The Computational Complexity Column

by

Lance FORTNOW

NEC Laboratories America
4 Independence Way, Princeton, NJ 08540, USA
fortnow@nec-labs.com
http://www.neci.nj.nec.com/homepages/fortnow/beatcs

This issue Steve Fenner demystifies quantum computing for those of us without a physics Ph.D.

A Physics-Free Introduction to the Quantum Computation Model

Stephen A. Fenner[1]
Computer Science and Engineering Department
University of South Carolina
Columbia, SC 29208 USA
fenner@cse.sc.edu

Abstract This article defines and proves basic properties of the standard quantum circuit model of computation. The model is developed abstractly in close analogy with (classical) deterministic and probabilistic circuits, without recourse to any physical concepts or principles. It is intended as a primer for theoretical computer scientists who do not know—and perhaps do not care to know—any physics.

1 Why Read This?

As an area of research, quantum computation has attracted considerable attention in the last few years. It has drawn physicists, computer scientists, mathematicians, engineers, and even philosophers together into an ever-widening investigation. The two big questions are (1) can we build a reliable

[1] Partially supported by US Army Research Office DAAD.

large-scale quantum computer? and (2) what could we ultimately do with it if or when we build it? The first question is rightfully the domain of physics and engineering, and can be informed by computer scientific investigations. The second question, however, is more computer scientific in flavor, closer to algorithms and computational complexity.

Unfortunately, the subject of quantum computation is daunting to many computer scientists—the very people who may be best equipped to address the second question, above, and advance the frontier of knowledge in the field. Expositions of quantum computation often use physical concepts to explain such things as qubits (quantum bits), and so tacitly assume some physical background, leading nonphysicists to think that they must learn physics, especially (heaven forbid) quantum mechanics, in order to understand what is going on. The purpose of this article is to show how incorrect this thinking is; one can gain a solid, precise grasp of the standard quantum model of computation—quantum circuits—with no physics background, and without having to learn any physics along the way. (I am not being completely fair to some of the better expositors of the subject of quantum computing, such as Nielsen and Chuang [1], who stress the simple axiomatic nature of the quantum mechanics needed for quantum computation. Yet their book, being much more comprehensive than the current article, gives a good deal of information that is not immediately relevant to a basic grasp of quantum circuits.)

I will introduce quantum circuits using a simple and close analogy with classical (that is, nonquantum) Boolean and probabilistic circuits. The goal is to introduce as few concepts as possible that are foreign to computer science. To these ends, I will first review classical deterministic Boolean circuits. My approach will be nonstandard, but clearly equivalent to the standard approach. I will then add probabilistic, "coin-flip" gates to the model to arrive at the probabilistic circuit model. The coin-flip gate is an example of a nondeterministic gate. The quantum model is obtained by replacing coin-flip gates with a certain other type of nondeterministic gate.

I assume some knowledge on the reader's part of linear algebra, Boolean logic, and computational complexity, such as polynomial time, P, and NP.

1.1 A Few More Remarks

One cannot really split the two big questions above so cleanly into traditional academic disciplines. There has been, and continues to be, much useful collaboration going on between the two realms. The fact that there is a simple, abstract model of quantum computation at all—one that we can divorce from physical considerations—owes much to the foundational work of people in both areas, such as L. Adleman, C. Bennett, E. Bernstein, G. Brassard, J. DeMarrais, D. Deutsch, R. Feynman, M.-D. Huang, U. Vazirani, A. Yao, and many others. Although quantum circuits are currently the preferred way to represent quantum computation, there are other ways, such as quantum Turing machines. Quantum Turing machines and quantum circuits are equivalent for describing quantum computation, with modest overhead for one model to simulate the other. There is a lot of detailed background on these topics which I will not go into here. I suggest looking to Nielsen and Chuang [1] for more information and bibliographic references.

2 Acknowledgments

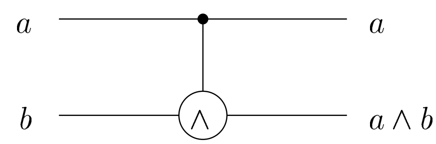
This article grew out of a somewhat impromptu introductory talk I gave at Dagstuhl[2] in the Fall of 2002. I have enjoyed many rewarding encounters and discussions at this and previous Dagstuhl seminars, and I wish to thank the organizers of the seminar, Harry Buhrman, Lance Fortnow, and Thomas Thierauf, for inviting me. Thanks also to the European Community for providing financial assistance to me and the other guests. Finally, I thank Lance Fortnow for suggesting (the night before) that I give a talk along these lines, and for inviting me write it up for BEATCS.

