THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

DEPARTMENT OF
COMPUTER SCIENCE
Te Tari Rorohiko

2025

Contributors

J. Turner

R. Mercado

V. Moxham-Bettridge

A. Hinze

C. Pilbrow

N. Kanji

T. Elphick

S. Cunningham

J. Kasmara

BOOLEAN ALGEBRA

Last week we introduced the basics of database design and SQL. Today we will introduce you to boolean algebra, one of the fundamental building blocks of the modern computer.

What is Boolean algebra?

Boolean algebra is a form of mathematical logic different to other forms of algebra you may have encountered before. It is built upon *boolean values* in which there are two acceptable options, either something is true or something is false.

The basic rules of this algebra were written by George Boole in 1847, later refined and applied to set theory. While having many uses, it constitutes the basis for the design of circuits used in electronic digital computers.

Boolean algebra allows us to reason and find the truth of certain equations. In regards to digital circuits we can develop, simplify, transform circuits and so on. In programming, we can use it to simplify our conditional if and while statements. Meanwhile for relational databases, we can use it to assist in information retrieval tasks using SQL.

How does Boolean algebra work?

A logical assertion (statement) in boolean algebra is called a proposition and is made up of boolean values and operators. There are three main operators we introduce, and $\wedge$, or $\vee$ and not $\neg$. The and and or operators take two boolean values and return one boolean value, while the not operation takes one boolean value and returns a single boolean value.

Boolean algebra has a set of defined semantics, that is, a set of defined mathematical rules which define how the algebra works, there are as follows:

- A boolean term is either true or false.
- A boolean variable such as a is either true or false.
- The basic operators are: and $\wedge$, or $\vee$ and not $\neg$.

We can describe the meaning of a proposition by use of a truth table. Next we will describe the truth tables for each of the basic operators.

AND

The meaning of $a \wedge b$ depends upon the truth of A and the truth of B. Informally, the overall term is true when both variables are true. Let's consider the truth table:

a	b	$a \wedge b$
F	F	F
F	T	F
T	F	F
T	T	T

Figure 41: The Truth Table for the And operator

In figure 41 the column $a \wedge b$ evaluates the truth of the proposition based on the values of a and b. Note that here we list all the possible values combinations for both a and b in order to determine the truth of the proposition in all scenarios.

OR

The meaning of $a \vee b$ depends upon the truth of A and the truth of B. Informally, the overall term is true when either variable is true. Let's consider the truth table:

a	b	$a \vee b$
F	F	F
F	T	T
T	F	T
T	T	T

Figure 42: The Truth Table for the Or operator

Much like our and example the column $a \vee b$ evaluates the truth of the proposition based on the values of a and b. Similarly to and, we list all the possible values combinations for both a and b in order to determine the truth of the proposition in all scenarios.

NOT

The meaning of $\neg a$ depends upon the truth of a. Informally, the overall term becomes the opposite value of the original a. Let's consider the truth table:

a	$\neg a$
F	T
T	F

Figure 43: The Truth Table for the Not operator

Unlike our previous operators, the not operator's truth is only dependent on one variable. Note that we still list all possible options for a in order to determine the truth of the proposition in all scenarios.

Boolean and Binary

While here we have described our truth tables with "T" for true and "F" for false, other mappings do exist. For instance, in binary we often map true to 1 and false to 0. A similar mapping exists in digital circuits and allows us to apply the rules of boolean logic to simplify circuit designs.

Exercises:
Fill in the following tables to evaluate the truth table for each expression.

1. $\neg a \barwedge b$

a	b	$\neg a$	$\neg a \barwedge b$
F	F		
F	T		
T	F		
T	T		

2. $\neg(a \wedge b)$

a	***b***	***a ? b***	$\neg(a \wedge b)$
F	F		
F	T		
T	F		
T	T		

3. $(a \wedge b) \vee \neg b$

a	***b***	$a \wedge b$	$\neg b$	$(a \wedge b) \vee \neg b$
F	F			
F	T			
T	F			
T	T			

Equations

Just as you would in normal algebra, boolean algebra has equations/laws which are considered to always hold true thus allowing us to simplify an expression. In the following, variables X, Y and Z stand for any statements X, Y, and Z:

- Constant
  - $X \wedge T = X$
  - $X \vee F = X$
  - $X \vee T = T$
  - $X \wedge F = F$
- Negation
  - $\neg T = F$
  - $\neg F = T$

- Double Negation
  - $\neg \neg X = X$

- Associativity
  - $X \wedge (Y \wedge Z) = (X \wedge Y) \wedge Z$
  - $X \vee (Y \vee Z) = (X \vee Y) \vee Z$

