

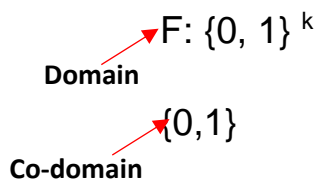
Proposition logic:

Every proposition has one of two possible truth values assigned to it (true- also written as 1 or false- also written as 0). It is assumed that propositions are independent of each other.

Truth or falsity of different proposition is unrelated and so counting of number of possibilities based on Product Rule

Product Rule

Each proposition has two possible truth values (options): Truth (also known as 1) or False (also known as 0). So if there are k propositional variables, then the number of assignments is 2^k . 2 is the number of possible truth values of the formula on any assignment. It follows that 2^{2^k} is the number of semantically distinct formulae on k propositional variables (or propositions).



For example for $k = 3$:

$\{0, 1\}^3 =$

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

	Proposition1 (P1)	Proposition2 (P2)	Function
Assignment 1	0	0	0/1
Assignment 2	0	1	0/1
Assignment 3	1	0	0/1
Assignment 4	1	1	0/1

$$2^{2^k} = 2 \times 2 \times 2 \times 2$$

Syntax for Propositional Logic (Boolean Logic):

$P = \{p_0, p_1, \dots, \dots\}$ are Atomic Propositions. They are called atomic as they cannot be simplified any further.

Structural Induction: Applied to operators. Proves for complicated structures by assuming statement for simple structures.

$\neg p \vee q$
 $p \Rightarrow q$

Not Same \Rightarrow Not someone not acquainted with semantics.

$p \Rightarrow q$
 $p \Rightarrow q$

Not Same \Rightarrow Computer

Tautology

$$P1 \vee (\neg P1) = \text{True}$$

P1	$\neg P1$	R ($P1 \vee (\neg P1)$)
0	1	1
1	0	1

Contradiction

$$P1 \wedge (\neg P1) = \text{False}$$

P1	$\neg P1$	R ($P1 \wedge (\neg P1)$)
0	1	0
1	0	0

$$\text{GCD}(a,b) = 1$$

$$b \neq 0$$

$$\Rightarrow \sqrt{2} = a / b$$

$$\Rightarrow 2 = a^2 / b^2$$

$$\Rightarrow a^2 = 2b^2$$

$$\Rightarrow (2t)^2 = 2b^2$$

$$\Rightarrow 4t^2 = 2b^2$$

$$\Rightarrow 2t^2 = b^2$$

Syntax:

what are the legal formulae Propositional / Boolean Logic?

Rules:

(A) Elementary Case:

$P_0, P_1, P_2, P_3, \dots, \dots, \dots, \dots$

(B) Complex Formulae:

- (i) If Ψ is a formula then so is: $\neg(\Psi)$
- (ii) If Ψ_1 is a formula and Ψ_2 is a formula then so are :
 - a. $\Psi_1 \vee \Psi_2$
 - b. $\Psi_1 \wedge \Psi_2$
 - c. $\Psi_1 \Rightarrow \Psi_2$
 - d. $\Psi_1 \Leftrightarrow \Psi_2$
 - e. $\Psi_1 \text{ XOR } \Psi_2$

Example:

If $P_5 \Rightarrow ((\neg p_1 \vee \neg p_2) \wedge (\neg p_3 \Rightarrow p_4))$ is formula

$\neg(p_5) \Rightarrow \neg((\neg p_1 \vee \neg p_2) \wedge (\neg p_3 \Rightarrow p_4))$ is also a formula

P1	P2	$P1 \Leftrightarrow P2$	$P1 \text{ XOR } P2$
0	0	1	0
0	1	0	1

1	0	0	1
1	1	1	0

We can replace $(P1 \leq P2)$ by $\neg(P1 \leq P2)$

P1	P2	$\neg(P1 \leq P2)$	P1 XOR P2
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

So that xor **P1 XOR P2** is same as $\neg(P1 \leq P2)$

Using d morgan's law we can simplify the formula.

Example:

P1	P2	$P1 \wedge P2$	$\neg(\neg P1 \vee \neg P2)$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

We can replace \leq formula with $=$ and \wedge

We can replace $=$ and \wedge formula with \neg and \vee

So basically for everything either \neg and \vee or \neg and \wedge are sufficient which is called minimal set of collectives