3 Boolean Circuits

Here is a quick review of the Boolean circuit model. Our approach is slightly unorthodox—for reasons that may become clear later—but is clearly equivalent to the traditional approach.

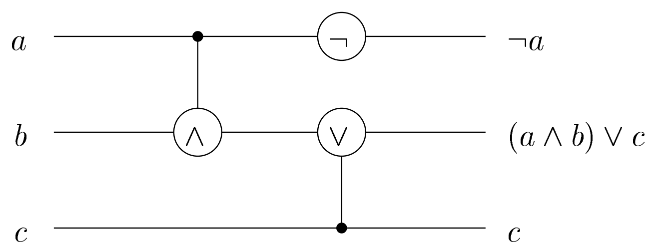
We imagine n registers, each capable of holding a single bit (possible values: 0 for false, or 1 for true). A *Boolean gate* computes some logical operation of some registers and places the result in a register. We label the gate with the logical operation it performs. For the Boolean case, we can restrict our attention to monadic and dyadic gates (i.e., gates operating on one or two bits) that place the result in one of the operand registers. For example, in this diagram,

[2] Schloss Dagstuhl International Conference and Research Center for Computer Science, Seminar 02421, "Algebraic Methods in Quantum and Classical Models of Computation," October 2002.



we have a single gate acting on two registers (the horizontal lines). It computes the logical AND of the two register values, and sets the second (lower) register to the result, leaving the first register unchanged. For this reason, the second bit is called the *target*, and the first bit the *control*.[3] In all our diagrams, we consider time flowing from left to right, so that inputs to the gate appear to the left, and outputs to the right. We consider a gate to be a transformation on all bits it acts on, even though some bits values may not change (e.g., the control).

A *Boolean circuit* is a sequence of gates applied chronologically to the registers. For example, this circuit



yields the values shown on the right, given arbitrary input values $a, b, c \in \{0, 1\}$. It makes no difference whether the NOT gate occurs before or after the OR gate, since they involve different registers. We can thus depict them as acting simultaneously, but if we must choose, we'll say that the NOT gate acts first.

If we label the registers involved in a circuit as $r_1, \ldots, r_n$, then a circuit can also be described as a straight-line program with assignment instructions of the form $r_i := r_i$ op r_j where op is a dyadic Boolean connective, or of the form $r_i := \neg r_i$. The program corresponding to the circuit above is

$$r_2 := r_2 \ \wedge \ r_1$$

$$r_1 := \neg r_1$$

$$r_2 := r_2 \ \vee \ r_3$$

[3] This particular example is not quite in keeping with standard usage of these terms in electrical engineering. There, if the control bit is off, then nothing should happen to the target, which is clearly not the case here.

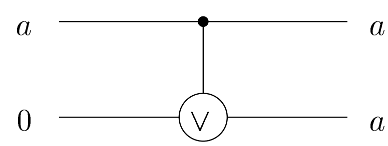
We'll denote the *state* of the registers at any given time by $|\vec{v}\rangle$, where $\vec{v}$ is a vector of n bits, one for each register. There are a total of 2^n possible states. In the circuit above, the initial state is $|a, b, c\rangle$. After the first gate is applied, the state is $|a, (a \wedge b), c\rangle$, and so on. The complete progression of states is

$$|a, b, c\rangle \mapsto |a, (a \wedge b), c\rangle \mapsto |\neg a, (a \wedge b), c\rangle \mapsto |\neg a, ((a \wedge b) \vee c), c\rangle.$$

Thus a circuit describes a mapping of states to states.

