4 Quantum Algorithms: Applicable Algebra and Quantum Physics

Thomas Beth and Martin Rötteler

4.1 Introduction

Classical computer science relies on the concept of Turing machines as a unifying model of universal computation. According to the modern Church–Turing Thesis, this concept is interpreted in the form that every physically reasonable model of computation can be *efficiently* simulated on a probabilistic Turing machine. Recently this understanding, which was taken for granted for a long time, has required a severe reorientation because of the emergence of new computers that do *not* rely on classical physics but, rather, use effects predicted by quantum mechanics.

It has been realized that, by using the principles of quantum mechanics, there are problems for which a putative quantum computer could outperform any classical computer. Quantum algorithms benefit from the application of the superposition principle to the internal states of the quantum computer, which are considered to be states in a (finite-dimensional) Hilbert space. As a result, these algorithms lead to a new theory of computation and might be of central importance to physics and computer science. Striking examples of quantum algorithms are Shor's factoring algorithm, Grover's search algorithm and algorithms for quantum error-correcting codes, all of which will be part of this contribution.

We shall introduce the complexity model of quantum gates, which are most familiar to researchers in the field of quantum computing, and shall give many examples of the usefulness and conciseness of this formalism. Quantum circuits provide a computational model equivalent to quantum Turing machines. This means that, very much like the situation in classical computing, there are several ways of describing computations by appropriate theoretical models.

Amongst such quantum circuits, quantum signal transforms form basic primitives in the treatment of controlled quantum systems. A surprising and important result, in view of the algorithms of Shor, is the fact that it is possible to compute a Fourier transform (of size 2^n) on a quantum computer by means of a quantum circuit which requires only $O(n^2)$ basic operations. This is a substantial speedup compared to the classical case, where the fast Fourier transform [171] yields an algorithm that requires $O(n2^n)$ arithmetic operations. Applications of such Fourier transforms to finite abelian groups arise in the algorithms of Simon and Shor. We shall present these algorithms and the underlying principle leading to their surprisingly fast solution on a quantum computer.

As already mentioned, one of the basic results is that in the complexity model of quantum circuits, the Fourier transform can be realized with an exponential speedup compared with the classical case. However, in the quantum regime the only way to extract information from a system is to make measurements and thereby project out nearly all aspects of the whole system. Thus, the art and science of designing quantum algorithms lies in the ability to obtain enough information from measurements, i.e. to choose the right bases from which relevant information can be read off. On the basis of the example of the Fourier observable, which represents the most important case of such a base change, we explain the underlying principle by means of the so-called hidden-subgroup algorithms and present an analysis of sampling in the Fourier basis with respect to the appropriate groups.

We then show how recent results in the theory of signal processing (for a classical computer) can be applied to obtain fast quantum algorithms for various discrete signal transforms, including Fourier transforms for nonabelian groups. Finally, we give a brief introduction to the theory of (quantum) error-correcting codes and their algorithmic implementation.

4.2 Architectures and Machine Models

The definition of an architecture and a machine model, on which the computations are considered to be carried out, is indispensable if one is to have a common computational model for which algorithms can be devised.

Each reasonable model of computation should give us the possibility of performing arbitrary operations, up to a desired accuracy, on the system by execution of elementary operations. By counting the elementary operations necessary to complete a given task, we arrive at complexity models. Finally, if different approaches defining universal computational models are possible, it is desirable to show the equivalence of these models, in the sense that they can simulate each other with a slowdown that is polynomial in the size of the input. In the case of quantum computing, we give two models for universal quantum computation, namely quantum networks in Sect. 4.2.1 and quantum Turing machines in Sect. 4.2.6. We shall put more emphasis on gates and networks, relying on the result that these two models are equivalent in the sense described.

One remark concerning the architecture is in order: we restrict ourselves to the case of operational spaces with a dimension that is a power of two, which are called *qubit architectures*. These systems incorporate the features necessary to do quantum computing, i.e. superposition of an exponentially

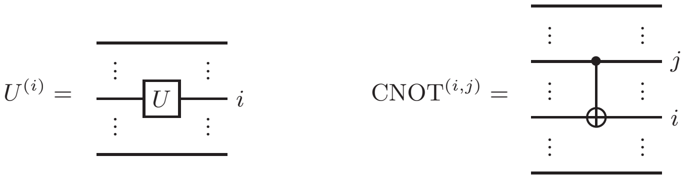

Fig. 4.1. Elementary quantum gates

growing number of states, interference between computational paths and entanglement between quantum registers.

It is possible to perform an embedding of an arbitrary finite-dimensional operational space into a qubit architecture (see Sect. 4.2.3); however, this reduction involves a suitable encoding of the states of the system into the basis states of the qubit architecture, and hence genuine properties of the system might be lost by this procedure.

4.2.1 Quantum Networks

The state of a quantum computer is given by a normalized vector in a Hilbert space $\mathcal{H}_{2^n}$ of dimension 2^n, which is endowed with a natural tensor structure $\mathcal{H}_{2^n} \cong \mathcal{C}^2 \otimes \ldots \otimes \mathcal{C}^2$ (n factors). The standard basis for this Hilbert space is the set $\{|x\rangle : x \in \mathbf{Z}_2^n\}$ of binary strings of length n. Restricting the computational space to Hilbert spaces of this particular form is motivated by the idea of a quantum register consisting of n quantum bits. A quantum bit, also called a *qubit*, is a state corresponding to one tensor component of $\mathcal{H}_{2^n}$ and has the form

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \ , \quad \alpha, \beta \in \mathcal{C} \ , \quad |\alpha|^2 + |\beta|^2 = 1 \ .$$

The possible operations that this computer can perform are the elements of the unitary group $\mathcal{U}(2^n)$. To study the complexity of performing unitary operations on n-qubit quantum systems, we introduce the following two types of computational primitives: *local unitary operations* on a qubit i are matrices of the form $U^{(i)} = \mathbf{1}_{2^{i-1}} \otimes U \otimes \mathbf{1}_{2^{n-i}}$, where U is an element of the unitary group $\mathcal{U}(2)$ of 2×2 matrices and $\mathbf{1}_N$ denotes the identity matrix of size N. Furthermore, we need operations which affect two qubits at a time, the most prominent of which is a so-called *controlled NOT gate* (also called a measurement gate) between the qubits j (control) and i (target), denoted by $\mathrm{CNOT}^{(i,j)}$. On the basis vectors $|x_n, \ldots, x_1\rangle$ of $\mathcal{H}_{2^n}$, the operation $\mathrm{CNOT}^{(i,j)}$ is defined by

$$|x_n, \ldots, x_{i+1}, x_i, x_{i-1}, \ldots, x_1\rangle \mapsto |x_n, \ldots, x_{i+1}, x_i \oplus x_j, x_{i-1}, \ldots, x_1\rangle \ ,$$

where the addition $\oplus$ is performed in $\mathbf{Z}_2$.

In graphical notation, using quantum wires, these transformations are written as shown in Fig. 4.1. Lines correspond to qubits, unaffected qubits are omitted and a dot • sitting on a wire denotes a control bit. Note that we draw the qubits according to their significance, starting with the most significant qubit on top. Quantum circuits are always read from left to right.

The two types of gates shown in Fig. 4.1 suffice to generate all unitary transformations, i.e. they form a universal set of gates. This is the content of the following theorem [172].

Theorem 4.1. $\mathcal{G}_1 := \{U^{(i)}, \mathrm{CNOT}^{(i,j)} \mid U \in \mathcal{U}(2),\ i,j \in \{1,\ldots,n\},\ i \neq j\}$ *is a generating set for the unitary group* $\mathcal{U}(2^n)$*.*

This means that for each $U \in \mathcal{U}(2^n)$ there is a word $w_1 w_2 \ldots w_k$ (where $w_i \in \mathcal{G}_1$ for $i = 1, \ldots, k$ is an elementary gate) such that U factorizes as $U = w_1 w_2 \ldots w_k$. On the basis of theorem 4.1, we now define a complexity measure for unitary operations on qubit architectures.

Definition 4.1. *Let* $U \in \mathcal{U}(2^n)$ *be a given unitary transformation. Then* $\kappa(U)$ *is defined as the minimal number* k *of operations in* $\mathcal{G}_1$ *necessary to write* $U = w_1 w_2 \ldots w_k$ *as a sequence of elementary gates.*

For the complexity measure κ, the following holds: $\kappa(A \otimes B) \leq \kappa(A) + \kappa(B)$ for all $A \in \mathcal{U}(2^{n_1})$ and $B \in \mathcal{U}(2^{n_2})$, because tensor products are free of cost in a computational model based on quantum mechanical principles. Also, by concatenation of operations, we obtain $\kappa(A \cdot B) \leq \kappa(A) + \kappa(B)$ for $A, B \in \mathcal{U}(2^n)$. Note that whereas in the usual linear complexity measure L_c [173, 174] permutation matrices are free (i.e. $L_c(\pi) = 0$, for all $\pi \in S_n$), we have to take them into account when using the complexity measure κ. Instead of the universal set of gates $\mathcal{G}_1$ we can, alternatively, use the set $\mathcal{G}_2 := \{U^{(i,j)} : U \in \mathcal{U}(4),\ i,j \in \{1,\ldots,n\},\ i \neq j\}$ of all two-bit gates, changing the value of κ by only a constant.

Whereas the complexity measure κ is used in cases where we want to implement a given unitary operation U *exactly* in terms of the generating sets $\mathcal{G}_1$ and $\mathcal{G}_2$, it is also expedient to consider unitary *approximations* by quantum networks. By this we mean a sequence of operations $w_1, \ldots, w_n$ which approximates U up to a given ϵ, i.e. such that $\|U - \prod_{i=1}^{n} w_i\| < \epsilon$, where $\|\cdot\|$ denotes the spectral norm.[1] We denote the corresponding complexity measure by κ_ϵ.

Remark 4.1. The following facts concerning approximation by elementary gates are known:

- There are two-bit gates which are universal [175, 176], i.e. there exists a unitary transformation $A \in \mathcal{U}(4)$ with respect to which it is possible to approximate any given U up to $\epsilon > 0$ by a sequence of applications of A

[1] Recall that the spectral norm of a matrix $A \in \mathcal{C}^{n\times n}$ is given by $\max_{\lambda \in \mathrm{Spec}(A)} |\lambda|$.

to two tensor components of $\mathcal{H}_{2^n}$ only: $\|U - \prod_{i=1}^{k} A^{(m_i,n_i)}\| < \epsilon$. Even though the much stronger statement of universality of a *generic* two-bit gate is known to be true, it is hard to prove the universality for a *given* two-bit gate [176, 177].

- Small generating sets are known; for instance, we can choose

$$\mathcal{G}_3 := \left\{ \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & (1+\mathrm{i})\sqrt{2} \end{pmatrix}, \mathrm{CNOT}^{(k,l)},\ k \neq l \right\},$$

where the first two operations generate a dense subgroup in $\mathcal{U}(2)$. The Hadamard transformation, which is part of this generating set, is denoted by

$$H_2 := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and is an example of a Fourier transformation on the abelian group $\mathbf{Z}_2$ (see Sect. 4.4).

- Knill has obtained a general upper bound $O(n4^n)$ for the approximation of unitary matrices using a counting argument [178, 179].
- We cite the following approximation result from Sect. 4.2 of [180]: Fix a number n of qubits and suppose that $\overline{\langle X_1, \ldots, X_r \rangle} = \mathcal{SU}(2^n)$, i.e. $X_1, \ldots, X_r$ generate a dense subgroup in the special unitary group $\mathcal{SU}(2^n)$. Then it is possible to approximate a given matrix $U \in \mathcal{SU}(2^n)$ with given accuracy $\epsilon > 0$ by a product of length $O\{\mathrm{poly}[\log(1/\epsilon)]\}$, where the factors belong to the set $\{X_1, \ldots, X_r, X_1^{-1}, \ldots, X_r^{-1}\}$. Furthermore, this approximation is constructive and efficient, since there is an algorithm with running time $O\{\mathrm{poly}[\log(1/\epsilon)]\}$ which computes the approximating product. However, we remind the reader that this holds only for a fixed value of n; the constant hidden in the O-calculus grows exponentially with n (see Theorem 4.8 of [180]).

From now on, we put the main emphasis on the model for realizing unitary transformations exactly and on the associated complexity measure κ. In general, only exponential upper bounds for the minimal length occuring in factorizations are known. However, there are many interesting classes of unitary matrices in $\mathcal{U}(2^n)$ that lead to only a polylogarithmic word length, which means that the length of a minimal factorization grows asymptotically like $O[p(n)]$, where p is a polynomial.

In the following we give some examples of transformations, their factorization into elementary gates and their graphical representation in terms of quantum gate arrays. The operations considered in these examples admit short factorizations and will be useful in the subsequent parts of this chapter.

Example 4.1 (Permutation of Qubits). The symmetric group S_n is embedded in $\mathcal{U}(2^n)$ by the natural operation of S_n on the tensor components (qubits).

Let $\tau \in S_n$ and let Π_τ be the corresponding permutation matrix on 2^n points. Then $\kappa(\Pi_\tau) = O(n)$: to prove this, we first note that each element $\sigma \in S_n$ can be written as a product $\sigma = \tau_1 \cdot \tau_2$ of two involutions τ_1 and $\tau_2 \in S_n$, i.e. $\tau_1^2 = \tau_2^2 = id$. To see that it is always possible to find a suitable τ_1 and τ_2, we can, when considering the decomposition of σ into disjoint cycles, restrict ourselves to the case of an n-cycle. Now the decomposition follows immediately from the fact that there is a dihedral group of size $2n$ containing σ as the canonical n-cycle and that this rotation is the product of two reflections.

The unitary transformation corresponding to Π_τ, where $\tau \in S_n$ is an involution, can be realized by swappings of quantum wires, which, in turn, can be performed efficiently and in parallel. To swap two quantum wires we can use the well-known identity $\Pi_{(1,2)} = \mathrm{CNOT}^{(1,2)} \cdot \mathrm{CNOT}^{(2,1)} \cdot \mathrm{CNOT}^{(1,2)}$, yielding a circuit of depth three. Writing an arbitrary permutation Π_τ of the qubits as a product of two involutions, we therefore obtain a realization by a circuit of depth six at most (see also [181]).

As an example, the permutation $(1,3,2)$ of the qubits (which corresponds to the permutation $(1,4,2)(3,5,6)$ on the register) is factored as $(1,3,2) = (1,2)(2,3)$ (see Fig. 4.2).

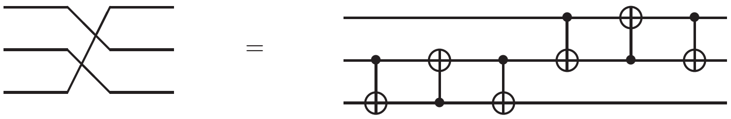

Fig. 4.2. Factorization $(1,3,2) = (1,2)(2,3)$

Example 4.2 (Controlled Operations). Following [172], we introduce a special class of quantum gates with multiple control qubits, yielding a natural generalization of the controlled NOT gate. This class of gates is given by the transformations $\Lambda_k(U)$, where U is a unitary transformation in $\mathcal{U}(2^l)$. The gate $\Lambda_k(U)$ is a transformation acting on $k+l$ qubits, where the k most significant bits serve as control bits and the l least significant bits are target bits: the operation U is applied to the l target bits if and only if all k control bits are equal to 1. Denoting by $M := 2^l(2^k-1)$ the number of basis vectors on which $\Lambda_k(U)$ acts trivially, the corresponding unitary matrix is given by $\mathbf{1}_M \oplus U$, where we have used $\oplus$ to denote a direct sum of matrices.

