Thinking
Mathematically

Logic and Reasoning

Logic is the study of what is true and false. It analyzes:

- **logical statements**, which can be true or false
  e.g. "every dog is an animal" , "every animal is a dog"

- **logical arguments**, which can be used to show that a statement is true or false
  e.g. "Collie is a dog, so she is an animal"

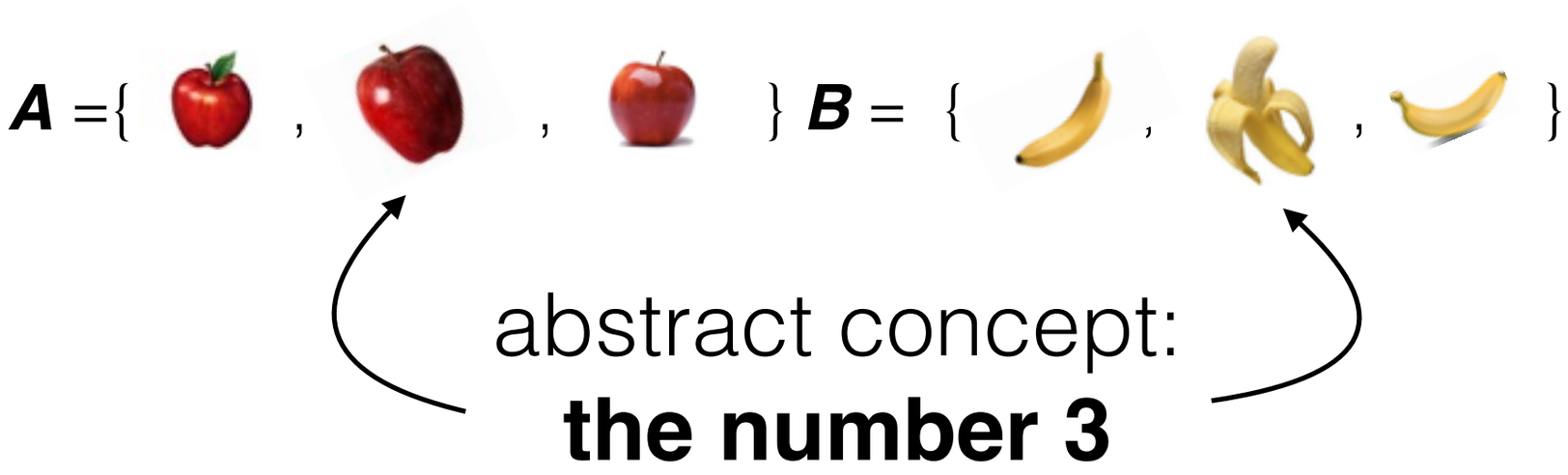
In this topic, we will talk about logical statements.
This will give us the tools we need discuss logical arguments.

Logic is Math

Like other fields of mathematics, logic deals with **abstractions.**

It focuses only on certain aspects of statements and arguments, which can be described and manipulated in a systematic way.

The system allows us to reason about problems, and reliably reach conclusions by focusing only on the relevant aspects. So it can be quite powerful and useful.



True and False

In the real world, things are not always absolutely true or absolutely false. We make many hidden assumptions.

e.g. "if you win the race, you will get a gold medal"
assumption: no one will steal the gold medal before the ceremony

e.g. "the sun will rise tomorrow"
assumption: the Earth will keep rotating tomorrow

But in math, we are dealing with idealized concepts, so we can be sure that something is true:

e.g. if a and b are both even numbers, then a + b is an even number

Boolean Algebra

In previous topics, we explored numbers and algebra.

We had variables that came from a set, such as $a, b \in \mathbf{N}$
and we defined algebraic operations like: $a + b$, $a \bullet b$, $-a$

To study logic, we will now define a set with just two values:
{ False, True }. Boolean variables can take on one of these values.
So we can represent logical statements using *boolean variables*.

And we will have three algebraic operations on these values:

a AND *b* *a* OR *b* NOT *a*

Boolean AND

Just like with addition and multiplication, you can think of *AND* as a function that takes two inputs and gives one output.

