Quantum Computation: a formal introduction

Antoine Groudiev

Last updated December 31, 2023

One of the most widely used system for secure data transmission is the RSA scheme; its security relies on the assumption that the integer factorization problem is hard. To this day, there is indeed no known efficient algorithm to factor an integer *on a classical computer*.

Nevertheless, this problem can be solved in polynomial time on a quantum computer, using Shor's algorithm. A large enough quantum computer would therefore be able to break the RSA encryption scheme.

We will define the behavior of a quantum computer using two quantum computational models, and delve into quantum complexity theory, the study of complexity classes of problems solved using these quantum models. We will end by describing the Deutsch-Jozsa algorithm, to prove that a quantum computer can be exponentially faster than a deterministic classical computer.

Contents

1 Introduction to Quantum Computers

A classical computer uses the *bit* as a basic unit, a system which can be in exactly one of the two states `0` and `1`. Similarly, a quantum computer uses a basic unit called *qubit*, which can be in a *superposition* of two states. A qubit can be described as a linear combination of both states, each coefficient being related to the probability that the qubit is in the given state. This way, a qubit can store more information than a simple bit, by storing information for both states at a time.

1.1 Dirac notation

While representing the state of a classical bit is quite simple, manipulating qubits requires more advanced notation. The state of a qubit is often represented using the Dirac notation, also known as the Bra-ket notation. [6].

A qubit of state Ψ is noted $|\Psi\rangle$, and is in a superposition of the two state $|0\rangle$ and $|1\rangle$. This superposition can be written as:

$$|\Psi\rangle = \alpha|\Psi\rangle + \beta|\Psi\rangle$$

where $\alpha, \beta \in \mathbb{C}$ are called the *states amplitudes*.

Example (Schrödinger's cat)**.**

$$|\text{Schrödinger's cat}\rangle = \alpha|\text{dead}\rangle + \beta|\text{alive}\rangle$$

The probability to measure the qubit in the state $|0\rangle$ (and respectively $|1\rangle$) is $|\alpha|^2$ (respectively $|\beta|^2$). This interpretation gives us the *normalization condition*:

$$|\alpha|^2 + |\beta|^2 = 1$$

since the qubit must be measured in one of the two states.

Example (LFCC exam)**.** *Assume that you have a probability $p \in [0, 1]$ of passing the next LFCC exam. Your result at the exam can be represented using one qubit in the following state:*

$$|\text{exam}\rangle = \sqrt{p}\,|\text{pass}\rangle + \sqrt{1-p}\,|\text{fail}\rangle$$

We can verify that the normalization condition is respected:

$$(\sqrt{p})^2 + (\sqrt{1-p})^2 = 1$$

1.2 Vector representation

Writing qubits is nice, but we would like to be able to change their state too. The Dirac notation is not the most practical one when the possible states ($|0\rangle$ and $|1\rangle$) are explicit. Instead, we can use a vector representation:

$$|\Psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

This notation allows us to use linear algebra tools to express physical conditions and changes. The normalization condition simply becomes:

$$\||\Psi\rangle\|_2 = 1$$

Furthermore, we can model an action that changes the qubit's state simply by a matrix U. If a qubit of initial state $|\Psi\rangle$ goes through some quantum gate (more on this later) represented by U, the final state $|\Psi'\rangle$ would be:

$$|\Psi'\rangle = U|\Psi\rangle$$

Obviously, we want $|\Psi'\rangle$ to respect the normalization condition for every qubit Ψ. This requires U to be a unitary matrix, that is $UU^\dagger = I_2$, where $U^\dagger$ is the conjugate transpose of U[1].

1.3 The Bloch sphere

Visualizing the state of a qubit - and the transformations of this state as physical transformations are applied to it - can be challenging. A common way to visualize a qubit is the Bloch sphere.

Since $\||\Psi\rangle\|_2 = 1$, we can uniquely[2] write $|\Psi\rangle$ as:

$$|\Psi\rangle = \cos(\theta/2)\,|0\rangle + e^{i\phi}\sin(\theta/2)\,|1\rangle$$

where $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$. This parametrization of $|\Psi\rangle$ corresponds to spherical coordinates, and uniquely defines a vector $\vec{a} \in \mathbb{R}^3$:

$$\vec{a} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}$$

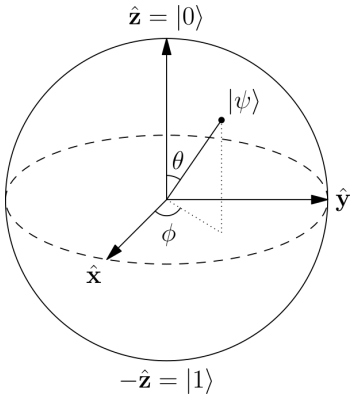

Figure 1: A Bloch sphere

The measure of the qubit $|\Psi\rangle$ can be seen as the projection of this vector on either $|0\rangle$ or $|1\rangle$, i.e. on $\vec{z}$ or $-\vec{z}$. The closer the point on the sphere is to a pole, the higher the probability to be measured in this state. When the vector is pointing directly to the top, the probability to be measured in $|0\rangle$ is 1; when pointing directly down, the probability is 0, and when the vector is orthogonal to $\vec{z}$, the probability is $\frac{1}{2}$, as shown in Figure 2.

