# Logic

## Learning Outcomes

On successful completion of this Unit students will be able to,

- Describe logical statements in logic notation
- Construct simple truth tables

## A Brief Background on Logic

Last lesson when we learned about er'rythang being represented with 1's and 0's. This only explains the format things are stored in, and not the methods by which stored, or calculated, in the first place. What magic lays behind everything from *IF* statements, to *multiplication*.

It all boils down to logic. In particular Boolean logic. *Boolean logic is the basis for short circuit logic in Javascript*

Say we make a statement such as "there is a cat". This statement would evaluate to `true` or `1` when there is a cat, otherwise it would be `0`.
Often we will represent this as a symbol, e.g. $letc = cat$ now $c = 1$ if there is a cat, and $c = 0$ if there isn't a cat.

These statements can be combined together:

> "If I was a rich girl (na, na)
> See, I'd have all the money in the world"
> **- Gwen Stefani, Rich Girl**

This should be familiar to y'all. It's a very short example of an IF statement. **If** Gwen Stefani "was a rich girl" **then** she would have all the money in the world.

r := "Gwen Stefani is a rich girl"
m := "Gwen Stefani has all the money in the world"

We would then say that **r** *implies* **m**. If r, then m. Or written in logic notation:

$$r \Rightarrow m$$

*Always choose a nice lower case letter to represent a bit of logic, and express which one you have chosen so that everyone knows what you mean when you use it in formulae*

So **if** we said r = `True`, or $r = 1$ (Which are both essentially the same statement), **then** $m = 1$.
Does that mean we can assert "If Gwen Stefani has all the money in the world, she would be a rich girl".
Not necessarily; in fact that's why the arrow is there. A better example of this might be:

> "If it is raining I will have my umbrella"
> **- Me, uninspired example.**

That does not necessarily mean "If I have my umbrella then it is raining".

I like to visualise it this way.

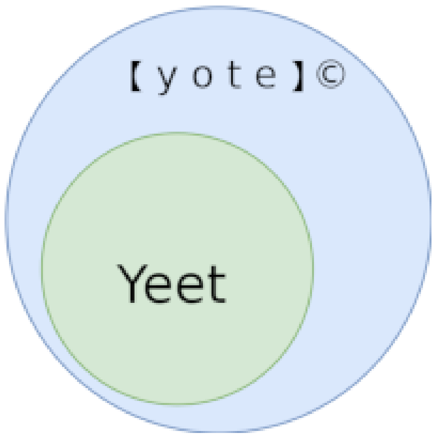

Figure 1: Logical Implies Diagram

You will notice that while yote© is in the blue circle, yeet is in both circles. This means we can assume something in the green circles is in the blue circle, but something in the blue circle isn't necessarily in the green circle.

Logic statements are not always straightforward, however.

> "If you wanna be my lover, you gotta get with my friends"
> **- The Spice Girls, Wannabe**

l := "Be my lover"
f := "Get with my friends"

If we were too haphazard here we might write $l \Rightarrow f$, following the same ordering as the previous examples. This would be *WRONG*. That means you would be saying "If you gotta get with my friends you wanna be my lover" which as we just addressed is not quite the same thing. Rather, it would be

$$f \Rightarrow l$$

At the core of it all we are still just trying to evaluate whether the statement is true though. Think about when the whole statement evaluates to true. It can be true if $l$ and $f$ are both true, or if $l$ is true.

There are several logic operations which take input(s) and do interesting things with them. There are *AND*s, *OR*s, *XOR*s. When combined with with a *Negation* operator we even get an additional *NAND*, *NOR*, and the furtive *XNOR*. This is link between theoretical logic and physical circuitry, as these all can be represented as arrangements of transistors, and with enough of these logical operations you can perform any operation!

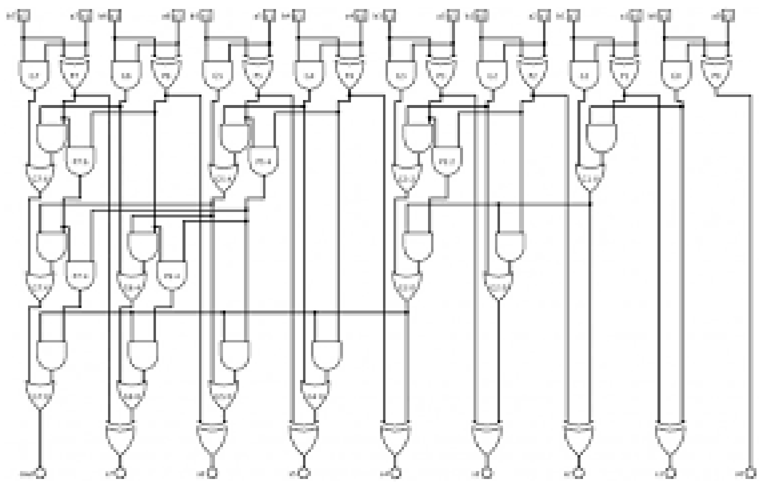

Figure 2: 8-Bit Ladner-Fischer Adder

Look! Lo and behold! It adds two numbers together bit by bit and all it uses are ANDs, NANDs, ORs, and XORs!

