

True or False: Evaluating Logical Expressions

Ronald J. Fehd, Centers for Disease Control and Prevention, Atlanta, GA, USA

ABSTRACT

Description : The SAS® language provides methods to evaluate logical expressions which then allow conditional execution of parts of programs. In cases where logical expressions contain combinations of intersection (and), negation (not), and union (or) users and later readers doing maintenance may legitimately and thoughtfully question whether the expression is correct.

Topics covered include theory of logic and logical evaluation in data step, macros, and procedures (where clause).

Audience : intermediate to advanced users.

Keywords : binary, boolean, DeMorgan's law, evaluation, fuzzy logic, logical operators, ternary

Information : Truth tables of binary, boolean, and DeMorgan's laws

Download : This paper and its programs are available on the web:

Contents

Logical Ideas describing Permutations of Truth Values	2
Not: Logical Negation	3
Both: And	3
Either: not Both	4
De Morgan's Rules for Nand and Nor	5
Neither: Nor	5
Summary of Rows in Two by Two Truth Table	6
Combinations	6
Exclusive Or: Xor	6
Either or Both: Or	7
Not And: Nand	7
Truth Tables	8
Data Step	9
Merging Data Sets	10
Test Data	11
Conclusion	11

Introduction

Overview

This paper provides a vocabulary of ideas that describe logical operators, set theory, and both sql joins and set operations.

Numbers are stored in computers as bits, which values may be either zero or one. Arithmetic operations are performed using sets of bits. Logical operations are performed on single bits with zero meaning false and one meaning true. This paper describes patterns of logical values, first using natural language ideas, then showing the practical applications in data step processing, merging data sets, and sql joins and set operations.

This table provides a short overview of the logical ideas and operators discussed in the following sections.

logic		values			logical			
P	Q	P	Q	sum(P,Q)	idea	operators		
T	T	1	1	2	both	and	or	and
T	F	1	0	1	only left	xor	or	nand
F	T	0	1	1	only right	xor	or	nand
F	F	0	0	0	neither	nor	nor	nand

Style

Some logical operators are not provided by the SAS language. When the logical concept is not available then the name is in *italics*; SAS language operators and functions are in `monospace text`.

Logical Ideas describing Permutations of Truth Values

Overview

This section discusses the logical ideas:

Not: Logical Negation	3
Both: And	3
Either: not Both	4
De Morgan's Rules for Nand and Nor	5
Neither: Nor	5
Summary of Rows in Two by Two Truth Table	6

The purpose of this section is to name logical operators that identify each permutation of logical values.

The ideas *not* and *and* are key to understanding all other logical operators.

Not: Logical Negation

Overview

The logical operator *not* is a unary operator: it acts on one argument. The function *not* changes its argument to the opposite truth value: not true is false, not false is true.

Truth Table

This is the truth table for *not*.

logic		values	
P	not P	P	not P
T	F	1	0
F	T	0	1

Notes: common expressions are without parenthesis: *not* P and with parenthesis: *not*(*logical expression*). There may be a space between *not* and the open parenthesis: *not* (P).

Both: And

Overview

The logical operator *and* is defined as both arguments being true; the sum of P plus Q is 2, which identifies this row.

Truth Table

This is the truth table for *and*.

logic			values			
P	Q	P and Q	P	Q	P and Q	sum(P,Q)
T	T	T	1	1	1	2
T	F	F	1	0	0	1
F	T	F	0	1	0	1
F	F	F	0	0	0	0

The combination logical operator *or* also identifies this row; the logical expression is P *or* Q.

Either: not Both

Overview

The logical idea either is defined as one of its arguments being true, but not both; the sum of P plus Q is 1.

Note the the sum function can differentiate between either.left: $\text{sum}(P, \text{not } Q)$ equals 2, and either.right: $\text{sum}(\text{not } P, Q)$ equals 2.