3.1 Input and Output

We'll designate the first k registers as *inputs* (for some $0 \leq k \leq n$) and the first ℓ registers as *outputs* (for some $0 \leq \ell \leq n$). Each noninput register is given an initial value either 0 or 1, and this value is considered part of the description of the circuit. Noninput, nonoutput registers are sometimes called *ancillas*. For example, we can use an ancilla to copy a bit:

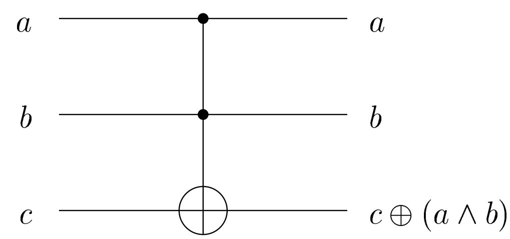


At the end of the circuit, we observe the value in the output registers as the result of the circuit, discarding the nonoutput registers. In this way, a Boolean circuit C computes a function $\{0,1\}^k \to \{0,1\}^\ell$. If $\ell = 1$, then we regard C as recognizing a subset of $\{0,1\}^k$.

A *circuit family* is an infinite sequence $C_0, C_1, C_2, \ldots$ of circuits such that each C_i has exactly i inputs and one output. A circuit family computes a language $L \subseteq \{0,1\}^*$ in the usual way. A circuit family is *ptime uniform* if there is a polynomial-time deterministic computation that outputs (a description of) C_i on input 1^i. Ptime uniform families of Boolean circuits capture the language class P in this sense: a language L is in P if and only if there is a ptime uniform family of Boolean circuits computing L.

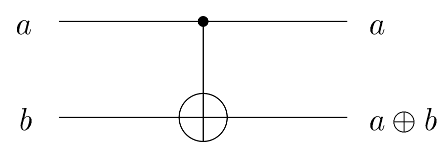
3.2 Reversibility

The AND and OR gates described above won't quite work in the quantum circuit model. To be considered a legitimate quantum gate, the gate must act *reversibly*. No information can be lost from input to output; in other words, the input values of the gate must be recoverable from the output values. Fortunately, using just reversible gates we can do everything we did before with AND, OR, and NOT gates with just a constant factor of overhead. Consider the three-bit *Toffoli gate* with two controls and a target (here, $\oplus$ means exclusive or):



This gates is reversible; in fact, it is its own inverse. Moreover, it is not hard to see (exercise) how the Toffoli gate, along with appropriate ancillas, can simulate the AND and NOT gates and can copy a bit. (If we only allow 0 as an initial ancilla value, then we must also allow the NOT gate. This is no problem, because the NOT gate is reversible.)

Another often-used reversible gate is the *controlled NOT* or CNOT gate

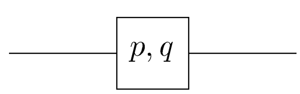


which can be implemented easily using a Toffoli gate and an ancilla.

If we do use one or more ancillas to implement a gate as a subcircuit, we will insist that the ancillas be used *cleanly.* That means that the ancillas end with the same values they started with, regardless of the values of the other registers. Go back and make sure that all your ancillas were used cleanly.

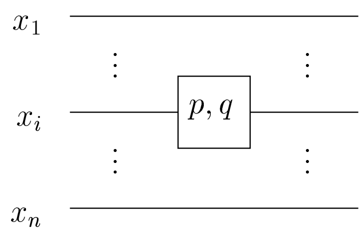
4 Probabilistic Circuits

To implement probabilistic computation with circuits, we need to introduce a new type of gate to our model. For any rational numbers $0 \leq p, q \leq 1$, we will allow a *biased coin-flip gate*



Informally, this gates behaves as follows. If the input register is 0, then a coin with bias p is flipped, and the output register is 0 with probability p and 1 with probability $1 - p$. If the input register is 1, then a coin with bias q is flipped, and the output register is 0 with probability q and 1 with probability $1-q$. One or both biases may be $\frac{1}{2}$.

To keep track of the probabilities, we now need to redefine our notion of state. Assume all 2^n tuples $|x_1, \ldots, x_n\rangle$ form a basis of a real vector space $\mathcal{H}$. That is, $\mathcal{H}$ is the 2^n-dimensional free real vector space over the set of tuples. We call the set of tuples the *computational basis* (the tuples themselves being *basis states*), and we use this basis to identify $\mathcal{H}$ with $\mathbb{R}^{2^n}$. We redefine a *state* to be a certain vector in $\mathcal{H}$—a linear combination (or "superposition") of basis states whose coefficients are probabilities. Then gates will now correspond to linear mappings from $\mathcal{H}$ to $\mathcal{H}$. In particular,