We can construct a table listing all possible inputs to the function and its output, called a *truth table:*

a	*b*	*a AND b*
F	F	F
F	T	F
T	F	F
T	T	T

Obviously *a* AND *b* = True **if, and only if,** both *a* = True and *b* = True.

Boolean OR

Similarly, *OR* is a function that takes two inputs.

Here is its truth table:

a	*b*	*a OR b*
F	F	F
F	T	T
T	F	T
T	T	T

Obviously *a* OR *b* = True **whenever** either *a* = True or *b* = True.

Boolean NOT

Like the operation of negating a number, *NOT* is a function that takes one input. So its truth table has only two columns:

a	NOT *a*
F	T
T	F

Obviously NOT *a* = True **whenever** *a* = False, and vice versa.

Logical Equivalence

We've been using the term "**if and only if**" in the previous slides. It actually means that the expression on the left is true whenever the one on the right is true, and false whenever the one on the right is false. In other words, the two expressions have the exact same value for *all* values variables *a* and *b*.

It's like saying two functions are equal for all inputs: $f(a, b) = g(a, b)$

a	NOT *a*	NOT (NOT *a*)
T	F	T
F	T	F

The above table shows that *a* is logically equivalent to NOT (NOT *a*).

The mathematical notation for this is $a \iff \text{NOT (NOT } a))$

NOT AND

Let's look at the truth table of *a* AND *b* again.
When is *a* AND *b* = False?

a	*b*	*a* AND *b*
F	F	F
F	T	F
T	F	F
T	T	T

We see *a* AND *b* = False **if and only if** *a* = False OR *b* = False.

De Morgan's Law: AND

This allows us to figure out how to negate the statement *a* AND *b*

We see *a* AND *b* = False **whenever** *a* = False OR *b* = False.

So NOT (*a* AND b) $\iff$ (NOT *a*) OR (NOT *b*)

To negate the statement *a* AND *b*, you have to negate *a*, negate *b*, and turn the AND into an OR. Here is the truth table to illustrate:

a	*b*	*a* AND *b*	NOT *a*	NOT *b*	(NOT *a*) OR (NOT *b*)
F	F	F	T	T	T
F	T	F	T	F	T
T	F	F	F	T	T
T	T	T	F	F	F

NOT OR

Similarly, take a look at the truth table of *a* OR *b*.
When is *a* OR *b* = False?

a	*b*	*a* OR *b*
F	F	F
F	T	T
T	F	T
T	T	T

We see *a* OR *b* = False **if and only if** *a* = False AND *b* = False.

De Morgan's Law: OR

This allows us to figure out how to negate the statement *a* OR *b*

We see *a* OR *b* = False **whenever** *a* = False AND *b* = False.

So NOT (*a* OR b) $\iff$ (NOT *a*) AND (NOT *b*)

To negate the statement *a* OR *b*, you have to negate *a*, negate *b*, and turn the OR into an AND. Here is the truth table to illustrate:

a	*b*	*a* OR *b*	NOT *a*	NOT *b*	(NOT *a*) AND (NOT *b*)
F	F	F	T	T	T
F	T	T	T	F	F
T	F	T	F	T	F
T	T	T	F	F	F

Digital Logic: AND

In computers, instead of the set { False, True } we use the set { 0, 1 }.

It turns out that logical operations are very similar to operations with the natural numbers 0 and 1. *AND* behaves like multiplication:

a	b	$a \cdot b$
0	0	0
0	1	0
1	0	0
1	1	1

Digital Logic: OR

However, there is a difference between OR and addition:

a	*b*	*a* + *b*
0	0	0
0	1	1
1	0	1
1	1	2

The last row has a 2 instead of a 1.

That's why, in many computer programming languages, any value other than zero is usually considered true. These languages make the OR behave like addition.

