59. Quantum and Biocomputing – Common Notions and Targets

Mika Hirvensalo

Biocomputing and quantum computing are both relatively novel areas of information processing sciences under the umbrella *natural computing* established in the late twentieth century. From the practical point of view one can say that in both bio and quantum paradigms, the purpose is to replace the traditional media of computing by an alternative. Biocomputing is based on an appropriate treatment of biomolecules, and quantum computing is based on the physical realization of computation on systems so small that they must be described by using quantum mechanics. The efficiency of the proposed biomolecular computing is based on massive parallelism, which is implementable by already existing technology for small instances. In a sense, also quantum computing involves parallelism. From time to time, there are proposals or attempts to create a uniform approach to both biocomputational and quantum parallelism. The main purpose of this article is the explain why this a very challenging task. For this aim, we present the usual mathematical formalism needed to speak about quantum computing and compare quantum parallelism to its biomolecular counterpart.

59.1 Overview

In 1994 *Adleman* aroused a lot of attention by describing a biomolecular solution to the traveling salesman problem [59.1] (see also [59.2]). It is by no means exaggerated to say that Adleman actually set the establishment of a new kind of science, even though the ideas of bio-inspired computational models are far older. For example, the theoretical properties of artificial neural networks had been studied decades before, and a notable exposition was published by *Minsky* and *Papert* in 1969 [59.3]. However, the previous studies on bio-inspired computing seemed to focus on analogies of biological processes, and that is exactly where *Adleman* took one step further: he proposed that instead of simulation, it could be useful to utilize directly the biochemical processes to perform computation.

By a coincidence, another branch of new science gained a lot of attention in 1994, too. In that year Shor published his famous polynomial-time algorithm for integer factorization [59.4]. Shor's study was remarkable for several reasons. First, since antiquity, there has been an unsuccessful quest for an efficient procedure for integer factorization. Another reason making Shor's discovery important is a very practical one. The security of the broadly used RSA encryption system is based on the assumption that no efficient method for factoring integers exists. The third important feature of Shor's results was that the factoring method was designed for *quantum computers* and could not be directly described in terms of traditional *Turing machines* [59.5].

It is also true that *quantum computing* did not begin with Shor's article. The first ideas of quantum information were coined by *von Neumann* in 1927 [59.6, 7], but the first ideas of the quantum computer were introduced as late as early 1980s by *Benioff* [59.8, 9], and very notably by *Feynman* [59.10], who actually suggested that quantum computers may be more efficient that the traditional ones. The theory was further developed by *Deutsch*, who introduced quantum Turing machines [59.11] and quantum networks [59.12]. Also the first examples of the superior efficiency of quantum computers were given by *Deutsch* and *Jozsa* [59.13] in 1992. In 1994, *Simon* presented a more usable example [59.14], and Shor actually built his factoring procedure on Simon's algorithm. In 1997 *Bernstein* and *Vazirani* established quantum complexity theory using an improved version of Deutsch's quantum Turing machine [59.15].

Since the early days, the theory of quantum computing has been under strong development, but many basic questions still remain unresolved. For instance, we know that factoring integers would be feasible by quantum computers, but we cannot *prove* it unfeasible for classical computers. All we know is that no-one has discovered any feasible factoring for classical computers. In fact we do not know for sure of *any* problems for which quantum computers could provably be more powerful. This lack of knowledge becomes understandable when noticing that many basic questions on classical computing remain unresolved, too: as of 2013, we do not know whether polynomial-time nondeterministic computing any more powerful than its deterministic counterpart. This so-called P versus NP-problem is generally acknowledged as one of the most difficult problems in contemporary mathematics [59.16], and there is no reason to believe that analogous problems on the relations between quantum and classical computing were any simpler to resolve.

Quantum computing has an important common feature with molecular computing: in both paradigms, the idea is to perform computation on non-traditional hardware. To run quantum algorithms requires a quantum computer, i. e., a computer capable of storing and handling quantum information. For that purpose, the information should be presented by using physical systems so small that the quantum effects occur. From time to time, the analogies between quantum and biomolecular computing encourage authors to submit an idea of joint computing model. However, the success of such models has been very limited so far, and the main purpose of this article is to explain why to a reader only weakly familiar with quantum computing concepts. This will be done by introducing the basic notions of quantum computing in a superficial way and pointing out the essential differences between the two computing paradigms. One single and perhaps the most essential difference between the computational paradigms can be introduced without presenting any deeper structures. The objects of molecular computing (DNA molecules) are microscopic to humans, but yet macroscopic from the quantum computing perspective. Hence the information in molecular computing is treated as classical information, whereas the starting point of quantum computing is the *quantumness* of information.

As a secondary purpose, we describe briefly the four types of existing quantum algorithms and some restrictions of quantum computing to present an idea of what can be achieved by using quantum computing. The types presented cover almost all known quantum algorithms.

59.2 Biomolecular and Quantum Parallelism

Adleman designed and expressed his algorithm for the traveling salesman problem (actually Adleman's formulation was a problem which should be called the Hamiltonian path problem) by using biomolecular operations. Anyone interested can learn details about those operations in [59.1], but roughly speaking, *Adleman*'s procedure can be described as follows. The problem itself is to decide whether in a given directed graph (*city map*) there is a path beginning and ending at fixed vertices and visiting every vertex exactly once. Adleman's solution was to encode each vertex and edge into a single-stranded DNA-sequence in such a way, that if there is an edge e from vertex c_1 to c_2, then *half* of the strand encoding c_1 is complementary to a half of the strand encoding e, and the latter half of e is complementary to a half encoding c_2. As there may be multiple incoming and outgoing edges, there may also be multiple encodings of cities. In a test tube containing multiple copies of DNA strands encoding both vertices and edges, the single (let us call them *lower*) strands $c_1, c_2, \ldots$ tend to form longer strands $c_{i_1}c_{i_2}c_{i_3}\ldots$ bounded by the *upper* (single) strands $e_{i_1}e_{i_2}\ldots$ (e_{i_1} extends over c_{i_1} and c_{i_2}, e_{i_2} over c_{i_2} and c_{i_3}, etc.) The encoding ensures that c_{i_j} and $c_{i_{j+1}}$ will be adjacent only if there is an edge from c_{i_j} to $c_{i_{j+1}}$. This means that the chemical tendency of DNA sticking to its complementary counterpart will generate various double-stranded DNA sequences encoding paths in the graph that was originally encoded.

