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).

F: $\{0, 1\}^k$

Domain

$\{0,1\}$

Co-domain

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

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

<u>Syntax for Propositional Logic (Boolean Logic):</u>

P = {p0, p1, ..., ..., ... } are Atomic Propositions. They are called atomic as they cannot be simplified any further.

<u>Structural Induction:</u> Applied to operators. Proves for complicated structures by assuming statement for simple structures.

¬p V q

Not Same => Not someone not acquainted with sementics.

p => q

p => q

Not Same => Computer

p => q

Tautology

P1 V (¬P1) = True

P1	¬P1	R (P1 V (¬P1))
0	1	1
1	0	1

Contradiction

P1∧(¬P1) = False

P1	¬P1	R (P1∧(¬P1))
0	1	0
1	0	0

GCD(a,b) = 1
$b \neq 0$

- ⇨ $\sqrt{2} = a / b$
- ⇨ $2 = a^2 / b^2$
- ⇨ $a^2 = 2b^2$
- ⇨ $(2t)^2 = 2b^2$
- ⇨ $4t^2 = 2b^2$
- ⇨ $2t^2 = b^2$

Syntax:
what are the legal formulae Propositional / Boolean Logic?

Rules:

(A) Elementary Case:

P0, P1, P2, P3, …, …., …. , ….

(B) Complex Formulae:

(i) If Ψ is a formula then so is: ¬ (Ψ)

(ii) If Ψ1 is a formula and Ψ2 is a formula then so are :

a. Ψ1 V Ψ2

b. Ψ1 ∧ Ψ2

c. Ψ1 => Ψ2

d. Ψ1< = > Ψ2

e. Ψ1 XOR Ψ2

Example:

If P5 => ((¬p1 V ¬p2) ∧ (¬p3=>p4)) is formula

¬(p5)=> ¬((¬p1 V ¬p2) ∧ (¬ p3 => p4)) is also a formula

P1	**P2**	**P1 < = > P2**	**P1 XOR P2**
0	0	1	0
0	1	0	1

1	0	0	1
1	1	1	0

We can replace (**P1 < = > P2**) by **¬(P1 < = > P2**)

P1	**P2**	**¬(P1 < = > 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 **¬(P1 < = > P2**)

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

Example:

P1	**P2**	**P1 ∧ P2**	**¬ (¬P1 V ¬P2)**
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

We can replace **< = > formula with = > and ∧**

We can replace **= > and ∧ formula with ¬ and v**

So basically for everything either ¬ and v or ¬ and ∧ are sufficient which is called minimal set of collectives