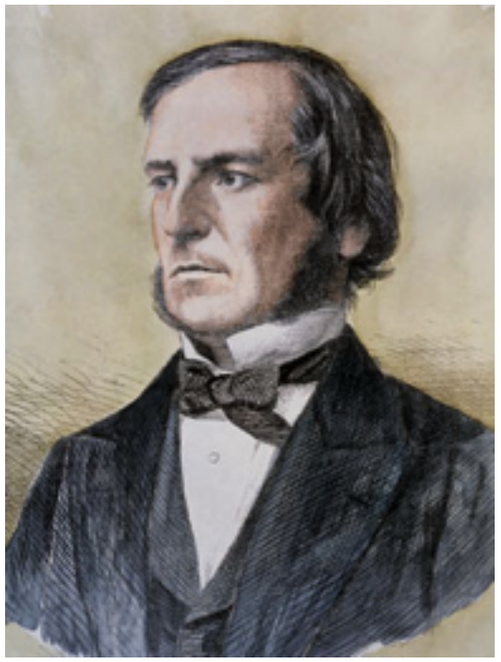
Digital Logic: NOT

In those computer languages, NOT 0 becomes 1, and NOT 1 becomes 0, but so does NOT 3, NOT 4, NOT 5.

a	NOT *a*
0	1
1	0
2	0
...	...

Computers today store information in "bits", which are the smallest part of computer memory, capable of storing either a 0 or 1. CPUs work by having electric current be either going through a circuit, or not. That is why sound, video and other things are stored "digitally" in computer storage, whereas before, on cassette tapes, it was analog.

Augustus DeMorgan
1806-1871



George Boole
1815-1864

Charles Babbage
1791-1871

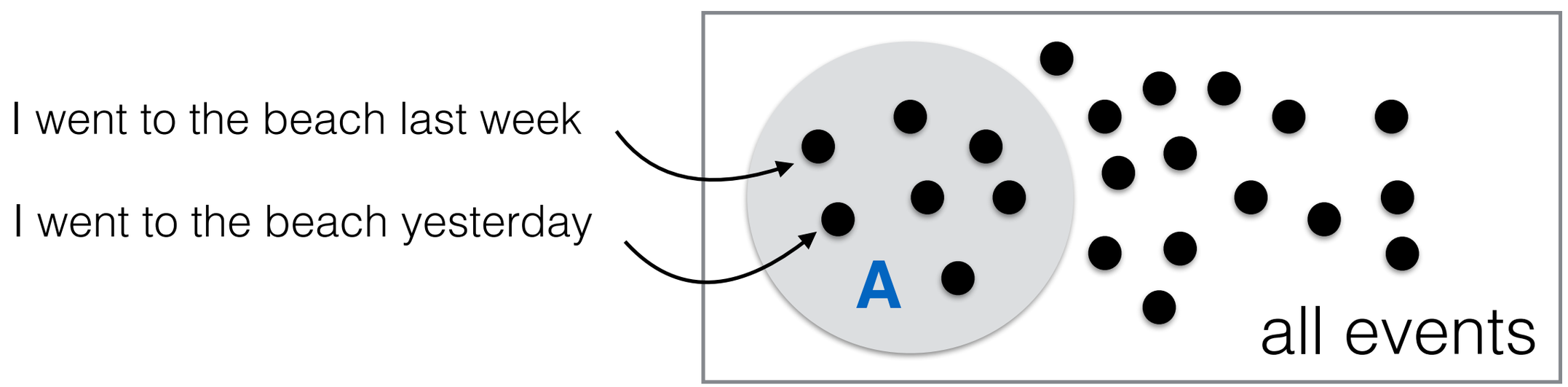
Alan Turing
1912-1954

John Von Neumann
1903-1957

Sets and Logic

Let's explore another way of illustrating the truth of statements. Consider the set *X* of all the pixels on your screen. You can form sets *A, B, etc.* out of these pixels. Given one of these sets, a particular pixel is either in the set or not. In mathematical notation, for any $x \in X$, either $x \in A$ or $x \notin A$.

You can think of a set *A* as all the set of all **events** where *x* is true. For example *A* is the set of all events where "I went to the beach."