The problem is then to detect whether a strand encoding a desired path exists. Using electrophoresis it is possible to filter out DNA strands of wrong length, and additional existing techniques suffice to filter out exactly the desired paths, if there are any. The crucial point is that it is known how to the duplicate the existing DNA. Once the encodings are available, they can be duplicated to the extent guaranteeing that the above procedure will detect a desired solution with high probability, if any exists.

The above description emphasizes that Adleman's DNA-based solution actually utilizes heavy parallelism. A test tube containing multiple copies of vertex and edge encodings acts like a nondeterministic device generating potential paths, and the problem is to filter out the desired path – a solution, if any exists. An existing biotechnique is sufficient to do the filtering.

In quantum computing, there also occurs parallelism – quantum parallelism, which will be described in the rest of this chapter. The presentation here will be merely informal, but the notions will be defined in the next chapter. It is not necessary to focus on any specific NP-complete problem, and we choose to study a general version.

59.2.1 A General NP-Complete Problem

- Input: $N \in \mathbb{N}$ and $f : \{0, 1\}^N \to \{0, 1\}$ a polynomial-time (in N) computable function.
- Output:

$$\begin{cases} 1, & \text{if there is } \boldsymbol{x} \in \{0,1\}^N \quad \text{so that } f(\boldsymbol{x}) = 1\,, \\ 0, & \text{otherwise}\,. \end{cases}$$

Here and hereafter, $\{0, 1\}^N$ stands for the bit strings of length N, so a general NP-complete problem is typically a search problem: one has to decide whether there is an N-bit string $\boldsymbol{x}$ so that $f(\boldsymbol{x}) = 1$. A solution to this problem can evidently be obtained via exhaustive search, but that is computationally expensive: try all 2^N candidates $\boldsymbol{x} \in \{0, 1\}^N$ and check if any of them satisfies $f(\boldsymbol{x}) = 1$. By assumption, any value $f(\boldsymbol{x})$ can be computed in polynomial time (in N), but there are exponentially many (2^N) possibilities to be checked.

In quantum computing, it is possible to form a state

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \frac{1}{\sqrt{2^N}} |\boldsymbol{x}\rangle\,, \tag{59.1}$$

so-called *superposition* of all bit strings $\boldsymbol{x} \in \{0, 1\}^N$, and then to compute function f on all possible inputs simultaneously by a cost of single computation to obtain

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \frac{1}{\sqrt{2^N}} |\boldsymbol{x}\rangle |f(\boldsymbol{x})\rangle\,. \tag{59.2}$$

This is what quantum parallelism means: all values $\boldsymbol{x} \in \{0, 1\}^N$ occur, in a sense, parallel in (59.1), and all values $f(\boldsymbol{x})$ are, again in a sense, computed simultaneously.

However, this so-called quantum parallelism is very different from the biomolecular parallelism described earlier. In (59.1) there are no 2^N physical systems each consisting of N bits, but only a single physical system of N bits in a *state*, which allows, in a sense, an interpretation as any $\boldsymbol{x} \in \{0, 1\}^N$. Quantum parallelism is not comparable to DNA computing parallelism, and the incomparability is underlined by the physical interpretation of (59.2). Observation of (59.2) will give any pair $(\boldsymbol{x}, f(\boldsymbol{x}))$, each with probability $\left|1/\sqrt{2^N}\right|^2 = \frac{1}{2^N}$, but on a measurement, (59.2) is destroyed irreversibly.

It is worth noticing that the quantum parallelism, as described above, is not far apart from *probabilistic parallelism*: Toss N times a fair coin to obtain a random bit string $\boldsymbol{x} \in \{0, 1\}^N$, then compute $f(\boldsymbol{x})$. A string $\boldsymbol{x}$ with the property $f(\boldsymbol{x}) = 1$ (if any exists) will be found exactly with the same probability as observing (59.2) would give. Hence the problem with straightforward quantum parallelism (59.2) is the same as with the *probabilistic parallelism*. If there are only a few, or even only one string $\boldsymbol{x}$ (let us call it *solution*) such that $f(\boldsymbol{x}) = 1$, then the cases *solution exists* and *no solution* cannot be distinguished from each other with any better probability than $\frac{1}{2^N}$.

However, as *Deutsch* and *Josza*'s [59.13], *Simon*'s [59.14], and *Shor*'s [59.4] discoveries demonstrated, quantum computing offers possibilities to solve *some* problems more efficiently than any known classical procedure allows. But it may be useful to underline right now that it is strongly believed (although not proved) that quantum computers cannot solve NP-complete problems in polynomial time.

59.3 Quantum Computing Preliminaries

In this chapter, the basics of quantum information will be presented only very superficially, and a reader desiring more detailed exposition is advised to consult [59.17,18] or [59.19]. To understand the formalism, it is necessary to accept the fact that quantum mechanics is a stochastic theory, meaning that in general, the complete description of the system, *the state*, cannot in general result into any deterministic description, only a probability distribution over potential outcomes. The probabilistic structure of quantum mechanics is very well studied, and lots of results are already available. We could ask, for instance, whether there could be a deterministic theory lying under quantum mechanics, and the probability distribution is only due to unknown boundary values (hidden variables). For instance, when tossing a classical coin, one could imagine that if the initial circumstances are known precisely enough, one could be always predict the outcome.

