Joy of logic

Man, wolf, sheep and cabbage
3-Hats Problem
Logic Gates and Binary Numbers
Turing Machine and computers

Ravi Raizada PhD, FBCS

This "mini-book" aims to introduce the complex subject of Logic to 10-16 year olds, to get them interested and provide enough tools to explore further. It includes:

"All men are liars" (logical conundrum)
The Turing Machine concept
Boolean Logic and modern world
ASCII and other computer codes
Boolean Algebra and Binary Arithmetic
How Electronic Logic Gates work
Logic gates add binary numbers
3-Hats Problem (logical puzzle)
Man with Wolf, Sheep and Cabbage (logical puzzle)

Ravi Raizada **www.JoyOfLearning.com**

JOY OF LOGIC

If you saw a man holding a placard reading, **"All men are liars"**, Fig. 1, would you think that:

(a) As it says so on the placard, this man too is a liar.
(b) If he is a liar, the message on the placard is not true?
(c) He is a teacher in the local school trying to recruit students for his course on logic?

The above is a typical example of a logical conundrum. The same placard held by a woman would be a different proposition! **Can you believe that you can use formal logic to resolve such ambiguities?**

Fig. 1 – All Men are liars

Formal logic is not new, and was invented in China, India and Greece a long time ago. **Panini** (5th Century BC) in India established rules of logic (very similar to Boolean Logic, see later) for Sanskrit grammar. However, **Aristotle**, the Greek Philosopher (322 BC) is credited as the father of Western logic. He used logic to avoid false conclusions from otherwise valid assumptions. Aristotle's theory of **syllogism**, and **propositional calculus** had enormous influence on Western thought. Later on, philosophers like **Frege, Hegel, Godel, Russell, Whitehead** and others made important additions to logic and the basis of mathematics itself.

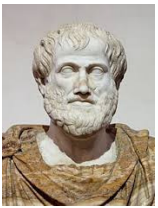


Fig. 2 – Aristotle

In the 19th century, with **George Boole** (1814-1864) and **Augustus De Morgan** (1806-1871), logic entered a new era. They spoke of

solving logical problems using mechanical means and invented **Boolean** algebra.

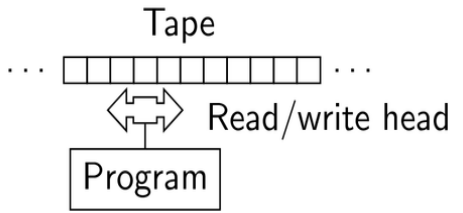


Fig. 3 The Turing Machine

Later on, **Alan Turing** (1912-1954) proposed a theoretical device to manipulate symbols on a strip of tape according to a table of rules. Claude Shannon and John von Neumann proposed similar ideas in the USA at about the same time. The **Turing Machine** became the "model" for modern computing. However, it was not till the

invention of the transistor by **Shockley** et al (Bell Labs, 1947), followed by the invention of the microprocessor that Boolean logic came to dominate many aspects of our lives – from the humble kitchen dishwasher, to computers in aerospace, banking, the Internet, mobile phones and Tablet computers. The **Microprocessor** (Intel 1968) is perhaps the most important invention of the 20th century, along with the **Internet, laser and mobile phone.**

Fig. 4 – Transistor, Microprocessor and Mobile Phone

If you continue your quest, you will soon discover the **Joy of Logic** and learn how it can help you resolve some seemingly complex problems. Logic can be both fun and immensely useful.

Boolean Logic and the modern world

Propositional Calculus, based on logical variables of **Conjunction** (^), **Disjunction** (v) and **Negation** (¬) as used by philosophers, soon gave way to the practicalities offered by **Boolean Logic**. Using **Boolean algebra**, one could ***solve complex logical problems using the simple rules of Algebra!*** Boolean logic (and algebra) uses just two variables – **True or False. Conjunction** is replaced by **AND** (.), **Disjunction** by **OR** (+), **Negation** by **NOT** (-). Logical **NOT** is represented by a line drawn over the variable. The commonly used

logical function of **EX-OR** (XOR or Exclusive-OR) appears to have no classical equivalent and **Implies** is not used in Boolean Logic.

So, how can computers use Boolean Logic to add numbers, print cheques and display complex graphics on the LCD screen in our mobile phones and laptops? All of this manufactured cheaply and to work at high speeds?

