Electronic Colloquium on Computational Complexity - Reports Series 1996 - available via:
ECCC
TR96-003
FTP: ftp.eccc.uni-trier.de:/pub/eccc/
WWW: http://www.eccc.uni-trier.de/eccc/
Email: ftpmail@ftp.eccc.uni-trier.de with subject 'help eccc'

Quantum measurements and the Abelian Stabilizer Problem

A. Yu. Kitaev

L.D.Landau Institute for Theoretical Physics,
Kosygina St. 2, Moscow 117940, Russia
e-mail: kitaev@itp.ac.ru

December 4, 1995

Abstract

We present a polynomial quantum algorithm for the Abelian stabilizer problem which includes both factoring and the discrete logarithm. Thus we extend famous Shor's results [7]. Our method is based on a procedure for measuring an eigenvalue of a unitary operator. Another application of this procedure is a polynomial quantum Fourier transform algorithm for an arbitrary finite Abelian group. The paper also contains a rather detailed introduction to the theory of quantum computation.

Introduction

It has been known for long time that all "reasonable" computation models are equivalent. Moreover, every universal machine A can simulate any other machine B with at most polynomial slowdown. For instance, a computation, which takes time t on a random access memory (RAM) machine, can be done in time $O(t^2)$ on a Turing machine. (The slowdown is nonlinear because the Turing machine has to scroll its tape to access a distant memory cell). In view of this equivalence, theoretical computer scientists classify algorithms as polynomial[1] and superpolynomial, the former being considered efficient, the latter inefficient. A polynomial algorithm remains polynomial when adapted to another machine model.

Many physical phenomena can be simulated on a computer in polynomial time, although it is sometimes impracticable because of the great number of particles involved. However, simulation of quantum mechanics may be computationally expensive even with few particles. Consider a system with 2 states. If we take n copies of this system we will get a new system with 2^n states. Its quantum evolution (for a given time interval) is characterized by a unitary matrix of size $2^n \times 2^n$. Unless one invents a more intelligent method, simulation of the evolution amounts

[1] An algorithm (for a given problem and a given machine model) is called polynomial if the number of steps of the algorithm grows not faster than some power of the size of the input.

to multiplication of evolution matrices corresponding to very short time intervals. It takes exponential time to compute one separate item of the product. (However, such computation can be done with polynomial memory).

But if quantum mechanics is really difficult to simulate, a quantum computer should be more powerful than the classical one. How to know it for certain? A quantum computer is still an imaginary device which has not been constructed yet. Not thinking about technology, there are 3 fundamental questions to be answered.

1. Is there any simple and universal model of quantum computation?

2. Can a quantum computer solve a computational problem which is known to be hard for a classical computer?

3. As far as the group of unitary transformations $\mathbf{U}(2^n)$ is continuous: To what extent is quantum computation sensitive to perturbation? And is it possible to organize computation so that a moderate perturbation would not affect the result?

Quantum devices for doing classical computation were suggested by Benioff [1], Peres [2] and Feynmann [3]. Deutsch [4, 5] was the first to give an explicit model of quantum computation. He defined both quantum Turing machines and quantum circuits. Yao [6] showed that these two models are equivalent. More specifically, quantum Turing machines can simulate, and be simulated by, uniform families of polynomial size quantum circuits, with at most polynomial slowdown. Quantum circuits are generally more convenient for developing quantum algorithms.

Quantum circuits are rather generic quantum systems which can simulate other quantum systems. We have seen that such simulation may be problematic with a classical computer, so the answer to the second question is probably "yes". However, we do not know whether simulating quantum mechanics on a classical computer is really hard. In fact, if no efficient algorithm is known for a problem, it doesn't mean that such an algorithm doesn't exist. Unfortunately, no reasonable computational problem has been *proven* to be hard yet. So it is interesting to find efficient quantum algorithms for problems which are considered as hard by computer science experts. The most remarkable result of this type has been obtained by Shor [7] who invented polynomial quantum algorithms for the discrete logarithm and factoring of integers. However, it is not clear yet whether a polynomial quantum algorithm exists for an NP-complete problem.

In order to obtain a correct result under perturbation, every step of the computation must be done with precision $c\,(\text{number of steps})^{-1}$ (the constant c depends on the allowed error probability, see Sec. 2.4). Thus the number of precision bits, needed to specify each elementary quantum operator (gate), is logarithmic [8]. This precision requirement is rather weak, which gives hope that quantum computation can be done by a physical device. Note that exponential precision (i.e. polynomial number of precision bits) is almost certainly infeasible; fortunately, it is not needed for quantum computation. However, even polynomial precision may prove to be impractical. A fully satisfactory solution would be to do arbitrarily long computation with fixed gate precision, by use of some error correction procedure. Alternatively, one should ensure high precision by some physical mechanism beyond the formal computation model. Precision still remains the most important problem in the field of quantum computation.

In this paper we suggest a polynomial quantum algorithm for a so-called Abelian Stabilizer Problem (ASP) which includes both factoring and the discrete logarithm. Thus we reproduce Shor's result by a different method. Another special case of the ASP was studied by Grigoriev [9]

in connection with the shift equivalence problem for polynomials. The ASP should have some applications to computational problems in number theory and algebraic geometry, but this topic needs a separate study.

The key point of our solution is a concept of quantum measurement. We also use a generalization of Simon's procedure [10] for finding a certain group of characters. In Sec. 5 we demonstrate a more subtle use of quantum measurements by describing a polynomial algorithm for the Quantum Fourier Transform (QFT) on an arbitrary finite Abelian group. This doesn't solve any classical computational problem because the QFT is defined in terms of quantum mechanics. However, the construction itself may be interesting. Polynomial QFT algorithms were known for groups $(\mathbf{Z}_2)^k$ [11] and $\mathbf{Z}_q$, where $q = 2^n$ [12] or q is a smooth number, i.e. contains no prime power factor larger than $(\log q)^c$ [7].

1 The Abelian Stabilizer Problem

Let G be a group acting on a finite set M. Suppose that this action and the group operations in G can be computed easily. Compute the stabilizer of a given element $a \in M$. This problem (still to be formulated in a more rigorous language) includes many interesting cases, e.g. graph isomorphism. Unfortunately, we are not able now to treat the problem in its generality. Rather, we will assume that the group G is Abelian. As far as any finitely generated Abelian group is a homomorphic image of $\mathbf{Z}^k$, we may set w. l. o. g. $G = \mathbf{Z}^k$. [2] We will also assume that the set M can be identified, by some one-to-one coding, with a subset of a Boolean cube $\mathbf{B}^n = \{0,1\}^n$. (Our algorithm does not work if each element of M have many representations in $\mathbf{B}^n$, even if the equivalence of these representations can be checked by an efficient procedure). This restricted problem is called the Abelian Stabilizer Problem (ASP). We proceed with an exact definition.

An ASP (more exactly, an instance of the ASP) consists of the following items:

- Two positive integers k and n. The pair (k, n) is called the size of the problem.
- An element $a \in \mathbf{B}^n$.
- A function $F : \mathbf{Z}^k \times M \to M \quad (a \in M \subseteq \mathbf{B}^n)$, such that

$$F(0, x) = x \qquad F(g + h, x) = F(g, F(h, x)) \qquad \text{for any} \quad g, h \in \mathbf{Z}^k, \quad x \in M$$

The function F should be regarded as a blackbox subroutine which receives an input $(g, x) \in \mathbf{Z}^k \times \mathbf{B}^n$ and produces an output $y \in \mathbf{B}^n$, so that $y = F(g, x)$ for every $g \in \mathbf{Z}^k$, $x \in M$. (If $x \notin M$, the subroutine may fail or give an arbitrary result. We do not assume that the condition $x \in M$ is checkable). This subroutine F can be invoked by a quantum computer in the way precisely defined in Sec. 2.

Remark. In all reasonable applications (see examples below) the function F can be computed in polynomial time. A quantum computer can do this job itself, so there is no need to use a blackbox subroutine in this case. Let us describe this situation more exactly. Denote by $size(g)$ the number of bits needed to represent an element $g \in \mathbf{Z}^k$ (in a reasonable coding).[3] Let *poly*

[2] For the group $G = (\mathbf{Z}_p)^k$, Grigoriev [9] designed a quantum algorithm which was polynomial in k and p (but not in $\log p$).

[3] In different reasonable codings $size(g)$ may differ at most by a constant factor.

stand for any function that grows not faster than a polynomial, i.e. $poly(x) = x^{O(1)}$. Suppose that the subroutine F is a classical or even quantum machine (see Sec. 2 for explicit models) which computes $F(g,x)$ in time $poly(size(g)+n)$ at most. With a fixed function $poly$, this defines a restricted class of ASPs. In this case we will get a polynomial quantum algorithm which uses a *description* of the machine F rather than invokes it as a subroutine.

The stabilizer of a with respect to F is the set $\mathrm{St}_F(a) = \{g \in \mathbf{Z}^k : F(g,a) = a\}$. This is a subgroup in $\mathbf{Z}^k$ of index $\leq |M| \leq 2^n$. Hence $\mathrm{St}_F(a)$ is isomorphic to $\mathbf{Z}^k$ and has a basis $(g_1,\ldots,g_k)$ of polynomial size, meaning that $\sum_{j=1}^k size(g_j) \leq poly(n+k)$. Any such basis is acceptable as a solution of the ASP. There is an efficient procedure which checks whether $(g_1,\ldots,g_k)$ and $(g'_1,\ldots,g'_k)$ represent the same subgroup in $\mathbf{Z}^k$. Given a subgroup $A \subseteq \mathbf{Z}^k$ of rank k represented by a polynomial size basis, one can compute (by a very simple polynomial algorithm) a unique *canonical basis* $(h_1,\ldots,h_k)$. This basis is given by the columns of the matrix $(m_{ij},\ i,j=1,\ldots,k)$ uniquely characterized by the conditions

$$\begin{aligned} m_{ij} &= 0 \qquad \text{if} \quad i > j \\ m_{ii} &> 0 \\ 0 \leq m_{ij} &< m_{ii} \qquad \text{if} \quad i < j \end{aligned} \tag{1}$$

Thus finding an arbitrary polynomial size basis for the stabilizer is equivalent to finding the canonical one.

Factoring and the discrete logarithm can be reduced to the ASP. Let M be the ring of integers modulo q, G the group of invertible elements of M. If $g_1,\ldots,g_k \in G$ then $F_{g_1,\ldots,g_k} : (m_1,\ldots,m_k,x) \mapsto g_1^{m_1},\ldots,g_k^{m_k}x \quad (m_i \in \mathbf{Z},\ x \in M)$ is an action of $\mathbf{Z}^k$ on M. Consider two cases.

