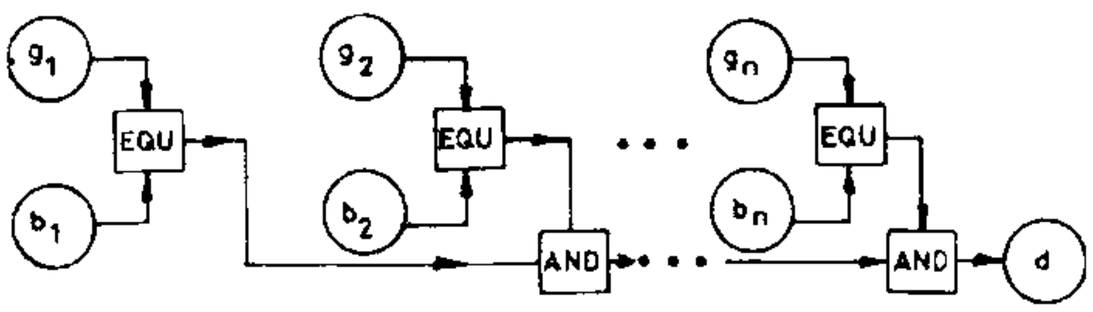

Figure 2(e). Circuit to check the equivalence of two sets G and B, analogous to checking for common elements between them as shown in figure 2(b).

Consider figure 1. The matrix $|R_{ij}|$ corresponding to the relation **R** is given and can be represented as signals having the values 1 or 0 in the $m \times n$ values of R_{ij}. Then, if the Boolean values of a_1, a_2, a_3 are input, all the three Boolean outputs b_1, b_2, b_3 (corresponding to three linear equations) are obtained straightaway and the vector $\langle b|$ becomes available as output. Thus a complicated problem which in conventional computer usage is calculated by a sequence of successive statements, can all be implemented in one step, by inputting all a_i, when all the b_j become available straightaway.

Coming to the implementation of Eq. (2) which is given in figure 2(a), the process is as follows. First, the outcome of the matrix product $\sum_i a_i R_{ij} = g_j$ is constructed in exactly the same manner as in figure 1. We denote the output as **g**. Then, the binary relation $\langle a|R|b\rangle = c$ is equivalent to the Boolean equation (9), in which g_j ($j = 1$ to n) stands for the l.h.s of (8):

$$(g_1 \otimes b_1) \oplus (g_2 \otimes b_2) \oplus \ldots \oplus (g_n \otimes b_n) = c \quad (9)$$

Eq. (9) is faithfully simulated by the logic circuit in figure 2(a) and the two equations (8) and (9) together give the truth value c of the binary relation $\langle a|R|b\rangle = c$. The meaning of this binary relation corresponding to the values of c equal to 0 or 1 is as follows (For more details see ref 3).

(i) If *some* element(s) of A are related by **R** to *some* element(s) of B, then $\langle a|R|b\rangle = 1$ and *vice versa*. (9a)

(ii) If *no* element of A is related to *any* element of B, then $\langle a|R|b\rangle = 0$ and *vice versa*. (9b)

Thus, if there is at least one pair (a_i, b_j) which is related, then it yields $a_i R_{ij} b_j = 1$, and makes one of the terms in the Boolean sum in (9) to be non-zero, so that, $c = 1$. If none are so related, then c is obviously equal to 0. We shall generalize the results in (9a) and (9b) in Section 5.

(*b*) *Typical problems for practical application*

We consider the set of students $\mathscr{S}$ and of professors $\mathscr{P}$ related by the direct relation $\mathrm{s} \xrightarrow{P} \mathrm{p}$ with matrix $|P|$ as its representation and the reverse relation from professors $\mathscr{P}$ to students $\mathscr{S}$ denoted by $\mathrm{p} \xrightarrow{S} \mathrm{s}$, and having the matrix $|S| = |P^t|$. We shall also use the non-relations p^c and S^c as defined in Eq. (4). Using these matrices, examples involving unary relations and their vector-matrix representation are given in table 1(a). Similarly, some examples of applications of binary relations are given in table 1(b).

