10 Quantum Complexity Theory I

Just as the theory of computability had its foundations in the Church-Turing thesis, computational complexity theory rests upon a modern strengthening of this thesis, which asserts that any "reasonable" model of computation can be efficiently simulated on a probabilistic Turing machine (by "efficient" we mean here a runtime that is bounded by a polynomial in the runtime of the simulated machine). For example, computers that can operate on arbitrary length words in unit time, or that can exactly compute real numbers with infinite precision are unreasonable models, since it seems clear that they cannot be physically implemented. It had been argued that the Turing machine model is the inevitable choice once we assume that we can implement only finite precision computational primitives. Given the widespread belief that NP $\neq$ BPP, this would seem to put a wide range of important computational problems (the NP-hard problems) well beyond the capability of computers.

However, the Turing machine is an inadequate model for all physically realizable computing devices for a fundamental reason: the Turing machine is based on a classical physics model of the universe, whereas current physical theory asserts that the universe is quantum physical. Can we get inherently new kinds of (discrete) computing devices based on quantum physics? The first indication that such a device might potentially be more powerful than a probabilistic Turing machine appeared in a paper by Feynman about two decades ago. In that paper, Feynman pointed out a very curious problem: it appears to be impossible to simulate a general quantum physical system on a probabilistic Turing machine without an exponential slowdown. The difficulty with the simulation has nothing to do with the problem of simulating a continuous system with a discrete one – we may assume that the quantum physical system to be simulated is discrete, some kind of quantum cellular automaton. It has rather to do with a phenomenon in quantum physics that allows different outcomes of a quantum physical effect to interfere with each other in a strange way. In view of Feynman's observation, we must re-examine the foundations of computational complexity theory and the complexity-theoretic form of the Church-Turing thesis, and study the computational power of computing devices based on quantum physics.

A first precise model of a quantum physical computer, called the quantum Turing machine, was formulated by Deutsch. This model may be thought of as a quantum physical analogue of a probabilistic Turing machine: it has an infinite tape and a finite state control, and the actions of the machine are local and completely specified by this finite state control. Furthermore, on a given input a quantum Turing machine produces a random outcome according to a probability distribution. However, instead of a probabilistic Turing machine, the quantum Turing machine can generate probability distributions in which the amplitudes of certain configurations can have complex instead of just positive real numbers.

10.1 Mathematical foundations

In order to understand how quantum computers work, we need some background in linear algebra and complex numbers.

Complex numbers

The set of complex numbers $\mathbb{C}$ is defined as the set $\mathbb{R} \times \mathbb{R}$ with the following two operations:

- For all $(a,b),(a',b') \in \mathbb{C}$, $(a,b)+(a',b') = (a+a', b+b')$.
- For all $(a,b),(a',b') \in \mathbb{C}$, $(a,b)\cdot(a',b') = (a\cdot a' - b\cdot b', a\cdot b' + b\cdot a')$.

When representing $(1,0)$ as 1 and $(0,1)$ as the imaginary number $i = \sqrt{-1}$, we can also write every number $z \in \mathbb{C}$ as $z = a+ib$ for some $a,b \in \mathbb{R}$, and we can use the standard addition and multiplication operations on these numbers. This is correct because for any $z = a+ib$ and $z' = a'+ib'$ it follows that

- $z+z' = (a+a') + i(b+b')$ and
- $z\cdot z' = (a+ib)(a'+ib') = aa' + iab' + iba' + i^2bb' = (aa'-bb') + i(ab'+ba')$,

which matches the rules for addition and multiplication above.

There is yet another way of representing a complex number. Any $z = a+ib$ can be viewed as a 2-dimensional vector of length $\ell = \sqrt{a^2+b^2}$ and angle $\alpha = \tan(b/a)$. Since the Euler function extended to the complex numbers has the property that

$$e^{i\alpha} = \cos\alpha + i\sin\alpha$$

it follows that z can also be represented as $\ell \cdot e^{i\alpha}$. This immediately implies that

$$z^2 = (\ell\cdot e^{i\alpha})^2 = \ell^2 \cdot e^{i2\alpha}$$

i.e., z^2 is a 2-dimensional vector of length ℓ^2 and angle 2α. On the other hand, this implies that $\sqrt{z}$ is a 2-dimensional vector of length $\sqrt{\ell}$ and angle $\alpha/2$. Hence, if we set $z = (-1,0) = -1$, then $\sqrt{z} = (0,1) = i$, and so it is natural to define $i = \sqrt{-1}$ above.