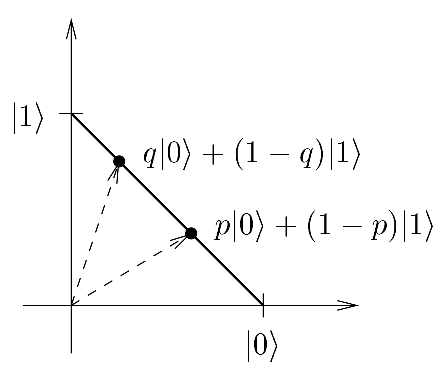
maps the basis state $|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n\rangle$ to the state

$$p|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n\rangle + (1-p)|x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n\rangle,$$

and maps the basis state $|x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n\rangle$ to

$$q|x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n\rangle + (1-q)|x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n\rangle,$$

Note that the values of the bits besides the ith bit are unaffected. Ignoring the other bits for a moment, this gate maps the one-bit basis state $|0\rangle$ to $p|0\rangle + (1-p)|1\rangle$ and likewise maps $|1\rangle$ to $q|0\rangle + (1-q)|1\rangle$. These two resulting states can be described geometrically as the points $(p, 1-p)$ and $(q, 1-q)$ on the line segment connecting $(1,0)$ and $(0,1)$:



In this example, $p = \frac{5}{8}$ and $q = \frac{1}{4}$. The gate always maps this line segment into itself.

We can represent states as a column vectors of probabilities. Then the action of the coin-flip gate on its single bit can be described succinctly by the 2×2 columnwise stochastic[4] matrix

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

We extend the action of each Boolean gate of Section 3 to a linear map on $\mathcal{H}$. Each maps basis states to basis states, so it corresponds to a matrix with entries in $\{0,1\}$. Each column of this matrix has exactly one 1, and so the matrix is also columnwise stochastic. If the gate is reversible, then the corresponding matrix is a permutation matrix. So for example, the (irreversible) AND gate depicted in Section 3 has the matrix

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where we assume that the column vector corresponding to a state always has its coefficients listed in increasing lexicographical order by basis state—in this case, $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$. The Toffoli gate depicted there has the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

A *probabilistic circuit* is one that allows only Boolean gates and biased coin-flip gates. The gates are applied in order from left to right, as before. We require the initial state to be a basis state, corresponding to a particular Boolean input as in Section 3. The final state of the registers is some vector

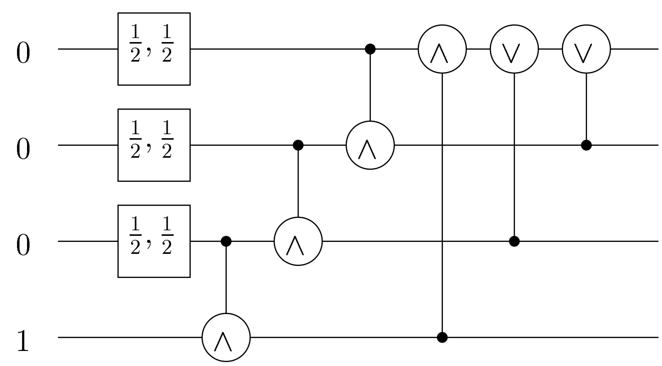
$$|\text{final}\rangle = \sum_{x \in \{0,1\}^n} p_x |x\rangle,$$

[4] A matrix is *columnwise stochastic* if all its entries are nonnegative real, and all columns sum to 1.

where the p_x are real coefficients. Because each gate is stochastic, it preserves the ℓ_1-norm (sum of coefficients) of the state vector, so that the intermediate states and the output state all have unit ℓ_1-norm, and thus $\sum_x |p_x| = 1$. Furthermore, all matrix entries are nonnegative, so $p_x \geq 0$ for all x. We interpret the p_x as probabilities; namely, p_x represents the probability that the registers will be in basis state $|x\rangle$ at the end of the computation. Thus the final state corresponds to a probability distribution of basis states, as we would expect.

Thinking geometrically again for a moment, define the *standard simplex* in $\mathcal{H}$ to be the set of all convex linear combinations of the basis states.[5] This generalizes to m dimensions the line segment shown above, which is the standard 1-dimensional simplex in $\mathbb{R}^2$. A probabilistic circuit corresponds to a linear transformation on $\mathcal{H}$ that maps the standard simplex into itself. The initial state is always a basis state—which is in the simplex—so the final state is also in the simplex.