To provide further examples of the graphical notation for quantum circuits, we give in Fig. 4.3 a $\Lambda_1(U)$ gate for $U \in \mathcal{U}(2^n)$ with a normal control qubit, a gate $\overline{\Lambda}_1(U)$ with an inverted control qubit, and the matrices represented. Lemmas 7.2 and 7.5 of [172] show that for $U \in \mathcal{U}(2)$, the gate $\Lambda_k(U)$ can be realized with gate complexity $O(n)$, for $k < n-1$. If there are auxiliary qubits (so-called ancillae) available, a gate $\Lambda_{n-1}(U)$ can also be computed using $O(n)$ operations from $\mathcal{G}_1$; otherwise, we have $\kappa[\Lambda_{n-1}(U)] = O(n^2)$.

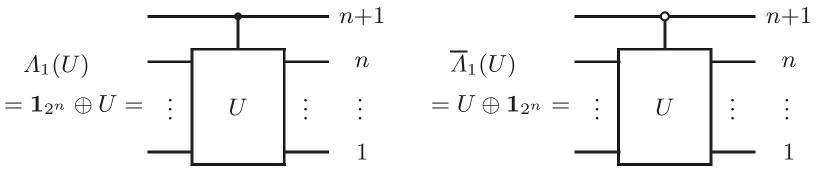
$\Lambda_1(U)$
$= \mathbf{1}_{2^n} \oplus U =$
U
$n{+}1$
n
1
$\overline{\Lambda}_1(U)$
$= U \oplus \mathbf{1}_{2^n} =$

Fig. 4.3. Controlled gates with (*left*) normal and (*right*) inverted control bit. Here $\oplus$ is used to denote the block-direct sum of matrices

We remark that, if $U \in \mathcal{U}(2^n)$ can be realized in p elementary operations then, $\Lambda_1(U) \in \mathcal{U}(2^{n+1})$ can be realized in $c \times p$ basic operations, where $c \in \mathbb{N}$ is a constant that does not depend on U. To see this, we first assume, without loss of generality, that U is decomposed into elementary gates. Therefore, we have to show that a doubly controlled NOT (also called a Toffoli gate, see Sect. 4.2.3) and a singly controlled $U \in \mathcal{U}(2)$ gate can be realized with a constant increase of length. It is possible to obtain the bound $c \leq 17$ according to the following decompositions [172]: for each unitary transformation $U \in \mathcal{U}(2)$ we can write $\Lambda_1(U) = A^{(1)} \cdot \mathrm{CNOT}^{(1,2)} \cdot B^{(1)} \cdot \mathrm{CNOT}^{(1,2)} \cdot C^{(1)}$ with suitably chosen $A, B, C \in \mathcal{U}(2)$, i.e. we need at most five elementary gates for the realization of $\Lambda_1(U)$. To bound the number of operations necessary to realize $\tau := \Lambda_2(\sigma_x)$ with respect to the set $\mathcal{G}_1$, we choose a square root R of σ_x, i.e. $R^2 = \sigma_x$, and use the identity

$$\tau = [\mathbf{1}_2 \otimes \Lambda_1(R)] \cdot \mathrm{CNOT}^{(2,3)} \cdot [\mathbf{1}_2 \otimes \Lambda_1(R^\dagger)] \cdot \mathrm{CNOT}^{(2,3)} \cdot \Lambda_1(\mathbf{1}_2 \otimes R).$$

This shows that we need at most $5 + 1 + 5 + 1 + 5 = 17$ elementary gates to realize τ.

Example 4.3 (Cyclic Shift). Let $P_n \in S_{2^n}$ be the cyclic shift acting on the states of the quantum register as $x \mapsto x + 1 \bmod 2^n$. The corresponding permutation matrix is the 2^n-cycle $(0, 1, \ldots, 2^n{-}1)$. The unitary matrix P_n can be realized in a polylogarithmic number of operations; see Fig. 4.4 for a realization using Boolean gates only.

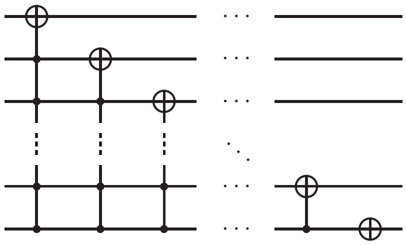

Fig. 4.4. Realizing a cyclic shift on a quantum register

Other, non-Boolean factorizations are also possible: using a basic fact about group circulants (see also Sect. 4.5.1) and anticipating the fact that the discrete Fourier transform DFT_{2^n} can be performed in $O(n^2)$ operations

(which is shown in Sect. 4.4.1), we can use the identity

$$\begin{aligned}\mathrm{DFT}_{2^n}^{-1} \cdot P_n \cdot \mathrm{DFT}_{2^n} &= \mathrm{diag}(\omega_{2^n}^i : i = 0, \ldots, 2^n - 1)\\ &= \mathrm{diag}(1, \omega_{2^n}^{2^{n-1}}) \otimes \cdots \otimes \mathrm{diag}(1, \omega_{2^n})\end{aligned}$$

to obtain $\kappa(P_n) = O(n^2)$.

4.2.2 Boolean Functions and the Ring Normal Form

Boolean functions are important primitives used throughout classical informatics. Denoting the finite field with q elements[2] by $GF(q)$, we obtain the Boolean numbers as the special case $q = 2$. A multivariate Boolean function $f : GF(2)^n \to GF(2)$ can be represented in various ways. Besides the truth table, which is a common but uneconomic way to represent f as the sequence of its values $f(0 \ldots 0), \ldots, f(1 \ldots 1)$ for all binary strings of length n, prominent examples of normal forms are the conjunctive and disjunctive normal forms [183] which originate from predicate logic and are used in transistor circuitry.

For quantum computational purposes, another way of representing f offers itself, namely the ring normal form (RNF), defined as the (unique) expansion of f as a polynomial in the ring R_n of Boolean functions of n variables. This ring is defined by $R_n := GF(2)[X_1, \ldots, X_n]/(X_1^2 - X_1, \ldots, X_n^2 - X_n)$ [184]. Multiplication and addition in R_n are the usual multiplication and addition of polynomials modulo the relations given by the ideal $(X_1^2 - X_1, \ldots, X_n^2 - X_n)$, and addition is usually denoted as "$\oplus$". Therefore f is represented as

$$f(X_1, \ldots, X_n) := \bigoplus_{u=(u_1,\ldots,u_n)\in\{0,1\}^n} a_u \prod_{i=1}^n X_i^{u_i}, \tag{4.1}$$

with coefficients $a_u \in GF(2)$.

Example 4.4. The logical complement is given by $\mathrm{NOT}(X) = 1 \oplus X$. The RNF of the AND function on n variables $X_1, \ldots, X_n$ is $\mathrm{AND}(X_1, \ldots, X_n) = \prod_{i=1}^n X_i$. The RNF of the PARITY function of n variables is given by $\mathrm{PARITY}(X_1, \ldots, X_n) = \bigoplus_{i=1}^n X_i$. Finally, the RNF of the OR function on n variables is given by

$$\mathrm{OR}(X_1, \ldots, X_n) = 1 \oplus \prod_{i=1}^n (1 \oplus X_i) = \bigoplus_m m(X_1, \ldots, X_n)\ ,$$

[2] Necessarily, we have $q = p^n$, where p is a prime and $n \geq 1$ [182]. Finite fields are also called *Galois fields* after Évariste Galois (1811–1832).

where the last sum runs over all nonconstant multilinear monomials m in the ring R_n.

We can implement a Boolean function given in the RNF shown in (4.1) with the use of the so-called Toffoli gate τ. The action of τ on the basis states of the Hilbert space $\mathcal{H}_8$ is given by $\tau : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus x \cdot y\rangle$ [185].

Horner's rule for multivariate polynomials yields a method for implementing the function f given in (4.1). To achieve this, we write $f(X_1, \ldots, X_n) = f_1(X_1, \ldots, X_{n-1}) \cdot X_n \oplus f_2(X_1, \ldots, X_{n-1})$ and observe that this function can be computed using one Toffoli gate, assuming that f_1 and f_2 have already been computed. Therefore, we obtain a recursive factorization for f, which in general will make use of auxiliary qubits.

4.2.3 Embedded Transforms

This section deals with the issue of embedding a given transform A into a unitary matrix of larger size. We start by considering the problem of realizing a given matrix A as a submatrix of a unitary matrix of larger size. The following theorem (see also [186]) shows that the only condition A has to fulfill in order to allow an embedding involving one additional qubit is to be of bounded norm, i.e. $\|A\| \leq 1$, with respect to the spectral norm.

Theorem 4.2. *Let $A \in \mathcal{C}^{n \times n}$ be a given matrix of norm $\|A\| \leq 1$. Then*

$$U_A := \begin{pmatrix} A & (\mathbf{1}_n - AA^\dagger)^{1/2} \\ (\mathbf{1}_n - A^\dagger A)^{1/2} & -A^\dagger \end{pmatrix} \tag{4.2}$$

yields a unitary matrix $U_A \in \mathcal{U}(2n)$ which contains A as the $n \times n$ submatrix in the upper left corner.

Proof. Observe that the $n \times 2n$ matrix $U_1 := (A, (\mathbf{1}_n - A^\dagger A)^{1/2})^t$ has the property $U_1^\dagger \cdot U_1 = \mathbf{1}_n$. Analogously, for the matrix $U_2 := [A, (\mathbf{1}_n - AA^\dagger)^{1/2}]$, the identity $U_2 \cdot U_2^\dagger = \mathbf{1}_n$ holds. An easy computation shows that (4.2) is indeed unitary. $\square$

Since each matrix in $\mathcal{C}^{n \times n}$ can be renormalized by multiplication with a suitable scalar to fulfill the requirement of a bounded norm, we can realize all operations up to a scalar prefactor by unitary embeddings. The embedding (4.2) is by no means unique. However, it is possible to parametrize *all* embeddings by $(\mathbf{1}_n \oplus V_1) \cdot U_A \cdot (\mathbf{1}_n \oplus V_2)$, where $V_1, V_2 \in \mathcal{U}(n)$ are arbitrary unitary transforms.

We are naturally led to a different kind of embedding if the given transformation is unitary and we want to realize it on a qubit architecture, i.e. if we restrict ourselves to matrices whose size is a power of 2. Then, a given unitary matrix $U \in \mathcal{U}(N)$ can be embedded into a unitary matrix in $\mathcal{U}(2^n)$ by choosing $n = \lceil \log N \rceil$ and padding U with an identity matrix $\mathbf{1}_{2^n - N}$ of size

$2^n - N$. This is noncanonical since we have degrees of freedom in the choice of the subspace on which this newly formed matrix acts as the identity.

In general, it is a difficult problem to find the optimal embedding for a given transform. An example in which it is not natural to go to the next power of 2 is given by $U = U_1 \otimes U_2 \in \mathcal{U}(15)$, where $U_1 \in \mathcal{U}(3)$ and $U_2 \in \mathcal{U}(5)$. We then have the possibilities $U \oplus \mathbf{1}_1 \in \mathcal{U}(2^4)$ and the embedding $(U_1 \oplus \mathbf{1}_1) \otimes (U_2 \oplus \mathbf{1}_3) \in \mathcal{U}(2^5)$, which respects the tensor decomposition of U.

A third type of embedding occurs in the context of quantum and reversible computing, where a general method is required to make a given map $f : X \to Y$ bijective (here X and Y are finite sets). If we consider the map $\tilde{f} : X \times Y \to X \times Y$ which maps $(x, y_0) \mapsto (x, f(x))$, where y_0 is a fixed element in the codomain Y of f, then this map is obviously injective when restricted to the fibre $X \times \{y_0\}$. Observe now that it is always possible to extend $\tilde{f}|_{X\times\{y_0\}}$ to a unitary operation on the Hilbert space $\mathcal{H}_{X,Y}$ spanned by the basis consisting of $\{|x\rangle|y\rangle : x \in X, y \in Y\}$. Hence, it is always possible to construct a unitary operation $V_f : \mathcal{H}_{X,Y} \to \mathcal{H}_{X,Y}$ which has the property

$$|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle \ , \quad \text{for all } \ x \in X \ , \tag{4.3}$$

i.e. V_f implements the graph $\Gamma_f = \{(x, f(x)) : x \in X\}$ of f in the Hilbert space $\mathcal{H}_{X,Y}$. Here, we have identified the special element y_0 with the basis vector $|0\rangle \in \mathcal{H}_Y$.

Example 4.5. Let $f : GF(2)^2 \to GF(2)$ be the AND function, i.e. let $f = x \cdot y$ be the RNF of f. Note that $\tilde{f}$ can be chosen to be the function $(x, y, z) \mapsto (x, y, z \oplus f(x, y))$ since the codomain is endowed with a group structure. Overall, we obtain the function table of $\tilde{f}$ given in Fig. 4.5. The variables with a prime correspond to the values after the transformation has been performed.

x	y	z	x'	y'	z'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1

x	y	z	x'	y'	z'
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

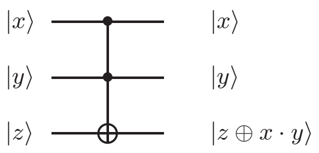

Fig. 4.5. Truth table for the Toffoli gate and the corresponding quantum circuit

We recognize $\tilde{f}$ as the unitary operation $\tau : |x\rangle|y\rangle|z\rangle \mapsto |x\rangle|y\rangle|z \oplus x \cdot y\rangle$ on the Hilbert space $\mathcal{H}_8$, which is the Toffoli gate [185].

The method described in Example 4.5 is quite general, as the following theorem shows (for a proof see [187]).

Theorem 4.3. *Suppose $f : \{0,1\}^n \to \{0,1\}^m$ is a Boolean function which can be computed using c operations from the universal set* {AND, NOT} *of classical gates. Then $\tilde{f} : \{0,1\}^{n+m} \to \{0,1\}^{n+m}$, defined by $(x, y) \mapsto (x, y \oplus f(x))$, is a reversible Boolean function which can be computed by a circuit of length $2c + m$ built up from the set $\{\mathrm{CNOT}, \tau\}$ of reversible gates.*

Even though the construction described in Theorem 4.3 works for arbitrary $f : \{0,1\}^n \to \{0,1\}^m$, in general only $r_f = \lceil \log \max_{y \in \{0,1\}^m} |f^{-1}(y)| \rceil$ additional bits are necessary to define a reversible Boolean function $f_{\mathrm{rev}} : \{0,1\}^{n+r_f} \to \{0,1\}^{n+r_f}$ with the property $f_{\mathrm{rev}}|_{\{0,1\}^n} = f$. The reason is that by using the additional r_f bits, the preimages of f can be separated via a suitable binary encoding. However, the complexity of a Boolean circuit of a realization of f_{rev} constructed in such a way is such that the circuit cannot be controlled as easily as for the function defined in Theorem 4.3.

4.2.4 Permutations

We have already mentioned in Sect. 4.2.1 that on a quantum computer permutations of the basis states have to be taken into account when considering the complexity: in general, for the cost $\kappa(\pi)$, where π is a permutation matrix in $\mathcal{U}(2^n)$ and κ is the complexity measure introduced in Sect. 4.2.1, nothing better is known than an exponential upper bound of $O(n4^n)$.

Nevertheless, there are quite a few classes of permutations admitting a better, even polylogarithmic word length, as the examples of permutations of quantum wires and of the cyclic shift $P_n : x \mapsto x + 1 \bmod 2^n$ on a quantum register have shown (see Examples 4.1 and 4.3).