2 Quantum Computational Models

To study the complexity of different problems with a quantum point of view, we need a model for a quantum computer. Much like classical computers, two approaches coexist: the representation of a quantum computer as a circuit, using quantum logic gates, and an abstract model called quantum Turing machine, by analogy with the classical Turing machine.

[1]The conjugate transpose is usually noted $U^\star$, but I guess that quantum physicists prefer the medieval vibe of †.

[2]Except when $|\Psi\rangle = |0\rangle$ or $|\Psi\rangle = |1\rangle$, since ϕ can take any value.

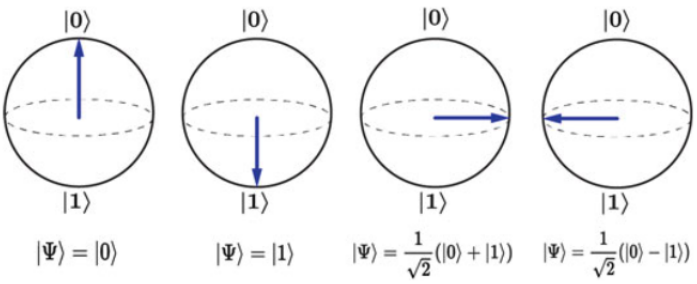

Figure 2: Examples of vectors on the Bloch sphere for some simple states

2.1 Quantum Circuits

A *quantum circuit* is the counterpart of the classical circuit; from an abstract point of view, it is composed of a multiple qubits, a sequence of *quantum gates*, and special gates called *measurements*, the quantum phenomenon which projects the qubit in one pure state. Consider the circuit below as an example. Each line of the circuit represents one qubit, and the x-axis represents time.

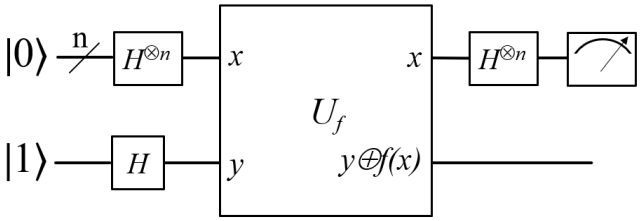

Figure 3: An example circuit (Deutsch-Jozsa's algorithm, which we will study later)

A quantum gate takes into input n qubits, and modifies the probability of each qubit to be in each state. This can be seen visually: the quantum gate changes the position of the qubit on the surface of the Bloch sphere. Therefore, quantum gates are often represented as unitary matrices of size $2^n \times 2^n$, i.e. elements of $U(2^n, \mathbb{C})$. We will now present a few basic quantum gates.

2.1.1 The X Gate

The X gate is the quantum equivalent of the classical NOT gate. It takes one qubit as an input, and permutes its amplitudes. Formally:

$$X : \alpha|0\rangle + \beta|1\rangle \mapsto \beta|0\rangle + \alpha|1\rangle$$

The unitary matrix associated to X is:

$$X \cong \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

2.1.2 The Z gate

The Z gate does not have a classical equivalent; its action over one qubit is simply to leave the $|0\rangle$ component untouched, and to multiply the $|1\rangle$ component by -1:

$$Z : \alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0\rangle - \beta|1\rangle$$

Or, written in matrix form:

$$Z \cong \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

2.1.3 Hadamard Gate

The Hadamard gate is a one-qubit gate without classical equivalent; one of its most basic use is to generate superposition on a qubit. The unitary matrix associated to the Hadamard gate, written H, is the following:

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

An immediate result is that:

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

meaning that a qubit starting in a *pure* state ($|0\rangle$ or $|1\rangle$) is transformed into a qubit in a balanced superposition of $|0\rangle$ and $|1\rangle$.

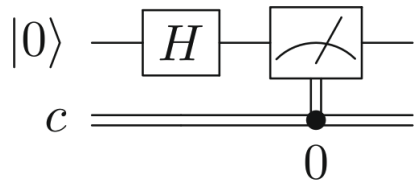

Figure 4: A basic circuit with the H gate

If we consider the circuit 2.1.3 on the left, the probability to measure $|0\rangle$ is exactly $\frac{1}{2}$. (Note that this is possibly the only method to have true random numbers.) If we were to replace the initial state of the qubit from $|0\rangle$ to $|1\rangle$. Similarly, if we were to modify the circuit to add an X gate between the Hadamard gate and the measuring "gate", the outputs probabilities would not change.

