CS 880: Quantum Information Processing 9/7/10

Lecture 2: From Classical to Quantum Model of Computation

Instructor: Dieter van Melkebeek Scribe: Tyson Williams

Last class we introduced two models for deterministic computation. We discussed Turing Machines, which are models of sequential computation, and then families of uniform circuits, which are models of parallel computation. In both models, we required the operators to be physically realizable and imposed a uniformity condition; namely, that the state transitions could be described by a finite set of rules independent of the input.

In this lecture, we shall develop a model for probabilistic computation, from which our model for quantum computation will follow.

1 Model for Probabilistic Computation

1.1 Overview

Probabilistic computers can use randomness to determine which operations to perform on their inputs. Thus, the state at any given moment and the final output of a computation are both random variables. One way to represent a state $|\psi\rangle$ of dimension m is as a probability distribution over base states, $|s\rangle$ for $s \in \{0,1\}^m$,

$$|\psi\rangle = \sum_{s\in\{0,1\}^m} p_s |s\rangle \qquad 0 \le p_s \le 1, \sum_{s\in\{0,1\}^m} p_s = 1$$

where p_s denotes the probability of observing base state $|s\rangle$. These state vectors have an L_1 norm of 1.

Since the output is now a random variable, we require a computation to provide the correct answer with high probability. That is, given relation R, input x, and output y,

$$(\forall x) \Pr[(x,y) \in R] \ge 1 - \epsilon$$

where ϵ denotes the probability of err. If ϵ were smaller than another bad event, such as the computer crashing during the computation, then we are satisfied. In contrast, $\epsilon = 1/2$ is no good for decision problems, because the algorithm can just flip a fair coin and return the result. If R is a function, then $\epsilon = 1/3$ is good enough because we can rerun the algorithm a polynomial number of times, take the majority answer, and achieve exponentially small error via the Chernoff bound. In fact, any ϵ bounded away from $1/2$ will suffice.

1.2 Local Operations

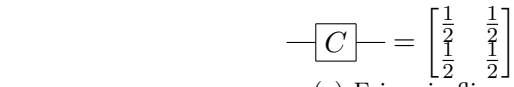
In the probabilistic setting, a transition operator can depend on probabilistic outcomes (i.e., coin flips). Thus, the local effect of a transition operator can be described as the multiplication of a (left) stochastic matrix T with a state vector $|\psi\rangle$,

$$(\forall j) \sum_i T_{ij} = 1 \qquad 0 \le T_{ij} \le 1.$$

We interpret T_{ij} as the probability of entering state i after applying T to state j. As before, the state after an operation is $T|\psi\rangle$ because

$$(|\psi_{\text{after}}\rangle)_i = (T|\psi_{\text{before}}\rangle)_i = \sum_j T_{ij} (|\psi_{\text{before}}\rangle)_j .$$

The matrix for a deterministic operator, which is an all zeros matrix except for a single 1 per column, is just a special case of a stochastic matrix. See Figure 1 for examples of stochastic matrices for a fair coin flip and a biased coin flip.



(a) Fair coin flip (b) Biased coin flip

Figure 1: Coin flip gates

The following exercise shows that, in a strong sense, coin flips are the only genuinely probabilistic operations we need.

Exercise 1. *Given a probabilistic circuit, C, of size t and depth d, there is an equivalent probabilistic circuit C' of size $O(t)$ and depth $O(d)$ such that the first level of C' consists only of biased coin flips and all other levels of C' are deterministic. Here, equivalent means that for any input x the distribution of outputs y is the same for C and C'.*

1.3 Uniformity Condition

We can think of a deterministic Turing Machine as having a Boolean measure of validity associated with every possible transition between configurations. A 1 signifies a completely valid transition, while a 0 denotes a completely invalid transition:

$$\delta : (Q \backslash \{q_{\text{halt}}\} \times \Gamma) \times (Q \times \Gamma \times \{L, P, R\}) \to \{0, 1\}$$

A probabilistic TM will have a probability of validity associated with every transition:

$$\delta_p : (Q \backslash \{q_{\text{halt}}\} \times \Gamma) \times (Q \times \Gamma \times \{L, P, R\}) \to [0, 1]$$

It is important to note that, in order to satisfy the uniformity condition, these probabilities must be easily definable. In particular, we require the n^{th} bit of any bias be computable in time $poly(n)$. If we did not impose this constraint, we could use the probabilities to encode information, such as "0." followed by the characteristic sequence of the halting language. To decide if the n^{th} Turing machine halts, we could repeatedly sample from such a biased coin flip gate in order to estimate p. After we are confident in the value of the n^{th} bit, we return that bit, thereby solving the halting language.

This uniformity condition allows for an infinite number of basic operations. If this is a problem, then we can also consider having just the fair coin flip gate as the only source of randomness. In this case, we would use this gate to get good estimates for any biased coin flips gates that we need. However, we would also have to relax the universality condition. Instead of being required to sample exactly from the distribution of any probabilistic circuit, we would only be required to sample approximately. We will discuss this notion of universality in the next lecture.

1.4 A More Abstract View

We define a *pure state*, $|\psi\rangle$, as a convex combination of base states, $|s\rangle$. That is, $|\psi\rangle = \sum_s p_s |s\rangle$, where p_s is the probability of being in base state $|s\rangle$, $\sum_s p_s = 1$, and $0 \leq p_s \leq 1$. A *mixed state*, is a discrete probability distribution over pure states.

We can think of the probabilistic model as allowing two operations on any pure state $|\psi\rangle$.