- Commutativity
  - $X \wedge Y = Y \wedge X$
  - $X \vee Y = Y \vee X$

- Distributivity
  - $X \wedge (Y \vee Z) = (X \wedge Y) \vee (X \wedge Z)$
  - $X \vee (Y \wedge Z) = (X \vee Y) \wedge (X \vee Z)$
- Idempotence
  - $X \vee X = X$
  - $X \wedge X = X$

- Absorption
  - $X \vee (X \wedge Y) = X$
  - $X \wedge (X \vee Y) = X$
- De Morgan's
  - $\neg(X \wedge Y) = \neg X \vee \neg Y$
  - $\neg(X \vee Y) = \neg X \wedge \neg Y$

We can use truth tables to prove these laws are correct. We do this by calculating the truth table for each side of the equation and then comparing the final column truth table.

Exercises:

1. Calculate the truth table for de Morgan's $\neg(X \wedge Y) = \neg X \vee \neg Y$ using the table below:

X	***Y***	***¬X***	***¬Y***	***¬X ∨ ¬Y***	***X ∧ Y***	***¬(X ∧ Y)***
F	F					
F	T					
T	F					
T	T					

2. Calculate the truth table for absorption $X \cdot (X + Y) = X$ using the table below:

X	Y	$X + Y$	$X \cdot (X + Y)$
F	F		
F	T		
T	F		
T	T		

3. Calculate the truth table for distributivity $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$ using the table below:

X	Y	Z	$Y + Z$	$X \cdot (Y + Z)$	$X \cdot Y$	$X \cdot Z$	$(X \cdot Y) + (X \cdot Z)$
T	T	T					
T	T	F					
T	F	T					
T	F	F					
F	T	T					
F	T	F					
F	F	T					
F	F	F					

Simplifying Expressions

We can use the laws described above to simplify a boolean algebra expression like we would in traditional algebra. Next we give an example of applying these rules. The aim here is to simplify an expression such that it has the minimal number of variables and operators which preserve the same meaning. Consider the following expression: $\overline{X} \cdot (X + Y)$

We cannot simply apply absorption here as $\neg X$ and X are not the same. Therefore, we will first apply distributivity to get the following: $(\neg X \vee X) \wedge (\neg X \vee Y)$

If we consider the truth table of $\neg X \vee X$ we can show that this simplifies to true:

X	$\neg X$	$\neg X \vee X$
F	T	T
T	F	T

Note that in this instance we need not consider all four possible values as if X is true $\neg X$ is false and vice versa. Therefore our expression simplifies to: $T \wedge (\neg X \vee Y)$

But by our constant law anything and true is simply the expression itself, therefore our expression simplifies to just: $(\neg X \vee Y)$.

In terms of layout we would define this simplification as follows:

$\neg X \vee (X \wedge Y)$	
$\Leftrightarrow (\neg X \vee X) \wedge (\neg X \vee Y)$	*(distributive law)*
$\Leftrightarrow T \wedge (\neg X \vee Y)$	$(\neg X \vee X \Leftrightarrow T)$
$\Leftrightarrow \neg X \vee Y$	$(X \wedge T = X)$

Note that the "$\Leftrightarrow$" symbol simply means each line is logically equivalent, it is similar to the "=" sign in standard algebra.

Exercises:
Simplify the following expressions:

1. $(\neg A \vee B) \wedge A)$
2. $X \vee (\neg X \wedge \neg Y \vee Y)$

Digital Circuits

The logical operators that we have introduced so far can be used in digital circuits, where 0 means false and 1 means true. The operators behave as you would expect in boolean algebra but use what we call *logical gates* to represent the different operators.

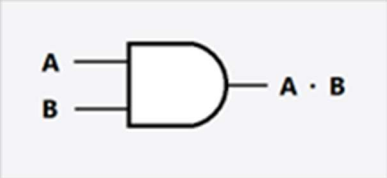

Figure 44: And gate

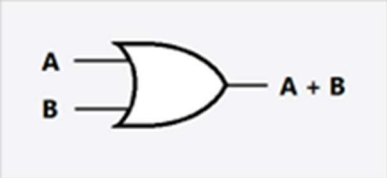

Figure 45: Or gate

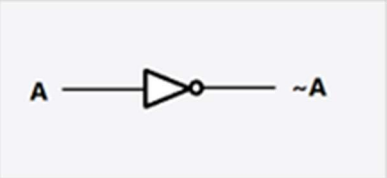

Figure 46: Not gate

Note that in these diagrams a slightly different notation is used to what we used in the previous section, but the operators still behave in the same way. When creating a digital circuit we give inputs and outputs to our different gates. We can then switch inputs "on" and "off" to determine what the expected output will be. Let's experiment with this using the CircuitVerse simulator (https://circuitverse.org/simulator).