Note that like every quantum gate represented by a unitary matrix, and unlike many classical gates, the Hadamard gate is reversible. This might seem counter-intuitive since the previous example suggests that we "lost" the information on the qubit.

Remark. *Similarly to XOR, which can be constructed using only AND and OR, the gates presented above are not "independent": it is possible*[3] *to construct the X gate using only Z and H gates.*

2.1.4 Entanglement

The set of gates above lack expressivity, since they operate on a single qubit only. We will now present a gate acting on multiple qubits. This highlights a pure quantum phenomenon which is *entanglement*, meaning that multiple qubits are correlated.

Example (Bell's entanglement)**.** *Consider two untangled qubits Ψ and Φ. Their default state is:*

$$|\Psi\Phi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$$

meaning that each outcome of the universe $\{(0,0);(1,1);(1,0);(1,1)\}$ is equally probable. By entangling Ψ and Φ, it is possible to put them in the Bell state*, that is:*

$$|\Psi\Phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Where the only possible outcomes are $(0,0)$ and $(1,1)$. By measuring Ψ, we can deduct the "value" of Φ, exactly as if it was measured, which breaks the quantum superposition phenomenon, no matter the distance between both qubits.[4]

[3] The proof is left as an exercise to the reader

[4] This is what Einstein called the "spooky action at a distance"

2.1.5 CNOT gate

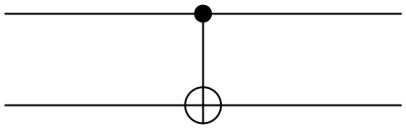

Figure 5: Circuit representation of CNOT

The *Controlled NOT* gate, (or CNOT gate) is a fundamental gate which allows to entangle qubits. It acts on two qubits, a *control qubit*, and a *target qubit*. Intuitively, if the control qubit is $|0\rangle$, the target qubit is unchanged; if the control qubit is $|1\rangle$, we apply an X gate to the target qubit. (Note that this description was expressed in the $(|0\rangle, |1\rangle)$ basis, but that the control qubit is in general in a superposition of pure states.)

This gate can be interpreted as the quantum equivalent of the classical "reversible XOR" gate, a 2×2 gate taking in input x and y, and returning x and $x \oplus y$[5].

The action of the CNOT gate corresponds to the following matrix:

$$CNOT \cong \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

or, seen otherwise, the CNOT gate transforms

$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

in

$$\alpha|00\rangle + \beta|01\rangle + \delta|10\rangle + \gamma|11\rangle.$$

2.2 Quantum Finite Automaton

The formalization of quantum computers in formal circuits is useful to design quantum algorithms and quantum programming languages. Nevertheless, the study of quantum complexity theory requires formal tools that are easier to handle, similar to the tools provided by formal languages in classical computation. This lead to the introduction [10] of quantum languages and abstract machines around them.

2.2.1 Quantum Languages

Let Σ be an alphabet, and $L \subseteq \Sigma^\star$ a language over the alphabet Σ. Another way to see this language, apart from a subset of $\Sigma^\star$, is using its characteristic function χ_L:

$$\chi_L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{otherwise} \end{cases}$$

By analogy to this, we can define a *quantum language* as a function mapping words to probabilities: [10]

Definition (Quantum language)**.** A quantum language over the alphabet Σ is a function f such as:

$$f : \Sigma^\star \to [0, 1]$$

f is a classical language if $f(\Sigma^\star) \subseteq \{0, 1\}$.

[5] $\oplus$ is used in quantum mechanics to denote the addition modulo 2, or similarly the XOR operation

2.2.2 Definition of QFAs and QRLs

Definition (QFA)**.** A *Quantum Finite Automaton* $Q = (H, s_{\text{init}}, H_{\text{accept}}, P_{\text{accept}}, \Sigma, \delta)$ consists of:

- a Hilbert space[6] H of dimension n. (This will generalize the set Q of states of a classical DFA.)
- an initial normalized vector $s_{\text{init}} \in H$ (i.e. $\|s_{\text{init}}\|^2 = 1$). (This is the equivalent of the initial state in a DFA.)
- a subspace $H_{\text{accept}} \subseteq H$, and an operator P_{accept} which projects onto it
- an alphabet Σ
- a function $\delta : \Sigma \to U_n(\mathbb{C})$[7], mapping each letter of $a \in \Sigma$ to a unitary matrix $U_a \in U_n(\mathbb{C})$.

We will write $\delta^\star(w = w_1 \cdots w_{|w|}) = \delta(w_{|w|}) \cdots \delta(w_1) = U_{w_{|w|}} \cdots U_{w_1}$. Finally, the language recognized by Q is the following function:

$$f_Q : w \mapsto \|P_{\text{accept}} \delta^\star(w) s_{\text{init}}\|^2$$

(It is convenient to see the vector and matrix multiplications being computed from right to left as the word is being read by the automata.)

