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

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

4(c) *Circuitry for deletion and branching*

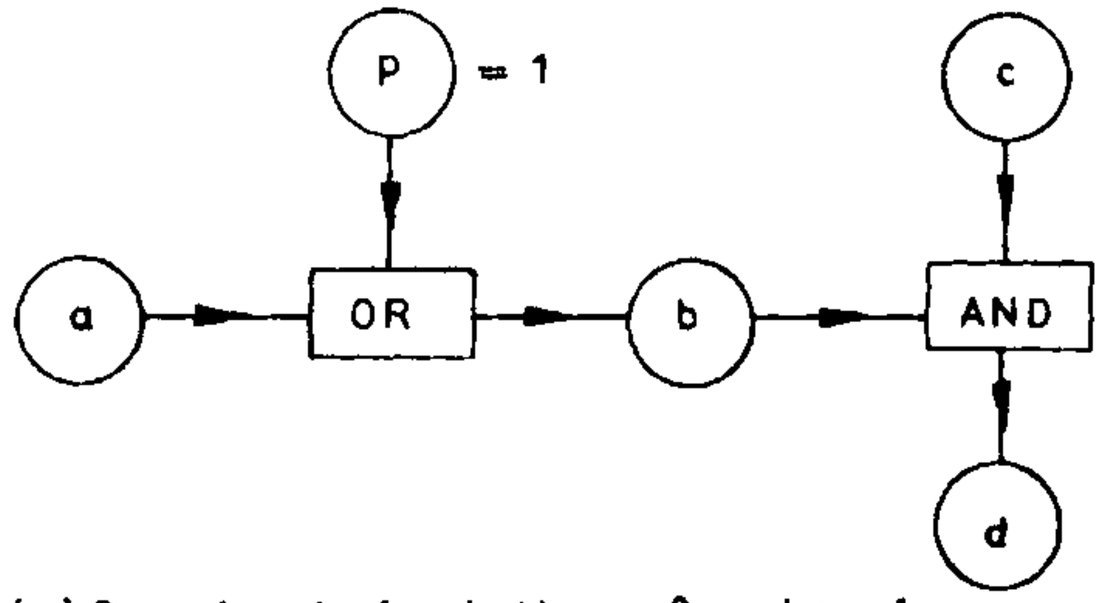
(*i*) *The delete gate D:* We have seen the application of the delete gate D in figure 4. We shall now give the nature of this gate which can be obtained by using a single logical chip OR. The circuit is shown in figure 5(a), where it is labelled D_1, for obvious reasons which will become clear below. If the pilot signal p for the OR gate is set as 1 and the other input is a, with b as the output of the gate, then it is readily verified that $b = 1$ *irrespective of the value of a being equal to* 0 *or* 1. If this value of b is an input into an AND gate, to which the other input is c and the output is d, then c will go unchanged to d. Effectively the information content of a will not pass on to the chain of operations containing c and d. Hence the name "delete" gate.

However, if the pilot signal p is set 0, then *a goes unchanged to b.* Consequently b will enter into the processing of the chain from c to d by AND. Therefore by setting p as 0 or 1, we can make the signal coming from a enter into the chain from c to d or not. This small circuit is all that is needed for the operator D in figure 4.

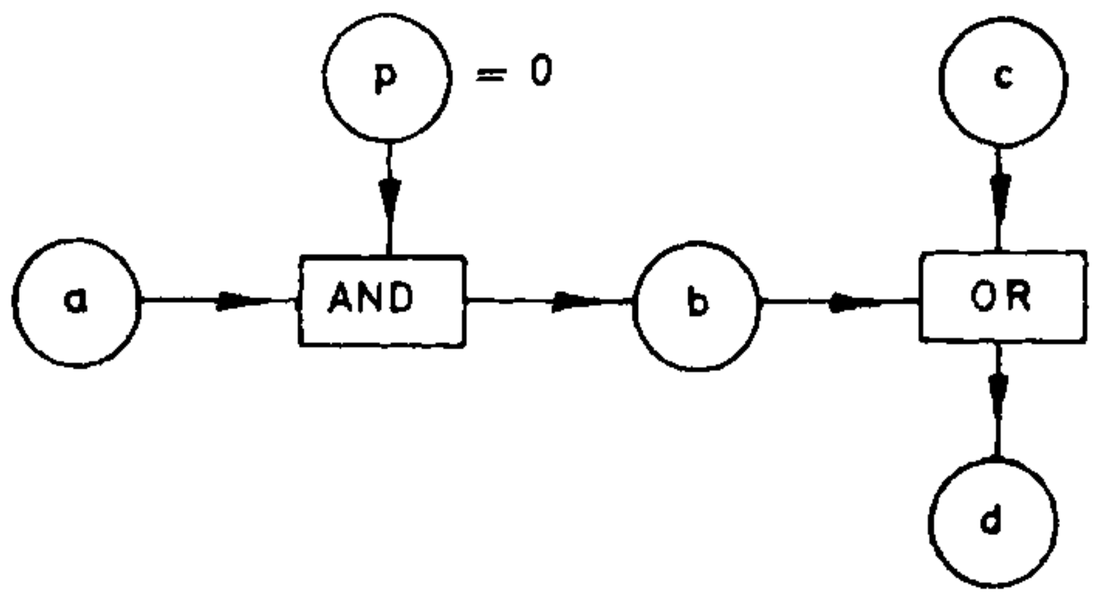
Figure 5(b) gives a circuit diagram for another "delete" gate, which we shall label as D_0. In this case, the values of c and d are connected by an OR gate with a being the second input to OR. The logic chip AND and the pilot signal p modifies this so that when p is 0, it makes b to be always 0 irrespective of a being 1 or 0. Therefore this signal on passing into the OR gate on the right makes no change in the transfer of signal from c to d. If, however, $p = 0$, then a goes unchanged into b and enters the chain c to d *via* OR.