1. **Factoring.** The stabilizer of 1 with respect to F_g gives the order of an element g in the group G. There is a randomized reduction from factoring to the order of an element [13]. (A sketch of this reduction can be found in Shor's paper [7]).

2. **Discrete logarithm.** Let q be a prime, $\zeta \in G \cong \mathbf{Z}_{q-1}$ a primitive element, $g \in G$ an arbitrary element. The stabilizer of 1 with respect to $F_{\zeta,g}$ is $P = \{(m,r) \in \mathbf{Z}^2 : \zeta^m g^r = 1\}$. Given a basis of the subgroup $P \subseteq \mathbf{Z}^2$, we can find an element of the form $(m,-1) \in P$. Then $\zeta^m = g$.

2 Computation models

This section is intended mostly for a reader not familiar with the subject. We define the models usually used in the field of quantum computation. A more experienced reader should just pay attention to a few non-common terms and notations.

In Sec. 2.1 we define Boolean circuits and operation sequences. These two models are trivially equivalent. The language of operation sequences is not quite common but we find it convenient. It is closer to an intuitive model of computation and allow simpler notations. (Circuits can be nicely represented by diagrams, but we do not use diagrams in this paper). We also briefly discuss the concept of uniformity.

In Sec. 2.2 we overview the concept of reversible computation introduced by Lecerf [14] and Bennett [15]. This is an important link between the standard models (e.g. Boolean circuits) and quantum computation. The results of this section have quantum analogues (see Sec. 5).

In Sec. 2.3 we summarize the basic concepts and notations of quantum mechanics.

In Sec. 2.4 we give a formal model of quantum computation and discuss its basic properties.

2.1 Boolean circuits and operation sequences

From now on, we often use functions of type $f : \mathbf{B}^n \to \mathbf{B}^m$. We write $n = \delta(f),\ \ m = \rho(f)$.

Let $\mathcal{B}$ be a set of such functions to be used as elementary blocks for building more complicated functions. The set $\mathcal{B}$ is called a *basis*; its elements are called *gates*. Usually one uses the standard basis $\mathcal{C} = \{\neg, \wedge\}$ (negation and the "and" function). This basis is *complete*, that is any Boolean function can be represented as a composition of the basis elements.

Let $F : \mathbf{B}^n \to \mathbf{B}^m$ be an arbitrary function. A *Boolean circuit* for F is a procedure which converts an input $x \in \mathbf{B}^n$ to the output $y = F(x) \in \mathbf{B}^m$ working with auxiliary Boolean variables $z_1, \ldots, z_K$ according to the following instructions:

1. Copy x to $(z_1, \ldots, z_n)$.
2. Compute $z_{n+1}, \ldots, z_K$ in sequel, using some gates $f_i \in \mathcal{B}\ \ (i = 1, \ldots, L)$ and variables already computed. More specifically,
$$\Big(z_{k_i+1}, \ldots, z_{k_i+\rho(f_i)}\Big) \ = \ f_i\Big(z_{\alpha(i,1)}, \ldots, z_{\alpha(i,\delta(f_i))}\Big) \qquad \alpha(i,1), \ldots, \alpha(i,\delta(f_i)) \ \le \ k_i \qquad (2)$$
where $k_1 = n,\ \ k_{i+1} = k_i + \rho(f_i)$.
3. Read y from $(z_{\beta(1)}, \ldots, z_{\beta(m)})$.

Thus a Boolean circuit is defined by the sequence of functions $f_1, \ldots, f_L \in \mathcal{B}$ and the numbers $\alpha(i,j),\ \beta(j)$. The number L is called the *size* of the circuit. For more generality, we may assume F to be a *partial function* $\mathbf{B}^n \to \mathbf{B}^m$, that is a function $N \to \mathbf{B}^m$, where $N \subseteq \mathbf{B}^n$. In this case the output y must coincide with $F(x)$ for every $x \in N$.

Circuits and algorithms. A Boolean circuit can only work with inputs of fixed size. However, computation problems are usually defined for inputs of variable size — consider, for example, the number addition problem $(x, x') \mapsto x + x'$. Any reasonable computational problem can be represented by a family of functions $F|_s : \mathbf{B}^s \to \mathbf{B}^{poly(s)}$, each corresponding to a particular input size s. One needs a separate Boolean circuit for each $F|_s$. If a polynomial algorithm exists for F then each function $F|_s$ can be computed by a circuit $\mathcal{F}_s$ of size $poly(s)$. In fact, a computer (e.g. a Turing machine), working in space $\times$ time $\le t \times t$, can be simulated by a Boolean circuit of size $O(t^2)$.

However, the existence of a polynomial size circuit $\mathcal{F}_s$ for each $F|_s$ does not necessarily imply that the total function F can be efficiently computed. For this, one must be able to construct the circuits $\mathcal{F}_s$ efficiently. More exactly, the function $s \mapsto \mathcal{F}_s$ must be computable on a Turing machine in polynomial time. A family of circuits $(\mathcal{F}_s)$, which satisfies this condition, is called *uniform*. Thus the machine produces a circuit, and the circuit computes the function.

This two-level construction is especially good for defining non-standard computation models, including quantum computation. In Sec. 2.4 we will define some theoretical quantum devices which can compute Boolean functions. Although these devices operate in a quantum way, they allow classical description, i.e. each particular device can be represented by a binary word. By a quantum algorithm for a problem F we will mean a *classical* algorithm which constructs a quantum device Φ_s for each function $F|_s$.

In a Boolean circuit the value of each variables z_i is computed only once. However, in real computers memory cells can be reused to operate with new information. At each step the computer does some operation with a few memory cells. Let us give a simple model of such computation.

Denote by $\Delta = \{1, \dots, K\}$ the memory to be used in computation. Each memory element (bit) $i \in \Delta$ represents a Boolean variable z_i. Any ordered collection of bits is called a *register*. Associated with a register $A = (A_1, \dots, A_n)$ is the variable $z_A = (z_{A_1}, \dots, z_{A_n})$ taking values from $\mathbf{B}^n$. (Here $A_i \in \Delta$ and $A_i \neq A_j$ if $i \neq j$). Given a Boolean operator $g : \mathbf{B}^n \to \mathbf{B}^n$, we can define its action $g[A] : z_A \mapsto g(z_A)$ on the set of states of the register A. (By a *Boolean operator* we mean an arbitrary mapping of a Boolean cube into itself). We may regard $g[A]$ as an operator on the total set of memory states, $\Gamma = \mathbf{B}^\Delta$.

Now take some set (basis) $\mathcal{B}$ of Boolean operators. Operators of the form $g[A]$ $(g \in \mathcal{B})$ will be called *operations*. The new model is a procedure of the following type:

1. Place the input into some register X. Set all the other bits equal to 0.
2. Do some operations $g_1[A_1], \dots, g_L[A_L]$ $(g_i \in \mathcal{B})$ one by one.
3. Read the output from some register Y.

A Boolean circuit may be considered as the sequence of operators

$$g_i : \ \Big(u_1, \dots, u_{\delta(f_i)},\ v_1, \dots, v_{\rho(f_i)}\Big) \ \mapsto \ \Big(u_1, \dots, u_{\delta(f_i)},\ f_i(u_1, \dots, u_{\delta(f_i)})\Big) \tag{3}$$

applied to the registers $A_i = \Big(\alpha(i,1), \dots, \alpha(i, \delta(f_i)),\ k_i+1, \dots, k_i+\rho(f_i)\Big)$. On the other hand, any sequence of L operation can be simulated by a Boolean circuit of size L — one should just reserve a separate variable for each new Boolean value that appears during computation. So these two models are equivalent.

2.2 Reversible computation

The models defined above, as well as operation of a real computer, are not reversible. In fact, even erasing a bit (i.e. setting it equal to 0) is not reversible. However, the laws of quantum mechanics are reversible, since the inverse of a unitary matrix exists and is also a unitary matrix. So, before passing on to quantum computation, one must be able to do classical computation reversibly.

Of course, reversible computation must use only bijective gates $g_i : \mathbf{B}^n \to \mathbf{B}^n$, i.e. permutation on Boolean cubes. A simple but important example is the bijective operator $\tau_n : (u, v) \mapsto (u,\ v \oplus u)$ on $\mathbf{B}^{2n}$, where "$\oplus$" stands for the bitwise addition modulo 2. (Obviously, applying τ_n is the same as to apply the operator $\tau = \tau_1$ to each pair of bits). The operator τ_n allows to copy the content of one register into another, provided the second register is empty. For a more general example, consider an arbitrary function $F : N \to \mathbf{B}^m$ $(N \subseteq \mathbf{B}^n)$, then

$$F_\tau : \ N \times \mathbf{B}^m \to N \times \mathbf{B}^m \ : \qquad (u, v) \ \mapsto \ (u,\ v \oplus F(u)) \tag{4}$$

is a bijection. It is quite clear now how to simulate a Boolean circuit by a sequence of bijective operations. Instead of the operators (3) one should take the operators $(f_i)_\tau$. The result will

be the same because $v = (v_1, \ldots, v_{\rho(f_i)}) = (z_{k_i+1}, \ldots, z_{k_i+\rho(f_i)})$ is zero before the operator $(f_i)_\tau$ is applied.

Formally, this observation is enough to proceed with quantum computation. However, the above computation with bijective gates is not truly reversible. In fact, besides the output it produces some "garbage", i.e. extra information which have to be forgotten after the computation is finished. Without this garbage the computer cannot run back from the output to the input. We will see that garbage does not allow to use the result of a computation in an essentially quantum way. It is worth noting that a real computer also produces some sort of garbage, namely heat. (Actually, the existing computers produce much more heat than necessary). It is rather surprising that the garbage in our model can be avoided.

First of all, we are to give an exact definition of computation without garbage, usually called *reversible computation*. In what follows we assume the memory Δ to be the union of two disjoint registers, an input-output register X and an auxiliary register W. Thus a state of the memory is denoted as (x, w), where $x \in \mathbf{B}^X$, $w \in \mathbf{B}^W$.