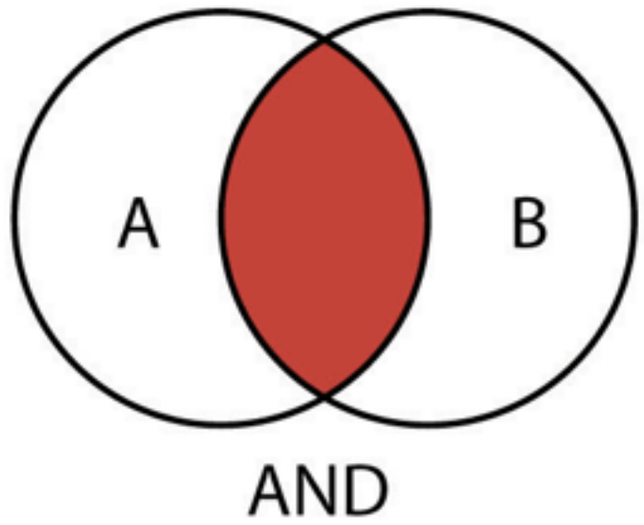


Algebra on Sets: AND

It turns out that operations on sets have the same properties as corresponding operations on logical statements!

Given two sets *A* and *B*, we can define their **intersection**

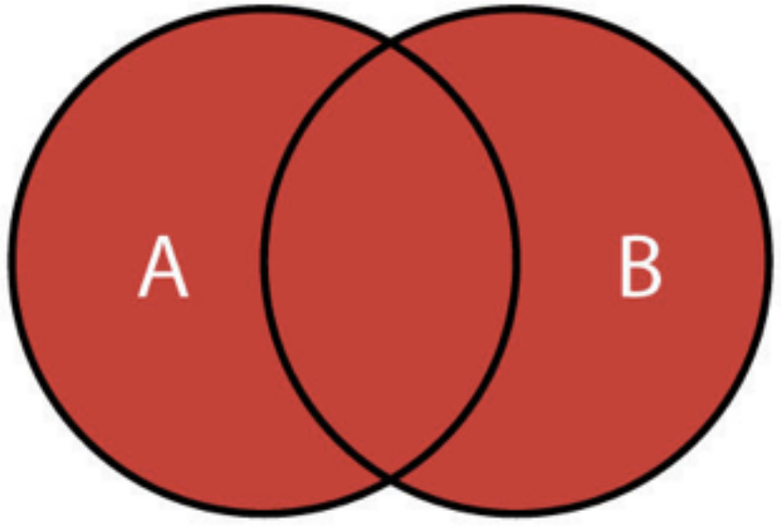
$$A \cap B = \{ x : x \in A \ \underline{\textbf{and}} \ x \in B \}$$



Algebra on Sets: OR

Similarly, given two sets *A* and *B*, we can define their **union**

$$\boldsymbol{A} \cup \boldsymbol{B} = \{ x : x \in A \ \underline{\textbf{or}} \ x \in B \}$$

