UNIVERSITY OF WATERLOO

Faculty of Mathematics
Waterloo, Ontario N2L 3G1

Centre for Education in
Mathematics and Computing

Grade 7/8 Math Circles

Fall 2018 - November 6/7/8

Boolean Logic

Logic is everywhere in our daily lives. We use logic to determine whether someone is lying. We use logic to create the best plan of action to complete a task. Algebra problems and Sudoku puzzles use logic. In fact, mathematics itself is almost entirely based on logic. At its core, logic is the study of arguments, their validity, and their truth values.

Making Arguments

Every argument has premises and a conclusion. If the conclusion follows from the premises, then the argument is *valid*. If the premises are true and the argument is valid, then the argument is *sound*. A **statement** is a sentence that has a definite state of being either true or false. For example,

1. $2 + 3 = 6$. (A false statement.)
2. 29 is a prime number. (A true statement.)
3. $\pi + 1 \geq 5$. (A false statement.)

The **truth value** of a statement is true if the statement is true or false if the statement is false.

We will look at some arguments today that have two premises and one conclusion (altogether three statements). Some statements can be written as if/then statements. For example, if I eat a lot of candy, then I will be sick. Let's accept this statement as true. Now we look at an argument:

Premise 1: If I eat a lot of candy, then I will be sick.
Premise 2: I just ate 1000 Twizzlers.
Conclusion: I will be sick.

Our argument has the form **If A then B. A. Therefore B.** which must give a true conclusion if the premises are true. Consider this argument:

Premise 1: If I eat a lot of candy, then I will be sick.
Premise 2: I am sick.
Conclusion: I ate way too much candy earlier.

Is the conclusion true?

No. There are many reasons for being sick and candy is not the only cause. Therefore we cannot draw this conclusion.

Logical Operators

An operator works on two inputs to give a result. These two inputs can be numbers and the operator can be addition, subtraction, multiplication, and division. The two inputs can also be statements (which are either true or false) which can be combined using logical operators. Often we use **0 to represent false** and **1 to represent true**. Here are four logical operators (and a Google Doodle, for George Boole's 200th birthday, utilizing them `https://g.co/doodle/cjk8vt`):

NOT takes one statement and produces the opposite statement (that has the opposite truth value). NOT is represented by the symbol '$\neg$'. For example, $\neg True \rightarrow False$ and $\neg False \rightarrow True$.

OR takes two statements and produces true if at least one of the statements is true and false if both statements were false. OR is represented by the symbol '$\vee$'. For example, $True \vee False \rightarrow True$.

AND takes two statements and produces true if both statements are true and false if at least one of the statements is false. AND is represented by the symbol '$\wedge$'. For example, $True \wedge False \rightarrow False$.

XOR takes two statements and produces true if both statements' truth values are different and false if both statements' truth values are the same. XOR is represented by the symbol '$\veebar$'. For example, $True \veebar True \rightarrow False$.

We can immediately combine the last three operators with NOT to create NOR, NAND, and XNOR. Put simply, they produce the opposite outputs from OR, AND, and XOR, respectfully.

NOR takes two statements and produces false if at least one of the statements is true and true if both statements are false. NOR is represented by the symbol '↓'. For example, $True \downarrow False \rightarrow False$.

NAND takes two statements and produces false if both statements are true and true if at least one of the statements was false. NAND is represented by the symbol '↑'. For example, $True \uparrow False \rightarrow True$.

XNOR takes two statements and produces true if both statements have the same truth values and false if and both statements have different truth values. For example, $True$ XNOR $True \rightarrow True$.

Thinking Question What is the truth value of $\neg\neg True$?
$\neg\neg True = \neg False = True$. Thus, two NOT operators cancel each other out.

Truth Tables

Truth tables are summaries of logical operators. Let A and B be statements. The A and B columns of our truth tables will, together, list all possible truth value combinations of the statements for the operator to use. This is relatively easy since statements only have two possible truth values — true or false — we only have four possible combinations. For the our logical operators today, the order does not matter.

