Grade 7/8 Math Circles
February 23, 2022
Boolean Algebra

Booleans

For those of you that participated in the Computer Science lessons from the Fall, you might be familiar with the term "Boolean". A Boolean, in the context of Computer Science, is a data type whose only possible values are True and False.

Let us define A to be a **Boolean variable**. A can be assigned one of two values: 1 or 0. In the context of Boolean algebra, 1 and 0 correspond directly with True and False. Here are some other phrases that are synonymous with 1 and 0.

1	0
True	False
Yes	No
On	Off

Not only do Booleans have a funny sounding name (pronounced "boo - lee - un"), but Boolean logic is essentially binary logic, and is the most fundamental language of computers. In this week's lesson, we are going to be thinking both as computer engineers and as computers themselves. We're going to make sense of 0's and 1's and explore how Boolean algebra compares to the algebra we have already come to know.

However, the concept of Booleans can be extended outside of computers. Some examples of non-computerized objects or situations that can be represented by Booleans are:

- A statement that is either True or False.
- A simple light-bulb that switches between on or off.
- An item on a To Do List that is either complete or incomplete.

Example 1
Consider the question "Are you ready to learn about Booleans?".
A Boolean response to this question would be "True".

Exercise 1

Consider the question "Do you have a pet cat?".

Give a Boolean response to this question.

Boolean Operators

You should be familiar with mathematical operators $(+, -, \times, \div)$ which we use to algebraically manipulate numbers. We'll now look at how we can manipulate Booleans and their truth values using **Boolean operators** (or logical operators).

Negation (NOT)

The first Boolean operator that we'll introduce in this section is $\neg$, meaning "negation" or "**NOT**".

Given a Boolean A, the negation of that variable, $\neg A$ will have the opposite Boolean value.

$$\neg 0 = 1$$
$$\neg 1 = 0$$

So for variables, $A = 1$ and $B = 0$, their negations have the following values:

$$\neg A = \neg(1) = 0$$
$$\neg B = \neg(0) = 1$$

The remaining operators will take two Boolean variables, A and B and combine them to make compound Boolean expressions.

Conjunction (AND)

We define Boolean operator $\wedge$ to mean "**AND**" and the Boolean expression $(A \wedge B)$ to mean "A and B". This can be referred to as the **conjunction** of A and B.

Algebraically, all possible $\wedge$ operations are as follows:

$$\begin{aligned} 0 \wedge 0 &= 0 \\ 0 \wedge 1 &= 0 \\ 1 \wedge 0 &= 0 \\ 1 \wedge 1 &= 1 \end{aligned}$$

Thus $A \wedge B = 1$ only when *both* of A and B are 1, that is $A = B = 1$. Otherwise, $A \wedge B = 0$.

Example 2

Consider the question "Do you like apples and bananas?". When would you answer "No"? When would you answer "Yes"?

Solution:

- If I disliked apples and disliked bananas, I would answer "No", since I like neither.
- If I liked bananas but disliked apples, I would answer "No", since I like only one of the two fruits.
- If I liked apples but disliked bananas, I would answer "No" for the same reason.
- If I liked apples and liked bananas, I would answer "Yes".

Compare the algebraic operations of AND with the responses in the previous example. Which scenario connects with which operation?

Disjunction (OR)

We define the Boolean operator $\vee$ to mean "**OR**" and the Boolean expression $(A \vee B)$ to mean"A or B". We might also refer to this as the **disjunction** of A and B.

Algebraically, all possible $\vee$ operations are as follows:

$$\begin{aligned} 0 \vee 0 &= 0 \\ 0 \vee 1 &= 1 \\ 1 \vee 0 &= 1 \\ 1 \vee 1 &= 1 \end{aligned}$$

We have $A \vee B = 1$ when *at least one* (including both) of A, B is equal to 1. $A \vee B = 0$ when both are equal to 0.

Stop and Think

OR does not have the same meaning as the "or" we use in everyday English.
In English we ask "X or Y?" when we want to decide between thing X and thing Y; we reply by saying one of "X" or "Y".
In math and logic, when we ask "A or B?", we expect the response to be a Boolean value: "True", when one or both of A and B are "True", or "False", if both A and B are "False".

Exclusive Or (XOR)

The last operator that we'll define is $\oplus$ which means "**XOR**" (pronounced "exor"). The Boolean expression $(A \oplus B)$ then means "A **exclusive-or** B".