We shall not comment on these, except to point out that only relations of the type contained in Defns (1) and (2) are employed for this purpose and these can be taken care of by the circuitry in figures 1 and 2(a). In addition, we need operators for logical relations such as union and intersection which are represented in our notation by $\oplus$ and $\otimes$. The implementation of these is done by using Eqns (10a,b) below for two n-vectors:

$$(A \cup B = C) \equiv (\mathrm{a} \oplus \mathrm{b} = \mathrm{c}) \mapsto$$

$$a_j \,\mathrm{OR}\, b_j = c_j, j = 1 \text{ to } n \quad (10a)$$

$$(A \cap B = C) \equiv (\mathrm{a} \otimes \mathrm{b} = c) \mapsto$$

$$a_j \,\mathrm{AND}\, b_j = c_j, j = 1 \text{ to } n \quad (10b)$$

For complementation, we use

$$(\sim A = B) \equiv (a^c = b) \mapsto a_j^c = b_j, j = 1 \text{ to } n \quad (10c)$$

in which $a_j^c = 1 - a_j$ from Eq. (4), yielding $a_j^c = 0$ if $a_j = 1$ and $a_j^c = 1$ if $a_j = 0$. The circuitry for all these and other special circuit elements, or components, are described in Section 3(c). In Section 4, we shall consider the application of our ideas for building a general circuit block (figure 4) which can be applied for

Table 1(a) *Examples illustrating queries connected with unary Boolean relations*

Sl No	Description of the relation employed	Boolean vector-matrix equation
1	Who are all the professors (p) who teach the students in s	$\langle s\|P\| = \langle p\|$
2	Who are all the professors (p) who do not teach any student in s?	$\langle s\|P^c\| = \langle p'\|$
3	Who are all the students (S') attending classes taken by professors that teach the set s	$\langle s\|P\|P^t\| = \langle s'\|$
4	Which professors (p′) in p teach some student in s?	$\langle s\|P\| \otimes \langle p\| = \langle p'\|$
5	Which students (s′) not in the set s (but in $\mathscr{S}$) take classes with professors in the set s?	$\langle p\|P^t\| \langle \otimes \langle s^c\| = \langle s'\|$

Table 1(b) *Examples to illustrate use of binary Boolean relations*

Sl No	Description of the relation considered	BVM-equation for the condition to be checked
1	Check if at least one student of s is related by P to one of p	$\langle s\|P\|p\rangle = 1$
2	Check if all members of set p are not related by S to any of the students s.	$\langle p\|S^c\|s\rangle = 0$
3	Check if the professors p teaching s have any members in common with the professors p′ teaching s′.	$\langle s\|P\|P^t\|s'\rangle = 1$
4	In serial number 3 if the answer is "yes", list the set of professors (p_0) common to both.	$\langle s\|P\| = \langle p\|$ $\langle s'\|P\| = \langle p'\|$ $\langle p\| \otimes \langle p'\| = \langle p_0\|$

answering any query about a list of objects having a number of attributes, as are required for information processing. In Sections 4(b) and 4(c) circuits specially relevant to that in figure 4 are presented.

(c) General circuits for Boolean algebra in BVMF and for logical checks

(i) *Complementation*: For obtaining the complement $\langle a^c$ of an n-vector $\langle a| = (a_1, a_2, \ldots, a_n)$, an elegant procedure is to use the circuit in figure 2(b) with the switch σ_a being set to give either 1 or 0. As indicated in figure 2(b), all the components a_j are complemented to give $a'_j = a^c_j$ if σ_a is set to 0, and they are left unchanged if σ_a is set to 1 so that $a'_j = a_j$ for all j.

It is obvious that this complementation switch and the associated circuitry shown in figure 2(b) can be generalized to the complementation of any n-vector or any $m \times n$ Boolean matrix.

(ii) *Union and Intersection*: The circuitry for these are given in figures 2(c), (d) based on the operations given by Eqns (10a) and (10b) which are to be applied between Boolean numbers a_j and b_j in order to obtain c_j, the components of the n-vector c which represents set C which is the union, or the intersection, of A and B respectively.

(iii) *Binary relation using the operator* E: This is illustrated in figure 2(a) where the circuitry in the bottom half implements the equation g E b = c. Its mode of action from a general point of view may be described by the following equations (10d) and (10e):

$$g_j \text{ AND } b_j = c_j, j = 1 \text{ to } n \qquad (10d)$$

$$\sum_j c_j = c; c_1 \text{ OR } c_2 \text{ OR } c_3 \ldots \text{ OR } c_n = c \qquad (10e)$$

As mentioned earlier, the value $c = 1$ indicates that there is at least one member in common between the n-vectors g and b, and $G \cap B \neq \emptyset$. If $c = 0$, it indicates that there are none—*i.e.* $G \cap B = \emptyset$.

