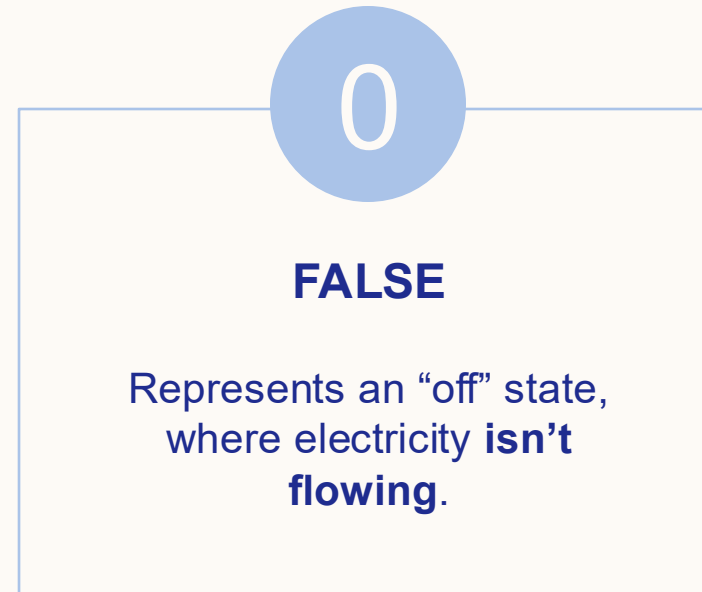
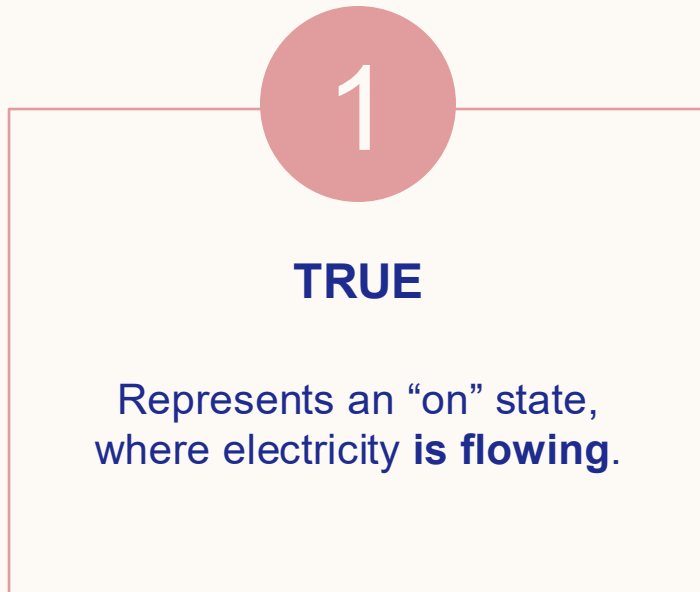


BOOLEAN ALGEBRA

Kira Clements, University of Bristol

BINARY REPRESENTATION

Every digital device uses **binary** data representation - “*two states*”



To look at how these signals can be manipulated, we use **Boolean algebra**...

PROPOSITIONS

To explore boolean algebra we first need to understand **propositions**: statements that are either true or false. A good way to test for a proposition is by using the phrase *“it is true that...”* before the statement.

For example, for a triangle with defined sides a, b, and c, we have the following **propositions**:

side a is the same length as side b \longrightarrow x

side b is the same length as side c \longrightarrow y

side c is the same length as side a \longrightarrow z

These can be replaced by **propositional variables**, which represent the truth value of each proposition. We can give these variables any name we like but use letters by convention and for efficiency.

what type of triangle is it?

side a + side b

Non-propositions can be turned
into propositions

it is an equilateral triangle

side a + side b = side c

LOGICAL OPERATORS

 $\neg A$ $A \wedge B$ $A \vee B$

Formal	Negation / Complement	Conjunction	Disjunction
Natural	not A	A and B	A or B
Alternative	\bar{A}	$A \bullet B$	$A + B$
C	! A	A && B	A B

These are generally known as the basic operators and are a particular subset of all operators that can be used to create any logical expression.

This logic can be expressed through symbols and words, but there are also representations that more explicitly show how they deal with different truth values as inputs.