These two delete gates D_1 and D_0 would find large application in a variety of problems.

(*ii*) *Branching gates BIN and BOUT:* The major circuit that we shall discuss below is the comparison of two binary numbers a and ℓ to find out whether $a > \ell$, $a = \ell$ and $a < \ell$. In this, it would be greatly beneficial if a circuit element, which we have named BIN (Branch-in), is used as one of the elementary components, just as AND and OR are used. A similar circuit element BOUT (Branch-out) also will greatly help in building complicated logic circuits. The properties of BIN and BOUT are given in figures 6(a) and (b). As in the case of the delete gate, both BIN and BOUT employ a pilot input p which can be set as 0 or 1. In BIN, if $p = 0$, a_1 is output as b, and when $p = 1$, a_2 is output as b. The Boolean numbers 0 and 1 for p are marked against the inputs



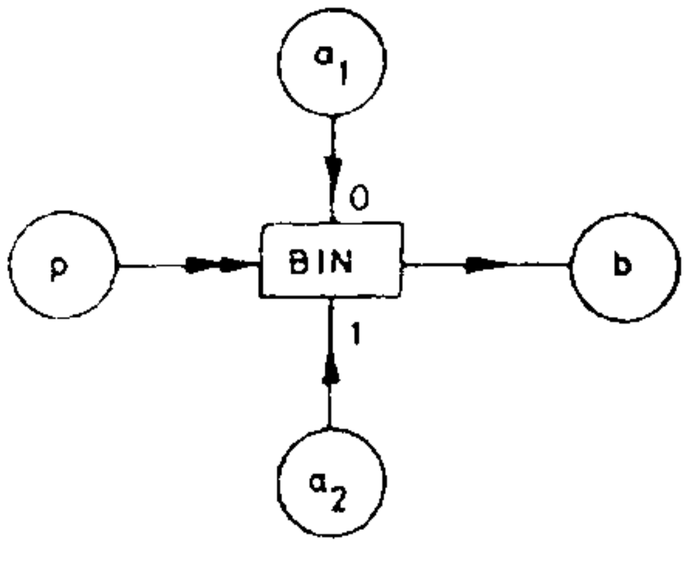
(a) D_1 : b = 1, for both a = 0 and a = 1



(b) D_0 : b=0, for both a=0 and a=1

Figure 5. "Delete" gates D_1 and D_0. The former gives an output $b = 1$ irrespective of a being 1 or 0 and passes a unchanged when $p = 0$. Similarly D_0 gives $b = 0$ irrespective of a being 1 or 0 and passes a unmodified when $p = 1$.

*This part II is incontinuation of Part I, which appeared in the last issue of this journal *Curr. Sci.*, Vol. 55, No. 1. January 5, 1986, pp. 12–18. The abstract is in Part I.



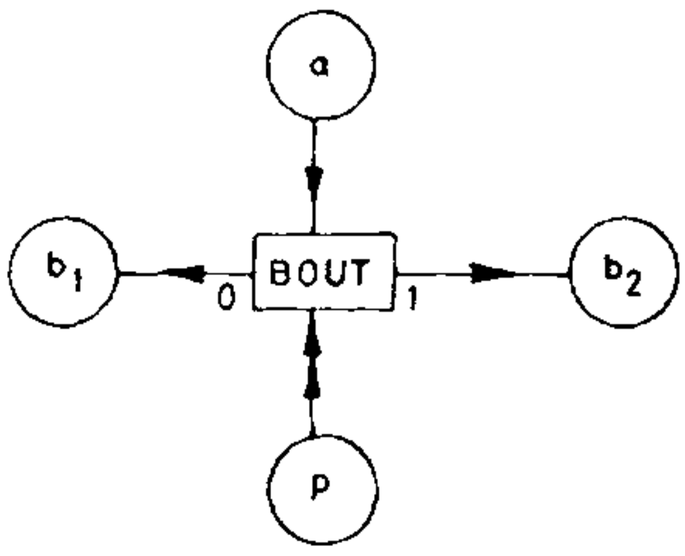


Figure 6. BIN and BOUT gates activated by the pilot signal $p = 0$ or 1. BIN has two inputs a_1, a_2 and one output b while BOUT has one input a and two ouputs b_1, b_2. See text for details.

from a_1 and a_2 respectively in figure 5(a). In exactly the same way, the component BOUT also has a pilot input p which is Boolean, having values of only 0 and 1. In this case, if $p = 0$, a is output as b_1; while if $p = 1$, a is output as b_2. We shall not give a description of these with AND and OR gates, but shall treat BIN and BOUT themselves as elementary circuit components. The utility of this is particularly evident from the application given below for arithmetic checking of equality, greater than, and less than, relations.

The equations involving BIN and BOUT are written as follows:

$$p \text{ BIN } (a_1, a_2) = b \text{ and } p \text{ BOUT } a = (b_1, b_2)$$

In each case, a_1, a_2 or b_1, b_2 correspond, respectively, to the values of $p = 0, 1$.

(d) Circuitry for checking the relations equal to, greater than, and less than.

