Boolean algebra is a formalism for manipulating truth values (logical operations).

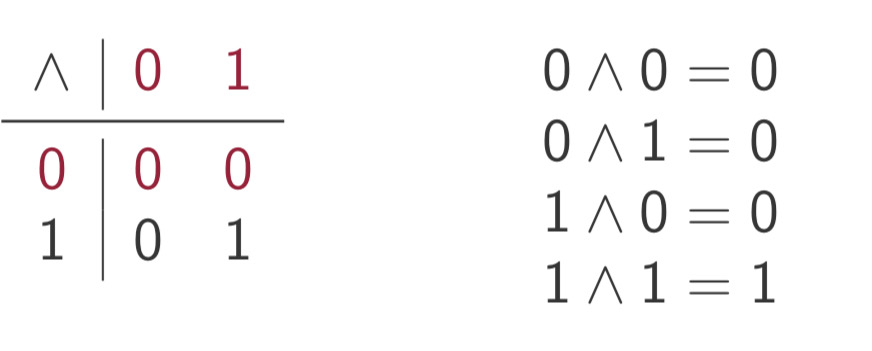
Only two truth values:

* true (1)
* False (0)

Three Boolean operators, conjunctions:

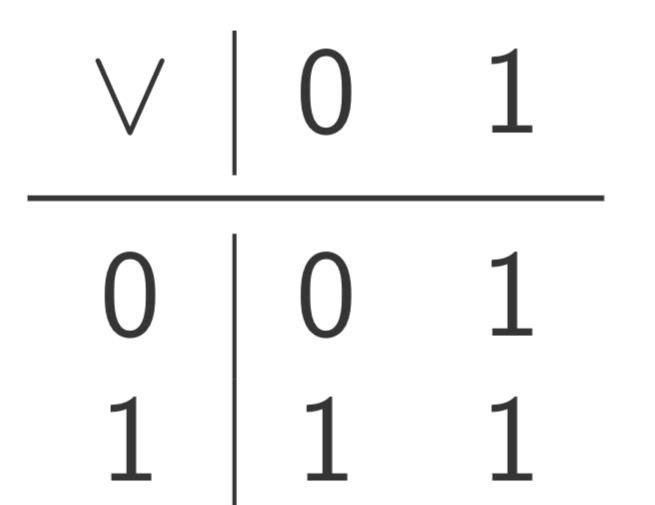
^ AND

* A conjunction of (two) truth values is true if and only if (iff) both values are true