Algebraically, all possible $\oplus$ operations are as follows:

$$0 \oplus 0 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$
$$1 \oplus 1 = 0$$

We have $A \oplus B = 1$ when *exactly one* of A, B is 1. $A \oplus B = 0$ is either when both A, B are 1, or both are 0.

If XOR is the "exclusive or", you could refer to OR as the "inclusive or". The two disjunctions differ when both A **and** B equal 1: $A \oplus B = 1$ when exactly one of the variables is 1 (i.e. the other variable is 0), whereas $A \vee B = 1$ when at least one variable is equal to 1. You can think of XOR as the "one-or-the-other" operator.

Exercise 2

Consider the question "Do you like apples **or** bananas?"
Suppose that you like apples and you like bananas. Would you respond with "Yes" or "No"?

Operator Truth Table

We can summarize all of our Boolean operators in a **truth table**.

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

The first columns contain our variables A and B. The subsequent rows contain every possible combination of 0's and 1's between the two.

The remaining columns are headed by a Boolean expression/operation of A and B. The remaining rows are then determined by evaluating that row's value of A and B with the column's operation.

For example, if we wanted to know the value of $1 \vee 0$, we would look at the column for $A \vee B$ and then the row where $A = 1$, $B = 0$. We find $1 \vee 0 = 1$.

Exercise 3

Evaluate the following Boolean expressions. You may use the truth table as a reference if needed.

(a) $\neg 0$

(b) $0 \wedge 1$

(c) $1 \wedge 1$

(d) $0 \vee 0$

(e) $1 \vee 1$

(f) $1 \oplus 0$

Boolean Algebra

Now that we have defined our Boolean operators, let's put them to good use. In regular algebra, you should recall something known as the Order of Operations (i.e. BEDMAS, PEMDAS, etc.). The Order of Operations dictates the order in which we apply algebraic operators, or what operations are evaluated first.

Brackets $\rightarrow$ Exponents $\rightarrow$ Multiplication & Division $\rightarrow$ Addition & Subtraction

Example 3 - Order of Operations Review

Evaluate the expression $45 - 9 \times (7 - 2)$.

Solution: As per BEDMAS, we evaluate the addition in the brackets first, then do the multiplication, and finally the subtraction.

$$\begin{aligned} 45 - 9 \times (7 - 2) &= 45 - 9 \times (5) \\ &= 45 - (45) \\ &= 0 \end{aligned}$$

The order in which we apply **Boolean operators** will be as follows.

Brackets () $\rightarrow$ NOT ($\neg$) $\rightarrow$ AND ($\wedge$) $\rightarrow$ XOR ($\oplus$) $\rightarrow$ OR ($\vee$)

And just like regular algebra, we evaluate from left to right if there is equal precedence/ordering.

Example 4

Evaluate $B \wedge (A \vee \neg A) \vee (B \oplus A)$ where $A = 1$ and $B = 0$.

Solution: First we can substitute in our given values, then following the order of operations we get

$$\begin{aligned} 0 \wedge (1 \vee \neg 1) \vee (0 \oplus 1) &= 0 \wedge (1 \vee (0)) \vee (0 \oplus 1) && \text{(NOT in Brackets)} \\ &= 0 \wedge (1) \vee (1) && \text{(XOR in Brackets)} \\ &= (0) \vee 1 && \text{(AND before OR)} \\ &= 1 \end{aligned}$$

Exercise 4

Evaluate the following in respect to the order of operations where $A = 1$, $B = 1$, and $C = 0$.

(a) $A \vee C \wedge B$

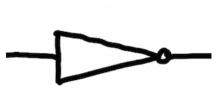
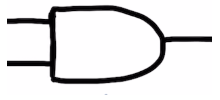
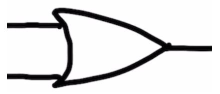
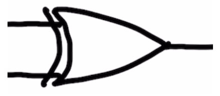
(b) $(A \oplus C) \wedge \neg B$

One of the best things about Boolean algebra is that there are no large values; we only ever work with 0s or 1s. Boolean algebra is simple if we are familiar with the operators and their ordering. As such, we can then evaluate or simplify complex expressions without much difficulty. We'll look at different tricks and rules that will make larger expressions more easily solved in the Problem Set.

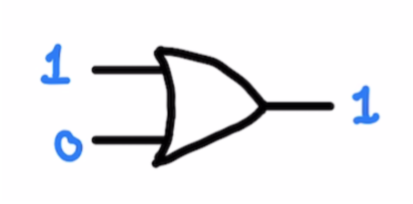
Logic Gates

In the fields of computer science, computer engineering or physics, a direct application of Boolean algebra is in electronic circuits. Our Boolean operators each have a corresponding physical **logic gate** that takes Boolean inputs and to produce the appropriate Boolean output. Logic gate symbols are used when constructing **circuit diagrams**, which are useful for planning out wiring or digital circuits before they are manufactured physically.