Intuitively, we start from the s_{init} vector (or state), and we follow w by multiplying the current vector by the matrix associated to the current letter of the input word. When the word is finished, we project the current vector (which remains of norm 1) on the space H_{accept} using the operator P_{accept}. If $\delta^\star(w) s_{\text{init}} \in H_{\text{accept}}$, the P_{accept} will leave $\delta^\star(w) s_{\text{init}}$ untouched, and the output probability will be 1; otherwise, the output probability decreases as the norm projection of $\delta^\star(w) s_{\text{init}}$ decreases.

The end of the process corresponds to a measurement, and $f_Q(w)$ can be interpreted as the probability that this measurement has an acceptable outcome.

Definition (QRL)**.** A *Quantum Regular Language* is a quantum language recognized by a QFA.

2.2.3 A few properties of QRLs

Interestingly enough, QRLs inherit some properties from classical regular languages, like closure properties and an equivalent of the pumping lemma.

Theorem (Closure of QRLs by product)**.** *Let f, g be QRLs. Then the product fg is a QRL.*

Remark. *In the context of to classical languages L_1 and L_2, the product $\chi_{L_1}\chi_{L_2}$ corresponds to the intersection $L_1 \cap L_2$. Recall that if L_1 and L_2 are regular languages, $L_1 \cap L_2$ is also regular.*

Theorem (Closure of QRLs by linear combination)**.** *Let f_i be QRLs, and c_i be constants such as $\sum_i c_i \leq 1$. Then, $\sum_i c_i f_i$ is a QRL.*

A powerful result to show that a language is not a (classical) regular language is the pumping lemma. Quantum regular languages have a similar result, which is the following:

Theorem (Quantum Pumping Lemma)**.** *If f is a QRL, then for any word $w \in \Sigma^\star$ and any $\varepsilon > 0$, there is a $k \in \mathbb{N}^\star$ such that $\|f(uw^k v) - f(uv)\| < \varepsilon$ for any words u, v. Moreover, if f's automaton is n-dimensional, there is a constant c independent of ε such that $k < (c\varepsilon)^{-n}$.*

The idea mimics the behavior of an irrational rotation of a circle; for every word w, there is a k such as applying w k times (which is like applying a rotation to the state) brings us back within a distance of at most ε from the word uv from which we started.

This result can be used to show that the regular language of words over $\Sigma = \{a, b\}$ which do not contain the subword bb is not a quantum regular language [10].

[6] Recall that a Hilbert space is an inner product space complete for the distance induced by the inner product

[7] $U_n(\mathbb{C})$ is the set of unitary matrices of size n

2.3 Quantum context-free languages

Definition (Quantum Grammar[8])**.** A *Quantum Grammar* $G = (V, T, I, P)$ of *dimensionality* n consists of:

- an alphabet V of variables
- an alphabet T of terminals
- an initial variable $I \in V$
- a finite set of productions P.

A *production* of P is of the form $\alpha \to \beta$, where $(\alpha, \beta) \in V^\star \times (T \cup V)^\star$. To each production in P is associated a set of complex amplitudes $(c_k(\alpha \to \beta))_{1 \leq k \leq n}$.

Consider one derivation $\alpha \Rightarrow \beta$, a chain of strings in which at each step one substring is replaced with another according to a production rule. We define the k-th amplitude c_k of *one* derivation to be the product of the k-th amplitudes of the productions used in the chain. Furthermore, we define $c_k(\alpha \Rightarrow \beta)$ to be the product of the c_k's for *all the derivation* from α to β.

The k-th amplitude of a word $w \in T^\star$ is defined as $c_k(w) := c_k(I \Rightarrow w)$. Finally, G *generates* the quantum language f defined as:

$$f(w) = \sum_{k=1}^{n} \|c_k(w)\|^2$$

Remark. *This construction appears to be much more complicated than the standard grammar construction. This is mainly because of the introduction of the n different amplitudes (c_k). Why do we need multiple amplitudes instead of just one? The reason is pretty technical, but multiple amplitudes are needed to prevent multiple paths from interfering.*

Definition (QCFG)**.** A quantum grammar G is *context-free* if the only productions with non-zero amplitudes are the productions $\alpha \to \beta$ with $\alpha \in V$.

Definition (QCFL)**.** A *Quantum Context-Free Language* is a language generated by a quantum context-free grammar.

As you could have guessed, an important result on QCFL is that a quantum language is context-free if and only if it is recognized by the quantum equivalent of a push-down automaton.

