Oracle Quantum Computing*

André Berthiaume[†] and Gilles Brassard[‡]

Département IRO
Université de Montréal.
C.P. 6128, succursale centre-ville
Montréal (Québec) Canada H3C 3J7

1 April 1994

Abstract

Building on work of Deutsch and Jozsa, we construct oracles relative to which (1) there is a decision problem that can be solved with certainty in worst-case polynomial time on the quantum computer, yet it cannot be solved classically in *probabilistic* expected polynomial time if errors are not tolerated nor even in *nondeterministic* polynomial time, and (2) there is a decision problem that can be solved in exponential time on the quantum computer, which requires *double* exponential time on *all but finitely many* instances on any classical deterministic computer.

* The results given in this paper have been presented at the *7th Annual IEEE Symposium on Structure in Complexity Theory*, Boston, June 1992, and at the *Workshop on Physics and Computation, PhysComp '92*, Dallas, October 1992; they have appeared in preliminary form in those Proceedings [7, 8].

† Supported in part by an NSERC postgraduate fellowship.

‡ Supported in part by Canada's NSERC E. W. R. Steacie Memorial Fellowship and Québec's FCAR.

1 Introduction

In his 1982 keynote address to the first workshop on the physics of computation, Richard Feynman asked the question of whether or not physics can be simulated with computers [15]. Although he considered the idea of using a quantum computer for this purpose, he called it “a side remark”, whereas he considered the question of simulating quantum processes with a classical computer to be “the interesting” question. He expressed the belief that it is not possible to simulate physics *in real time* with a classical computer, whereas it may be possible with a quantum computer. If true, this would imply that there are tasks the quantum computer can handle significantly faster than any classical computer ever could. Feynman discussed quantum computers in more detail in a plenary talk he presented at the 1984 CLEO/IQEC Meeting [16]. Soon thereafter, David Deutsch formalized the concept of the quantum computer and the question of whether it might be more powerful than classical computers from a computational complexity point of view [11, 12].

What makes the quantum computer different from a classical computer is the possibility it offers for massive parallelism within a single piece of hardware. Let f be a computable function. If you have to compute it on two different inputs x and y, you may have to compute $f(x)$ and then start afresh to compute $f(y)$, thus requiring about twice as long as if you had needed only one of these values. The quantum computer allows you to prepare an input that encodes both x and y in *quantum superposition*. If you run your program that computes f on that input, and if your computer can keep its quantum coherence, you obtain the quantum superposition of $f(x)$ and $f(y)$ in the time needed to compute the function once only. Even better, you can prepare in linear time the quantum superposition of an *exponential* number of inputs, after which a single call to the program computes the quantum superposition of all the corresponding outputs. If this sounds too good to be true, it is in a sense. The catch is that even though you have produced exponentially many outputs for the price of one, quantum mechanics severely restricts the way in which you can look at the result. Moreover, by virtue of Heisenberg's uncertainty principle, any measurement of the output state that yields information in a classical form induces an irrevocable destruction of some remaining information, making further measurements less informative or even completely useless.

Nevertheless, some functions of these outputs—such as the exclusive-or of any two of them—are *computable by quantum parallelism*. This means that there exists a measurement so that the correct result is obtained with nonzero probability when the output state is subjected to it; otherwise, the measurement yields “bad luck”. Thus, such a measurement never gives a wrong answer, but sometimes it fails to answer at all, much like a classical *Las Vegas* algorithm. Regardless of success, the state is irreversibly randomized by the measurement process, so that it cannot undergo additional measurements. Therefore, the entire effort was in vain when the answer is “bad luck”. Nevertheless, this outcome is given probabilistically according to the laws of quantum mechanics, and it may well not reoccur if you start the whole process all over again. Unfortunately, the probability of success is never better than 2^{-n} in the worst case when there are 2^n outputs in equally weighted superposition [20], unless you wish to compute a constant function of the outputs. The simplest example of such behaviour is useless but instructive: any single one of the 2^n outputs can be obtained with probability 2^{-n} of success. (This is useless because the same can be achieved classically: secretly choose one input r at random and compute $f(r)$. Then, if you need the value of $f(x)$ for some x chosen independently from r, just check whether $x = r$ and say "success"—with probability 2^{-n}—if that is so.)

However, Deutsch gave a situation in which this type of quantum parallelism can be interesting [11]. Consider the case of a function $f : X \times \Sigma \to \Sigma$, where X is arbitrary and Σ denote the binary alphabet $\{0, 1\}$. Assume that computing $f(x, b)$ takes almost one day on any given input $\langle x, b\rangle$. Let us imagine that $f(x, 0) \oplus f(x, 1)$ represents a crucial decision that must be made within 24 hours after the value of x becomes available, such as buying or selling on the stock market, or making weather predictions for tomorrow ("$\oplus$" denotes the exclusive-or operation, or addition modulo 2). With a classical computer, nothing can ever be achieved: it is too late to make any use of the result by the time both $f(x, 0)$ and $f(x, 1)$ have been computed. In sharp contrast, the quantum computer allows us to obtain this exclusive-or before deadline with probability 50%. Even though either approach requires two days on average to compute the answer, the quantum computer gives a useful answer every other day, while the classical computer never does.

If the logical OR function of exponentially many bits in quantum superposition could be computed with non negligible probability of knowing that the answer has been obtained, the quantum computer could be used to solve the satisfiability problem efficiently, which in turn could be used to solve all **NP**–complete problems efficiently, such as the travelling salesperson problem [10, 21]. This would be a major breakthrough. Unfortunately, even the OR of *two* bits in quantum superposition is *not* computable by quantum parallelism: no measurement can give the OR with nonzero probability of learning that the correct answer has been obtained [20]. Nevertheless, Deutsch and Jozsa have discovered that the OR of exponentially many bits in quantum superposition can be computed efficiently *under the promise* [19] that either all the outputs are 0 or that *exactly* half of them are 1 [13]. Moreover, this can be done *with certainty* in the sense that the result of the measurement is always correct and never "bad luck", provided that the promise is honoured.