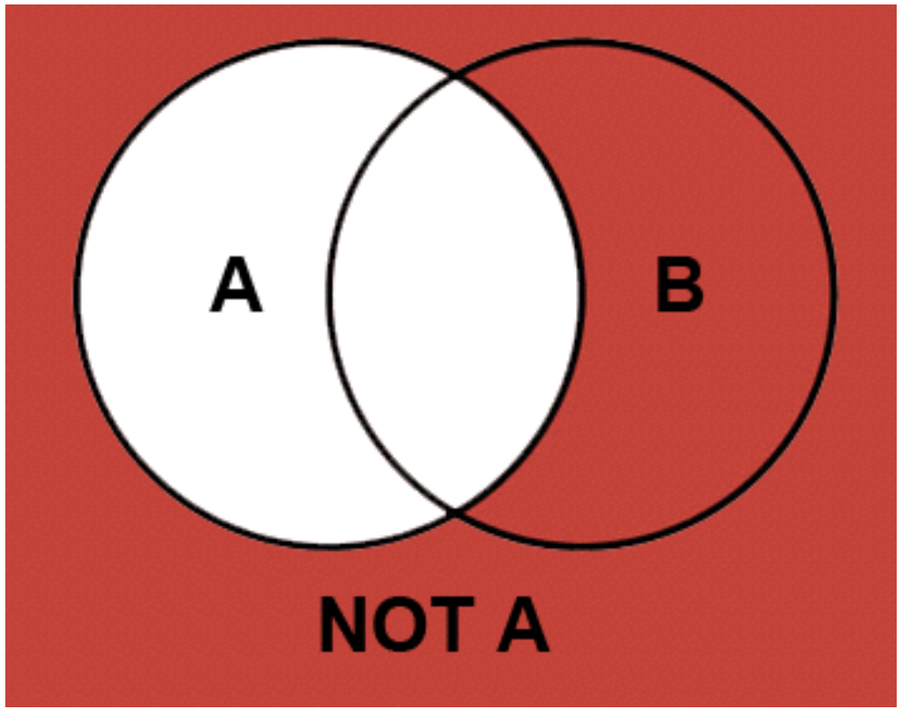


OR

Algebra on Sets: NOT

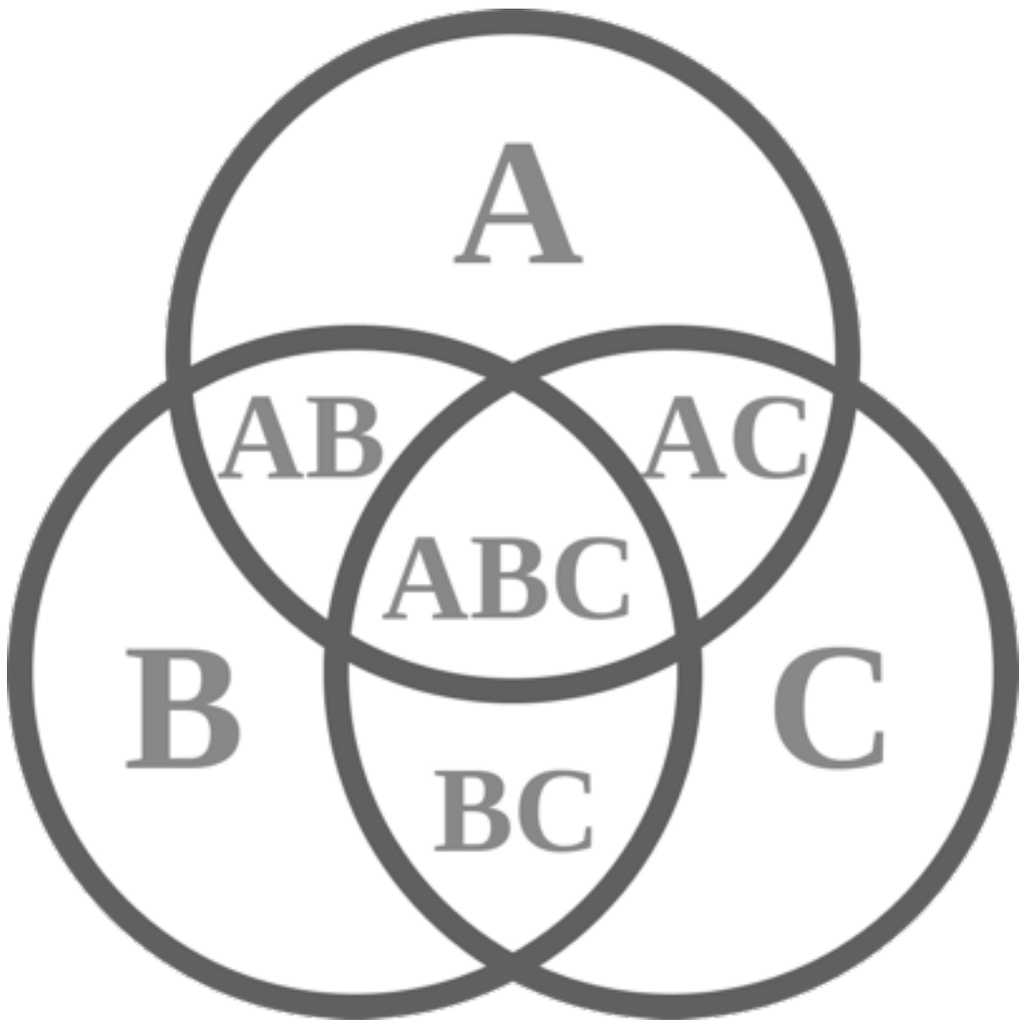
Finally, given a set *A*, we can define its **complement**

