SYĀD NYĀYA SYSTEM (SNS)—A NEW FORMULATION OF SENTENTIAL LOGIC AND ITS ISOMORPHISM WITH BOOLEAN ALGEBRA OF GENUS 2*

G. N. RAMACHANDRAN
CSIR Distinguished Scientist, Gita, 10A Main Road, Malleswaram West, Bangalore 560 055

ABSTRACT

While the anaology of logical OR and AND of standard sentential logic with Boolean sum and product in BA-1, having only two elements 1 and 0, is well-known, the use of BA-*n*, generated by *n* Boolean elements, does not seem to have been recognized. It is shown here that BA-2 is an excellent algebra, for representing isomorphically all connectives of propositional calculus, *via* 2-element Boolean vectors ($T = (1\ 0)$, $F = (0\ 1)$) and 2×2 Boolean matrices, not only for 'forward' connectives, but also for 'reverse' connectives as defined here for the first time. It is computer implementable and has two new states 'doubtful' $D = (T \oplus F) = (1\ 1)$ and 'impossible' $X = (T \otimes F) = (0\ 0)$. It has been named Syad Nyaya (may be logic) System. Two new operators *U* (for unanimity) and *V* (for vidya = knowledge) also occur in SNS corresponding to Boolean sum and Boolean product in BA-2. *V* has the capability of extracting T or F in the presence of D in one of the terms, and showing "impossibility" when the two sources combined by *V* form the contradictory pair, T and F. BA-3 with three independent generators has 8 (7 possible + 1 impossible) states which is isomorphic to the extension of the states $\forall$, $\exists$, and (ϕ) of quantified logic. The third state of Brouwer's Intuitionistic Logic (B) is very reasonable since T, B, F can form the generators of a BA-3 algebra. It is pointed out that Ancient Indian (Jaina) Philosophy had recognized the four states of SNS, or BA-2, in their Syād-Vāda (doctrine of doubt) and had even listed the seven possible states of BA-3 in their Sapthabhangi (Theory of seven parts).

INTRODUCTION

In standard classical logic (CL, also known as propositional calculus, or sentential logic), a term or sentence can have only two truth values, namely T and F. Also it is well-known[1] that the logical interconnections between different terms can always be represented in terms of only three connectives 'OR' ($\vee$), 'AND' (&) and 'NOT' ($\neg$). The properties of the abovementioned logic can also be represented by taking the simplest of Boolean algebras (BA-1) having only two elements 1 and 0 and putting the elements in isomorphism with T and F respectively. Then the three logical connectives become representable by the Boolean operations of 'sum' ($\oplus$), 'product' ($\otimes$) and 'complement' ($'$). Also, the well-known result in mathematics[2] that all Boolean algebras are Boolean rings (which proves the completeness of BA-1) also establishes the completeness of CL, so long as the connective operators are applied only in the forward direction.

However, when the connectives 'AND' and 'OR' are applied in the 'reverse' direction (as discussed in the next section), even if only the states T and F are input, the output is not always either T or F alone. We found that a more powerful Boolean algebra, namely one of genus 2 (BA-2), is required to describe such operations. (The term 'genus *n*' refers to a Boolean algebra arising from *n* independent generators[2].) Two new logical states – 'doubtful' (D) and 'impossible' (X)—occur in the system (see

*MATPHIL Reports No. 24, April 1982.

Section 3, and also two new connectives for 'unanimity'($\oplus$) and 'vidya'($\otimes$) are found to be absolutely necessary (see Section 4).

We have named the logical system which employs T, F, D, X, as the basic states, as Syād Nyāya System (Syād = may be, and Nyāya = logic, in Sanskrit). Since the introduction of the doubtful state is the essentially new aspect in this system, and since this truth state D is describable as "may be either true or false," the name 'Syād' is justified for the system. This article gives a brief outline of the nature and properties of the SNS formulation, and shows how CL forms only a part of this. This follows from the result that the corresponding Boolean algebras BA-1 and BA-2 are such that the former is contained in the latter, forming only a subalgebra of it.

To make the presentation fairly complete, we consider very briefly, in Section 5, BA-3, which is shown to be isomorphic with the states of first order quantified predicate logic (QPL). Here again, when this analogy is fully examined, it is found that the four logical states 'for all' ($\forall$), 'there exists' ($\exists$), 'for none' (ϕ) and 'not for all' ($\wedge$) of QPL had to be supplemented by four more states in order to lead to the complete set of elements of BA-3.

'REVERSE' OPERATORS AND NEW LOGICAL STATES 'DOUBTFUL' AND 'IMPOSSIBLE'

The working out of problems in CL is most conveniently done (and implemented in a computer) by making use of the isomorphism T ≡ 1, F ≡ 0 between CL and BA-1. When this is done, truth tables of the type shown in table 1 result. In this table, the first two rows indicate the notation of the operators in classical logic[1]. To indicate explicitly that a connective (e.g. "or") is operative only in CL, but not necessarily in SNS, we use capital letters (e.g. "OR") and its operation is shown by row 3 of table 1. In this forward direction, the output is always again T or F. A connective operative in SNS is indicated by a single heavy letter as in the last row of table 1.