In other words, consider any function $f : \Sigma^* \to \Sigma$. Assume that f has the property that, for each integer n, either $f(x) = 0$ for all x in Σ^n or else $f(x) = 1$ for exactly 2^{n-1} strings x in Σ^n. (See Section 4 for a definition of Σ^n and Σ^*.) Consider a classical algorithm A capable of computing $f(x)$ in worst-case time in $O(t(n))$ for all x of length n, where $t(n) \geq n$ for all n. The result of Deutsch and Jozsa implies that the quantum computer can decide for all n in worst-case time in $O(t(n))$ whether or not $f(x)$ is identically zero on inputs of length n.

In this paper we show how powerful this result of Deutsch and Jozsa is in relativized settings, that is if we assume the availability of arbitrarily complex oracles capable of instantly answering a precise set of questions. This is a familiar approach in theoretical computer science and computational complexity theory, pioneered by Baker, Gill and Solovay [2]. The direct oracle interpretation of their result is the existence of an oracle under which there is a decision problem that can be solved in linear time on the quantum computer, yet any deterministic classical computer would require exponential time on infinitely many instances to solve the same problem (Theorem 1).

One might be tempted to think at first that the quantum computer gets its advantage over classical deterministic computers not from the use of quantum superposition, but merely from the much more mundane availability of randomization, which is inherent to quantum computation. After all, it is generally believed that classical randomized computers enjoy a computational advantage over deterministic computers, even if the randomized computer is not allowed to ever yield an erroneous result. One purpose of this paper is to defuse this uninteresting interpretation by showing that relativized quantum computers can efficiently handle tasks that are beyond not only error-free classical probabilistic computers (Theorem 7) but even nondeterministic computers (Theorem 4). Later results by Bernstein and Vazirani [6] and Simon [24] extend this to the case in which classical probabilistic computers are not required to be error-free. We also exhibit a relativized decision problem that the quantum computer can solve exponentially faster than any classical deterministic computer *on all but finitely many* instances (Theorem 6).

2 Quantum Interference

Classical probabilistic computers may be more powerful than deterministic machines because many different states can be reached from the same input state after a fixed number of steps, according to the laws of probabilities. The probability of ending up at a specific final state is given by the sum of the probabilities of reaching that state by all possible probabilistic paths leading to it. Quantum machines may be more powerful still because they allow *quantum interference* between the possible paths leading to any given state. This allows "probabilities" along different paths not only to add up, but also to subtract and even cancel out. For instance, it may be that some final state x follows with nonzero probability from intermediary state a, and that it also follows with the same probability from state b, yet if both a and b are equally likely to occur during the computation, state x may never occur at all as final state because its probability to follow from a may be cancelled out by its probability to follow from b. In the words of Feynman [15], it is as if "somehow or other it appears as if the probabilities would have to go negative". This is something that "we do not know [...] how to simulate" classically, which would explain why the quantum computer is (perhaps) computationally more powerful than classical computers. (Read [17] for Feynman's in-depth and entertaining look at negative probabilities.) Quantum interference can also work to reinforce probabilities rather than annihilating them: the probability of reaching final state x through intermediary states a and b—note that we did not say a or b—can be *four* times greater than the probability of reaching it through either intermediary state alone.

To describe quantum computing rigorously, we represent the *state* of the computer by a vector in a Hilbert space (that is, a vector space with complex numbers in the underlying field). The *evolution* of the computer is determined by a matrix U. Each entry U_{ij} represents the *amplitude* of finding the system in state j given that it was in state i at the previous step. The corresponding probability is obtained by squaring the magnitude of the amplitude, but amplitudes are translated into probabilities only when the state of the computer is observed. Moreover, observing that the computer is in some state causes a *collapse* of the quantum superposition, so that the amplitude of the observed state becomes 1 after observation, at the expense of all other states whose amplitude drops down to 0. (It is possible to collapse only part of the state by performing an incomplete measurement, leaving the rest of the state in quantum superposition.)

For example, if the system is in state i with certainty at some time (meaning that the magnitude of the amplitude of state i is 1), then it will be in state j with probability $|U_{ij}|^2$ if it is observed after the next step. More generally, if v is the current *state vector* at some moment in the computation, which means that the computer is in state i with amplitude v_i for each i, then it will be in state j with amplitude $(U^n v)_j$ after n steps. It will thus be found in state j with probability $|(U^n v)_j|^2$ if it is observed at that time but not before. Quantum physics restricts the evolution of any quantum system—including quantum computers—to be described by *unitary* matrices, which means that the inverse of U must be equal to its own conjugate transpose. This ensures that the sum of probabilities of all possible states is always equal to 1 at any given time; it also forces quantum computing processes to be reversible. (A technique to implement the effect of arbitrary unitary matrices by the application of the proper Hamiltonian is given by Feynman [16].)

Now we can see how quantum interference comes about. Let i and j be two different possible states of the quantum computer at a given step of the computation. Suppose state k follows directly from both i and j with nonzero amplitude, meaning that both U_{ik} and U_{jk} are nonzero, and that there is no other way to reach state k but through i or j. Let α_1 and α_2 be the amplitudes of reaching k from the initial state through i and j, respectively. The probability of observing state k is $p = |\alpha_1 + \alpha_2|^2$. Let us consider the two extreme cases. If $\alpha_1 = \alpha_2$, we have $p = 4|\alpha_1|^2$, which is twice the sum of the classical probabilities of reaching state k through i and of reaching it through j. On the other hand, if $\alpha_1 = -\alpha_2$, then $p = 0$ and we *never* observe state k even though each of i and j could lead to k with nonzero probability! Such interference is impossible in the classical world. It gives the quantum computer intriguing capabilities. As an example, consider the following problem:

> Is it possible to find an operation W such that applying it *twice* to bit b produces the Boolean negation of b? (Such a W is called the *square root* of **not**.)

Such an operation cannot be implemented classically. Yet, consider

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

and let 0 and 1 be represented by vectors

$$v_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and } v_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix},$$

respectively. It is easy to see that W is unitary and $W^2 v_b = \pm v_{1-b}$ for $b = 0$ and 1. This achieves the desired result because the sign of the amplitude does not change the corresponding probability if the state is measured (unless interference occurs between the double application of W and the measurement). Observe the interference patterns at work: while the first application of W on v_b generates both v_0 and v_1 with equal amplitude (up to sign), the second application precisely annihilates v_b, leaving only v_{1-b} with amplitude ± 1 (and thus with probability 1 if the state is observed). This power of interference is what (seems to) give an edge to the quantum computer over classical machines. More formal and complete descriptions of the quantum computer are given in [11, 6]; a more detailed intuitive description can be found in [24].