ASCII and other computer codes

The modern world has to deal with vast amount of numerical data, a plethora of written and spoken languages, pictures and graphics. ***Just how are two "binary" variables of 1 and 0 adequate to deal with such a complex world?*** Actually, by using a sequence of 0's and 1's, we can deal with large numbers and all sorts of characters. The

ASCII code is now universally used to represent numbers and letters (thus 17 = 0001 0001, letter A = 0100 0001) in the 8-bit binary notation. However, **Unicode** (UTF-8 and UTF-16), the "extended" coding system (which includes ASCII as a subset), is used to represent characters in non-European languages and some commonly used graphical symbols (e.g. £, $ etc). .

Boolean Algebra

Boolean algebra uses **"operators"** for addition (+), multiplication (. dot) and parenthesis as in school algebra. The only additional operator used is the **"Complement"** or **"NOT"** donated by a short line drawn above the variable, Thus:

$A + \overline{A} = A$ **A .A = A** **A + B = B + A**
1 + 1= 1 **1 . 1 = 1** **1 + 0 = 1** etc.

Parenthesis can be used to group similar items together, such as:

A.(C+D)= AC + AD **B.C + C.D = C.(B + D)**

.

Some of the rules above may seem a bit "weird" to you (e.g. 1 + 1 = 1), but you will soon see how we can use them to simplify complex logical function.

So, why do computers use binary numbers at all? The reason is actually very simple. It is practically impossible to design reliable electronic circuits to add, subtract, multiply and divide decimal

numbers, where ***as it is comparatively easier to perform binary arithmetic using cheap electronic "gates" as we shall see later***.

Lets us now look at decimal numbers from 0 to 15 using the **"unsigned"** 4-bit binary system.

:

Decimal	**Binary**	**Decimal**	**Binary**
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010 (A)
3	0011	11	1011 (B)
4	0100	12	1100 (C)
5	0101	13	1101 (D)
6	0110	14	1110 (E)
7	0111	15	1111 (F)

You may notice that we have used letters A, B, C, D, E and F next to numbers greater than 9. This is actually the **"hex"** notation used in computers to read long sequence of binary numbers. For example:

Binary	**Hex**	**Decimal**
0100 0011 1110 1111 (4) (3) (E) (F)	43EF	?

Each bit in the binary number is **"weighted".** The **"least significant bit"** to the right has a "weight" of 1, the next bit to the left is 2, and so on. Thus 1010 (hex A) is equivalent to decimal 12 and 0010 0110 (hex 26) is decimal 36.

Can you work out the decimal equivalent of 43EF hex? .

Binary Arithmetic

Rules for binary addition are very simple:

0 plus 0 equals 0	0 plus 1 equal 1
1 plus 0 equals 1	1 plus 1 equals 0 ***with a carry of 1***

Subtraction, multiplication and division are also performed using similar rules. We can add two binary numbers using rules above:

Decimal 3	0000 0011	(hex 03)
Decimal 5	0000 0101	(hex 04)

Sum is	0000 1000 (Decimal 8)	

How electronic Logic Gates work?

A logic gate is either **"open"** or **"shut"** (never half open). Just like farm gates, ***if the gate is "open" it lets the logic (animal) through.*** Inputs to logic gates themselves can be outputs from other logic gates, forming a ***network of fast-acting "processors" which respond at lightening speed***.

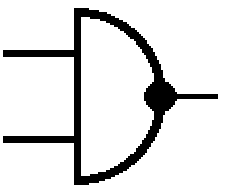


Fig. 5 – 2-input Nand

One of the **"basic"** electronic logic gates is the **NAND** gate, (Fig. 5). You can build the most complex electronic computer using just NAND gates and the earliest electronic computers did just that. ***NAND gates can be built using a single transistor, a couple of resistors and diodes and powered by***

batteries (3 x HP2= 4.5v). **Logic 0** is nearly **0V** *(less than 0.4V)* and **Logic 1** is nearly **at the power supply level** *(always over 2.4v).* The 2-Input **NAND** gate shown in Fig.5.has two inputs (on the left) and one output (on the right). **The output (T) is at Logic 0 only when *both* inputs (A and B) are simultaneously at Logic 1:**

$$\overline{T} = A \cdot B$$ (Boolean expression for **NAND**)

A	**B**	**T**
0	0	1
0	1	1
1	0	1
1	1	0

The line on top of a logic variable means that it is **False** (logic 0). We can also write the above as a **"Truth Tables"** as shown on the left, Fig. 6, for the two input **NAND** gate.