We assume that the two numbers a and ℓ are given in the binary system, and that they have atmost r binary digits. We shall illustrate the principle of the circuitry by taking r to be 8, with a and ℓ being integers. The digits are numbered in reverse order by the index $j = 1$ to r ($= 8$) and the values of a_j and ℓ_j are denoted by $a_{\alpha,j}$ and $a_{\beta,j}$. These are indicated in the top two rows of table 2. The principle of the logic involved is given by Eqns (14a to e) which relate the values of $(b_{\alpha,j-1}\, b_{\beta,j-1})$ to the values of $(b_{\alpha,j}\, b_{\beta,j})$ for the digit j. In the intermediate steps, we have the quantities $(d_{\alpha,j}\, d_{\beta,j})$ and p_j as given by Eq. (14a, b, c)

$$b_{\alpha,j-1} \text{ OR } a_{\alpha,j} (= a_j) = d_{\alpha,j} \tag{14a}$$

$$b_{\beta,j-1} \text{ OR } a_{\beta,j} (= \ell_j) = d_{\beta,j} \tag{14b}$$

$$d_{\alpha,j} \text{ AND } d_{\beta,j} = p_j \tag{14c}$$

$$p_j \text{ BIN } (b_{\alpha,j-1}, d_{\alpha,j}) = b_{\alpha,j} \tag{14d}$$

$$p_j \text{ BIN } (b_{\beta,j-1}, d_{\beta,j}) = b_{\beta,j} \tag{14e}$$

We start with $(b_{\alpha,0}\, b_{\beta,0}) = (0\,0)$ corresponding to $j = 0$ and apply the set of Eqns (14) for $j = 1$ to 8, when the data in the last two rows of table 2 are obtained by stepwise calculation of the b's from $j-1$ to j.

The logical equations in (14) have the property of giving the output $(b_{\alpha,r}\, b_{\beta,r})$ as (1 0), (0 0) or (0 1) according as $a > \ell$, $a = \ell$ or $a < \ell$. The principle involved is the fact that if two integers in the binary system, namely a and ℓ, with digits $(a_j\, b_j) \equiv (a_{\alpha,j}\, a_{\beta,j})$ have the relation $a > \ell$, then, on comparing the Boolean values of $a_{\alpha,j}$ with $a_{\beta,j}$, the two will be found to be equal for all j from 1 to some $j_1 - 1$ and we will find them different for $j = j_1$. Suppose $a_{\alpha,j_1} = 1$ and $a_{\beta,j_1} = 0$ for the first time for some $j_1 < r$. The requirement we have arranged is that $(b_{\alpha,j}\, b_{\beta,j})$ will remain (0 0) until this value of $j = j_1$ is attained, when $(b_{\alpha,j}\, b_{\beta,j})$ changes over into (1 0), and remains unchanged thereafter, for higher values of $j > j_1$, until j reaches the value r, when it is output as $b = (b_\alpha\, b_\beta) = (1\,0)$ (indicating $a > \ell$).

Table 2. *Digit-wise progression of the checking of two binary integers for equality, greater than and less than**

j	0	1	2	3	4	5	6	7	8
$a_j (= a_{\alpha,j})$	–	0	1	1	0	0	1	0	1
$\ell_j (= a_{\beta,j})$	–	0	1	0	0	1	1	0	0
$d_{\alpha,j}$	–	0	1	1	1	1	1	1	1
$d_{\beta,j}$	–	0	1	0	0	1	1	0	0
p_j	–	0	1	0	0	1	1	0	0
$b_{\alpha,j}$	0	0	0	1	1	1	1	1	1
$b_{\beta,j}$	0	0	0	0	0	0	0	0	0

*See text for the explanation of the contents.

In exactly the same way, if $a < \ell$, the input $\mathbf{b}_0 = (0\ 0)$ goes unchanged as $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{j_1-1}$ until a difference between $a_{\alpha, j1} = 0$ and $a_{\beta j_1} = 1$ is noticed for the first time for $j = j_1$. Then $\mathbf{b}_j$ becomes (0 1), and remains so thereafter until it is output for $j = r$ as $\mathbf{b} = (b_\alpha\ b_\beta)$, and this happens irrespective of the values of $\mathbf{a}_j$ found for $j = (j_1 + 1)$ to r.

If, however, $a = \ell$, then necessarily $a_{\alpha,j} = a_{\beta,j}$ for all j, and $\mathbf{b}_0 = (0\ 0)$ comes out unchanged as the output $\mathbf{b} = \mathbf{b}_r = (0\ 0)$. Thus the three possibilities $a > \ell$, $a = \ell$, $a < \ell$ are simultaneously checked by the circuit, and the information as to which is true becomes available in the form of the output 2-vector $(b_\alpha\ b_\beta)$, which will have the value (1 0), (0 1) or (0 0), corresponding respectively to $a > \ell$, $a < \ell$ and $a = 0$.

The circuitry in figure 7 takes care of the logical operations (14a to e) for one digit (j), with $(b_{\alpha,j-1}\ b_{\beta,j-1})$ as input and $(b_{\alpha,j}\ b_{\beta,j})$ as output. We shall briefly explain the way in which Eqns (14a–e) corresponding to figure 7 are utilized in the complete process of checking against ℓ, by taking the example in table 2.

Starting from $(b_{\alpha,0}\ b_{\beta,0}) = (0\ 0)$ in the first column with $j = 0$, the data for $j = 1$ do not introduce any new features and the values (0 0) go unchanged into $(b_{\alpha,1}\ b_{\beta,1})$ *via* $(d_{\alpha,1}\ d_{\beta,1})$. The digits (1 1) of $(a_{\alpha,j}\ a_{\beta,j})$ for $j = 2$ makes $p_j = 1$, so that the vector $\mathbf{b}_1$ again goes unchanged *via* the BIN gate into $\mathbf{b}_2$. When, however, a difference between a_j and ℓ_j is noticed *for the first time* for $j = j_1 = 3$, Eqns (14a and b) give (1 0) for $\mathbf{d}_j$. Consequently p_{j_1} is 0 and $\mathbf{d}_{j_1}$ goes in as $\mathbf{b}_{j_1} = (1\ 0)$, indicating that we have noticed the first digit in decreasing order (j_1) at which $a_j > \ell_j$. (Obviously if for the first time (for $j = j_1$) it is found that $a_{j_1} < \ell_{j_1}$, then $\mathbf{b}_{j_1}$ will become (0 1). Also, if $a_{j_1} = \ell_{j_1}$, $\mathbf{b}_{j_1}$ becomes (0 0).) Thus, the requisite property mentioned above for the simultaneous checking of the three relations $a > \ell$, $a = \ell$ and $a < \ell$ according to the value of the vector $\mathbf{a}_j$ becomes established for j_1, the digit for which the difference is noticed for the first time.