The truth table for NOT:

A	$\neg A$
True	False
False	True

The truth table for OR:

A	B	$A \vee B$
True	True	True
True	False	True
False	True	True
False	False	False

Complete the truth table for NOR:

A	B	$A \downarrow B$
True	True	False
True	False	False
False	True	False
False	False	True

Now complete the truth table for AND:

A	B	$A \wedge B$
True	True	True
True	False	False
False	True	False
False	False	False

Now complete the truth table for NAND:

A	B	$A \uparrow B$
True	True	False
True	False	True
False	True	True
False	False	True

The truth table for XOR:

A	B	$A \veebar B$
True	True	False
True	False	True
False	True	True
False	False	False

Now complete the truth table for XNOR:

A	B	A XNOR B
True	True	True
True	False	False
False	True	False
False	False	True

Truth tables are useful in proving that certain logical operations are equivalent to other logical operations. The symbol used to denote equivalency is '$\equiv$'.

Example 1:

(a) Write out $A \downarrow B \equiv \neg(A \vee B)$ in words.

A NOR B is equivalent to NOT the result of A OR B.

(b) Prove $A \downarrow B \equiv \neg(A \vee B)$ using a truth table.

A	B	$A \downarrow B$	$A \vee B$	$\neg(A \vee B)$
True	True	False	True	False
True	False	False	True	False
False	True	False	True	False
False	False	True	False	True

Since the columns representing $A \downarrow B$ and $\neg(A \vee B)$ are identical for the corresponding truth values of the components, we can conclude that $A \downarrow B \equiv \neg(A \vee B)$.

Example 2: Let A, B, and C be statements. Use the following truth table to verify the distributive law $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.

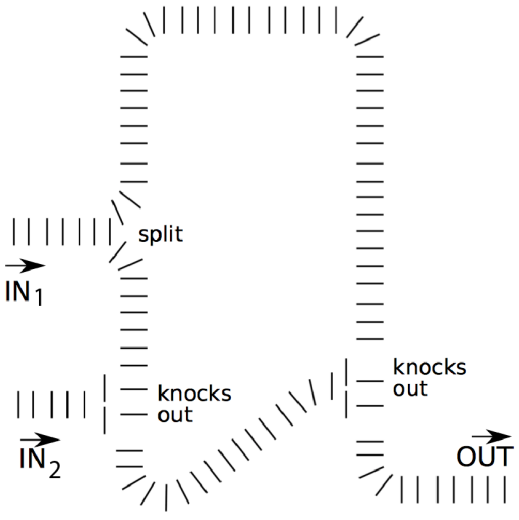
A	B	C	$B \wedge C$	$A \vee (B \wedge C)$	$A \vee B$	$A \vee C$	$(A \vee B) \wedge (A \vee C)$
True	True	True	True	True	True	True	True
True	True	False	False	True	True	True	True
True	False	True	False	True	True	True	True
True	False	False	False	True	True	True	True
False	True	True	True	True	True	True	True
False	True	False	False	False	True	False	False
False	False	True	False	False	False	True	False
False	False	False	False	False	False	False	False

Since the columns representing $A \vee (B \wedge C)$ and $(A \vee B) \wedge (A \vee C)$ are identical for the corresponding truth values of the components, we can conclude that $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$.

The 10,000 Domino Computer

A few years ago, a mathematician named Matt Parker and a team of people built a domino computer that was capable of adding any two three-digit binary numbers and returning the four-digit binary number answer. To watch the computer add 4 and 6 together, watch the youtube video `www.youtube.com/watch?v=0pLU__bhu2w` until the 5 min mark.

In the video, Matt describes a smaller circuit that they use in the computer. We can describe all the possible outcomes of this circuit mathematically using ones and zeroes. If a domino is standing up, we will use a zero. If a domino falls over, we will use a one.



For example, if input1 doesn't fall over and input2 doesn't fall over then the output will not fall over. Putting this in our table we get a row of all zeroes. Fill in the rest of the rows.

INPUT1	INPUT2	OUTPUT
0	0	0
1	0	0
0	1	0
1	1	1

What truth table does this table of inputs and outputs remind you of? Recall that in computing, 0 means false and 1 means true. An AND truth table.
This circuit is an **AND circuit**.

Let's take another look at the truth table for XOR and turn it into a table of inputs and outputs that represents an XOR domino circuit.