Either: Only Left : This is the truth table for Only.Left.

logic			values				
			only		not		sum
P	Q	P	P	Q	P	Q	(P, not Q)
T	T	F	1	1	0	0	1
T	F	T	1	0	1	1	2
F	T	F	0	1	0	0	0
F	F	F	0	0	0	1	1

Either: Only Right : This is the truth table for Only.Left.

logic			values				
			only		not		sum
P	Q	Q	P	Q	Q	P	(not P, Q)
T	T	F	1	1	0	0	1
T	F	F	1	0	0	0	0
F	T	T	0	1	1	1	2
F	F	F	0	0	0	1	1

The combination logical operators `xor`, `or` and `nand` includes both these permutations.

The logical expressions are:

- $P \text{ xor } Q$, in SAS: `bxor(P,Q)`
- $P \text{ or } Q$
- $P \text{ nand } Q$: in SAS: `either not(P and Q) or not P or not Q`

De Morgan's Rules for Nand and Nor

Explanation

De Morgan offered two explanations of each of two logical ideas: nand and nor.

1. nand

(a) Negation of a conjunction is the same as the disjunction of the negation of each of both terms.

(b) $\text{not } (P \text{ and } Q) = \text{not } P \text{ or not } Q$

2. nor

(a) Negation of a disjunction is the same as the conjunction of the negation of each of both terms.

(b) $\text{not } (P \text{ or } Q) = \text{not } P \text{ and not } Q$

Neither: Nor

Overview

The last row of the two by two truth table is identified by the logical idea *neither* and the logical operator *nor*. This row is identified by the sum of P and Q equal 0.

Truth Table

This is the truth table for the logical operator *nor*: not or.

logic			values					
nor			nor		not	not	sum	
P	Q	P, Q	P	Q	P, Q	P	Q	(not P, not Q)
T	T	F	1	1	0	0	0	0
T	F	F	1	0	0	0	1	1
F	T	F	0	1	0	1	0	1
F	F	T	0	0	1	1	1	2

De Morgan identifies two logical expression for *nor*:

- $\text{not } (P \text{ or } Q)$
 - $\text{not } P \text{ and not } Q$
-

Summary of Rows in Two by Two Truth Table

Overview

Each row is a permutation of truth values and can be identified by

- a combination of the `not` and `and` logical operators.
- the sum of a logical expression that equals two

Combinations

Overview

This section discusses the logical ideas:

Exclusive Or: Xor	6
Either or Both: Or	7
Not And: Nand	7

The purpose of this section is to provide logical operators which define sets of permutations of truth values.

Exclusive Or: Xor

Overview

The combination logical operator `xor` identifies rows whose sum equals 1.

Truth Table

This is the truth table for `xor`; note that `xor` identifies both `Only.Left` and `Only.Right`.

logic			values				
P	Q	P xor Q	P	Q	P xor Q	sum(P,not Q)	sum(not P,Q)
T	T	F	1	1	0	1	1
T	F	T	1	0	1	2	0
F	T	T	0	1	1	0	2
F	F	F	0	0	0	0	0

The set defined by the combination logical operator `xor` is included in the logical operators `or` and `nand` which are described next.

The SAS language provides the bit-wise function `bxor` to implement the logical operator `xor`; the logical expression is `bxor(P,Q)`.

Either or Both: Or

Overview

The combination logical operator *or* identifies rows whose sum is greater than or equal to one. It includes the sets: *xor* and *and*.

Truth Table

This is the truth table for *or*;

logic			values			
P	Q	P or Q	P	Q	P or Q	sum(P,Q)
T	T	T	1	1	1	2
T	F	T	1	0	1	1
F	T	T	0	1	1	1
F	F	F	0	0	0	0

Not And: Nand

Overview

The combination logical operator *nand* — not and —

- identifies rows whose sum is less than or equal to 1
- contains the sets of *xor* and *nor*

De Morgan identifies two logical expression for *nand*:

- not (P and Q)
- not P or not Q

logic				values						
P	Q	P and Q	P nand Q	P	Q	not(P and Q)	sum (P,Q)	not P	not Q	not P or not Q
T	T	T	F	1	1	0	2	0	0	0
T	F	F	T	1	0	1	1	0	1	1
F	T	F	T	0	1	1	1	1	0	1
F	F	F	T	0	0	1	0	1	1	1

Truth Tables

Overview

This section contains these programs:

Data Step	9
Merging Data Sets	10
Test Data	11

The programs shown below provide code examples which illustrate each permutation and combination of logical operators.