3 Quantum Complexity Classes

We assume the reader is familiar with classical computational complexity classes such as **P**, **NP** and **PSPACE**. The most natural quantum complexity class is denoted **EQP**; it is the class of decision problems that can be solved with certainty in worst-case polynomial time by the quantum computer. (The letters stand for **E**xact **Q**uantum **P**olynomial-time.) Even though the quantum computer is allowed randomization by the very nature of quantum physics, note that it is allowed (in the context of **EQP**) neither a non-zero error probability, nor a non-zero probability of exceeding its polynomial time bound. The same requirement for a classical probabilistic Turing machine would define nothing but the class **P** of decision problems that can be solved classically in polynomial time by a *deterministic* computer since coin-tossing is useless in such an austere setting. It is not immediately obvious that $\mathbf{P} \subseteq \mathbf{EQP}$ because of the restriction that the evolution of the quantum computer must be given by a unitary matrix, which forces each operation of the quantum computer to be reversible. Fortunately, Lecerf [22] and Bennett [3] have proven independently that classical computers can be programmed reversibly without significant degradation in running time, and it is elementary to implement any reversible computational process with a unitary evolution. The most basic natural question is whether or not the inclusion $\mathbf{P} \subseteq \mathbf{EQP}$ is proper. Unfortunately, this question is likely to be very hard to settle because $\mathbf{EQP} \subseteq \mathbf{PSPACE}$ [6] and it is still unknown whether or not $\mathbf{P}$ is a proper subset of $\mathbf{PSPACE}$, despite considerable effort in the classical community to solve this outstanding open problem. This is why we resort to relativized arguments in this paper. It is equally natural to define quantum versions of many other classical complexity classes, including probabilistic ones. We defer the definition of some of those classes to Section 8.

4 Notation and Definition

We use $\mathbf{N}$ to denote the set of strictly positive integers and Σ to denote the binary alphabet $\{0,1\}$. For each $n \geq 0$, Σ^n denotes the set of all binary strings of length n and Σ^* denotes the set of all finite-length binary strings. Consider a set $X \subseteq \Sigma^*$. We say that $B(X)$ holds if for each integer n either $X \cap \Sigma^n = \emptyset$ or there are exactly 2^{n-1} strings in $X \cap \Sigma^n$. The (sparse) set S_X is defined as

$$S_X = \{1^n \mid X \cap \Sigma^n = \emptyset\}.$$

In other words, S_X contains only strings of ones, and $1^n \in S_X$ if and only if there are no strings of length n in X.

Unless necessary, we shall not distinguish between Turing machines and computers. An *oracle Turing machine* is a computer (deterministic, probabilistic, nondeterministic or quantum) that is allowed to ask questions of an oracle. We assume that oracles are deterministic, in that a given oracle will always answer a given query in the same way. It is convenient to think of oracles as subsets of Σ^*. When asked question "x", oracle X answers "yes" or "no" depending on whether or not $x \in X$. Even though the oracle answers instantaneously, we assume it takes at least n steps for an oracle Turing machine to prepare a question of size n, because it must write it down on its tape symbol by symbol, and one step to ask it. In the case of quantum computers, we assume that the oracle is capable of entering quantum superposition and to become entangled in a coherent state with the machine. The original result of Deutsch and Jozsa [13] implies that S_X can be recognized in linear time on the quantum computer if oracle X is available, provided $B(X)$ holds.

If M is an oracle Turing machine and $X \subseteq \Sigma^*$, M^X denotes machine M equipped with oracle X. Similarly, if $\mathbf{C}$ is a complexity class (classical or quantum) and $X \subseteq \Sigma^*$,

$\mathbf{C}^X$ denotes the class corresponding to $\mathbf{C}$ if the machines are allowed to access X as oracle. For example, $\mathbf{EQP}^X$ is the class of decision problems that can be solved with certainty in worst-case polynomial time by a quantum computer that is allowed to ask questions to oracle X. In particular, the result of Deutsch and Jozsa implies that $S_X \in \mathbf{EQP}^X$ for all $X \subseteq \Sigma^*$ for which $B(X)$ holds.

Let σ be a standard bijection between Σ^* and $\mathbf{N}$, such as $\sigma(x) = bin(1x)$, where "$1x$" denotes string x with a "1" appended in front of it and $bin(y)$ is the integer whose binary representation is y. For instance, $\sigma(101) = bin(1101) = 13$. With this encoding, $\sigma(x) < 2^{\ell+1}$ for all $x \in \Sigma^*$, where ℓ is the length of x. Let $\tau : \mathbf{N} \to \Sigma^*$ be the functional inverse of σ. When we need to encode the information present in S_X in an exponentially more compact form, we use

$$V_X = \{x \in \Sigma^* \mid X \cap \Sigma^{\sigma(x)} = \emptyset\}.$$

Clearly, V_X can be recognized in exponential time when S_X can be recognized in linear time since $x \in V_X$ if and only if $1^{\sigma(x)} \in S_X$.

Finally, let $\alpha : \mathbf{N} \to \mathbf{N}$ be an arbitrary function such that for each integer i there are infinitely many integers n for which $\alpha(n) = i$. Let $\wp : \mathbf{N} \to \mathbf{N}$ be defined recursively as

$$\wp(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2^{\wp(n-1)} & \text{if } n > 1 \end{cases}$$

5 Quantum Can Beat Deterministic

Recall that the result of Deutsch and Jozsa [13] implies that $S_X \in \mathbf{EQP}^X$ for all $X \subseteq \Sigma^*$ for which $B(X)$ holds. To obtain a relativized separation between $\mathbf{P}$ and $\mathbf{EQP}$, it suffices to construct an oracle X such that $B(X)$ holds, yet $S_X \notin \mathbf{P}^X$. This is accomplished by diagonalization. At each stage of the construction, a deterministic oracle Turing machine is knocked down by forcing it either to give an incorrect answer concerning S_X or to spend too much time. Each machine is knocked down an infinite number of times by the construction. We give only a rough sketch of the formal proof of this theorem because it is a special case of Theorem 4, which we prove in detail. Moreover, a detailed (and somewhat different) proof of Theorem 1 is given in [7].