In what follows we consider a further class of permutations of a quantum register that admits efficient realizations, which operate by linear transformations on the *names* of the kets. Recall that the basis states can be identified with the binary words of length n and hence, with the elements of $GF(2)^n$, the n-dimensional vector space over the finite field $GF(2)$ of two elements. Denoting the group of invertible linear transformations of $GF(2)^n$ by $\mathrm{GL}(n, GF(2))$, we see that each transform $A \in \mathrm{GL}(n, GF(2))$ corresponds to a permutation of the binary words of length n and, hence, to a permutation matrix Φ_A of size $2^n \times 2^n$.

It turns out that these permutations are efficiently realizable on a quantum computer (see Sect. 4 of [188]). First we need the following lemma.

Lemma 4.1. *Let K be a field and let $A \in \mathrm{GL}(n, K)$ be an invertible matrix with entries in K. Then there exist a permutation matrix P, a lower triangular matrix L and an upper triangular matrix U such that $A = P \cdot L \cdot U$.*

In numerical mathematics this decomposition is also known as the "LU decomposition" (see, e.g., Sect. 3.2. of [189]). The statement is a consequence of Gauss's algorithm. We are now ready to prove the following theorem.

Theorem 4.4. *Given $A \in \mathrm{GL}(n, GF(2))$, $\varPhi_A$ can be realized in $O(n^2)$ elementary operations on a quantum computer.*

Proof. First, decompose A according to Lemma 4.1 into $A = P \cdot L \cdot U$ and observe that the permutation matrix P is a permutation of the quantum wires, and hence $\kappa(P) = O(n)$ (see Example 4.1). The matrices L and U can be realized using CNOT gates only. Without loss of generality, we consider the factorization of L: proceeding along the diagonals of these matrices, we find all diagonal entries to be 1 (otherwise the matrices would not be invertible). Therefore $\varPhi_L$ maps the basis vector $|e_i\rangle$, where $e_i = (0 \dots 1 \dots 0)$ is the ith basis vector in the standard basis of $GF(2)^n$, to the sum $\sum_{j \geq i} \alpha_j e_j$, where the vector $(\alpha_i)_{i=1,\dots,n}$ is the ith column of A. Application of the sequence $\prod_{j \geq i} \mathrm{CNOT}^{(i,j)}$, where the product runs over all $j \neq 0$, has the same effect on the basis vector e_i.

Proceeding column by column in L yields a factorization into $O(n^2)$ elementary gates. Combining the factorizations for P, L and U, we obtain $\kappa(A) = O(n^2)$. $\square$

As an example, we take a look at the matrix

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \in \mathrm{GL}(2, GF(2)) \ .$$

To see what the corresponding $\varPhi_A$ looks like, we compute the effect of A on the basis vectors:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mapsto \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

i.e. $\varPhi_A = \mathrm{CNOT}^{(2,1)}$ in accordance with Theorem 4.4, since A is already lower triangular.

As further examples of permutations arising as unitary transforms on a quantum computer, we mention gates for modular arithmetic [190, 191]. More specifically, we consider the following operation, acting on kets which have been endowed with the group structure of $\mathbf{Z}_N := \mathbf{Z}/N\mathbf{Z}$, i.e. $\{|x\rangle : x \in \mathbf{Z}_N\}$ is a basis of this operational space $\mathcal{H}$. Then

$$\varUpsilon_a : |x\rangle|0\rangle \mapsto |x\rangle|a \cdot x \bmod N\rangle \ ,$$

where $a \in \mathbf{Z}_N^\times$ is an element of the multiplicative group of units in $\mathbf{Z}_N$; this can be extended to a permutation of the whole space $\mathcal{H} \otimes \mathcal{H}$ using the methods of Sect. 4.2.3. Using a number of ancilla qubits which is polynomial in $\log[\dim(\mathcal{H})]$, it is possible to realize $\varUpsilon_a$, as well as other basic primitives known from classical circuit design [183], such as

- adders modulo N: $|x\rangle|y\rangle \mapsto |x\rangle|x + y \bmod N\rangle$
- modular exponentiation: $|x\rangle|0\rangle \mapsto |x\rangle|a^x \bmod N\rangle$,

in polylogarithmic time on a quantum computer [190, 191].

4.2.5 Preparing Quantum States

If we are interested in preparing particular quantum states by means of an effective procedure, in most cases it is straightforward to write down a quantum circuit which yields the desired state when applied to the ground state $|0\rangle$. For instance, by means of the quantum circuit given in Fig. 4.6, a Schrödinger cat state $|\Psi_n\rangle$ on n qubits can be prepared using $n+1$ elementary gates. These are the states

$$|\Psi_n\rangle := \frac{1}{\sqrt{2}} \underbrace{|0\dots0\rangle}_{n \text{ zeros}} + \frac{1}{\sqrt{2}} \underbrace{|1\dots1\rangle}_{n \text{ ones}} ,$$

and we remind the reader that $|\Psi_2\rangle$ is locally equivalent to a so-called EPR state [24] and $|\Psi_3\rangle$ is a so-called GHZ state [37].

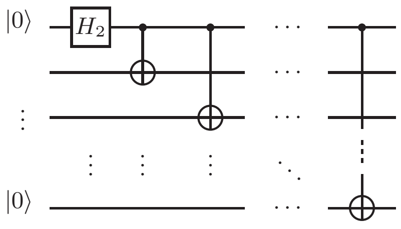

Fig. 4.6. Quantum circuit that prepares a cat state

We remark that there is an algorithm to prepare an arbitrary quantum state $|\varphi\rangle$ starting from the ground state $|0\rangle$, i.e. to construct a quantum circuit U_φ yielding $U_\varphi|0\rangle = |\varphi\rangle$.

Algorithm 1 *Let $|\varphi\rangle = \sum_{x \in \mathbf{Z}_2^n} \alpha_x |x\rangle$ be a quantum state which we would like to prepare. Do the following in a recursive way. Write*

$$|\varphi\rangle = a|0\rangle|\varphi_0\rangle + b|1\rangle|\varphi_1\rangle ,$$

where a and b are complex numbers fulfilling $|a|^2 + |b|^2 = 1$. The states

$$|\varphi_0\rangle := \sum_{x \in \mathbf{Z}_2^{n-1}} \alpha_x^{(0)} |x\rangle \qquad \textit{and} \qquad |\varphi_1\rangle := \sum_{x \in \mathbf{Z}_2^{n-1}} \alpha_x^{(1)} |x\rangle$$

appearing on the right-hand side of the above equation can be prepared by induction hypotheses using the circuits U_1 and U_2. Let A be the local transform

$$A := \begin{pmatrix} a & -\overline{b} \\ b & \overline{a} \end{pmatrix} \otimes \mathbf{1}_{2^{n-1}} .$$

Then $|\varphi\rangle$ can be prepared by application of the circuit $A \cdot \Lambda_1(U_1) \cdot \overline{\Lambda_1}(U_2)$ to the ground state $|0\rangle$.

In general, the quantum circuit U_φ for preparing a state $|\varphi\rangle \in \mathcal{H}_{2^n}$ generated by this algorithm has a complexity $\kappa(U_\varphi) = O(2^n)$, which is linear in the dimension of the Hilbert space but exponential in the number of qubits. However, as the example of cat states previously mentioned shows, there are states which admit much more efficient preparation sequences. In such a set of states, we also find the so-called symmetric states [192]

$$\frac{1}{\sqrt{n+1}}(|00\ldots0\rangle + |10\ldots0\rangle + |01\ldots0\rangle + \ldots + |00\ldots1\rangle) ,$$

i.e. the union of the orbits of $|00\ldots0\rangle$ and $|10\ldots0\rangle$ under the cyclic group acting on the qubits. As shown in Sect. 4 of [192], these states can be prepared using $O(n)$ operations and a quadratic overhead of ancilla qubits.
Finally, we give circuits for preparation of the states $|\psi_\nu\rangle := (1/\sqrt{\nu})\sum_{i=1}^{\nu}|i\rangle$ for $\nu = 1,\ldots,2^n$, which represent equal amplitudes over the first ν basis states of $\mathcal{H}_{2^n}$. The states $|\psi_\nu\rangle$ can be efficiently prepared from the ground state $|0\rangle$ by the following procedure (using the principle of binary search [193]), which is described in Sect. 4 of [187].

Since $|\psi_{2^n}\rangle$ can easily be prepared by application of the Hadamard transformation $H_2^{\otimes n}$, we can assume $\nu < 2^n$ without loss of generality. We now choose $k \in \mathbb{N}$ such that $2^k \leq \nu < 2^{k+1}$ and apply the transformation

$$U := \frac{1}{\sqrt{\nu}}\begin{pmatrix} \sqrt{2^k} & -\sqrt{\nu-2^k} \\ \sqrt{\nu-2^k} & \sqrt{2^k} \end{pmatrix}$$

to the first bit of the ground state $|0\rangle$. Next we achieve equal superposition on the first 2^k basis states $|0\ldots0\rangle,\ldots,|0\ldots01\ldots1\rangle$ by application of an $(n-k)$-fold controlled $\overline{\Lambda}_1(H_2^{\otimes k})$ operation, which can be implemented using $O(n^2)$ operations. Finally, we apply the preparation circuit for the state $|\psi_{\nu-2^k}\rangle$ (which has been constructed by induction), conditioned on the $(k+1)$th bit. Overall, we obtain a complexity for the preparation of $|\psi_\nu\rangle$ of $O(n^3)$ operations.

4.2.6 Quantum Turing Machines

Quantum circuits provide a natural framework to specify unitary transformations on finite-dimensional Hilbert spaces and give rise to complexity models when factorizations into elementary gates, e.g. with respect to the universal sets $\mathcal{G}_1$ or $\mathcal{G}_2$, are taken into account.

Besides the formalism of quantum networks, there are other ways of describing computations performed by quantum mechanical systems. In the following we briefly review the model of quantum Turing machines (QTMs) defined by Deutsch [7]. We remind the reader that Turing machines [194] provide a unified model for *classical* deterministic and probabilistic computation (see, e.g., [195]). The importance of Turing machines as a unifying

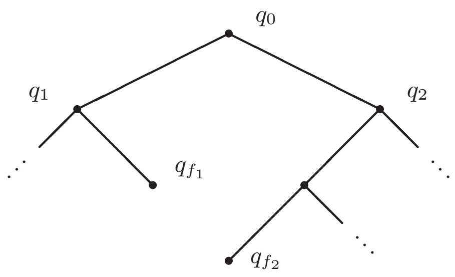

Fig. 4.7. Configurations of a Turing machine

concept for classical computing manifests itself in the Church–Turing thesis [196, 197], which, in its modern form, claims that every physically reasonable model of computation can be efficiently simulated on a probabilistic Turing machine.

Definition 4.2. *A* deterministic *Turing machine T is defined by the data (Q, Σ, q_0, F, t), where Q is a finite set of states, Σ a set of symbols, $q_0 \in Q$ a distinguished initial state, $F \subseteq Q$ the set of final states, and $t : Q \times \Sigma \longrightarrow Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$ the transition function. The admissible actions of T are movements of a read–write head, which in one computational step can move to the left, stay where it is or move to the right (we have denoted these actions by $\{\leftarrow, \downarrow, \rightarrow\}$). Along with the Turing machine T comes a infinite tape of cells (the cells are in bijection with $\mathbf{Z}$) which can take symbols from Σ.*

A *configuration* of a Turing machine T is therefore given by a triplet $(v, p, q) \in (\Sigma^{\mathbf{Z}}, \mathbf{Z}, Q)$, consisting of the state v of the tape, the position p of the head and the internal state q. We obtain a tree of configurations by considering two configurations c_1 and c_2 to be adjacent if and only if c_2 is obtained from c_1 by an elementary move, i.e. scanning a symbol from the tape, changing the internal state, writing back a symbol to the tape and moving the head. The initial state is the root of this tree, whereas the final states constitute its leaves (see Fig. 4.7).

A *probabilistic* Turing machine differs from a deterministic one only in the nature of the transition function t, which is then a mapping

$$t : Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\} \longrightarrow [0, 1] ,$$

which assigns probabilities from the real interval $[0, 1]$ to the possible actions of T. A normalization condition, which guarantees the well-formedness of a probabilistic Turing machine, is that for all configurations the sum of the probabilities of all successors is 1. Therefore, the admissible state transitions of a probabilistic Turing machine T can be described by a stochastic matrix $S_T \in [0, 1]^{\mathbf{Z} \times \mathbf{Z}}$, where stochasticity means that the rows of S_T add up to 1 and the successor c_{succ} of c is obtained by $c_{\text{succ}} = S_T \cdot c$. Note that a deterministic Turing machine is a special case of a probabilistic Turing machine with a “subpermutation” matrix S_T, i.e. S_T can be obtained from a suitable permutation matrix by deleting some of its rows.

One more variation of this idea is needed to finally arrive at the concept of a *quantum* Turing machine: we have to require that the transition function t is a mapping with a normalization condition

$$t : Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\} \longrightarrow \mathcal{C} \tag{4.4}$$

from a configuration to possibly many successors, each of which is given a complex amplitude. Here the normalization constraint says that the matrix U_T describing the dynamics of T on the state space is *unitary*.

Observe that there is one counterintuitive fact implied by this definition: as t assigns complex amplitudes to $Q \times \Sigma \times Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}$ (according to (4.4)), one can interpret a configuration of T as being in a superposition of (i) tape symbols in each individual cell, (ii) states of the finite-state machine supported by Q and (iii) positions of the head. The last point in particular might look uncomfortable at first sight, but we remind the reader of the fact that in classical probabilistic computation each individual configuration is assigned a probability, so one can think of a probabilistic computation as traversing an exponentially large configuration space! The main difference of the QTM model is that because of negative amplitudes, computational paths in this configuration space can cancel each other out, i.e. the effects of interference can force the Turing machine into certain paths which ultimately may lead to the desired solution of the computational task.

Now that the computational model of a QTM has been defined, the question arises as to what can be computed on a QTM, compared with a classical deterministic or probabilistic Turing machine. An important result in this context is that everything which can be computed classically in polynomial time can also be computed on a QTM because of the following theorem, which relies on some results of Bennett for reversible Turing machines [55, 198, 199] and was adapted to the QTM setting in [200]. As usual, we denote by L^* the language $\{0,1\}^*$ consisting of all binary strings and denote by $|x|$ the length of the word $x \in L^*$.

Theorem 4.5. *Let $f : L^* \to L^*$ be a polynomial-time computable function such that $|f(x)|$ depends only on $|x|$. Then there is a polynomial-time QTM T that computes $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$. The running time of T depends only on $|x|$.*

Proof. The basic idea is to replace each elementary step in the computation of f by a reversible operation (using theorem 4.3), keeping in mind that ancilla qubits are needed to make the computation reversible (see Sect. 4.2.3). We now adjoin additional qubits to the system, which are initialized in the ground state $|0\rangle$, and apply a controlled NOT operation using the computational qubits holding the result $f(x)$. Of course, after the application of this operation the state is highly entangled between the computational register and the additional register holding the result. Next, we run the whole computation that was done to compute f, backwards, reversibly, on the computation register to get rid of the garbage which might destroy the coherence, and end up with the state $|x, 0, f(x)\rangle$, where the $|0\rangle$ refers to the ancilla bits used in the first step of this procedure. $\square$

Remark 4.2.

