



I-D2-1

BOOLEAN SUPPORT OF ERN LOGIC

INTRODUCTION

Let's start with a short quotation of Boole:

-They who are acquainted with the present state of the theory of Symbolic Algebra, are aware that the validity of the processes of analysis does not depend upon the interpretation of the symbols which are employed, but solely upon the laws of their combination.-

The common usage forces us to use the term Propositional Calculus (PC). However, we shall use it as a strict synonym of Boolean Algebra (BA). Operands and operators of BA are associated with a variable ranging over binary digits 0-1, which we call "Plausibility" for the sake of consistency with the Fuzzy Inference of our ultimate objective, the ERN Logic (chapter "ERN LOGIC").

Original BA is 2 dimensional: its expressions consist of an operator taking plausibility of two operands as input and determining its own plausibility as output. Operands are determined uniquely as carriers of plausibility and don't admit other interpretation.

Operators may become in turn operands of higher level expressions, in which case they are treated as operands, i.e. plausibility carriers and nothing else. 2 dimensional BA admits 16 operators and in its improved Polish Notation of Lukasiewicz supports binary gates underlying computing and electronic technology.

I-D2-2

Its pretended synonym, the pseudo-logical "Propositional Calculus" is a misnomer confusing Operand with "proposition" following noumenal Logic's traditional striving to be rooted in natural languages and via them in the noumenal reification. The same tendency confuses the binary, 0-1 range of plausibility with noumenalistic meaningless terms "truth-falsity". Throughout its history, noumenal Logic considered propositions as its unique operands. It never inquired where the "truth" of premises comes from, assumed it arbitrarily and dealt exclusively with its propagation to conclusions. Corrupted Boolean Algebra has been snatched and incorporated into noumenal Logic as an efficient propagation tool.

We have seen in "Natural Model" that, being an enhancement of the natural faculty, extrinsic logical systems may only be justified by extending and simulating the ER structures of the natural inferring faculty and the fuzziness of the physical reality.

While direct use of BA as "logical" Propositional Calculus is irrational, extended BA may be useful as a conceptual base of the rational Logic embodied in the ER Network (ERN). It will be presented in the chapter "ERN LOGIC". Here we shall briefly mention that ERN may be conceived as BA extended over:

1. Network structure with Operands in vertices and Operators in edges.

2. N dimensional vertices (associated with N edges), which involves a very fast increase of operators number (ON) in function of N: $ON = 2^{(2^N)}$.

I-D2-3

Original 2D BA has 16 Operators which may be easily learned like the multiplication table and used spontaneously. For 3D we get 256 operators and for 100 sensors watching malfunctions in a plane, $2^{(2^{100})}$ operators, whose enumeration, let alone definition would overflow the Congress Library. And the dimensions of DNA exceed 6000. We discuss practical ways of dealing with this problem in "N DIMENSIONAL PROPOSITIONAL CALCULUS" below. Let's note that dimensionality is local, determined for each vertex by the number of associated edges.

3.continuous plausibility spanning real number range $\{0-1\}$ supporting fuzzy inference of the ERN logic.

4.Operands defined as, or reducible to Expressions, which makes ERN apt to formulate and to process most complex scientific and practical problems.

One might object to the apparent circularity: Logic being supposed to found Mathematics, cannot be properly founded in Boole Algebra, which is a mathematical construct. However, the circularity is only apparent. ERN is derived from inquiry about Mind and Reflection ("NATURAL MODEL") and our encounter with Boole Algebra during ERN derivation is coincidental.

I-D2-4

2 DIMENSIONAL PROPOSITIONAL CALCULUS

Propositional Calculus (PC) is an instance of Boole Algebra defining operands and operators carrying binary variable taking values 0/1, which we call "plausibility" for the sake of consistency with Fuzzy Inference of the ERN logic.

"Propositional Calculus" is a misnomer confusing "operand" with "proposition" due to Logic traditionally striving to be rooted in Noumenal View as expressed by natural languages. The same tendency confuses the 0-1 range of plausibility with naive noumenalistic terms "truth-falsity". Throughout its history, Logic considered propositions as its only operands, was not concerned with the plausibility of premises, assumed arbitrarily and dealt exclusively with extending it to conclusions. It incorporated misnamed PC's algebra as an efficient extending tool.

Our phenomenal ERN Logic starts by inquiring how and to which operands plausibility may be premised before extending it to other conclusion-operands. In this latter task we find PC as a useful support, considering it strictly as algebra and disregarding its pretended linguistic implications.