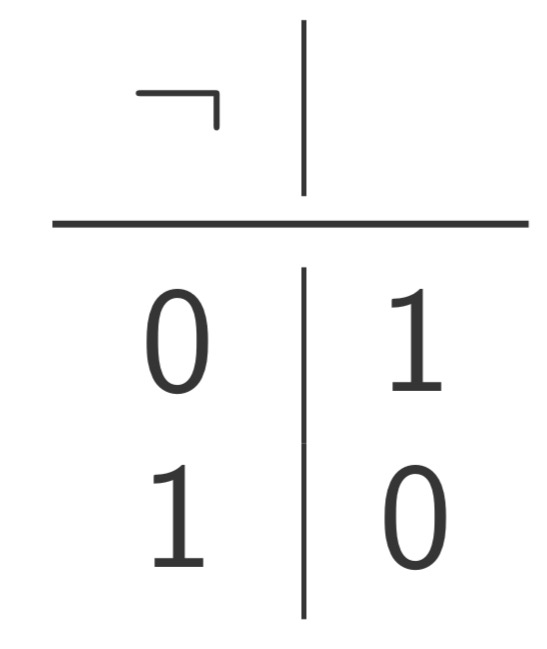
V OR

A disjunction of truth values is true iff at least one value is true



NOT ¬

The negation of a truth value is true iff the value is false

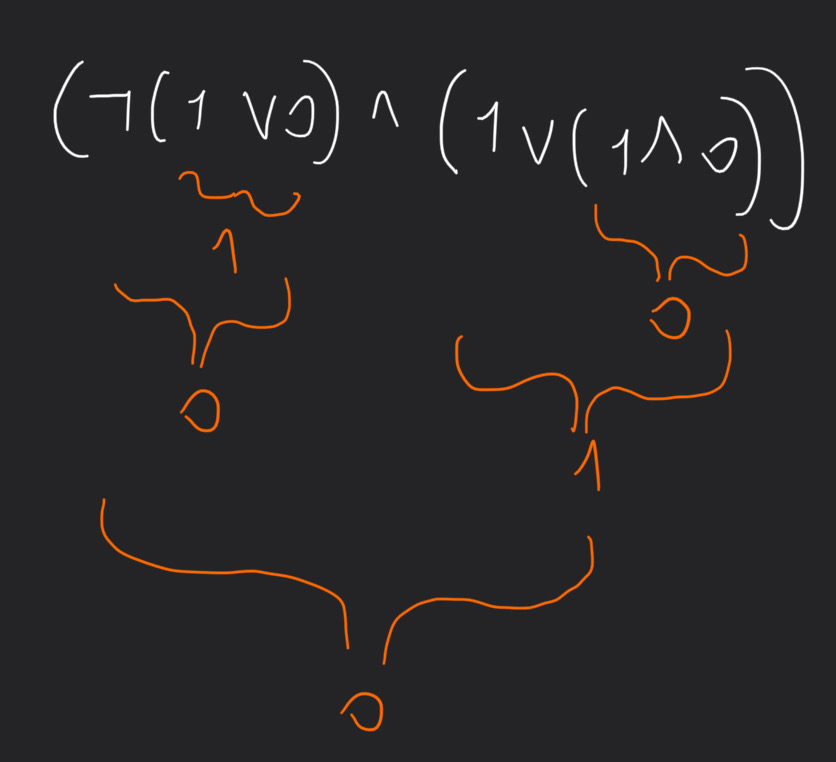


**Exercise:**

Build a table for the XOR operator which is true iff exactly one value is true

**Complex expressions:**

The operations allows us to compete the value of a complex expression



Kind of boring if we only have constants (0/1)

What if we use variables instead?

**PROPOSITIONAL LOGIC**

Propositional logic is a formalism for combining propositions (sentences, statements) to make conclusions about their truth value

Atomic propositions state one specific fact or property

* *mammas are vertebrate*
* *Vertebrates are vegetarians*
* *Cars have umbrellas*
* *X* (there is no implicit meaning)

(Sentences can be true of fase, it can be also be words without meaning)

Formulas combine propositions to make complex statements

Remember that a logic is just a language

We must specify

* which expressions are allowed → **syntax**
* What do they mean -> **semantics**

Take an arbitrary set of propositional variables (atomic propositions)

The class of propositional formulas is defined by

* Every propositional variable is a formula
* If A and B are formulas, they so are:
  + ¬A
  + A^B
  + AvB

Operator precedence: ¬ , ^, v

**Example of formulas:**

W

X

Y

Z

¬W

¬ W ^ Z

X v Y

¬ (X v Y)

¬ (X v Y) ^ ( ¬ (W ^ Z)

…

Anything that is “well formed” is a formula in the intuitive sense

X ^ v Y is NOT a formula

**Intuition of the semantics**

Recall the atomic propositions may be true or false

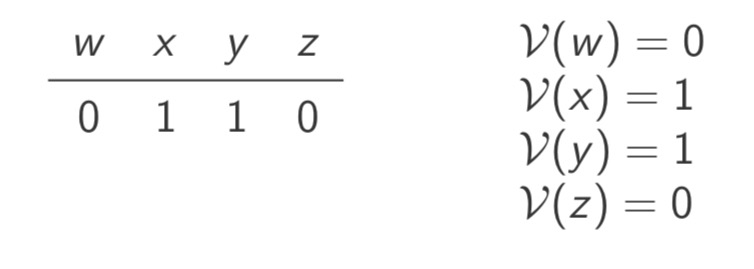
What about formulas?