TABLE 1

Examples of isomorphism between CL and BA-1

CL	A	OR	B	AND	C	NOT	D	IF
	$a \vee b = c$		$a \,\&\, b = c$		$\neg a = b$		$\neg a \vee b = c$	
BA-1	$a \oplus b = c$		$a \otimes b = c$		$a' = b$		$a' \oplus b = c$	
	ab	1 0	ab	1 0	a	b	ab	1 0
	1	1 1	1	1 0	1	0	1	1 0
	0	1 0	0	0 0	0	1	0	1 1
SNS	$a\mathbf{O}b = c$		$a\mathbf{A}b = c$		$a\mathbf{N} = b$		$a\mathbf{I}b = c$	

On the other hand, for a 'reverse' operator (as defined in 1a and b), the output state could sometimes be neither completely true, nor completely false. This is best illustrated by taking the CL truth table for a OR b = c in table 1A, and defining the 'reverse' relation as follows:

Reverse 'or' : $c \overset{\leftarrow}{\vee} a = b$ (1a)

is equivalent to

Given $a \vee b = c$ and the states of c and a, what is the state of b? (1b)

An inspection of table 1A shows that if c is true, then if a is also true, b *may be either true or false*; and if a is false b is necessarily true. On the other hand, if c is false, and a is true, there is no solution for b,—i.e. b can be *neither true nor false*, which means that the question is self-contradictory and the solution is 'impossible'. However, if c is false and a is false, b is necessarily true. We have named the two new states ('either T or F' and 'neither T nor F') as 'doubtful' (D) and 'impossible' (X), as in (2a, b):

$$T \vee F = D \quad (2a)$$

$$(\neg T) \,\&\, (\neg F) = X \quad (2b)$$

It is interesting that, in Ancient India (in particular Jaina) Philosophy, the four states, consisting of the states T and F, and the two extra ones D and X explained above, are specifically defined. We may quote the following from the section, dealing with "Syād-Vāda" (Doctrine of doubt) of

Ref. 3 regarding the philosophy of truth and knowledge: "... thus adding, with characteristic love for subtlety, two more alternatives "both 'is' and 'is not'" and "neither" 'is' nor 'is not'", to the well-known ones of 'is' and 'is not'".

In conventional CL questions of the type given in (1a and b), involving reverse connectives do get asked and the results that come out are usually carried over—e.g. by taking both T and F to be possible (corresponding to our D state) and considering each possibility in turn while analysing the results of further steps in the argument. Obviously if more than one D state occurs during this process, then *several* possibilities will have to be worked out one by one. We have found that, by having the D state and carrying on with that state in all the later steps of the argument, it is much simpler to solve such problems. However, there is a need for an explicit representation of these two new states for systematising the working out of the logic and also for converting the steps into algorithms for a computer[4,5]. One method is to take the states D and X as given in Eq. (2), and make their interpretation exact, by having two electrical lines, each having two voltage states 1 and 0. If these lines are called α and β then obviously T corresponds to $\alpha = 1$, $\beta = 0$ and F to $\alpha = 0$, $\beta = 1$. Also, it follows from (2a) and (2b) that for D, $\alpha = 1$ and $\beta = 1$, while for X, $\alpha = 0$ and $\beta = 0$. Thus the four states T, F, D, X can be beautifully implemented by electronic circuits[4,6].

Thus, if we represent, in general, the logical state of a term a by the electrical *states* (α β) *of the two lines*, then, the four states are representable by Eq. (4) in the next Section 3. We came to this representation only by such an intuitive approach. It was long afterwards that the identity of the algebra of this with BA-2 was discovered and established.

NECESSITY OF BOOLEAN ALGEBRA OF GENUS 2 FOR SNS

It is clear that while the one-element Boolean algebra is satisfactory for CL, the answers to questions, involving reverse operators, as in 2(a) and (b), are not representable in that system. Therefore, we have to consider the next simplest of Boolean algebras, namely BA-2. Since by its very name it has two generators, it is obvious that all terms in this logic can be represented by two Boolean elements α and β. Thus, in BA-2 the state of a term, in general, can be represented by a two-element Boolean vector

$$a = (a_\alpha \ a_\beta) \tag{3}$$

The four possible states that this can assume are,

$$T = (1\ 0),\ F = (0\ 1),\ D = (1\ 1),\ X = (0\ 0). \tag{4}$$

It is obvious that, this representation of BA-2, satisfies equations 2(a) and 2(b) *via* 5(a) and 5(b), and that it also satisfies the condition that the states T and F are mutually exclusive by being the complements of one another,—as in (5c) and (5d).

$$T \oplus F = D \text{ (a)},\ T \otimes F = X \text{ (b)},$$
$$T' = F \text{ (c)},\ F' = T \text{ (d)}. \tag{5}$$

It is also seen that the right hand sides of all the four equations in (5) are again contained within the set of four vectors in (4). This means at once that the set of four vectors in (4) are closed under any number of 'forward' applications of the operations $\oplus$, $\otimes$, $'$.

We have already seen that SNS requires four states T, F, D, X satisfying the equations in (4) and (5). We shall therefore examine its properties more in detail, and show that the mathematical formalism of BA-2 contains all that is necessary for implementing the logical terms, connectives and equations of SNS (i.e. CL extended to take care of D and X, in addition to T and F).