Given a complex number $z = a+ib$,

- $z^* = a - ib$ is called the *complex conjugate* of z and
- $||z||^2 = z\cdot z^* = a^2+b^2$ is called the *square norm* of z

Linear algebra

We start with linear algebra over the real numbers. Let $M(m,n;\mathbb{R})$ be the space of all $m\times n$-dimensional *matrices*

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

with $a_{ij} \in \mathbb{R}$ for every $1 \le i \le m$ and $1 \le j \le n$. If $n = 1$, we just call $\boldsymbol{A}$ a *vector* and denote it by $\boldsymbol{a}$ instead of using a capital letter. When m and n are clear from the context, then we also write $\boldsymbol{A} = (a_{ij})$ and $\boldsymbol{a} = (a_i)$. Addition and multiplication over matrices is defined as follows:

- For all $\boldsymbol{A},\boldsymbol{B} \in M(m,n;\mathbb{R})$, $\boldsymbol{A}+\boldsymbol{B} = \boldsymbol{C} \in M(m,n;\mathbb{R})$ with $c_{ij} = a_{ij} + b_{ij}$ for all i,j.
- For all $\boldsymbol{A} \in M(m,n;\mathbb{R})$ and $\boldsymbol{B} \in M(n,p;\mathbb{R})$, $\boldsymbol{A}\cdot\boldsymbol{B} = \boldsymbol{C} \in M(m,p;\mathbb{R})$ with $c_{ij} = \sum_{k=1}^{m} a_{ik}\cdot b_{kj}$ for all i,j.

Given a matrix $\boldsymbol{A} \in M(m,n;\mathbb{R})$, the *transpose* $\boldsymbol{A}^T = (a'_{ij}) \in M(n,m;\mathbb{R})$ of $\boldsymbol{A}$ is defined as $a'_{ij} = a_{ji}$ for all i, j.

Given a vector $\boldsymbol{a} \in \mathbb{R}^n$, its ℓ_1*-norm* is defined as

$$|\boldsymbol{a}| = |a_1| + |a_2| + \ldots + |a_n| \ ,$$

its ℓ_2*-norm* or *Euclidean length* is defined as

$$||\boldsymbol{a}|| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$$

and its *square norm* is defined as $||\boldsymbol{a}||^2$.

When applying the matrix product to vectors, we obtain some interesting insights. Given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, it holds for the angle α between $\boldsymbol{a}$ and $\boldsymbol{b}$ that

$$\cos\alpha = \frac{\boldsymbol{a}^T \cdot \boldsymbol{b}}{||\boldsymbol{a}|| \cdot ||\boldsymbol{b}||}$$

Hence, $(\boldsymbol{a}^T \cdot \boldsymbol{b})/||\boldsymbol{a}||$ is equal to the length of the projection of $\boldsymbol{b}$ onto $\boldsymbol{a}$. A projection can be seen as the shadow of one vector on another. As an example, if the two vectors are *orthogonal* to each other, i.e., they form a right angle, then their scalar product is 0, but if the two vectors are parallel to each other, then $(\boldsymbol{a}^T \cdot \boldsymbol{b})/||\boldsymbol{a}|| = ||\boldsymbol{b}||$. Orthogonal vectors are also called *linearly independent*.

The *identity matrix* $\boldsymbol{I}_n = (e_{ij}) \in M(n,n;\mathbb{R})$ is the matrix with 1's along the diagonal and all other entries being 0, i.e., $e_{ii} = 1$ for all i and $e_{ij} = 0$ for all $i \neq j$. The vector $\boldsymbol{e}_k = (e_i)$ has entries that are defined as $e_k = 1$ and $e_i = 0$ for all $i \neq k$.

A subset $U \subseteq M(m,n;\mathbb{R})$ is called a *subspace* of $M(m,n;\mathbb{R})$ if

- $U \neq \emptyset$,
- for all $\boldsymbol{A}, \boldsymbol{B} \in U$, $\boldsymbol{A} + \boldsymbol{B} \in U$, and
- for all $\alpha \in \mathbb{R}$ and $\boldsymbol{A} \in U$, $\alpha\boldsymbol{A} \in U$.