Definition (QPDA)**.** A *Quantum Push-Down Automata* $Q = (H = Q \otimes \Sigma, T, Q_{\text{acccept}}, s_{\text{init}}, A, \delta)$ consists of:

- a Hilbert space H (the space of configurations of the automata). More specifically, we must have $H = Q \otimes \Sigma$ for some spaces Q, Σ[9]
- Q is a finite-dimensional space called the *control state*
- Σ is an inifinite-dimensional space called the *stack space*
- each basis vector of Σ is a finite word over the *stack alphabet* T
- an initial state s_{init}, which must be a superposition of a finite number of initial control states and stack states
- a subspace $Q_{\text{acccept}} \subseteq Q$, such as the accepting space $H_{\text{acccept}} = Q_{\text{acccept}} \otimes \{\varepsilon\}$ (this requires that the stack is empty at the end of the word for it to be accepted by the automata)
- an operator P_{acccept} projecting on $Q_{\text{acccept}} \otimes \{\varepsilon\}$
- an input alphabet A

[8] `https://xkcd.com/1090/`

[9] $Q \otimes \Sigma$ denotes the tensor product of spaces Q and Σ

– a transition function δ, such as $\forall a \in A$, $\delta(a)$ is a unitary endomorphism of H

To ensure that the stack has indeed a LIFO[10] behavior, we add some constraints on the transitions probabilities. Let $q_1, q_2 \in Q$ be control states, and $\sigma_1, \sigma_2 \in T$ be stack states; then the transition amplitude from (q_1, σ_1) to (q_2, σ_2) can only be non-zero if there is $t \in T$ such as:

$$(\sigma_1 t = \sigma_2) \vee (\sigma_1 = \sigma_2 t) \vee (\sigma_1 = \sigma_2)$$

meaning that the transition can only push, pop, or leave the stack unchanged. Furthermore, transition amplitudes can only depend on the top (rightmost) symbol of σ_1 and σ_2.

Similarly to QFAs, the language recognized by Q is the following function:

$$f_Q : w \mapsto \|P_{\text{accept}} \delta^\star(w) s_{\text{init}}\|^2$$

Remark. *Unlike a classical PDA, transitions can depend on both the top symbol of the stack, and the symbol below it. This is because we allowed the dependency on both σ_1 and σ_2. This is necessary for the endomorphisms to be unitary.*

Theorem. *A quantum language is a QCFL if and only if it is recognized by a QPDA.*

2.4 Quantum Turing Machine

Similarly to the classical case, the previous constructions can be generalized to gain expressivity. This leads to the following definition of a Quantum Turing Machine[1][2], the equivalent of a classical Turing machine, but using the previous constructions using Hilbert spaces and transition matrices:

Definition (QTM)**.** A *Quantum Turing Machine* (or QTM) $M = (H, \Gamma, b, \Sigma, \delta, q_0, Q_{\text{accept}})$ consists of:

– a Hilbert space Q (replacing the classical set of states Q)
– another Hilbert space Γ (replacing the classical set of tape symbols)
– a blank symbol $\sqcup \in \Gamma$
– an alphabet of input and output symbols Σ
– an initial state $q_0 \in Q$
– a subset $Q_{\text{accept}} \subseteq Q$ of accepting states (and, as usual, an operator P_{accept} which projects onto it)
– a transition function δ such as:

$$\delta : \Sigma \times Q \otimes \Gamma \to \Sigma \times Q \otimes \Gamma \times \{L, R\}$$

An interpretation of this transition function is that to each letter of Σ is associated an automorphism of $Q \otimes \Gamma$, which changes the state of the machine and the tape.
Like a classical TM, this QTM is given an infinite tape indexed by $\mathbb{Z}$, on which the head can read, write, and move left or right.

Remark. *There is different common definitions of the classical Turing machine, but they remain all quite similar; the differences are often about whether the head can stay in place, whether the tape is indexed by $\mathbb{N}$ or $\mathbb{Z}$, etc. In the case of a quantum Turing machine, many definitions coexist, with major differences in the handling of the tape, the measurement, and the output. The main idea to use unitary matrices to transform both the Hilbert space Q and the tape was introduced by Deutsch in [4]. Since this first publication, another definition by Bernstein and Vazirani in [2] has been vastly used for quantum complexity theory. Recent papers, such as [7], try to unify these two standard approaches of QTM, but no clear standard definition of a QTM has emerged. The definition given above is more of a sketch of a definition which tries to unify both approaches.*

[10] Last In, First Out

3 Quantum Complexity Theory

Similarly to the study of classical algorithms, we can regroup quantum problems in classes depending on their solvability by a quantum computational model under time and space constraints.

3.1 Quantum complexity classes

3.1.1 The BQP class

An important class of problem is the BQP class; it is the equivalent of the BPP class (Bounded-error, Probabilistic Polynomial time) for classical probabilistic computers.

Definition (BQP)**.** The *Bounded-error Quantum Polynomial time* (BQP) class is the set of decision problems which can be solved in polynomial time by a quantum Turing machine, with an error of at most $\frac{1}{3}$.

Remark. *The formal definition often involves a maximum error of $\frac{1}{3}$ in one run, but any bound strictly lower than $\frac{1}{2}$ is sufficient, since multiple runs of the algorithm can be chained to reduce the error.*

Definition (Promise-BQP)**.** The *Promise-BQP* class is the generalization of the BQP class, in which the input is *promised* to belong to a specific subset of all inputs.