Definition 1. *Let $G : N \to M$ $(N, M \subseteq \mathbf{B}^n)$ be an arbitrary bijection. A sequence of bijective operations $g_i[A_i]$ $(i = 1, \ldots, L)$ is said to* represent *G, or compute G reversibly, if their composition $g_L[A_L] \circ \ldots \circ g_1[A_1]$ maps $(x, 0)$ to $(G(x), 0)$ for every $x \in N$.*

Lemma 1. *Suppose that a function $F : N \to \mathbf{B}^m$ $(N \subseteq \mathbf{B}^n)$ is computable in a basis $\mathcal{B}$ by a Boolean circuit of size L. Then F_τ can be represented in the basis $\mathcal{B}_\tau = \{f_\tau : f \in \mathcal{B}\} \cup \{\tau\}$ by an operation sequence of length $2L + m$.*

Proof. The Boolean circuit can be simulated by a sequence of L operations from the basis $\mathcal{B}_\tau$. We may assume that this simulation uses registers U, W and Y for input, intermediate results and output, respectively, where $U \cap W = \emptyset$ and $Y \subseteq U \cup W$. The total effect of the simulation can be represented by an operator $G = G[U, W]$. Let V be a new register of size m. Then $X = U \cup V$ can be used as an input-output register for reversible computation of the function F_τ. We can denote a memory state as (u, v, w), where u, v and w stand for the contents of U, V, and W, respectively. The needed reversible computation is given by the operator

$$(G[U,W])^{-1} \circ \tau_m[Y,V] \circ G[U,W] \ : \qquad (u, v, 0) \mapsto (u,\, v \oplus F(u),\, 0) \qquad (u \in N,\ v \in \mathbf{B}^m)$$

Indeed, the operator $G[U, W]$ computes $F(x)$, the operator $\tau_m[Y, V]$ adds it to v modulo 2, and $(G[U, W])^{-1}$ removes the garbage, that is makes w equal to 0. $\square$

Lemma 2. *Let $G : N \to M$ $(N, M \subseteq \mathbf{B}^n)$ be a bijection. Suppose that G and G^{-1} are computable in a basis $\mathcal{B}$ by Boolean circuits of size L and L', respectively. Then G can be represented in the basis $\mathcal{B}_\tau$ by an operation sequence of length $2L + 2L' + 4n$.*

Proof. Let X be the input-output register, Y an auxiliary register of the same size n. We should add also another auxiliary register W to be used implicitly in the reversible subroutines G_τ and $(G^{-1})_\tau$. By the previous lemma, these subroutines need $2L + n$ and $2L' + n$ operations, respectively. The required computation is given by the operator

$$\tau_n[X,Y] \circ \tau_n[Y,X] \circ (G^{-1})_\tau[Y,X] \circ G_\tau[X,Y]$$

Indeed, $(x, 0) \mapsto (x, G(x)) \mapsto (0, G(x)) \mapsto (G(x), G(x)) \mapsto (G(x), 0)$. $\square$

Corollaries.

1. Any permutation of n bits can be done by $4n$ operations τ.

2. The basis $\mathcal{C}_\tau$ is complete for reversible computation.

The gate $\neg_\tau \in \mathcal{C}_\tau$ may be replaced with $\neg$. Thus we get another complete basis $\mathcal{R} = \{\neg, \tau, \wedge_\tau\}$. [4] We will always use this basis unless we speak about so-called relative computation, that is *computation with a blackbox subroutine.*

Definition 2. *Let $F : \mathbf{B}^n \to \mathbf{B}^m$ be an arbitrary function, possibly partial. Reversible computation in the basis $\mathcal{R} \cup \{F_\tau\}$ is called* reversible computation with subroutine F.

This definition is natural due to Lemma 1. Actually, we will need only one particular case of a blackbox subroutine.

Let $F : \mathbf{Z}^k \times M \to \mathbf{B}^n$ be the function from the definition of an ASP. It is not a Boolean function, so the above definition should be modified. Let

$$\mathcal{Z}_s^k \;=\; \Big\{ g \in \mathbf{Z}^k : \; size(g), size(-g) \le s \Big\} \tag{5}$$

We can identify $\mathcal{Z}_s^k$ with a certain subset of $\mathbf{B}^s$. Denote by $F|_{s+n}$ the restriction of F to $\mathcal{Z}_s^k \times M$. By computation with the subroutine F we will mean computation with $F|_{s+n}$, where $s = poly(k+n)$. Our quantum algorithm will use the following bijection

$$G : \mathbf{Z}^k \times M \;\to\; \mathbf{Z}^k \times M \;\; : \qquad (g, x) \;\mapsto\; (g, F(g, x)) \tag{6}$$

Note that $G^{-1} : (g, x) \mapsto (g, F(-g, x))$. The function $G|_{s+n}$, the restriction of G, may be considered as a partial bijective operator on $\mathbf{B}^s \times \mathbf{B}^n$. By Lemma 2, this function can be easily computed with the subroutine $F|_{s+n}$.

2.3 The quantum formalism

In this subsection we remind the reader the quantum formalism for a system with a finite set of states Γ. In the computation-theoretic context, $\Gamma = \mathbf{B}^\Delta$ is the set of states of a computer memory Δ.

A *quantum state* is characterized by a unit vector $|\psi\rangle$ in the complex space $\mathbf{C}(\Gamma) = \mathbf{C}^\Gamma$ equipped with a Hermitian scalar product $\langle \cdot | \cdot \rangle$. To be exact, the term "quantum state" is usually used to denote a one-dimensional subspace of $\mathbf{C}(\Gamma)$, i.e. a unit vector up to a phase factor $e^{i\phi}$. [5] Corresponding to the *classical states* $a \in \Gamma$ are the *standard vectors* $|a\rangle \in \mathbf{C}(\Gamma)$ which form an orthonormal basis of $\mathbf{C}(\Gamma)$. Time evolution of a quantum system is given by a transformation of the form $|\psi\rangle \mapsto U|\psi\rangle$, where U is a unitary operator. Any bijection $G : \Gamma \to \Gamma$ may be regarded as a unitary operator acting by the rule $G|a\rangle = |G(a)\rangle$. Such operators are called *classical.*

Elements of $\mathbf{C}(\Gamma)$ are usually denoted like $|\xi\rangle$, even if the symbol in the brackets is never used alone. The scalar product of two vectors $|\xi\rangle, |\eta\rangle \in \mathbf{C}(\Gamma)$ is denoted by $\langle \xi | \eta \rangle$. Thus $\langle \xi |$

[4] The gate τ can be represented in terms of $\wedge_\tau$ and $\neg$, so it is not necessary.

[5] This definition is motivated by the fact that the probability (7) is invariant under the transformation $|\psi\rangle \mapsto e^{i\phi}|\psi\rangle$, so the phase factor may be neglected in many cases.

stands for the linear functional $|\eta\rangle \mapsto \langle\xi|\eta\rangle$ on $\mathbf{C}(\Gamma)$. The space of such functionals is denoted by $\mathbf{C}(\Gamma)^*$. If $|\xi\rangle = \sum_{j\in\Gamma} c_j|j\rangle$ then $\langle\xi| = \sum_{j\in\Gamma} c_j^*\langle j|$. In the coordinate representation

$$|\xi\rangle \;=\; \begin{pmatrix} c_1 \\ \vdots \\ c_k \end{pmatrix} \qquad\qquad \langle\xi| \;=\; \left(c_1^*, \ldots, c_k^*\right)$$

If $|\xi\rangle, |\eta\rangle \in \mathbf{C}(\Gamma)$ then $|\xi\rangle\langle\eta|$ is an element of $\mathbf{C}(\Gamma) \otimes \mathbf{C}(\Gamma)^*$ and thus may be considered as a linear operator on $\mathbf{C}(\Gamma)$. The result of the application of a linear operator $A : \mathbf{C}(\Gamma) \to \mathbf{C}(\Gamma)$ to a vector $|\xi\rangle$ is denoted by $A|\xi\rangle = |A\xi\rangle$. Thus

$$\langle\xi|A\eta\rangle \;=\; \langle\xi|A|\eta\rangle \;=\; \langle A^\dagger\xi|\eta\rangle \qquad\qquad \langle\xi|A \;=\; \langle A^\dagger\xi|$$

where $A^\dagger$ is the operator adjoint to A.

The algebra of linear operators $\mathbf{C}(\Gamma) \to \mathbf{C}(\Gamma)$ is denoted by $\mathbf{L}(\Gamma)$, while $\mathbf{U}(\Gamma)$ denotes the group of unitary operators.

Let $\Pi_{\mathcal{M}}|\xi\rangle$ denote the orthogonal projection of a vector $|\xi\rangle$ onto a linear subspace $\mathcal{M} \subseteq \mathbf{C}(\Gamma)$. The projection operator $\Pi_{\mathcal{M}}$ can be represented as $\sum_{j=1}^k |e_j\rangle\langle e_j|$, where $(|e_j\rangle,\ j = 1, \ldots, k)$ is an arbitrary orthonormal basis of $\mathcal{M}$.

Two things are most important in the quantum formalism: the probabilistic interpretation of quantum mechanics and the relation between a system and its subsystems. From the mathematical point of view, the probabilistic interpretation is just a definition of some function called "probability". After the definition is given, one can check that this function does have some basic properties of classical probability. Here we just give the definition. The analogy with the classical case will be fully developed in the Sec. 3 where we introduce conditional probabilities.

The classical probability $P(\mu, M) = \mu(M) = \sum_{j\in M} \mu(j)$ is a function of two arguments: a probability measure μ on Γ and a subset $M \subseteq \Gamma$. (As far as the set Γ is finite, a probability measure is simply a positive function $\mu : \Gamma \to \mathbf{R}$, such that $\sum_{j\in\Gamma} \mu(j) = 1$). Correspondingly, the quantum probability depends on a quantum state $|\xi\rangle$ and a linear subspace $\mathcal{M} \subseteq \mathbf{C}(\Gamma)$

$$P(\xi, \mathcal{M}) \;=\; \langle\xi|\Pi_{\mathcal{M}}|\xi\rangle \tag{7}$$

This quantity can be also represented as $\mathrm{Tr}(\rho\Pi_{\mathcal{M}})$, where $\rho = |\xi\rangle\langle\xi|$ is the *density operator* associated with the state $|\xi\rangle$. In a more general setting, a density operator on Γ is an arbitrary positive Hermitian operator $\rho \in \mathbf{L}(\Gamma)$ with trace 1; the set of such operators is denoted by $\mathbf{D}(\Gamma)$. In this case we write

$$P(\rho, \mathcal{M}) = \;\mathrm{Tr}(\rho\Pi_{\mathcal{M}}) \tag{8}$$

This definition includes the classical probability. Indeed, let $\mathcal{M}$ be the subspace generated by the standard vectors $|a\rangle : a \in M$. Let also $\rho = \sum_{a\in\Gamma} \mu(a)|a\rangle\langle a|$, where μ is a probability measure on Γ. Then $P(\rho, \mathcal{M}) = P(\mu, M)$. Like the classical probability, the quantum probability is additive. Specifically, if $\mathcal{M}$ and $\mathcal{N}$ are orthogonal subspaces then $P(\rho,\, \mathcal{M} \oplus \mathcal{N}) = P(\rho, \mathcal{M}) + P(\rho, \mathcal{N})$. A generic density operator is said to represent a *mixed state* of the system, while quantum states defined above are called *pure*. Time evolution of a density operator is given by the formula $\rho \mapsto U\rho U^\dagger$.