Theorem 1 *There exists an oracle relative to which there is a set that can be recognized in worst-case linear time by the quantum computer, yet any deterministic Turing machine that recognizes it must take exponential time on infinitely many inputs.*

Proof. Let $M_1, M_2, \ldots$ be a standard enumeration of all deterministic oracle Turing machines. We shall construct oracle X in stages. Initially, set $X_1 = \emptyset$. At stage $n \geq 1$, simulate machine $M_{\alpha(n)}$ on input $1^{\wp(n)}$ for up to $2^{\wp(n)-1}$ steps, using X_n as oracle.

- $\diamond$ If the machine has not stopped within $2^{\wp(n)-1}$ steps, or if it rejects its input, set $X_{n+1} = X_n$ and continue with the next stage.

- $\diamond$ Otherwise, let Q_n be the set of oracle questions asked by the machine during the simulation. Note that Q_n contains at most $2^{\wp(n)-1}$ strings because at least one step of the machine is needed to ask each oracle question. But there are $2^{\wp(n)}$ strings of length $\wp(n)$, and therefore we can form a set $R_n \subset \Sigma^{\wp(n)}$ such that $R_n \cap Q_n = \emptyset$ and the number of elements in R_n is exactly $2^{\wp(n)-1}$. Set $X_{n+1} = X_n \cup R_n$ and continue with the next stage.

Let $X = \bigcup_{n \geq 1} X_n$. It is easy to show that S_X can be recognized in linear time with the quantum computer if oracle X is available, but that any deterministic oracle Turing machine M_i that recognizes S_X with the help of oracle X requires more than $2^{\ell-1}$ steps on input 1^ℓ when $\ell = \wp(n)$ for each n for which $\alpha(n) = i$. We omit the details because they are very similar to those of the more interesting proof of Theorem 4 below. ∎

Corollary 2 *There exists an oracle X such that $\mathbf{P}^X \subset \mathbf{EQP}^X$ (strict inclusion).*

Proof. This is immediate from Theorem 1 because a polynomial cannot exceed an exponential on more than finitely many points. ∎

Corollary 3 *Consider any strictly increasing time-constructible function t of the natural numbers. There exists an oracle relative to which there is a set that can be recognized in quantum time $O(t(n))$, yet any deterministic Turing machine that recognizes it requires $\Omega(2^{t(n)})$ time on infinitely many inputs. In particular, there is an oracle that separates deterministic from quantum exponential time.*

Proof. The proof of Theorem 1 applies *mutatis mutandis* provided we replace S_X by

$$\hat{S}_X = \{1^n \mid X \cap \Sigma^{t(n)} = \emptyset\}$$

and $\wp(n)$ by

$$\hat{\wp}(n) = \begin{cases} 1 & \text{if } n = 1 \\ 2^{t(\hat{\wp}(n-1))} & \text{if } n > 1 \end{cases}$$

Details are left to the reader. ∎

6 Quantum Can Beat Nondeterministic

A relativized separation between **EQP** and **NP** can also be obtained by diagonalization, but it is slightly more difficult because we must knock down machines that can have exponentially many computation paths on each input. Recall that a nondeterministic Turing machine *accepts* input x within t steps if and only if there exists at least one accepting path of length at most t in the computation tree of the machine on input x. Thus a nondeterministic machine accepts even if all but one of the paths are rejecting or non-halting, and it accepts within t steps even if all but one of the accepting paths need more than t steps. The time spent by a nondeterministic machine on rejected inputs is undefined.

Theorem 4 *There exists an oracle relative to which there is a set that can be recognized in worst-case linear time by the quantum computer, yet any nondeterministic Turing machine that recognizes it must take exponential time on infinitely many accepted inputs.*

Proof. Let $N_1, N_2, \ldots$ be a standard enumeration of all *non*deterministic oracle Turing machines. We shall construct oracle Y in stages. At the end of stage n, it will be defined on all strings of length less than $\wp(n+1)$. Initially, set $Y_1 = \emptyset$. At stage $n \geq 1$, simulate every path of nondeterministic machine $N_{\alpha(n)}$ on input $1^{\wp(n)}$, using Y_n as oracle, for up to $2^{\wp(n)-1}$ steps for each path in the computation tree.

- ⋄ If the machine accepts its input within $2^{\wp(n)-1}$ steps, select the shortest accepting path (ties are broken arbitrarily) and let Q_n be the set of oracle questions asked along that path. Note that Q_n contains at most $2^{\wp(n)-1}$ strings because at least one step of the machine is needed to ask each oracle question. But there are $2^{\wp(n)}$ strings of length $\wp(n)$, and therefore we can form a set $R_n \subset \Sigma^{\wp(n)}$ such that $R_n \cap Q_n = \emptyset$ and the number of elements in R_n is exactly $2^{\wp(n)-1}$. Set $Y_{n+1} = Y_n \cup R_n$ and continue with the next stage.
- ⋄ Otherwise, set $Y_{n+1} = Y_n$ and continue with the next stage.

Let $Y = \bigcup_{n \geq 1} Y_n$. Note that $B(Y)$ holds because we took care at each step of either putting nothing in Y, or of putting in exactly half the strings of a given length (and we never put in strings of the same length more than once). Therefore, the quantum computer can recognize S_Y in linear time given Y as oracle. Consider now an arbitrary nondeterministic Turing machine N_i that purports to recognize S_Y using oracle Y. Consider any integer n such that $\alpha(n) = i$. We claim that N_i^Y must accept input $1^{\wp(n)}$, and that it must take more than $2^{\wp(n)-1}$ steps to do so.