$$-\boldsymbol{A} = \{\, x : x \notin A \,\}$$



Venn Diagrams

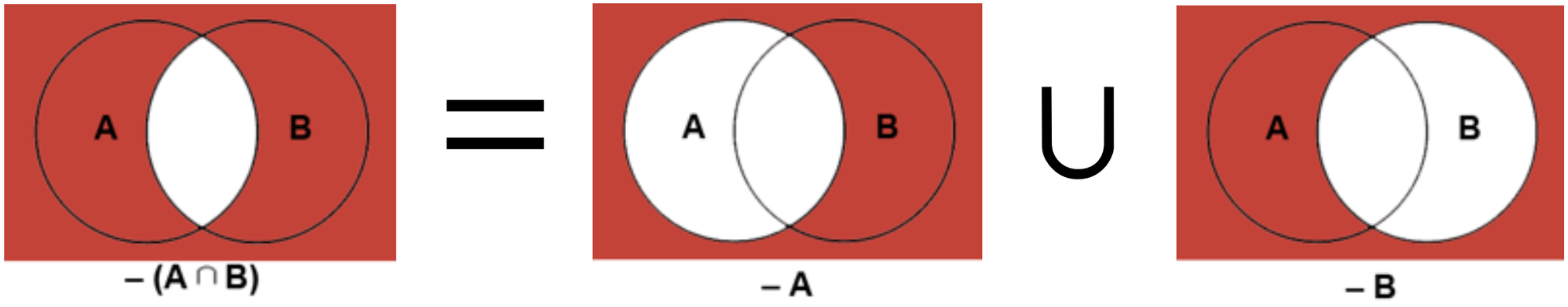
We can illustrate Boolean algebra operations *visually* with sets!
"And" means intersection. "Or" means union.
"Not" means complement.



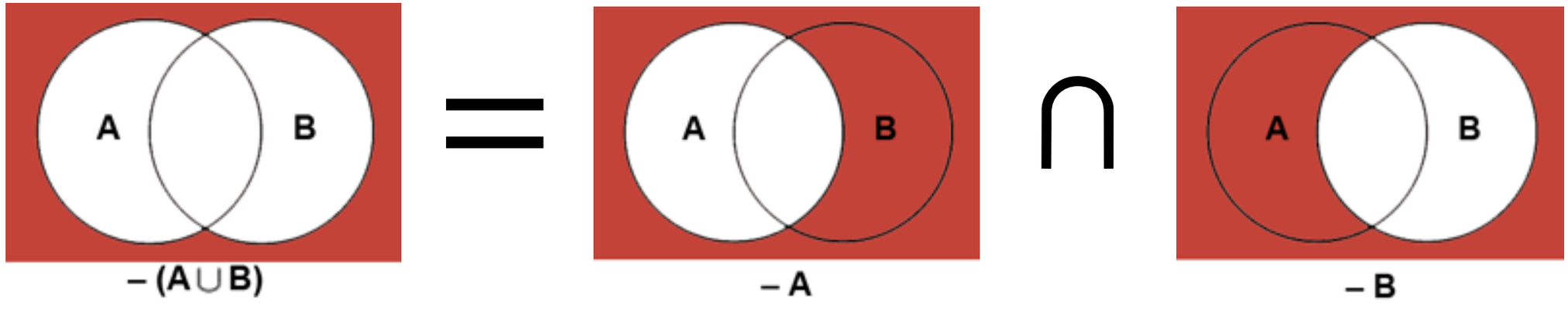
Venn Diagrams

Here are De Morgan's laws illustrated:

NOT (*a* AND *b*) = **(NOT *a*)** **OR** **(NOT *b*)**



NOT (*a* OR *b*) = **(NOT *a*)** **AND** **(NOT *b*)**

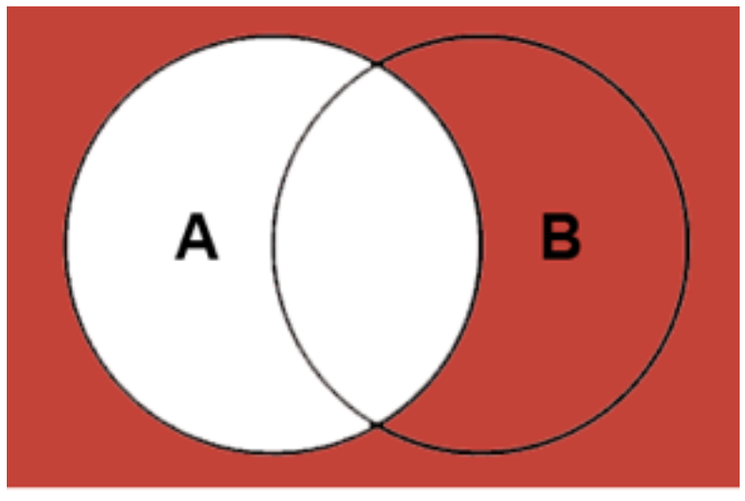


Boolean Algebra

We have already found out some useful rules of this new algebra. Just like with algebra on natural numbers, we can now work with *variables* that can stand for any logical statement.

Let's say we know that *"George's cotton shorts are now dry."*

a = *"George swam a minute ago"*
b = *"George swam in his cotton shorts"*
Since his shorts are now dry, we know that
a AND *b* = False
NOT (*a* AND *b*) = (NOT *a*) OR (NOT *b*) = True
so we can deduce:
(George didn't swim a min ago) or (he didn't swim in his shorts).



George didn't swim a minute ago

Obvious Rules

Both operations *AND* and *OR* are **commutative:**

$$a \text{ AND } b \iff b \text{ AND } a$$

$$a \text{ OR } b \iff b \text{ OR } a$$

So you can switch the order of the symbols if you need.

More Useful Rules

Associative laws work like in regular algebra, you can move parentheses around as long as you are using the same operation:

a OR (*b* OR c) $\iff$ (*a* OR *b*) OR *c*

I ate or (I slept or I swam) $\iff$ *(I ate or I slept) or I swam*

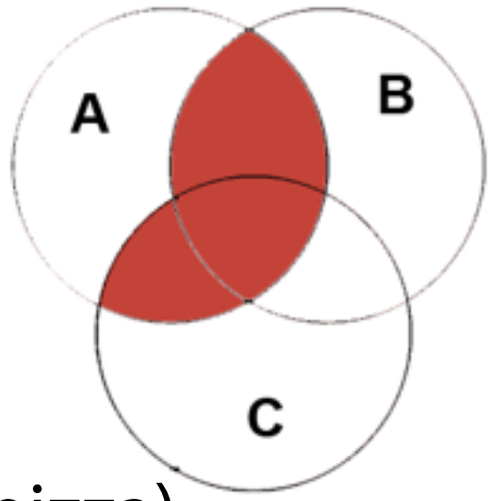
a AND (*b* AND *c*) $\iff$ (*a* AND *b*) AND *c*