As an example, consider any set of matrices $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_k \in M(m,n;\mathbb{R})$. Then

$$\langle \boldsymbol{A}_1, \ldots, \boldsymbol{A}_k \rangle = \{\sum_{i=1}^{k} \alpha_i \boldsymbol{A}_i \mid \alpha_1, \ldots, \alpha_k \in \mathbb{R}\}$$

is a subspace of $M(m,n;\mathbb{R})$. We will be mostly interested in subspaces generated by vectors, i.e., subspaces of the form $U = \langle \boldsymbol{a}_1, \ldots, \boldsymbol{a}_k \rangle$ with $\boldsymbol{a}_i \in \mathbb{R}^n$ for every i. We say that $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k$ form a *basis* of U if they are linearly independent, i.e., $\boldsymbol{a}_i \perp \boldsymbol{a}_j$ resp. $\boldsymbol{a}_i^T \cdot \boldsymbol{a}_j = 0$ for all $i \neq j$. In this case, the *dimension* of a subspace U is equal to k. Two subspaces U_1 and U_2 are *linearly independent*, or $U_1 \perp U_2$, if and only if any two vectors $\boldsymbol{a} \in U_1$ and $\boldsymbol{b} \in U_2$ are linearly independent.

Matrices and vectors can also be defined over the complex domain. The *conjugate transpose* $\boldsymbol{A}^\dagger$ of a matrix $\boldsymbol{A} \in M(m,n;\mathbb{C})$ is defined by taking the transpose of $\boldsymbol{A}$ and conjugating all of its entries. A matrix $\boldsymbol{A} \in M(n,n;\mathbb{C})$ is called *unitary* if and only if $\boldsymbol{A}^\dagger \cdot \boldsymbol{A} = \boldsymbol{I}_n$.

The square norm of a vector $\boldsymbol{a} = (a_i) \in \mathbb{C}^n$ is defined as

$$||\boldsymbol{a}||^2 = \sum_{i=1}^{n} ||a_i||^2$$

Similar to the real numbers, given any two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, the value $(\boldsymbol{a}^\dagger \cdot \boldsymbol{b})/||\boldsymbol{a}|| \in \mathbb{C}$ is equal to the complex value of the projection of $\boldsymbol{b}$ onto $\boldsymbol{a}$.

Dirac notation

In the quantum physics literature, people often use the Dirac notation, or bracket notation, when dealing with vectors. Given a vector $\boldsymbol{a} \in \mathbb{C}^n$, the *bra* of $\boldsymbol{a}$ is denoted as $\langle \boldsymbol{a}|$ and represents $\boldsymbol{a}^\dagger$ (i.e., the conjugate transpose of $\boldsymbol{a}$) whereas the *ket* of $\boldsymbol{a}$ is denoted as $|\boldsymbol{a}\rangle$ and simply represents $\boldsymbol{a}$. Finally, the *bracket* of $\boldsymbol{a}$ is equivalent to its square norm, namely,

$$\langle \boldsymbol{a}|\boldsymbol{a}\rangle = \langle \boldsymbol{a}| \cdot |\boldsymbol{a}\rangle = \boldsymbol{a}^\dagger \cdot \boldsymbol{a} = ||\boldsymbol{a}||^2 .$$

For all vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{C}^n$ it holds:

- $\langle \boldsymbol{a}|\boldsymbol{b}\rangle = \langle \boldsymbol{b}|\boldsymbol{a}\rangle^*$,
- $\langle \boldsymbol{a}|\alpha\boldsymbol{b}\rangle = \alpha\langle \boldsymbol{a}|\boldsymbol{b}\rangle$ for all $\alpha \in \mathbb{C}$,
- $\langle \boldsymbol{a}+\boldsymbol{b}|\boldsymbol{c}\rangle = \langle \boldsymbol{a}|\boldsymbol{c}\rangle + \langle \boldsymbol{b}|\boldsymbol{c}\rangle$, and
- $\langle \boldsymbol{a}|\boldsymbol{b}+\boldsymbol{c}\rangle = \langle \boldsymbol{a}|\boldsymbol{b}\rangle + \langle \boldsymbol{a}|\boldsymbol{c}\rangle$.

Because $\langle \boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\rangle = \mathbb{C}^n$, i.e. the unit vectors $\boldsymbol{e}_i$ over the complex numbers form a basis of $\mathbb{C}^n$, it holds that any vector $\boldsymbol{a} = (a_i) \in \mathbb{C}^n$ can be given as a linear combination of the unit vectors $\boldsymbol{e}_k \in \mathbb{C}^n$. More precisely,

$$|\boldsymbol{a}\rangle = \sum_{i=1}^{n} a_i|\boldsymbol{e}_i\rangle = \sum_{i=1}^{n}\langle \boldsymbol{e}_i|\boldsymbol{a}\rangle|\boldsymbol{e}_i\rangle$$

In general, it holds for any basis $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ of $\mathbb{C}^n$ that

$$|\boldsymbol{a}\rangle = \sum_{i=1}^{n} \frac{\langle \boldsymbol{b}_i|\boldsymbol{a}\rangle}{||\boldsymbol{b}_i||^2} \cdot |\boldsymbol{b}_i\rangle$$

because $\langle \boldsymbol{b}_i|\boldsymbol{a}\rangle/||\boldsymbol{b}_i||$ is the (complex) length of the projection of $\boldsymbol{a}$ onto $\boldsymbol{b}_i$ and we have to divide by another $||\boldsymbol{b}_i||$ because we multiply the term with $|\boldsymbol{b}_i\rangle$.

