ORMC Beginners 2: Logic Gates Continued

Hongyu (Tesla) Zhu, Andy Shen

April 15, 2022

1 Introduction: Simple Logic Gates, Truth Tables

In the packet, we will continue our study of logic gates. We've already covered the AND, OR, and NOT operations in the previous handout, secretly in the form of gates. But what differentiates operations and logical gates?

Logic gates are tiny gadgets that can manipulate electronic signals. By encoding Boolean (true or false) values as signals, we can directly manipulate logic using gates. For this reason, you can find logic gates in most electronic devices in our daily life, including (most importantly) the computer.

Logic gates operate on electronic signals, specifically bits of inputs. What is a bit?

Definition (bit)**.** A bit is a binary digit, i.e. a digit that takes the value of either 0 or 1.

There are many ways you can think about a bit.

Interpretations/Examples of a bit.

1. A bit can be thought of as a switch: 1 means it is turned on, and 0 means off.
2. A bit can be the truth value of a statement (for example, "Today is sunny"), with 0 meaning false (so today is not sunny) and 1 meaning true (so today is indeed sunny).
3. A bit can be an electronic signal, with 0 meaning low voltage (0 volts) and 1 meaning high voltage (say, 5 volts). This interpretation of bit is what is being used in computers.

Problem 1.1. *Give at least one more example of a bit.*

Now we can (somewhat informally) describe a logic gate.

Definition (logic gate)**.** A logic gate is a "black box" which:

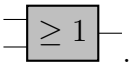
- accepts a certain number of bits as input;
- outputs a certain number of bits;
- has a deterministic rule for manipulating bits.

By deterministic we mean the following: if we feed the logic gate a fixed input (say, 1) and repeat this experiment any number of times, it should always output the same result. In other words, the output of your logic gate can't be random. For example, if you flip a coin and use that as the output (regardless of the input), it will not be a logic gate.

Problem 1.2. *Does the order of inputs matter for an AND gate? What about an OR gate?*

The AND gate is represented by the diagram 

. Some notations for x AND y (with x, y being the two inputs) include $x \wedge y, x\&y$.

The OR gate is represented by the diagram 

. Some notations for x OR y include $x \vee y, x|y$. The ≥ 1 indicates that the OR gate outputs 1 if and only if $x + y \geq 1$.

We can combine logic gates by simply "connecting the wires": if an output of a gate is connected to an input of another one, then we just feed that output of the first gate to that input of the second gate, and let the second gate do its job.

Problem 1.3. *Describe to your neighbor what the following combination of gates does.*

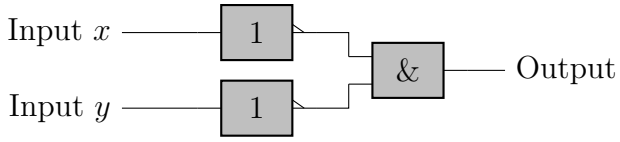


Figure 1: $\neg x \wedge \neg y$

Problem 1.4. *Convince yourself that any combination of gates is still a gate. (If you are confused by the wording, go back to the definition.)*

Definition (truth table)**.** A truth table of a fixed logic gate has two columns: the first column lists all possible inputs of the gate, and the second column lists the corresponding output of the gate.

For example, here is the truth table of the NOT gate.

Input	Output
0	1
1	0

And here is the truth table of the AND gate.

Input	Output
(0,0)	0
(0,1)	0
(1,0)	0
(1,1)	1

If there are exactly two inputs (as is the case with the AND gate), we can also write the truth table in the following form (with x being the first input and y second):

	$x = 0$	$x = 1$
$y = 0$	0	0
$y = 1$	0	1

Problem 1.5. *Write down the truth table for an OR gate. Then write down the truth table for the gate combination in Problem 1.3. What do you discover?*

Usually, we only care about what a logic gate does, regardless of how it does it. This is captured by the following definition.

Definition (logical equivalence)**.** Two logic gates are logically equivalent if and only if they have the same number of input bits, output bits, and the same truth table. We also say that they are equivalent (as a shorthand for logically equivalent).

We consider two logic gates "the same" if they are equivalent. This is in the same spirit as saying two geometric figures are "the same" if they are congruent, etc.

Problem 1.6. *Describe a logic gate with 1 input that "does nothing" to it. (Is it really a logic gate? Don't take my words for granted! Check the definition!)*

Problem 1.7. *Show that the "do nothing" logic gate is equivalent to two NOT gates connected in series.*

Problem 1.8. *Describe (or write down the truth tables) of two logic gates with 0 inputs that are not equivalent to each other. Are there any other gate with 0 inputs up to logical equivalence?*

Definition (Exclusive Or)**.** The exclusive or (XOR) operation takes as input two bits, and outputs 1 if and only if the bits are different. It corresponds to the way we use the phrase "either, or" in English.

For example, if your parents say "We can either go to Disneyland or Universal Studios today.", you won't go to both Disneyland and Universal Studios in the same day. (Unless you happen to be a very smooth talker.) Below is a truth table for the XOR operation.

	$x = 0$	$x = 1$
$y = 0$	0	1
$y = 1$	1	0

Problem 1.9. *Implement an XOR gate using a combination of AND, OR, and NOT gates.*

(Hint: Try to think about the cases in which an exclusive or in plain English is true.)

Definition (Not And)**.** The not and (NAND) operation takes as input two bits, and outputs 0 if and only if both bits are 1. In other words, it is the inverse of the and operation. Below is a truth table for the NAND operation.

	$x = 0$	$x = 1$
$y = 0$	1	1
$y = 1$	1	0

Problem 1.10. *Implement a NOT gate using any number of NAND gates.*

Problem 1.11. *Implement an OR gate using any number of NAND gates.*

Problem 1.12. *Implement an AND gate using any number of NAND gates.*

Definition (Universality)**.** In solving Problem 1.10 - Problem 1.12, we've shown that any logical statement that uses AND, OR, and NOT connectives can be rewritten with just NAND operations.

This property is called universality, or functional completeness. What this means is that any combination of logic gates, however convoluted, can be implemented with just NAND gates.

Problem 1.13. *Create a logic gate using a combination of AND, OR and NOT gates on the left. On the right, redraw the same logic gate using only NAND gates.*

Problem 1.14. *Below is a truth table for a not or (NOR) operation. Similar to the NAND operation, the output of a NOR gate is the inverse of a regular OR gate. Below is a truth table for NOR.*

	$x = 0$	$x = 1$
$y = 0$	1	0
$y = 1$	0	0

Prove that the NOR gate is also universal.

(Hint: How did we show that NAND was universal?)

Problem 1.15. *Recall that a logic gate is uniquely defined by its input and output bits, as well as its truth table. How many unique two input logic gates are there?*

(Hint: Consider using a counting argument and/or writing down all possible truth tables.)

Problem 1.16. *(Challenge) Find 4 other universal logic gates.*