To prove this claim, first assume for a contradiction that N_i^Y rejects input $1^{\wp(n)}$. By assumption that N_i^Y recognizes S_Y, it follows that $1^{\wp(n)} \notin S_Y$ and thus $Y \cap \Sigma^{\wp(n)} \neq \emptyset$ by definition of S_Y. By construction of the oracle, this means that the first case held during stage n for otherwise no strings of length $\wp(n)$ would have been placed into Y. Therefore N_i accepts $1^{\wp(n)}$ within $2^{\wp(n)-1}$ steps when run with oracle Y_n. Consider the accepting path used in that stage. The set R_n is designed so that it contains none of the oracle questions asked along this path. Moreover, none of those questions can concern a string of size larger than $2^{\wp(n)}-1$ because there would not have been enough time even to have formulated such a large question. Since Y_n and Y differ on strings of length smaller than $\wp(n+1) = 2^{\wp(n)}$ only in that the strings in R_n belong to Y but not to Y_n, the same accepting path of N_i on $1^{\wp(n)}$ exists when oracle Y is used instead of Y_n. Although N_i^Y and $N_i^{Y_n}$ could behave differently on all other paths, one accepting path is enough for a nondeterministic machine to accept, and therefore N_i^Y accepts $1^{\wp(n)}$. This is is a contradiction.

Now that we know that N_i^Y accepts $1^{\wp(n)}$, assume for a contradiction that it does so in at most $2^{\wp(n)-1}$ steps. By assumption that N_i^Y recognizes S_Y, it follows that $1^{\wp(n)} \in S_Y$ and thus $Y \cap \Sigma^{\wp(n)} = \emptyset$. This implies that Y and Y_n are identical on all strings of length less than $\wp(n+1) = 2^{\wp(n)}$, and therefore N_i behaves identically for the first $2^{\wp(n)} - 1$ steps whether it uses Y or Y_n as oracle. In particular, the short accepting path of N_i^Y on input $1^{\wp(n)}$ also exists when N_i uses Y_n as oracle. We conclude that the first case holds at stage n in the construction of Y. Therefore, $Y \cap \Sigma^{\wp(n)} = R_n \neq \emptyset$. This is also a contradiction.

The only remaining possibility is that N_i^Y accepts $1^{\wp(n)}$ but that it requires more than $2^{\wp(n)-1}$ steps to do so. Since this is true for all n such that $\alpha(n) = i$ and since there are infinitely many such n by definition of α, we conclude that any nondeterministic oracle Turing machine that recognizes S_Y given Y as oracle requires exponential time on infinitely many accepted inputs. ∎

Recall that a decision problem belongs to **co–NP** if and only if its complement is in **NP**. Although the set S_Y defined in the proof of Theorem 4 is not in $\mathbf{NP}^Y$, it is in $\mathbf{co\text{–}NP}^Y$ since it is easy to decide the complement of S_Y with a nondeterministic oracle Turing machine that has access to Y as oracle. We now prove that the quantum computer can get beyond **NP** and **co–NP** simultaneously.

Corollary 5 *There exists an oracle Y such that*

$$\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y \cup \mathbf{co\text{–}NP}^Y .$$

Proof. Consider the oracle Y constructed in the proof of Theorem 4. We already know that $S_Y \in \mathbf{EQP}^Y$. Since any nondeterministic Turing machine that recognizes S_Y must take exponential time on infinitely many inputs, it follows that $S_Y \notin \mathbf{NP}^Y$. This proves that $\mathbf{EQP}^Y \not\subseteq \mathbf{NP}^Y$. Because **EQP** is closed under Boolean operations, it is clear that the complement of S_Y testifies to the fact that $\mathbf{EQP}^Y \not\subseteq \mathbf{co}\text{–}\mathbf{NP}^Y$. To prove the corollary, however, we need a single set that is in $\mathbf{EQP}^Y$ but in neither $\mathbf{NP}^Y$ nor $\mathbf{co}\text{–}\mathbf{NP}^Y$. We leave it for the reader to verify that

$$\tilde{S}_Y = S_Y \cup \{0^n \mid 1^n \notin S_Y\}$$

is such a set. ∎

7 Faster Almost Everywhere

Although a classical computer (even nondeterministic) must take much more time than the quantum computer to recognize the set S_Y defined in the proof of Theorem 4 when Y is available as an oracle, this is only true of rare instances. Indeed, any $x \in \Sigma^*$ that contains at least one 0 is automatically outside of S_Y, and 1^m belongs to S_Y for all m that are not in the (very sparse) image of $\wp$. Thus it is only on inputs of the form $x = 1^{\wp(n)}$ that deciding whether or not $x \in S_Y$ represents a challenge. To get a stronger result, we now consider problems that require exponential time on the quantum computer.

Theorem 6 *There exists an oracle relative to which there is a set that can be recognized in worst-case exponential time by the quantum computer, yet any deterministic Turing machine that recognizes it must take* double *exponential time on* all but finitely many *inputs.*

Proof. We shall construct an oracle Z such that $B(Z)$ holds and any deterministic Turing machine that recognizes V_Z using Z as oracle requires double exponential time on all but finitely many inputs. This construction is inspired by Manuel Blum's beautiful proof of his compression theorem [9]. We shall construct oracle Z in stages. At the end of stage n, the oracle will be defined on all strings of length up to n. The key ingredient is that we keep track of a list of *cancelled* machines. Intuitively, whenever a machine is cancelled, we have made sure that it does not recognize V_Z correctly given Z as oracle. Machines that have not yet been cancelled are called *live*. As in the proof of Theorem 1, let $M_1, M_2, \ldots$ be a standard enumeration of all deterministic oracle Turing machines.

Initially, none of the machines are cancelled and $Z_3 = \emptyset$. (For a technical reason we start at stage 3.) At stage $n \geq 3$, run each of $M_1, M_2, \ldots, M_n$ on all the inputs $\tau(m)$ for $1 \leq m \leq n$, using oracle Z_n, for a maximum of $2^{n-1}/n^2$ steps on each run. (Recall that τ is our bijection from the positive integers onto Σ^*, defined in Section 4.) Let Q_n stand for the set of all oracle questions asked during those runs. Note that Q_n contains at most $n \times n \times (2^{n-1}/n^2) = 2^{n-1}$ questions.

- ⋄ If none of the live machines using oracle Z_n stop within $2^{n-1}/n^2$ steps on input $\tau(n)$, set $Z_{n+1} = Z_n$ and continue with the next stage.