Now we have all the necessary ingredients to view computations by classical and quantum computers as operations in a linear algebra.

10.2 Classical computing

All known computational models are based on two kinds of operations acting on some *state space* Ω: *computations* and *observations.*

The most general model of a classical computer is the probabilistic Turing machine. Consider any probabilistic Turing machine M. Let B denote the set of all possible configurations or basis states of M and let δ be the transition function of M. The state of a probabilistic Turing machine can be an arbitrary probability distribution on the basis states. Hence, the state space $\Omega \subseteq [0,1]^B$ of M is defined as

$$\Omega = \{\boldsymbol{p} = (p_C) \in [0,1]^B \mid |p| = \sum_{C \in B} p_C = 1\}$$

Furthermore, δ can be characterized as a matrix $\boldsymbol{A}_\delta = (a_{C,C'})$ where $a_{C,C'}$ denotes the probability of moving from configuration C' to C in one step. Hence, $\boldsymbol{a}_{C'} = (a_{C,C'})_C$ is equal to $\delta(C')$ for all C'. Since $\delta(C') \in \Omega$ for all C' and therefore $\sum_C a_{C,C'} = 1$, $\boldsymbol{A}_\delta$ is *stochastic*.

Given that M starts with distribution $\boldsymbol{p}_0$, the definition of $\boldsymbol{A}_\delta$ implies that the state of M at time t is equal to

$$\boldsymbol{A}_\delta^t \cdot \boldsymbol{p}_0$$

Besides computations we can also define observations for a probabilistic Turing machine. Suppose that the Turing machine is currently in state $\boldsymbol{p}$ and we observe a certain collection of cells on the tape. Then there is a collection of alternative outcomes of this observation. Let the outcomes be denoted by $E_1, \ldots, E_k$. Treating these outcomes as events, i.e., subsets of B, it must hold for them that

- $E_i \cap E_j = \emptyset$ for all $i \neq j$ and
- $E_1 \cup \ldots \cup E_k = B$

to form a complete, valid set of outcomes.

As an example, suppose that cell c is examined on the tape. For any tape symbol a, let the event E_a be defined as the set of all configurations in B that have cell c set to a. Then it can be verified that this collection of events indeed satisfies the requirements above.

In the linear algebra terminology, the outcomes $E_1, \ldots, E_k$ can be specified as subspaces $U_1, \ldots, U_k$ of Ω. More precisely, we define

$$U_i = \langle \boldsymbol{e}_C : \; C \in E_i \rangle \; ,$$

that is, U_i is generated by all unit vectors $\boldsymbol{e}_C$ representing configurations that belong to E_i. In this case, if $E_1, \ldots, E_k$ satisfy the conditions of a complete, valid set of outcomes, then

- $U_i \perp U_j$ for all $i \neq j$ and
- $U_1 \times \ldots \times U_k = \mathsf{IR}^B$.

The former condition is true because $\boldsymbol{e}_C^T \cdot \boldsymbol{e}_{C'} = 0$ for any pair C, C' with $C \in E_i$, $C' \in E_j$, and $i \neq j$, and the former condition is true because

$$U_1 \times \ldots \times U_k = \langle \boldsymbol{e}_C : \; \exists i \; C \in E_i \rangle = \langle \boldsymbol{e}_C : \; C \in B \rangle = \mathsf{IR}^B \; .$$

Suppose now that we observe outcome E_1. Then the state of the Turing machine *collapses* to E_1, which means that only those configurations C with $C \in E_1$ survive. In this case, when starting with probability distribution $\boldsymbol{p}$, we are left with

$$\boldsymbol{p}_1 = \sum_{C \in E_1} p_C \boldsymbol{e}_C \; .$$

However, this may not be a valid probability distribution, and therefore, we have to make sure that we normalize $\boldsymbol{p}_1$ to a vector $\boldsymbol{p}'$ with $|\boldsymbol{p}'| = \sum_{C \in E_1} p'_C = 1$. But this is easy:

$$\boldsymbol{p}' = \frac{1}{|\boldsymbol{p}_1|} \sum_{C \in E_1} p_C \boldsymbol{e}_C \ .$$

10.3 Quantum computing

The important difference between quantum computing and classical computing is that the state of quantum computers is defined over the complex space instead of the real space. Young's celebrated two-slit experiment will serve as a background to illustrate this phenomenon.