TRUTH TABLES

Every combination of possible input values and their corresponding output values for a given formula.

A	B	$\neg A$
0	0	1
0	1	1
1	0	0
1	1	0

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

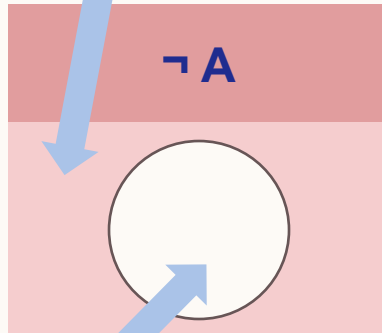
VENN DIAGRAMS

Output

 True/1

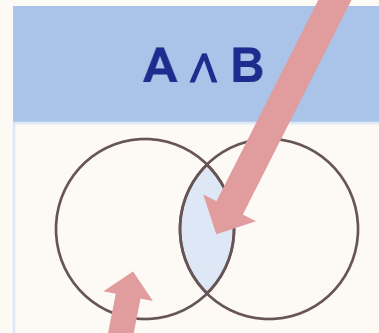
 False/0

Every instance
where A is false



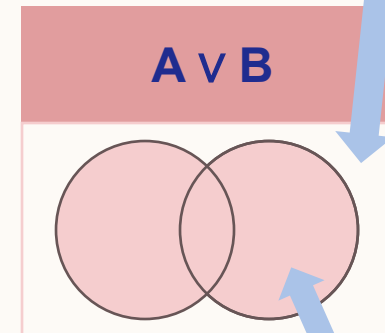
Every instance
where A is true

Every instance where
both A and B are true



Every instance where
A is true, but B is false

Every instance where
both A and B are false



Every instance where
B is true, but A is false

COMBINING OPERATORS

Going back to our previous propositions, we can combine them with logical operators to create **compound propositions**. Brackets are used to clarify precedence, as necessary.

x = side a is the same length as side b

y = side b is the same length as side c

z = side c is the same length as side a

What logical formula will give us the truth value of whether the triangle is an equilateral triangle (all sides are equal)?

$$x \wedge y \wedge z$$

What logical formula will give us the truth value of whether the triangle is a scalene triangle (no sides are equal)?

$$\neg x \wedge \neg y \wedge \neg z$$

What logical formula will give us the truth value of whether the triangle is an isosceles triangle (exactly two sides are equal)?

$$(x \vee y \vee z) \wedge \neg(x \wedge y \wedge z)$$

MORE LOGICAL OPERATORS

$$A \oplus B$$

$$A \rightarrow B$$

$$A \leftrightarrow B$$

Formal

Disjunction (exclusive)

Implication

Equivalence

Natural

exclusively A or B (xor)

A implies B

A is equivalent to B

These can be expressed using the basic operators!

$$(\neg A \vee \neg B) \wedge (A \vee B)$$

$$\neg A \vee B$$

$$(\neg A \vee B) \wedge (\neg B \vee A)$$

$$A \rightarrow B \wedge B \rightarrow A$$


Using the natural form, variables can be replaced with propositions to create a sentence in natural language that can help us understand the intuition behind these operators.