- ◇ Otherwise, let i be the smallest integer such that M_i is live and stops within $2^{n-1}/n^2$ steps on input $\tau(n)$ using oracle Z_n.
  - If $M_i^{Z_n}$ rejects $\tau(n)$, set $Z_{n+1} = Z_n$, cancel machine i, and continue with the next stage.
  - Otherwise, let R_n be an arbitrary subset of Σ^n containing exactly 2^{n-1} elements such that $Q_n \cap R_n = \emptyset$ (which is possible since Q_n contains at most 2^{n-1} elements). Set $Z_{n+1} = Z_n \cup R_n$, cancel machine i, and continue with the next stage.

Let $Z = \bigcup_{n \geq 3} Z_n$. Note that $B(Z)$ holds because we took care at each step n of either putting nothing in Z, or of putting in exactly half the strings of length n. Therefore, the quantum computer can recognize S_Z in linear time, and thus V_Z in exponential time, given Z as oracle.

Consider now an arbitrary deterministic Turing machine M_i that purports to recognize language V_Z when given Z as oracle. The first claim is that it is not possible for M_i ever to be cancelled. To show this, assume for a contradiction that it is cancelled at some stage n. In this case, the construction of Z makes sure that M_i using oracle Z_n rejects input $\tau(n)$ if and only if there are no strings of length n in Z, which means that $\tau(n) \in V_Z$. Therefore, M_i does not recognize V_Z when it uses oracle Z_n. To reach the desired contradiction, note that all the questions asked by M_i on input $\tau(n)$ under oracle Z_n find their way into Q_m for all $m \geq n$ and therefore the final oracle Z will not differ from Z_n on any of those questions, which implies that M_i does not recognize V_Z even when it uses oracle Z, contrary to our assumption. (We started at stage 3 to make sure that $2^{m-1}/m^2 \geq 2^{n-1}/n^2$ for all $m \geq n \geq 3$.)

Let n_0 be large enough that all machines of index smaller than i that will eventually be cancelled have been cancelled by stage n_0. Consider any $n > \max(i, n_0)$. We know that machine M_i cannot be cancelled at stage n. Since no machine of smaller index is cancelled at that stage either, it must be that M_i using oracle Z_n spends more than $2^{n-1}/n^2$ steps on input $\tau(n)$. By an argument similar to the one above, this behaviour applies also when the final oracle Z is used. But the length of $\tau(n)$ is logarithmic in n and therefore $2^{n-1}/n^2$ is doubly exponential in the length of $\tau(n)$. Because τ is onto Σ^*, we conclude that if M_i^Z recognizes V_Z it must take at least double exponential time on all sufficiently large inputs. ∎

Note that simulating this quantum computation with classical parallelism would require the availability of a double exponential number of processors since exponentially many processors running for exponential time can only perform an exponential amount of work.

8 Quantum Can Beat Probabilistic

In the classical world, it is generally believed that probabilistic computers can outperform deterministic computers. There are four main classical types of probabilistic algorithms. Each of them gives rise to a class of decision problems when the algorithm is restricted to expected polynomial time.

- A *zero error probability* algorithm is allowed to toss coins during computation, but the answer it returns must always be correct. These algorithms are often called *Las Vegas* and they give rise to the class **ZPP** [18].

- A probabilistic algorithm for a decision problem is *biased* if it always returns the correct answer on instances for which the correct answer is "no" and if it returns the correct answer on all other instances with probability at least ρ for some fixed positive ρ. You can be certain that the answer that has been obtained is correct whenever it is "yes" but that you cannot be sure when you obtain the answer "no". Nevertheless, you can increase your confidence in a "no" verdict by repeating execution of the algorithm: the probability of reaching an incorrect conclusion is at most $(1-\rho)^k$ if you repeat execution of the algorithm k times. This type of probabilistic algorithm gives rise to the class **RP** [1] (which is denoted **R** in [1]). We say that a decision problem belongs to **co–RP** if its complement belongs to **RP**.

- A *bounded error probability* algorithm returns the correct answer on all instances with probability at least ρ for some fixed $\rho > \frac{1}{2}$. The error probability can be brought down efficiently by repeating execution of the algorithm and returning the majority answer. It is now recognized that bounded error probability algorithms are just as good as exact algorithms for all practical purposes, and even preferable if they run significantly faster. These algorithms are often called *Monte Carlo* and they give rise to the class **BPP** [18].

- Finally, an *unconstrained* probabilistic algorithm is required to return the correct answer with probability strictly greater than 50% on all inputs, but this probability is allowed to be arbitrarily close to one-half. Contrary to the other three types of probabilistic algorithms, unconstrained algorithms have no practical value because the probability of being correct can be so close to one-half that one would have to repeat execution of the algorithm an exponential number of times to bring the error probability below a reasonable level. This type of probabilistic algorithm gives rise to the class **PP** [18].

Even though we allow those algorithms to take *expected* polynomial time, **BPP** would be the same if we had required that the algorithms must never exceed their polynomial time bound because the error probability can only change very little if the computation is terminated and an arbitrary answer is given after the expected time has been significantly exceeded. The same remark holds for **RP**. It is obvious that

$$\mathbf{P} \subseteq \mathbf{ZPP} \subseteq \mathbf{RP} \subseteq \mathbf{BPP} \subseteq \mathbf{PP}$$

and that $\mathbf{RP} \subseteq \mathbf{NP}$ but none of these inclusions are known to be proper. Those inclusions also hold relative to arbitrary oracles. Finally note that $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{co\text{–}RP}$ because successive calls to an **RP** and a **co–RP** algorithms on the same instance will eventually yield an answer that is guaranteed to be correct.

These definitions of classical probabilistic algorithms and complexity classes extend naturally to the quantum computer. We define **ZQP**, **RQP**, **BQP** and **PQP** in ways analogous to **ZPP**, **RP**, **BPP** and **PP**, respectively. For example, **ZQP** is the class of decision problems that can be solved by the quantum computer with zero error probability in expected polynomial time. Just as **BPP** captures properly the notion of what can be decided efficiently with a classical computer, it can be argued that **BQP** is probably the most important class of decision problems for the quantum computer. At the time of this writing, however, we are not aware of a Monte Carlo algorithm that has been proposed for the quantum computer, whose efficiency depends on the fact that errors are tolerated.