A deep investigation has shown that the quantum randomness cannot emerge from any deterministic procedure as described above, but randomness in inherently inseparable feature of quantum mechanics (see [59.20], for instance).

59.3.1 Hilbert Space Basic Structure

The *Hilbert space* formalism of quantum computing is usually based on *pure states* and requires some basic notions. *An n-level quantum system* means a (quantum) physical system with n different states which are mutually distinguishable with certainty. *An n-dimensional Hilbert space* H_n is a complex vector space $\mathbb{C}^n$ equipped with a Hermitian inner product $\langle \boldsymbol{x}|\boldsymbol{y}\rangle = x_1^* y_1 + \ldots + x_n^* y_n$. The inner product induces a norm $\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}|\boldsymbol{x}\rangle}$. For any element $(x_1, \ldots, x_n) \in \mathbb{C}^n$, a *A ket-vector* is defined as column vector ($n \times 1$-matrix)

$$|\boldsymbol{x}\rangle = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} ,$$

and a bra-vector as $1 \times n$-matrix (row vector)

$$\langle \boldsymbol{x}| = \left(x_1^*, x_2^*, \ldots, x_n^*\right) .$$

Usually $\mathbb{C}^n$ is identified with the space of ket-vectors (n×1-matrices), and we say that H_n is the *state space* of the quantum system. The mathematical description of an n-level quantum system is based on n-dimensional Hilbert space in the following way: an orthonormal basis $\{|\boldsymbol{x}_1\rangle, \ldots, |\boldsymbol{x}_n\rangle\}$ is fixed as a *computational basis*, and a general state of the system is presented as a *superposition* of computational states

$$\alpha_1|\boldsymbol{x}_1\rangle + \cdots + \alpha_n|\boldsymbol{x}_n\rangle , \tag{59.3}$$

so that $|\alpha_1|^2 + |\alpha_2|^2 + \cdots + |\alpha_n|^2 = 1$. In other words, a general state of an n-level system can be represented as unit-length vectors in H_n. Basically any basis for H_n could be chosen for representation (59.3), but some bases may preferred because of the physical implementation. Hence the term *computational basis* should not be understood as any mathematical definition, but as a chosen reference basis. States of a computational basis are also called *basis states* and complex coefficients α_i *amplitudes*.

An *observable* of quantum system H_n is a collection of mutually orthogonal subspaces $\{V_1, \ldots, V_k\}$ so that $H_k = V_1 \oplus \cdots \oplus V_k$. The intuitive meaning of the notion is that each subspace V_i refers to a physical property the system can have. For example, the computational basis itself induces an observable $\{L(\boldsymbol{x}_1), \ldots, L(\boldsymbol{x}_n)\}$, where $L(\boldsymbol{x}_i)$ stands for the subspace generated by $\boldsymbol{x}_i$.

The *minimal interpretation* of quantum physics is an axiom connecting the mathematical structure to the real world. For this representation, it is sufficient to introduce the minimal interpretation in the following way: let $\{V_1, \ldots, V_k\}$ be an observable and $\boldsymbol{x} = \alpha_1\boldsymbol{x}_1 + \cdots + \alpha_k\boldsymbol{x}_k$ a presentation of state $\boldsymbol{x}$ so that $\boldsymbol{x}_i \in V_i$ and $\|\boldsymbol{x}_i\| = 1$ for each i. Then the probability that quantum system in state $\boldsymbol{x}$ is seen to have property V_i is

$$\mathbb{P}(i) = |\alpha_i|^2 . \tag{59.4}$$

It may be worth mentioning here that usually an extra element is associated to an observable: a real number λ_i to each subspace V_i. Number λ_i is the observable value, and equation (59.4) should read as

$$\mathbb{P}(\lambda_i) = |\alpha_i|^2 ,$$

meaning that the probability that the measured value of the observable is λ_i equals $|\alpha_i|^2$, However, when just studying quantum computation, it is not usually necessary to address explicit values $\lambda_i \in \mathbb{R}$, but it is enough to identify λ_i with its index i.

According to the *projection postulate*, the quantum system *collapses* to the observed state, and the superposition is irreversibly lost. That is, if property i was observed, then the state immediately after the observation is $\boldsymbol{x}_i$. The projection postulate is among the most problematic features in quantum mechanics, but this article cannot be extended to treat that specifically.

Example 59.1: A two-level quantum system is referred as to a *quantum bit*, or *qubit* for short. We fix an orthonormal computational basis $|0\rangle = (1, 0)^{\mathrm{T}}$, $|1\rangle = (0, 1)^{\mathrm{T}}$ for H_2 (T stands for transposition), and a general state of a quantum bit is a vector

$$\alpha|0\rangle + \beta|1\rangle \,, \tag{59.5}$$

where $|\alpha|^2 + |\beta|^2 = 1$ (meaning that the length of (59.5) is 1). Let $C = \{L(|0\rangle), L(|1\rangle)\}$ be an observable consisting of two subspaces generated by $|0\rangle$ and $|1\rangle$, respectively, and $C' = \{L(|0'\rangle), L(|1'\rangle)\}$ another observable, where $|0'\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1'\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. If a quantum bit is in state (59.5), and observable C is measured, then 0 is seen with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$.

Hence a qubit state (59.5) may look like a generalized probability distribution, but that is not the case. It is perfectly possible to measure also observable C' in state (59.5), and the outcome may be totally different. In fact,

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = 1 \cdot |0'\rangle + 0 \cdot |1'\rangle \,, \tag{59.6}$$

so measuring observable C results in 0 with probability $\frac{1}{2}$ and 1 with probability $\frac{1}{2}$. On the other hand, measuring observable C' results in $0'$ with probability 1 and $1'$ with probability 0. This is to emphasize that the state of a quantum system cannot be treated as a probability distribution. In fact, for every state of a finite-level (excluding the trivial case $n = 1$) quantum system there is a nontrivial (i. e., with more than 1 potential values) observable so that a single value will be observed with probability 1.