- The class of quantum Turing machines allows the definition and study of the important complexity class BQP [200],[3] as well as the relation of BQP to other classes known from classical complexity theory.
- There are programming primitives for QTMs, such as composition, loops and branching [200], as in the classical case. However, a problem arises in realizing while-loops, since the predicate which decides whether the loop terminates can be in a superposition of true and false, depending on the computation path. Therefore all computations have to be arranged in such a way that this predicate is never in a superposed state, i.e. the state of the predicate has to be classical. As a consequence, we obtain the result that a quantum Turing machine can only perform loops with a prescribed number of iterations, which in turn can be determined by a classical Turing machine.
- An important issue is whether QTMs constitute an analog or discrete model of computation. One might be tempted to think of the possibility of encoding an arbitrary amount of information into the transition amplitudes of t, i.e. of producing a machine model which could benefit from computing with complex numbers to arbitrary precision (for the strange effects of such models see, e. g., [201]). However, see [200, 202] for a proof of the fact that it is sufficient to take transition amplitudes from the finite set $\{\pm 3/5, \pm 4/5, \pm 1, 0\}$ in order to approximate a given QTM to arbitrary precision. The reason for this is that the Pythagorean-triple transformation

$$\frac{1}{5}\begin{pmatrix} 3 & -4 \\ 4 & 3 \end{pmatrix} \in \mathrm{SO}(2)$$

  has eigenvalues of the form $\mathrm{e}^{2\pi \mathrm{i}\nu}$ with $\nu \notin \mathbb{Q}$ and, therefore, generates a dense subgroup in SO(2). The statement then follows from the fact that the full unitary group on $\mathcal{H}_{2^n}$ can be parametrized by SO(2) matrices applied to arbitrary basis states and phase rotations [172, 203].
- Yao has shown [204] that the computational models of QTM and uniform families of quantum gates (see Sect. 4.2.1) are polynomially equivalent, i.e. each model can simulate the other with polynomial time overhead.

[3] BQP stands for "bounded-error quantum polynomial time".

4.3 Using Entanglement for Computation: A First Quantum Algorithm

Entanglement between registers holding quantum states lies at the heart of the quantum algorithm which we shall describe in this section. The formulation of a problem on which a quantum computer will exceed the performance of any classical (probabilistic) computer may appear artificial. However, this was one of the first examples of problems on which a quantum computer could provably outperform any classical computer, with an exponential speedup.

Because of its clarity and methodology, we present the quantum algorithm of Simon, in which many of the basic principles of quantum computing, namely the superposition principle, computing with preimages and the use of the Fourier transform, become apparent. We briefly remind the reader of the problem and mention that we are considering here a slightly generalized situation compared with the original setup (see also [57, 205, 206]).

Quantum algorithms relying on the same principles have been given in [56, 200]. As in the case of Simon's problem described below, these algorithms rely on the Fourier transform for a suitably chosen abelian group. In both cases it has been shown that these quantum algorithms provide a superpolynomial gap over any classical probabilistic computer in the number of operations necessary to solve the corresponding problems.

In the following, we denote by $\mathbf{Z}_2^n$ the elementary abelian 2-group of order 2^n, the elements of which we think of as being identified with binary strings of length n, and denote addition in $\mathbf{Z}_2^n$ by $\oplus$.

Definition 4.3 (Simon's Problem). *Let $f : \mathbf{Z}_2^n \to \mathbf{Z}_2^n$ be a function given as a black-box quantum circuit, i.e. f can be evaluated on superpositions of states and is realized by a unitary transform V_f specified by*

$$|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle, \quad \textit{for all } x \in \mathbf{Z}_2^n \ ,$$

as described in Sect. 4.2.3 (see, in particular, (4.3)). In addition, it is specified that there is a subgroup $U \subseteq \mathbf{Z}_2^n$ (the "hidden" subgroup) such that f takes a constant value on each of the cosets $g \oplus U$ for $g \in \mathbf{Z}_2^n$ and, furthermore, f takes different values on different cosets. The problem is to find generators for U.

We can now formulate a quantum algorithm which solves Simon's problem in a polynomial number of operations on a quantum computer. This algorithm uses $O(n)$ evaluations of the black-box quantum circuit f, and the classical postcomputation, which is essentially linear algebra over $GF(2)$, and also takes a number of operations which is polynomial in n.

Algorithm 2 *This algorithm needs two quantum registers of length n, holding elements of the domain and codomain of f, and consists of the following steps.*

1. *Prepare the ground state*

$$|\varphi_1\rangle = |0\dots0\rangle \otimes |0\dots0\rangle$$

in both quantum registers.

2. *Achieve equal amplitude distribution in the first register, for instance by an application of a Hadamard transformation to each qubit:*

$$|\varphi_2\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbf{Z}_2^n} |x\rangle \otimes |0\dots0\rangle \ .$$

3. *Apply V_f to compute f in superposition. We obtain*

$$|\varphi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \mathbf{Z}_2^n} |x\rangle|f(x)\rangle \ .$$

4. *Measure the second register to obtain some value z in the range of f. Owing to the condition on f specified, the first register now holds a coset $g_0 \oplus U$ of the hidden subgroup U, namely the set of elements equal to $z = f(g_0)$:*

$$|\varphi_4\rangle = \frac{1}{\sqrt{|U|}} \sum_{f(x)=z} |x\rangle|z\rangle = \frac{1}{\sqrt{|U|}} \sum_{x \in g_0 \oplus U} |x\rangle|z\rangle \ .$$

5. *Application of the Hadamard transformation $H_2^{\otimes n}$ to the first register transforms the coset into the superposition $\sum_{y \in U^\perp} (-1)^{y \cdot g_0} |y\rangle$. The supported vectors of this superposition are the elements of $U^\perp$, which is the group defined by $U^\perp := \{y \in \mathbf{Z}_2^n : x \cdot y = \sum_{i=1}^n x_i y_i = 0\}$, i.e. the orthogonal complement of U with respect to the scalar product in $\mathbf{Z}_2^n$ (see also Sect. 4.4.2).*
6. *Now measure the first register. We draw from the set of irreducible representations of $\mathbf{Z}_2^n$ having U in the kernel, i.e. we obtain an equal distribution over the elements of $U^\perp$.*
7. *By iterating steps 1-6, we produce elements of $\mathbf{Z}_2^n$ which generate the group $U^\perp$ with high probability. After performing this experiment an expected number of n times, we generate with probability greater than $1-2^{-n}$ the group $U^\perp$.*
8. *By solving linear equations over $GF(2)$, it is easy to find generators for*

$$(U^\perp)^\perp = U \ .$$

Therefore we obtain generators for U by computing the kernel of a matrix over $GF(2)$ in time $O(n^3)$.

Analysis of Algorithm 2

- We first address the measurement in step 4. If we do not perform this measurement, we are left with the state

$$|\varphi_4'\rangle = \frac{1}{\sqrt{2^n}} \sum_{\sigma \in \mathbf{Z}_2^n/U} \sum_{x \in U} |\sigma \oplus x\rangle |f(\sigma)\rangle \ ,$$

  and if we continue with step 5 we shall obtain the state

$$|\varphi_5'\rangle = \frac{1}{\sqrt{2^n}} \sum_{\sigma \in \mathbf{Z}_2^n/U} \sum_{y \in U^\perp} (-1)^{\sigma \cdot y} |y\rangle |f(\sigma)\rangle \ .$$

  Therefore, sampling of the first register as in step 6 will yield an equal distribution over $U^\perp$ and we can go on as in step 7. Hence we can omit step 4.
- The reason for the application of the transformation $H_2^{\otimes n}$ and the appearance of the group $U^\perp$ will be clarified in the following sections. As it turns out, $H_2^{\otimes n}$ is an instance of a Fourier transform for an abelian group and $\perp$ is an antiisomorphism of the lattice of subgroups of $\mathbf{Z}_2^n$.
- For the linear-algebra part in step 8, we refer the reader to standard texts such as [207]. Gauss's algorithm for computing the kernel of an $n \times n$ matrix takes $O(n^3)$ arithmetic operations over the finite field $\mathbf{Z}_2 = GF(2)$. Overall, we obtain the following cost: $O(n)$ applications of V_f, $O(n^2)$ elementary quantum operations (which are all Hadamard operations H_2), $O(n^2)$ measurements of individual qubits, and $O(n^3)$ classical operations (arithmetic in $GF(2)$).

4.4 Quantum Fourier Transforms: the Abelian Case

In this section we recall the definition and basic properties of the discrete Fourier transform (DFT) and give examples of its use in quantum computing.

In most standard texts on signal processing (e.g. [208]), the DFT_N of a periodic signal given by a function $f : \mathbf{Z}_N \to \mathcal{C}$ (where $\mathbf{Z}_N$ is the cyclic group of order N) is defined as the function F given by

$$F(\nu) := \sum_{t \in \mathbf{Z}_N} \mathrm{e}^{2\pi \mathrm{i} \nu t/N} f(t) \ .$$

If, equivalently, we adopt the point of view that the signal f and the Fourier transform are vectors in $\mathcal{C}^N$, we see that performing the DFT_N is a matrix vector multiplication of f with the unitary matrix

$$\mathrm{DFT}_N := \frac{1}{\sqrt{N}} \cdot \left[\omega^{i \cdot j}\right]_{i,j=0,\dots,N-1} ,$$

where $\omega = \mathrm{e}^{2\pi \mathrm{i}/N}$ denotes a primitive Nth root of unity.

From an algebraic point of view, the DFT_N gives an isomorphism Φ of the group algebra $\mathcal{C}\mathbf{Z}_N$,

$$\Phi : \mathcal{C}\mathbf{Z}_N \longrightarrow \mathcal{C}^N ,$$

onto the direct sum of the irreducible matrix representations of $\mathbf{Z}_N$ (where multiplication is performed pointwise). This means that DFT_N decomposes the regular representation of $\mathbf{Z}_N$ into its irreducible constituents. It is known that this property allows the derivation of a fast convolution algorithm [171] in a canonical way and can be generalized to more general group circulants. Viewing the DFT as a decomposition matrix for the regular representation of a group leads to the generalization of Fourier transforms to arbitrary finite groups (cf. [171] and Sect. 4.6.1).

The fact that DFT_N can be computed in $O(N \log N)$ arithmetic operations (counting additions and multiplications) is very important for applications in classical signal processing. This possibility to perform a fast Fourier transform justifies the heavy use of the DFT_N in today's computing technology. In the next section we shall show that the $O(N \log N)$ bound, which is sharp in the arithmetic complexity model (see Chap. 4 and 5 of [174]) can be improved with a quantum computer to $O[(\log N)^2]$ operations.

4.4.1 Factorization of DFT_N

From now on we restrict ourselves to cases of DFT_N where $N = 2^n$ is a power of 2, since these transforms naturally fit the tensor structure imposed by the qubits.

The efficient implementation of the Fourier transform on a quantum computer starts from the well-known Cooley–Tukey decomposition [209]: after the row permutation Π_τ, where $\tau = (1, \ldots, n)$ is the cyclic shift on the qubits, is performed, the DFT_{2^n} has the following block structure [171]:

$$\begin{aligned}\Pi_\tau\, \mathrm{DFT}_{2^n} &= \left(\begin{array}{c|c} \mathrm{DFT}_{2^{n-1}} & \mathrm{DFT}_{2^{n-1}} \\ \hline \mathrm{DFT}_{2^{n-1}}\, W_n & -\mathrm{DFT}_{2^{n-1}}\, W_n \end{array}\right) \\ &= (\mathbf{1}_2 \otimes \mathrm{DFT}_{2^{n-1}}) \cdot T_n \cdot (\mathrm{DFT}_2 \otimes \mathbf{1}_{2^{n-1}}) \ .\end{aligned}$$

Here we denote by

$$T_n := \mathbf{1}_{2^{n-1}} \oplus W_n, \quad W_n := \begin{pmatrix} 1 & & & & \\ & \omega_{2^n} & & & \\ & & \omega_{2^n}^2 & & \\ & & & \ddots & \\ & & & & \omega_{2^n}^{2^{n-1}} \end{pmatrix}$$

the matrix of *twiddle factors* [171]. Taking into account the fact that W_n has the tensor decomposition

$$W_n = \bigotimes_{i=0}^{n-2} \begin{pmatrix} 1 & \\ & \omega_{2^n}^{2^i} \end{pmatrix},$$

we see that T_n can be implemented by $n-1$ gates having one control wire each. These can be factored into the elementary gates $\mathcal{G}_1$ with constant overhead.

Because tensor products are free in our computational model, by recursion we arrive at an upper bound of $O(n^2)$ for the number of elementary operations necessary to compute the discrete Fourier transform on a quantum computer (this operation will be referred to as “QFT”).

In Fig. 4.8, the derived decomposition into quantum gates is displayed using the graphical notation introduced in Sect. 4.2.1. The gates labeled by D_k in this circuit are the diagonal phase shifts $\operatorname{diag}(1, \mathrm{e}^{2\pi \mathrm{i}/k})$ and, in addition, we have used the abbreviation $N = 2^n$. We observe that the permutations Π_n, which arose in the Cooley–Tukey formula, have all been collected together, yielding the so-called bit reversal, which is the permutation of the quantum wires $(1, n)\,(2, n-1) \ldots (n/2, n/2 + 1)$ when n is even and $(1, n)\,(2, n-1) \ldots ((n-1)/2, (n+3)/2)$ when n is odd.

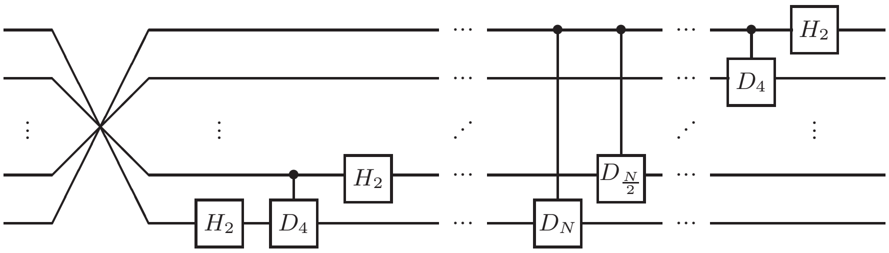

Fig. 4.8. Quantum circuit that computes a Fourier transform QFT_{2^n}

If we intend to use the Fourier transform as a sampling device, i.e. the application of QFT_{2^n} to a state $|\psi\rangle$ followed directly by a measurement in the standard basis, we can use the structure of this quantum circuit that computes a Fourier transform QFT_{2^n} to avoid nearly all quantum gates [210].

In order to give circuits for the QFT for an arbitrary abelian group, we need to describe the structure of these groups and their representations first, which is done in the next section.

4.4.2 Abelian Groups and Duality Theorems

Let A be a finite abelian group. Then A splits into a direct product of its p-components: $A \cong A_{p_1} \times \ldots \times A_{p_n}$, where $p_i, i = 1, \ldots, n$ are the prime divisors of the order $|A|$ of A (see [211], Part I, paragraph 8).

Furthermore, each A_{p_i}, which consists of all elements annihilated by a power of p_i, has the form $A_{p_i} \cong \mathbf{Z}_{p^{r_{i,1}}} \times \ldots \times \mathbf{Z}_{p^{r_{i,k_i}}}$ (see [211], Part I, paragraph 8). Both statements follow immediately from the structure theorem for finitely generated modules over principle ideal rings.

We remark that a Fourier transform for a finite abelian group can easily be constructed from this knowledge: given two groups G_1 and G_2, the irreducible representations of their direct product $G_1 \times G_2$ can be obtained from those of G_1 and G_2 as follows.

Theorem 4.6. *The irreducible representations of $G = G_1 \times G_2$ are given by*

$$\mathrm{Irr}(G) = \{\phi_1 \otimes \phi_2 : \phi_1 \in \mathrm{Irr}(G_1), \phi_2 \in \mathrm{Irr}(G_2)\} \ .$$

(For a proof see [212].) If we therefore encode the elements of A according to the direct-product decomposition as above, the matrix

$$\mathrm{DFT}_A = \bigotimes_{i=1}^{n} \bigotimes_{j=1}^{k_i} \mathrm{DFT}_{{p_i}^{r_{i,j}}}$$

is a decomposition matrix for the regular representation of A.