Moreover, the circuitry in figure 7 has also the property of maintaining the value of $\mathbf{b}_j$ for all values of $j > j_1$ by virtue of the nature of the Eqns (14a to e). Thus, if the two digits a_j and ℓ_j are equal to 0, as for $j = 4$, the circuit passes $\mathbf{b}_j = (1\ 0)$ unchanged from $j = 3$ to $j = 4$. Similarly, for $j = 5$, even though the inequality between a_j and ℓ_j is reversed to make $\mathbf{a} = (0\ 1)$, still since $d_\alpha = 1$ and $d_\beta = 1$ makes the pilot signal $p_j = 1$, $\mathbf{b}_j$ is passed on as $\mathbf{b}_{j-1}$, and the value (1 0) is passed on for $j = 5$ also. For the next digit with $j = 6$, we have $a_j = 1$ and $\ell_j = 1$ yielding $p = 1$ *via* $d_\alpha = 1$ and $d_\beta = 1$, so that once again $\mathbf{b}_5$ passes on unchanged as $\mathbf{b}_6 = (1\ 0)$. What happens for $j = 7$ is a repetition of that for $j = 4$. Also, for $j = 8$, we encounter once again $a_j = 1$, $\ell_j = 0$ and it only transmits the vector $\mathbf{b}_7 = (1\ 0)$ into the output signal $\mathbf{b}_8 = (1\ 0)$.

Thus we have obtained a simple design, using only five logic chips for each digit, which can compare two binary numbers a and ℓ and give the output signal in the form of one of the three 2-vectors (1 0), (0 0), (0 1), according as $a > \ell$, $a = \ell$, $a < \ell$ respectively. The utilization of these so as to obtain any one of the matching conditions mentioned in Eq. (13) is simple and this is indicated in figure 8, and briefly explained in the next subsection (e).

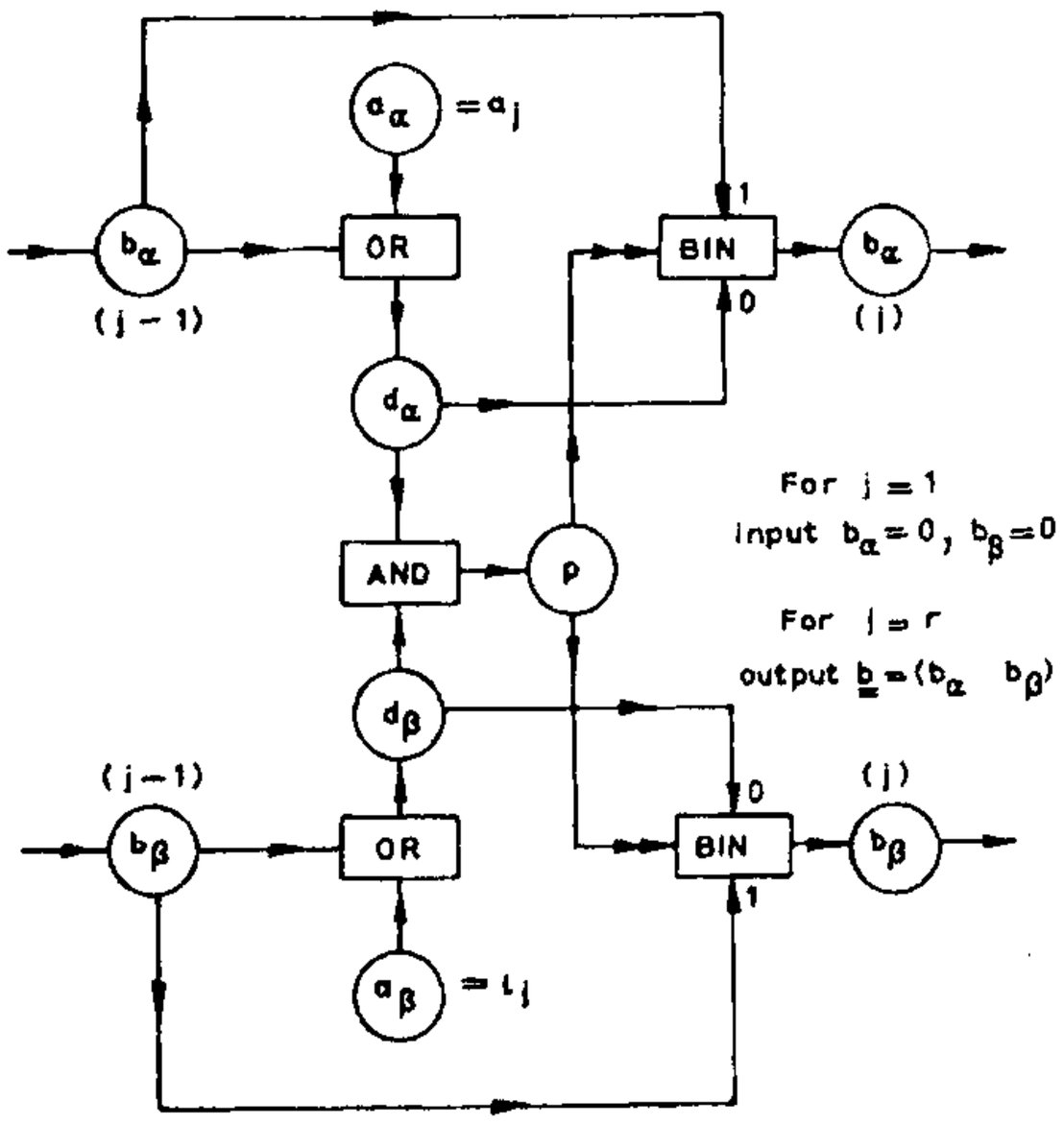


Figure 7. Circuit diagram for the analysis of the relation between two binary numbers as being greater than, equal to, or less than. The set of hardware components needed for one digit j is shown.