We shall also show that the Boolean vector representation of the possible states of a term in SNS is perfectly compatible with CL also. What is more interesting is the fact that, not only are the CL operators like AND, OR, NOT representable by 2×2 Boolean matrices in BA-2, but that we can also implement, *via* BA-2, in a very natural manner the *reverse* operators mentioned earlier (see tables 2 and 3). We shall now discuss this vector-matrix formalism of BA-2—first, exclusively for CL, in Section 3, and then, generally for SNS, in Section 4. Lastly, in Section 5, we shall consider briefly the relevance of BA-3 for quantifier logic.

VECTOR-MATRIX FORMULATION OF BA-2 AND ITS USE FOR CL

Since all vectors in BA-2 have two elements, a Boolean matrix $|Z|$ connecting them will be 2×2, with 4 elements (corresponding to $Z_{\alpha\alpha}$, $Z_{\alpha\beta}$, $Z_{\beta\alpha}$, $Z_{\beta\beta}$), each of which can have only the values 1 and 0. Hence, there are only 2^4(= 16) different matrices in BA-2 and these are listed in table 2. This table is in two parts. Part A contains ten matrices, which correspond to known connectives of CL. The identification of the matrix corresponding to the connective is made by calculating the truth table of the connective in CL (last column of table 2) and obtaining the matrix by putting T = 1, F = 0 for the four entries of the truth table. We shall not comment on Part B of table 2, since these operators do not occur as such in sentential logic, and they are expressible as combinations of two or more matrix operators listed in Part A—e.g. $|R| = |A| \oplus |I^c|$, $|D| = |O| \oplus |I|$, $|D^c| = |A| \otimes$ I^c

Coming back to Part A, the ways in which these matrices can be used to implement the

TABLE 2

The 16 Boolean matrices of BA-2 and the corresponding SNS operators

Sl. No	Syād nyāya system			Classical logic	
	Matrix	Name	Symbol	Symbol	Truth table
	A. Known in classical logic				
1	$\begin{pmatrix}1&0\\0&1\end{pmatrix}$	Equivalent	**E**	$\Leftrightarrow$	$\begin{matrix}T&F\\F&T\end{matrix}$
2	$\begin{pmatrix}0&1\\1&0\end{pmatrix}$	Negation, exclusive or	$\mathbf{E}^c(\mathrm{N})$	$\not\Leftrightarrow$	$\begin{matrix}F&T\\T&F\end{matrix}$
3	$\begin{pmatrix}1&0\\0&0\end{pmatrix}$	And	**A**	&	$\begin{matrix}T&F\\F&F\end{matrix}$
4	$\begin{pmatrix}0&1\\1&1\end{pmatrix}$	Nand	$\mathbf{A}^c$	$\mid$	$\begin{matrix}F&T\\T&T\end{matrix}$
5	$\begin{pmatrix}0&1\\0&0\end{pmatrix}$	Not imply	$\mathbf{I}^c$	$\not\Rightarrow$	$\begin{matrix}F&T\\F&F\end{matrix}$
6	$\begin{pmatrix}1&0\\1&1\end{pmatrix}$	Imply	**I**	$\Rightarrow$	$\begin{matrix}T&F\\T&T\end{matrix}$
7	$\begin{pmatrix}0&0\\1&0\end{pmatrix}$	Not implicate	$\mathbf{J}^c(\mathbf{I}'^c)$	$\not\Leftarrow$	$\begin{matrix}F&F\\T&F\end{matrix}$
8	$\begin{pmatrix}1&1\\0&1\end{pmatrix}$	Implicate	$\mathbf{J}(\mathbf{I}')$	$\Leftarrow$	$\begin{matrix}T&T\\F&T\end{matrix}$
9	$\begin{pmatrix}0&0\\0&1\end{pmatrix}$	Nor	$\mathbf{O}^c$	$\veebar$	$\begin{matrix}F&F\\F&T\end{matrix}$
10	$\begin{pmatrix}1&1\\1&0\end{pmatrix}$	Or	**O**	$\vee$	$\begin{matrix}T&T\\T&F\end{matrix}$

Table 2 (Continued)

Sl. No.	Syād nyāya system			Classical logic	
	Matrix	Name	Symbol	Symbol	Truth table
	B. Remaining six possible matrices				
11	$\begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}$	Row true[a]	R(R$_1$)		T T F F
12	$\begin{pmatrix}0 & 0\\ 1 & 1\end{pmatrix}$	Row false[a]	R^c(R$_2$,R^t)		F F T T
13	$\begin{pmatrix}1 & 0\\ 1 & 0\end{pmatrix}$	Column true[a]	C(C^t)		T F T F
14	$\begin{pmatrix}0 & 1\\ 0 & 1\end{pmatrix}$	Column false[a]	C^c(C$_2$,C^t)		F T F T
15	$\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$	Doubter[a] (Unrelater[a])	D		T T T T
16	$\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$	Contradicter[a] (Nullifier[a])	D^c(X)		F F F F

[a]These names have been newly coined, and these matrices are included for completeness. All of them can be obtained from two suitable matrices in Part A, *via* a Boolean sum or product (see text).

operation of logical connectives (unary and binary) are given in a summarized form in table 3. This table has been prepared with SNS in view, but it is interesting that the matrix calculus is equally valid for CL, provided it is remembered that the inputs can only be T or F in CL and that the outputs are also to be expressed in terms of T and F. All matrix products employ Boolean sums and products (as in table 1A and 1B). The Boolean Dirac bracket product $\langle a|K|b\rangle$ is defined as follows:

Let

$$\langle a| = (a_\alpha \; a_\beta), \quad \langle b| = (b_\alpha \; b_\beta) \tag{7a}$$

and

$$|K| = (K_{\lambda\mu}), \quad \lambda = \alpha, \beta; \quad \mu = \alpha, \beta \tag{7b}$$

Then

$$\langle a|K|b\rangle = k \text{ (a scalar) } = \sum_{\lambda = \alpha\beta} \sum_{\mu = \alpha\beta} = K_{\lambda\mu} a_\lambda b_\mu \tag{7c}$$

We also need the complement of a vector $\langle a|$ or matrix $|K|$, denoted by $\langle a^c|$ and $|K^c|$, in which *each* element of the vector, or matrix, assumes its Boolean complement value in BA-1. We also use the superscript t to indicate the transpose of a matrix $|Z|$. Thus;

$$Z^c_{\lambda\mu} = (Z'_{\lambda\mu}) \tag{7d}$$

$$Z^t_{\lambda\mu} = Z_{\mu\lambda}; \; \lambda, \mu = \alpha, \beta \tag{7e}$$

a) Binary connectives

It can be shown that the BA-2 formula contained in table 3H, in terms of Boolean Dirac bracket products, is the representation in BA-2 of the corresponding binary connective operation in CL of table 3G. The proof is simple if we note that both $\langle a|$ and $\langle b|$ in the Dirac bracket can only have the pure states T and F, in CL. Then, only one term of (7c) survives, and the corresponding $K_{\lambda\mu}$ is 1 or 0, accordingly as the logical truth table has T or F in the location (λ, μ).

It is again easy to show that, in CL, the second equation $\langle a|Z^c|b\rangle = c_\beta$, in table 3H, is redundant, and follows from the first, namely

TABLE 3
Analogy between SNS and BA-2

	SNS (also for CL)	BA-2[a]
Valid for both CL and SNS	*A. Terms* a, b, etc., having T, F, D, X	*B. Vectors* (a_α, a_β), with $a_\alpha, a_\beta = 0, 1$
	C. Connectives Examples are N = negation, A = and. All others are derivable from N and A	*D. Matrices* (2 × 2 *Boolean) like* $\lvert N\rvert = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$; $\lvert A\rvert = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$
	E. Unary relation e.g. ¬ a = b corresponds to a N = b	*F. Matrix product* e.g. $\langle a\rvert N\rvert = \langle b\rvert$, which yields $a_\alpha = b_\beta, a_\beta = b_\alpha$
	G. Binary (forward) relation e.g. a & b = c In general a Z b = c	*H. Dirac bracket product* $\langle a\rvert Z\rvert b\rangle = c_\alpha$ $\langle a\rvert Z^c\rvert b\rangle = c_\beta (Z^c_{\lambda\mu} = Z'_{\lambda\mu})$ $\langle c\rvert = (c_\alpha c_\beta)$
	I. Binary (reverse) relation e.g. c $\overleftarrow{Z}$ a = b stands for a Z b = c is given, and the states of c and a are known. To find state of b	*J. via Unary relations* If c = T, $\langle a\rvert Z\rvert = \langle b\rvert$ If c = F, $\langle a\rvert Z^c\rvert = \langle b\rvert$ If c = D,X; b = D, X
Only in SNS	*K. SNS binary relation*[b] Unanimity operator *U* gives a *U* b = c Vidya (knowledge) operator *V* gives a *V* b = c	*L. Boolean algebra operation* *U*: $a_\alpha \oplus b_\alpha = c_\alpha$; $a_\beta \oplus b_\beta = c_\beta$ *V*: $a_\alpha \otimes b_\alpha = c_\alpha$; $a_\beta \otimes b_\beta = c_\beta$

[a] All additions and multiplications follow BA-1, as in Table 1.
[b] In addition, there is a purely SNS unary operator *M*, which stands for complementation of the vector. Thus for *M* a = b, we obtain $a_\alpha = b_\beta, a_\beta = b_\alpha$.

$\langle a\rvert Z\rvert b\rangle = c_\alpha$. Thus if one of c_α, c_β is 0, which is always so for the pure states T and F of CL, only one combination, say $Z_{\lambda\mu}$, is relevant, and accordingly, if $a_\lambda Z_{\lambda\mu} b_\mu = 1(0)$ then $a_\lambda Z^c{}_{\lambda\mu} b_\mu = 0(1)$.

However, in SNS, the values of c_α and c_β are independent, and the full capabilities of the formulae in table 3 are required for implementing SNS *via* BA-2. Here again, the proof follows from the equation

$$\langle a_T\rvert Z\rvert b\rangle \cdot \oplus \langle a_F\rvert Z\rvert b\rangle = \langle a_T \oplus a_F\rvert Z\rvert b\rangle = \langle a_D\rvert Z\rvert b\rangle \quad (7)$$

and similarly when b_D occurs, and hence also when a_D and b_D both occur.

b) Unary operators

There is one operator in CL, namely NOT(¬), which has the property that, for any a, ¬ ¬ a = a. If we take the matrix corresponding to this from table 2, and apply it twice, it follows that

$$\langle a\rvert N\rvert N\rvert = . \langle a\rvert E\rvert = \langle a\rvert \quad (8)$$

This is the proof for the first tautological equivalence given in the set of these on p. 34 of Ref. [1].