Fig. 6 – Truth Table for the 2-input NAND Gate

More complex logic gates, and sub-systems are now available at low prices and are widely used in computers. Integrated circuits, like the DIL (dual-in-line) chip shown on the left, can be identified only by the numbers stamped on it. The 7400 Series TTL logic integrated circuits are very easy to use and work happily with just 3xHP2 batteries (4.5 volts).

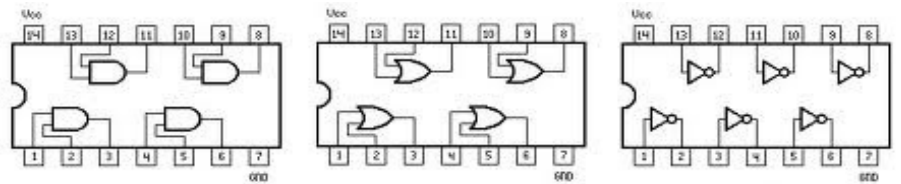


Fig.7 – TTL 7408. 7432 and 7404 Integrated Circuits

Digital logic **"integrated"** circuits are usually supplied in multiple packs as **14-pin** or **16-pin DIL** chips, fig. 7. ***Beginners should avoid CMOS circuits as they are more difficult to use and are easily damaged***. Larger and more complex integrated circuits are supplied as 18, 20, 22, 24 or 40 pin DIL.

Some microprocessor chips, Fig. 8, can have several million logic gates and a DIL package is not suitable. The active part of the microprocessor **"wafer"** is usually just a **few millimetres** across. However, it can have more than a hundred output pins. Special sockets are often used to wire microprocessors

Fig. 8 – The Microprocessor can have hundreds of outputs

on to complex **"motherboards"** consisting of multi-layer printed circuit boards. The **"hardware"** part of the computer requires *capital-intensive* industries, like Intel, and is beyond the capabilities of most developing countries. China and India are, however, making inroads into this industry now.

Computers need something even more important, the software, to make it work. Computers would be useless without the software to control orderly switching of millions of logic gates. Microsoft, Oracle and Google are some well-known software companies. ***Software companies, on the other hand, require little capital*** (just a lot of brain-power) and you can start your own software company with just a laptop. You never know you may be tomorrow's Google or Microsoft!!

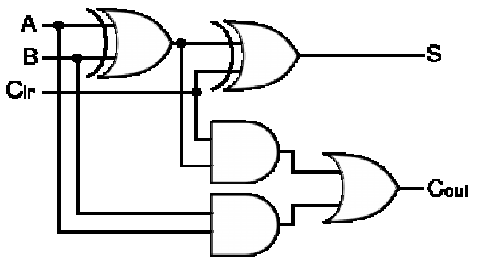
Logic Gates add binary numbers

We will now look at logic and logic gates in a little more detail.

Binary digits, 0 and 1, are also sometimes called **"bits**" and a group of **8 binary bits** is called a **"Byte"** e.g. 1010 1101 (hex AD) is a byte. However, sometime "bits" and "bytes" can be confusing. For example, most **Windows 8** computers use **"64-bit"** data, but their hard disks are measured in **GigaBytes** (GB). ***A gigabyte is one million million bytes, or 8 million million bits.*** For the moment, we will, however, confine ourselves to a very small, 1-bit, computer to understand how logic gates add numbers. The Logic circuit shown on the next page uses just two logic gates to perform addition of two **"bits"** and is often called a **"Half Adder".** Boolean logic equations and the Truth Table for **"Half Adder"** are given on the next page.