Based on the above definitions, we can now clarify the role of the *computational basis* a little bit. In fact, observing state (59.3) would generally require specification of the observable to be measured, but it is traditional to use the terminology *observing a state*, if the observable is induced by the computational basis.

59.3.2 Compound Systems

The states of a quantum system consisting of two distinguishable subsystems can be presented by using a *tensor product construction*. For the purposes of this article, it is not necessary to define the notion of tensor product exactly, it is enough just to know that the tensor product is (essentially) associative and distributive, but a non-commutative product of vectors obeying the obvious scalar rules. For more details, see [59.17] or [59.19]. It is also worth emphasizing that the counterpart of the tensor product in *concrete* objects such as matrices is the *Kronecker product*.

Now if H_m and H_n are the state spaces of m- and n-level quantum systems with computational bases $\{|\boldsymbol{x}_1\rangle, \ldots, |\boldsymbol{x}_m\rangle\}$ and $\{|\boldsymbol{y}_1\rangle, \ldots, |\boldsymbol{y}_n\rangle\}$, then the state space of the compound system is mn-dimensional if tensor product $H_m \otimes H_n$, whose computational basis can be chosen as

$$\{|\boldsymbol{x}_i\rangle \otimes |\boldsymbol{y}_j\rangle \mid (i, j) \in \{1, \ldots, m\} \times \{1, \ldots, n\}\} \,.$$

It is common to use shorthand notations $|\boldsymbol{x}_i\rangle \otimes |\boldsymbol{y}_j\rangle = |\boldsymbol{x}_i\rangle|\boldsymbol{y}_j\rangle = |\boldsymbol{x}_i, \boldsymbol{y}_j\rangle$, (even $|\boldsymbol{x}_i\boldsymbol{y}_j\rangle$ is used if there is no danger of confusion) so the state of the compound system can be represented as

$$\sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_{ij}|\boldsymbol{x}_i, \boldsymbol{y}_j\rangle \,,$$

where

$$\sum_{i=1}^{m}\sum_{j=1}^{n} |\alpha_{ij}|^2 = 1 \,. \tag{59.7}$$

It is clear that the observables of subsystems give raise to observables of the compound system.

State (59.7) is called *decomposable* if it can be presented as a product state

$$\left(\sum_{i=1}^{m} \alpha_i|\boldsymbol{x}_i\rangle\right)\left(\sum_{j=1}^{n} \beta_j|\boldsymbol{y}_j\rangle\right) \,,$$

otherwise, the state is called *entangled*.

Example 59.2: A two-qubit state

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$$

is decomposable, as

$$\frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \,.$$

On the other hand, a two-qubit state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{59.8}$$

is entangled, since assumption

$$\begin{aligned}&\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\\ &= (\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle)\\ &= \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle\end{aligned}$$

leads into equations $\alpha_0\beta_0 = \alpha_1\beta_1 = \frac{1}{\sqrt{2}}$ and $\alpha_0\beta_1 = \alpha_1\beta_0 = 0$, which are clearly impossible.

Entangled state (59.8) is historically and philosophically of great interest. Indeed, *Bohm* used [59.21] a state analogous to it to reformulate an apparent paradox of quantum mechanics introduced by *Einstein*, *Podolsky*, and *Rosen* [59.22]. That formulation eventually led *Bell* to present a resolution of the paradox [59.23] (for an exposition, see [59.20]). State (59.8) is called *an EPR state*, and a pair of qubits in state (59.8) an *EPR pair* for the aforementioned reasons.

The minimal interpretation implies directly that if the state (59.8) is observed (i. e., observable generated by the computational basis is measured), then we will see "00" with probability of $\frac{1}{2}$, and "11" with probability of $\frac{1}{2}$, too. Hence the quantum bits in an EPR state (59.8) are perfectly correlated; when observed, they always have the same value, which, however, can be 0 or 1, either with probability $\frac{1}{2}$.

It has been experimentally demonstrated that the correlation of the EPR pairs as described above is detectable even if the two physical systems (quantum bits) are spatially separated by 144 km [59.24]. However, the correlation over distance should not be surprising or anything specific to quantum physics; it is left to the reader to describe a non-quantum bipartite system with distant correlations analogous to the EPR state.

Whereas the correlation itself is not specific to quantum mechanics, the *violation of Bell inequalities* is [59.20], for instance. Violation of Bell inequalities has been experimentally detected over a physical distance of 144 km [59.24].

The mathematical description of compound systems with more than 2 subsystems is again based on tensor product construction. In this article, we will not focus on details, but will merely present an example.

Example 59.3: A system of N quantum bits has its description in a state space $H_2 \otimes \cdots \otimes H_2$, a Hilbert space isomorphic to H_{2^N}. A general state of H_{2^N} can be described as

$$\sum_{x\in\{0,1\}^N} \alpha_{x}|x\rangle\ ,$$

where

$$\sum_{x\in\{0,1\}^N} |\alpha_{x}|^2 = 1\ .$$

State

$$\begin{aligned}&\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\cdots\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\\ &= \frac{1}{\sqrt{2^N}}\sum_{x\in\{0,1\}^N}|x\rangle\end{aligned} \tag{59.9}$$

presents a uniformly distributed superposition over all basis states $|x\rangle$. It is worth noticing that presentation (59.9) shows that the state is clearly decomposable.