(e) Circuits for utilizing comparison test in (d).

We have seen in the previous section how a single check of all the binary digits of two integers a and ℓ can give three different outputs for $(b_\alpha\ b_\beta)$ according as the arithmetical relation $>$, $=$, $<$, is satisfied for a REL ℓ. In figure 8, we show how we can get an one-element Boolean number (1 or 0) for the truth and falsity of any one of the five relations $>$, $=$, $<$, $\geqslant$ or $\leqslant$, using the output of figure 7, namely the vector $\mathbf{b} = (b_\alpha\ b_\beta)$. The essence of the circuit is that each of the three possibilities for the 2-element vector, namely (1 0), (0 0), (0 1) is

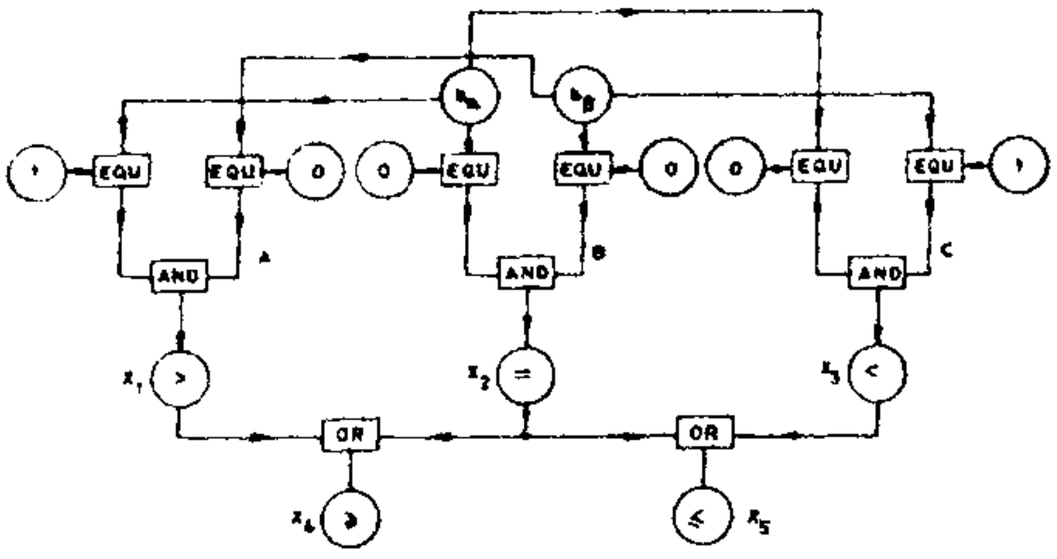


Figure 8(a). Circuit for utilizing the BA-2 vector output ($b_{\alpha r}$ $b_{\beta r}$) of figure 7 so as to obtain Boolean (BA-1) outputs for the relations REL = $>$, $=$, $<$, $\geqslant$, $\leqslant$.

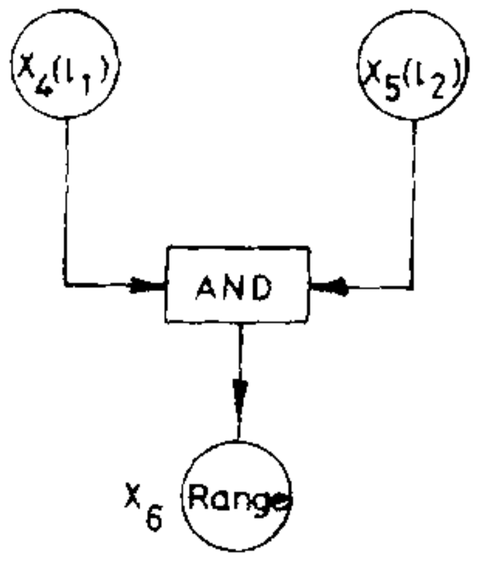


Figure 8(b). Boolean output x_6 for the 'Range' relation REL = $\ell_1 \leqslant a \leqslant \ell_2$.

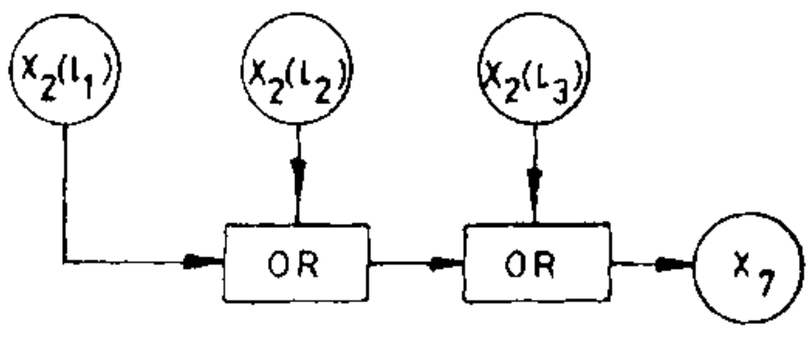


Figure 8(c). Boolean output x_7 for the relation $a = \ell_1$ or ℓ_2 or ℓ_3

compared *via* the two EQU operators to the corresponding elements (b_α b_β). Consequently, when a match is obtained for one of the testing circuits marked A, B, C then, the corresponding output is 1 and the outputs of the other two will be 0. Thus, if (b_α b_β) = (1 0), it will give a Boolean output $x_1 = 1$ in A corresponding to $a > \ell$ and both x_2 and x_3 will be 0; similarly for the other two checks in B for equality and C for less than. Thus, effectively, we have obtained the required outputs 0 or 1 of the check for a REL ℓ for each of the relations REL = $>$, $=$, $<$. Similarly by combining x_1 and x_2 by the connective OR to obtain x_4 we get a Boolean check for $\geqslant$ for REL, and by combining x_2 and x_3 by OR we obtain x_5, giving an one-element Boolean output for $\leqslant$.