Let our system consist of two subsystems, A and B, that is $\Gamma = \Gamma_A \times \Gamma_B$, where Γ_A and Γ_B are the classical state sets of the subsystems. Two vectors $|\xi_A\rangle \in \mathbf{C}(\Gamma_A)$, $|\xi_B\rangle \in \mathbf{C}(\Gamma_B)$ can be combined to give the vector

$$|\xi_A, \xi_B\rangle \;=\; |\xi_A\rangle \otimes |\xi_B\rangle \;\in\; \mathbf{C}(\Gamma_A) \otimes \mathbf{C}(\Gamma_B) \;=\; \mathbf{C}(\Gamma)$$

One can also define tensor product of linear subspaces, linear operators, unitary operators and density operators. It is clear that

$$P(\rho_A \otimes \rho_B,\, \mathcal{M}_A \otimes \mathcal{M}_B) \;=\; P(\rho_A, \mathcal{M}_A)\, P(\rho_B, \mathcal{M}_B) \tag{9}$$

The most striking difference between quantum mechanics and classical mechanics is that a quantum state of a whole system can not be generally decomposed into states of subsystems. In fact, one can not even define any natural linear mapping $\mathbf{C}(\Gamma) \to \mathbf{C}(\Gamma_A)$. (That is the reason why we have to avoid garbage in computation, see an explanation below). However, a density operator ρ on Γ can be "projected" onto Γ_A to give the density operator

$$\rho_A \;=\; \mathrm{Tr}_B\, \rho \;=\; \sum_{a,b \in \Gamma_A} |a\rangle \left(\sum_{c \in \Gamma_B} \langle a,c|\rho|b,c\rangle \right) \langle b| \tag{10}$$

One may pass on to the projection and consider its evolution separately if the subsystem A does not interact with B in future. Indeed,

$$\mathrm{Tr}_B\Big((U_A \otimes U_B)\, \rho\, (U_A \otimes U_B)^\dagger\Big) \;=\; U_A\, (\mathrm{Tr}_B\, \rho)\, U_A^\dagger \qquad\qquad P\Big(\rho,\, \mathcal{M}_A \otimes \mathbf{C}(\Gamma_B)\Big) \;=\; P(\mathrm{Tr}_B\, \rho,\, \mathcal{M}_A)$$

Note that the projection of a pure state is generally a mixed state.

Finally, let us introduce a concept of a quantum variable, or *observable*.[6] Let Ω be a family of mutually orthogonal linear subspaces of $\mathbf{C}(\Gamma)$. Denote by $\mathcal{V}_?$ the orthogonal complement to $\bigoplus_{\mathcal{V} \in \Omega} \mathcal{V}$. In this setting, we say that an observable z_Ω is defined. Let $\rho \in \mathbf{D}(\Gamma)$, $\mathcal{V} \in \Omega$. Then the quantity $P(\rho, \mathcal{V})$ is called *the probability of z_Ω to have the value $\mathcal{V}$*. Obviously, $\sum_{\mathcal{V} \in \Omega} P(\rho, \mathcal{V}) + P(\rho, \mathcal{V}_?) = 1$. Thus $P(\rho, \mathcal{V}_?)$ is the probability that z_Ω has no value. If $\mathcal{A}$ is a predicate on Ω (i.e. a function $\Omega \to \{\mathrm{true}, \mathrm{false}\}$) then $\mathrm{Prob}_\rho[\mathcal{A}(z_\Omega)]$ denotes the probability of $\mathcal{A}(z_\Omega)$ being true

$$\mathrm{Prob}_\rho[\mathcal{A}(z_\Omega)] \;=\; \sum_{\mathcal{V} \in \Omega:\, \mathcal{A}(\mathcal{V})} P(\rho, \mathcal{V})$$

For example, $\mathrm{Prob}_\rho[z_\Omega = \mathcal{V}] = P(\rho, \mathcal{V})$. The notation $\mathrm{Prob}_\rho[\ldots]$ is convenient because it expresses the intuitive meaning of probability.

2.4 Quantum computation

2.4.1 The basic model

Before giving a formal model of quantum computation, we will describe elementary operations with a quantum system which seem feasible from the physical point of view. From now on, we assume that $\Gamma = \mathbf{B}^\Delta$, where $\Delta = \{1, \ldots, K\}$ is a memory used in computation.

Let $\Delta = A \cup B$, where A and B are two disjoint registers. Thus $\Gamma = \Gamma_A \times \Gamma_B$, where Γ_A and Γ_B are the sets of states of the registers A and B. Let $U \in \mathbf{U}(\mathbf{B}^n)$, where $n = |A|$. As far as the Boolean cube $\mathbf{B}^n$ can be identified with Γ_A, we can define the action $U[A]$ of the operator U on the space $\mathbf{C}(\Gamma_A)$. By tensoring with the unit operator $1[B] \in \mathbf{U}(\Gamma_B)$, we can make $U[A]$ to be an operator on the space $\mathbf{C}(\Gamma)$ corresponding to the whole system. Physical implementation of such an operator seems feasible provided the number n is small.

[6] Our definition of an observable slightly differs from the conventional one.

For each $a \in \Gamma_A$ denote by $\mathcal{W}_a$ the subspace $(|a\rangle) \otimes \mathbf{C}(\Gamma_B) \subseteq \mathbf{C}(\Gamma)$. These subspaces are mutually orthogonal. Thus the *standard observable* z_A associated with the register A is defined. It always have some value $a \in \mathbf{B}^n$, meaning that $\bigoplus_{a \in \mathbf{B}^n} \mathcal{W}_a = \mathbf{C}(\Gamma)$. Given a quantum state $|\xi\rangle$, it is possible to measure the value of the observable z_A, that is to organize some physical procedure which gives a result a with probability $P(\xi, \mathcal{W}_a)$. For this, it is enough to measure the state of the whole memory (the result $c \in \Gamma$ is obtained with probability $P(\xi, c) = |\langle c|\xi\rangle|^2$) and then ignore information contained in the register B. (Certainly, this works for mixed states as well). The measurement destroys the quantum state, so it must be done in the end of computation.

We are going to define a quantum model which is similar to general (i.e. garbage-producing) sequences of bijective operations. (Reversible quantum computation will be considered later on). We assume that computer memory Δ is a disjoint union of the input register X and an auxiliary register W, the output register $Y \subseteq \Delta$ being arbitrary ($|X| = n$, $|Y| = m$). Thus a classical state of the memory can be denoted as (x, w), where $x \in \mathbf{B}^X = \mathbf{B}^n$, $w \in \mathbf{B}^W$.

Definition 3. *Let $\mathcal{B}$ be a basis of unitary operators, $0 < \epsilon < \frac{1}{2}$ an arbitrary constant. A sequence of operations $U_1[A_1], \ldots, U_L[A_L]$ ($U_i \in \mathcal{B}$) is said* to compute a function $F : N \to \mathbf{B}^m$ ($N \subseteq \mathbf{B}^n$) with error probability $\leq \epsilon$ *if*

$$\forall\, x \in N \qquad \mathrm{Prob}_{U|x,0\rangle} \Big[z_Y = F(x) \Big] \;\geq\; 1 - \epsilon \qquad\qquad \textit{where} \quad U \;=\; U_L[A_L] \ldots U_1[A_1]$$

The error probability can be made arbitrary small by repeating the computation several times. Indeed, let us take k different copies of the memory and do the same computation in each of them independently, with the same input $x \in N$. Due to (9), the corresponding outputs $y_1, \ldots, y_k$ may be considered as independent random variables. By definition, the eventual result is y if more than a half of all y_i are equal to y. The total probability of an error or failure does not exceed $\sum_{j \geq k/2} \binom{k}{j} \epsilon^j (1-\epsilon)^{k-j} \;\leq\; \lambda^k$, where $\lambda = 2\,(\epsilon(1-\epsilon))^{1/2} < 1$. Within the scope of polynomial computation, the error probability can be made as small as $\exp(-\,\mathit{poly}(n))$, where *poly* is an arbitrary function of polynomial growth. Note that the original choice of the constant ϵ is not important; one usually sets $\epsilon = \frac{1}{3}$.

Remark. The above procedure can be represented by the formula $y \;=\; \mathrm{MAJ}(y_1, \ldots, y_k)$, where MAJ is a partial function called the *majority function*. To make it work, one must be able to compute this function in the basis $\mathcal{B}$. This is possible, for example, in the classical basis $\mathcal{R}$.

The choice of the basis. In this paper we use the basis $\mathcal{Q} = \mathbf{U}(\mathbf{B}^1) \cup \{\tau, \wedge_\tau\}$. Note that $\neg \in \mathbf{U}(\mathbf{B}^1)$, so $\mathcal{R} \subseteq \mathcal{Q}$. Hence any classical reversible computation can be done in the basis $\mathcal{Q}$. Actually, this basis is complete for quantum computation; even its proper subset $\mathbf{U}(\mathbf{B}^1) \cup \{\tau\}$ is a complete basis [16]. If a blackbox subroutine F is given, we add the operator F_τ to the basis.[7] There is still one problem with our choice: the basis $\mathcal{Q}$ is infinite so infinite information is needed to specify its element. Fortunately, quantum computation can be done with polynomial gate precision (see below). Hence logarithmic number of precision bits is sufficient.

[7] If the F is a partial function, the operator F_τ is partial. In general, a partial unitary operator is a bijective norm-preserving linear operator between two subspaces.

2.4.2 Precision

Precision of a vector $|\xi\rangle \in \mathbf{C}(\Gamma)$ can be characterized by means of the usual (Hermitian) norm $\||\xi\rangle\| = \sqrt{\langle\xi|\xi\rangle}$. There are two natural norms on the space of linear operators $\mathbf{L}(\Gamma)$, the usual operator norm

$$\|A\| \;=\; \sup_{|\xi\rangle\neq 0} \frac{\|A|\xi\rangle\|}{\||\xi\rangle\|}$$

and the trace norm

$$\|A\|_{\mathrm{tr}} \;=\; \mathrm{Tr}\,\sqrt{A^\dagger A} \;=\; \inf\left\{\sum_j \||\xi_j\rangle\|\,\|\langle\eta_j|\| \;:\; \sum_j |\xi_j\rangle\langle\eta_j| = A\right\} \;=\; \sup_{B\neq 0}\frac{|\,\mathrm{Tr}\,AB|}{\|B\|}$$

The most important properties of these norms are as follows

$$\|AB\| \;\le\; \|A\|\,\|B\| \qquad \|AB\|_{\mathrm{tr}},\, \|BA\|_{\mathrm{tr}} \;\le\; \|B\|\,\|A\|_{\mathrm{tr}} \qquad |\,\mathrm{Tr}\,A| \;\le\; \|A\|_{\mathrm{tr}} \tag{11}$$

We say that a unitary operator $\tilde{U}$ *represents a unitary operator* U *with precision* δ if $\|\tilde{U}-U\| \le \delta$. The following lemma shows that errors are simply added through computation but are not amplified.