59.3.3 Quantum Operations

It was explained in the previous sections, in a very simplified way, how to present quantum information in pure states. It is, however, clear that the stagnant pictures of quantum states are not sufficient for using the theory. Instead, it is necessary to describe how quantum systems change in time. For most quantum computing models, and also for this article, it is sufficient to describe *closed* quantum system transformations, which will be mathematically formalized as follows: a *(closed) quantum system state transformation* is a *unitary* mapping $H_n \to H_n$. A linear mapping U is *unitary*, if $U^*U = UU^* = I$ (identity mapping), where U^* is the complex conjugate of the transpose of U. A closed quantum system state transformation is also called a *quantum gate*, see [59.25] for a study on quantum gates.

Example 59.4:

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

If is straightforward to verify that $H^* = H$, and that $H^*H = HH^* = HH = I$, meaning that H is unitary. H is hence a *unary* quantum gate, i. e., a gate on one qubit. The action of H on computational basis $\{|0\rangle = (1,0)^{\mathrm{T}}, |1\rangle = (0,1)^{\mathrm{T}}\}$ is given by

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

Gate H is called a *Hadamard transform* or a *Walsh transform*.

Example 59.5: Mapping

$$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

can be easily verified to be unitary. C is a binary gate called *controlled not*, and its name is justified by computing its action. First, matrix presentations of $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$ are obtained by using the Kronecker product

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix},$$

$$|01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix},$$

and

$$|11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

The action of C is then easy to compute: $C|00\rangle = |00\rangle$, $C|01\rangle = |01\rangle$, $C|10\rangle = |11\rangle$, and $C|11\rangle = |10\rangle$, meaning that the second qubit is flipped exactly when the first, the control bit, equals 1.

It is possible to establish quantum computing on quantum gates only, and that would lead into *quantum circuit formalism* [59.17, 18]. On the other hand, there are also other possible ways to establish the formalism of quantum computing. In fact, for any classical model for computing, there is a canonical way of transforming it into a quantum version. For example, for the definitions of quantum finite automata, see [59.26, 27] and for quantum Turing machines, see [59.17] or [59.19]. It must, however, be emphasized that unitary mappings are invertible by definition, and therefore all quantum computing models based on them are reversible. For quantum Turing machines the reversibility does not bring any disadvantage, as it is well known that all computation can be made reversible by introducing extra space [59.28]. On the other hand, many unitary models of finite automata are strictly weaker than the traditional one [59.26, 27], just because the transformation into a reversible machine would require extra space. The unitarity can be relaxed by using *open* system transforms, but representing them would require too much space in this article. See [59.29] for an automaton model with open time evolution.

Let us now revisit quantum parallelism and add more details. If $f : \{0,1\}^N \to \{0,1\}$ is computable in polynomial time, there is also a polynomial-size quantum circuit computing f. In fact, an algorithm computing f can be efficiently turned into a quantum circuit that computes f [59.17]. Usually it is necessary to add some auxiliary bits to bypass the reversibility requirement, but those extra bits are not usually written down explicitly. This implies that it is possible to construct a unitary mapping U_f by using a polynomial number of simple quantum gates (selected from a finite set) with the following action

$$U_f|\boldsymbol{x}\rangle|0\rangle = |\boldsymbol{x}\rangle|f(\boldsymbol{x})\rangle\,,$$

where $\boldsymbol{x}$ is a sequence (register) of N quantum bits. Applying the Hadamard transform to N first quantum bits in state

$$|\mathbf{0}\rangle|0\rangle$$

will result into state

$$\frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{x}\in\{0,1\}^N} |\boldsymbol{x}\rangle|0\rangle\,,$$

(59.9), and a further application of U_f will lead into state

$$\frac{1}{\sqrt{2^N}} \sum_{\boldsymbol{x}\in\{0,1\}^N} |\boldsymbol{x}\rangle|f(\boldsymbol{x})\rangle\,, \tag{59.10}$$

an equally balanced superposition over all potential pairs $(\boldsymbol{x}, f(\boldsymbol{x}))$. This is the state (59.2) of a previous example. From the computational complexity point of view, it is important to realize that the state (59.10) can be generated by N Hadamard gate actions plus the number of quantum gates required to implement U_f (polynomial in N). This is exactly what quantum parallelism means: by a polynomial number of actions it is possible to generate state (59.10) extending over exponentially many basis states.

Unfortunately (59.10) is only a mathematical description of a state of a physical system. In particular, the exponentially many values do not exist physically observable to us, but observing (59.10) will give only a single pair $(\boldsymbol{x}, f(\boldsymbol{x}))$, each with probability $\frac{1}{2^N}$, and observation will make (59.10) to collapse into state $|\boldsymbol{x}\rangle|f(\boldsymbol{x})\rangle$.

Hence it is not possible to use quantum parallelism, at least not in this straightforward way, to resolve efficiently NP-complete problems.

59.4 Quantum Algorithms

The last example of the previous section clearly justifies the following question: Why should we regard quantum parallelism any better than a simple probability distribution? In fact, we could obtain equally good results just by selecting N random bits to form a bit string $\boldsymbol{x}$, then to compute $f(\boldsymbol{x})$. If necessary, we could even invent a notation for probability distribution. Let us agree that notation

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \frac{1}{2^N}[\boldsymbol{x}, 0]$$

stands for a probability distribution over $N+1$ bit strings $\boldsymbol{x}$, 0, where each $\boldsymbol{x} \in \{0, 1\}^N$ occurs with a probability $\frac{1}{2^N}$. Then, computing f results into

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \frac{1}{2^N}[\boldsymbol{x}, f(\boldsymbol{x})] \,,$$

and any pair $(\boldsymbol{x}, f(\boldsymbol{x}))$ is seen with a probability of $\frac{1}{2^N}$, as in the case of (59.10).

The answer to that question is that the straightforward use of (59.10) is obviously not the only possible strategy. It should be noted that the amplitudes can be negative as well, and consequently it may be possible to design quantum computing in such a way that the desirable basis states would gain more *visibility* because their amplitudes would sum up, and nondesirable ones could cancel each other. Should that happen, we would call the former *constructive interference* and the latter *destructive interference*.

