Notes 8.370/18.435 Fall 2022

Lecture 8 Prof. Peter Shor

Today we continue discussing classical Boolean circuits.

Quantum mechanics is reversible. Because of this, we will study quantum circuits that are made up from reversible quantum gates. We thus first need to study classical reversible circuits. Reversible circuits are made from reversible gates, both to shed light on reversible quantum circuits, and because we will be using reversible classical circuits as subroutines in our quantum algorithms. Last time we saw that any two-bit reversible gates can be made from NOT, CNOT, and SWAP gates (in fact, we don't need the SWAP gate). While two-bit classical gates are sufficient for universal classical computation, two-bit reversible classical gates are not. We will see this later in this class.

There are, however, three-bit gates that are universal. One of these is the Toffoli gate (discovered by Tom Toffoli, who worked at MIT at the time). This gate negates the third bit if the first two are both 1. It can thus be described as

$$\begin{aligned} x' &= x \\ y' &= y \\ z' &= z \oplus (x \wedge y), \end{aligned}$$

where $\oplus$ is binary addition, or XOR. Its truth table is

variables	
x, y (in)	x', y' (out)
0,0,0	0,0,0
0,1,0	0,1,0
1,0,0	1,0,0
1,1,0	1,1,1
0,0,1	0,0,1
0,1,1	0,1,1
1,0,1	1,0,1
1,1,1	1,1,0

We show that together with the NOT gate, the Toffoli gate is universal for reversible computation. How do we prove this? What we do is that we show that the Toffoli gate can simulate AND, and FANOUT gates. More specifically,

$$\begin{aligned} \text{AND}: \quad T(x, y, 0) &= (0, 0, x \wedge y) \\ \text{FANOUT}: \quad T(1, x, 0) &= (1, x, x) \end{aligned}$$

The OR gate can be composed of AND and NOT gates. You can check that $x \vee y = \neg(\neg x \wedge \neg y)$. Thus, since Toffoli gates can generate AND, OR and FANOUT gates, together with NOT gates, they can be used to perform any computation.

In fact, if there is a source of bits that are set to 1, Toffoli gates can simuate NOT gates as well

$$T(1, 1, x) = T(1, 1, \neg x).$$

But if you don't start with any 1 bits, or with just one 1 bit, Toffoli gates cannot create any new 1 bits. So all we really need the NOT gates for was to generate two 1 bits.

Note that using Toffoli gates as we did above for arbitrary computations generates a lot of extra bits which are not part of the output. For example, to compute an AND, we start with one extra bit initialized to the 0 value, and end up with two extra copies of our input bits that aren't generated by a classical AND gate. Thus, reversible computation is generally less space-efficient that classical computation. What is worse for quantum comptuation, however, is that we have extra "garbage" bits lying around. You don't have enough background to understand this now, but these will destroy interference interactions that are necessary for quantum computation to work. There is a theorem, however, that if you keep the input around, you can reset all these workbits to their initial values (either 0 or 1, say). We now prove this theorem.

Theorem 1 *If there is a classical circuit with a gates taking x_{in} to x_{out}, there is a reversible circuit with $O(a + |x_{out}|)$ gates consisting of NOT gates, CNOT gates, and Toffoli gates taking $(x_{in}, \bar{0}, \bar{0})$ to $(x_{in}, x_{out}, \bar{0})$. Here $\bar{0}$ means a string of 0s of the appropriate legnth.*

Proof:

First, to turn Toffoli gates into AND and NOT gates, we need to initialize some bits to 0 and 1. However, since we are allowing NOT gates, we can use a NOT gate to turn a 0 into a 1, and initialize all our extra workbits to 0.

Now, we can take our classical circuit and, with extra bits initialized to 0, turn all the OR and AND gates into Toffoli gates. This circuit C_1 takes

$$(x_{in}, \bar{0}, \bar{0}) \rightarrow (x_{garbage}, x_{out}, x'_{garbage}),$$

where $x_{garbage}$ and $x'_{garbage}$ are the output bits from the Toffoli gates that we don't use.

