A quick introduction to quantum programming

Jules Jacobs

September 22, 2021

Abstract

This note is a quick introduction to quantum programming in the circuit model. A quantum computer on k bits gets as input a *quantum circuit description*, and produces as output a random string of k bits according to a probability distribution determined by the quantum circuit. A quantum programming language in this model is a language for creating such quantum circuits.

This note contains a quick but formal introduction to these concepts. After reading it, you will be able to write a computer program that simulates such a quantum computer (albeit exponentially more slowly than an actual quantum computer would execute a circuit, which is the point!).

1 Quantum states

Imagine that we have a box with some physical system inside of it, with a finite set S of possible states. A probability distribution over S is a vector $\vec{p}$ of probabilities, one probability $p_x \in [0,1]$ for each state $x \in S$, such that $\sum_x p_x = 1$.

A *quantum state* over S, on the other hand, is a vector $\vec{\phi}$ of *probability amplitudes*, one complex number $\phi_x \in \mathbb{C}$ for each state $x \in S$. If we *measure* such a quantum state, we obtain outcome $x \in S$ with probability $p_x = |\phi_x|^2$. Thus, in order for ϕ to be a proper quantum state, we must have $\sum_x |\phi_x|^2 = 1$.

2 Time evolution in quantum mechanics

Imagine that the system in the box evolves in time according to some laws of physics. In quantum mechanics, the state evolution is given by a matrix U that multiplies the state every time step. If the state is currently ϕ, then at the next time step the state is $U\phi$. If there are $n = |S|$ possible states, then U is an $n \times n$ matrix. Only matrices that preserve the condition that the probabilities sum to 1 are allowed: if $\sum_x |\phi_x|^2 = 1$ we must have $\sum_x |(U\phi)_x|^2 = 1$. Such matrices are called *unitary*.

It might be helpful to compare with probabilistic evolution of the state as in a Markov chain. In that case we model the state with a probability vector $\vec{p}$ and we multiply this vector with a matrix M at each time step. If the state is currently p, then at the next time step the state is Mp. Matrices that preserve the condition that all probabilities are non-negative and that their sum remains 1 are called *stochastic matrices*. The entry $M_{x,y}$ of the matrix is the probability that the system will step to state y, if the state is currently x. Similarly, the entry $U_{x,y}$ of the unitary matrix, is the *probability amplitude* of next state being y, if the state is currently x.

3 What a quantum computer is

A quantum computer with state set S is a device where we can *input* such a matrix U and an initial state ϕ. It will then do one step of time evolution to $\phi' = U\phi$, and it will *measure* the new state

ϕ' and tell us which outcome $x \in S$ it got. This outcome is random, and we will get answer x with probability $|\phi'_x|^2$. Thus, a quantum computer is a kind of universal quantum mechanics simulator:

1. We *input* the initial state ϕ and state evolution matrix U
2. The quantum computer *outputs* answer $x \in S$ with probability $|(U\phi)_x|^2$

We will refine this description in the next section.

4 Quantum circuits

In physics, the state set S is often infinite, and sometimes even uncountably infinite (e.g. the position of a particle), but in quantum programming the set S is taken to be strings of k bits, so that $|S| = 2^k$. Still, U is a 2^k-by-2^k matrix. One might wonder how we even input the U to the quantum computer, if it contains an exponential amount of data.

The answer is that we can't quite input *any* matrix U; it must be encoded as a *quantum circuit*. A quantum circuit is a list of operations we do on the state of n bits, where each operation operates on some small subset of the bits and leaves the rest of the bits alone.

Often, a small set of primitive operations is used, such as the *Hadamard gate* and the *CNOT gate*. The Hadamard gate operates on one bit, and the CNOT gate operates on two bits.

In order to describe what they do, we introduce a bit of notation for *definite states*. We use the notation $\phi = |01001\rangle$ for the definite state ϕ where $\phi_{01001} = 1$ and $\phi_x = 0$ otherwise, *i.e.,* the state that puts all probability amplitude on 01001.

The Hadamard gate H operates on one bit, and is defined as:

$$\mathsf{H}\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$\mathsf{H}\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Equivalently, we can define it using matrix notation, as

$$\mathsf{H} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

If we have n bits in the state, then we have Hadamard gates $\mathsf{H}_1, \mathsf{H}_2, \cdots, \mathsf{H}_n$, each operating on a different bit. This is what H_1 does:

$$\mathsf{H}_1\,|0b_1b_2\cdots b_n\rangle = \frac{1}{\sqrt{2}}(|0b_1b_2\cdots b_n\rangle + |1b_1b_2\cdots b_n\rangle)$$
$$\mathsf{H}_1\,|1b_1b_2\cdots b_n\rangle = \frac{1}{\sqrt{2}}(|0b_1b_2\cdots b_n\rangle - |1b_1b_2\cdots b_n\rangle)$$

Try writing down H_1 as a 2^n-by-2^n matrix, and you'll see why this notation is useful.

The CNOT gate is defined as:

$$\mathsf{CNOT}\,|00\rangle = |00\rangle$$
$$\mathsf{CNOT}\,|01\rangle = |01\rangle$$
$$\mathsf{CNOT}\,|10\rangle = |11\rangle$$
$$\mathsf{CNOT}\,|11\rangle = |10\rangle$$

The CNOT gate implements a classical boolean gate, in the sense that if you input a definite state it also outputs a definite state, but we extend it to superpositions by linearity. In order for the operation to be unitary, all possible states have to appear on the right hand sides, *i.e.,* it wouldn't be valid to have an operation with $M|00\rangle = |00\rangle$ and $M|01\rangle = |00\rangle$, as this wouldn't be unitary.

The CNOT gate flips the second bit if the first bit is 1. Similarly, there is the CCNOT gate, which operates on 3 bits, and flips the third bit if both the first and second bits are 1. Like with the Hadamard gate, if we have n bits we have CNOT_{ij} and CCNOT_{ijk} gates, operating on those bits. The Hadamard and CCNOT gates are a universal set of gates, which means that any unitary 2^n-by-2^n matrix can be arbitrarily closely approximated as a product of the H_i and the CCNOT_{ijk} gates.

Thus, we input the matrix U into the quantum computer as a list of operations, e.g.

$$U = H_1 \cdot \mathsf{CNOT}_{12} \cdot H_2 \cdots H_4$$

The initial state is required to be a definite state $\phi = |x\rangle$.

We arrive at a more refined description of what a quantum computer is:

- Its input is a 2^k-by-2^k matrix U represented compactly as a circuit, and an initial state x.
- Its output is the bit string y with probability $|U_{x,y}|^2$.

5 The Deutsch-Jozsa algorithm

TODO

6 A quantum circuit simulator

1. Hadamard gate
2. Classical f xor gate

References