Below are the logic gate symbols that correspond to each Boolean operator when drawing circuit diagrams.

NOT ($\neg$)	**AND** ($\wedge$)	**OR** ($\vee$)	**XOR** ($\oplus$)
			

Below is an example of an OR logic gate. It takes two inputs, 1 and 0, on its left. Since $1 \vee 0 = 1$, the gate will output a 1.

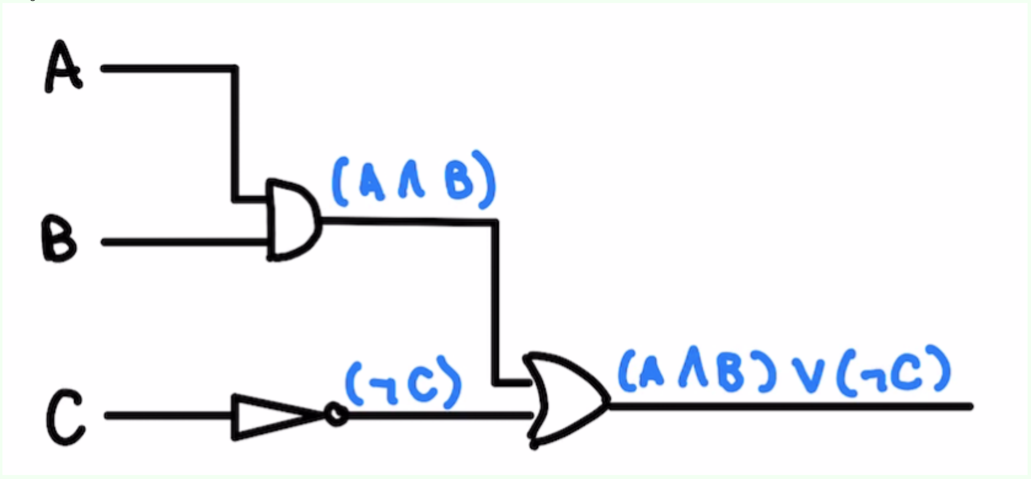


In general, logic gates take inputs from the left and output on the right. As before, we can combine logic gates to represent more complicated Boolean expressions. The order of operations that we defined earlier will be important for accurately drawing and reading circuit diagrams.

We will be mainly working with circuit diagrams with variable inputs. You can think of the inputs as switches/buttons in this way; they might sometimes be on and they might sometimes be off.

Example 5

Here is a circuit diagram for the expression $(A \wedge B) \vee (\neg C)$. The output of each logic gate is labeled for your convenience.

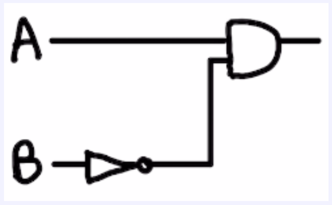


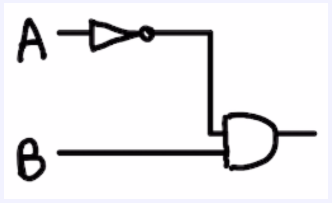
If, for example, we knew that $A = 1$, $B = 0$, and $C = 1$, we can determine the output of the expression by following the circuit in a series of steps:

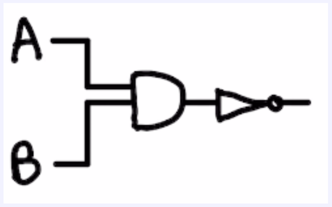
- $A = 1$ and $B = 0$ meet at their AND gate, and since one of the inputs is 0, the output of the AND gate is also 0.
- The input of the NOT gate is $C = 1$ so the output is 0.
- The OR gate inputs are 0 and 0 and therefore the output of the circuit is 0, at least for these given inputs.

Exercise 5

Determine values for inputs A and B such that the output of the circuit is 1.