↓

Depends on the truth status of its components (subformulas)

A valuation is an assignment of a truth value to each variable

For example the valuation V



The assignment of a valuation is extended to formulas

V ( ¬ a) = ¬ V(a)

! be careful with the meanings!

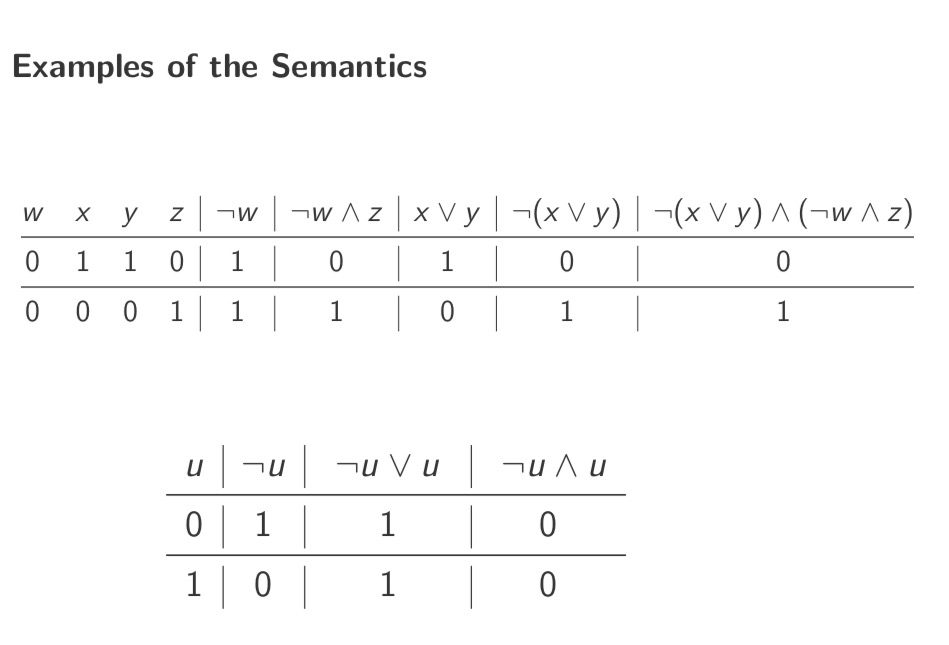
V( ¬ a) = 1 iff V (a) = 0

V (a ^ b) = V(a) ^ V(b)

V(a ^ b) = 1 iff V(a) = 1 and V(b) = 1

V(A v B) = V(A) v V(B)

V(A v B) = 1 iff V(a) = 1 or V(b) = 1



**Three kinds of Formulas**

There are three types of propositional formulas:

* tautologies are always evaluated to 1, regardless of the valuation
* Contradictions are always evaluated to 0, regardless of the valuation
* Non-tautological satisfiable formulas are all others