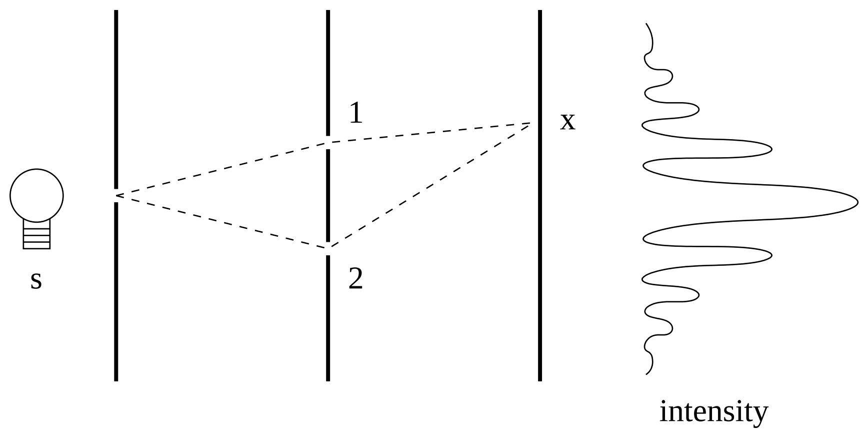

Figure 1: Young's two-slit experiment.

In Young's experiment (Figure 1), light coming out of a hole in the left wall must go through two small holes in the center wall. A detector on the right wall measures the light intensity at different positions along the length of the wall. If only one hole is open, the intensity reaches its maximum at a position directly in line with that hole and the source s. As the detector moves away from that position, the intensity slowly fades and eventually vanishes. When *both* holes are open, the intensity pattern is *not* the sum of the two one-hole intensities, as one would expect, but an alternation of bright and dark fringes. This effect is caused by the *interference* of the light coming out from both holes. Surprisingly, the interference persists even when the source s is dim enough to send only one photon at a time: if many runs are made and a photon count is kept for various positions, the same pattern of bright and dark fringes appears. Each photon seems to interfere with itself.

The self-interference appearing in Young's experiment is just one example illustrating that classical intuition cannot be applied to quantum systems. Basically, instead of probabilities over over the real domain, one has to argue with *probability amplitudes* over the complex domain. Using such probability amplitudes would allow us to explain the outcome of Young's experiment.

In a quantum computer, we also have a discrete set B of possible configurations or basis states. However, the *state* of a quantum computer can be any vector $\boldsymbol{p} \in \mathbb{C}^B$ with $||\boldsymbol{p}||^2 = 1$.

That is, our set Ω is now defined as

$$\Omega = \{\boldsymbol{p} = (p_C) \in \mathbb{C}^B \mid ||p||^2 = \sum_{C \in B} ||p_C||^2 = 1\}$$

Using Dirac notation, a vector $\boldsymbol{p} \in \mathbb{C}^B$ is also written as

$$|\boldsymbol{p}\rangle = \sum_{C \in B} p_C |\boldsymbol{e}_C\rangle = \sum_{C \in B} p_C |C\rangle$$

i.e., we identify the unit vector $\boldsymbol{e}_C$ with its configuration C.

The transition function δ of a quantum computer can specify a complex probability amplitude $a_{C,C'}$ for any move from C' to C in one step. We can organize all these amplitudes in a matrix $\boldsymbol{A}_\delta = (a_{C,C'})$. However, in order for $\boldsymbol{A}_\delta$ to represent a valid computation, it must preserve the square norm. More precisely, for any $\boldsymbol{p} \in \Omega$ it must hold that

$$||\boldsymbol{A}_\delta \cdot \boldsymbol{p}||^2 = 1$$

It turns out that this property is satisfied if and only if $\boldsymbol{A}_\delta$ is unitary, i.e., $\boldsymbol{A}_\delta^\dagger \cdot \boldsymbol{A}_\delta = I$. This is true because if $\boldsymbol{A}_\delta$ is unitary, then

$$||\boldsymbol{A}_\delta \cdot \boldsymbol{p}||^2 = \langle \boldsymbol{A}_\delta \cdot \boldsymbol{p} | \boldsymbol{A}_\delta \cdot \boldsymbol{p}\rangle = \langle \boldsymbol{p} | \boldsymbol{A}_\delta^\dagger \cdot \boldsymbol{A}_\delta | \boldsymbol{p}\rangle = \langle \boldsymbol{p} | \boldsymbol{p}\rangle = ||\boldsymbol{p}||^2$$

Since every valid transition must be unitary, it follows that every transition must be reversible. In fact, the transition reversing $\boldsymbol{A}_\delta$ is $\boldsymbol{A}_\delta^\dagger$. This, at first glance, seems to limit quantum computing, because not all computations in classical computers are reversible, but as we will see later, this requirement is not a limitation.