Corollary 4.1. *Let A be a finite abelian group of order 2^n. Then a Fourier transform for A can be computed in $O(n^2)$ elementary operations.*

Proof. The decomposition of DFT_{2^n} has already been considered and an implementation in $O(n^2)$ many operations has been derived from the Cooley–Tukey formula in Sect. 4.4.1. Since tensor products are free in our computational model, we can conclude that a direct factor $\mathbf{Z}_{2^n}$ is already the worst case for an implementation. □

Example 4.6. The Fourier transform for the elementary abelian 2-group $\mathbf{Z}_2^n$ is given by the tensor product $\mathrm{DFT}_2 \otimes \cdots \otimes \mathrm{DFT}_2$ of the Fourier transform for the cyclic factors and hence coincides with the Hadamard matrix $H_2^{\otimes n}$ used in Algorithm 2.

The Dual Group. Given a finite abelian group A, we can consider $\mathrm{Hom}(A,\mathcal{C}^*)$, i.e. the group of characters of A (NB: in the nonabelian case a character is generalized to the traces of the representing matrices and hence is not a homomorphism anymore). The following theorem says that A is isomorphic to its group of characters.

Theorem 4.7. *For a finite abelian group A, we have $A \cong_\phi \mathrm{Hom}(A, \mathcal{C}^*)$.*

We shall make the isomorphism ϕ explicit. Choose $\omega_{p_1}, \ldots, \omega_{p_n}$, where the ω_i are primitive p_ith roots of unity in $\mathcal{C}^*$. Then ϕ is defined by the assignment $\phi(e_i) := \omega_{p_i}$ on the elements $e_i := (0, \ldots, 1, \ldots, 0)$, where the 1 is in the ith position. The isomorphism ϕ is not canonical, since a different choice of primitive roots of unity will yield a different isomorphism.

Next we observe that there is a pairing β (i.e. a bilinear map) between A and $\mathrm{Hom}(A, \mathcal{C}^*)$ via $\beta : (a, f) \mapsto f(a) \in \mathcal{C}^*$. We suppose that a is orthogonal to f (in symbols, $a \perp f$) if $\beta(a, f) = 1$. For a given subgroup $U \subseteq A$ we can now define (see also Algorithm 2) the orthogonal complement $U^\perp := \{y \in A : \beta(y, x) = 1, \quad \forall x \in U\}$. The following duality theorem holds (see [184]):

Theorem 4.8. *For all subgroups $U, U' \subseteq A$, the following identities hold:*

(a) Self-duality of $\perp$:

$$(U^\perp)^\perp = U \ .$$

(b) Complementarity:

$$(U \cap U')^\perp = \langle U^\perp, U'^\perp \rangle \ .$$

(c) The mapping $\perp$ is an inclusion-reversing antiisomorphism on the lattice of subgroups of A (i.e. $\perp$ is a Galois correspondence).

4.4.3 Sampling of Fourier Coefficients

In this section we address the problem of gaining information from the Fourier coefficients of a special class of states. More precisely, we consider the Fourier transforms of the (normalized) characteristic function

$$|\chi_{c+U}\rangle := \frac{1}{\sqrt{|U|}} \sum_{x \in c+U} |x\rangle \tag{4.5}$$

of a coset $c + U$ of a subgroup $U \subseteq A$ of an abelian group A. The question of what conclusions about U can be drawn from measuring the Fourier transformed state (4.5) is of special interest, as we have already seen in Sect. 4.3. The following theorem shows that for an abelian group, the Fourier transform maps subgroups to their duals.

Theorem 4.9. *Let* $\mathrm{DFT}_A = (1/\sqrt{|A|}) \sum_{x,y \in A} \beta(x, y) |y\rangle \langle x|$ *be the Fourier transform for the abelian group A. Then for each subgroup $U \subseteq A$, we have*

$$\mathrm{DFT}_A \frac{1}{\sqrt{|U|}} \sum_{x \in U} |x\rangle = \frac{1}{\sqrt{|U^\perp|}} \sum_{y \in U^\perp} |y\rangle \ .$$

Proof. Since

$$\begin{aligned} \mathrm{DFT}_A \frac{1}{\sqrt{|U|}} \sum_{x \in U} |x\rangle &= \frac{1}{\sqrt{|U|}} \left(\sum_{x,y \in A} \beta(x, y) |y\rangle \langle x| \right) \sum_{x \in U} |x\rangle \\ &= \frac{1}{\sqrt{|A||U|}} \sum_{y \in A} \sum_{x \in U} \beta(x, y) |y\rangle \ , \end{aligned}$$

it suffices to show that $\sum_{x\in U} \beta(x,y) = 0$ for $y \notin U^\perp$, but this statement follows from the fact that the existence of x_0 with $\beta(x_0, y) \neq 0$ implies that $\sum_{x\in U} \beta(x,y) = \sum_{x\in U} \beta(x+x_0,y) = \beta(x_0,y)\sum_{x\in U} \beta(x,y)$. The other case, $\sum_{x\in U} \beta(x,y) = |U|$ for $y \in U^\perp$, is obvious. □

Hence, measuring the Fourier spectrum of $|\chi_U\rangle$ yields an equal distribution on the elements of the dual group $U^\perp$. Also, in the case of the characteristic function of a coset $c+U$ instead of U, we obtain the same probability distribution on $U^\perp$ since the Fourier transform diagonalizes the group action completely in the abelian case, i.e. the translation by c corresponds to a pointwise multiplication by phases in the Fourier basis:

$$\mathrm{DFT}_A \frac{1}{\sqrt{|U|}} \sum_{x\in c+U} |x\rangle = \frac{1}{\sqrt{|U^\perp|}} \sum_{y\in U^\perp} \varphi_{c,y} |y\rangle \ ,$$

where $\varphi_{c,y} \in \mathcal{U}(1)$ are phase factors which depend on c and y but are always eth roots of unity, where e denotes the exponent of A. Since making measurements involves taking the squares of the amplitudes, we obtain an equal distribution over $U^\perp$. The states $|\chi_{c+U}\rangle$, which in general will be highly entangled, make the principle of *interference* and its use in quantum algorithms apparent: only those Fourier coefficients remain which correspond to the elements of $U^\perp$ (constructive interference), whereas the amplitudes of all other elements vanish (destructive interference).

4.4.4 Schur's Lemma and its Applications in Quantum Computing

In this section we explore the underlying reason behind Theorem 4.9, namely Schur's lemma. We present further applications of this powerful tool from representation theory. For a proof of Schur's lemma we refer the reader to Sect. 2 of [213].

Lemma 4.2 (Schur's lemma). *Let $\rho_1 : G \to \mathrm{GL}_{\mathcal{C}}(V)$ and $\rho_2 : G \to \mathrm{GL}_{\mathcal{C}}(W)$ be irreducible complex representations of a group G. Suppose that the element $A \in \mathrm{End}(V,W)$ has the property*

$$\rho_1(g) \cdot A = A \cdot \rho_2(g), \forall g \in G \ ,$$

i.e. A commutes with all pairs of images $\rho_1(g), \rho_2(g)$. Then exactly one of the following cases holds:

(i) $\rho_1 \not\cong \rho_2$. In this case $A = \mathbf{0}_{\mathrm{End}(V,W)}$.
(ii) $\rho_1 \cong \rho_2$. In this case A is a homothety, i.e. $A = \lambda \cdot \mathbf{1}_{\mathrm{End}(V,W)}$ with an element $\lambda \in \mathcal{C}$.

We give two applications of Schur's lemma which make its importance in the context of Fourier analysis apparent. First, we present a reformulation of Theorem 4.9 in representation-theoretical terms and state then a theorem which is useful in the sampling of functions having a hidden normal subgroup.

Theorem 4.10. *Let G be a finite abelian group,* $\mathrm{Hom}(G, \mathcal{C}^*)$ *the dual group, and $\beta : G \times \mathrm{Hom}(G, \mathcal{C}^*) \to \mathcal{C}^*$ the canonical pairing defined by $\beta(g, \varphi) := \varphi(g)$. Then for each subgroup $U \subseteq G$, the following holds (normalization omitted):*

$$\mathrm{DFT}_G \sum_{u \in U} |u\rangle = \sum_{\substack{\varphi \in \mathrm{Hom}(G,\mathcal{C}^*) \\ U \subseteq \mathrm{Ker}(\varphi)}} |\varphi\rangle \ .$$

Moreover, we obtain the following identity for the cosets $u_0 + U$:

$$\mathrm{DFT}_G \sum_{u \in U} |u_0 + u\rangle = \sum_{\substack{\varphi \in \mathrm{Hom}(G,\mathcal{C}^*) \\ U \subseteq \mathrm{Ker}(\varphi)}} \beta(u_0, \varphi) \cdot |\varphi\rangle \ .$$

Proof. The mapping $\mathrm{DFT}_G : \mathcal{C}G \to \bigoplus_{i=1}^{|G|} \mathcal{C}$ is given by evaluation of elements of G for the irreducible representations $\{\varphi_1, \ldots, \varphi_s\}$ of G, which are all one-dimensional (i.e. $s = |G|$) and hence are characters, since G was assumed to be abelian. Therefore, the coefficient for the irreducible representation φ_i is computed from

$$\begin{aligned}\sum_{u \in U} \varphi_i(u_0 + u) &= \sum_{u \in U} \varphi_i(u_0) \cdot \varphi_i(u) \\ &= \varphi_i(u_0) \cdot \sum_{u \in U} \varphi_i(u) \ .\end{aligned}$$

Considering the restricted characters $\varphi_i \downarrow U$, we use Schur's lemma (Lemma 4.2) to deduce that $\sum_{u \in U} \varphi_i(u) = 0$ iff $U \not\subseteq \mathrm{Ker}(\varphi_i)$. □

Theorem 4.11. *Let G be an arbitrary finite group and $N \lhd G$ a normal subgroup of G. Then for each irreducible representation ρ of G of degree d, exactly one of the following cases holds:*

$$\frac{1}{|N|} \sum_{n \in N} \rho(n) = \lambda \cdot \mathbf{1}_{d \times d} \ , \quad \text{or} \quad \frac{1}{|N|} \sum_{n \in N} \rho(n) = \mathbf{0}_{d \times d} \ ,$$

where the first case applies iff N is contained in the kernel of ρ.

Proof. Let $A := (1/|N|) \sum_{n \in N} \rho(n)$ denote the equal distribution over all images of N under ρ. Then, from the assumption of normality of N, we obtain

$$\rho(g)^{-1} \cdot A \cdot \rho(g) = A \ ,$$

i.e. A commutes with the irreducible representation ρ. Using Schur's lemma, we conclude that $A = \lambda \cdot \mathbf{1}_{d \times d}$. If $N \subseteq \mathrm{Ker}(\rho)$ we find $A = \mathbf{1}_{d \times d}$. On the other hand, if $N \not\subseteq \mathrm{Ker}(\rho)$, we conclude that A is the zero matrix $\mathbf{0}_{d \times d}$, because otherwise an element $n_0 \notin N$ equal to $\rho(n_0) \neq \mathbf{1}_{d\times d}$ would lead to the contradiction

$$\rho(n_0) \cdot \sum_{n\in N} \rho(n) = \sum_{n\in N} \rho(n_0 \cdot n) = \sum_{n\in N} \rho(n) \ ,$$

since from this we could conclude $\rho(n_0) = \mathbf{1}_{d\times d}$, contrary to the assumption $n_0 \notin \mathrm{Ker}(\rho)$. □

4.5 Exploring Quantum Algorithms

4.5.1 Grover's Algorithm

We give an outline of Grover's algorithm for searching an unordered list and present an optical implementation using Fourier lenses.

The search algorithm allows one to find elements in a list of N items fulfilling a given predicate in time $O(N^{1/2})$. We assume that the predicate f is given by a quantum circuit V_f and, as usual in this setting, we count the invocations of V_f (the "oracle"). It is straightforward to construct from V_f an operator S_f which flips the amplitudes of the states that fulfill the predicate, i.e. $S_f : |x\rangle \mapsto (-1)^{f(x)}|x\rangle$ for all basis states $|x\rangle$. The Grover algorithm relies on an averaging method called *inversion about average*, which is described in the following.

Consider the matrix

$$D_n := \begin{pmatrix} -1+\frac{2}{2^n} & \frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \frac{2}{2^n} & -1+\frac{2}{2^n} & \dots & \frac{2}{2^n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{2^n} & \frac{2}{2^n} & \dots & -1+\frac{2}{2^n} \end{pmatrix} . \tag{4.6}$$

D_n is a circulant matrix [214], i.e. $D_n = \mathrm{circ}_G(-1+(2/2^n), (2/2^n), \dots, (2/2^n))$ for any choice of a finite group G. To see this, we recall the definition of a general group circulant,

$$\mathrm{circ}_G(v) := (v_{g_i^{-1}\cdot g_j})_{1\leq i,j\leq |G|}$$

for a fixed ordering $G = \{g_1, \dots, g_{|G|}\}$ of the elements of G and for a vector v which is labeled by G. To implement (4.6) on a quantum computer we apply the circulant for the group $\mathbf{Z}_2^n$, making use of the following theorem.

Theorem 4.12. *The Fourier transform* DFT_A *for a finite abelian group* A *implements a bijection between the set of* A*-circulant matrices and the set of diagonal matrices over* $\mathcal{C}$*. Explicitly, each circulant* C *is of the form*

$$C = \mathrm{DFT}_A^{-1} \cdot \mathrm{diag}(d_1, \dots, d_n) \cdot \mathrm{DFT}_A$$

and the vector $\mathbf{d} = (d_1, \dots, d_n)$ *of diagonal entries is given by* $\mathbf{d} = \mathrm{DFT}_A \cdot \mathbf{c}$*, where* $\mathbf{c}$ *is the first row of* C*.*

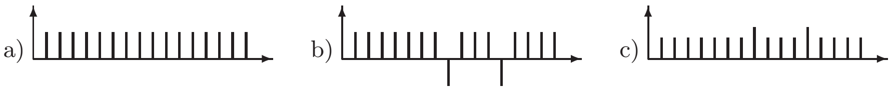

Fig. 4.9. (**a**) Equal distribution, (**b**) flip solutions, (**c**) inversion about average

Grover's original idea was to use correlations to amplify the amplitudes of the states fulfilling the predicate, i.e. to correlate – starting from an initial distribution, which is chosen to be the equal distribution $P(X = i) = 1/N, i = 1, \dots, N$ – the vector $(-1, 1, \dots, 1)$ with the probability distribution obtained by flipping the signs of the states fulfilling the predicate.

Starting from the equal distribution $H_2^{\otimes n}|0\rangle$, the amplification process in Grover's algorithm consists of an iterated application of the operator $-D_n S_f$ $O(\sqrt{2^n})$ times [215]. In Fig. 4.9, the steps of this procedure are illustrated in a qualitative way.

We present a realization of the Grover algorithm with a diffractive optical system. This is not a quantum mechanical realization in the sense of quantum computing, since such a system does not support a qubit architecture. However, the transition matrices are unitary and hence we can consider simulations of quantum algorithms via optical devices. These optical setups scale linearly with the dimension of the computational Hilbert space rather than with the logarithmic growth of a quantum register; nevertheless they have some remarkable properties, the best known of which is the ability to perform a Fourier transform by a simple application of a cylindrical lens [216]. This resembles the classical Fourier transform (corresponding to $\mathrm{DFT}_{\mathbf{Z}_{2^n}}$ rather than $\mathrm{DFT}_{\mathbf{Z}_2^n}$), which is correct because of the comments preceding Theorem 4.12.