In this case, the binary relation a N b = c corresponds to the logical connective XOR. In the same way, the other 8 binary connectives in table 2A can also have a unary counterpart, but these are best considered after the reverse connectives are discussed in Section 4.

c) Verification of standard tautologies

We shall first consider the tautological equivalences (listed on p. 34 of Ref. [1]). All of them are readily verified using tables 2 and 3. However, we shall take a few of these, and illustrate how beautifully these are fitted by the vector-matrix formalism. In the equations below, the l.h.s. are the tautologies as listed in Suppes' book, and on the r.h.s. are the corresponding matrix equations from our formalism.

$$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P) \qquad |I| = |N|J|N| \qquad (9a)$$

De Morgan's Laws

$$\neg(P \ \& \ Q) \Leftrightarrow \neg P \vee \neg Q \qquad |A^c| = |N|O|N| \qquad (9b)$$

$$\neg(P \vee Q) \Leftrightarrow \neg P \ \& \ \neg Q \qquad |O^c| = |N|A|N| \qquad (9c)$$

Commutative Laws

$$P \& Q \Leftrightarrow Q \& P \qquad |A| = |A^t| \qquad (9d)$$

Implication expressed as disjunction

$$P \Rightarrow Q \Leftrightarrow \neg P \vee Q \qquad |I| = |N|O| \qquad (9e)$$

We shall give one example of a tautological implication. It is

$$(P \Rightarrow Q) \ \& \ (Q \Rightarrow R) \Rightarrow (P \Rightarrow R) \qquad (10a)$$

Anticipating the discussion in Section 4 about unary operations, we write the vector matrix equations of l.h.s. of (10a) as

$$\langle P|I| = \langle Q|; \qquad \langle Q|I| = \langle R| \qquad (10b)$$

so that

$$\langle P|I|I| = \langle R| = \langle P|I|,$$

$$\text{since } |I|I| = |I| \qquad (10c)$$

which is the r.h.s. of (10a).

VECTOR-MATRIX FORMULATION OF SNS, INCLUDING REVERSE OPERATORS AND NEW SNS CONNECTIVES VIDYA AND UNANIMITY

We have seen in the last section that the Dirac product formula in table 3H is valid for all inputs, including the state D(1 1) for a and b. We shall now examine the nature of 'reverse' connectives.

a) Reverse operators

If only the classical states T, or F, form the inputs in a relation of the type $c \overleftarrow{Z} a = b$, then, the equations given in table 3J, can be used to determine b. We shall illustrate its use with CL inputs for c and a, to obtain the state of b, for $c \overleftarrow{O} a = b$ by considering the four possible combinations of (c, a), namely (T, T), (T, F), (F, T) and (F, F), in Eqs. (11) and (12) below.

$$\text{If } c = T; \ \langle a|O| = \langle b| \qquad (11a)$$

$$\text{For } a = T, \ b = (1 \ \ 0)\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix} = (1 \ \ 1) = D \qquad (11b)$$

and

$$\text{for } a = F, \ b = (0 \ \ 1)\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix} = (1 \ \ 0) = T \qquad (11c)$$

$$\text{If } c = F, \ \langle a|O^c| = \langle b| \qquad (12a)$$

$$\text{For } a = T, \ b = (1 \ \ 0)\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} = (0 \ \ 0) = X \qquad (12b)$$

and

$$\text{for } a = F, \ b = (0 \ \ 1)\begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} = (0 \ \ 1) = F \qquad (12c)$$

It is very interesting that, even for such a simple connective as OR, the effect of reversing $a \ O \ b = c$ as $c \ \overleftarrow{O} \ a = b$, can lead to all four states T, F, D, X as outputs for b, even though only CL states are fed in. Thus CL gets extended to SNS when 'reverse' operators are introduced. Interestingly, introducing the SNS states also as inputs for c or a does not lead to any further extension in the number of states. Thus, c = D makes b = D in the reverse connectives equation, whether a is T, F or D. For a = D, the formulae

$$\langle a|Z| = \langle b| \qquad (13a)$$

$$\langle a|Z^c| = \langle b| \qquad (13b)$$

accordingly as c = T or F, is equally valid for the SNS input of $\langle a| = D$.

We have examined the utility of the equations for forward and reverse connectives in SNS in a variety of problems and have even written a FORTRAN program[6], named MATLOG for this purpose. We have also worked out algorithms whereby a collection of statements at random can be rearranged to be in a sequential manner, such that the logical inputs for any statement are available as outputs of previous steps, or as original inputs of the problem. A discussion of these is reserved for a separate communication.

b) *The doubtful state*

We shall now consider the physical nature of the states D and X, in some detail. Thus, the doubtful state D stands for one in which the term "may be true", or "may be false". If we think of the states T and F as the 'pure' states, which are mutually exclusive, then D becomes a 'mixed' state (very much like a mixed state in quantum mechanics) composed of a mixture of the pure states (1 0) and (0 1). Consequently, if we ask the question, "Can the entity be true?" when it is in the D state, the answer would be 'yes'. If we then ask the question "Can it be false?", the answer again would be 'yes'. In effect, our present knowledge of the state is indefinite and it is this indefiniteness of knowledge that is embodied in the symbolic representation of D as (1 1).