Example 59.6: State $|0\rangle$ turns into $\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$, if affected by the Hadamard transform. If the affected state were observed, one would see 0 and 1, both with probability $\frac{1}{2}$. On the other hand, if the state is not observed, but another Hadamard transform is applied, we get the following

$$\begin{aligned}
&H\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\\
&\quad=\frac{1}{\sqrt{2}}(H|0\rangle+H|1\rangle)\\
&\quad=\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\right)\\
&\qquad+\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle)\right)\\
&\quad=\left(\frac{1}{2}+\frac{1}{2}\right)|0\rangle+\left(\frac{1}{2}-\frac{1}{2}\right)|1\rangle=|0\rangle\,,
\end{aligned}$$

which demonstrates in detail how the amplitudes $\frac{1}{2}$ and $\frac{1}{2}$ sum up to 1, and $\frac{1}{2}$ and $-\frac{1}{2}$ cancel each other.

A parallel to classical information processing could be as follows: the Hadamard transform may be interpreted as a fair coin toss. Beginning either from $|0\rangle$ or $|1\rangle$, one reaches state $1/\sqrt{2}(|0\rangle \pm |1\rangle)$, where 0 and 1 are both seen with probability $\frac{1}{2}$ (single coin toss). If the coin is tossed twice in the classical settings, then again 0 and 1 are seen both with 50% probability. But in this example, the second *coin toss* returns the state into $|0\rangle$, and hence 0 is seen with 100% probability. This is a feature that is clearly impossible with classical information.

Powerful quantum algorithms, such as Shor's factoring algorithm are indeed all algorithms utilizing quantum interference in a clever manner. Unfortunately, the interference behavior of a quantum algorithm is quite difficult to control in practice, and consequently only a few families of quantum algorithms are known to date.

The known quantum algorithm families enclosing almost all known quantum algorithms are:

1. Quantum algorithms based on Fourier transforms
2. Amplitude amplification methods
3. Quantum random walks
4. Adiabatic quantum algorithms.

59.4.1 Quantum Algorithms Based on Fourier Transforms

This class of quantum algorithms usually provide an apparent exponential speedup over their classical counterparts. The algorithms in this class attempt to construct a superposition

$$\sum_{\boldsymbol{x}} \alpha_{\boldsymbol{x}}|\boldsymbol{x}\rangle$$

whose *amplitudes* $\alpha_{\boldsymbol{x}}$ form a periodic or almost periodic sequence. In many cases, it is then possible to perform the discrete Fourier transform *on amplitudes* (exponentially many) with only a polynomial number of operations on quantum bits (see, e.g., [59.17]). This structure takes care of interfering quantum computational paths by using centuries old knowledge on discrete Fourier transforms.

The approach has been very successful. The algorithm by *Deutsch–Jozsa* [59.13], *Simon*'s algorithm [59.14], and *Shor*'s algorithm [59.4] are all quantum algorithms that control their amplitudes by a structure given by discrete Fourier transforms. The most prominent examples of exponential speed-ups are provided by Fourier transform-based quantum algorithms. For details, see [59.17] or [59.19].

59.4.2 Amplitude Amplification Methods

The amplitude amplification method was presented by *Grover* in 1996 [59.30], and it has been extended thereafter. The basic form of Grover's method involves a function $f:\{0,1\}^N \to \{0,1\}$ assumed to be computable in polynomial time, and is applied to a superposition

$$\sum_{\boldsymbol{x}\in\{0,1\}^N} \alpha_{\boldsymbol{x}}|\boldsymbol{x}\rangle \; .$$

The purpose is to use quantum interference to increase the joint squared absolute values of those amplitudes $\alpha_{\boldsymbol{x}}$, for which $f(\boldsymbol{x}) = 1$, to make such values $\boldsymbol{x}$ more likely to be observed. Grover presented an iterative procedure for that purpose. An iteration step typically involves one evaluation of f.

Among all quantum algorithms, the amplitude amplification methods deserve the first right to be called *quantum-most* methods, as all the other ones have an analog or a counterpart in classical computing. Grover's method does not have any; it is a method purely originating from quantum computing purposes.

The most remarkable consequence of Grover's method is that a general NP-complete problem can be solved (with a high probability) by quantum computers using only $O(\sqrt{N})$ evaluations of function f. When comparing this to $O(N)$ evaluations in the classical case, this is an essential improvement, but not yet not an exponential one: $\sqrt{2^N} = (\sqrt{2})^N$ is again an exponential function, although with a smaller base. For details, see [59.17] or [59.19].

59.4.3 Quantum Random Walks

Quantum random walks is a straightforward analog of classical random walks. It is known that quantum random walks sometimes have exponentially faster hitting times than their classical counterparts [59.31], and the technique can be attempted for a great variety of computational problems. This article is too narrow to address quantum random walks in detail, but for an exposition, see [59.32].

59.4.4 Adiabatic Quantum Algorithms

Adiabatic quantum algorithms should not be called algorithms, but just a technique for designing quantum algorithms. It, or at least its generality can be loosely compared to classical evolutionary algorithms. Adiabatic quantum computing can be adapted to any computational problem.

Adiabatic quantum computing is based on the *adiabatic theorem*, which says that if a *Hamiltonian operator* H_0 is transformed into another Hamiltonian H_1 slowly enough, then the ground state $|\boldsymbol{x}_0\rangle$ of H_0 is transformed into the ground state $|\boldsymbol{x}_1\rangle$ of H_1, as well. The terminology is not explained here, but the reader is advised to consult [59.33]. Instead of detailed definitions and descriptions, we just mention that the adiabatic computation has been shown to be equally as powerful as quantum computing based on quantum gates [59.33]. The technique of adiabatic computing just provides an advantage that the algorithm design can be circumvented in some cases.