Observe that by starting from this operation, we can easily perform correlations since this corresponds to a multiplication with a circulant matrix. Every circulant matrix can be realized optically using a so-called $4f$ setup, which corresponds to a factorization of a circulant matrix C into diagonal matrices and Fourier transforms (following Theorem 4.12): $C = \mathrm{DFT}^{-1} \cdot D \cdot \mathrm{DFT}$, where D is a given diagonal matrix (see also Fig. 4.10).

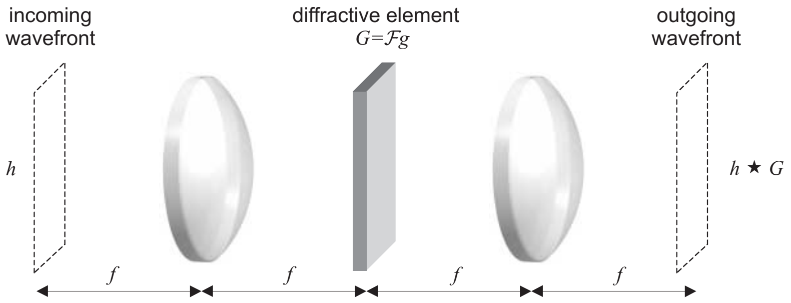

Fig. 4.10. Optical $4f$ setup that computes a convolution of h and g

Remark 4.3. The question of what linear transforms can be performed optically is equivalent to the question of what matrices can be factored into diagonal matrices and Fourier transforms, which correspond to diagonal matrices and circulant matrices. It has been shown in [217] that for fields $K \notin \{GF(3), GF(5)\}$, every square matrix M with entries in K can be written as a product of circulant and diagonal matrices with entries in K. Furthermore, if M is unitary the circulant and diagonal factors can also be chosen to be unitary [217].

Remark 4.4. There have been several generalizations and modifications of the original Grover algorithm.

- The case of an unknown number of solutions fulfilling the given predicate is analyzed in [218].
- The issue of arbitrary initial distributions (instead of an equal distribution) is considered in [219].
- In [215] it is shown that instead of the diffusion operators D_n, almost any (except for a set of matrices of measure zero) unitary matrix can be used to perform the Grover algorithm.
- There is a quantum algorithm for the so-called collision problem which needs $O(\sqrt[3]{N/r})$ evaluations of a given r-to-one function f to find a pair of values which are mapped to the same element [220].
- The Grover algorithm has been shown to be optimal, i.e. the problem of finding an element in an unordered list takes $\Theta(\sqrt{N})$ operations on a quantum computer [200].
- Some other problems have been solved by a subroutine call to the Grover algorithm, e.g. a problem in communication complexity: the problem of deciding whether $A, B \subseteq \{1, \ldots, N\}$ are disjoint or not. Using the Grover algorithm, it can be shown that it is sufficient for the two parties, one holding A and the other holding B, to communicate $O(\sqrt{N} \log N)$ qubits to solve this problem [221].

4.5.2 Shor's Algorithm

In this section we briefly review Shor's factorization algorithm and show how the Fourier transform comes into play. It is known that factoring a number N is easy under the assumption that it is easy to determine the (multiplicative) order of an arbitrary element in $(\mathbf{Z}_N)^\times$. For a proof of this, we refer the reader to [222] and to Shor's original paper [9].

Once this reduction has been done, the following observation is the crucial step for the quantum algorithm. Let y be randomly chosen and let $\gcd(y, N) = 1$. To determine the multiplicative order r of y mod N, consider the function

$$f_y(x) := y^x \bmod N \ .$$

Clearly $f_y(x+r) = f_y(x)$, i.e. f_y is a periodic function with period r. The quantum algorithm to determine this period is as follows:

Algorithm 3 *Let N be given; determine $M = 2^m$ such that $N^2 \leq M \leq 2N^2$. This number M will be the length of the Fourier transform to be performed in the following.*

1. *Randomly choose y with $\gcd(y, N) = 1$.*
2. *Prepare the state $|0\rangle \otimes |0\rangle$ in two registers of lengths m and $\lceil \log_2 N \rceil$.*
3. *Application of the Hadamard transform $H_2^{\otimes m}$ to the left part of the register results in a superposition of all possible inputs*
$$\frac{1}{\sqrt{M}} \sum_{x=0}^{M-1} |x\rangle \otimes |0\rangle \ .$$
4. *We construct a unitary operation which computes the (partial function) $|x\rangle|0\rangle \mapsto |x\rangle|y^x \bmod N\rangle$ following Sect. 4.2.3 and 4.2.4. Calculation of $f_y(x) = y^x \bmod N$ yields for this superposition (normalization omitted)*
$$\sum_{x=0}^{M-1} |x\rangle \otimes |y^x \bmod N\rangle \ .$$
5. *Measuring the right part of the register gives a certain value z_0. The remaining state is the superposition of all x satisfying $f_y(x) = z_0$:*
$$\sum_{y^x = z_0} |x\rangle|z_0\rangle = \sum_{k=0}^{s-1} |x_0 + kr\rangle|z_0\rangle \ , \quad \text{where } y^{x_0} = z_0 \text{ and } s = \left\lfloor \frac{M}{r} \right\rfloor \ .$$
6. *Performing a QFT_M on the left part of the register leads to*
$$\sum_{l=0}^{M-1} \sum_{k=0}^{s-1} \mathrm{e}^{2\pi \mathrm{i}(x_0+kr)l/M} |l\rangle|z_0\rangle \ .$$
7. *Finally, a measurement of the left part of the register gives a value l_0.*

Application of this algorithm produces data from which the period r can be extracted after classical postprocessing involving Diophantine approximation (see Sect. 4.5.2).

A thorough analysis of this algorithm must take into account the overhead for the calculation of the function $f_y : x \mapsto y^x \bmod N$. However, after this function has been realized as a quantum network once (which can be obtained from a classical network for this function in polynomial time), the superposition principle applies since all inputs can be processed by *one* application of f_y.

The role played by Fourier transforms in this algorithm is twofold. In step 3 it is used to generate a superposition of all inputs from the ground state $|0\rangle$. This could have been done by *any* unitary transformation having an all-one vector in the first column. In step 6 we use the QFT_M to extract the information about the period r which was hidden in the graph of the function f_y.

Remark 4.5. It should be noted that for small numbers, as in the example of Fig. 4.11 and 4.12, an optical setup using Fourier lenses (cf. Fig. 4.10) could implement Shor's algorithm. This very example was initially simulated with the DigiOpt® system [223].

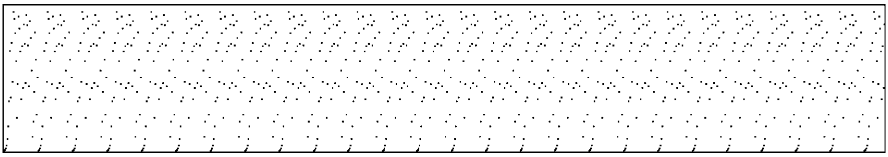

Fig. 4.11. Function graph of $f(x) = 2^x \bmod 187$

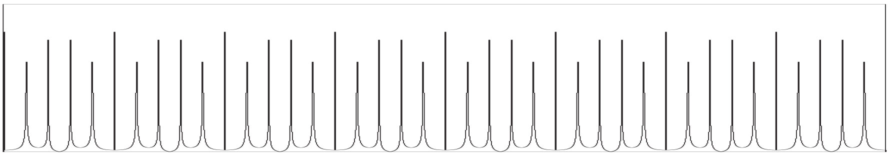

Fig. 4.12. The function shown in Fig. 4.11. transformed by a DFT of length 1024

An important question is the behavior of the states obtained after step 6 if the length M of the Fourier transform QFT_M we are using does not coincide with the period r of the function which is transformed. Note that this period r is exactly what has to be determined, and therefore the length N must be chosen appropriately in order to gain enough information about r from sampling. As it turns out, the choice $M = 2^m$ according to the condition $N^2 \leq M \leq 2N^2$ yields peaks in the Fourier domain which are sharply concentrated around the values lM/r [9].

In Fig. 4.11 and 4.12, this effect is illustrated for $N = 187$ and $M = 1024$. The choice of M does not fulfill the condition $N^2 \leq M \leq 2N^2$. Nevertheless, the characteristic peaks in the Fourier spectrum, which are sharply peaked around multiples of the inverse of the order, are apparent.

Let us now consider this state, which has been *oversampled* by transforming it with a Fourier transform of length M followed by a measurement, a little more closely. Constructive interference in $\sum_{k=0}^{s-1} \mathrm{e}^{2\pi \mathrm{i}(x_0+kr)l/M}|l\rangle$ will occur for those basis states $|l\rangle$ for which lr is close to M. The probability of measuring a specific l in this sum can be bounded by

$$\Big|\frac{1}{\sqrt{sM}}\sum_{k=0}^{s-1} \mathrm{e}^{2\pi \mathrm{i} krl/M}\Big|^2 \geq \Big|\frac{1}{\sqrt{sM}}\sum_{k=0}^{s-1} \mathrm{e}^{\pi \mathrm{i} r/M}\Big|^2 = \frac{1}{sM}\frac{|1-\mathrm{e}^{\pi \mathrm{i} rs/M}|^2}{|1-\mathrm{e}^{\pi \mathrm{i} r/M}|^2}$$
$$= \frac{1}{sM}\frac{|\sin[\pi rs/(2M)]|^2}{|\sin[\pi r/(2M)]|^2} \geq \frac{4}{\pi^2 r} \ .$$

The fractions l/M that we obtain from sampling fulfill the condition

$$\Big|\frac{l}{M} - \frac{p}{r}\Big| \leq \frac{1}{2M} \ ,$$

for some integer p. Because of the choice $M \geq N^2$ we obtain the result that l/M can be approximated efficiently by continued fractions, as described in the following section. This classical postprocessing completes the description of Shor's algorithm for finding the order r of y. To factor N we compute the least common divisors of $(y^{r/2}+1, N)$ and $(y^{r/2}-1, N)$, obtaining nontrivial factors of N if r is even and $y^{r/2} \not\equiv \pm 1 \mod N$.

A similar method can be applied to the discrete logarithm problem [9]. We mention that both the factoring and the discrete logarithm problem can be readily recognized as hidden-subgroup problems (see also Sect. 5): in the case of factoring, this corresponds to the group U generated by y and we are interested in the index $[\mathbf{Z} : U]$, which equals the multiplicative order of y. The discrete logarithm problem for $GF(q)^\times$ can be considered a hidden-subgroup problem for the group $G = \mathbf{Z} \times \mathbf{Z}$ and the function $f : G \rightarrow GF(q)^\times$ given by $f(x, y) \rightarrow \zeta^x \alpha^{-y}$, where ζ is the primitive element and α is the element for which we want to compute the logarithm.

The main difference from the situation in Simon's algorithm (see Sect. 4.3) is that in these cases we cannot apply the Fourier transform for the parent group G, since we do not know its order a priori. Rather, we have to compute larger Fourier transforms; preferably, the length is chosen to be a power of 2, to oversample. We then obtain the information from the sampled Fourier spectrum by classical postprocessing.

The basic features of the method of Fourier sampling which we invoked in Shor's factoring algorithm are also incorporated in Kitaev's algorithm for the abelian stabilizer problem [187]. At the very heart of Kitaev's approach is a method to measure the eigenvalues of a unitary operator U, supposing that the corresponding eigenvectors can be prepared. This estimation procedure becomes efficient if, besides U, the powers $U^{2^i}, i = 0, 1, \ldots,$ can also be implemented efficiently [180, 187]. We must also mention that the method of Diophantine approximation is crucial for the phase estimation.

Diophantine Approximation. In the following we briefly review some properties of the continued-fraction expansion of a real number. An important property a number can have is to be one of the convergents of such an expansion. More specifically, the following theorem holds, which is important in the sampling part of Shor's algorithm for recovering the exact eigenvalues of an operator from the data sampled.

Theorem 4.13. *For each $x \in \mathbb{R}$ and each fraction p/q fulfilling*

$$\left| x - \frac{p}{q} \right| < \frac{1}{2\,q^2} \ ,$$

the following holds: p/q is a convergent of the continued-fraction expansion of the real number x.

A proof of this theorem can be found in standard texts on elementary number theory (e.g. [224]). There exists a simple algorithm for producing an expansion of a rational number into a continued fraction:

Algorithm 4 *Let $x \in \mathbb{Q}$. Define $a_0 := \lfloor x \rfloor, x_1 := 1/(x-a_0), a_1 := \lfloor x_1 \rfloor, x_2 := 1/(x_1 - a_1)$ and so on, until we obtain $x_i = a_i$ for the first time. Then we can write x as*

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{\cdots + \frac{1}{a_n}}}} \ .$$

We mention in addition that Algorithm 4 not only yields optimal approximations (if applied to elements $x \in \mathbb{R}$) but is also very efficient, since in principle just a Euclidean algorithm is performed.

4.5.3 Taxonomy of Quantum Algorithms

The quantum algorithms which have been discovered by now fall into two categories, the principles of which we shall describe in the following.

Entanglement-Driven Algorithms. Suppose we are given a function $f : X \to Y$ from a (finite) domain X to a codomain Y. This function does not have to be injective; however, for a quantum computer to be able to perform f with respect to a suitably encoded X and Y, the function f has to be embedded into a unitary matrix V_f (cf. Sect. 4.2.3). We then can compute simultaneously the images of all inputs $x \in X$ using $|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$ for all $x \in X$ by preparing an equal superposition $\sum_{x \in X} |x\rangle$ in the X register and the ground state $|0\rangle$ in the Y register first, and then applying the quantum circuit V_f to obtain $\sum_{x \in X} |x\rangle|f(x)\rangle$. This entangled state can then be written as

$$\sum_{y \in \mathrm{Im}(f)} \Big(\sum_{x : f(x) = y} |x\rangle \Big) |y\rangle \ ,$$

i.e. we obtain a separation of the preimages of f. Measuring the second register leaves us with one of these preimages.

Pars pro toto, we mention the algorithms of Shor, which have been described in Sect. 4.5.2. In this case the function f is given by $f(x) = a^x \bmod N$, where N is the number to be factored and a is a random element in $\mathbf{Z}_N$.

Superposition-Driven Algorithms. The principle of this class of algorithms is to amplify to a certain extent the amplitudes of a set of "good" states, e.g. states which are specified by a predicate given in the form of a quantum circuit, and on the other hand to shrink the amplitudes of the "bad" states.

Pars pro toto, we mention the Grover algorithm for searching an unordered list. We gave an outline of this algorithm in Sect. 4.5.1 and presented an optical implementation using Fourier lenses.

4.6 Quantum Signal Transforms

Abelian Fourier transforms have been used extensively in the algorithms in the preceding sections. We now consider further classes of unitary transformations which admit efficient realizations in a computational model of quantum circuits. In Sect. 4.6.1 we introduce generalized Fourier transforms which yield unitary transformations parametrized by (nonabelian) finite groups. Classically, advanced methods have been developed for the study of fast Fourier transforms for *solvable* groups [171, 174, 225, 226]. While it is an open question whether efficient quantum Fourier transforms exist for all finite solvable groups, it is possible to give efficient circuits for special classes (see Sect. 4.6.1). In Sect. 4.6.2 we consider a class of real orthogonal transformations useful in classical signal processing, for which quantum circuits of polylogarithmic size exist.

4.6.1 Quantum Fourier Transforms: the General Case

In Sect. 4.4 we have encountered the special case of the discrete Fourier transform for abelian groups. This concept can be generalized to arbitrary finite groups, which leads to an interesting and well-studied topic for classical computers. We refer to [171, 174, 225, 226] as representatives of a vast number of publications. The reader not familiar with the standard notations concerning group representations is referred to these publications and to standard references such as [213, 227]. Following [228], we briefly present the terms and notations from representation theory which we are going to use, and recall the definition of Fourier transforms.