This table provides an overview of the sets.

truth values L R	idea	operators				single rows	expressions		
		single row	combinations of rows				combination of rows		
T T	both					L and R			L or R
T F	either and only.L		nand	xor	or	L and not R	not(L and R)	bxor(L,R)	L or R
F T	either only.R		nand	xor	or	not L and R	not(L and R)	bxor(L,R)	L or R
F F	neither nor		nand			not L and not R	not(L and R)		
nor and nand may also be expressed as:						not (L or R)	not L or not R		

Data Step

Program

The program `truth-table.sas` shows logical expressions used in a data step.

```
1 %Let List = 1,0; * boolean;
2
3 DATA Work.TruthTable;
4 length default = 4; * all _numeric_ are integers;
5
6 label P      = 'P: left'
7       Q      = 'Q: right'
8       and     = 'P and Q'
9       nand1   = 'nand.1: not(P and Q)'
10      nand2   = 'nand.2: not P or not Q'
11      only_P  = 'only.P: P and not Q'
12      only_Q  = 'only.Q: not P and Q'
13      xor     = 'P xor Q'
14      or      = 'P or Q'
15      nor1    = 'nor.1: not(P or Q)'
16      nor2    = 'nor.2: not P and not Q'
17      notP    = 'not P'
18      notQ    = 'not Q'
19      sumPQ   = 'sum(P, Q)'
20      sumPnQ  = 'sum(P, not Q)'
21      sumnPQ  = 'sum(not P, Q)'
22      sumnPnQ = 'sum(not P, not Q)';
23
24 do P      = &List.;
25   do Q    = &List.;
26     and     = P and Q;
27     nand1   = not (P and Q);
28     nand2   = not P or not Q;
29     only_P  = P and not Q;
30     only_Q  = not P and Q;
31     xor     = bxor(P, Q);
32     or      = P or Q;
33     nor1    = not (P or Q);
34     nor2    = not P and not Q;
35     notP    = not P;
36     notQ    = not Q;
37     sumPQ   = sum ( P, Q);
38     sumPnQ  = sum ( P, not Q);
39     sumnPQ  = sum (not P, Q);
40     sumnPnQ = sum (not P, not Q);
41     output;
42   end;
43 end;
44 stop;
45 run;
46
47 PROC Print data = &SysLast. label noobs;
48 title3 &SysLast.;
49 run;
```

Listing

P: left	Q: right	P and Q	nand.1: not(P and Q)	nand.2: not P or not Q	only.P: P and not Q	only.Q: not P and Q	P xor Q	P or Q	nor.1: not(P or Q)	nor.2: not P and not Q	not P	not Q	sum (P, Q)	sum (P, not Q)	sum (not P, Q)	sum (not P, not Q)
1	1	1	0	0	0	0	0	1	0	0	0	0	2	1	1	0
1	0	0	1	1	1	0	1	1	0	0	0	1	1	2	0	1
0	1	0	1	1	0	1	1	1	0	0	1	0	1	0	2	1
0	0	0	1	1	0	0	0	0	1	1	1	1	0	1	1	2

Merging Data Sets

Program

The program `truth-table-merges.sas` shows logical expressions used in a data step merge.