We have BQP $\subset$ Promise-BQP. We are going to present a Promise-BQP-complete problem; it is interesting to notice that no BQP-complete problem is known.

3.1.2 A Promise-BQP-complete problem

Definition (APPROX-QCIRCUIT-PROB problem[11])**.** Having as an input two numbers α, β such as $0 \leq \beta < \alpha \leq 1$, and the description of a quantum circuit C such as:

- the circuit acts on n qubits
- the circuit contains m gates
- the number of gates m is polynomial in n
- the probability P to measure the first output qubit of the circuit initialized at $|0\rangle^{\otimes n}$ is either $P \geq \alpha$ or $P \leq \beta$ (this is the promise)[12]

the algorithm needs to output whether $P \geq \alpha$ or $P \leq \beta$.

Theorem. *Any BQP problem reduces to APPROX-QCIRCUIT-PROB.*

3.1.3 Relationship to classical complexity classes

We know the following inclusions:

$$P \subseteq BPP \subseteq BQP \subseteq PP \subseteq PSPACE \subseteq EXP$$

Most of these proofs use APPROX-QCIRCUIT-PROB.

These inclusions are interesting but simply show that a quantum computer is at least as good as a classical computer. An important question is the relationship between BQP and NP, which remains unsolved. A main reason for this is the uncertainty we have about whether

[11] I know, the name is slightly more complicated than "NP"

[12] The notation $|0\rangle^{\otimes n}$ means that the n input qubits are initialized to $|0\rangle$

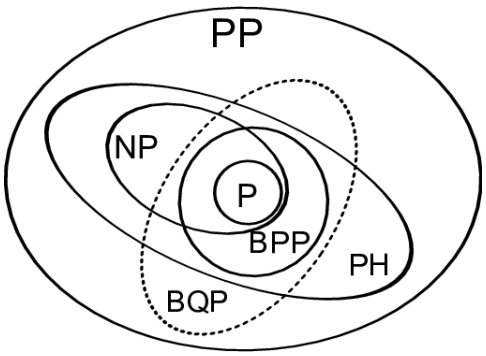

Figure 6: Known and suspected inclusions of BQP

P=NP. Nevertheless, it is believed that BQP contains problems that are not in P, such as integer factorization, which would imply that $BQP \cap NP \neq \varnothing$.

Furthermore, it has been recently proven[11] that there is an oracle relative to which BQP is not contained in PH, the polynomial hierarchy, which would suggest – but not prove – that $BQP \nsubseteq PH$.

3.2 Relationship to the Church-Turing Thesis

Some non-computer-scientists might tend to think that quantum computers are able to "do things that no traditional computer can". While we will see that this remains false in theory, the emergence of quantum computers questioned the possibilities of computation and efficient computation. Recall the Church-Turing thesis:

Thesis (Church-Turing Thesis)**.** A function on the natural numbers can be calculated by an effective method if and only if it is computable by a Turing machine.

Any quantum algorithm can be simulated either by hand, or on a classical computer, even if the resulting complexity of the computation is changed. Formally, it can be shown that a quantum Turing machine can be simulated on a Universal Turing Machine. Therefore, the existence of quantum computers do not disprove the Church-Turing thesis.

Nevertheless, the Church-Turing thesis do not state anything about the time complexity of the simulation. Historically, this lead to the following extension of the Church-Turing Thesis:

Thesis (Extended Church-Turing Thesis)**.** A probabilistic Turing machine can efficiently simulate any realistic model of computation.

Showing that $BPP \subsetneq BQP$ would imply that quantum computers are in certain cases more efficient than classical computers. This would therefore invalidate the extended Church–Turing thesis. A quantum-extended version of the thesis has therefore been proposed, and is now the basis of computational complexity theory [2]:

Thesis (Quantum-extended Church-Turing Thesis)**.** Any realistic physical computing device can be efficiently simulated by a fault-tolerant quantum computer.[8]

Overall, what these variations of the original Church-Turing thesis show, is that the existence of quantum computers do not change our fundamental understanding of what computation is. Quantum computers are only capable of solving problems we can already solve classically. Nevertheless, a sufficiently large quantum computer could *efficiently* solve problems that are to this day considered as "hard" in practice, such as the integer factorization problem.

4 The Deutsch-Jozsa Algorithm

The Deutsch-Jozsa algorithm[5] is an important quantum algorithm, not because of any practical use, but because it shows the interest of a quantum computer over a classical computer. The existence of the problem and algorithm proves that there is an *oracle separation* between the EQP and P class: this is a black box problem that can be solved in polynomial time using a quantum computer, but that cannot be solved by a classical computer in polynomial time.