(iv) *Agreement operator* G: Just as we checked in subsection (iii) above for there being at least *one* element in common between the n-vectors g and b, we can also set up a formula for checking if *all* elements g_j of g are identical with the corresponding elements b_j of b. The circuit diagram is given in figure 2(e), and it is based on the logical equations (10f) and (10g)—namely:

$$g_j \text{ EQU } b_j = d_j, j = 1 \text{ to } n \qquad (10f)$$

$$d_1 \text{ AND } d_2 \text{ AND } d_3, \ldots, \text{AND } d_n = d \qquad (10g)$$

It is obvious that d will be 1 only if every element g_j of g is equal to the corresponding b_j of b,—with both g_j and b_j being equal to 1, or with both g_j and b_j being equal to 0. This operator EQU in multivalued logic (between g and b) checks the set-theoretical relation $G \equiv B$, *via* the Boolean algebraic (BA-n) equation: $g_j = b_j$ for all $j = 1$ to n.

It is an interesting fact that Eqns (10d, e) and Eqns (10f, g) represent the truth value of the predicate logic equations between g and b with the four different types of quantifiers that are used in conventional predicate logic. These are

$$(\exists j)(g_j \,\&\, b_j) \text{ for (10d,e) with } c = 1 \qquad (10h)$$

$$\neg\,(\exists\, j)(g_j \,\&\, b_j) \text{ for (10d,}e\text{) with } c = 0 \qquad (10i)$$

$$(\forall\, j)(g_j \equiv b_j) \text{ for (10f,}g\text{) with } c = 1 \qquad (10j)$$

$$\neg\,(\forall\, j)(g_j \equiv b_j) \text{ for (10f,}g\text{) with } c = 0 \qquad (10k)$$

These equations, however, lead us deeply into predicate logic and the full consequences of these will be described in a separate report. (See ref 2 for the reason why the symbol $\neg$ is used for negating a quantifier, and not $\overline{\ }$, in common usage.)

Our purpose in this section has been to give some typical applications of our BVMF formalism for constructing logic circuits using standard logic gates and set these up to deal with different types of logical connectives and checks. A generalization of these is given in Section 5, in Part II.

4. CIRCUITS FOR ANSWERING QUERIES IN INFORMATION PROCESSING.

(a) *Simple example of examination marks*

We shall consider a particular example before we consider the general case. Suppose that we are given the marks a_{ij} secured by each candidate in tests labelled $j = 1$ to n, taken by m candidates, listed serially by numbers $i = 1$ to m. (We omit i for convenience in a_{ij}.) It is required to find out if the candidate (i) has secured marks $a_1, a_2, \ldots, a_n$, equal to or greater than the passing levels $\ell_1, \ell_2, \ldots, \ell_n$, for all the tests 1 to n. The relevant logical equation, giving an output $b = 1$ if the candidate has passed in all the tests, and $b = 0$ otherwise, can be given by Eq. (11) below:

$$(a_1 \geqslant \ell_1)\ \text{AND}\ (a_2 \geqslant \ell_2)\ \text{AND}\ \ldots\ \text{AND}\ (a_n \geqslant \ell_n) = b \qquad (11)$$

The circuitry for this is shown in figure 3 for one candidate whose serial number is taken as $i = 1$, and it will be readily seen that it is a closely faithful representation of Eq. (11) in hardware. It is only necessary to provide a hardware circuit for the check $a \geqslant \ell$. We shall give below in Section 4(b) a circuit with logical gates which will perform the arithmetical operation of finding out whether $a > \ell, a = \ell, a < \ell$. Obviously, we can have a series of such circuits with index $i = 1$ to m to give b_i, the outcome of the total examination.