Theorem 7 *There exists an oracle Y relative to which there is a set that can be recognized in worst-case linear time by the quantum computer, yet it cannot be recognized in expected polynomial time by a zero error classical probabilistic algorithm nor a biased probabilistic algorithm, and neither can its complement. In other words,* $\mathbf{EQP}^Y \not\subseteq \mathbf{ZPP}^Y$, $\mathbf{EQP}^Y \not\subseteq \mathbf{RP}^Y$ *and* $\mathbf{EQP}^Y \not\subseteq \mathbf{co\text{–}RP}^Y$.

Proof. This follows immediately from Corollary 5 since $\mathbf{ZPP}^Y \subseteq \mathbf{RP}^Y \subseteq \mathbf{NP}^Y$. ∎

Theorem 7 is not entirely satisfactory because the decision problem $\tilde{S}_Y$ used in the proof of Corollary 5 can be recognized classically in polynomial time by a bounded error probability algorithm given Y as oracle (provided $B(Y)$ holds). In other words, $\tilde{S}_Y \in \mathbf{BPP}^Y$. To see this, consider an instance x of the problem and let n be its length. Take two random strings y and z in Σ^n, chosen according to the uniform distribution, and ask the oracle if they belong to Y. If $x = 0^n$ and at least one of y or z belongs to Y, or if $x = 1^n$ and neither y nor z belongs to Y, then accept x. In all other cases, reject x. It is easy to verify that this algorithm gives the correct answer with probability exactly 75% on all inputs of the form 0^n or 1^n, and that it is always correct on all other inputs. Similarly, the set used in Section 7 to show that the quantum computer can go exponentially faster than a deterministic computer on all but finitely many inputs can be recognized as efficiently with a classical bounded error probability algorithm as with the quantum computer.

Since it is reasonable to consider **BPP** as the class of decision problems that admit efficient classical solutions in practice, it can be argued that none of the results presented in this paper provide convincing evidence that there are problems that can be solved efficiently by the quantum computer but not by classical computers. Fortunately, very exciting recent work by Ethan Bernstein and Umesh Vazirani has shown the existence of an oracle W relative to which there is a decision problem that can be solved in exact quantum polynomial time but not in bounded error probability classical expected polynomial time [6]. In other words, $\mathbf{EQP}^W \not\subseteq \mathbf{BPP}^W$. Subsequently, Daniel Simon proved the existence of a relativized *exponential* gap between the power of the quantum computer and that of classical Monte Carlo algorithms, but for this he had to leave the realm of *exact* quantum polynomial time: there exists an oracle relative to which there is a set that can be recognized with zero error probability in quantum *expected* polynomial time (**ZQP**), yet any classical probabilistic algorithm that can recognize the same set with bounded error probability requires expected exponential time on infinitely many inputs [24].

9 Toward an Unrelativized Theory

A major open question is to separate **P** from **EQP** in the real world, that is without resorting to oracles. As we said in Section 3, this is likely to be very difficult since a proof that $\mathbf{P} \neq \mathbf{EQP}$ would imply that $\mathbf{P} \neq \mathbf{PSPACE}$, thus settling a major open question in classical computational complexity theory. Nevertheless, it is possible to formulate a classical (as opposed to quantum) computational complexity assumption under which unrelativized $\mathbf{P} \subset \mathbf{EQP}$ (strict inclusion) follows immediately. This involves a variant on the class $\mathbf{C}_{==}\mathbf{P}$ (also known as **WPP**, see [14]), which is defined as the class of sets X for which there exists a nondeterministic polynomial-time Turing machine M and two polynomial-time computable functions f and g from Σ^* to the natural numbers such that $\#M(x) = f(x)$ for all $x \notin X$ whereas $\#M(x) = g(x)$ for all $x \in X$, where $\#M(x)$ denotes the number of accepting paths of machine M on input x. Valiant's better-known class **UP**, for instance, is the subclass of $\mathbf{C}_{==}\mathbf{P}$ for which $f(x) = 0$ and $g(x) = 1$ for all x.

Let us define $\mathbf{C}_{==}\mathbf{P}[\mathsf{half}]$ as the subclass of $\mathbf{C}_{==}\mathbf{P}$ in which $f(x) = 0$ for all x and $g(x)$ is exactly half the number of accepting paths of M on input x. (The number of paths can be made easily computable on a suitable universal enumeration of nondeterministic Turing machines, hence the function g is computable in polynomial time.) The result of Deutsch and Jozsa [13] implies immediately that $\mathbf{C}_{==}\mathbf{P}[\mathsf{half}] \subseteq \mathbf{EQP}$. Therefore, the somewhat reasonable assumption that $\mathbf{P} \subset \mathbf{C}_{==}\mathbf{P}[\mathsf{half}]$ suffices to conclude that $\mathbf{P} \subset \mathbf{EQP}$.

10 Open Questions and Conclusion

Although this paper is about oracle quantum computing, the really interesting open questions concern the removal of oracles from this theory. It is not clear whether or not quantum computers will ever be technologically feasible to build, but oracles never will be. Thus, the quantum computer is interesting in practice only if we can find tasks that it can do without oracles better than classical probabilistic computers. So far, quantum computing has been studied only when the quantum computer is not allowed to make mistakes. A promising direction for further research is to determine the power of bounded error probability quantum computing. In other words, what is the power of **BQP**?

Because a formal proof that $\mathbf{P} \subset \mathbf{EQP}$ is currently out of the question, there are two natural directions to investigate. The first is to prove that $\mathbf{P} \subset \mathbf{EQP}$ or, preferably, that $\mathbf{BPP} \subset \mathbf{BQP}$, under a classical computational complexity assumption more natural than $\mathbf{P} \subset \mathbf{C}_{==}\mathbf{P}[\mathsf{half}]$. For instance, can the desired conclusion be established under the sole assumption that $\mathbf{P} \neq \mathbf{NP}$ or, failing this, that one-way functions exist? (A one-way function is easy to compute but computationally infeasible to invert.) The other promising direction for investigation is to find a *really* useful problem that the quantum computer can solve efficiently, for which no efficient classical algorithms are known. This would be interesting even lacking a proof that the problem in question is genuinely hard for classical computers.