Besides computations, we can also define observations for quantum computers. In fact, observations can be modeled in exactly the same way as for classical computers, with the only difference that now we have to make sure that any vector $\boldsymbol{p}$ describing the state of the system must have a square norm of 1, i.e., $||\boldsymbol{p}||^2 = 1$. That is, the vector $\boldsymbol{p}_1$ in our example for classical computers has to be normalized to

$$|\boldsymbol{p}'\rangle = \frac{1}{||\boldsymbol{p}_1||} \sum_{C \in E_1} p_C |\boldsymbol{e}_C\rangle = \frac{1}{||\boldsymbol{p}_1||} \sum_{C \in E_1} p_C |C\rangle \ .$$

Furthermore, the probability that the observation gives E_1 is $||\boldsymbol{p}_1||^2$.

10.4 The quantum Turing machine

We are now able to define a quantum Turing machine.

Definition 10.1 *A* quantum Turing machine *(QTM) is denoted by*

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

where $Q, \Sigma, \Gamma, q_0, B, F$ are defined as for a probabilistic Turing machine. However, the transition function δ is now a mapping of the form

$$\delta : Q \times \Gamma \times Q \times \Gamma \times \{L, N, R\} \rightarrow \mathbb{C}$$

where $\delta(p, x, q, y, d)$ gives the amplitude with which the machine in state p reading a will write b, enter state q, and move in direction d. Furthermore, δ has to be well-formed, *that is, it always preserves the square norm.*

If we choose the set B to consist of all possible configurations of M, then δ specifies a linear mapping from B to $\mathbb{C}^B$. This linear mapping can be represented as a *transition matrix* $\boldsymbol{A}_\delta = (\alpha_{c_1,c_2})_{c_1,c_2 \in B}$ in which $\alpha_{c_1,c_2} \in \mathbb{C}$ specifies the amplitude of going to configuration c_1 given that the current configuration is c_2. Since $\boldsymbol{A}_\delta$ has to be unitary, the following conditions have to be satisfied for a transition function δ of a quantum Turing machine to be well-formed, i.e., to imply a unitary $\boldsymbol{A}_\delta$:

Theorem 10.2 *A transition function δ is well-formed if it satisfies the following conditions:*

- *The amplitude distribution leaving any state-symbol pair has a unit square norm:*

$$\forall p \in Q \ \forall x \in \Gamma : \quad \sum_{q,y,d} ||\delta(p,x,q,y,d)||^2 = 1 \ .$$

- *The amplitude distributions of written character, new state, and direction leaving any two different state-symbol pairs are orthogonal:*

$$\forall (p,x) \neq (p',x') : \quad \sum_{q,y,d} \delta(p,x,q,y,d) \cdot \delta^*(p',x',q,y,d) = 0 \ .$$

The proof of the theorem is left as an assignment. If δ is well-formed, then every transition of a quantum Turing machine is reversible (simply use $A_\delta^\dagger$ after using A_δ, and we would be back at the same state before A_δ). Note that classical computations are usually not reversible. For instance, the simple command of setting a variable back to 0 may have the effect that several configurations fall together, and therefore it is not possible to say afterwards from which configuration the Turing machine originally came. We formally define reversibility for the deterministic Turing machine as follows.

Definition 10.3 *A Turing machine M is* reversible *if reversing its arrows gives a deterministic mapping of configurations T that undoes the computation of M: for any pair of configurations c_1 and c_2, $c_1 \xrightarrow{M} c_2$ if and only if $c_2 \xrightarrow{T} c_1$.*

Note that the arrow reversal of a reversible Turing machine M is not necessarily a Turing machine, since the reverse of a transition that both writes and moves must write in a non-local cell. Hence, we only spoke about a mapping T instead of a Turing machine T in the definition.