Here is a simple example of a probabilistic circuit. It has no input registers, but rather computes the majority of three unbiased coin flips.



Recall that the output bit is in the first register. We *observe* the output bit as follows: write $|\text{final}\rangle$ as

$$\sum_{x_2,\dots,x_n} p_{0x_2\cdots x_n}|0, x_2, \dots, x_n\rangle + \sum_{x_2,\dots,x_n} p_{1x_2\cdots x_n}|1, x_2, \dots, x_n\rangle.$$

The *probability of seeing* 0 is then $\sum_{x_2,\dots,x_n} p_{0x_2\cdots x_n}$, and likewise the *probability of seeing* 1 is $\sum_{x_2,\dots,x_n} p_{1x_2\cdots x_n}$. These formulas generalize in the obvious way to the case of more than one output register being observed.

[5] A *convex linear combination* of vectors $v_1, \dots, v_n$ is a vector of the form $\sum_{i=1}^m c_i v_i$, where each $c_i \geq 0$ and $\sum_{i=1}^m c_i = 1$.

4.1 More Complexity Classes

Many well-known complexity classes can be characterized using ptime uniform families of probabilistic circuits and placing a threshold on the probabilities of observing 1 on a given input. Let an *acceptance criterion* be a pair (R, A) of disjoint subsets of the unit interval $[0, 1]$. A ptime uniform probabilistic circuit family $C_0, C_1, \ldots$ with acceptance criterion (R, A) *computes* a language L if, for all $n \geq 0$ and all input strings x of length n, if $x \in L$ then $p \in A$ and if $x \notin L$ then $p \in R$, where p is the probability of seeing 1 on the output bit of C_n when the input is x. Using ptime uniform probabilistic circuits, we get the following correspondences between acceptance criteria and complexity classes:

Class	Acceptance Criterion
P	$(\{0\}, \{1\})$
NP	$(\{0\}, (0, 1])$
RP	$(\{0\}, (\frac{1}{2}, 1])$
BPP	$([0, \frac{1}{3}], [\frac{2}{3}, 1])$
PP	$([0, \frac{1}{2}], (\frac{1}{2}, 1])$

4.2 Robustness

There is no essential reason to allow arbitrary rational $p, q \in [0, 1]$ for our coin-flip gates, at least as far as the above complexity class characterizations are concerned. It is well-known that we could restrict the value of (p, q) to be, say, $(0, \frac{1}{2})$, and the above classes would remain the same. Furthermore, we could restrict the location of coin-flip gates to appear only on the leftmost column of the circuit, being the first gates applied to their respective ancillas, whose initial values are all 1.

We will see similar robustness phenomena when we choose gates for quantum circuits in the next section.

5 Quantum Circuits

We'll define quantum circuits in much the same manner as we defined probabilistic circuits. States are vectors in the real vector space $\mathcal{H}$ as before, and gates correspond to certain linear transformations on $\mathcal{H}$ as before. We only make two seemingly minor changes in the kinds of gates we allow:

1. We drop the restriction that entries in matrices corresponding to gates be nonnegative. We now allow negative entries.

2. Instead of preserving the ℓ_1-norm of state vectors, gates must instead preserve the ℓ_2-norm (i.e., the Euclidean norm) of state vectors.

The ℓ_2-norm of a real vector $(a_1, \ldots, a_m)$ is $\sqrt{a_1^2 + \cdots + a_m^2}$. The linear transformations that preserve the ℓ_2-norm are exactly the ones represented by orthogonal matrices, i.e., matrices M such that $MM^{\mathrm{t}} = M^{\mathrm{t}}M = I$, or equivalently, matrices whose columns form an orthonormal set with respect to the usual inner product on column vectors. (Note that our description of the ℓ_2-norm implicitly makes the computational basis an orthonormal basis.) Because of these two changes, we can no longer interpret coefficients on basis states as probabilities—a problem we'll fix shortly.

We now call the registers *qubits* (quantum bits) instead of bits.