MORE TRUTH TABLES

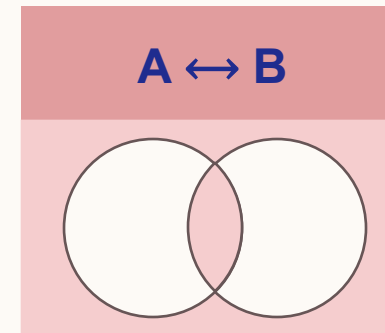
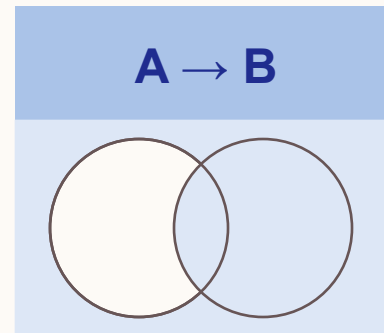
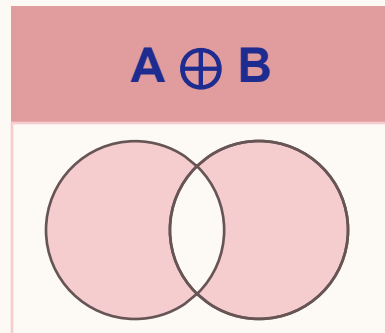
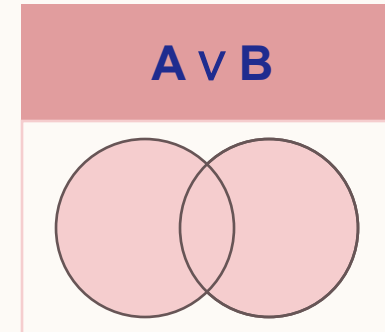
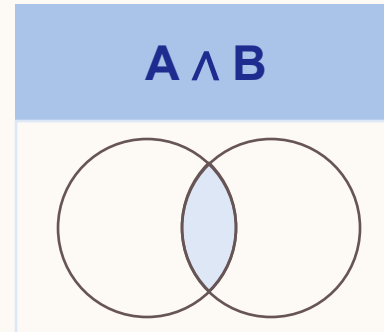
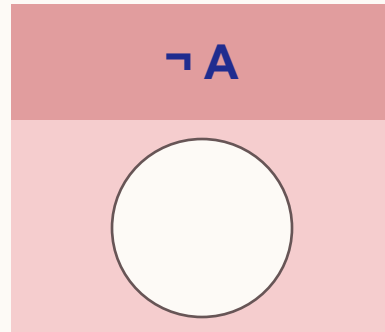
A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \oplus B$	$A \rightarrow B$	$A \leftrightarrow B$
0	0	1	0	0	0	1	1
0	1	1	0	1	1	1	0
1	0	0	0	1	1	0	0
1	1	0	1	1	0	1	1

MORE VENN DIAGRAMS

Output

  True/1

  False/0



COMPLETING THE TABLE

To complete the truth table for a more complicated Boolean expression, just replace each variable with its input value and simplify.

A	B	C	$((A \wedge B) \vee \neg C)$
0	0	0	$((0 \wedge 0) \vee \neg 0) \equiv (0 \vee 1) \equiv 1$
0	0	1	$((0 \wedge 0) \vee \neg 1) \equiv (0 \vee 0) \equiv 0$
0	1	0	$((0 \wedge 1) \vee \neg 0) \equiv (0 \vee 1) \equiv 1$
0	1	1	$((0 \wedge 1) \vee \neg 1) \equiv (0 \vee 0) \equiv 0$
1	0	0	$((1 \wedge 0) \vee \neg 0) \equiv (0 \vee 1) \equiv 1$
1	0	1	$((1 \wedge 0) \vee \neg 1) \equiv (0 \vee 0) \equiv 0$
1	1	0	$((1 \wedge 1) \vee \neg 0) \equiv (1 \vee 1) \equiv 1$
1	1	1	$((1 \wedge 1) \vee \neg 1) \equiv (1 \vee 0) \equiv 1$

TRUTH TABLES

Every logical formula has one truth table that matches it, yet every truth table has multiple logical formulas that match it. These are known as **logically equivalent** formula.

$A \vee B$

A	B	?
0	0	0
0	1	1
1	0	1
1	1	1

$A \vee B$

$\neg(\neg A \wedge \neg B)$

$B \vee A$

$\neg(\neg A \wedge \neg B) \wedge (A \vee \neg A)$

Truth tables can be used to validate logical equivalence, as well as help describe how an unknown logical formula should behave given all combinations of inputs.

Given a truth table of expected outputs, how can we generate a suitable logical formula?

WHY IS THIS RELEVANT?

Boolean logic is used throughout programming languages. Learning how to construct and manipulate Boolean expressions can help you translate real problems into code.

```
int a = 5, b = 10, c = 20;
```

```
if(a > b){  
    if(b > c){  
        a = b + c;  
    }  
}
```

```
if(c > a){  
    a = b + c;  
}
```

Simplified using
Boolean algebra

```
int a = 5, b = 10, c = 20;
```

```
if((a > b && b > c) || c > a){  
    a = b + c;  
}
```