The Wedderburn Decomposition. Let ϕ be a regular representation of the finite group G. Then a *Fourier transform* for G is any matrix A that decomposes ϕ into irreducible representations with the additional property that equivalent irreducibles in the corresponding decomposition are equal. Regular representations are not unique, since they depend on the ordering of the elements of G. Also, note that this definition says nothing about the choice of the irreducible representations of G, which in the nonabelian case are not unique.

On the level of algebras, the matrix A is a constructive realization of the algebra isomorphism

$$\mathcal{C}G \cong \bigoplus_i M_{d_i}(\mathcal{C}) \tag{4.7}$$

of $\mathcal{C}$-algebras, where $M_{d_i}(\mathcal{C})$ denotes the full matrix ring of $d_i \times d_i$ matrices with coefficients in $\mathcal{C}$. This decomposition is also known as *Wedderburn decomposition* of the group algebra $\mathcal{C}G$.

We remind the reader of the fact that the decomposition in (4.7) is quite familiar from the theory of error-avoiding quantum codes and noiseless subsystems [229, 230, 231, 232].

As an example of a Fourier transform, let $G = \mathbf{Z}_n = \langle x \mid x^n = 1\rangle$ be the cyclic group of order n with regular representation $\phi = 1_E \uparrow_T G$, $T = (x^0, x^1, \ldots, x^{n-1})$, and let ω_n be a primitive nth root of unity.[4] Now $\phi^A = \bigoplus_{i=0}^{n-1} \rho_i$, where $\rho_i : \; x \mapsto \omega_n^i$ and $A = \mathrm{DFT}_n = (1/\sqrt{n})[\omega_n^{ij} \mid i,j = 0\ldots n-1]$ is the (unitary) discrete Fourier transform well known from signal processing.

If A is a Fourier transform for the group G, then any fast algorithm for the multiplication with A is called a *fast Fourier transform* for G. Of course, the term *fast* depends on the complexity model chosen. Since we are primarily interested in the realization of a fast Fourier transform on a quantum computer (QFT), we first have to use the complexity measure κ, as derived in Sect. 4.2.1.

Classically, a *fast* Fourier transform is given by a factorization of the decomposition matrix A into a product of sparse matrices[5] [171, 174, 225, 233]. For a solvable group G, this factorization can be obtained recursively using the following idea. First, a normal subgroup of prime index $(G : N) = p$ is chosen. Using transitivity of induction, $\phi = 1_E \uparrow G$ is written as $(1_E \uparrow N) \uparrow G$ (note that we have the freedom to choose the transversals appropriately). Then $1_E \uparrow N$, which again is a regular representation, is decomposed (by

[4] The induction of a representation ϕ of a subgroup $H \le G$ with transversal $T = (t_1, \ldots, t_k)$ is defined by $(\phi \uparrow_T G)(g) := [\dot\phi(t_i g t_j^{-1}) \mid i,j = 1\ldots n]$, where $\dot\phi(x) := \phi(x)$ for $x \in H$ or else is the zero matrix of the appropriate size.

[5] Note that in general, sparseness of a matrix does not imply low computational complexity with respect to the complexity measure κ.

recursion), yielding a Fourier transform B for N. In the last step, A is derived from B using a recursion formula.

A Decomposition Algorithm. Following [228], we explain this procedure in more detail by first presenting two essential theorems (without proof) and then stating the actual algorithm for deriving fast Fourier transforms for solvable groups. The special tensor structure of the recursion formula mentioned above will allow us to use this algorithm as a starting point to obtain fast *quantum* Fourier transforms in the case where G is a 2-group (i. e. $|G|$ is a power of 2).

First we need Clifford's theorem, which explains the relationship between the irreducible representations of G and those of a normal subgroup N of G. Recall that G acts on the representations of N via inner conjugation: given a representation ρ of N and $t \in G$ we define $\rho^t : n \mapsto \rho(tnt^{-1})$ for $n \in N$.

Theorem 4.14 (Clifford's Theorem). *Let $N \trianglelefteq G$ be a normal subgroup of prime index p with (cyclic) transversal $T = (t^0, t^1, \ldots, t^{(p-1)})$, and denote by $\lambda_i : t \mapsto \omega_p^i$, $i = 0, \ldots, p-1$, the p irreducible representations of G arising from G/N. Assume ρ is an irreducible representation of N. Then exactly one of the two following cases applies:*

1. *$\rho \cong \rho^t$ and ρ has p pairwise inequivalent extensions to G. If $\overline{\rho}$ is one of them, then all are given by $\lambda_i \cdot \overline{\rho}$, $i = 0, \ldots, p-1$.*
2. *$\rho \ncong \rho^t$ and $\rho \uparrow_T G$ is irreducible. Furthermore, $(\rho \uparrow_T G) \downarrow N = \bigoplus_{i=0}^{p-1} \rho^{t^i}$ and*

$$(\lambda_i \cdot (\rho \uparrow_T G))^{D \otimes \mathbf{1}_d} = \rho \uparrow_T G \ , \quad D = \operatorname{diag}(1, \omega_p, \ldots, \omega_p^{(p-1)})^i \ .$$

The following theorem provides the recursion formula and was used earlier by Beth [171] to obtain fast Fourier transforms based on the tensor product as a parallel-processing model.

Theorem 4.15. *Let $N \trianglelefteq G$ be a normal subgroup of prime index p having a transversal $T = (t^0, t^1, \ldots, t^{(p-1)})$, and let ϕ be a representation of degree d of N. Suppose that A is a matrix decomposing ϕ into irreducibles, i.e. $\phi^A = \rho = \rho_1 \oplus \ldots \oplus \rho_k$, and that $\overline{\rho}$ is an extension of ρ to G. Then*

$$(\phi \uparrow_T G)^B = \bigoplus_{i=0}^{p-1} \lambda_i \cdot \overline{\rho} \ ,$$

where $\lambda_i : t \mapsto \omega_p^i$, $i = 0, \ldots, p-1$, are the p irreducible representations of G arising from the factor group G/N,

$$B = (\mathbf{1}_p \otimes A) \cdot D \cdot (\mathrm{DFT}_p \otimes \mathbf{1}_d) \ , \quad \text{and} \quad D = \bigoplus_{i=0}^{p-1} \overline{\rho}(t)^i \ .$$

If, in particular, $\overline{\rho}$ is a direct sum of irreducibles, then B is a decomposition matrix of $\phi \uparrow_T G$.

In the case of a cyclic group G the formula yields exactly the well-known Cooley–Tukey decomposition (see also Sect. 4.4.1 and [209]), in which D is usually called the *twiddle matrix*.

Assume that $N \trianglelefteq G$ is a normal subgroup of prime index p with Fourier transform A and decomposition $\phi^A = \rho = \bigoplus_{i=1}^m \rho_i$. We can reorder the ρ_i such that the first, say k, ρ_i have an extension $\overline{\rho}_i$ to G and the other ρ_i occur as sequences $\rho_i \oplus \rho_i^t \oplus \ldots \oplus \rho_i^{t^{(p-1)}}$ of inner conjugates (cf. Theorem 4.14; note that the irreducibles ρ_i, $\rho_i^{t^j}$ have the same multiplicity since ϕ is regular). In the first case the extension may be calculated by Minkwitz's formula [234]; in the latter case each sequence can be extended by $\rho_i \uparrow_T G$ (Theorem 4.14, case 2). We do not state Minkwitz's formula here, since we shall not need it in the special cases treated later on. Altogether, we obtain an extension $\overline{\rho}$ of ρ and can apply Theorem 4.15. The remaining task is to ensure that equivalent irreducibles in $\bigoplus_{i=1}^p \lambda_i \cdot \overline{\rho}$ are equal. For summands of $\overline{\rho}$ of the form $\overline{\rho}_i$ we have the result that $\lambda_j \cdot \overline{\rho}_i$ and $\overline{\rho}_i$ are inequivalent, and hence there is nothing to do. For summands of $\overline{\rho}$ of the form $\rho_i \uparrow_T G$, we conjugate $\lambda_j \cdot (\rho_i \uparrow_T G)$ onto $\rho_i \uparrow_T G$ using Theorem 4.14, case 2.

Now we are ready to formulate the recursive algorithm for constructing a fast Fourier transform for a solvable group G due to Püschel et al. [228].

Algorithm 5 *Let $N \trianglelefteq G$ be a normal subgroup of prime index p with transversal $T = (t^0, t^1, \ldots, t^{(p-1)})$. Suppose that ϕ is a regular representation of N with (fast) Fourier transform A, i.e. $\phi^A = \rho_1 \oplus \ldots \oplus \rho_k$, fulfilling $\rho_i \cong \rho_j \Rightarrow \rho_i = \rho_j$. A Fourier transform B of G with respect to the regular representation $\phi \uparrow_T G$ can be obtained as follows.*

1. *Determine a permutation matrix P that rearranges the ρ_i, $i = 1, \ldots, k$, such that the extensible ρ_i (i.e. those satisfying $\rho_i = \rho_i^t$) come first, followed by the other representations ordered into sequences of length p equivalent to $\rho_i, \rho_i^t, \ldots, \rho_i^{t^{(p-1)}}$. (Note that these sequences need to be equal to $\rho_i, \rho_i^t, \ldots, \rho_i^{t^{(p-1)}}$, which is established in the next step.)*
2. *Calculate a matrix M which is the identity on the extensibles and conjugates the sequences of length p to make them equal to $\rho_i, \rho_i^t, \ldots, \rho_i^{t^{(p-1)}}$.*
3. *Note that $A \cdot P \cdot M$ is a decomposition matrix for ϕ, too, and let $\rho = \phi^{A \cdot P \cdot M}$. Extend ρ to G summand-wise. For the extensible summands use Minkwitz's formula; the sequences $\rho_i, \rho_i^t, \ldots, \rho_i^{t^{(p-1)}}$ can be extended by $\rho_i \uparrow_T G$.*
4. *Evaluate $\overline{\rho}$ at t and build $D = \bigoplus_{i=0}^{p-1} \overline{\rho}(t)^i$.*
5. *Construct a block-diagonal matrix C with Theorem 4.14, case 2, conjugating $\bigoplus_{i=0}^{p-1} \lambda_i \cdot \overline{\rho}$ such that equivalent irreducibles are equal. C is the identity on the extended summands.*

   Result:
$$B = (\mathbf{1}_p \otimes A \cdot P \cdot M) \cdot D \cdot (\mathrm{DFT}_p \otimes \mathbf{1}_{|N|}) \cdot C \tag{4.8}$$
is a fast Fourier transform for G.

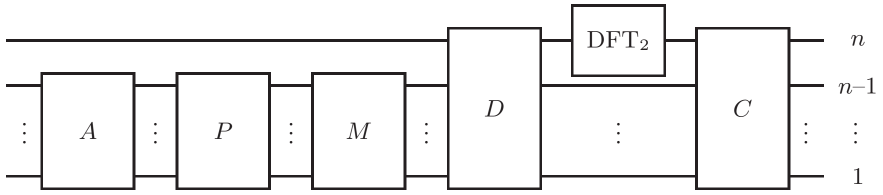

Fig. 4.13. Coarse quantum circuit visualizing Algorithm 5 for 2-groups

It is obviously possible to construct fast Fourier transforms on a classical computer for any solvable group by recursive use of this algorithm.

Since we are restricting ourselves to the case of a quantum computer consisting of qubits, i.e. two-level systems, we apply Algorithm 5 to obtain QFTs for 2-groups (i.e. $|G| = 2^n$ and thus $p = 2$). In this case the two tensor products occuring in (4.8) fit very well and yield a coarse factorization, as shown in Fig. 4.13. The lines in the figure correspond to the qubits as in Sect. 4.2.1, and a box ranging over more than one line denotes a matrix admitting no a priori factorization into a tensor product.

The remaining problem is the realization of the matrices A, P, M, D, C in terms of elementary building blocks as presented in Sect. 4.2.1. At present, the realization of these matrices remains a creative process to be performed for given (classes of) finite groups.

In [228] Algorithm 5 is applied to a class of nonabelian 2-groups, namely the 2-groups which contain a cyclic normal subgroup of index 2. These have been classified (see [235], Sect. 14.9, pp. 90–91), and for $n \geq 3$ there are exactly the following four isomorphism types:

1. the dihedral group $D_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^2 = 1,\ x^y = x^{-1} \rangle$
2. the quaternion group $Q_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^4 = 1,\ x^y = x^{-1} \rangle$
3. the group $QP_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^2 = 1,\ x^y = x^{2^{n-1}+1} \rangle$
4. the quasi-dihedral group $QD_{2^{n+1}} = \langle x, y \mid x^{2^n} = y^2 = 1,\ x^y = x^{2^{n-1}-1} \rangle$.

Observe that the extensions 1, 3 and 4 of the cyclic subgroup $\mathbf{Z}_{2^n} = \langle x \rangle$ split, i.e. the groups have the structure of a semidirect product of $\mathbf{Z}_{2^n}$ with $\mathbf{Z}_2$. The three isomorphism types correspond to the three different embeddings of $\mathbf{Z}_2 = \langle y \rangle$ into $(\mathbf{Z}_{2^n})^\times \cong \mathbf{Z}_2 \times \mathbf{Z}_{2^{n-2}}$.

In [228] quantum circuits with polylogarithmic gate complexity are given for the Fourier transforms for each of these groups. See also [236, 237] for quantum Fourier transforms for nonabelian groups.

An Example: Wreath Products. In this section we recall the definition of wreath products in general (see also [235, 238]) and, as an example, give efficient quantum Fourier transforms for a certain family of wreath products.

Definition 4.4. *Let G be a group and $H \subseteq \mathrm{S}_n$ be a subgroup of the symmetric group on n letters. The wreath product $G \wr H$ of G with H is the set*

$$\{(\varphi, h) : h \in H, \varphi : [1, \ldots, n] \to G\}$$

equipped with the multiplication

$$(\varphi_1, h_1) \cdot (\varphi_2, h_2) := (\psi, h_1 h_2)i \ ,$$

where ψ is the mapping given by $i \mapsto \varphi_1(i^{h_2})\varphi_2(i)$ for $i \in [1, \ldots, n]$.

In other words, the wreath product is isomorphic to a semidirect product of the so-called *base group* $N := G \times \ldots \times G$, which is the n-fold direct product of (independent) copies of G with H, in symbols $G \wr H = N \rtimes H$, where H operates via permutation of the direct factors of N. So we can think of the elements as n-tuples of elements from G together with a permutation τ, and multiplication is done componentwise after a suitable permutation of the first n factors:

$$(g_1, \ldots, g_n; \tau) \cdot (g'_1, \ldots, g'_n; \tau') = (g_{\tau'(1)} g'_1, \ldots, g_{\tau'(n)} g'_n; \tau\tau') \ .$$

In this section we show how to compute a Fourier transform for certain wreath products on a quantum computer. We show how the general recursive method to obtain fast Fourier transforms on a quantum computer described in [228] can be applied directly in the case of wreath products $A \wr \mathbf{Z}_2$, where A is an arbitrary abelian 2-group. The recursion of the algorithm follows the chain

$$A \wr \mathbf{Z}_2 \rhd A \times A \rhd E \ ,$$

where the second composition factor is the base group. We first want to determine the irreducible representations of $G := A \wr \mathrm{Z}_2$. Let G^* be the base group of G, i. e. $G^* = A \times A$. G^* is a normal subgroup of G of index 2. Denoting by $\mathcal{A} = \{\chi_1, \ldots, \chi_k\}$ the set of irreducible representations of A, recall that the irreducible representations of G^* are given by the set $\{\chi_i \otimes \chi_j : i, j = 1, \ldots, k\}$ of pairwise tensor products (e.g. Sect. 5.6 of [212]).

