Reversibility, Quantum circuits

1 Review of quantum circuit model

Recall that in the quantum circuit model we have n qubits that we can manipulate in the following ways:

1. Initialization: The qubits can be initialized to the state $|0^n\rangle$. Preparing a different input state $|x\rangle$, $x \in \{0,1\}^n$ can be done by flipping the required bits.
2. Universal set of gates: Certain sets of one- and two-qubit gates can approximate any constant dimensional unitary transformation sufficiently closely. For example, the CNOT gate together with all one qubit transformations (rotations on the Bloch sphere) forms a universal gate set.
3. Measurement of some (or all) of the qubits output by the quantum circuit. For example if $|\psi\rangle = \sum_x \alpha_x |x\rangle$ and we do a full measurement in the standard/computational basis, then we measure x with probability $|\alpha_x|^2$. If we only measure the first k bits of x, then the probability of measuring $z \in \{0,1\}^k$ is $\sum_{x:z \text{ is a prefix of} x} |\alpha_x|^2$. The resulting partially collapsed quantum state is, up to normalization, $\sum_{x:z \text{ is a prefix of} x} \alpha_x |x\rangle$.
4. Classical postprocessing of the measured value to get the solution to the problem being solved. Quantum computers are expensive and rare (!), so we would probably prefer to use classical processing as much as possible.

The size of a quantum circuit is the number of gates in the circuit. We are interested in finding *efficient* circuits for problems, i.e., circuits for which the total size of the circuit is polynomially bounded in the number of input bits. For example, a family of circuits of size $c \cdot n^5$ is good. But exponentially large – 2^n – circuits are bad.

Why do we consider polynomially-bounded circuits to be good? After all, the constant c in the size bound $c \cdot n^d$ could be extremely large! There are two main reasons:

- Qualitatively, a polynomial-size circuit indicates that we have had a non-trivial insight into the structure of the problem, since it is better than a brute force search.
- Quantitatively it has often been found that a polynomial-time algorithm leads to an algorithm with good constants.

2 Reversible Computation

Quantum evolution is unitary; a quantum circuit corresponds to a unitary operator U acting on kets in $\mathscr{C}^{2^n}$. Being unitary means $UU^\dagger = U^\dagger U = I$, or equivalently that U preserves angles.

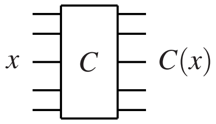
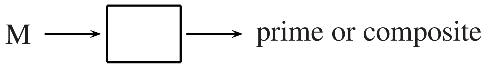


Figure 1: The classical, possibly nonreversible circuit C

Quantum computation originally (in the late 70s and early 80s) studied to understand whether unitary constraint on quantum evolution provided *limits* beyond those explored in classical computation. A unitary transformation taking basis states to basis states must be a permutation. (Indeed, if $U|x\rangle = |u\rangle$ and $U|y\rangle = |u\rangle$, then $|x\rangle = U^{-1}|u\rangle = |y\rangle$.) Therefore quantum mechanics imposes the constraint that classically it must be reversible computation.

If we are given a classical circuit, for example for primality testing,



(notice that the length of the input M is $\lceil \log_2 M \rceil$), can we find a similarly small reversible circuit? Any classical circuit can be built from three basic pieces: the AND gate $\wedge(a,b) = a \cdot b$, the NOT gate $NOT(a) = 1-a$, and fan-out (copying). The NOT gate is a reversible, unitary one-qubit operation. But the AND gate clearly cannot be reversible; since it takes two bits to just one, some information must be lost. (In particular, after applying a NOT gate, we cannot distinguish the cases where the inputs were 00 versus 01 or 10.) The AND gate erases information, which is not reversible. Copying of quantum states is not unitary, but we can copy the (mutually orthogonal) classical basis states, and copying is also reversible.

One way of implementing classical computation is reversible is with an extremely precise, frictionless, billiard ball table, in which the balls collide elastically. Such an algorithm is clearly reversible since reflecting all the balls simultaneously will cause the computation to run backwards. However, it is also unstable; any small error will rapidly grow and the computation will break down. We'll give a less picturesque, but stable method.

There are a number of ways to show construct a reversible circuit corresponding to a nonreversible circuit. For example, one can build the necessary pieces out of a controlled swap gate (Fredkin gate), a CNOT gate, and a NOT gate. Here, we will show how to construct a reversible circuit using the Toffoli gate and the NOT gate. The Toffoli gate is a doubly-controlled NOT gate; it takes (a,b,c) to $(a,b,c+ab \bmod 2)$. (It is its own inverse, so in particular is reversible, and is also unitary.)

We will show how, given a standard circuit C taking input x to $y = C(x)$, we can use the Toffoli and NOT gates to build a reversible circuit $\hat{C}$ taking $(x,0^k)$ to (x,y). Actually, some number of ancilla bits, initialized to 0 will also be useful, so $\hat{C}$ takes $(x,0^k,0^m)$ to $(x,y,0^m)$. (Notice that the m ancilla bits are unchanged...)