Although building a quantum computer is beyond current technology (but see [23] for a proposal), this should not discourage research into quantum computing. Indeed, quantum physics has been used successfully for purposes closely related to computation in a prototype that demonstrates the technological feasibility of quantum cryptography [4, 5]. Ten years ago, quantum cryptography was still pure science-fiction. A good way of pumping funding into the building of an actual quantum computer would be to find an efficient quantum factoring algorithm!

It is our hope that quantum computers will come into existence during our lifetime and that they will be harnessed to computational tasks beyond the reach of even the fastest possible classical computers. But should it turn out that there is nothing "useful" the quantum computer can do faster than a classical computer, or should building it remain forever beyond the reach of technology, the words of Feynman would echo in our minds: "Such nonsense is very entertaining to professors like me. I hope you will find it interesting and entertaining also" [16].

Acknowledgements

This work has benefited from discussions with Charles H. Bennett, Ethan Bernstein, Claude Crépeau, David Deutsch, Artur Ekert, Lane Hemachandra, Richard Jozsa, Dan Simon, Seinosuke Toda and Umesh Vazirani. The idea of considering oracle worlds was sparked by the need to give a talk at the first New Mexico Theory Day, organized by Tim Long. The seed for Corollary 3 was planted by a question asked by Lane Hemachandra during that talk, and he also deserves our thanks for suggesting that we investigate the relevance of the class $\mathbf{C_{==}P}$. We are grateful to Charles H. Bennett for pointing out to us that our original relativized separation between **EQP** and **ZQP** (Theorem 7) actually provided a separation between **EQP** and **NP**. The observation that an exponential number of processors working in parallel for exponential time would not suffice to simulate the quantum computer's calculation of V_Z is due to Lev Levitin.

We are also grateful to the Rank Foundation (England), the I.S.I. & Elsag–Baley (Italy), and the I.E.S.S. (Germany) for making possible the 1993 workshops on quantum information theory, quantum computing and quantum cryptography, which contributed to bring these fascinating topics to the forefront of international scientific research.

References

[1] Adleman, L. and K. Manders, "Reducibility, randomness, and intractability", *Proceedings of the 9th Annual ACM Symposium on Theory of Computing*, 1977, pp. 151 – 163.

[2] Baker, T., J. Gill and R. Solovay, "Relativizations of the $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ question", *SIAM Journal on Computing*, Vol. 4, 1975, pp. 431 – 442.

[3] Bennett, C. H., "Logical Reversibility of Computation", *IBM Journal of Research and Development*, Vol. 17, 1973, pp. 525 – 532.

[4] Bennett, C. H., F. Bessette, G. Brassard, L. Salvail and J. Smolin, "Experimental quantum cryptography", *Journal of Cryptology*, Vol. 5, no. 1, 1992, pp. 3 – 28.

[5] Bennett, C. H., G. Brassard and A. K. Ekert, "Quantum Cryptography", *Scientific American*, October 1992, pp. 50 – 57.

[6] Bernstein, E. and U. Vazirani, "Quantum Complexity Theory", *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, 1993, pp. 11 – 20.

[7] Berthiaume, A. and G. Brassard, "The quantum challenge to structural complexity theory", *Proceedings of 7th Annual IEEE Symposium on Structure in Complexity Theory*, June 1992, pp. 132 – 137.

[8] Berthiaume, A. and G. Brassard, "Oracle Quantum Computing", *Proceedings of the Workshop on Physics and Computation — PhysComp '92*, October 1992, IEEE Press, pp. 195 – 199.

[9] Blum, M., "A machine independent theory of the complexity of recursive functions", *Journal of the ACM*, Vol. 14, no. 2, 1967, pp. 322 – 336.

[10] Cook, S. A., "The complexity of theorem proving procedures", *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, 1971, pp. 151 – 158.

[11] Deutsch, D., "Quantum theory, the Church–Turing principle and the universal quantum computer", *Proceedings of the Royal Society*, London, Vol. A400, 1985, pp. 97 – 117.

[12] Deutsch, D., in *Quantum Concepts in Space and Time* (R. Penrose and C. Isham, editors), Clarendon Press, Oxford, 1986, pp. 215 – 225.

[13] Deutsch, D. and R. Jozsa, "Rapid solution of problems by quantum computation", *Proceedings of the Royal Society*, London, Vol. A439, 1992, pp. 553 – 558.

[14] Fenner, S. A., L. Fortnow and S. Kurtz, "Gap-definable counting classes", *Proceedings of 6th Annual IEEE Conference on Structure in Complexity Theory*, 1991, pp. 30 – 42.

[15] Feynman, R., "Simulating physics with computers", *International Journal of Theoretical Physics*, Vol. 21, nos. 6/7, 1982, pp. 467–488.

[16] Feynman, R., "Quantum mechanical computers", *Foundations of Physics*, Vol. 16, no. 6, 1986, pp. 507–531. (Originally appeared in *Optics News*, February 1985.)

[17] Feynman, R., "Negative probability", in *Quantum Implications—Essays in Honour of David Bohm* (B. J. Hiley and F. D. Peat, editors), Routledge & Kegan Paul, London, 1987, pp. 235–248.

[18] Gill, J., "Computational complexity of probabilistic Turing machines", *SIAM Journal on Computing*, Vol. 6, 1977, pp. 675 – 695.

[19] Grollmann, J. and A. L. Selman, "Complexity measures for public-key cryptosystems", *SIAM Journal on Computing*, Vol. 17, 1988, pp. 309 – 335.

[20] Jozsa, R., "Characterizing classes of functions computable by quantum parallelism", *Proceedings of the Royal Society*, London, Vol. A435, 1991, pp. 563 – 574.

[21] Karp, R., "Reducibility among combinatorial problems", in *Complexity of Computer Computations* (R.E. Miller and J.W. Thatcher, editors), Plenum Press, New York, 1972, pp. 85 – 104.

[22] Lecerf, Y., "Machines de Turing réversibles. Récursive insolubilité en $n \epsilon N$ de l'équation $u = \theta^n$ ou θ est un isomorphisme de codes", *Comptes Rendus de L'Académie Française des Sciences*, Vol. 257, 1963, pp. 2597 – 2600.

[23] Lloyd, S., "A potentially realizable quantum computer", *Science*, Vol. 261, 17 September 1993, pp. 1569 – 1571.

[24] Simon, D., "On the power of quantum computation", Manuscript, 1993.