4.1 Problem description

Let's consider a function f operating on n bits or qubits. That is:

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

This function is assumed to have a specific property: it is either *constant* or *balanced*:

$$|f^{-1}(\{0\})| = n \vee |f^{-1}(\{1\})| = n \vee \left(|f^{-1}(\{0\})| = |f^{-1}(\{1\})| = \frac{n}{2}\right)$$

This means that either the function produces only `0`, or it produces only `1`, or it produces exactly half of `0` and half of `1`. This is the promise of the problem.

The f function behaves like a black box, or an *oracle*: it is able to output the outcome of $f(x)$ in a single operation for any x. The algorithm is asked to determine whether the function is constant or balanced.

4.2 Classical solution

A deterministic computer needs in the worst case to measure more than half the 2^n possible outcomes for f, i.e. $2^{n-1} + 1$; the best time complexity is therefore exponential. Nevertheless, this problem can be solved efficiently on a probabilistic computer with a high probability, making it in BPP.

4.3 Quantum algorithm

We will describe the algorithm in the case $n = 1$, simply known as Deutsch's algorithm. Interestingly, the same procedure simply extends to any value of n, as we will see in the next subsection.

The implementation of the function f is assumed to be of the form:

$$f : |x\rangle|y\rangle \mapsto |x\rangle|f(x) \oplus y\rangle$$

This is needed to ensure that the gate is unitary.

The algorithm uses two qubits. Its circuit representation is the following:

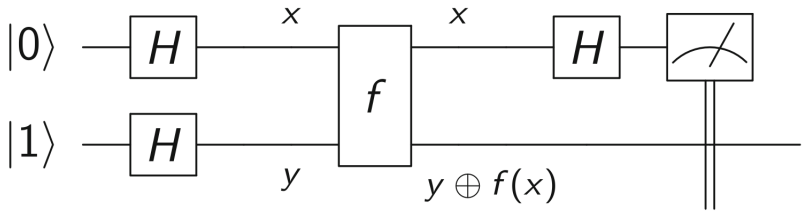

Figure 7: The quantum circuit used by Deutsch's algorithm

Note that we will only measure the first qubit.[13] The goal of this circuit is to obtain the value of $f(0) \oplus f(1)$. If the function is constant, $f(0) \oplus f(1) = 0$, and if the function is balanced, $f(0) \oplus f(1) = 1$. Therefore, by knowing the value of $f(0) \oplus f(1)$, we can determine using the promise whether f is constant or balanced. We will now describe the algorithm step-by-step.

1. Initialize two qubits such as the *product state* of the qubits is $|0\rangle|1\rangle$.
2. Apply to each qubit a Hadamard transform (fancy name for applying a Hadamard gate). The state changes to:
$$\left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right)\left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right)$$
3. Apply the f gate to the qubits. According to the definition of the gate, the state changes to:
$$\begin{aligned}\frac{1}{2}\Big[|0\rangle\Big(|0 \oplus f(0)\rangle - |1 \oplus f(1)\rangle\Big) + |1\rangle\Big(|0 \oplus f(1)\rangle - |1 \oplus f(1)\rangle\Big)\Big] \\ = \frac{1}{2}\Big[(-1)^{f(0)}|0\rangle\Big(|0\rangle - |1\rangle\Big) + (-1)^{f(1)}|1\rangle\Big(|0\rangle - |1\rangle\Big)\Big] \\ = (-1)^{f(0)}\frac{1}{2}\Big(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\Big)\Big(|0\rangle - |1\rangle\Big)\end{aligned}$$
This last form is interesting since we wrote the product state as an actual product of the two qubits states. If we look only at the state of the first qubit, we have after normalization:
$$\frac{1}{\sqrt{2}}\Big(|0\rangle + (-1)^{f(0)\oplus f(1)}|1\rangle\Big)$$
4. Apply one final Hadamard gate to the first qubit. We obtain:
$$\frac{1}{2}\Big[\Big(1 + (-1)^{f(0)\oplus f(1)}\Big)|0\rangle + \Big(1 - (-1)^{f(0)\oplus f(1)}\Big)|1\rangle\Big]$$
5. Measure the qubit. Note that if $f(0) \oplus f(1) = 0$, the qubit is in state $|0\rangle$, and if $f(0) \oplus f(1) = 1$, the qubit is in state $|1\rangle$. Therefore, if we measure $|0\rangle$, the probability that f is constant is 1, and if we measure $|1\rangle$, the probability that f is balanced is 1. We solved the problem.

4.4 General case

The ideas for the general case of the Deutsch-Jozsa algorithm are the exact same as the ones in Deutsch's algorithm. The main change is to replace the first qubit by a bus of n qubits:

You can show, using calculations similar to what we did before, that the final probability to measure $|0\rangle^{\otimes n}$ is:
$$\left\|\frac{1}{2^n}\sum_{x=0}^{2^n-1}(-1)^{f(x)}\right\|^2$$
which is 1 if and only if f is constant. Therefore, by measuring the n first qubits at the end, you can answer the problem by simply browsing in linear time the result of the measures; if a non-zero state is measured, f is balanced (according to the promise), otherwise f is constant.