59.4.5 Restrictions of Quantum Computing

Theoretical computer science seems to suffer from powerful absolute limitations of computational models. It has been previously mentioned that question $\mathrm{P} \neq \mathrm{NP}$ is waiting for resolution, but there are many analogous unsolved problems. From a quantum computing point of view, the most important such problem is probably: Is polynomial-time quantum computing more powerful than its classical counterpart, at least for some instances? Very likely this problem is at least as difficult as the P versus NP problem, and in the sight of the present understanding, there is no apparent route how to even approach these problems.

However, there are easy ways to obtain *relativized* lower bounds for computational complexity. For relativization in computing, we refer to [59.34], but the basic results are easy to state as follows: a general NP-complete problem described in an earlier section is *relativized* in the sense that no structure of function f is available. I should apologize to complexity theory experts for the simplifications in this explanation (but on second thought, I will not do that), but the intuitive idea in a relativized lower bound is the following: if *nothing* is known of the structure of function f (but the values are chosen arbitrarily), then to decide about the existence of $\boldsymbol{x} \in \{0, 1\}^N$ so that $f(\boldsymbol{x}) = 1$ will inherently take 2^N evaluations of f, since for any process asking less values, there is a possibility to introduce f assigning a *wrong* value to the non-queried string. This certainly gives reasons to believe that $\mathrm{P} \neq \mathrm{NP}$, but does not constitute any proof of that, since no polynomial-time computable function exists without a structure.

Relative lower bounds are known for quantum computing, too. It is, for example, possible to say, that for a general NP-complete problem, quantum computing offers no polynomial time solution. Instead, at least $\sqrt{2^N}$ computational steps are needed for a solution (this is to say that Grover's method is asymptotically optimal). There are various techniques for obtaining relativized lower bounds for quantum computing, and as the most notable ones we can mention the *polynomial technique* [59.35] and the *adversary technique* [59.36].

59.4.6 Physical Realization of Quantum Algorithms

As of 2013, quantum algorithms have been under development almost for three decades, and many things are known about them. In a previous section, we listed four general families of quantum algorithms and mentioned some techniques for proving lower bounds for quantum computing. Even though the development may seem modest in some sense, there are enough interesting quantum algorithms to justify the quantum computer development project. Unfortunately, to build a quantum computer has turned out to be a very challenging task.

In principle, any quantum physical two-state system could serve as a quantum bit. Unfortunately such a system is always very vulnerable to external disturbances, and consequently in many realizations, the lifetime of a qubit is only a tiny fraction of a second. The following physical realizations (among others) have been proposed. Cold trapped ions [59.37], nuclear spin [59.38], and photon polarization [59.39] are all potential implementations of quantum bits, but the most advanced quantum computer (with respect to the number of quantum bits) in modern technology allows us to hold only 12 quantum bits [59.40]. Quantum factoring algorithm for N-bit integers will require approximately $2N$ qubits [59.41]. This implies that the current quantum computers cannot handle enough bits to perform universal computation to truly challenge RSA or other public-key cryptosystems used currently.

Hence we have to conclude that with modern technology, we cannot yet realize very much quantum computing. The most important lesson the 30-year lasting research on quantum computing provides us is, therefore, new insights into the theory of computing and relations between the theory of computation and physical world.

59.5 Biological Applications of Quantum Computing

Even though large-scale quantum computers do not exist yet, we already know various potential applications. A fast integer factoring algorithm would be very influential, even though its influences may not be called entirely positive. Quantum algorithms providing an exponential speed-up over known classical ones are almost all designed by using quantum Fourier transform, and hence they are applicable only for problems having a suitable periodic structure. Such structures do not typically exist in biological problems, and therefore they are not very likely to have an exponential speed-up on biological problems.

On the other hand, a quantum computer would provide a quadratic speed-up on all search problems, and there are various potential applications. For instance, protein folding problems are typical search problems that could benefit from a quadratic speed-up, but as this would apply for any search problem, we are not going to list especially biological search problems.

Instead, we will conclude this article by pointing out a specific problem that quantum computers are good at and which may have consequences in biology, as well. This specific problem also encloses the circle: quantum computers are good at simulating quantum physics, just as *Feynman* explained in his article [59.10]. In general, simulating a quantum mechanical system of N particles with classical computers seems to lead into an exponential slowdown in the simulation efficiency, and Feynman proposed that a quantum computer could be used to avoid the slowdown.

As quantum mechanics governs microsystems, and large biomolecules are built of smaller particles, one could expect that quantum mechanics to have an important explanatory value on biomolecular processes [59.42]. Perhaps it is so, but there is not much existing research on this topic. One reason for this is that structures like DNA are so complex from the physical perspective that even the modeling becomes extremely hard, to say nothing of the explicit solutions. Another reason is that the help provided by computers will not lead very far – as long as we do not have quantum computers.

References

59.1 L.M. Adleman: Molecular computation of solutions to combinatorial problems, Science **266**(11), 1021–1024 (1994)

59.2 R.J. Lipton: DNA solution of hard computational problems, Science **268**(5210), 542–545 (1995)

59.3 M.L. Minsky, S.A. Papert: *Perceptrons* (MIT Press, Cambridge 1969)

59.4 P.W. Shor: Algorithms for quantum computation: Discrete log and factoring, Proc. 35th Annu. IEEE Symp. Found. Comput. Sci. (1994) pp. 20–22

59.5 A.M. Turing: On computable numbers, with an application to the entscheidungsproblem, Proc. Lond. Math. Soc. **2**(42), 230–265 (1936)

59.6 J. von Neumann: Thermodynamik quantummechanischer Gesamheiten, Nachr. Ges. Wiss. Gött. **1**, 273–291 (1927)

59.7 J. von Neumann: *Mathematische Grundlagen der Quantenmechanik* (Springer, Berlin 1932)

59.8 P.A. Benioff: The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, J. Stat. Phys. **22**(5), 563–591 (1980)

59.9 P.A. Benioff: Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: Application to turing machines, Int. J. Theor. Phys. **21**(3/4), 177–202 (1982)

