And-Not-Or Set Matrix

by Sven Nilsen, 2020

In this paper I derive the complexity of Boolean Algebra of `n` objects, using an and-not-or set matrix.

Since `and(a, b) = and(b, a)` and `or(a, b) = or(b, a)`, one can put both inside a set matrix:

	00	**01**	**10**	**11**
00	11	01	10	11
01	00	10	11	11
10	00	00	01	11
11	00	01	10	00

and	not	or

The diagonal is filled with the inverse, since `and(a, a) = a` and `or(a, a) = a`.

This `m × m` set matrix is a complete representation of a Boolean Algebra of sets with size `m`.

Since `and[not] <=> or`, the whole matrix can be generated from a triangle using the diagonal:

$m \cdot (m - 1) / 2 + m$ Size of a strictly triangular matrix plus the diagonal

For example:

or(01, 10) = not(and(not(01), not(10)) = not(and(10, 01)) = not(00) = 11

One can use `nand` alone since `nand(a, a) = not(a)` and `not(nand(a, b)) = and(a, b)`.
However, this has the same complexity since `nand(a, a)` fills the diagonal.

For `n` objects, the number of sets are `$2^n$`, so this gives the complexity of the Boolean Algebra:

$$2^n \cdot (2^n - 1) / 2 + 2^n$$

n	**complexity**								
0	1	2	3	4	5	6	7	8	9
1	3	10	36	136	528	2080	8256	32896	131328

The `n = 0` algebra contains only the empty set, characterized by `not(a) = a`.
The `n = 1` algebra is isomorphic to the Boolean Algebra on bits.

When `n` goes to infinity, the ratio of `complexity(n+1) / complexity(n)` converges to 4.