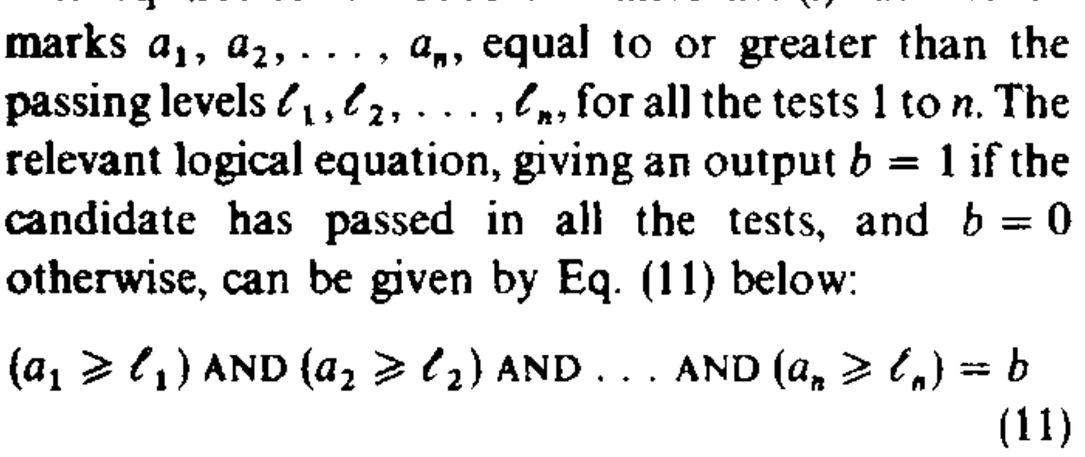

Figure 3. Simple circuit for checking the passing marks of a candidate in a number of tests.

It is obvious that there will be a great saving in time since all the checks for the n tests taken by the candidate are done in one go. So also, we can input the data for m candidates and have all of them processed to give b_1 to b_m in one cycle of the computer. The practical application of the circuit in VLSI is worth study.

(b) *General circuit for answering many queries.*

We shall discuss first the hardware circuitry for this which is given in figure 4. Taking the first horizontal row, and ignoring the symbol $m = 1$, the relevant Boolean algebraic equation can be written as follows:

$$(a_1\ \mathbf{R}_1\ \ell_1)\ \text{AND}\ (a_2\ \mathbf{R}_2\ \ell_2)\ \text{AND}\ \ldots\ \text{AND}\ (a_n\ \mathbf{R}_n\ \ell_n) = b \qquad (12)$$

This Eq. (12) differs from (11) only in the fact that the relation between the a_j's and ℓ_j's can be more general than the check given in Eq. (11). Hence in figure 4 it is referred to as REL, standing for 'relation'. Therefore it should deal with various types of relations as given in (13) below

$$a > \ell, a = \ell, a < \ell, a \geqslant \ell, a \leqslant \ell;$$

$$\ell_1 < \mathrm{a} < \ell_2, \mathrm{a} = \ell_1\ \text{OR}\ \ell_2\ \text{OR}\ \ell_3 \ldots \ell_k \qquad (13)$$

The circuit for all these is described in Section 4(c), in Part II. Here we accept the fact that the relation REL