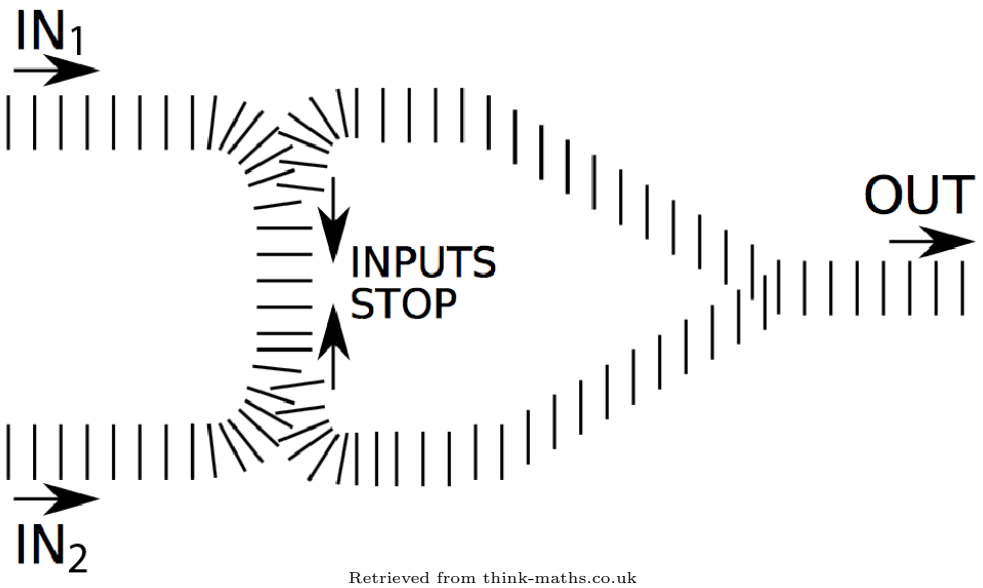
The truth table for XOR:

A	B	$A \veebar B$
True	True	False
True	False	True
False	True	True
False	False	False

Now fill in the table for a domino XOR circuit:

INPUT1	INPUT2	OUTPUT
1	1	0
1	0	1
0	1	1
0	0	0

Thinking Question How could you build a XOR circuit using dominos?



Retrieved from think-maths.co.uk

Binary Numbers: The Basics

A **number system** is the mathematical notation for representing a set of numbers using digits or symbols. The **base** is the number of unique digits, including zero, used to represent numbers. A binary number is a number that is represented by **zeroes** and **ones**. Binary numbers are represented by a **base 2 number system**. All of our electronic devices use binary numbers – including computers.

The **decimal number system** is how we represent numbers in everyday life. It is base **10**.

We can determine the number that a binary number represents by expanding it using powers of 2. First, fill out this table to help us with our calculations.

n	0	1	2	3	4	5	6	7
2^n	$2^0 = 1$	$2^1 = 2$	$2^2 = 4$	$2^3 = 8$	$2^4 = 16$	$2^5 = 32$	$2^6 = 64$	$2^7 = 128$

When expanding, multiply each digit by a power of two. The rightmost digit is multiplied by 2 to the exponent of 0 and the exponent is increased by one as you move to the left.

Let's try an example and rewrite the binary number 101101 as a number in base 10.

$(1)(2^5) + (0)(2^4) + (1)(2^3) + (1)(2^2) + (0)(2^1) + (1)(2^0)$

$= 32 + 0 + 8 + 4 + 0 + 1$

$= 45$

Example 3: Convert each of the following binary numbers to decimal form.

(a) 110 $(1)(2^2) + (1)(2^1) + (0)(2^0) = 4 + 2 + 0 = 6$

(b) 00111 $(0)(2^4) + (0)(2^3) + (1)(2^2) + (1)(2^1) + (1)(2^0) = 0 + 0 + 4 + 2 + 1 = 7$

(c) 100001 $(1)(2^5) + (0)(2^4) + (0)(2^3) + (0)(2^2) + (0)(2^1) + (1)(2^0) = 32 + 0 + \cdots + 0 + 1 = 33$

Binary Numbers: Addition

Binary addition follows the same rules of regular addition (using the decimal system) but instead of carrying over a 1 when the values added equal 10, carry over happens when the values added equal 2. So, in the binary system:

$0+0=$ 0 $\quad 0+1=$ 1 $\quad 1+0=$ 1 $\quad 1+1=$ 0, carry over the 1 (i.e. 10)

Example 4: Add the following binary numbers together.