The physical description of the 'impossible' state X is given in the next Section 4(d).

c) *The new connective for 'unanimity' U*

We shall discuss below the process of "purification" of the doubtful state, which involves an understanding of the logic of the two new operators *U* and *V*, as well as the nature of the state X. The definitions of the purely SNS connectives *U* and *V* are contained in table 3K and L. It is clear from these that they are describable in terms of the Boolean *additions and multiplications of state vectors*, which is quite different from the application of the *matrix operators* for CL connectives. However, they are not really so different as all that. This will be clear from the representation of the effects, on the individual α and β lines of SNS vectors, of the four connectives **O**, A, *U* and *V* given below in (14) and (15). In the former two (**O** and A), the CL equations on the left of (14) for the α-components are duplicated on the right for β-components by the de Morgan relations.

$$\mathbf{O}:\ a_\alpha \text{OR } b_\alpha = c_\alpha;\ a_\beta \text{AND } b_\beta = c_\beta \qquad (14a)$$

$$\text{A}:\ a_\alpha \text{AND } b_\alpha = c_\alpha;\ a_\beta \text{OR } b_\beta = c_\beta \qquad (14b)$$

On the other hand, Eqns (15a and b) also use AND and OR for individual lines α and β, but the CL operator is the *same for both* the α and β components, following the Boolean sum and product in BA-2, as in table 3J. The β-line is not derivable from the α-line, and the operator (*U* or *V*) acts on an SNS state ($\alpha\beta$).

$$U:\ a_\alpha \text{OR } b_\alpha = c;\ a_\beta \text{OR } b_\beta = c_\beta \qquad (15a)$$

$$V:\ a_\alpha \text{AND } b_\alpha = c;\ a_\beta \text{AND } b_\beta = c_\beta \qquad (15b)$$

We have implemented in an analog computer, using IC chips for AND and OR, the effects on the α and β-lines of **O** and A, and all other matrix connectives. The same technique has been employed for *U* and *V* also. Since it employs two lines α and β, and uses the Syād state D, it is named ESNY (Electronic Syād Nyāya Yantra (= machine))[7].

TABLE 4

SNS truth tables for U and V

A. a *U* b

a \ b	T	F	D	X
T	T	D	D	T[a]
F	D	F	D	F[a]
D	D	D	D	D[a]
X	T[a]	F[a]	D[a]	.X

B. a *V* b

a \ b	T	F	D	X
T	T	X	T	X
F	X	F	F	X
D	T	F	D	X
X	X	X	X	X

[a]These follow from the algebra of BA-2; but their utility for logic has to be considered further.

Using (15a and b), it is a simple matter to work out 4 × 4 truth tables for the four SNS states, and this is given in table 4 for *U* and *V*. Considering the truth table 4A for U, it is seen that a pure state (T or F) combined with itself by the unanimity operator gives again the same state, just as for **O** and **A**. On the other hand TUF = D, FUT = D and all other combinations give D as output for $a U b$. (X is an impermissible input for a or b, and hence the fourth row and column need not be considered. (What is given in table 4A are consequences of BA-2.)) Thus, U has the very simple property that, if a and b have the same (permissible) state, the c has also the same state; otherwise c = D. The same will be true of $a_1 U a_2 U \ldots\ldots U a_n = c$, for which c is T (or F) only if *all* a_j are T (or F); otherwise it has no information value. Hence the name 'unanimity' for this operator.

d) Vidya operator and the impossible state X

Considering $a V b$ in table 4B, we find the beautiful result that if one of a or b is T, but the other is F, then $a V b$ = X, showing that the result ($c = a V b$) is 'impossible' or 'contradictory'. It means that one of the inputs a, or b, is invalid or wrong. Therefore, one aspect of V is to check for consistency between a and b, and to detect contradiction.

On the other hand, if we take $a_{T(F)} V b_D$, then we get the D state purified to T, or F, as the case may be. This is, in fact, an extremely useful process, for this is the way in which knowledge (Vidya) is acquired about anything. Initially, the state (of knowledge) is one of absence of definite information,—Syād (may be true or false)—or $a_D^{(1)}$. When, however, definite information (as to T or F) is obtained, $a_{T(F)}$, then the data from the two sources can be put together and combined by V as $a_D^{(1)} V a_{T(F)}^{(2)}$. This gives for a the resultant state $a_{T(F)}^{(2)}$, i.e. definitely true, or definitely false and justifies the name Vidya for this operator. A little reflection will show that none of the ten CL connectives given in table 2 is suitable for this purpose, and only V has all the required properties.

More generally, the appearance of a contradiction (X) at any stage indicates that one of the previous steps is wrong (or inconsistent), and this can be used to trace the argument back and pinpoint exactly where the original cause of the contradiction is located. A general algorithm, using SNS formalism, is being worked out for this purpose, and will be reported elsewhere.

The possible uses of the matrices 11 to 16 in table 2B will not be discussed here, although matrices 15 and 16 occur quite often.

e) Logical analysis of a problem using SNS

We take a simple problem in which a person has to be judged guilty, or not guilty, of a crime, from the evidence given by two witnesses, one (w_1), who gives eye-witness evidence, and the other (w_2) giving *alibi* for the accused. The logical steps are summarised in table 5. We shall comment on the equations in Parts B and C. Considering the evidential value e_1 and e_2, if the first eye-witness says he saw the crime, then e_1 is T; but if he did not witness the crime (w_1 = F) no evidential value comes from him. Hence, the relation between w_1 and e_1 is the CL connective "implies" ($\Rightarrow$). Thus, $w_1 I e_1$ is true and w_1 is given, so that $\langle w_1 | I | = \langle e_1 |$. In the same way, possible *alibi* gives e_2 as F, while absence of *alibi* has no evidentiary value (e_2 = D). Hence, we again have an implication $w_2 I$, but it gives $\neg e_2$. Hence we have the relation, $\langle e_2 | = w_2 | I | N |$.