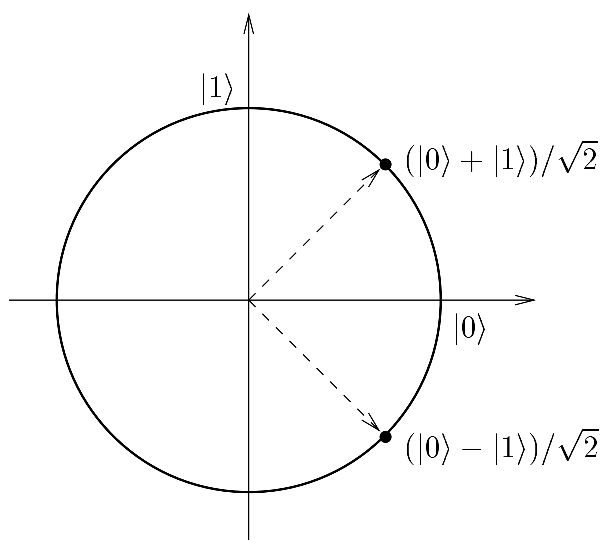
A simple and very useful quantum gate is the one-qubit *Hadamard* gate, denoted by H:

$$\text{———}\boxed{H}\text{———}$$

Its matrix is

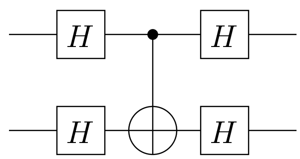
$$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

This gate maps the one-bit basis state $|b\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + (-1)^b|1\rangle)$, for $b \in \{0,1\}$. The two possible resulting states can be described geometrically as the following points on the unit circle:

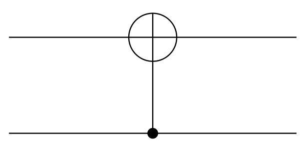


The transformation amounts to a reflection in the $|0\rangle$-axis followed by a counterclockwise rotation through $\pi/4$. As with any legal one-qubit quantum gate, it maps the unit circle onto itself. Note that $H^2 = I$, the identity map. That is, H is its own inverse.

A *quantum circuit* is a circuit that allows only quantum gates. It corresponds to an orthogonal linear transformation of $\mathcal{H}$, and thus it maps the unit sphere in $\mathcal{H}$ onto itself. Here's an example taken from Nielsen and Chuang [1, Exercise 4.20]. This particular example is interesting in that it blurs the distinction between the control and target qubits. I'll justify below that the CNOT gate qualifies as a quantum gate.



As an exercise, write out the state just after the CNOT gate is applied but before the two final Hadamard gates, assume the initial state is $|00\rangle$. This circuit is actually equivalent to



5.1 Input and Output

Input and output registers are defined as before. The initial state of the circuit is a basis state as before, and the final state is

$$|\text{final}\rangle = \sum_{x \in \{0,1\}^n} a_x |x\rangle,$$

where the a_x are real coefficients. By the preservation of the ℓ_2-norm, $|\text{final}\rangle$ has unit ℓ_2-norm, so we have $\sum_x a_x^2 = 1$. This suggests that we interpret a_x^2 as the probability associated with the basis state $|x\rangle$ in the final state. This is indeed what we do; the a_x are known as *probability amplitudes*. We observe the output qubit in the final state and see 0 and 1 with probabilities

$$\sum_{x_2,\cdots,x_n} a^2_{0x_2\cdots x_n} \quad \text{and} \quad \sum_{x_2,\cdots,x_n} a^2_{1x_2\cdots x_n},$$

respectively. These formulas generalize in the obvious way to the case of more than one output register being observed.

Since it is the squares of the amplitudes that affect the probabilities, the sign of an amplitude (that is a versus $-a$) in $|\text{final}\rangle$ has no observable effect. The upshot of this is that we can and often do ignore an *unconditional* discrepancy of sign. For example, the two gates H and $-H$ are completely interchangeable in any circuit; swapping them will lead to all the same observation probabilities in the end. The unconditionality is important here; the sign change must apply to the whole matrix. The following two gates are *not* interchangeable, even though corresponding entries differ at most by a change of sign:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

To see that the two gates cannot be interchanged, compare the circuit

$$-\boxed{H}-\boxed{I}-\boxed{H}-$$

with

$$-\boxed{H}-\boxed{Z}-\boxed{H}-$$

with initial state $|0\rangle$ for both. The first circuit does nothing, since $HIH = H^2 = I$, so its the final state is $|0\rangle$. For the second circuit, however, we have

$$|0\rangle \overset{H}{\mapsto} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \overset{Z}{\mapsto} \frac{|0\rangle - |1\rangle}{\sqrt{2}} \overset{H}{\mapsto} |1\rangle,$$

And it can be easily checked that $|1\rangle$ maps to $|0\rangle$. Thus the second circuit is equivalent to a NOT gate.