(a) 100 & 110

$$\begin{array}{r} 100 \\ +110 \\ \hline 1010 \end{array}$$

(b) 10001 & 101

$$\begin{array}{r} 10001 \\ +\quad 101 \\ \hline 10110 \end{array}$$

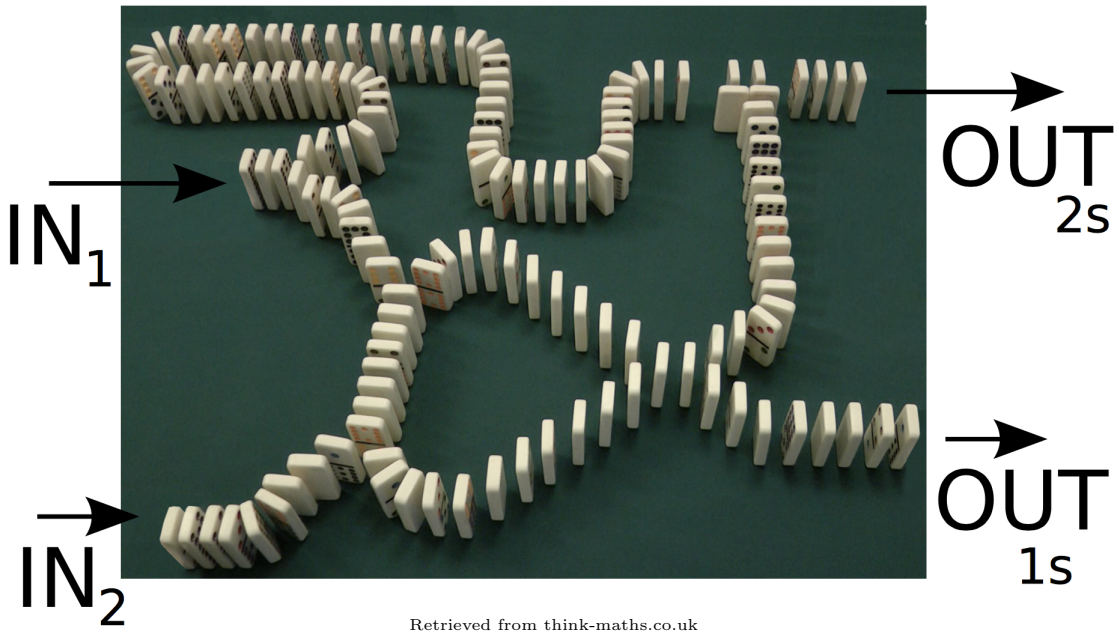
How does the Domino Computer Work?

Let's look at another input table that represents a domino circuit except this time there will be two outputs instead of one.

INPUT1	INPUT2	OUTPUT1	OUTPUT2
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

Notice that output2 acts as a "units" column. That is to say that output2 is the same as taking the results in the column and multiplying it by 2^0 or 1. This gives the result that either one input is knocked over or no inputs are knocked over. Similarly, output1 represents the "twos" column or 2^1. So this column tells us that whether both inputs were knocked over. Thus the columns count the number of inputs that got bumped – but using a binary number system. This system made of dominos can count and thus can work as a computer. The 10,000 Domino Computer seen in the youtube video is this circuit but more complex so it can add binary numbers that have many digits.

This circuit is called a **binary half adder**. It is called this because it does half of what you need in order to do full binary addition.



Retrieved from think-maths.co.uk

Thinking Question: Reflection

What are some possible problems that could occur when a domino computer adds two numbers?

- Some dominos might not be lined up correctly so some circuits don't function as they should.
- The signal (path of falling dominos) might bleed and create phantom signals that weren't meant to happen.
- The timing could be off. The parts that are meant to delay a signal might be too long or too short.

Problem Set

1. Recall that an argument can be valid but not sound. Soundness depends on whether or not the premises are actually true. An argument can follow logically from untrue premises. So you can make logical arguments that are not true. Here are three arguments: one is **invalid and not sound**, one is **valid but not sound**, and the other is **valid and sound**. Determine which argument is which out of the **three options**.

   - All Canadian residents over 18 have the right to vote in an election.
     You are in grade 8 and are a Canadian resident.
     Therefore, you have the right to vote.
     Not valid and not sound.
   - All whole numbers greater than 1 have a prime factorization.
     1960 is a whole number.
     Therefore, it has a prime factorization and it is $2 \times 2 \times 2 \times 5 \times 7 \times 7$.
     Valid and sound.
   - All animals with wings can fly.
     Penguins have wings.
     Therefore, penguins can fly.
     Valid but not sound.