Note that the data set `Integers` is used to show the result of the logical operator *nor*.

```
----- truth-table-merges.sas -----
1  * see make-demo-data.sas;
2  %let In_Lib = Library;
3  *let In_Lib = Work;
4
5  %let Left = Evens;
6  *let Left = Odds;
7  %let Right = Fibonacci;
8
9  Title3 "Truth Table merging &Left. and &Right.";
10
11 DATA Work.Truth_Table_merges;
12 attrib N          length = 4
13         HaveLeft   length = 4
14         HaveRight   length = 4
15         &Left.      length = 4 label = "left: &Left."
16         &Right.     length = 4 label = "right: &Right." ;
17
18 length default = 4; * all numeric are integers;
19
20 label and          = 'L and R'
21       nand1        = 'nand.1: not(L and R)'
22       nand2        = 'nand.2: not L or not R'
23       only_L       = 'only.L: L and not R'
24       only_R       = 'only.R: not L and R'
25       xor          = 'L xor R'
26       or           = 'L or R'
27       nor1         = 'nor.1: not(L or R)'
28       nor2         = 'nor.2: not L and not R' ;
29
30 array Logicals &Left. &Right. and nand1 nand2
31              only_L only_R xor or nor1 nor2;
32 drop I;
33 do until(EndOfFile);
34   merge &In_Lib..Integers
35         &In_Lib..&Left.  (in = HaveLeft )
36         &In_Lib..&Right. (in = HaveRight)
37         end = EndOfFile;
38   by N;
39   do I = 1 to dim(Logicals);
40     Logicals(I) = .;
41   end;
42   if HaveLeft then &Left. = N;
43   if HaveRight then &Right. = N;
44
45   if HaveLeft and HaveRight then and = N;
46   if not (HaveLeft and HaveRight) then nand1 = N;
47   if not HaveLeft or not HaveRight then nand2 = N;
48
49   if HaveLeft and not HaveRight then only_L = N;
50   if not HaveLeft and HaveRight then only_R = N;
51   if bxor(HaveLeft, HaveRight) then xor = N;
52   if HaveLeft or HaveRight then or = N;
53
54   if not (HaveLeft or HaveRight) then nor1 = N;
55   if not HaveLeft and not HaveRight then nor2 = N;
56   output;
57 end;
58 stop;
59 run;
60
61 PROC Print data = &SysLast. label noobs;
62   title4 &SysLast.;
63 run;
```

Listing

N	left: Evens	right: Fibonacci	L and R	nand.1: not(L and R)	nand.2: not L or not R	only.L: L and not R	only.R: not L and R	L xor R	L or R	nor.1: not(L or R)	nor.2: not L and not R
1	.	1	.	1	1	.	1	1	1	.	.
2	2	2	2	.	.	.	.	2	2	.	.
3	.	3	.	3	3	.	3	3	3	.	.
4	4	.	.	4	4	4	.	4	4	.	.
5	.	5	.	5	5	.	5	5	5	.	.
6	6	.	.	6	6	6	.	6	6	.	.
7	.	.	.	7	7	.	.	.	.	7	7
8	8	8	8	.	.	.	.	8	8	.	.
9	.	.	.	9	9	.	.	.	.	9	9

Test Data

Examples

These data sets are used in truth-table-merges program.

```
1  %Let Out_Lib = Library;
2  *Let Out_Lib = Work;
3
4  DATA  &Out_Lib..Integers
5         &Out_Lib..Evens
6         &Out_Lib..Odds
7         &Out_Lib..Fibonacci;
8
9  length default = 4; * all _numeric_ are integers;
10
11 do N = 1 to 9;
12     output &Out_Lib..Integers;
13     if not mod(N,2) then output &Out_Lib..Evens;
14     else output &Out_Lib..Odds;
15     if N in (1,2,3,5,8) then output &Out_Lib..Fibonacci;
16     end;
17 stop;
18 run;
19
20 Proc Print data = &Out_Lib..Integers
21 Proc Print data = &Out_Lib..Evens;
22 Proc Print data = &Out_Lib..Odds;
23 Proc Print data = &Out_Lib..Fibonacci;
```

Conclusion

Logical ideas describe permutations of truth values. Combination logical operators include sets of the permutations. This deconstruction of operators explains logical precedence.

Author: Ronald Fehd
Centers for Disease Control MS-G23
4770 Buford Hwy NE
Atlanta GA 30341-3724