For two-dimensional Calculus operators operate on 2 operands and have a plausibility variable being function of those of both operands. In order to comply with the common usage we may say that operator or operand is "true/false", implying by it the values "1/0" of Boolean binary plausibility.

I-D2-5

For example operator "and" is true if and only if both operands are true, which may be symbolized for a couple of operands p, q as follows:

```

and(p,q)
1  1 1
0  1 0
0  0 1
0  0 0

```

For above 4 combinations of p,q we have clearly 16 operators (o1-o16) listed below:

```

p q  o1  o2  o3  o4  o5  o6  o7  o8
1 1  0   1   1   1   0   0   0   1
1 0  1   0   1   1   0   1   1   0
0 1  1   1   0   1   1   0   1   0
0 0  1   1   1   0   1   1   0   1

```

```

p q  o9  o10 o11 o12 o13 o14 o15 o16
1 1  1   1   0   0   0   1   0   1
1 0  0   1   0   0   1   0   0   1
0 1  1   0   0   1   0   0   0   1
0 0  0   0   1   0   0   0   0   1

```

Following operators are most frequently used (Lists of their plausibilities may be more concisely shown as horizontal strings.):

operator	horizontal string
o14: and	1000
o4: or	1110
o7: orr, exclusive or, either or	0110
o2: implication	1011
o8: equivalence	1001

I-D2-6

Operators may in turn become operands, which allows chains of operations, known as "inference chains", or shortly "inference" as shown in the simple example:

```
orr((and(p,q)),(or(r,s)))
      1,0      1,0
    0          1
1
```

Inference may extend over thousands of operations. Plausibilities of the lowest level (p,q,r,s) may be factual and inference determines their logical consequences. Besides the above mentioned binary operators the Calculus encompasses a unary operator "not" which operates on one operand and negates its plausibility replacing 1 by 0 and vice versa:

```
not(p)
0  1
1  0
```

Besides the essential symbols, i.e. operators, operands and brackets, we shall introduce for convenience expressions, explained in the following example.

Example of Calculus' operations.

An expression marked "En" encompasses a main operator followed by bracketed structure of operators/operands. Expression has an associated plausibility string which it shares with the main operator. En, Em having identical strings are equal and marked En=Em. Otherwise, they are not equal and marked En!=Em.

"=" symbol is also used to associate expression with its main operator: E1=and(pq).

Operators and expressions can become in turn operands.

I-D2-7

Thus, searching, for instance, the solution of:

(1) if $\text{or}(pq)$ implies $\text{and}(pq)$ then $X(pq)$ (where X is an unknown operator), we may write it in form of expressions:

$E1 = \text{imp}(\text{or}(pq), \text{and}(pq))$

$E2 = X(pq)$

$E3 = \text{or}(pq)$

$E4 = \text{and}(pq)$

thus $E1 = \text{imp}(E3, E4)$

We evaluate expressions starting by those in brackets:

$E3 = \text{or}(pq)$

1 1 11

1 1 10

1 1 01

0 0 00

$E4 = \text{and}(pq)$

1 1 11

0 0 10

0 0 01

0 0 00

$E1 = \text{imp}(E3, E4)$

1 1 1 1

0 0 1 0

0 0 1 0

1 1 0 0

Now, by equalizing $E1$ with $E2$ we may search X .

$E1 = E2 = X(pq)$

1 1 1 11

0 0 0 10

0 0 0 01

1 1 1 00

Looking up the operators table we see that $X = \text{eq}$ and we can write the solution of (1):

I-D2-8

if $\text{or}(pq)$ implies $\text{and}(pq)$ then q is equivalent to p

The same chain of operations could be written without simplification by E3, E4 in one step:

```
E1=imp(or(pq).and(pq))=E2=eq(pq)
1 1 1 11 1 11 1 1 11
0 0 1 10 0 10 0 0 10
0 0 1 01 0 01 0 0 01
1 1 0 00 0 00 1 1 00
```

It seems more difficult, but with a little practice it becomes very easy to write directly such summaries even for much more complex expressions. However, when in doubt, it's advisable to follow Descartes and to decompose a complex expression into several simple ones.

Let's note: Calculus tells us that if $\text{or}(pq)$ implies $\text{and}(pq)$ then p and q are equivalent. Result not quite obvious and rather useful for instance in programming where we can replace " $\text{imp}(\text{or}(pq).\text{and}(pq))$ " with simpler " $\text{eq}(pq)$ ".