Part C of table 4 uses the Vidya operator V for combining the information provided by two different sources. The initial state is clearly D, since there is no evidence either for or against the guilt, to start with. When the evidence e_1 is obtained, the intermediate state of the guilt (g_2) is obtained by combining the initial state g_1 with the first evidence e_1 *via* V to give g_2. In the same way, we combine g_2 with e_2, the second evidence, via V, to give the final conclusion g. The truth values given in section C of table 5 arise from the four possible combinations of (w_1, w_2)—namely (T, T), (T, F), (F, T), (F, F). It is interesting that all four possible states T, F, D, X occur for the state of the final conclusion from the four different input combinations.

It will be seen from this example, that the SNS type of analysis of a logical problem is not only very informative, but is also in such a form that it can be implemented by a computer program. In fact, all operations involve only Boolean addi-

TABLE 5

Practical example requiring the new states D and X and the Vidya operator

Case	1	2	3	4	Remarks
	A. State of existence of the two evidences				
Of eye-witness w_1	T	T	F	F	T = saw crime
Of *alibi* w_2	T	F	T	F	T = *alibi* available
	B. Evidential value for proving guilt				
e_1 from $w_1 \Rightarrow e_1$	T	T	D	D	The implications are obvious
e_2 from $w_2 \Rightarrow \neg e_2$ (see text)	F	D	F	D	(see text)
	C. Steps for deciding whether guilty or not				
Initial state g_1	D	D	D	D	No evidence for or against guilt is available
First step $g\, V\, e_1 = g_2$	T	T	D	D	g_2 is the state of g when only e_1 is considered
Second step $g_2\, V\, e_2 = g$	X	T	F	D	Here, the second evidence e_2 is also taken into account
Nature of final conclusion	Contradictory[a]	Found guilty	Not guilty	Indefinite[b]	All four states can occur

[a]Obviously one of the two witnesses is giving false evidence.
[b]The available evidence is inconclusive, and new information is called for.

tions, multiplications and complementations, and equating the components of a pair of Boolean vectors.

EXTENSION TO BA-*n* AND MULTIVALUED LOGIC

a) QPL and its extension SBL isomorphic to BA-3

When the matrix calculus of SNS logic became quite clear, we attempted to extend this to Quantified Predicate Logic (QPL). As is well known, the quantifiers commonly employed in QPL are four in number—namely,

For all $\forall$, There exists $= \exists$, For none $= \Phi$, Not for all $= \Lambda$ (16)

It is also known that these are interconnected *via* operations involving negations of the quantifier states as well as negations of the state of the term which is quantified. Thus,

$$\neg \forall = \Lambda,\ \forall \neg = \Phi,\ \neg \vee \neg = \exists \qquad (17)$$

We tried to represent these in some vector-matrix form so that they could be manipulated with, in the same manner as in SNS *via* BA-2. We found at a very early stage that this required *three* electrical lines (γ, δ, ϵ), each of which can have the two Boolean states 1 and 0. We then examined the nature of γ, δ, ϵ, when we found that γ corresponds to $\forall$ (for all), and ϵ to Φ (for none). However, the third line δ did not correspond to either one of the other two quantified states $\exists$ and Λ. Very soon we found that δ corresponds to a new state, which we have named 'for some' (Σ), whose properties are best represented by the equation

$$\exists \;\&\; \neg \forall = \Sigma$$

Physically, 'some' is seen to be the state which covers the whole range of existence excepting the extremes of complete absence and complete

TABLE 6

Complete set of states in SBL isomorphic to BA-3

Name [a]	Symbol	Description as per Śaptabhangi	SBL description	BA-3 description
For all,	$\forall$	asti (is)	Pure	(1 0 0)
For none,	Φ	nāsti (is not)	Pure	(0 0 1)
For some,	Σ	avaktavyah (inexpressible)	Pure	(0 1 0)
There exists,	$\exists$	asti ca avaktavyah (is and inexpressible)[c]	$\forall \oplus \Sigma$	(1 1 0)
All or none,	θ	asti nāsti (is and is not)[c]	$\forall \oplus \Phi$	(1 0 1)
Not for all,	$\wedge$	nāsti ca avaktavyah (is not and inexpressible)[c]	$\Phi \oplus \Sigma$	(0 1 1)
Universal doubt,	$\triangle$	asti ca nāsti ca avaktavyah (is, is not and inexpressible)[c]	$\forall \oplus \Phi \oplus \Sigma$	(1 1 1)
Impossible (in SBL)	$\star$	No term is available for this	$\forall \otimes \Sigma = \Sigma \otimes \Phi = \forall \otimes \Sigma$	(0 0 0)

[a]The last four symbols have been coined by us.

[b]This is copied from Ref. 5. The word Syād(may be) is omitted.

