Valerio Mante, Matthew Cook, Benjamin Grewe, Giacomo Indiveri, Daniel Kiper, Wolfger von der Behrens
Lecture 11
Asst: Ioana Calangiu, Raphaela Kreiser, Nicoletta Risi {calangiu@ini.uzh.ch, rkreiser@student.ethz.ch, risi@ini.uzh.ch}

# Exercise 11.1: The McCulloch-Pitts Neuron

The first computational models for artificial neurons were proposed by McCulloch and Pitts in 1943. One of these models (the "relative inhibition" one) is still used today, and it is often referred to now as the "McCulloch-Pitts" neuron. It is also referred to in the circuit complexity literature as a "Linear Threshold" (LT) gate.

The inputs $\mathbf{x} = (x_1, \ldots, x_n)$ and the output $y$ have binary values (0 or 1).

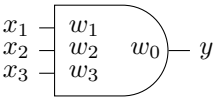


Figure 1: Example McCulloch-Pitts neuron with three inputs.

The output of such a neuron is determined by multiplying each input $x_i$ with the corresponding weight $w_i$ and summing these values over all inputs, and then adding the bias term $w_0$. If the resulting *activation* is positive (or at least 0), the neuron becomes *active* ($y = 1$), otherwise it becomes *inactive* ($y = 0$).

$$y = \begin{cases} 1 & \text{if } \; w_0 + \sum_{i=1}^{n} w_i x_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

1. Find similarities and differences to biological neurons.
2. Characterize the functions that can be realized by a McCulloch-Pitts neuron. For that, consider a neuron with two inputs. From all possible mappings of two binary inputs to one binary output, find two mappings that can be realized and two that cannot be realized. Explain why some mappings cannot be realized.
3. Figure out what the effect of any of the following modifications is. Does it change the set of functions that can be realized with a single neuron (using any weights)?
   (a) Represent inactivity by $-1$ instead of 0. So each $x_i$ is $-1$ or 1, and $y$ should be $-1$ or 1. But the formula for $y$ remains the same (except the 0 on line 2 becomes $-1$).
   (b) Allow only integer values for the weights (i.e., $w_i \in \mathbb{Z}$ for all $i$).
   (c) Set the threshold $w_0$ to always be 0.

# Exercise 11.2: Computing Parity with Linear Threshold (LT) Gates

The circuit in figure 2 was shown in class. The first gate detects whether at least two inputs are 1, and the second gate calculates the sum of the inputs, minus 2 if the sum is 2 or more. Since the sum of the inputs is either 0 or 1 or 2 or 3, the final sum is either 0 or 1 or 0 or 1. The bias of -1 makes the final answer match this final sum. (Why is that?) So the unit outputs 1 when the sum of the inputs is odd, and it outputs 0 when the sum of the units is even. This is exactly the definition of the XOR function.

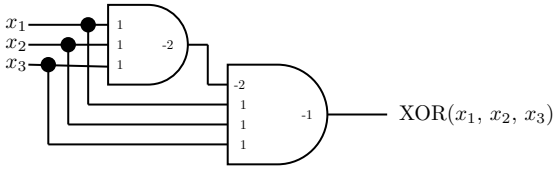


Figure 2: A circuit for XOR of three inputs, as shown in class.

The circuit shown in figure 3 is exactly the same as the circuit in figure 2, but with a more convenient notation for the input wires. The inputs are shown as a vector $\overrightarrow{X}$, and the three wires are shown as a single line with a mark on it. This is a useful notation for *symmetric functions*, that is, functions which give the same

answer when the inputs are rearranged. (AND, OR, and XOR are symmetric functions.) Symmetric functions depend only on the number of 0s and the number of 1s in the input, or in other words, they can be calculated given just the sum of the inputs.

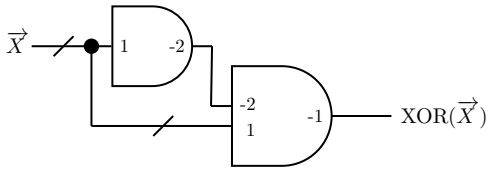


Figure 3: The same circuit as in figure 2, but here the three inputs are shown as the vector $\vec{X}$, and the three wires are shown as a single line with a diagonal mark on it. Where this "wire bundle" goes into a gate, the weight shown is used for every wire in the bundle.

Actually, the circuit in figure 3 is only the same as figure 2 if the length of the vector $\vec{X}$ is 3.

1. Is the circuit in figure 3 a valid circuit for XOR if $\vec{X}$ is just of length 2?
2. Is the circuit in figure 3 a valid circuit for XOR when $\vec{X}$ has length 4?

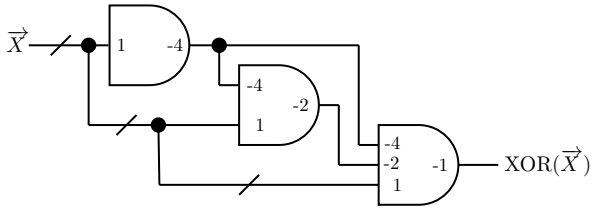


Figure 4: Here is a bigger XOR circuit.

3. How many inputs can the circuit in figure 4 process reliably?

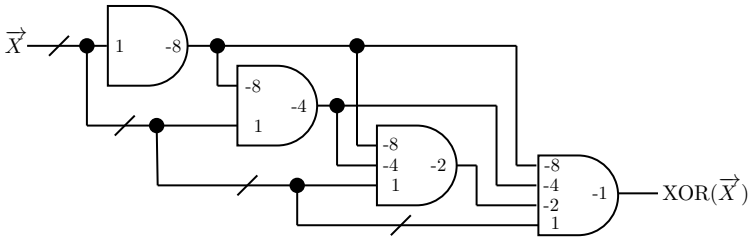


Figure 5: Here is an even bigger XOR circuit.

4. How many inputs can the circuit in figure 5 process reliably?
5. Looking at the circuits in figures 3, 4, and 5, you can see that they follow a pattern. Draw the next circuit in this pattern, and determine how many inputs it can process.
6. Using circuits with this pattern, how many gates are required to compute the XOR of $n$ inputs?
7. Show that (i.e., explain why) for any circuit of AND/OR/NOT (AON) gates that computes the XOR of its inputs, you can add a NOT gate to one input (before it goes to any other gates), and a NOT gate to the output, and it will still compute the XOR function.
8. Show that if you are given a circuit that computes XOR, and you change one of the inputs to always be 0 instead of being an input, then the resulting circuit computes XOR of one fewer inputs.
9. Show that if you change an input to always be 0, then you can simplify the circuit (without affecting its output) so that it has at least one fewer AND or OR gates. (You may need to use the trick from question 7 before reducing the number of gates. Remember, for AON circuit size we only count AND and OR gates.)
10. Use the results of questions 8 and 9 to show that an AON circuit requires at least $n$ gates to process $n$ inputs (when $n > 1$), even when the AND and OR gates are allowed to have any number of inputs.
11. Use the results of questions 6 and 10 to show that LT circuits need exponentially fewer gates than AON circuits, for computing XOR.