59.10 R.P. Feynman: Simulating physics with computers, Int. J. Theor. Phys. **21**(6/7), 467–488 (1982)

59.11 D. Deutsch: Quantum theory, the Church–Turing principle and the universal quantum computer, Proc. R. Soc. A **400**, 97–117 (1985)

59.12 D. Deutsch: Quantum computational networks, Proc. R. Soc. A **425**, 73–90 (1989)

59.13 D. Deutsch, R. Jozsa: Rapid solutions of problems by quantum computation, Proc. R. Soc. A **439**, 553 (1992)

59.14 D.R. Simon: On the power of quantum computation, Proc. 35th Annu. IEEE Symp. Found. Comput. Sci. (1994) pp. 116–123

59.15 E. Bernstein, U. Vazirani: Quantum complexity theory, SIAM J. Comput. **26**(5), 1411–1473 (1997)

59.16 Clay Mathematics Institute: http://www.claymath.org/millennium/P_vs_NP/

59.17 M. Hirvensalo: *Quantum Computing*, 2nd edn. (Springer, Berlin, Heidelberg 2004)

59.18 M. Hirvensalo: Mathematics for quantum information processing. In: *Handbook of Natural Computing*, ed. by G. Rozenberg, T. Bäck, J. Kok (Springer, Berlin, Heidelberg 2011)

59.19 M.A. Nielsen, I.L. Chuang: *Quantum Computation and Quantum Information* (Cambridge Univ. Press, Cambridge 2000)

59.20 M. Hirvensalo: EPR Paradox and Bell Inequalities, Bulletin EATCS **92**, 115–139 (2007)

59.21 D. Bohm: *Quantum Theory* (Prentice-Hall, Englewood Cliffs 1951) pp. 614–619

59.22 A. Einstein, B. Podolsky, N. Rosen: Can quantum-mechanical description of physical reality be considered complete?, Phys. Rev. **47**, 777–780 (1935)

59.23 J.S. Bell: On the Einstein–Podolsky–Rosen paradox, Physics **1**, 195–200 (1964)

59.24 R. Ursin, F. Tiefenbacher, T. Schmitt-Manderbach, H. Weier, T. Scheidl, M. Lindenthal, B. Blauensteiner, T. Jennewein, J. Perdigues, P. Trojek, B. Ömer, M. Fürst, M. Meyenburg, J.G. Rarity, Z. Sodnik, C. Barbieri, H. Weinfurter, A. Zeilinger: Free-space distribution of entanglement and single photons over 144 km, Nat. Phys. **3**(7), 481–486 (2007)

59.25 A. Barenco, C.H. Bennett, R. Cleve, D.P DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, H. Weinfurter: Elementary gates for quantum computation, Phys. Rev. A **52**(5), 3457–3467 (1995)

59.26 A. Kondacs, J. Watrous: On the power of quantum finite state automata, Proc. 38th Annu. Symp. Found. Comput. Sci. (1997) pp. 66–75

59.27 C. Moore, J.P. Crutchfield: Quantum automata and quantum grammars, Theor. Comput. Sci. **237**(1/2), 275–306 (2000)

59.28 C.H. Bennett: Logical reversibility of computation, IBM J. Res. Dev. **17**, 525–532 (1973)

59.29 M. Hirvensalo: Quantum automata with open time evolution, Int. J. Nat. Comput. Res. **1**, 70–85 (2010)

59.30 L.K. Grover: A fast quantum-mechanical algorithm for database search, Proc. 28th Annu. ACM Symp. Theory Comput. (1996) pp. 212–219

59.31 J. Kempe: Discrete quantum walks hit exponentially faster, Probab. Theory Relat. Fields **133**(2), 215–235 (2005)

59.32 J. Kempe: Quantum random walks – an introductory overview, Contemp. Phys. **44**(4), 307–327 (2003)

59.33 D. Aharonov, W. van Dam, J. Kempe, Z. Landau, S. Lloyd, O. Regev: Adiabatic quantum computation is equivalent to standard quantum computation, SIAM J. Comput. **37**, 166–194 (2007)

59.34 C.H. Papadimitriou: *Computational Complexity* (Addison-Wesley, Reading 1994)

59.35 R. Beals, H. Buhrman, R. Cleve, M. Mosca, R. Wolf: Quantum lower bounds by polynomials, Journal ACM **48**(4), 778–797 (2001)

59.36 A. Ambainis: Quantum lower bounds by quantum arguments, J. Comput. System Sci. **64**(4), 750–767 (2002)

59.37 J.I. Cirac, P. Zoller: Quantum computations with cold trapped ions, Phys. Rev. Lett. **74**, 4091–4094 (1995)

59.38 I.L. Chuang, N. Gershenfeld, M. Kubinec: Experimental implementation of fast quantum searching, Phys. Rev. Lett. **80**, 3408–3411 (1998)

59.39 J.L. O'Brien: Optical quantum computing, Science **318**, 1567–1570 (2007)

59.40 C. Negrevergne, T.S. Mahesh, C.A. Ryan, M. Ditty, F.-Y. Cyr-Racine, W. Power, N. Boulant, T. Havel, D.G. Cory, R. Laflamme: Benchmarking quantum control methods on a 12-qubit system, Phys. Rev. Lett. **96**, 170501 (2006)

59.41 S. Beauregard: Circuit for Shor's algorithm using $2n+3$ qubits, Quantum Inform. Comput. **3**(2), 175–185 (2003)

59.42 V.V. Nelayev, K.N. Dovzhik, V.V. Lyskouski: Quantum effects in biomolecular structures, Rev. Adv. Mater. Sci. **20**, 42–47 (2009)