[c]The 'and' in these descriptions is to be taken to mean "may be this and that".

presence. 'There exists' ($\exists$) has already the property that it omits 'none'; but includes all other types of existence, including 'all'. So the best way to understand this state 'some' is to say that it corresponds to 'there exists, but not for all'.

The Boolean algebraic nature of the complete set of 8 states generated by presence or absence of the voltage in three electrical lines γ, δ, ϵ, (which are mutually exclusive), thus becomes clear. The three are isomorphous to the three generators of the BA-3 representing an extension of QPL. Then any combination of these *via* the Boolean operators $\oplus$ and $\otimes$ should also lead to a state within the complete Boolean algebra. When this was written down, it was clear that both 'there exists' and 'not for all' are superpositions of two states (see table 6). In addition to these, three more new states became defined, namely θ, $\triangle$, $\star$, as shown in table 6. We have thus extended the limited Boolean algebra of the four quantifier states in standard QPL to form an 8-element Boolean algebra BA-3, which is closed under Boolean addition and multiplication.

Just as in the case of Syād (D) in SNS, the Jaina philosophers of ancient India had recognized the seven possible states of BA-3 (omitting the eighth impossible state)—see Ref. 3, p. 164. The names given by them for the seven states are also given in table 6 and they form what they called the *Saptabhangi* (7-fold state of truth). Therefore, we have given the name SBL Saptabhangi logic) for the extended form of QPL made so as to form a closed complete algebra BA-3. We shall not discuss anything further about QPL and SBL, since the detailed implementation of problems using this logic has yet to be worked out.

Lastly, we shall comment on the third state, besides T and F, that is contained in Brouwer's "Intuitionistic Logic" (IL). This state, which we may denote by B (the first letter in the name of its originator), appears to have some properties in common with our D state, at first sight. But there is a great difference, since D can be purified to T or F, while B cannot be modified in this way, because, in IL, a proposition, in the B state, *can never* be proved to be either T or F. Brouwer's third state (B) is, in fact, very similar to the third state (Σ) in SBL, and even the SBL description 'avaktavyah' for this state is similar to the IL concept of "can never be proved to be either true or false".

Thus, the CL states T and F, together with B, has the same algebraic structure as SBL, or BA-3. We can therefore say that our Boolean vector-matrix formalism, in terms of analogies with BA-1, BA-2 and BA-3, explains in a very simple manner the nature of the interconnections between terms, connectives, etc., which occur in most of the logic that is used now-a-days.

In fact, the Boolean vector-matrix formulation seems to be a very suitable one for implementing multi-valued logic by the Boolean algebras BA-*n*, but this is beyond the scope of this article.

ACKNOWLEDGEMENTS

This article contains in a summarised form, the results of a number of studies made during the last two and a half years, and which are partly embodied in the Matphil Reports listed in the references, and others. A complete electronic implementation of BA-2 for SNS has also been designed but not yet constructed. In all these studies, the author had the able assistance of Mr. T. A. Thanaraj and Mr. R. E. C. Johnson, respectively in the field of software and hardware of the implementations, for which he is most grateful. Although the paper was written entirely after his assuming the CSIR Distinguished Professorship, the work was carried out by the author mostly at the Indian Institute of Science, Bangalore, and partly at the Centre for Cellular and Molecular Biology, Hyderabad, and he is grateful to the Directors of these Institutes for all facilities that were provided for studies in this unorthodox field. The author is deeply grateful to Mr. M. Suresh for carefully typing several drafts of this paper in the evenings and on holidays.

1. Suppes, P., 'Introduction to Logic', 1957, van Nostrand, New York.
2. Birkhoff, G. and MacLane, S., 'A Survey of Modern Algebra', 4th Edn., 1977, Macmillan, New York.
3. Hiriyanna, M., 'Outlines of Indian Philosophy', 1921, George Allen & Unwin, London.
4. Ramachandran, G. N., 'Computerization of Logic', Matphil Reports No. 8, 1980.
5. Ramachandran, G. N. and Thanaraj, T. A., 'Fortran Program for Sentential Logic', Matphil Reports No. 12, 1980.
6. Ramachandran, G. N. and Thanaraj, T. A., 'Matrix Algebra for Sentential Logic and the Fortran Program MATLOG', Matphil Reports No. 18, 1981.
7. Ramachandran, G. N., Johnson, R. E. C. and Thanaraj, T. A., 'Electronic Syād Nyāya Yantra (ESNY-2)—Analog Computer for Sentential Logic', Matphil Reports No. 13, 1980.

ANNOUNCEMENT

JOURNAL OF MINES, METALS AND FUELS—SPECIAL NUMBER ON THICK SEAM MINING

The Journal intends to provide a technology assessment on thick seam coal mining in the above referred Special Number which provides a detailed country-wise assessment of the problems and prospects of thick seam mining, and also a global overview, carrying contributions, illustrated with photos and diagrams of equipment, from the leading mining engineers in India and abroad. The issue serves as the window to the world of thick seam mining, an area of technology which has been much neglected hitherto.

The special number, running into 130 pages has been priced at Rs. 35.00; £ 10.00 or $ 20.00 plus postage and registration charges of Rs. 3.00; £ 0.30 or $ 0.70 (sea mail). It is hoped that in terms of its rich contents the Special Number would veritably serve as an advanced text on the subject.

The special number of the Journal is published by M/s. Books & Journals (P) Ltd., 6/2 Madan Street, Calcutta 700 072.