Now, suppose we have a Turing machine M which computes a function f, i.e. it outputs $f(x)$ on input x. If f is not injective, then there can clearly be no reversible Turing machine that computes f (i.e. has $f(x)$ on its output tape and all other work tapes erased). However, Lecerf and independently Bennett showed that for any deterministic Turing machine M computing a function f there is a reversible Turing machine M' such that M' computes $x\,f(x)$ on input x. The runtime of M' is within a constant factor of the runtime of M.

A reversibility condition can be also formulated for probabilistic and quantum Turing machines: the transition function δ' resulting from δ by reversing the arrows and conjugating the amplitudes of δ undoes the computation of δ. That is, for any superpositions v_1 and v_2, $v_1 \xrightarrow{\delta} v_2$ if and only if $v_2 \xrightarrow{\delta'} v_1$. Since this is a generalization of reversibility for deterministic Turing machines and quantum Turing machines fulfill this reversibility condition (unless observations are made), it is not surprising that Benioff and Deutsch were able to show that quantum Turing machines can efficiently simulate classical reversible Turing machines.

Theorem 10.4 *Any function f that can be computed efficiently by a classical (deterministic or probabilistic) Turing machine can also be computed efficiently by a quantum Turing machine.*

Theorem 10.4 implies that quantum computing is at least as powerful as classical computing. But does there exist a quantum computer that can execute arbtrary quantum programs in an efficient way? Or in other words: is there a universal quantum Turing machine that can simulate any specific quantum Turing machine in an efficient way? The answer to this was given by Bernstein and Vazirani. They showed the following theorem.

Theorem 10.5 *There exists a universal quantum Turing machine U which when given a description of any quantum Turing machine M and any $t > 0$, simulates M for t steps with accuracy ϵ and with slowdown polynomial in t and $1/\epsilon$.*

Hence, in principle it is possible to construct a quantum computer that can execute arbitrary quantum programs. However, how susceptible would such a machine be to errors in the computation? Bernstein and Vazirani showed in their proof of Theorem 10.5 that $O(t)$ bits of accuracy are sufficient to simulate t steps of computation with accuracy ϵ. However, Lipton pointed out that in order to claim that a model of computation is truly discrete, it is necessary that any constants in the specification should only be required to be accurate up to $O(\log t)$ bits. This is because a constant that is accurate up to k bits can lie in one of 2^k distinct ranges, and the task of verifying that the machine meets the specifications seems to require effort that is proportional to 2^k. The following theorem shows that we need only $O(\log t)$ bits of precision in the specification of each of the complex numbers defining a single-step transition function of a quantum Turing machine. The theorem is due to Bennett, Bernstein, Brassard, and Vazirani. $|\phi_t^M\rangle$ in the theorem denotes the superposition of a quantum Turing machine M at time t.

Theorem 10.6 *For any two quantum Turing machines M and $\hat{M}$ with the same configuration set and tape alphabet and the property that for every transition amplitude λ of M, the corresponding transition amplitude $\hat{\lambda}$ of $\hat{M}$ satisfies $||\hat{\lambda} - \lambda||^2 \leq \epsilon$, it holds that for all inputs that $|||\phi_t^M\rangle - |\phi_t^{\hat{M}}\rangle||^2 = O(\epsilon \cdot t)$ for any time step t.*

Proof. Let U be the unitary transformation specified by M, and $\hat{U}$ be the unitary transformation specified by $\hat{M}$. We will first prove that for any superposition $|\phi\rangle$, $U|\phi\rangle$ is close to $\hat{U}|\phi\rangle$. Let

$$|\phi\rangle = \sum_i \alpha_i |c_i\rangle$$

be some superposition over configurations c_i. Then,

$$U|\phi\rangle - \hat{U}|\phi\rangle = \sum_i \left(\sum_{j \in P(i)} (\lambda_{j,i} - \hat{\lambda}_{j,i})\alpha_j \right) |c_i\rangle$$

where $P(i)$ is the set of j such that configuration $|c_j\rangle$ can lead to $|c_i\rangle$ in a single step of M or $\hat{M}$. Note that every j can occur in at most c different sets $P(i)$, where c is some constant depending only on the number of states and tape symbols used in the definition of M. So we have,

$$\begin{aligned}
||U|\phi\rangle - \hat{U}|\phi\rangle||^2 &= \sum_i || \sum_{j\in P(i)} (\lambda_{j,i} - \hat{\lambda}_{j,i})\alpha_j||^2 \;\leq\; \sum_i \sum_{j\in P(i)} ||(\lambda_{j,i} - \hat{\lambda}_{j,i})\alpha_j||^2 \\
&\leq \sum_i \sum_{j\in P(i)} ||\lambda_{j,i} - \hat{\lambda}_{j,i}||^2 ||\alpha_j||^2 \;\leq\; \sum_i \sum_{j\in P(i)} \epsilon ||\alpha_j||^2 \\
&= \epsilon \sum_i \sum_{j\in P(i)} ||\alpha_j||^2 \;\leq\; \epsilon \cdot c \sum_j ||\alpha_j||^2 \\
&\leq \epsilon \cdot c\,.
\end{aligned}$$

We will use this to prove that the superpositions of M and $\hat{M}$ are close together for any input string.