Similarly:

if $(\text{or}(pq))$ does not imply $(\text{and}(pq))$
then p and q are mutually exclusive:

```
E1=not(imp(or(pq).and(pq)))=E2=orr(pq)
0 0 1 1 11 1 11 0 0 11
1 1 0 1 10 0 10 1 1 10
1 1 0 1 01 0 01 1 1 01
0 0 1 0 00 0 00 0 0 00
```

Exercise A

Many beginning programmers replace intuitively and wrongly $E1=\text{and}(\text{not}(p),\text{not}(q))$ with $E5=\text{not}(\text{and}(pq))$

I-D2-9

It's difficult to explain them their error without help of the Calculus and very easy to do it with help Calculus.

1.Show the error.

2.Find correct X in $E1=E2=\text{not}(X(pq))$.

First using intermediary expressions $E3=\text{not}(p)$
 $E4=\text{not}(q)$ and afterwards without them.

Exercise B

A theorem of Calculus says that any operator may be constructed from any two other ones with optional help of "and" and "not".

Show that "imp" may be constructed from "eq" and "or".

$E1=\text{imp}(pq) = E2=\text{or}(\text{eq}(pq), \text{and}(\text{not}(p), q))$

1 1 11 1 1 1 11 0 0 1 1

0 0 10 0 0 0 10 0 0 1 0

1 1 01 1 1 0 01 1 1 0 1

1 1 00 1 1 1 00 0 1 0 0

Thus $E1=E2$ QED

Exercise C

Show that "orr" may be constructed from "or" and "and".

N DIMENSIONAL PROPOSITIONAL CALCULUS

We have seen that for $n=2$, the 2-dimensional operators work on 2 operands (p,q) and that we have $2^{(2^n)}=16$ operators:

											1	1	1	1	1	1	
p	q	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6
1	1	0	1	1	1	0	0	0	1	1	1	0	0	0	1	0	1
1	0	1	0	1	1	0	1	1	0	0	1	0	0	1	0	0	1
0	1	1	1	0	1	1	0	1	0	1	0	0	1	0	0	0	1
0	0	1	1	1	0	1	1	0	1	0	0	1	0	0	0	0	1

The situation is simple, we know the 16 operators by heart, like the multiplication table and with a bit of practice can execute and program all operations of the 2d-PC from memory. For $n>2$ PC becomes much more complex. Let's start with $n=3$ and 3 operands p,q,r:

p	q	r	
1	1	1	01111111
1	1	0	10111111
1	0	1	11011111
1	0	0	11101111
0	1	1	11110111
0	1	0	11111011
0	0	1	11111101
0	0	0	11111110 etc

We have $2^{(2^3)}=256$ operators. Number of operators increases very fast with n. For $n=4$ we have $2^{(2^4)}=65536$ and for $n=5$ $2^{(2^5)}=2^{32}=4294967296$ operators. For practical applications 5 is small. We may have 20 symptoms of a disease or 100 "symptoms" of some breakdown in a jet plane. The respective diagnostic expert systems would extend over $2^{(2^{20})}$ and $2^{(2^{100})}$ operators. A bit too much to learn by heart, to describe in a textbook, or, for that matter, in the whole Congress Library.

I-D2-11

We have to look for some other procedures.
Let's come back to n=3. Some operators map from n=2 to n=3 as one to one, ex. "and", "or":

p	q	r	and	or
1	1	1	1	1
1	1	0	0	1
1	0	1	0	1
1	0	0	0	1
0	1	1	0	1
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

For any n they may be evaluated: "and" as product of all operands' plausibilities "or" as their maxof value.

However, for n=3 "orr" forks to 3 distinct operators "one-of", "two-of" and "not-all":

p	q	r	one-of	two-of	not-all
1	1	1	0	0	0
1	1	0	0	1	1
1	0	1	0	1	1
1	0	0	1	0	1
0	1	1	0	1	1
0	1	0	1	0	1
0	0	1	1	0	1
0	0	0	0	0	0

For n=20 "orr" will fork to 20 operators, from one-of to 19-of and not-all. On this example we see that for higher n's only a few operators can be chosen from endless lists in function of their utility for a particular problem.

As we have said before, the user has to tailor his logic to his problem by choosing pertinent operators and designing their evaluation algorithms.

Evaluation algorithms for some operators may become a bit complex even in the Exact PC. They become really difficult in the Fuzzy.

I-D2-12

Dimensions

Inference systems using PC are in general network structures. Each node is an Assertion. A node may be considered as an aggregate related top-down to several parts and as a part related bottom-up to several aggregates. A syndrome is an aggregate of its symptoms and a part of a disease. Relation Aggregate-part is "many-to-many": a syndrome may have several symptoms, a symptom may belong to several syndromes. Dimension of a node is the number of its parts.