Often there is a need to have a relation REL of the type $\ell_1 \leqslant a \leqslant \ell_2$. The circuit for this is shown in figure 8(b) in which we use the outputs of two comparison circuits based on figure 7 for a REL ℓ_1 and a REL ℓ_2. If we combine these two by the connective AND we obtain the Boolean output x_6 marked 'Range' which will test the truth or falsity of $\ell_1 \leqslant a \leqslant l_2$.

So also, it is possible to check for $a = \ell_1$ or $a = \ell_2$ or $a = \ell_3$ by combining the outputs of the circuits checking these, which we may denote by x_2 (ℓ_1), x_2 (ℓ_2), x_2 (ℓ_3) and connecting them logically by multiple OR gates to represent the logical equation (15).

$$(a = \ell_1) \text{ OR } (a = \ell_2) \text{ OR } (a = \ell_3) = b \qquad (15)$$

as shown in figure 8(c). It is obvious that b will be 1 if *any one* of the relations between a and ℓ mentioned in Eq. (15) is satisfied.

The circuits mentioned above are expected to be of good use even in orthodox computer circuitry. We shall not comment on this or other possible applications mentioned in this section, but consider in the next Section 5 some circuits of a general nature based on those we have discussed so far, which can find application in computer science in general.

5. *Computer circuitry to implement some general types of relations in multivalued logic (MVL).*

(i) *General relations between n-vectors.*

The logical relations in (10d, e, f, g) in the theory of relations can be generalized to take the forms in (16a, b) given below.

$$\bigvee_{j=1 \text{ to } n} (a_j \text{ REL } b_j) = \bigvee_j c_j = c \qquad (16a)$$

$$\bigwedge_{j=1 \text{ to } n} (a_j \text{ REL } b_j) = \bigwedge_j c_j = c \qquad (16b)$$

where a_j, b_j for $j = 1$ to n, as well as c, are Boolean numbers. Denoting the summation operator in (16a and b) generally by the symbol "CON" (standing for connective), the above two equations can be written for direct implementation in logic circuitry by the Eq. (17) below.

$$(a_1 \text{ REL } b_1) \text{ CON } (a_2 \text{ REL } b_2) \text{CON} \ldots \text{ CON } (a_n \text{ REL } b_n) = c \qquad (17)$$

Figure 9 gives the circuitry that implements Eq. (17), in which both REL and CON are BA-1 operators (logical

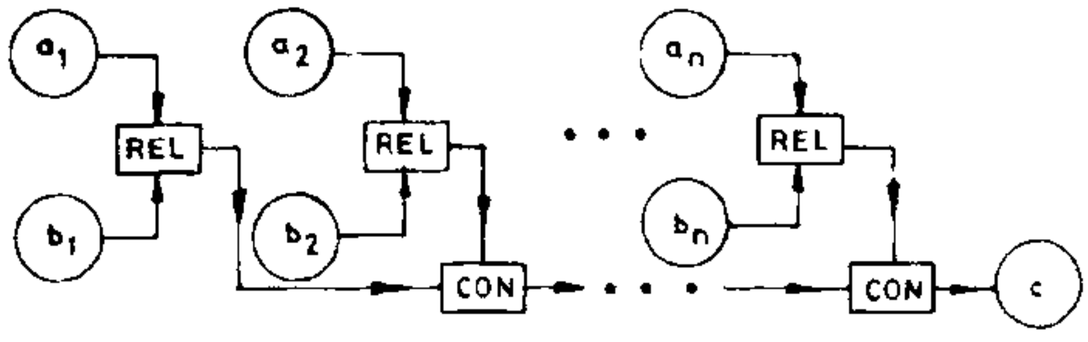


Figure 9. General multi-valued logic equation in BA-*n* represented by circuitry using only logic gates in BA-1

gates) such as AND, OR, EQU, IMP etc. In Eq. (17) REL-j for all $j = 1$ to n are taken to be the same and so are all the connectives CON. In this form, the generalization Eq. (17) has some very interesting consequences which are shown in table 3. We restrict ourselves to the connectives AND and OR for CON, while we use three possible BA-1 operators AND, OR and EQU to replace REL in Eq. (17) and figure 9. Then, according to the value of $c = 1$ or 0, and the nature of the connective CON and the relation REL, several QPL binary relations in predicate logic can be implemented. A partial list of the most prominent among these is given in table 3. It will be seen that all the four quantifiers $\forall$, $\neg\forall$, $\exists$, $\neg\exists$ are available by different combinations of AND and OR and of $c = 1$ and 0. An explanation of why these quantifiers occur becomes obvious by an inspection of figure 9 and the substitutions of the logical connectives for REL and CON.