[13]The second qubit is known as an *ancilla qubit*: `https://en.wikipedia.org/wiki/Ancilla_bit`

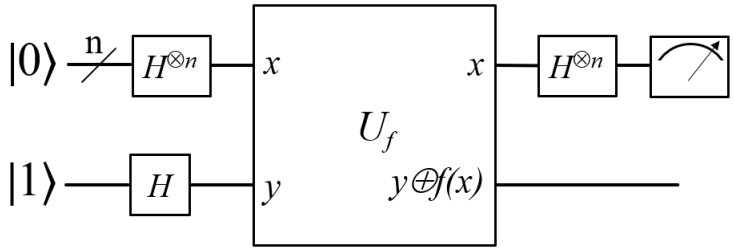

Figure 8: The quantum circuit used by Deutsch-Jozsa's algorithm in the general case

4.5 Complexity theory consequences

We saw that this problem could not be solved exactly by a classical computer in polynomial time, meaning that the problem is not in P. Nevertheless, the problem is exactly solvable by a quantum computer in polynomial time, hence it is a problem of the class EQP (Exact Quantum Polynomial time). Therefore, the problem yields an oracle relative to which $EQP \neq P$.

This shows that some problems, the most famous one being the integer factorization problem, are exponentially slower[14] on a classical computer than on a quantum computer.

Conclusion

We hope that this introduction to quantum computation gave you an overall understanding of how quantum computers work. The main idea emerging from the multiple approaches we took to describe a quantum computer (quantum circuits, formal languages, complexity theory) is that quantum computers are distinct from classical computers, not necessary better in every way. We saw that some problems could be efficiently solved on a quantum computer while no efficient classical solution exist. On the other hand, building a quantum computer is a technological challenge, making – to this day – most problems easier to solve on classical computers.

Nevertheless, the possibility of the existence of a large enough quantum computer is already starting to raise new theoretical questions in multiple fields of computer science: complexity theory, with the emergence of new complexity classes; cryptography, with the search for quantum-proof cryptography algorithms, ... The theoretical interest of quantum computation might not be what they *can* do, but instead the search of what they *cannot*.

End word Thank you for reading this far; if this interested you, and you want to know more about this broad topic, I recommend reading:

- [3] for more circuit-based complexity.
- [8] for an easy and applied approach of quantum computing
- [10] to delve further into the topic of Quantum Automata and Quantum grammars

References

[1] Molina A. and Watrous J. "Revisiting the simulation of quantum Turing machines by quantum circuits". In: *Proc Math Phys Eng Sci.* (2019).

[14] In the case of integer factorization, this is only true if P $\neq$ NP

[2] Ethan Bernstein and Umesh Vazirani. "Quantum Complexity Theory". In: *SIAM Journal on Computing* 26.5 (1997), pp. 1411–1473. URL: http://wpage.unina.it/pieroandrea.bonatti/didattica/complexity/slides/Bernstein-Vazirani.pdf.

[3] Richard Cleve. "An Introduction to Quantum Complexity Theory". In: *Quantum Computation and Quantum Information Theory*. https://arxiv.org/pdf/quant-ph/9906111.pdf. WORLD SCIENTIFIC, Jan. 2001, pp. 103–127.

[4] David Deutsch. "Quantum theory, the Church-Turing principle and the universal quantum computer". In: ed. by Proceedings of the Royal Society of London. 1985, pp. 97–117. URL: https://www.daviddeutsch.org.uk/wp-content/deutsch85.pdf.

[5] David Deutsch and Richard Jozsa. "Rapid Solution of Problems by Quantum Computation". In: *Proceedings of the Royal Society of London Series A* 439.1907 (Dec. 1992), pp. 553–558.

[6] Paul Dirac. "A new notation for quantum mechanics". In: *Mathematical Proceedings of the Cambridge Philosophical Society* (1939).

[7] Stefano Guerrini, Simone Martini, and Andrea Masini. "Quantum Turing Machines Computations and Measurements". In: *CoRR* abs/1703.07748 (2017). URL: http://arxiv.org/abs/1703.07748.

[8] Jack D. Hidary. *Quantum Computing: An Applied Approach*. Second. Springer Charm, Aug. 2019. URL: https://link.springer.com/book/10.1007/978-3-030-23922-0.

[9] Ciaran Hughes et al. *Quantum Computing for the Quantum Curious*. Springer Charm, Mar. 2021. URL: https://link.springer.com/book/10.1007/978-3-030-61601-4.

[10] Cristopher Moore and James P. Crutchfield. *Quantum Automata and Quantum Grammars*. 1997.

[11] Ran Raz and Avishay Tal. "Oracle Separation of BQP and PH". In: *Electronic Colloquium on Computational Complexity* (2018).