2. Recall that logical operators work in a similar way to how $+, -, \times, \div$ are used. Find out what these statements equal (it will either be **True** or **False**). You may want to review the symbols being used and what they mean.

   (a) $(True \wedge False) \vee True =$ True

   (b) $(\neg(True \uparrow False)) \downarrow (False \text{ XNOR } True) =$ True

   (c) $\neg((False \veebar True) \veebar True) =$ True

   (d) $(\neg(True \vee True)) \wedge False =$ False

3. I have three boxes that I know are labelled incorrectly. Their labels are "lego", "megablocks", and "lego and megablocks". If you can only open one of them, which box will you open to guarantee that you know how to label the boxes correctly?
   Open the lego and megablocks box because you know every box is labelled incorrectly so if lego is in that box, you know the lego label goes on it, the megablocks label goes

on the lego box and the lego and megablocks label goes on the megablocks box. Or, if megablocks are in the box you open, you know the megablocks label goes on it, the lego label goes on the megablocks box, and the lego and megablocks label goes on the lego box.

4. You have two jugs, one is three litres and the other five; how can you use these to measure four litres? Note: You need four litres exactly and each jug has no markings for any measurement other than three or five litres.
   Pour the 5 litre jug full and empty 3 litres into the 3L-jug. The 5L-jug now contains 2L. Empty the 3L-jug and pour the 2L into it. Fill the 5L jug. Now the 3L jug has 1L of empty space, so fill the 3L-jug with 1L from the 5L-jug. Now the 5L-jug contains 4L.

5. Prove De Morgan's Laws.

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

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

   Since the columns representing $\neg(A \wedge B)$ and $(\neg A) \vee (\neg B)$ are identical for the corresponding truth values of the components, we can conclude that $\neg(A \wedge B) \equiv (\neg A) \vee (\neg B)$.

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

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

   Since the columns representing $\neg(A \vee B)$ and $(\neg A) \wedge (\neg B)$ are identical for the corresponding truth values of the components, we can conclude that $\neg(A \vee B) \equiv (\neg A) \wedge (\neg B)$.

6. Convert each of the following binary numbers to decimal form (a number in base 10).

   (a) 011101 $(1)(2^4) + (1)(2^3) + (1)(2^2) + (0)(2^1) + (1)(2^0) = 16 + 8 + 4 + 0 + 1 = 29$

   (b) 1100100 $(1)(2^6)+(1)(2^5)+\ldots+(1)(2^2)+(0)(2^1)+(0)(2^0) = 64+32+\ldots+4+0+0 = 100$

   (c) 10000000 $(1)(2^7) + (0)(2^6) + \ldots + (0)(2^0) = 128 + 0 + \ldots + 0 = 128$

7. Add the following binary numbers.

   (a) 101010 & 01001

   $$\begin{array}{r} 1\,0\,1\,0\,1\,0 \\ +\quad 0\,1\,0\,0\,1 \\ \hline 1\,1\,0\,0\,1\,1 \end{array}$$

   (b) 1011 & 101

   $$\begin{array}{r} 1\,0\,1\,1 \\ +\quad 1\,0\,1 \\ \hline 1\,0\,0\,0\,0 \end{array}$$

8. * Add together the binary numbers 110111 and 011011.

   $$\begin{array}{r} 1\,1\,0\,1\,1\,1 \\ +0\,1\,1\,0\,1\,1 \\ \hline 1\,0\,1\,0\,0\,1\,0 \end{array}$$

9. * Complete the table to represent a **binary full adder**. Use your knowledge from the lesson about the binary half adder and binary addition.

INPUT1	INPUT2	INPUT3	OUTPUT1	OUTPUT2
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

For an explanation about the answers in this table, watch Matt Parker's interview on Numberphile at `https://youtu.be/lNuPy-r1GuQ?t=724`

10. ** Convert the following decimal (base 10) numbers into binary.
The full solution is given for (a). (b) and (c) are converted using the same technique.

(a) 24

Since we are working in a base 2 number system, we divide by 2 until our division results in 0.

Division	Remainder
$24 \div 2 = 12$	0
$12 \div 2 = 6$	0
$6 \div 2 = 3$	0
$3 \div 2 = 1$	1
$1 \div 2 = 0$	1

If we read the remainder column from the bottom to the top we get the answer that the decimal number 24 is converted to the binary number 11000.

(b) 13 1101

(c) 51 110011