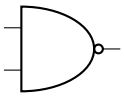
LAMC Intermediate I **Aug 15, 2025**

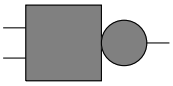
Oleg Gleizer, Brad Lowe
prof1140g@math.ucla.edu, bradlowe@g.ucla.edu

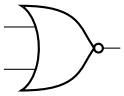
Universal Logic Gates and K-Maps

The following logic gates are called *universal* because all other logic gates can be built using just one of the universal gates.

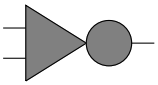


Nand gate





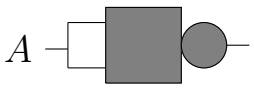
Nor gate

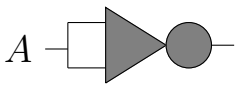


A	B	**Nand**(A,B)
0	0	1
1	0	1
0	1	1
1	1	0

A	B	**Nor**(A,B)
0	0	1
1	0	0
0	1	0
1	1	0

For example, the following circuits function as the **Not** gate.





Problem 1 *Please check the correctness of the statement above.*

Problem 2 *Realize the* **And** *gate as a circuit using the* **Nand** *gate only.*

Problem 3 *Realize the* **Or** *gate as a circuit using the* **Nand** *gate only.*

Aside: Gray Code

Let us look at a new way of counting in binary. The Gray Code, rather than counting up in numerical order like we are used to, only changes one bit (another name for one binary digit) at a time. Take a look:

Numerical

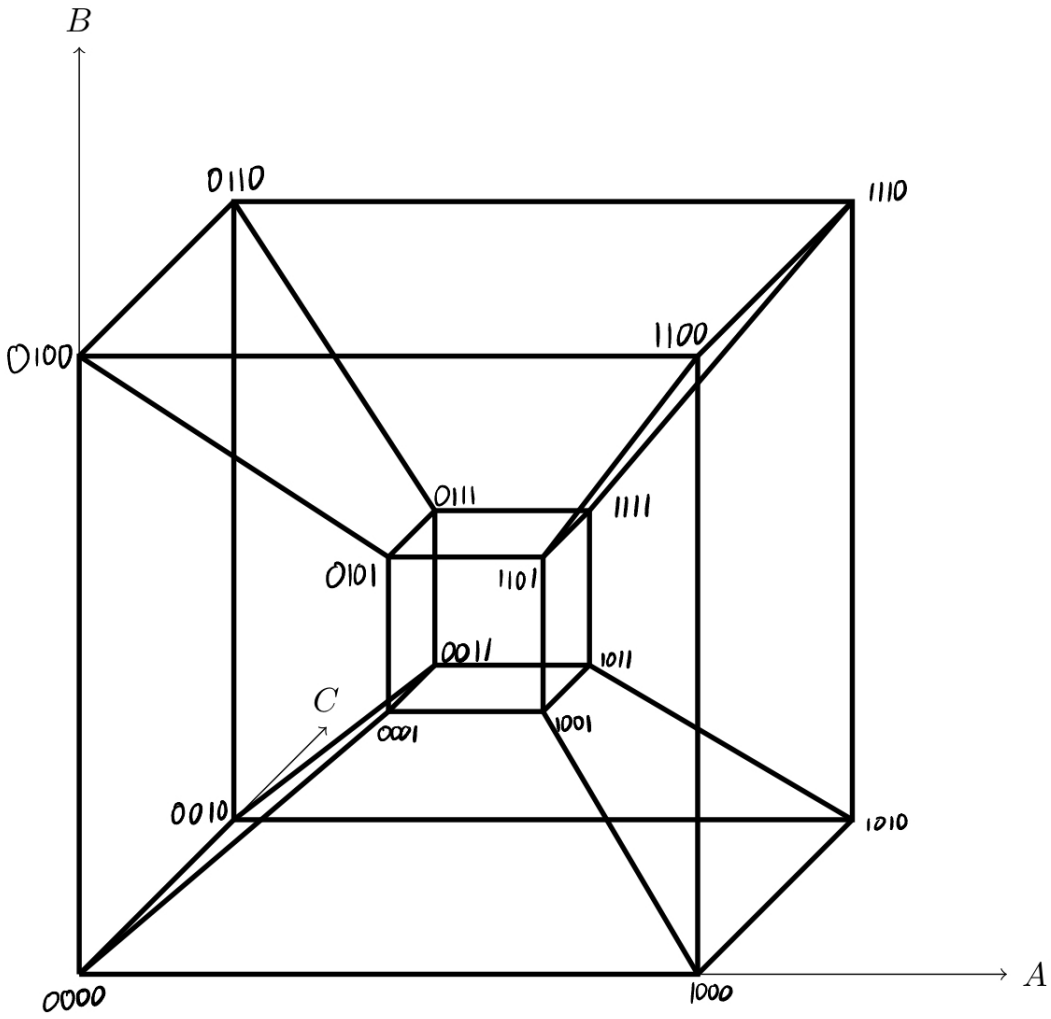
A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Gray Code