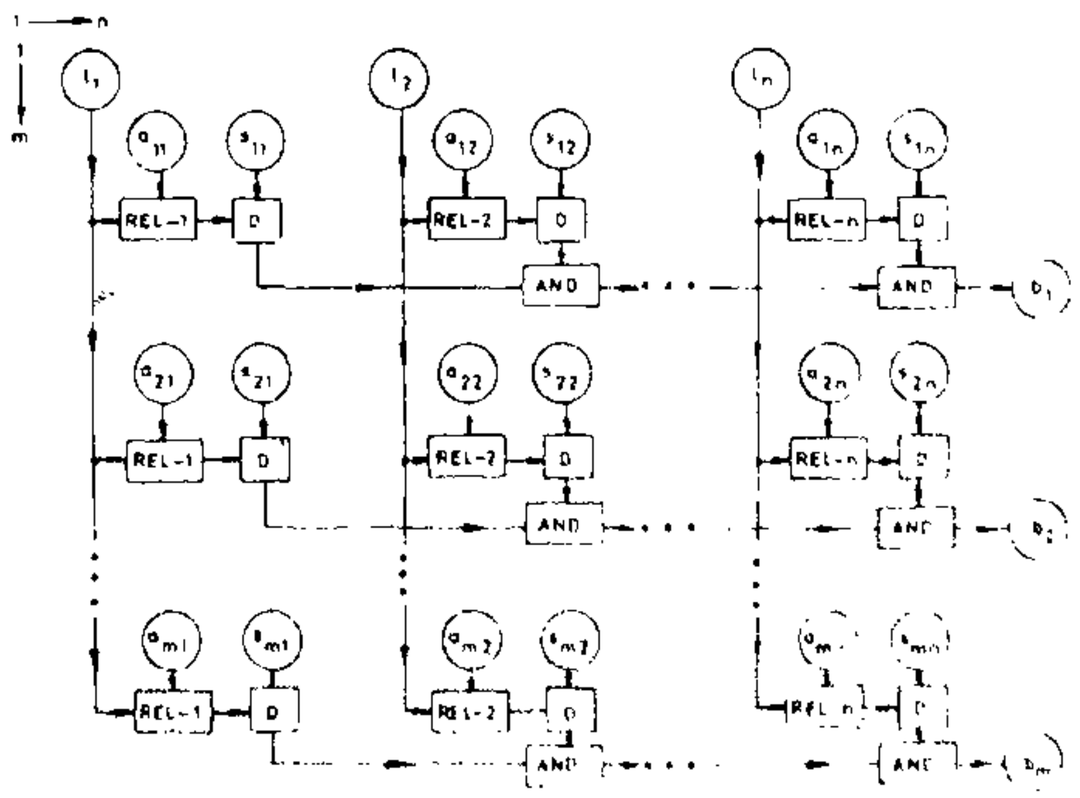

Figure 4. Hardware circuitry for the equation $\sum_{j=1}^{n} (a_{ij}\, R_j\, \ell_j) = b_i$, $i = 1$ to m.

having the logico-arithmetic forms as given in (13) can all be incorporated into one block.

As was evident for the various tests in an examination (figure 3), here also, we can have different checks REL-1, REL-2, . . . , REL-n, where each REL may be of a different type, and the common satisfying of all the n checking relations can be worked out simultaneously in hardware. If all the n tests are included for passing the total examination, then the logic gates named "delete", indicated by D in figure 4 is not needed. However, if only a partial checking is needed—say the checks *via* REL-1, REL-3, . . . , REL-n but not all of them—then those that are not needed can be deleted by the gate D which employs only very simple logical circuit elements. Thus, if the test j is not to be included, then we have only to set the switch s_j to produce a signal 1 for the appropriate D-gate to render the check of REL-j inoperative. If the signals s_j is set to be 0, then the Boolean output from REL-j is passed on, unchanged to the rest of the circuit by the deletion gate D_j. The circuitry for this is extremely simple and it is given in Section 4(c), in Part II.

With the above facilities provided by the logic block given in figure 4, it is believed that most of the needs in information processing related to queries can be met by this block. Further, all the processes are done in hardware, and therefore operatable in a small number of computer cycles, rather than in n steps as in software using sequential steps in the program.

25 November 1985.

1. Ramachandran, G. N., *Curr. Sci.*, 1982, **51**, 625.
2. Ramachandran, G. N., *Curr. Sci.*, 1983, **52**, Part I, p. 292, Part II, p. 335.
3. Ramachandran, G. N., "*Boolean Vector-Matrix Formalism for the Theory of Relations employed in Artificial Intelligence Studies*", Matphil Reports No. 42, June 1985.

ANNOUNCEMENTS

KARNATAK ASSOCIATION FOR THE CULTIVATION OF SCIENCE
[SCIENCE AWARDS for 1985]

The following are the receipients of the Science awards by the Karnatak Association for the Advancement of Science 1985. Dr B. V. Shela of ISRO Satellite Centre (Mathematics); Dr R. K. Somashekhar, Department of Botany, Bangalore University, Dr Nazeer Ahmed, Department of Zoology, Karnatak University, Dharwad (Biology).

NORMAN BORLAUG AWARD–1985

Dr K. L. Chadha, Director of the Indian Institute of Horticultural Research, Bangalore, has been awarded the 1985 Borlaug Award for his significant contributions to agriculture.

The Borlaug Award, instituted by the Coromandal Fertilisers in honour of the Nobel prize winning agricultural scientist Dr Norman Borlaug, carries a gold medal, cash prize of Rs.20,000 and citation.

Dr Chadha, an authority on mango breeding and cultivation, has made significant contributions in developing agro-technique for many horticultural crops.

Dr Chadha is the project coordinator in the Centre of Excellence in Tropical Horticulture of FAO-UNDP at Bangalore. Increased productivity and reduced cost of cultivation in fruit crops are his major research achievements. Dr Chadha is also taking keen interest in *Current Science*.