Lemma 3. *Let $U_1,\ldots,U_L$, $\tilde{U}_1,\ldots,\tilde{U}_L$ be unitary operators. If $\tilde{U}_j$ represents U_j with precision δ_j for $j=1,\ldots,L$ then $\tilde{U}_L\ldots\tilde{U}_1$ represents $U_L\ldots U_1$ with precision $\delta_1+\ldots+\delta_L$.*

Proof. If $L=2$ then $\|\tilde{U}_2\tilde{U}_1 - U_2U_1\| \le \|(\tilde{U}_2-U_2)\tilde{U}_1\| + \|U_2(\tilde{U}_1-U_1)\| \le \|(\tilde{U}_2-U_2)\| + \|\tilde{U}_1-U_1\|$, since a unitary operator has the norm 1. The general case follows by induction. □

The trace norm $\|\cdot\|_{\mathrm{tr}}$ is suitable to characterize precision of density operators. Note that if $|\xi\rangle, |\eta\rangle$ are unit vectors then

$$\Big\||\xi\rangle\langle\xi| - |\eta\rangle\langle\eta|\Big\|_{\mathrm{tr}} = \; 2\sqrt{1-|\langle\xi|\eta\rangle|^2} \;\le\; 2\,\Big\||\xi\rangle - |\eta\rangle\Big\| \tag{12}$$

Lemma 4. *Let Ω be a family of mutually orthogonal linear subspaces of $\mathbf{C}(\Gamma)$. Then for any pair of density operators ρ, γ*

$$\sum_{\mathcal{V}\in\Omega} |P(\rho,\mathcal{V}) - P(\gamma,\mathcal{V})| \;\le\; \|\rho-\gamma\|_{\mathrm{tr}}$$

Proof. The left hand side of this inequality can be represented as $\mathrm{Tr}((\rho-\gamma)B)$, where $B = \sum_{\mathcal{V}\in\Omega}(\pm\Pi_{\mathcal{V}})$. It is clear that $\|B\| \le 1$. Then use the norm properties (11). □

Combining Lemma 3 with the inequality (12) and Lemma 4, we obtain the following

Lemma 5. *Let an operation sequence of length L compute a function with error probability $\le \epsilon$. If each operation is represented with precision δ then the resulting error probability does not exceed $\epsilon + 2L\delta$.*

Thus the necessary gate precision is const L^{-1}. Note that classical (non-reversible) computation can be simulated without error accumulation, by use of error correcting codes.

The notion of precision is also applicable to partial operators. Let U and $\tilde{U}$ be *partial unitary operators on* $\mathbf{C}(\Gamma)$. In other words, $U : \mathcal{N} \to \mathcal{M}$ and $\tilde{U} : \tilde{\mathcal{N}} \to \tilde{\mathcal{M}}$ are bijective linear operators preserving the scalar product, where $\mathcal{N},\mathcal{M},\tilde{\mathcal{N}},\tilde{\mathcal{M}} \subseteq \mathbf{C}(\Gamma)$. We say that the operator $\tilde{U}$ represents U with precision δ if $\mathcal{M}\subseteq\tilde{\mathcal{M}}$, $\mathcal{N}\subseteq\tilde{\mathcal{N}}$ and $\|(U-A)|\xi\rangle\| \le \delta\||\xi\rangle\|$ for any $|\xi\rangle \in \mathcal{N}$. Note that $\tilde{U}^{-1}$ represents U^{-1} with the same precision. Lemma 3 remains valid for partial unitary operators.

2.4.3 Reversible quantum computation

Definition 1 can be extended to the quantum case in a straightforward way. One can also define approximate reversible computation. In view of the above consideration, it is convenient to use the language of partial operators. The set of partial unitary operators on $\mathbf{C}(\Gamma)$ will be denoted by $\check{\mathbf{U}}(\Gamma)$. Denote by ω the partial operator $|0\rangle \mapsto |0\rangle$ on $\mathbf{C}(\mathbf{B}^k)$ (for any k). Let the memory Δ be the union of two disjoint registers, an input-output register X and an auxiliary register W. A state of the memory is denoted as (x, w), where $x \in \mathbf{B}^X$, $w \in \mathbf{B}^W$.

Definition 4. *Let $U \in \check{\mathbf{U}}(\mathbf{B}^n)$. A sequence of operations $U_1[A_1], \ldots, U_L[A_L]$ is said to* represent *U (with precision δ) if the operator $U_L[A_L] \ldots U_L[A_L]$ represents the partial operator $U[X] \otimes \omega[W]$ (with precision δ).*

As in the classical case, a non-reversible quantum computation procedure can be converted into a reversible one (see Sec. 5 for more detail).

2.4.4 Quantum gates with control parameters

Let $U : \mathcal{N} \to \mathcal{M}$ $(\mathcal{N}, \mathcal{M} \subseteq \mathbf{C}(\mathbf{B}^n))$ be a linear operator. Define a new operator $\Lambda(U) : \mathbf{C}(\mathbf{B}^1) \otimes \mathcal{N} \to \mathbf{C}(\mathbf{B}^1) \otimes \mathcal{M}$ by the formula

$$\Lambda(U)\,|a, \xi\rangle \;=\; \begin{cases} |0, \xi\rangle & \text{if } a = 0 \\ |1\rangle \otimes\; U|\xi\rangle & \text{if } a = 1 \end{cases} \tag{13}$$

Thus the operator U is applied or not depending on whether an additional *control bit*[8] is equal to 1 or 0. For example, $\Lambda(\neg) = \tau$, $\Lambda(\tau) = \wedge_\tau$. Another example:

$$\Lambda(e^{i\phi}) \;=\; \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \qquad\qquad (\phi \in \mathbf{R})$$

(The number $e^{i\phi}$ can be considered as a unitary operator on $\mathbf{C}(\mathbf{B}^0)$). It is obvious that

$$\Lambda(UV) \;=\; \Lambda(U)\,\Lambda(V) \qquad\qquad \Lambda(V^{-1}UV)[1, A] \;=\; V^{-1}[A]\;\Lambda(U)[1, A]\;V[A] \tag{14}$$

For a classical operator U, the operator $\Lambda(U)$ can be computed by a Boolean circuit in the basis $\mathcal{C} \cup \{U\}$. Hence it can be represented in the basis $\mathcal{R} \cup \{U, U^{-1}\}$ (by Lemma 2). This does not work in the general case. However, the following statement holds.

Lemma 6. *Let U be a (partial) unitary operator on $\mathbf{C}(\mathbf{B}^n)$, such that $U|0\rangle = |0\rangle$. Then the operator $\Lambda(U)$ can be represented in the basis $\mathcal{Q} \cup \{U\}$ by an operation sequence of length $4n+1$, the gate U being used only once.*

Proof. Let the input-output register be $X = \{1\} \cup A$, where 1 denotes the control bit. Let B be an auxiliary register of size n. The required computation is given by the composition of operators

$$\Lambda(\tau_n)[1, A, B] \quad \Lambda(\tau_n)[1, B, A] \quad U[B] \quad \Lambda(\tau_n)[1, B, A] \quad \Lambda(\tau_n)[1, A, B]$$

□

[8] Note that in our model the control bit is quantum, as all the other bits.

Corollary. For any $U \in \mathbf{U}(\mathbf{B}^1)$ the operator $\Lambda(U)$ can be represented in the basis $\mathcal{Q}$. Indeed, U can be represented as $V^{-1}WVe^{i\phi}$, where $W|0\rangle = |0\rangle$.

Let us also consider a more general type of control. For any function $\mathcal{U} : \mathbf{B}^l \to \check{\mathbf{U}}(\mathbf{B}^n)$ we define the operator

$$\Lambda(\mathcal{U}) \in \check{\mathbf{U}}(\mathbf{B}^l \times \mathbf{B}^n) \qquad \Lambda(\mathcal{U})\,|a,\xi\rangle = |a\rangle \otimes \mathcal{U}(a)|\xi\rangle \tag{15}$$

Lemma 7. *Let $F : \mathbf{B}^k \to \mathbf{B}^l$ be a partial function; $\mathcal{U} : \mathbf{B}^l \to \check{\mathbf{U}}(\mathbf{B}^n)$. Consider two operators, $T = \Lambda(\mathcal{U}) \in \check{\mathbf{U}}(\mathbf{B}^l \times \mathbf{B}^n)$ and $F_T = \Lambda(\mathcal{U} \circ F) \in \check{\mathbf{U}}(\mathbf{B}^k \times \mathbf{B}^n)$. If the function F can be computed by a Boolean circuit of size L in a basis $\mathcal{B}$ then the operator F_T can be represented by an operation sequence of length $2L+1$ in the basis $\mathcal{B}_\tau \cup \{T\}$, the gate T being used only once.*

(Proof is quite similar to the proof of Lemma 1).

As an application of this lemma, we will show how to create an arbitrary unit vector $|\eta\rangle = u|0\rangle + v|1\rangle \in \mathbf{C}(\mathbf{B}^1)$ if u and v are given as control parameters. For simplicity, assume that $u, v \in \mathbf{R}$, that is $u = \cos\theta$, $v = \sin\theta$. Then the vector $|\theta,\eta\rangle$ can be obtained from $|\theta,0\rangle$ by applying the operator $R : |\theta,\xi\rangle \mapsto |\theta\rangle \otimes R_\theta|\xi\rangle$. Here θ is a real number represented, with some precision, in a binary form;

$$R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

Lemma 7 allows to construct the operator R from $\Lambda(R_\theta)$ with $\theta = 2\pi\,2^{-s}$ $(s = 1, 2, \ldots)$.

2.4.5 Some other properties of quantum computation

Simulating classical probability. To simulate classical probabilistic computation, one needs to create random bits. Let us take a quantum bit in the state $2^{-1/2}\big(|0\rangle + |1\rangle\big)$ and copy it to another bit by the operator τ. (Beware that the operator τ copies each classical state entering a quantum superposition, not the whole superposition!) Thus we get the two-bit quantum state $|\psi\rangle = 2^{-1/2}\big(|0,0\rangle + |1,1\rangle\big)$. Then discard the copy (or just not use it in computation). This situation can be described by transition to a density operator corresponding to the first bit only

$$\rho_1 = \mathrm{Tr}_2\big(|\psi\rangle\langle\psi|\big) = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$

This density operator corresponds to the classical probability measure $\mu(0) = \mu(1) = \frac{1}{2}$.