Table 3 *Special cases of REL and CON leading to binary relations in quantified predicate logic*

Sl No	BA-1 connective in a_j REL b_j	Summation via CON	Truth value of binary relation	QPL binary relation
1	AND	AND	1	$(\forall j)\,(a_j \,\&\, b_j)$
2	AND	AND	0	$\neg\,(\forall j)\,(a_j \,\&\, b_j)$
3	AND	OR	1	$(\exists j)\,(a_j \,\&\, b_j)$
4	AND	OR	0	$\neg\,(\exists j)\,(a_j \,\&\, b_j)$
5	EQU	AND	1	$(\forall j)\,(a_j \equiv b_j)$
6	EQU	AND	0	$\neg\,(\forall j)\,(a_j \equiv b_j)$
7	EQU	OR	1	$(\exists j)\,(a_j \equiv b_j)$
8	EQU	OR	0	$\neg\,(\exists j)\,(a_j \equiv b_j)$
9	OR	AND	1	$(\forall j)\,(a_j \vee b_j)$
10	OR	AND	0	$\neg\,(\forall j)\,(a_j \vee b_j)$
11	OR	OR	1	$(\exists j)\,(a_j \vee b_j)$
12	OR	OR	0	$\neg\,(\exists j)\,(a_j \vee b_j)$

(ii) *QPL binary relations covering multivalued relations of the type in Eq.* (16) *and* (17).

The third row of the data in table 3 has already been considered and illustrated in figure 2(a), with g_j replacing a_j of table 3. As will be seen from that, the output c is equal to 1 if any one of the relations g_j AND $b_j = c_j$ gives $c_j = 1$. Thus this circuit has the property that if any corresponding pair of elements g_j and b_j of the n-vectors g and b are both unity, then c_j is equal to 1. Hence, the QPL description in the third row of table 3, modified as $(\exists j)(g_j \,\&\, b_j)$, is seen to be satisfied if $c = 1$. On the other hand, if all the individual relations are not satisfied, we have $c_j = 0$ for all $j = 1$ to n, and the summation $\sum_j c_j = c$ will be 0. Hence there are no elements in common between the n-vectors a and b. Thus, the fourth row of table 3 corresponds to $\neg\,\exists\,(a_j \,\&\, b_j)$.

The 5th and 6th rows of table 3 have already been represented in figure 2(e), also with g replacing a. In this case, the relation REL is EQU, which is satisfied if both g_j and b_j are equal to 1 or if both are equal to 0. In this case, we get a complete match between the n-vectors g and b, which can be represented by $\mathbf{g} \equiv \mathbf{b}$. The corresponding QPL-1 relation $(\forall j)\,(g_j \equiv b_j)$ obviously represents this situation. On the other hand, when $c = 0$ as in row 6, we can only say that $a_j = b_j$ is not true for all j, yielding the quantifier $\neg\,\forall$.

In the above description, we have shown the genesis from MVL of the four types of quantifiers used in predicate logic, namely $\forall$, $\neg\,\forall$, $\exists$, $\neg\,\exists$. The relation within the bracket which is operated upon by the quantifier is seen to be the same as the BA-1 connective given in the first column of each row. We have only taken three BA-1 connectives namely AND (&), OR ($\vee$), EQU ($\equiv$) in table 3; but the technique is obviously generalizable for any logic chip of the types given in (18) below and implemented by summation *via* AND or OR to give all possible QPL binary relations which could be employed in standard QPL-1 equations.

AND, NAND, OR, NOR, EQU, XOR and others obtained by complementing a and/or b (18)

(iii) *Single logic circuit with facilities to implement any BA-1 connective of the type OR or AND*

Figure 10 given below is the circuit diagram of such a chip with three sign switches σ_a, σ_b and σ_c to take care of the eight possible relational connectives which can be obtained from OR and AND by negating either a, or b, or c, in the general equation a REL b = c. The