Let x be an arbitrary input string. Then we will show that the superpositions of M and $\hat{M}$ at time t, $|\phi_t^M\rangle$ and $|\phi_t^{\hat{M}}\rangle$, are close together.

$$\begin{aligned}
\hat{U}^t|x\rangle &= \hat{U}^{t-1}(U|x\rangle + |\Delta_1\rangle) \\
&= \hat{U}^{t-1}(U|x\rangle) + \hat{U}^{t-1}|\Delta_1\rangle \\
&= \hat{U}^{t-2}(U^2|x\rangle) + \hat{U}^{t-2}|\Delta_2\rangle + \hat{U}^{t-1}|\Delta_1\rangle \\
&= U^t|x\rangle + \sum_{i=1}^{t} \hat{U}^{t-i}|\Delta_i\rangle
\end{aligned}$$

where each $|\Delta_i\rangle$ has the property that $||\Delta_i||^2 \leq \epsilon \cdot c$. Therefore,

$$\begin{aligned}
|||\phi_t^M\rangle - |\phi_t^{\hat{M}}\rangle||^2 &= ||U^t|x\rangle - \hat{U}^t|x\rangle||^2 \;\leq\; ||\sum_{i=1}^{t} \hat{U}^{t-i}|\Delta_i\rangle||^2 \\
&\leq \sum_{i=1}^{t} ||\hat{U}^{t-i}|\Delta_i\rangle||^2 \;=\; \sum_{i=1}^{t} ||\Delta_i||^2 \\
&\leq t \cdot \epsilon \cdot c\,.
\end{aligned}$$

□

The theorem shows that in order to obtain a sufficiently good error bound after t steps, the error in each step has to be of order $O(1/t)$. Using error correcting codes, Shor showed that it is even possible to tolerate an error of $O(1/\log^c t)$ for some constant c in each step to obtain a sufficiently small overall error after t steps.

10.5 Quantum complexity classes

Similar to P we define the time complexity class QP to be the set of all languages L for which there exists a quantum Turing machine M with the property that

- for all $x \in L$, Pr[M accepts x] $= 1$ and
- for all $x \notin L$, Pr[M accepts x] $= 0$.

Furthermore, the class BQP is defined as the set of all languages L for which there exists a quantum Turing machine M with the property that

- for all $x \in L$, Pr[M accepts x] $\geq 2/3$ and
- for all $x \notin L$, Pr[M accepts x] $\leq 1/3$.

From the previous subsection we know that

$$\mathrm{P} \subseteq \mathrm{QP} \quad \text{and} \quad \mathrm{BPP} \subseteq \mathrm{BQP} .$$

Whether or not P = QP or BPP = BQP is still wide open. However, there is some evidence that the quantum complexity classes are more powerful than the classical counterparts. We will investigate this in more detail in the next section. An upper bound for the complexity of QP and BQP is provided by the following theorem. Its proof is left as an exercise.

Theorem 10.7 BQP $\subseteq$ PSPACE.

Only slighty better upper bounds are known for the complexity of BQP to date.

10.6 References

- C.H. Bennett. Local reversibility of computations. *IBM Journal of Research and Development* 17:525–532, 1973.
- C.H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing* 26(5):1510–1523, 1997.
- E. Bernstein and U. Vazirani. Quantum complexity theory. In *Proc. of the 25th ACM Symp. on Theory of Computing*, pp. 11–20, 1993.
- A. Berthiaume. Quantum Computation. In *Complexity Theory Retrospective II*, Springer Verlag, 1996
- D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. of the Royal Society*, London, A400:97–117, 1985.
- R.P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics* 21(6/7):467–488, 1982.
- Y. Lecerf. Machines de Turing réversibles. Récursive insolubilité en $n \in \mathbb{N}$ de l'équation $u = \Theta^n$ où Θ est un isomorphisme de codes. In *Comptes rendus de l'Académie française ds sciences* 257:2597–2600, 1963.
- P. Shor. Fault-tolerant quantum computation. In *Proc. of the 37th IEEE Symp. on Foundations of Computer Science*, pp. 56–65, 1996.