- To achieve an AND gate on a and b, we use the Toffoli gate on input $(a,b,0)$. The output is $(a,b,a\wedge b)$, so the third output wire has the result of the AND.
- To copy a bit a, use the Toffoli gate on input $(a,1,0)$. The output is $(a,1,a)$, so we have copied a. Note that this uses both 0 and 1 ancillas; to get a 1 ancilla, we can simply apply a NOT gate to a constant 0 wire.

This method allows us to construct a circuit C' corresponding to C which reversibly takes $(x,0^k,0^m)$ to

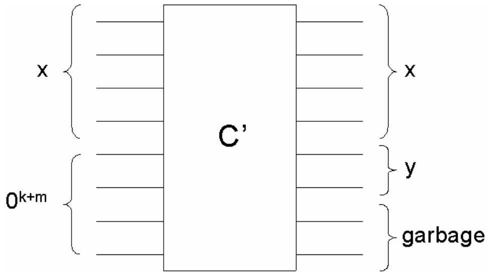


Figure 2: The reversible circuit C’ that produces extra garbage

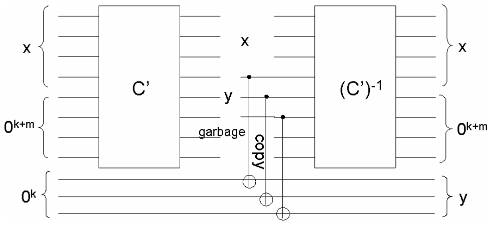


Figure 3: The clean reversible circuit $\hat{C}$ built out of C' and $(C')^{-1}$.

$(x, C(x), \text{garbage}_x)$. The garbage is left over in the original ancilla wires because our operations had extra inputs and outputs. For example to achieve a NOT we had three output wires, whereas the nonreversible NOT gate only has one output wire. The extra two wires are just garbage.

For the purposes of a quantum algorithm, the extra garbage left over is quite significant because it can prevent desirable destructive interference when we run the circuit on states which are not just computational basis states. Fortunately, there is a simple trick for removing all the garbage; we just run the circuit in reverse to erase it! Of course, we don’t want to forget our final answer, so before running C' in reverse we need to copy $y = C(x)$ to some additional ancilla wires. (This copying just uses one additional 1 ancilla, and does not create any further garbage.) The sequence of steps is then

$$(x, 0^k, 0^m, 0^k, 1) \xrightarrow{C'} (x, y, \text{garbage}_x, 0^k, 1) \xrightarrow{\text{copy } y} (x, y, \text{garbage}_x, y, 1) \xrightarrow{(C')^{-1}} (x, 0^k, 0^m, y, 1) \ .$$

Overall, this gives us a clean reversible circuit $\hat{C}$ corresponding to C:

3 Randomized computation

Many important classical algorithms are randomized. For example, primality testing:

For each x, for most choices of r, the circuit computes the correct answer.

To simulate quantumly:

1. First create the corresponding reversible circuit with inputs x, r and ancilla 0’s.
2. To randomize r, feed each $|0\rangle$ qubit wire through a Hadamard gate, giving $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Immediately after applying the Hadamard gate, measure each qubit of r.
3. Instead of measuring the qubits of r, it is sufficient to copy (with CNOT gates) the outputs of the Hadamard gates into fresh qubits. For example, we change $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle$ to $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

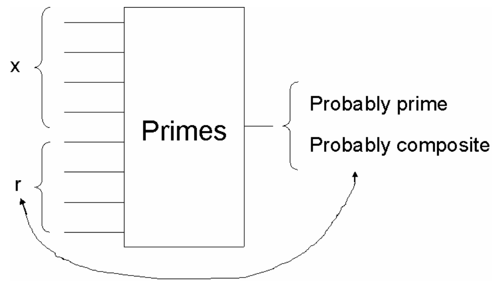


Figure 4: A circuit for primality testing which takes as additional input a random string r.

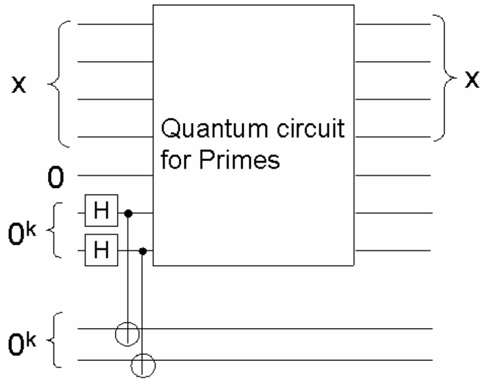


Figure 5: The corresponding quantum circuit; copying with a CNOT the qubits $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is equivalent to measuring them, giving a random string r.

Since they are entangled, measuring the bits into which we copied each computational basis state of r is equivalent to measuring the bits of r itself.

4. In fact, though, it doesn't matter whether we measure the fresh qubits before or after running the quantum circuit. In fact, we can delay their measurement arbitrarily long, or just avoid it altogether. This is known as the "principle of deferred measurement." Measurement is equivalent to entanglement of the system with its environment.