I have a dog and (I have a cat and an owl) $\iff$

(I have a dog and a cat) and I have an owl

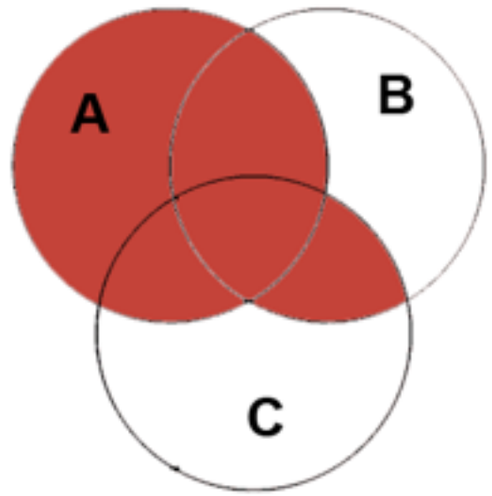
More Useful Rules

Distributive laws in Boolean algebra work as follows,
AND distributes over OR, and vice versa:



a AND (*b* OR *c*) $\Longleftrightarrow$ (*a* AND *b*) OR (*a* AND *c*)

I woke up and (ate sushi or ate pizza) $\Longleftrightarrow$
(I woke up and ate sushi) or (I woke up and ate pizza)



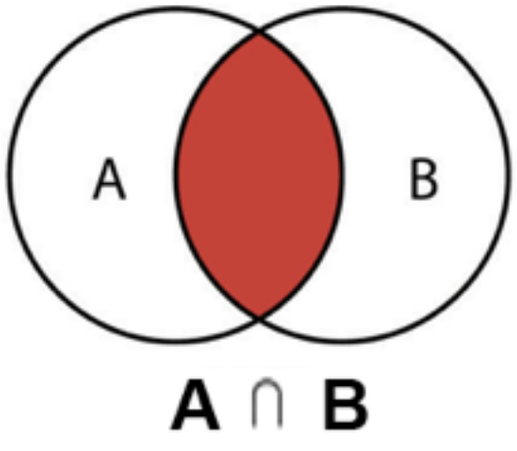
a OR (*b* AND *c*) $\Longleftrightarrow$ (*a* OR *b*) AND (*a* OR *c*)

Identity

Just like with addition and multiplication, we have:

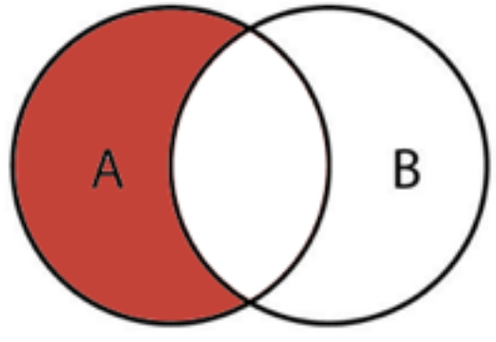
a AND True = *a* for any statement *a*
b OR False = *b* for any statement *b*

This can be used to simplify statements. For example:



A ∩ B

(*a* AND *b*) OR (*a* AND NOT *b*)
= *a* AND (*b* OR NOT *b*)
= *a* AND True
= *a*



A ∩ –B

"I went outside and smoked, or I went outside and didn't smoke"
"Aha, so we just know that you went outside".

Concept: Logic

In this topic, we applied the mathematical techniques that by now will be familiar:

We took a look at things from the real world: in this case, sentences people say, and parts of sentences.

Then we started to generalize and build up a system of notation and abstract concepts. The concepts in this case were ***logical statements*** that could only have two values: ***True and False***. So we defined a set which consisted of just these values, and defined three operations on it: ***AND, OR, NOT.***

We explored the algebra that uses these operations, and found out some of its properties. In particular, we found the ***commutative***, ***associative*** *and* ***distributive*** laws. Also we learned how to properly negate *a AND b* as well as *a OR b* using ***De Morgan's Laws***.

Finally, we discussed parallels to boolean algebra in ***digital computers*** *as well as with* ***set operations***. We found that the operations ***INTERSECTION, UNION, COMPLEMENT*** are exact analogues of the boolean operations above. Thus, set algebra is a boolean algebra, so you can use sets to illustrate logical statements. These illustrations are called ***Venn diagrams***.

In the next topics about deductive reasoning, as well as probability, we will make further use of operations on sets, as well as graphs and trees, to show how all these mathematical concepts are related.