The effect of garbage. Let $G : \mathbf{B}^n \to \mathbf{B}^n$ be a classical operator to be used in quantum computation. Assume that the operator G is computed by a sequence of bijective operations. We are to show that the operator G must be computed without garbage, otherwise quantum coherence will be destroyed. Suppose that garbage is produced. Then G is actually represented by an operator $U : (x,0) \mapsto (G(x), g(x))$ on the total set of memory states. The operator U transforms a quantum state $|\xi\rangle = \sum_x c_x|x\rangle$ into the state $|\psi\rangle = \sum_x c_x\,|G(x), g(x)\rangle$. As far as the garbage is ignored, we should take the trace with respect to the second variable. Thus we get the density operator

$$\rho = \sum_{x,y:\,g(x)=g(y)} c_x^* c_y\,|G(x)\rangle\langle G(y)|$$

If the garbage $g(x)$ is the same for all x then $\rho = G|\xi\rangle\langle\xi|G^\dagger$, so the operator U does what it is supposed to do. Now consider the worst case: different inputs produce different garbage. Then the density operator $\rho = \sum_x |c_x|^2 \, |G(x)\rangle\langle G(x)|$ is classical; it could be obtained if we first measured the value of x and then applied G in a classical way. We conclude that a classical operator can not be used in an essentially quantum way unless it is computed reversibly.

3 Quantum measurements

One of the physical assumptions, underlying the formal model of quantum computation, is the possibility to measure the classical state of the memory. Such measurement is a specific type of interaction between the quantum computer and an external physical device. Description of the measurement procedure is beyond the scope of our formal analysis. However, we can formally define and study another type of measurement in which one part of the computer works as a device measuring the state of another part. We will see that such measurement obeys the usual laws of conditional probability. So, if subsystems $A_1, A_2, \ldots$ measure each other in sequence, this process can be simulated by a Markov chain. This fact is very important for understanding the probabilistic interpretation of quantum mechanics in physical context. We may believe that the chain of measurements extends beyond the system in study, and the last measurement done by an external device is of the same type. Except for this philosophical remark, we will use quantum measurements as a concrete tool for developing quantum algorithms.

Definition 5. *Let A and D be two disjoint registers, Ω a family of mutually orthogonal subspaces of $\mathbf{C}(\mathbf{B}^A)$. Set $\mathcal{N} = \bigoplus_{\mathcal{V}\in\Omega} \mathcal{V}$.*

1. *A* measurement operator *for the observable z_Ω is a linear operator of the form*

$$U \;=\; \sum_{\mathcal{V}\in\Omega} \Pi_{\mathcal{V}} \otimes U_{\mathcal{V}} \quad : \qquad \mathcal{N} \otimes \mathbf{C}(\mathbf{B}^D) \to \mathcal{N} \otimes \mathbf{C}(\mathbf{B}^D)$$

   where $U_{\mathcal{V}}$ are arbitrary unitary operators on $\mathbf{C}(\mathbf{B}^D)$.

2. *A measurement operator U together with a register $C \subseteq D$ is called a* measurement *with result z_C. Let $|C| = m$. Denote by $\mathcal{W}_y$ the subspace of $\mathbf{C}(\mathbf{B}^D)$ corresponding to the situation $z_C = y$, i.e. $\mathcal{W}_y = (|y\rangle) \otimes \mathbf{C}(\mathbf{B}^{D\setminus C})$. The numbers*

$$P_{U,C}(\mathcal{V}, y) \;=\; P\Big(U_{\mathcal{V}}|0\rangle, \mathcal{W}_y\Big) \qquad\qquad (\mathcal{V} \in \Omega, \;\; y \in \mathbf{B}^m)$$

   are called the conditional probabilities *for the measurement (U, C).*

3. *A measurement (U, C) is said to* measure *the value of a function $F : \Omega \to \mathbf{B}^m$ with error probability $\leq \epsilon$ if $P_{U,C}(\mathcal{V}, F(\mathcal{V})) \geq 1 - \epsilon$ for every $\mathcal{V} \in \Omega$.*

For example, the operator $\tau_n[A, D]$ is a measurement for the observable z_A. Any quantum computation (see Definition 3) can be organized as a measurement with respect to its input. For this, it suffice to copy the input by the operator τ_n and use the copy instead of the original. Alternatively, one can use the bits of the input as control parameters, e.g. in operators $\Lambda(U)$.

Important example. Let U be a unitary operator on a subspace $\mathcal{N} \subseteq \mathbf{C}(\mathbf{B}^n)$. The eigenvalues of this operator have the form $\lambda(\phi) = \exp(2\pi i\phi)$, where ϕ is a real number (mod 1). Denote by $\mathcal{E}(U,\phi)$ the corresponding eigenspaces. Without risk of confusion, the corresponding observable may be denoted simply by ϕ.

Let the operator U act on a register A. Denote by 1 an additional bit and introduce the matrix

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

Then the operator

$$\Xi(U)[A,1] \;=\; S[1]\;\; \Lambda(U)[1,A]\;\; S[1] \tag{16}$$

is a measurement operator for the observable ϕ. If $|\xi\rangle \in \mathcal{E}(U,\phi)$ then $\Xi(U)|\xi,0\rangle \;=\; |\xi,\eta\rangle$, where

$$|\eta\rangle \;=\; \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \lambda(\phi) \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \;=\; \begin{pmatrix} \frac{1}{2}(1+\lambda(\phi)) \\ \frac{1}{2}(1-\lambda(\phi)) \end{pmatrix}$$

Hence the conditional probabilities $P_{\Xi(U)}(\phi,y) \;=\; P_{\Xi(U)[A,1],\,1}\,(\mathcal{E}(U,\phi),y)$ are as follows

$$P_{\Xi(U)}(\phi,0) \;=\; \frac{1}{2}\Big(1+\cos(2\pi\phi)\Big) \qquad\qquad P_{\Xi(U)}(\phi,1) \;=\; \frac{1}{2}\Big(1-\cos(2\pi\phi)\Big) \tag{17}$$

General properties of measurement operators and measurements are quite simple. Let us fix a register A and a family Ω of mutually orthogonal subspaces in $\mathbf{C}(\mathbf{B}^A)$. Set $\mathcal{N} = \bigoplus_{\mathcal{V}\in\Omega}\mathcal{V}$. We will consider measurement operators for the same observable z_Ω with different additional registers D.

Lemma 8.

1. *Let (U,C) be a measurement with an additional register $D \supseteq C$. Then for any quantum state $|\xi\rangle \in \mathcal{N}$ the composite probability formula holds*

$$\mathrm{Prob}_{\,U|\xi,0\rangle}\Big[z_C = y\Big] \;=\; \sum_{\mathcal{V}\in\Omega} P(|\xi\rangle,\mathcal{V})\;\, P_{U,C}(\mathcal{V},y) \tag{18}$$

2. *The product of several measurements operators is a measurement operator. Measurement operators with disjoint additional registers commute.*

3. *Let (U',C') and (U'',C'') be measurements with disjoint additional registers $D' \supseteq C'$ and $D'' \supseteq C''$. Set $U = U'U''$, $C = C' \cup C''$. Then*

$$P_{U,C}\Big(\mathcal{V},\,(y',y'')\Big) \;=\; P_{U',C'}(\mathcal{V},y')\;\, P_{U'',C''}(\mathcal{V},y'') \tag{19}$$

Proof.

1. It is clear that $U|\xi,0\rangle = \sum_{\nu\in\Omega} \Pi_\nu U|\xi,0\rangle$. Hence

$$\text{Prob}_{\,U|\xi,0\rangle}\Big[z_C = y\Big] \;=\; \sum_{\nu\in\Omega} \langle\xi,0|U^\dagger\,\Pi_\nu\Pi_{\mathcal{W}_y}\,U|\xi,0\rangle \;=\; \sum_{\nu\in\Omega} \langle\xi|\Pi_\nu|\xi\rangle\;\langle 0|U_\nu^\dagger\,\Pi_{\mathcal{W}_y}U_\nu|0\rangle$$

This gives the right hand side of eq. (18).

2. It follows from the definition.

3. It follows from the general property of quantum probability (9).
□

Now let us return to the example considered above. Suppose that the measurement operator $\Xi(U)$ is applied s times to the same register A and different additional bits $1,\dots,s$. Then one can measure the values $z_1,\dots,z_s$ of these bits and count how many 1's are contained in the resulting sequence of 0's and 1's. Denote this count by y. Thus a new measurement $\Xi_s(U)$ is defined, the number y being its result. Since $z_1,\dots,z_s$ behave as independent random variables, y/s is most likely close to $P_{\Xi(U)}(\phi,1)$. More exactly, for any given constant $\delta>0$

$$\text{Prob}\Big[\;|y/s - P_{\Xi(U)}(\phi,1)| \;>\; \delta\;\Big] \;\le\; 2\,\exp(-c(\delta)\,s)$$

where $c(\delta)>0$. Thus we measure the quantity $P_{\Xi(U)}(\phi,1) = \frac{1}{2}\Big(1-\cos(2\pi\phi)\Big)$ with precision δ and error probability $\le \exp(-c(\delta)\,s)$. If we substitute iU for U then $\cos(2\pi\phi)$ will change to $-\sin(2\pi\phi)$. So we can measure both $\cos(2\pi\phi)$ and $\sin(2\pi\phi)$; this information is enough to find ϕ. We have proved the following

Lemma 9. *Let $\delta>0$ be a constant. For any $\epsilon>0$, the value of the observable ϕ can be measured with precision δ and error probability $\le\epsilon$ by an operation sequence of length $O(\log(1/\epsilon))$ in the basis $\mathcal{Q}\cup\{\Lambda(U)\}$.*

Unfortunately, it is difficult to measure ϕ with arbitrary precision because the cost of measurement (i.e. the length of the operation sequence) grows polynomially in δ. However, the situation is different if we have in our disposal the operators $\Lambda(U^k)$ for all k. More specifically, consider the operator

$$U^{[0,r]}\;:\;\mathbf{C}(\{0,\dots,r\})\otimes\mathcal{N}\;\to\;\mathbf{C}(\{0,\dots,r\})\otimes\mathcal{N} \qquad\qquad U^{[0,r]}\,|a,\xi\rangle \;=\; |a\rangle\otimes U^a|\xi\rangle \quad (20)$$

where $r = 2^l-1$. (Note that the set $\{0,\dots,2^l-1\}$ can be naturally identified with $\mathbf{B}^l$). By Lemma 7, the operators $\Lambda\Big(U^{2^j}\Big)$ $(j=0,\dots,l-1)$ can be represented in terms of $U^{[0,r]}$. It takes $O(\log(l/\epsilon))$ operation to localize each of the numbers $2^j\phi \pmod 1$ in one of the 8 intervals $\Big[\frac{s-1}{8},\frac{s+1}{8}\Big]$ $(s=0,\dots,7)$ with error probability $\le\epsilon/l$. Using this information, one can find (by a polynomial algorithm) the value of ϕ with precision $\frac{1}{8}\,2^{-(l-1)} = 2^{-l-2}$ and error probability $\le\epsilon$. We have obtained the following result