But we ain't here to do electrical engineering.

## Common Logic Operations

### ¬ Not

Takes an input and flips it. Consider the following.

$$a = 1 \iff \neg a = 0$$

Which is to say "*a* is *True*" is logically equivalent to "*not a* is *False*". Really all not does is flip whatever comes after it. If there are brackets, then not flips the evaluated expression inside of the brackets.

$$\neg(\neg a)$$

The external not is applied to the not expression inside the brackets. In this case the two nots would cancel each other out and the expression would be as good as saying $a$ (*Two negatives make a positive*).

$\wedge$ **AND**

$$a \wedge b$$

$\wedge$ closely remembers an A. This can help you avoid confusion with symbols which are introduced later.

This statement evaluates to 1 ONLY when *both a* and *b* are true. I like to imagine there is a pipe with two valves in series, one after another. It is only when both valves are open that the *turmeric iced-latte* can pour through.

Used in a sentence:

> "He gon' skrrt and hit the dab like Wiz Khalifa"
> **- Mia Kalifa, Mia Khalifa on iLOVEFRIDAY ft. Mia Khalifa**

If he only either *skrrts* or *dabs* then this statement is false.

Or more simply:

> "You said"bell peppers and beef." There's no beef in here. So you wouldn't really call it "bell peppers **and** beef," now would you?"
>
> **- Spike Spiegel, aboard the Bebop.**

We can also show this in the form of a *truth table.*

| bell peppers | beef | $bellpeppers \wedge beef$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

It might not have been bell peppers *and* beef, but it could have been bell peppers *or* beef...

### $\vee$ **OR**

$$a \vee b$$

For this expression to evaluate to *true*, all we need is either $a$ or $b$ to be *true*. Two parallel pipes with a shared source and sink. So long as one of them is open then the *Sriracha flavoured Dr. Pepper* will flow.

I would like an Nintendo Switch or a Playstation 4. Both would be great but I really would accept either.

### $\oplus$ **XOR**

$$a \oplus b$$

*Exclusive or* (**xor**), is *false* when both $a$ and $b$ are *true*.

For example, my Grandfather infuriates my Grandmother. When she asks him "Tea or Coffee", he just responds "Yes".

The last or we dealt with was called an *inclusive* or, which means that when it is both $a$ and $b$ that are *true* it is as *true* as though only $a$ or $b$ were *true*.

Another example of exclusive or:

> "Hit or Miss"
> **- Mia Kalifa, Mia Khalifa by iLOVEFRIDAY ft. Mia Khalifa**

**Hit** can be *true* or **miss** can be *true*, but not both.

### Bitstrings!

If you take two strings of bits, A = `1010`, B = `0110`, you can actually combine them together with a logical operator by performing it on each bit one at a time. i.e. for $C = A \wedge B$ , $C_i = A_i \wedge B_i$ .

Therefore:

$$1010 \wedge 0110 = 0010$$

## Challenges

1. Convert the following into logic expressions:
   - If there is smoke then there's fire
   - If it is raining and I'm outside then I will have my umbrella
   - Either Trump is being blackmailed or he isn't, if he is then he is Putin's puppet

- I will wear bunny ears on twitch if I get another subscriber or a donation

2. Fill out a truth table as done for AND:
   - To show the difference between **or** and **xor**
   - To evaluate $(\neg p \vee q) \wedge p$ for all values of p or q.
   - To evaluate $\neg(p \Rightarrow q)$ for all values of p or q.
   - To compare $p \oplus q$ with $(p \vee q) \wedge \neg(p \wedge q)$
3. Evaluate these:
   - $1101 \vee 1001$
   - $0101 \wedge 1001$
   - $\neg 0110$
   - $0110 \oplus 1100$

**Beast Mode:**

*Hint: you may need to convert between bases, and treat the numbers as bitstrings*

- $56 \vee 72$
- $24 \wedge 11$

**Beast Mode+:**

*Hint: Feel free to find the hex code or decimal value for letters from an ascii table*

- $33 \oplus 5$
- "F" $\vee$ "a"

**Beast Mode ++:**

*Ever wondered how encryption works? XOR is at the foundation of AES, a common encryption algorithm*

Cyphertext$_1$0 = 27 14 4 18 21 26 28 3 80

Key = "password1"

To find the encrypted word; you must xor each of the characters of Cyphertext with those of the key.
*Hint: First letter is 'k'; and I recommend writing a quick script in ruby or JavaScript to do this.*

## References

**Source:** https://www.youtube.com/embed/e8LXrm0SUAk Discrete Math - Logic