1. Local, stochastic transformations, as specified by probabilistic transition matrices. These are L_1 preserving.

2. A terminal observation, which is a probabilistic process that transforms a mixed state into a base state after all transformations have been applied. That is, $|\psi\rangle \rightarrow |s\rangle$, where the probability of achieving $|s\rangle$ is p_s.

Exercise 2. *What happens if we allow observations at any point in time? That is, in between transitions? A motivation, consider the problem of composing two procedures, both of which observe their respective states after their transformations are complete?*

2 Model for Quantum Computation

2.1 Overview

As with the probabilistic model, the state of a system is described by a super-position of base states, but here:

1. the coefficients are complex numbers (usually denoted by α because it stands for an amplitude)

2. vectors have an L_2 norm of 1 (i.e., $\sum_s |\alpha_s|^2 = 1$)

A *qubit* is the quantum analog of a classical bit and satisfies the above two conditions. The interpretation is that $\Pr[\text{observing } |s\rangle] = |\alpha_s|^2$. Note, this is a valid interpretation because the above defines a valid probability distribution.

2.2 Local Operations

For consistency of interpretation, global operations have to preserve the 2-norm. It is necessary and sufficient that local operations are unitary transformations. That is,

$$T^*T = I = TT^*,$$

where T^* is the conjugate transpose[1] of T. Unitary matrices have a full basis of eigenvectors with eigenvalues $|\lambda| = 1$. Since the determinant is the product of the eigenvalues, $|\det| = 1$ as well.

Example: Does the classical "coin-flip" transformation describe a valid quantum gate? No, because its transistion matrix is not unitary. It does not even have full rank. ⊠

[1]The notation T^* for the conjugate transpose is more common in linear algebra while $T^\dagger$ is more common in quantum mechanics.

The quantum analog of a fair coin flip is the Hadamard gate. It is described by the following matrix, which *is* unitary:

$$-\boxed{H}- = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$$

If we apply the Hadamard gate to base states, we get the intuitive "fair coin" result. That is, regardless of which base state we are in, we end up with 50% probability of being in base state $|0\rangle$ and 50% probability of being in base state $|1\rangle$:

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

What if we apply the Hadamard gate to a super-position of base states?

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

Unlike in the probabilistic setting, we do not necessarily get a "fair coin" result. The above is an example of destructive interference, the key ingredient of quantum algorithm design. Quantum algorithms that run faster than their classical counterparts make constructive use of destructive interference, effectively canceling out wrong computation paths.

The tranformation matrix for the quantum analog of a biased coin flip is

$$\begin{bmatrix}\sqrt{p} & \sqrt{p}\\ \sqrt{1-p} & -\sqrt{1-p}\end{bmatrix}.$$

Another prevalent quantum gate is the rotation

$$-\boxed{R_\theta}- = \begin{bmatrix}1 & 0\\ 0 & e^{i\theta}\end{bmatrix},$$

which effectively adds θ to the phase of the 1-component.

Example: Can we use deterministic gates in the quantum setting? Consider the NAND gate. The matrix associated with the NAND gate's transformation is not unitary, as both $|00\rangle$ and $|10\rangle$ map to the same output state, $|10\rangle$. In general, deterministic gates are unitary if and only if they are permutations of base states. That is, if they are reversible. ⊠

An important gate is the CNOT gate, which is shown schematically in Figure 2. The matrix

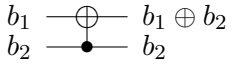


Figure 2: CNOT gate

associated with this transformation is given below:

$$T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

This gate flips its first input bit if the second bit, also known as the control bit, is a 1; otherwise it leaves the first input bit unchanged. Note that if $b_1 = 0$, then the CNOT gate effectively copies b_2.

2.3 Simulating classical gates

Even though classical gates, such as the NAND gate, do not translate directly into the quantum setting, they can be simulated. Given a transformation

$$f : \{0,1\}^* \to \{0,1\},$$

we can define a new transformation

$$\tilde{f} : \{0,1\}^* \times \{0,1\} \to \{0,1\}^* \times \{0,1\} : (x, b) \to (x, b \oplus f(x)).$$

Essentially, $\tilde{f}$ maintains a copy of its input in order to make the transformation reversible. One can perform this transformation on all classical gates.

Example: A reversible NAND gate is shown schematically in Figure 3. The additional third bit, which we need to simulate the classical gate, is called an *ancilla bit.* ⊠

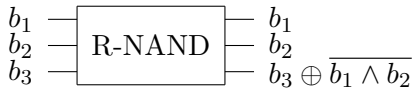


Figure 3: Reversible NAND gate

We can apply the above idea to an entire classical circuit. Sometimes, the "garbage" output due to the ancilla bits is problematic, as it is not defined by the original classical transformation. Specifically, this garbage output will prevent the destructive interference from happening as desired. We can circumvent this difficulty by copying the output of the circuit and then running the circuit in reverse as illustrated in Figure 4.

Theorem 1. *If f can be computed by a deterministic circuit of size t and depth d, then $\tilde{f}$ can be computed by a reversible circuit of size $O(t)$ and depth $O(d)$ using $O(t)$ ancilla bits.*

There are more efficient space usage transformations than specified by the above theorem, but this efficiency comes at the expense of time efficiency. It is an open question whether one can simulate a classical circuit in constant time and constant space relative to the original circuit.

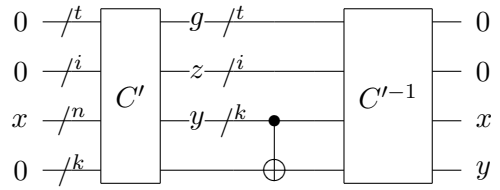


Figure 4: Computation of $\tilde{f}$ for arbitrary classical circuit C.