Exercises:

1. Let's start out simple and build a digital circuit with two inputs, a single output and an and gate. Your circuit should look like the following:

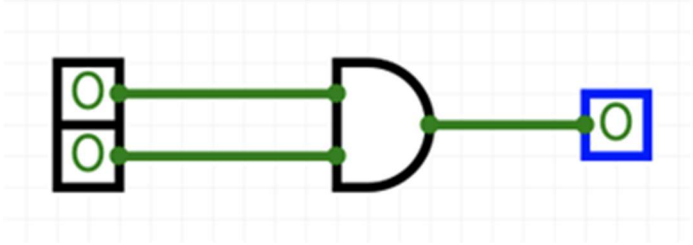

Figure 47: And Gate Digital Circuit

2. Try experimenting with the different inputs, what is the output "0" and "1" or vice versa? What is the output for "1" and "1"? Is this what you expected?
3. Let's build a circuit to match our rule for absorption $X \cdot (X + Y) = X$, our final output should match the value of X. Your diagram should look something like the following:

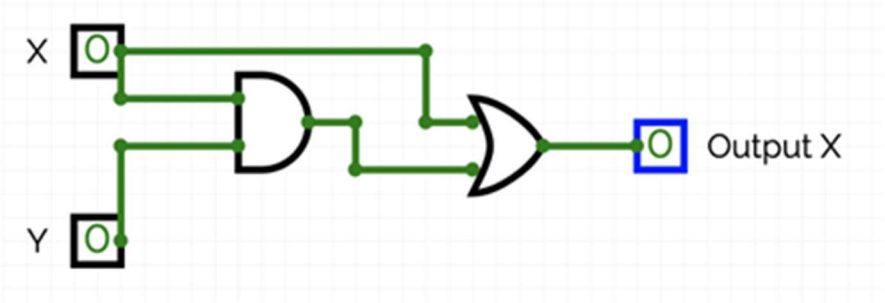

Figure 48: Absorption Digital Circuit

In a digital circuit we may want to set and reset a certain bit. We can do this by using a Set and Reset (SR) latch. This is comprised of two NOR gates (meaning not or). The SR latch allows us to remember the previous value (memory) but can be reset if we wish to use this bit again.

4. Build the SR latch circuit as depicted below:

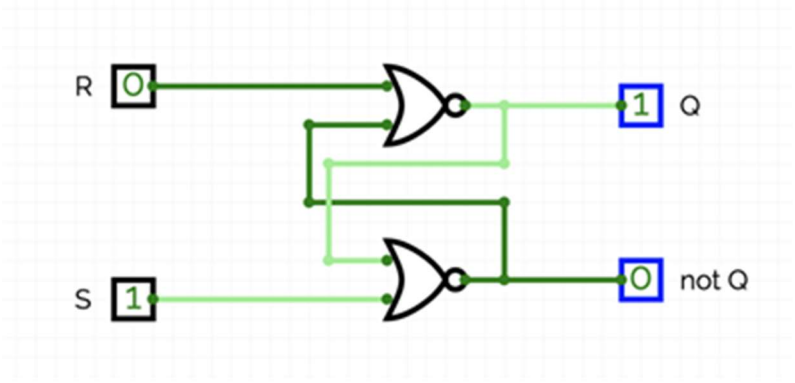

Figure 49: SR Latch Circuit

Notice that the output of the NOR gate is used as an input to the other NOR gate, this is called a cyclic circuit.

5. Does the SR latch behave as you would expect?
6. Can you determine the truth table for the SR latch?

Summary

In this session we introduced the basics of Boolean algebra and touched on digital circuits using logic gates. By understanding boolean algebra you can simplify any logical expression using the laws of the algebra in order to simplify your code or a digital circuit for instance. While we have focussed primarily on truth tables, the laws introduced can be used to prove propositions and formulas in first order logic.

Advanced Exercises

- Calculate the truth table for the following expression: $(\neg X \lor Y) \land (Z \lor X)$
- Simplify the following expression using the boolean algebraic laws: $\neg(\neg(A \lor \neg B) \lor A) \lor \neg A$
- Can you build a gated SR latch? What is the difference?
- Can you build a D-type latch? What is the difference to the previous two latches?

Useful Resources

- Simplifying Boolean Expressions: https://www.youtube.com/watch?v=XMCW6NFLMsg
- The Gated SR Latch: https://www.youtube.com/watch?v=HxAhOETcvr4
- The Gated D Latch: https://www.youtube.com/watch?v=y7Zf7Bv_J74
- The Mathematics of Boolean Algebra: https://plato.stanford.edu/entries/boolalg-math/