(a) 

(b) 

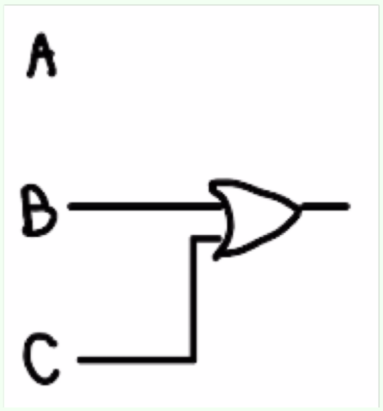
(c) 

Example 6 - Drawing Logic Gates

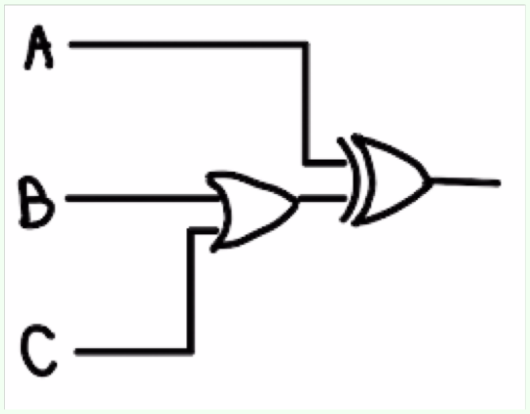
Let's draw a circuit diagram for the expression $C \wedge ((C \vee B) \oplus A)$.

We begin our circuit diagram with our three Boolean inputs, A, B, and C on the left.

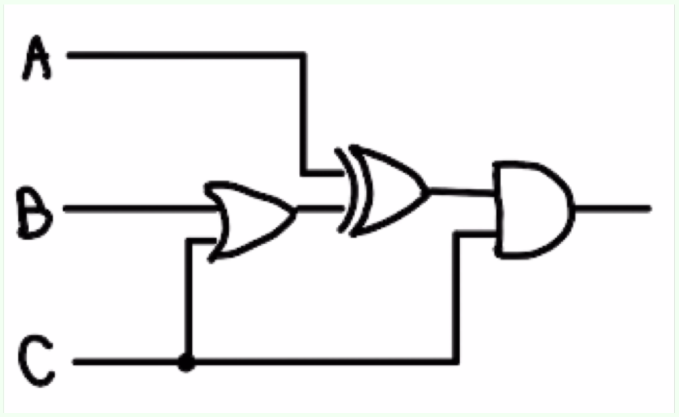
We continue with what comes first algebraically based on the order of operations: $C \vee B$. We draw lines from our variables so that they meet at the logic gate symbol corresponding to OR.



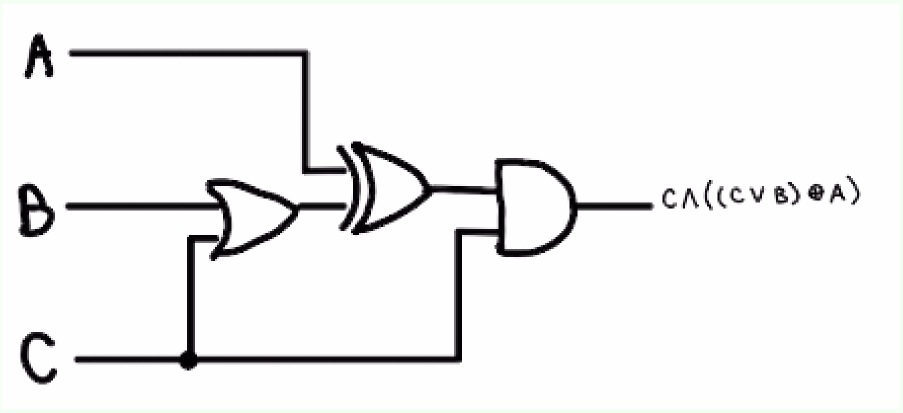
Next comes $(C \vee B) \oplus A$. We draw a line from A and join it with the output of $(C \vee B)$ with the logic gate symbol for XOR.



Finally we want to take the conjunction ($\wedge$) of our most recent output and C. Since we have already drawn a line from C, we add a branch off of that line to say that this is a separate path from the OR one. We join this branch with the output of the XOR gate with an AND gate.



Lastly, we label the circuit output with the full expression



You can imagine that different circuit configurations have different properties that make them useful. Although electronic circuits seem to be the most common use for logic gates, the generalized concept can be applied elsewhere.

If you are a fan of the video game Minecraft you might have heard of “Redstone” and its accompanying buttons, levers, and pistons. You might also know of the crazy things that people have been able to build using its technical properties such as digital computers. Some of these magnificent feats of Minecraft engineering can be accomplished with the help of logic gates. If you’re interested, you can do some research on how to construct and apply logic gates in Minecraft. You could even try some of the Problem Set questions using the logic gates you build for reference.