Now, let's take the previous circuit, and add a bunch of extra bits initialized to 0 to it. These extra bits should have just enough room to hold the desired output. We can now copy the output to these extra bits. This circuit $C_2 = C_{copy} \circ C_1$ takes

$$(x_{in}, \bar{0}, \bar{0}, \bar{0}) \rightarrow (x_{garbage}, x_{out}, x'_{garbage}, x_{out})$$

But our circuit C_1 is reversible, which means that we can run it backwards and take the output to the input. Doing this on the first three registers of the above state takes $(x_{garbage}, x_{out}, x'_{garbage})$ to $(x_{in}, \bar{0}, \bar{0}, \bar{0})$ Thus, the circuit $C_3 = C_1^{-1} C_{copy} C_1$ takes

$$(x_{in}, \bar{0}, \bar{0}, \bar{0}) \rightarrow (x_{in}, \bar{0}, \bar{0}, x_{out}).$$

We need $|x_{\rm out}|$ FANOUT gates to copy the output, and for each gate in the original circuit, we use a constant number of gates in the reversible circuit. Thus, we have proved our theorem.

The next thing that I did was to show

Theorem 2 *If there is a classical circuit with a gates taking x_{in} to x_{out}, and a classical circuit with b gates taking x_{out} to x_{in}, then there is a reversible circuit with $O(a + b + |x_{in}| + |x_{out}|$ gates that takes*

$$(x_{in}, \bar{0}) \rightarrow (x_{out}, \bar{0})$$

Proof:

This follows from the previous theorem. We know that we have a reversible circuit taking

$$(x_{in}, \bar{0}, \bar{0}) \rightarrow (x_{in}, x_{out}, \bar{0})$$

and one taking

$$(x_{out}, \bar{0}, \bar{0}) \rightarrow (x_{out}, x_{in}, \bar{0}).$$

To get the desired circuit, just apply the first circuit, swap x_{out} and x_{in}, and apply the reverse of the second circuit.

Also note that if you have a reversible circuit taking $(x_{in}, \bar{0}) \rightarrow (x_{out}, \bar{0})$, then this gives a classical circuit taking $x_{in} \rightarrow x_{out}$ and one taking $x_{out} \rightarrow x_{in}$, so we need both parts of our hypothesis.

The last thing I did in lecture was to show that CNOT SWAP, and NOT gates cannot do universal reversible computation. To do this, I proved the theorem

Theorem 3 *Suppose you have a circuit made of CNOT, SWAP, and NOT gates that takes x_1, x_2, x_3, x_4, ..., x_n to y_1, y_2, y_3, y_4, ..., y_n. Then for all y_j, either*

$$y_j = x_{i_1} \oplus x_{i_2} \oplus x_{i_3} \oplus i_1 \ldots \oplus i_k$$

or

$$y_j = x_{i_1} \oplus x_{i_2} \oplus x_{i_3} \oplus i_1 \ldots \oplus i_k \oplus 1$$

Proof Sketch: We prove the theorem by induction. It is easy to see it is true for the CNOT, SWAP, and NOT gates. We thus assume that it is correct for any circuit of at most n gates. Let us consider a circuit containing $n + 1$ gates. The last gate will have input that is of this form. It is fairly straightforward to see that the output is also of this form. For example,

$$NOT(x_{i_1} \oplus x_{i_2} \oplus x_{i_3} \oplus i_1 \ldots \oplus i_k) = x_{i_1} \oplus x_{i_2} \oplus x_{i_3} \oplus i_1 \ldots \oplus i_k \oplus 1$$

and if you take the XOR of two expressions of this form (as you do in a CNOT gate), and variables that are duplicated cancel, and you are left with something of this form. For example,

$$\Big(x_{i_3} \oplus x_{i_5} \oplus x_{i_7} \oplus x_{i_8} \oplus 1\Big) \oplus \Big(x_{i_1} \oplus x_{i_3} \oplus x_{i_7} \oplus 1\Big) = x_{i_1} \oplus x_{i_3} \oplus x_{i_8}.$$

We leave the remainder of the proof, namely, the case of SWAP to the reader.

These functions (which can be composed of CNOT, NOT, and SWAP gates) are called Boolean linear functions, since they can be expressed as

$$y = Mx + b \pmod 2$$

where M is a binary matrix and b is a binary vector.