Truth Table for Half Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	0



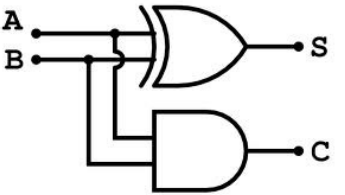
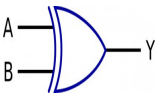


Fig. 9 – Half Adder (above)

A and **B** as the bits to be added, **S** is the **"sum"** and **C** is the **"Carry":**

$$S = A.\bar{B} + \bar{A}.B$$

$$C = A.B$$



The **"Half Adder", Fig. 9,** uses just two logic gates, the **Ex-OR** (Exclusive OR) and **AND.** The **EX-OR g**ate, shown on the left, **has its output at logic 1 only when inputs are dissimilar.** The **Ex-OR** gate is widely used in **"encoding"** and **"decoding"** in computers using a **"key".** The **NAND** gate, with a single input, simply **"inverts"** the logic input and is called a **NOT** gate. If the output of a **NAND** gate is passed through a **NOT** gate, it becomes an **AND** gate.

Fig. 10 - NOT and AND Gates

The more elaborate **"Full Adder"** Circuit on the page 18 also uses the **"carry-in"** from the previous stage. You can try to **"patch"** these circuits yourself on a breadboard (Fig. 11) or a Logic Tutor (Fig. 12). To patch the circuits on the breadboard, you should buy a

handful of inexpensive **TTL Digital ICs** including **7400, 7404, 7410, 7408, 7432 and the Dual JK Flip-flop 7476.**

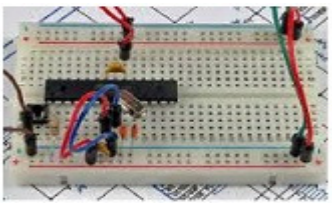


Fig. 11- Breadboard

Fig. 12 – Tutorkit Logic Tutor LT1

You can usually download the **"pin out"** data for each of the chips, from the Internet or find it in books on digital logic. You also need 3 x HP2 batteries (in series) to be used as your power supply.

The 3-Hats Problem

Now that we know a bit of logic, let us have some fun solving logical puzzles. There are many variations of this puzzle and here is one:

Three men have to choose a hat inside a dark closet from **three blue** and **two red** hats, and place it unseen upon their heads. Once outside the closet, the first man looks at the other two and says **"I can't tell what colour my hat is".** The second man too says **"I can't tell what colour my hat is"**. The third man, who can hear but is blind, says **"I know I am wearing a blue hat"**

How does the blind man work it out?

We can solve the riddle by making a **"Truth Table"** and considering each reply in turn. Let's label the man as **A**, **B** and **C.** It doesn't really matter if **C** is blind or not, he just has to be clever!

	A	B	C
1*	Blue	Red	Red
2	Blue	Red	Blue
3*	Blue	Blue	Red
4	Blue	Blue	Blue
5*	Red	Red	Red
6	Red	Red	Blue
7*	Red	Blue	Red
8	Red	Blue	Blue

All 8 combinations of Red and Blue

Line 5 is false as there are **two Red** hats.

Line 1 is false as **A** would have known immediately that he is wearing **Blue**.

Line 7 is false as B would have seen the other two red caps and would know he is wearing Blue. In line 7 (which is false), C is wearing Red, thus also eliminating line 3.

In all remaining "valid" lines, 2, 4, 6 and 8, C is wearing Blue.

Man with wolf, sheep and cabbage

This problem too has many variations. A man with a small rowing boat at his disposal is in charge of a strange cargo of wolf, sheep and cabbage. Everything is fine as long as he is supervising the flock. He has to ferry them across safely to the other side of a river. The boat is too small to carry more than one item at a time. Obviously, he can't leave the wolf and sheep together, or the sheep and cabbage together, without him. ***How does he transfer his cargo across the river safely?***

Trip 1 - Takes the sheep and leaves it on the other side, and rows back alone. *This is a temporary move.*

Trip 2 - Takes wolf over, leaves **wolf on the other side** and rows back with the sheep.
Trip 3 - Takes cabbage over, leaves the **cabbage on the other si**de with the wolf and rows back alone.
Trip 4 – Rows to the **other side with sheep**.
Boolean Logic equations for this problem are as follows:

$\overline{\mathbf{T}} = \mathbf{W.S + S.C}$ (not allowed conditions)
$\mathbf{T = W.C.}$ (OK, allowed)

Can you combine these Boolean functions and simulate on the Logic Tutor or breadboard using digital logic gates?

Inexpensive breadboards and digital logic integrated circuits can be purchased from most electronic suppliers (RS Components, Rapid Electronics, Farnell etc). LT1 and other electronics tutors and components can be purchased from Tutorkit Products, Wrexham, UK