Lemma 10. *Let l be a positive integer; $r=2^l-1$. For any $\epsilon>0$, the value of the observable ϕ can be measured with precision 2^{-l-2} and error probability $\le\epsilon$ by an operation sequence of length $O(l\,\log(l/\epsilon))+poly(l)$ in the basis $\mathcal{Q}\cup\{U^{[0,r]}\}$. (The gate $U^{[0,r]}$ is used at most $O(l\,\log(l/\epsilon))$ times).*

Now consider an important particular case: U is a permutation on a subset $N \subseteq \mathbf{B}^n$. Corresponding to each cycle of the permutation are eigenvalues of the form $\exp(2\pi i \frac{p}{q})$, where q is the length of the cycle. Hence the values of ϕ are rational numbers with denominators $\leq 2^n$. The minimal separation between such numbers is $\left(2^n(2^n-1)\right)^{-1}$. Consequently, the exact value of ϕ can be found by measuring it with precision 2^{-2n-1}. Moreover, the transition from the measured value to the exact one can be performed in polynomial time, using continuous fractions. What follows is a brief proof of this claim.

Suppose that the measurement produced a number $\phi' = p'/q'$ $(0 \leq p' < q' = 2^{2n+1})$, such that $|\phi' - \phi| \leq 2^{-2n-1} \pmod 1$. It is easy to check whether $\phi = 0$, so we will assume that $\phi \neq 0$. Thus $\phi = p/q$, where p and q are mutually prime, $0 < p < q \leq 2^n$. Let us define a sequence of positive integers $(k_1, \ldots, k_s)$ which can be obtained by applying Euclid's algorithm to the pair (q, p)

$$q_{j-1} \;=\; k_j q_j + q_{j+1}\,, \qquad 0 \;\leq\; q_{j+1} \;<\; q_j \qquad\qquad (j = 1, \ldots, s)$$

$$q_0 = q \qquad q_1 = p \qquad\qquad q_s = \text{g.c.d.}(q,p) = 1 \qquad q_{s+1} = 0$$

Since q and p are not known, we can not compute $k_1, \ldots, k_s$ directly. Instead of that, we can apply Euclid's algorithm to the pair (q', p') to get a sequence $(k'_1, \ldots, k'_{s'})$. It is easy to show that

$$\frac{q_j}{q_{j-1}} - \frac{1}{2q_{j-1}^2} \;\leq\; \frac{q'_{j-1}}{q'_j} \;<\; \frac{q_j}{q_{j-1}} + \frac{1}{(2q_{j-1} - 1/q_j)\, q_{j-1}} \qquad\qquad (j = 1, \ldots, s)$$

$$k'_j \;=\; k_j \qquad (j = 1, \ldots, s-1) \qquad\qquad k'_s \;=\; k_s \quad \text{or} \quad k'_s = k_s - 1, \quad k'_{s+1} = 1$$

It follows that $p/q = \mathrm{CF}(0, k'_1, \ldots, k'_s)$ or $p/q = \mathrm{CF}(0, k'_1, \ldots, k'_{s+1})$, where

$$\mathrm{CF}(m) \;=\; m, \qquad\qquad \mathrm{CF}(m_0, m_1, \ldots) \;=\; m_0 + \frac{1}{\mathrm{CF}(m_1, \ldots)}$$

To find $\phi = p/q$, we can compute the numbers $\phi_j = \mathrm{CF}(0, k'_1, \ldots, k'_j)$ $(j = 1, 2 \ldots)$ until $|\phi' - \phi_j| \leq 2^{-2n-1}$. Then $\phi = \phi_j$. We have proved the following

Theorem 1. *Let U be a permutation on a set $N \subseteq \mathbf{B}^n$. Then the value of the corresponding observable ϕ can be measured exactly with error probability $\leq \epsilon$ by an operation sequence of length $poly(n) + O(n)\log(1/\epsilon)$ in the basis $\mathcal{Q} \cup \{U^{[0,2^{2n}]}\}$. (The gate $U^{[0,2^{2n}]}$ is used at most $O(n\, \log(n/\epsilon))$ times).*

4 Quantum algorithm for the ASP

Let (k, n, a, F) be an instance of the ASP, $\mathrm{St}_F(a) = \{g \in \mathbf{Z}^k : F(g,a) = a\}$ its solution. Consider two finite Abelian groups[9]

$$E \;=\; \mathbf{Z}^k / \mathrm{St}_F(a) \qquad\qquad H \;=\; \mathrm{Hom}(E, \mathbf{T}) \;\subseteq\; \mathrm{Hom}(\mathbf{Z}^k, \mathbf{T}) \;=\; \mathbf{T}^k$$

[9]The group H is called the group of characters on E.

where $\mathbf{T} = \mathbf{R}/\mathbf{Z}$ is the group of real numbers modulo 1. Every element h of the group H can be represented by k rational numbers (mod 1) $\phi_1, \ldots, \phi_k \in \mathbf{T}$ with common denominator $q = |E| = |H| \le 2^n$. More specifically, $\phi_j = (h_j, V_j)$, where $(\cdot,\cdot) : H \times E \to \mathbf{T}$ is the natural bilinear mapping, $V_1, \ldots, V_k \in E$ are the images of the basis elements $g_1, \ldots, g_k \in \mathbf{Z}^k$. It is clear that computing $\mathrm{St}_F(a)$ is polynomially equivalent to finding H. (To find H means to find a polynomial subset of $\mathbf{T}^k$ that generates this group). We are going to show how to generate a random element of H using Theorem 1 (cf. [10]). The group H itself can be generated by sufficiently many random elements.

Consider the orbit $N = \{F(g,a) : g \in \mathbf{Z}^k\} \subseteq \mathbf{B}^n$. Obviously, $F(g,a)$ depends only on the image of g in the factor-group E, so we may use the notation $g(a)$ $(g \in E)$. Elements of the group E may be regarded as permutations on the set N. The following vectors are eigenvectors for all the operators $U \in E$

$$|\psi_h\rangle \;=\; \frac{1}{\sqrt{q}} \sum_{g \in E} \exp\Bigl(2\pi i (h,g)\Bigr)\; |g(a)\rangle \qquad\qquad (h \in H) \tag{21}$$

These vectors form an orthonormal basis of $\mathbf{C}(N)$ called the *Fourier basis*. The corresponding eigenvalues are $\lambda_h(U) = \exp(-2\pi i (h,U))$. In particular, if $h = (\phi_1, \ldots, \phi_k)$ then $\lambda_h(V_j) = \exp(-2\pi i \phi_j)$ $(j = 1, \ldots, k)$. Theorem 1 says that we can measure h with error probability $\le k\epsilon$ by an operation sequence of length $k\Bigl(poly(n) + O(n)\log(1/\epsilon)\Bigr)$ in the basis $\mathcal{Q} \cup \{G|_{O(n)}\}$. (The operator G was defined in eq. (6)).

Now a new trick comes. Prepare the classical state

$$|a\rangle \;=\; \frac{1}{\sqrt{q}} \sum_{h \in H} |\psi_h\rangle$$

and measure h. By the composite probability formula (18), the probability to obtain a given value of h is $P(h) = q^{-1} \sum_{h' \in H} P(h', h)$, where h' stands for the actual value of the measured observable; $P(h', h) \ge 1 - k\epsilon$. Hence

$$q^{-1}|L|\,(1 - k\epsilon) \;\le\; \mathrm{Prob}\,[h \in L] \;\le\; q^{-1}|L| + k\epsilon \qquad\qquad \text{for any } L \subseteq H$$

Thus we can generate a random element of H with almost uniform distribution.

Let $h_1, \ldots, h_l \in \mathbf{T}^k$ be independent random elements generated this way. We are to show that they generate H almost certainly, provided l is large enough.

All the elements $h_1, \ldots, h_l$ belong to H with probability $\ge 1 - kl\epsilon$. Suppose that they belong to H but do not generate H. Then $h_1, \ldots, h_l \in L$, where L is a maximal proper subgroup of H. For a given L, the probability of this event does not exceed $\Bigl(\mathrm{Prob}\,[h \in L]\Bigr)^l \le \Bigl(\frac{1}{2} + k\epsilon\Bigr)^l$. Maximal proper subgroups of H are in 1-to-1 correspondence with minimal nonzero subgroups of E. The number of such subgroups is less than $|E| \le 2^n$. Hence the overall probability for $h_1, \ldots, h_l$ not to generate H is less than $kl\epsilon + 2^{n-l}(1 + 2k\epsilon)^l$. (The first term corresponds to the possibility $\{h_1, \ldots, h_l\} \not\subseteq H$ while the second one accumulates contributions from all the subgroups L). Setting $l = n + 4$, $\epsilon = (6kl)^{-1}$ guarantees that the random elements $h_1, \ldots, h_l \in \mathbf{T}^k$ generate H with probability $\ge \frac{2}{3}$.

Thus the whole computation is organized as follows. We take $l = n + 4$ registers and prepare the initial state $|a\rangle$ in each of them. Then we do $O(kn \log(kn))$ elementary measurements $\Xi\Bigl(V_j^{2^s}\Bigr)$ $(1 \le s \le 2n)$ with each register. The results of these measurements are

processed in a classical way, which gives $h_1, \ldots, h_l$ and, eventually, the canonical basis of the stabilizer (with error probability $\leq \frac{1}{3}$). Through this computation, the blackbox subroutine F is invoked $O(kn^2 \log(kn))$ times for inputs of size $O(n)$. We emphasize that our procedure is uniform, meaning that not only the operation sequence has length $poly(k+n)$ but also it can be constructed in time $poly(k+n)$ by a classical Turing machine.

5 How to make quantum computation reversible?

In this section we will show that any quantum computation or quantum measurement can be performed reversibly, i.e. without producing garbage. This allows to use quantum algorithms as subroutines for other algorithms in a non-classical way. In particular, the eigenvalue measurement procedure can be used for the quantum Fourier transform (QFT).