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

Problem 4 *Take 2 minutes and think about the Gray Code. When do you think something like this might be useful? Discuss with your group.*

Remember hypercubes? Take a look at this one below, with all vertices marked. Recall that this represents 4 Boolean variables, let's call them ABCD. The D direction is 0 on the outside cube, and 1 on the inside cube. Take a second to look around this hypercube, and note that this follows the Gray Code: Any edges between two vertices differ by only one bit.



Logic Minimization with Karnaugh Maps

We previously learned how to use hypercubes to simplify Boolean expressions. To avoid having to draw a hypercube anytime you want to do this, let's introduce K-Maps.

		BC			
		00	01	11	10
A	0	0	1	1	0
	1	0	0	0	0

How do we read the map? This is essentially just a truth table that's written in a different configuration. The labels on each side show the input to the truth table, and the outputs are written in each tile of the grid. For instance, the K-Map above is the same as the truth table below with X as the output:

A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

A few things to notice: the K-Map must be written in Gray Code, which is why the BC axis is written $00 \to 01 \to 11 \to 10$. This is a requirement because our goal is to find things in common among the inputs to minimize the logic here, just like with the hypercubes previously.

Problem 5 *Write the FDNF form represented by the K-Map on the previous page, and simplify manually.*

		BC			
		00	01	11	10
A	0	0	1	1	0
	1	0	0	0	0

We can see that there is a 1 on the tiles for $\neg A \neg BC$ and $\neg ABC$. We can circle a group of these two 1s on the K-Map. These two terms share $\neg AC$, so that should be your answer for Problem 5.

Try it yourself!

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

AB \ CD	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

Problem 6 *What simplified expressions do the two above K-Maps represent?*

Like with the hypercubes, if there are multiple groups then we just add them together. For instance,

AB \ CD	00	01	11	10
00	0	1	0	0
01	0	1	0	1
11	0	1	0	1
10	1	1	0	0

This K-Map represents $\neg CD + A\neg B\neg C + BC\neg D$.

Problem 7 *By each group above, write which term it corresponds to.*

Problem 8 *Write the simplified expression for the map below. Make sure the groups you select are as big as possible.*

AB \ CD	00	01	11	10
00	1	1	1	1
01	0	1	1	0
11	1	1	1	0
10	0	0	0	0

Problem 9 *Simplify the following expressions by putting them into a K-Map. If you're having trouble, making a truth table first may help.*

- $ABC + A\neg B\neg C + \neg ABC + C$

- $\neg(\neg AB + \neg C) + \neg(\neg A\neg C + B)$

- $AC + AB + \neg A \neg B D + \neg A \neg B C \neg D$

Problem 10 *Create a reduced logic circuit for the following K-Map.*

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	0	0	0
11	0	1	1	0
10	1	0	0	1

Problem 11 *Now, turn this circuit into a circuit using only NAND gates! Hint: Look back at the first few problems.*

Problem 12 *In the previous packet, you came up with a logic circuit to solve whatever problem you wanted. Take that circuit and use a K-Map to simplify it! Was your circuit last time already simplified? Explain why or why not.*