A formula takes a valuation and returns a truth value

A formula with n variables is a function. A mapping from n truth values to 1

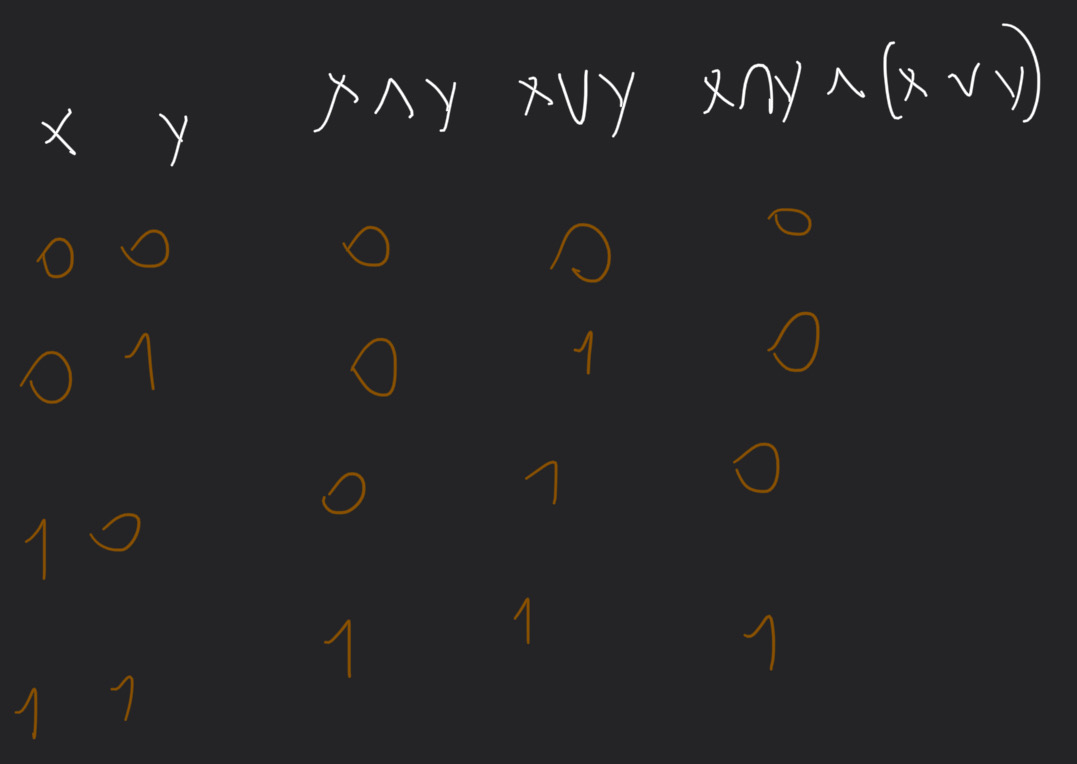
Example:



**Equivalence**

What characterize a formula is its behavior as a function

Two formulas are equivalents iff they have the same truth table



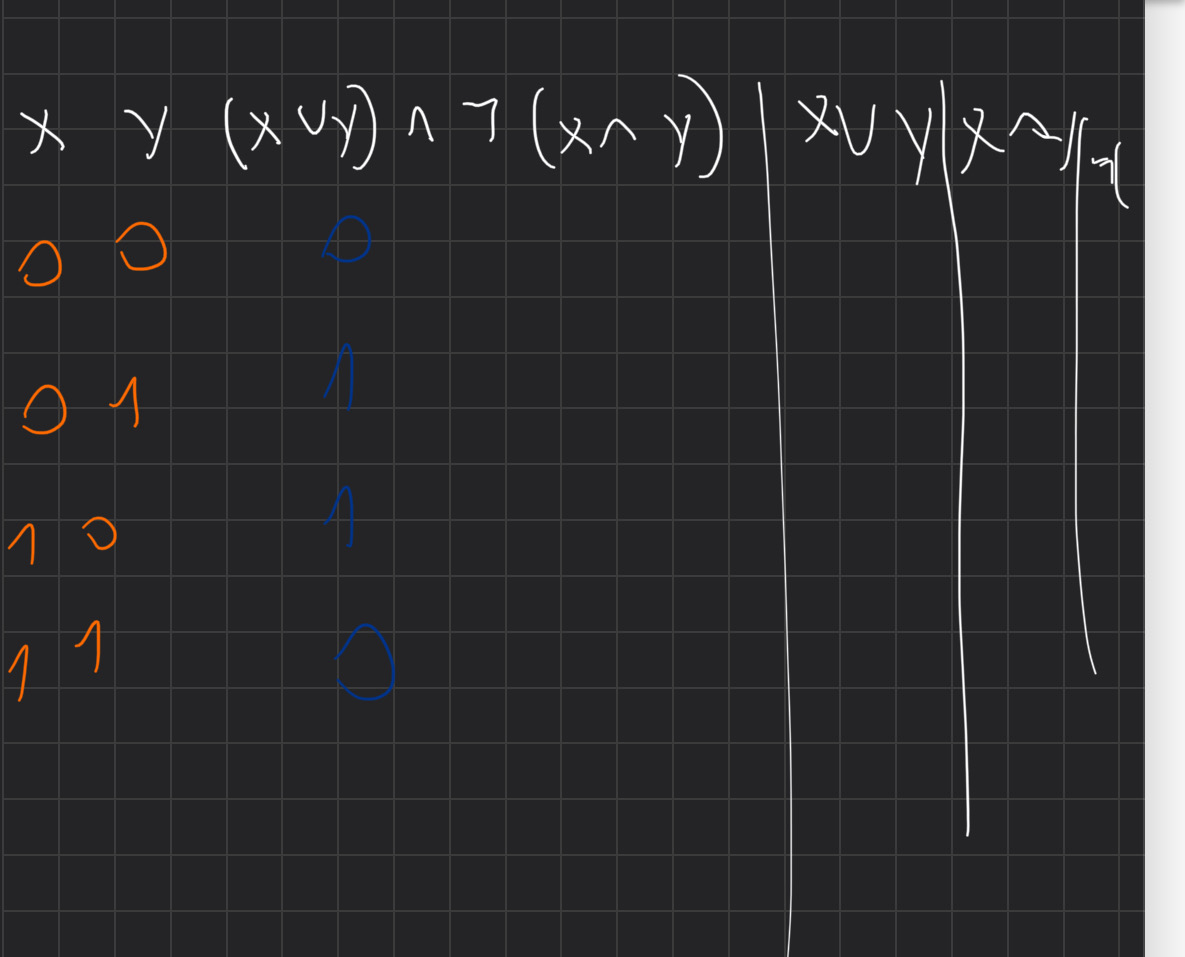
**Others operators**

We can imagine other binary operators (how many?)

* NAND (non and), not both true
* Implications x→ y if x than y. (*Whenever x is true then the variable y will be true. We are not saying anything about ¬ x*)
* …

All the operators can be expressed via ¬, ^, v

X XOR y = (X v Y) ^ ¬ (X ^ Y)



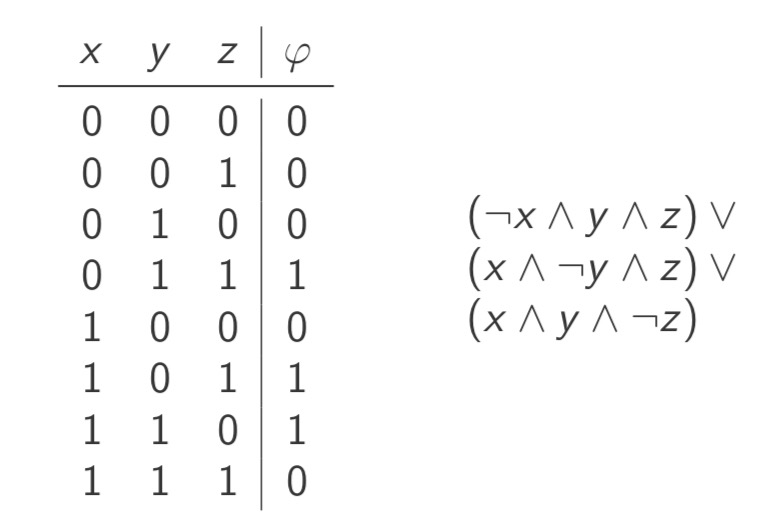
FINIRE ↑

X → Y = ¬ X v Y

Exercise: find an equivalent representation for NAND

**Completeness of Operators II**

The operators ¬ ^ v suffice to express any formula regardless of the number of variables



**De Morgan**

¬ (X ^ Y) = ¬ X v ¬ Y

¬ (X v Y) = ¬ X ^ ¬ Y

**Distributivity**

X ^ ( Y v Z) = (X ^ Y) v (X ^ Y)

X v (Y ^ Z) = (X v Y) ^ (X v Y)

5\*(6+1) = 5\*6 + 5\*1

**Involution**

¬ ¬ X = X