5.2 Still More Complexity Classes

As with probabilistic circuits, several new (and some old) complexity classes can be defined using ptime uniform families of quantum circuits with various acceptance criteria.

Class	Acceptance Criterion
EQP	$(\{0\}, \{1\})$
$\mathrm{C}_{\neq}\mathrm{P}$	$(\{0\}, (0,1])$
RQP	$(\{0\}, (\frac{1}{2}, 1])$
BQP	$([0, \frac{1}{3}], [\frac{2}{3}, 1])$
PP	$([0, \frac{1}{2}], (\frac{1}{2}, 1])$

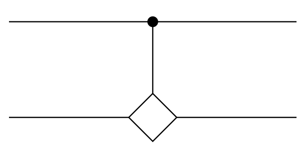
5.3 What Quantum Gates Should We Allow?

The happy answer to this question is that it largely does not matter. Several results in the literature show that a large variety of collections of quantum gates are all equivalent for defining the complexity classes above. Such collections are called *universal* for quantum computation. We'll describe a few universal collections here.

First we need to know: can a Boolean gate of Section 3 serve as a quantum gate? The answer is yes if and only if the gate is reversible. Recall that a Boolean gate corresponds to a matrix of 0s and 1s, and to be a quantum gate the matrix must be orthogonal. The only such matrices are permutation matrices, corresponding to reversible Boolean operations. Thus the AND and OR gates are not allowed, but the NOT, CNOT, and Toffoli gates are.

A recent result of Shi shows that the Hadamard gate H and the Toffoli gate together form a universal collection [2]. In fact, Shi showed that the Toffoli gate together with any single-qubit gate that maps some basis state to a linear combination of two or more basis states form a universal collection. (He also showed that the CNOT gate together with any single-qubit gate G such that G^2 maps some basis state to a linear combination of two or more basis states serves as a universal collection.) These are certainly minimalist universal collections. On the other end of the spectrum, we may allow any finite collection of quantum gates whose matrix entries are approximable in polynomial time. (A real number r is *polynomial-time approximable* if the nth digit in the binary expansion of r can be computed in time polynomial in n.)

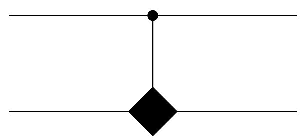
Here's one more universal collection. It consists of three gates: CNOT, Hadamard, and the two qubit-gate



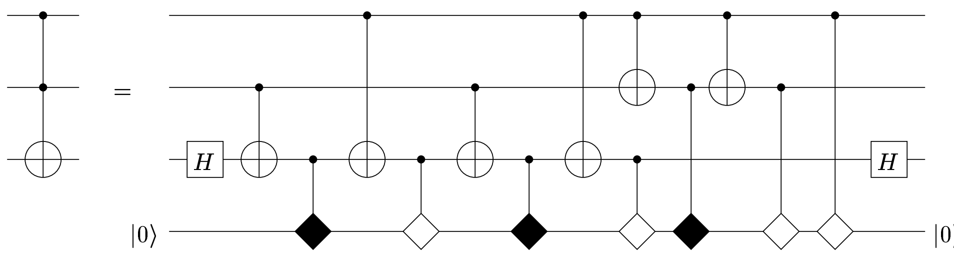
described by the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ 0 & 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix}.$$

We'll denote this gate, and its corresponding linear transformation, by T. Clearly $T^8 = I$. Thus $T^7 = T^{-1}$, and we denote this inverse gate by



The Toffoli gate can be simulated exactly by the following rather amazing circuit consisting of CNOT, Hadamard, and T-gates:



The fourth qubit on the right is an ancilla. Note that it is used cleanly here; the final state of the right circuit has no components where the value of the ancilla is 1, regardless of the initial state of the other three qubits. If we start with a quantum circuit with Toffoli gates, then we can systematically replace each Toffoli gate with the subcircuit on the right, and we can reuse the same ancilla repeatedly for each replacement.