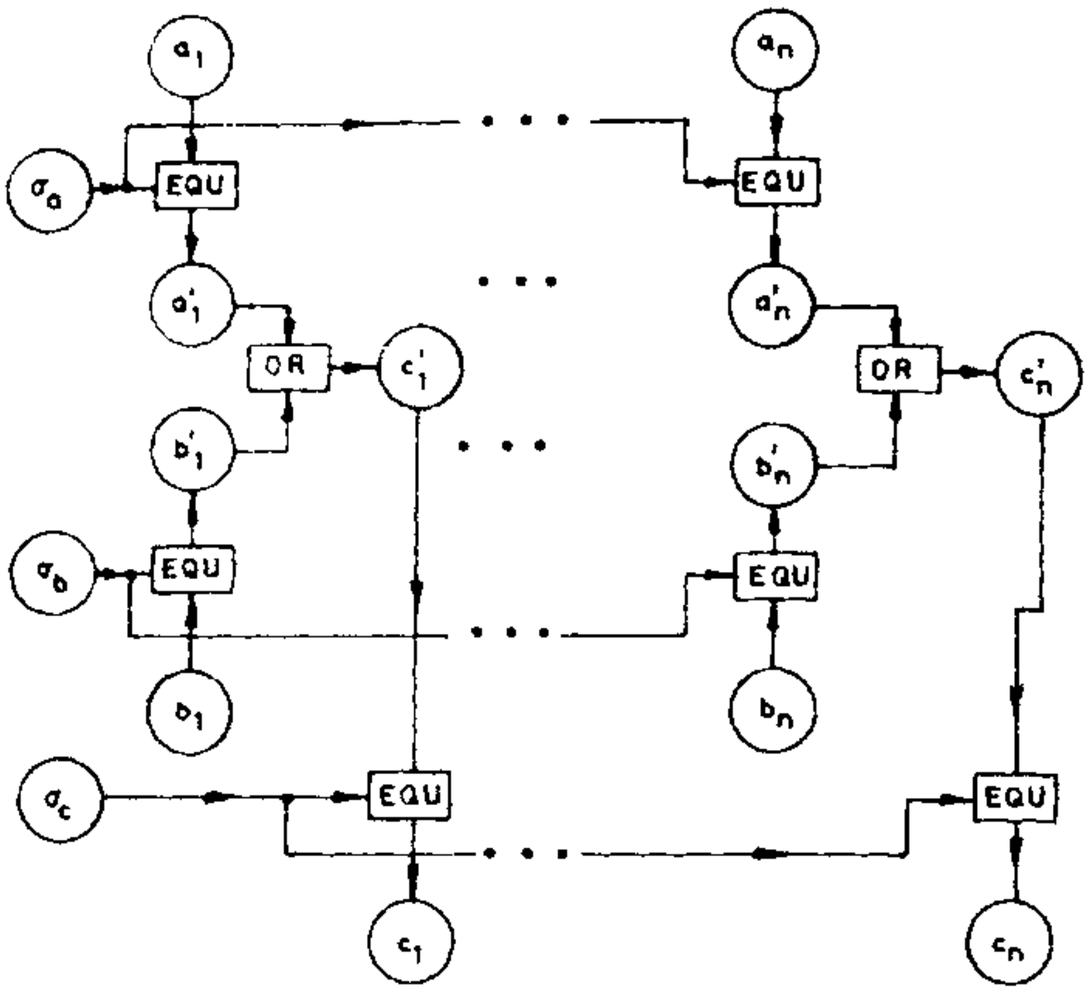


Figure 10. A general circuitry for a REL b = c, consisting of a_j REL $b_j = c_j$ for all j, in which REL can be set to be any one of the eight BA-1 relations **A**, $\mathbf{A}^c$, **O**, $\mathbf{O}^c$, **I**, $\mathbf{I}^c$, **J**, $\mathbf{J}^c$ by the setting of the three "sign" switches σ_a, σ_b, σ_c. (See table 4).

principle of the circuit is very simple. It is based on n logic gates which connect a'_j, b'_j with c'_j by the connective OR. Then, there are circuits, employing EQU as shown in figure 2(b), which can negate a_j, b_j or c_j. Then, since each one of the three n-vectors a, b, c can be affirmed or complemented there are eight possibilities for the effective BA-1 connective that is obtainable from OR by this process. We shall not explain them, but list them in table 4 corresponding to the different settings of σ_a, σ_b and σ_c. The standard name of the effective relational connective REL, as well as its symbol in our BVM formalism, are given in each row. Some of the symbols require explanation. Thus IMP stands for "implication" in the form $a \Rightarrow b$ and has the symbol I in our notation. The reverse relation for this, namely in the form b ⇒ a, is given the symbol RIMP, to stand for "reverse implication", and is denoted by J in our notation. Just as OR and AND can have their complemented forms NOR and NAND, we can also complement IMP and RIMP to obtain $\mathbf{I}^c$ and $\mathbf{J}^c$ respectively. The symbols for these ($\not\Rightarrow$ and $\not\Leftarrow$) are obvious, but since they are very rarely used, we have the 6 and 7 letter symbols NOT IMP and NOT RIMP for these.

The complete circuit shown in figure 10 can thus take care of any of the eight classical connectives of BA-1, used in propositional logic, extended to multi-valued logic.

We have not covered the connective EQU (**E**) and its negation XOR ($\mathbf{E}^c$) by the circuitry in figure 10. It is suggested that the same circuit as in figure 10 be employed for these, replacing the OR gates in that figure by EQU gates. Then the various possible settings of σ_a, σ_b and σ_c convert these two operators into one another, four of them being identical with EQU and four others with XOR.

Thus the ten standard connectives for binary relations in propositional calculus can be implemented *via* logic gates, and a row of n of these will take care of binary relations in n-valued logic and its associated Boolean algebra BA-n.

Table 4 *Nature of logical connective* REL *in a* REL b = c *for different settings of* σ_a, σ_b, σ_c *in Fig.* 10

Sl No	σ_a	σ_b	σ_c	Relational connective* REL	BVMF operator
1	1	1	1	OR ($\vee$)	**O**
2	1	1	0	NOR ($\overline{\vee}$)	$\mathbf{O}^c$
3	0	1	1	IMP ($\Rightarrow$)	**I = N O**
4	0	1	0	NOT IMP ($\not\Rightarrow$)	$\mathbf{I}^c$ **= A N**
5	1	0	1	RIMP ($\Leftarrow$)	**J = O N**
6	1	0	0	NOT RIMP ($\not\Leftarrow$)	$\mathbf{J}^c$ **= N A**
7	0	0	1	NAND ($\mid$)	$\mathbf{A}^c$
8	0	0	0	AND (&)	**A**

*IMP = imply (as in $a \Rightarrow b$), NOT IMP = does not imply, RIMP = imply in the reverse sense i.e. b ⇒ a, NOT RIMP = (b $\not\Rightarrow$ a)

We have not covered the connective EQU (**E**) and its negation XOR ($\mathbf{E}^c$) by the circuitry in figure 10. It is suggested that the same circuit as in figure 10 be employed for these, replacing the OR gates in that figure by EQU gates. Then the various possible settings of σ_a, σ_b and σ_c convert these two operators into one another, four of them being identical with EQU and four others with XOR.

Thus the ten standard connectives for binary relations in propositional calculus can be implemented *via* logic gates, and a row of n of these will take care of binary relations in n-valued logic and its associated Boolean algebra BA-n.

ACKNOWLEDGEMENTS

The author wishes to acknowledge his gratitude to Dr Veni Madhavan of the School of Automation of this Institute for indicating types of problems that are met with in information processing, which led to this investigation. The cooperation of Mr. T. A. Thanaraj in working out and constructing simple logic circuits for PC and QPL helped the author greatly in generalizing them and obtaining the results presented here. The author is particularly grateful to the Indian National Science Academy for providing him with an Emeritus Professorship (Albert Einstein Professor) in this Institute.