We start with generalizing the definition of quantum computation (Definition 3). Let Ω and Θ be families of mutually orthogonal subspaces in $\mathbf{C}(\mathbf{B}^n)$ and $\mathbf{C}(\mathbf{B}^m)$, respectively. We are going to define quantum computation for functions of type $F : \Omega \to \Theta$. As usually, computer's memory Δ contains an input register X of size n and an output register Y of size m. Elements $\mathcal{V} \in \Omega$, $\mathcal{W} \in \Theta$ may be regarded as linear subspaces of $\mathbf{C}(\mathbf{B}^X)$ and $\mathbf{C}(\mathbf{B}^Y)$, respectively. We will not make distinctions between $\mathcal{V}$ and $\mathcal{V} \otimes (|0_{\Delta \setminus X}\rangle)$, as well as between $\mathcal{W}$ and $\mathcal{W} \otimes \mathbf{C}(\mathbf{B}^{\Delta \setminus Y})$. In other words, all the bits from $\Delta \setminus X$ are initially set to 0, while all the bits from $\Delta \setminus Y$ are ignored in the end.

Definition 6. *A unitary operator $U \in \mathbf{U}(\mathbf{B}^\Delta)$ (usually represented by an operation sequence) is said to compute a function $F : \Omega \to \Theta$ with error probability $\leq \epsilon$ if*

$$\forall\ |\xi\rangle \in \mathcal{V} \in \Omega \qquad P\Big(U|\xi\rangle,\ F(\mathcal{V})\Big)\ \geq\ 1 - \epsilon$$

Now we are in position to formulate an extension of Lemma 7 which itself can be viewed as a generalization of Lemma 1. In the above setting, let $T = \sum_{\mathcal{W} \in \Theta} \Pi_{\mathcal{W}} T_{\mathcal{W}}$ be a measurement operator for the observable z_Θ with an additional register D. Consider the operator $F_T = \sum_{\mathcal{V} \in \Omega} \Pi_{\mathcal{V}} T_{F(\mathcal{V})}$ acting on the space $\mathcal{N} \otimes \mathbf{C}(\mathbf{B}^D)$, where $\mathcal{N} = \bigoplus_{\mathcal{V} \in \Omega} \mathcal{V}$. This is a measurement operator for the observable z_Ω. For applications, it is enough to consider functions of type $F : \Omega \to \mathbf{B}^m$ and take the operator τ_m for T. In this case the operator $F_T = F_{\tau_m}$ (or simply F_τ) measures the value of the function F without producing any garbage. Note that for classical functions F (of type $N \to \mathbf{B}^m$, where $N \subseteq \mathbf{B}^n$) the notation F_τ coincides with the notation from Sec. 2.2.

Theorem 2. *Let a unitary operator U compute a function $F : \Omega \to \Theta$ with error probability $\leq \epsilon$. Let also T be a measurement operator for the observable z_Θ. Then the operator $U^{-1}TU$ represents the operator F_T with precision $2\Big(|\Omega|\epsilon\Big)^{1/2}$.*

Proof. Let $|\xi\rangle \in \mathcal{N} \otimes \mathbf{C}(\mathbf{B}^D)$ be a unit vector. It can be represented as $\sum_{\mathcal{V} \in \Omega} c_{\mathcal{V}} |\xi_{\mathcal{V}}\rangle$, where $|\xi_{\mathcal{V}}\rangle \in \mathcal{V} \otimes \mathbf{C}(\mathbf{B}^D)$ are unit vectors, $c_{\mathcal{V}} \geq 0$ are real numbers. Note that $\sum_{\mathcal{V} \in \Omega} c_{\mathcal{V}}^2 = 1$, hence $\sum_{\mathcal{V} \in \Omega} c_{\mathcal{V}} \leq \|\Omega\|^{1/2}$. Represent each vector $U|\xi_{\mathcal{V}}\rangle$ as $|\zeta_{\mathcal{V}}\rangle + |\psi_{\mathcal{V}}\rangle$, where $|\zeta_{\mathcal{V}}\rangle = \Pi_{F(\mathcal{V})} U|\xi_{\mathcal{V}}\rangle \in F(\mathcal{V}) \otimes \mathbf{C}(\mathbf{B}^D)$. Then $\langle \zeta_{\mathcal{V}} | \zeta_{\mathcal{V}} \rangle = P\Big(U|\xi_{\mathcal{V}}\rangle,\ F(\mathcal{V})\Big) \geq 1 - \epsilon$, hence $\|\psi_{\mathcal{V}}\| \leq \sqrt{\epsilon}$.

By the definition of the operator T, $T|\zeta_{\mathcal{V}}\rangle = T_{F(\mathcal{V})}|\zeta_{\mathcal{V}}\rangle$, hence

$$U^{-1}TU\,|\xi_{\mathcal{V}}\rangle\ = T_{F(\mathcal{V})}|\xi_{\mathcal{V}}\rangle\ +\ U^{-1}(T-1)\,|\psi_{\mathcal{V}}\rangle$$

The norm of the last term does not exceed $2\sqrt{\epsilon}$. Summation over all $\mathcal{V} \in \Omega$ gives the desired result. $\square$

An interesting application of this theorem is a polynomial QFT algorithm for an arbitrary finite Abelian group G. W. l. o. g. we may take G to be a cyclic group $\mathbf{Z}_q$. (Transition to a direct product of cyclic groups is straightforward). Let $q \leq 2^n$, where n is a constant. We identify $\mathbf{Z}_q$ with the set $\{0, \ldots, q-1\} \subseteq \mathbf{B}^n$. Our purpose is to represent the QFT operator $V_q \in \mathbf{U}(\{0, \ldots, q-1\})$

$$V_q|a\rangle \;=\; |\psi_{q,a}\rangle \;=\; \frac{1}{\sqrt{q}} \sum_{b=0}^{q-1} \exp\left(2\pi i \frac{ab}{q}\right) |b\rangle \tag{22}$$

by an operation sequence in the basis $\mathcal{Q}$. We can also consider q as a control parameter and construct a representation for the operator $V : |q, \xi\rangle \mapsto |q\rangle \otimes V_q|\xi\rangle$.

The vectors $|\psi_{q,a}\rangle$ are eigenvectors of the cyclic permutation $|a\rangle \mapsto |(a+1) \bmod q\rangle$. The corresponding eigenvalues are $\lambda_{q,a} = \exp\bigl(-2\pi i(a/q)\bigr)$. By Theorem 1, we can measure the value of a. Theorem 2 allows us to perform this measurement reversibly, that is to represent the following partial operator on $\mathbf{C}(\mathbf{B}^n \times \mathbf{B}^n)$

$$Q_q\,|\psi_{q,a}, 0\rangle \;=\; |\psi_{q,a}, a\rangle \qquad\qquad (a = 0, \ldots, q-1) \tag{23}$$

The QFT operator V_q can be constructed from the operator Q_q and another operator T_q which creates the vector $|\psi_{q,a}\rangle$ for a given value of a

$$T_q\,|a, 0\rangle \;=\; |a,\, \psi_{q,a}\rangle \qquad\qquad (a = 0, \ldots, q-1) \tag{24}$$

This construction is quite similar to that used in the proof of Lemma 2. Let X and Y be two disjoint registers of size n. Then[10]

$$V_q[X] \otimes \omega[Y] \;=\; \bigl(Q_q[X,Y]\bigr)^{-1}\; T_q[Y,X]\;\; \tau_n[Y,X]\;\; \tau_n[X,Y] \tag{25}$$

Indeed, $|a,0\rangle \mapsto |a,a\rangle \mapsto |0,a\rangle \mapsto |\psi_{q,a},a\rangle \mapsto |\psi_{q,a},0\rangle$.

It is obvious that $T_q|a,0\rangle = U_q|a,\psi_{q,0}\rangle$, where

$$U_q\,|a,b\rangle \;=\; \exp\bigl(2\pi i(ab/q)\bigr)\,|a,b\rangle$$

The operator U_q can be easily constructed from $\Lambda\bigl(e^{2\pi i\, 2^s/q}\bigr)$, with $s = 0, \ldots, n-1$ (by Lemma 7). Thus the only remaining task is to create the vector $|\psi_{q,0}\rangle$. For this, we have to regard q as a variable. Our procedure is recursive. For simplicity, assume that $2^{n-1} < q < 2^n$. At the first step, the machine sets the first bit to the quantum state $(q_0/q)^{1/2}|0\rangle + (q_1/q)^{1/2}|1\rangle$, where $q_0 = 2^{n-1}$, $q_1 = q - q_0$. Then it looks at the value x of this bit and creates the vector $|\psi_{q_x,0}\rangle$ in the remaning $n-1$ bits. The result will be equal to $|\psi_{q,0}\rangle$.

Acknowledgements. This work was supported, in part, by the ISF grant M5R000. I am grateful to Sergei Tarasov for useful remarks.

[10] Recall that ω is the partial operator which maps the vector $|0\rangle$ to itself.

References

[1] P. Benioff, "Quantum mechanical Hamiltonian models of Turing machines", *J. Stat. Phys.* **29**, 515 (1982).

[2] A. Peres, "Reversible logic and quantum computers", *Phys. Rev.* **A 32**, 3266 (1985).

[3] R. P. Feynman, "Quantum mechanical computers", Optics News, February 1985, **11**, p. 11.

[4] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer", *Proc. R. Soc. Lond.* **A 400**, 97 (1985).

[5] D. Deutsch, "Quantum computational networks", *Proc. Roy. Soc. Lond.* **A 425**, 73 (1989).

[6] A. C.-C. Yao, "Quantum Circuit Complexity", *Proceedings of the 34th Annual Symposium on the Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1993), p. 352.

[7] P. W. Shor, "Algorithms for quantum computation: discrete log and factoring", *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1994), p. 124.

[8] E. Bernstein and U. Vazirani, "Quantum complexity theory", *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, (ACM Press, New York, 1993), pp. 11 – 20.

[9] D. Grigoriev, "Testing shift equivalence of polynomials using quantum machines" *(to appear)*

[10] D. Simon, "On the power of quantum computation", *Proceedings of the 35th Annual Symposium on the Foundations of Computer Science* (IEEE Computer Society Press, Los Alamitos, CA, 1994), p. 116.

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

[12] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring", *IBM Research Report RC19642* (1994)

[13] G. L. Miller, "Riemann's hypothesis and tests for primarity", *J. Comp. Sys. Sci*, **13**, 300–317 (1976)

[14] Yves Lecerf, "Machines de Turing reversibles. Recursive insolubilite en $n \epsilon N$ de l'equation $u = \theta^n$ ou θ est un "isomorphism de codes". *Comptes Rendus* **257**, 2597–2600 (1963)

[15] C. H. Bennett, "Logical reversibility of computation", *IBM Journal of Research and Development* **17**, 525 (1973).

[16] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. Smolin, and H. Weinfurter, "Elementary gates for quantum computation", quant-ph/9503016