6 Complex Probability Amplitudes

We've developed the quantum circuit model using real probability amplitudes only. This suffices, but more traditional approaches allow complex amplitudes. I'll show the connection between the two approaches.

We start by generalizing the inner product of two real vectors in $\mathbb{R}^m$ to the *Hermitian inner product* of complex vectors in $\mathbb{C}^m$ as follows: let $u = (u_1, \ldots, u_m)$ and let $v = (v_1, \ldots, v_m)$ be column vectors. Their Hermitian inner product is

$$\langle u|v\rangle = \sum_{i=1}^{m} \overline{u_i} v_i,$$

where $\overline{z}$ denotes the complex conjugate of z. Note that $\langle u|u\rangle = \sum_i |u_i|^2 \geq 0$, with equality holding iff $u = 0$. The *Hermitian norm* $|u|$ of u is $\sqrt{\langle u|u\rangle}$ A matrix M that preserves the Hermitian inner product (that is, $\langle Mu|Mv\rangle = \langle u|v\rangle$ for all u, v) is called *unitary.* The *adjoint* of a matrix M, written $M^\dagger$, is the conjugate transpose of M; that is, the (i,j)th entry of $M^\dagger$ is the complex conjugate of the (j,i)th entry of M. It is easy to see that a matrix M is unitary if and only if $MM^\dagger = M^\dagger M = I$. This is in close analogy with real orthogonal matrices; in fact, a real matrix is unitary if and only if it is orthogonal. This means that the real-amplitudes model of Section 5 embeds nicely in the present model, simply by restricting the amplitudes to be real.

The computational basis is as before, but allowing complex coefficients means that the space $\mathcal{H}$ is now identified with $\mathbb{C}^{2^n}$. Quantum gates now must correspond to unitary transformations. As previously, a quantum circuit starts in a basis state, which has unit Hermitian norm. The unitary gates preserve the norm of the state, so that the final state $\sum_{x\in\{0,1\}^n} a_x|x\rangle$ satisfies $\sum_x |a_x|^2 = 1$. We therefore interpret $|a_x|^2$ as the probability that the final state of the circuit is $|x\rangle$.

Does this give a more powerful model than the one in Section 5 using real amplitudes? No, not really. Both define the same complexity classes. In fact one can easily transform a quantum circuit with complex amplitudes into an equivalent quantum circuit with real amplitudes at the expense of including one extra ancilla and adding one to the arity of some of the gates.

If M is any $k\times\ell$ complex matrix (this includes row and column vectors), we transform it into a $2k\times 2\ell$ real matrix $\rho(M)$ as follows: replace every entry $x+yi$ of M by the 2×2 real matrix

$$\begin{bmatrix} x & -y \\ y & x \end{bmatrix}.$$

We have the following facts:

- $\rho(MN) = \rho(M)\rho(N)$, and $\rho(M_1 + aM_2) = \rho(M_1) + a\rho(M_2)$, where $a \in \mathbb{C}$, and M, M_1, M_2, and N have any appropriate dimensions.
- $\rho(M^\dagger) = \rho(M)^{\mathrm{t}}$
- M is unitary if and only if $\rho(M)$ is orthogonal. This follows from item 2.
- $\rho(I) = I$. Here the second I is of course bigger than the first. This follows from item 1.

If u is column vector in $\mathbb{C}^m$, then $\rho(u)$ is technically a $2m\times 2$ matrix. There are only $2m$ real degrees of freedom in u, however, so we can identify u with a vector in $\mathbb{R}^{2m}$. The real dimension is twice the complex dimension. Since adding a new qubit to a set of registers doubles the dimension of $\mathcal{H}$, this suggests that we can simulate a circuit with complex amplitudes by a circuit with real amplitudes and one additional ancilla, and any gates with nonreal entries are simulated by gates that interact with this ancilla. All this indeed works using the ρ transformation above. The T-gate defined in Section 5 is actually ρ applied to the one-qubit gate with matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix},$$

which is kind of "conditional phase shift" gate. The circuit simulating the Toffoli gate in Section 5 was derived from a well-known complex-amplitude quantum circuit (see [1], for example).

References

[1] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[2] Y. Shi. Both Toffoli and controlled-NOT need little help to do universal quantum computation. Unpublished, 2002, quant-ph/0205115.