Since $G^* \lhd G$, the group G operates on the representations of G^* via inner conjugation. Because G is a semidirect product of G^* with Z_2, we can write each element $g \in G$ as $g = (a_1, a_2; \tau)$ with $a_1, a_2 \in A$ and we conclude

$$(\chi_1 \otimes \chi_2)^g = (\chi_1^{a_1} \otimes \chi_2^{a_2})^\tau = (\chi_1 \otimes \chi_2)^\tau,$$

i.e. only the factor group $G/G^* = \mathrm{Z}_2$ operates via permutation of the tensor factors. The operation of τ is to map $\chi_1 \otimes \chi_2 \mapsto \chi_2 \otimes \chi_1$.

Therefore, it is easy to determine the inertia groups (see [171, 235] for definitions) T_ρ of a representation ρ of G^*. We have to consider two cases:

(a) $\rho = \chi_i \otimes \chi_i$. Then $T_\rho = G$, since permutation of the factors leaves ρ invariant.

(b) $\rho = \chi_i \otimes \chi_j, i \neq j$. Here we have $T_\rho = G^*$.

The irreducible representations of G^* fulfilling (a) extend to representations of G, whereas the induction of a representation fulfilling (b) is irreducible. In this case the restriction of the induced representation to G^* is, by Clifford theory, equal to the direct sum $\chi_1 \otimes \chi_2 \oplus \chi_2 \otimes \chi_1$.

Example 4.7. We consider the special case $W_n := \mathbf{Z}_2^n \wr \mathbf{Z}_2$, for which the quantum Fourier transforms have an especially appealing form.

Applying the design principles for Fourier transforms described in this section (see also [171, 174, 226, 228, 249]), we obtain the circuits for DFT_{W_n} in a straightforward way. Once we have studied the extension/induction behavior of the irreducible representations of G^*, the recursive formula

$$(\mathbf{1}_2 \otimes \mathrm{DFT}_{G^*}) \cdot \bigoplus_{t \in T} \Phi(t) \cdot (\mathrm{DFT}_{\mathbf{Z}_2} \otimes \mathbf{1}_{|A|^2}) \tag{4.9}$$

provides a Fourier transform for G. Here $\Phi(t)$ denotes the extension (as a whole) of the regular representation of G^* to a representation of G [171, 226, 228]. In the case of W_n, the transform DFT_{G^*} is the Fourier transform for $\mathbf{Z}_2^{2n}$ and therefore a tensor product of $2n$ Hadamard matrices.

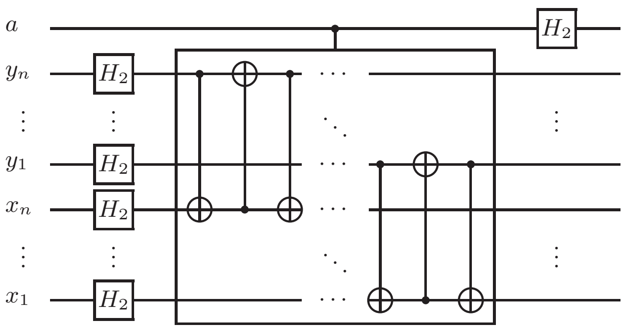

Fig. 4.14. The Fourier transform for the wreath product $\mathbf{Z}_2^n \wr \mathbf{Z}_2$

The circuits for the case of W_n are shown in Fig. 4.14. Obviously, the complexity cost of this circuit is linear in the number of qubits, since the conditional gate representing the evaluation at the transversal $\bigoplus_{t \in T} \Phi(t)$ can be realized with $3n$ Toffoli gates.

Nonabelian Hidden Subgroups. We adopt the definition of the hidden-subgroup problem given in [239]. The history of the hidden subgroup problem parallels the history of quantum computing, since the algorithms of Simon [57] and Shor [9] can be formulated in the language of hidden subgroups (see, e.g., [240] for this reduction) for certain abelian groups. In [206] exact quantum algorithms (with a running time polynomial in the number of evaluations of the given black-box function and the classical postprocessing) are given for the hidden-subgroup problem in the abelian case. For the general case we use the following definition, which should be compared with Simon's problem in Sect. 4.3.

Definition 4.5 (The Hidden Subgroup Problem). *Let G be a finite group and $f : G \to R$ a mapping from G to an arbitrary domain R fulfilling the following conditions:*

(a) The function f is given as a quantum circuit, i.e. f can be evaluated in superpositions.
(b) There exists a subgroup $U \subseteq G$ such that f takes a constant value on each of the cosets gU for $g \in G$.
(c) Furthermore, f takes different values on different cosets.

The problem is to find generators for U.

We have already seen that Simon's algorithm and Shor's algorithms for the discrete logarithm and factoring can be seen as instances of abelian hidden-subgroup problems.

The hidden-subgroup problem for nonabelian groups provides a natural generalization of these quantum algorithms. Interesting problems can be formulated as hidden-subgroup problems for nonabelian groups, e.g. the graph isomorphism problem, which is equivalent to the problem of deciding whether a graph has a nontrivial automorphism group [241]. In this case G is the symmetric group S_n acting on a given graph Γ with n vertices. To reduce the graph automorphism problem to a hidden-subgroup problem for S_n, we encode Γ into a binary string in $R := \{0,1\}^*$ and define f to be the mapping which assigns to a given permutation $\sigma \in S_n$ the graph Γ^σ. Progress in the direction of the graph isomorphism problem has been made in [242], but an efficient quantum algorithm solving the hidden-subgroup problem for S_n or the graph isomorphism problem is still not known.

In [239], the case of hidden subgroups of dihedral groups is addressed. The authors show that it is possible to solve the hidden-subgroup problem using only polynomially many queries to the black-box function f. However, the classical postprocessing takes exponential time in order to solve a nonlinear optimization problem. In [243] the hidden-subgroup problem for the wreath products $\mathbf{Z}_2^n \wr \mathbf{Z}_2$ is solved on a quantum computer, using polynomially many queries to f and efficient classical postprocessing which takes $O(n^3)$ steps. The fast quantum Fourier transforms for these groups, which have been derived in Sect. 5, have been used in this solution.

4.6.2 The Discrete Cosine Transform

In this section we address the problem of computing further signal transforms on a quantum computer. More specifically, we give a realization of the discrete cosine transforms (DCTs) of type II on a quantum computer, which is based on a well-known reduction to the computation of a discrete Fourier transform (DFT) of double length.

The DCT has numerous applications in classical signal processing, and hence, this transform might be useful in quantum computing and quantum state engineering as well. For efficient quantum signal transforms, and especially for wavelet transforms on quantum computers, see also [236, 244, 245, 246].

The discrete cosine transforms come in different flavors, varying slightly in their definitions. In this paper we restrict ourselves to the discrete cosine transform of type II ($\mathrm{DCT}_{\mathrm{II}}$) as defined below.

Recall that the $\mathrm{DCT}_{\mathrm{II}}$ is the $N \times N$ matrix defined by (see [247], p. 11)

$$\mathrm{DCT}_{\mathrm{II}} := \left(\frac{2}{N}\right)^{1/2} \left(k_i \cos \frac{i(j+1/2)\pi}{N}\right)_{i,j=0,\dots,N-1} ,$$

where $k_i = 1$ for $i = 1, \dots, N-1$ and $k_0 = 1/\sqrt{2}$.

In [247] the $\mathrm{DCT}_{\mathrm{II}}$ is shown to be a real and orthogonal (and hence unitary) transformation. Closely related is the discrete cosine transform $\mathrm{DCT}_{\mathrm{III}}$, which is defined as the transposed matrix (and hence the inverse) of $\mathrm{DCT}_{\mathrm{II}}$. Therefore, each efficient quantum circuit for the $\mathrm{DCT}_{\mathrm{III}}$ yields one for the $\mathrm{DCT}_{\mathrm{II}}$ and vice versa, since the inverse transform is obtained by reading the circuit backwards (where each elementary gate is conjugated and transposed).

Concerning the applications in classical signal processing, we should mention the well-known fact that the DCTs (of all families) are asymptotically equivalent to the Karhunen–Loève transform for signals produced by a first-order Markov process. The $\mathrm{DCT}_{\mathrm{II}}$ is also used in the JPEG image compression standard [247].

For a given (normalized) vector $|x\rangle = (x(0), \dots, x(N-1))^t$, we want to compute the matrix vector product $\mathrm{DCT}_{\mathrm{II}} \cdot |x\rangle$ on a quantum computer efficiently. Since $\mathrm{DCT}_{\mathrm{II}}$ is a unitary matrix, it has a factorization into elementary quantum gates, and we seek a factorization of *polylogarithmic* length. Note that we restrict ourselves to the case $N = 2^n$ since matrices of this size fit naturally into the tensor product structure of the Hilbert space imposed by the qubits.

Theorem 4.16. *The discrete cosine transform* $\mathrm{DCT}_{\mathrm{II}}(2^n)$ *of length* 2^n *can be computed in* $O(n^2)$ *steps using one auxiliary qubit.*

Proof. The main idea (following Chap. 4 of [247]) is to reduce the computation of $\mathrm{DCT}_{\mathrm{II}}$ to a computation of a DFT of double length.

Instead of the input vector $|x\rangle$ of length 2^n, we consider $|y\rangle$ of length 2^{n+1}, defined by

$$y(i) := \begin{cases} x(i)/\sqrt{2} & , \; i = 0, \dots, 2^n-1 \\ x(2^{n+1} - i - 1)/\sqrt{2} & , \; i = 2^n, \dots, 2^{n+1}-1 \end{cases} .$$

The input state of the quantum computer considered here, which has a register of length n carrying $|x\rangle$ and an extra qubit (which is initialized in the ground state), is $|\varphi_{in}\rangle = |0\rangle|x\rangle$ (written explicitly, this is $x(0)|00\dots0\rangle + \dots + x(2^n)|01\dots1\rangle$), which is transformed by the circuit given in Fig. 4.15 to yield $|y\rangle$.

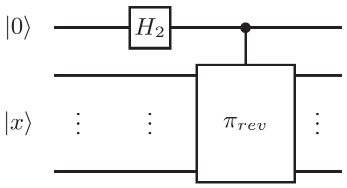

Fig. 4.15. Circuit that prepares the vector $|y\rangle$

As usual, H_2 is the Hadamard transformation and π_{rev} is the permutation obtained by performing a σ_x on each wire.

Application of $\mathrm{DFT}_{2^{n+1}} := (1/\sqrt{2^{n+1}})[\omega_{2^{n+1}}^{i\cdot j}]_{i,j=0,\dots,2^{n+1}-1}$, where $\omega_{2^{n+1}}$ denotes a primitive 2^{n+1}th root of unity, to the vector $|y\rangle$ yields the components

$$(\mathrm{DFT}_{2^{n+1}} \cdot y)(m) = \frac{1}{\sqrt{2^{n+2}}} \sum_{i=0}^{2^n-1} (\omega_{2^{n+1}}^{m\cdot i} + \omega_{2^{n+1}}^{m\cdot(2^n-i-1)})x(i) \ ,$$

which holds for $m = 0, \dots, 2^{n+1} - 1$.

Note that multiplication with $\mathrm{DFT}_{2^{n+1}}$ can be performed in $O(n^2)$ steps on a quantum computer [9, 248], which is quite contrary to the classical FFT algorithm, which requires $O(n2^n)$ arithmetic operations. However, the tensor product recursion formula [171, 249] is well suited for direct translation into efficient quantum circuits [203].

Multiplication of the vector component $y(m)$ by the phase factor $\omega_{2^{n+1}}^{m/2}$ yields

$$\omega_{2^{n+1}}^{m/2}\left(\omega_{2^{n+1}}^{mi} + \omega_{2^{n+1}}^{m(2^{n+1}-i-1)}\right) = \omega_{2^{n+1}}^{m(i+1/2)} + \omega_{2^{n+1}}^{m(2^{n+1}-i-1/2)} \ ,$$

which means that this expression is equal to $2\cos[m(i+1/2)\pi/2^n]$. The multiplication with these (relative) phase factors corresponds to a diagonal matrix $T = \mathbf{1}_2 \otimes \mathrm{diag}(\omega_{2^{n+1}}^{m/2}, m = 0, \dots, 2^n - 1)$, which can be implemented by a tensor product $T = \mathbf{1}_2 \otimes T_n \otimes \dots \otimes T_1$ of local operations, where $T_i = \mathrm{diag}(1, \omega_{2^{n+2}}^{2^{i-1}})$ for $i = 1, \dots, n$.

Looking at the state obtained so far, we see that $|\varphi_{in}\rangle$ has been mapped to $|\varphi'\rangle$, the lower 2^n components of which are given by

$$\varphi'(m) = \frac{1}{\sqrt{2^n}} \sum_{i=0}^{2^n-1} \cos[m(i+1/2)\pi/2^n]x(i) \ ,$$

where $m = 0, \dots, 2^{n+1} - 1$. Using an elementary property of the cosine, we see that the following holds for the components of this vector:

$$\varphi'(m) = -\varphi'(2^{n+1} - m), \text{ for } m = 1, \dots, 2^n - 1 ,$$

and, furthermore, $\varphi'(0) = \sum_{i=0}^{2^n-1} x(i)$ and $\varphi'(2^n) = 0$. Hence we are nearly finished, since, except for cleaning up the help register, the x register has been transformed according to $\mathrm{DCT}_{\mathrm{II}}$.

Cleaning up the help register can be accomplished by the matrix

$$\begin{pmatrix} 1 & & & & & \\ & \frac{1}{\sqrt{2}} & & & & -\frac{1}{\sqrt{2}} \\ & & \ddots & & \iddots & \\ & & \frac{1}{\sqrt{2}} & & -\frac{1}{\sqrt{2}} & \\ & & & 1 & & \\ & & \frac{1}{\sqrt{2}} & & \frac{1}{\sqrt{2}} & \\ & & \iddots & & \ddots & \\ & \frac{1}{\sqrt{2}} & & & & \frac{1}{\sqrt{2}} \end{pmatrix} = \pi^{-1} \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & & & \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & & \\ & & & & \ddots & & \\ & & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ & & & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \pi ,$$

where the permutation matrix π arranges the columns $0, \dots, 2^{n+1}$ in such a way that $(i, 2^{n+1} - i + 1)$ stand next to each other for $i = 1, \dots, 2^n - 1$. This can be achieved by a quantum circuit, used also in [228], where this permutation appeared in the reordering of irreducible representations according to the action of a dihedral group. The circuit implementing π is given in Fig. 4.16 and can be performed in $O(n^2)$ operations. Here P_n is the cyclic shift $x \mapsto x + 1 \bmod 2^n$ on the basis states, which can be implemented in $O(n^2)$ operations (see [228], Sect. 3).

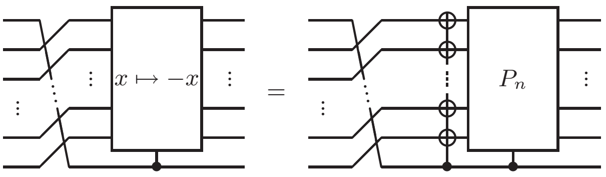

Fig. 4.16. Grouping the matrix entries pairwise via π

The matrix obtained after conjugation with π is a tensor product of the form $\mathbf{1}_{2^n} \otimes D_1$, where

$$D_1 := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} ,$$

up to a multiplication with the block diagonal matrix

$$\mathrm{diag}\Bigl(\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \mathbf{1}_{2^{n+1}-2}\Bigr) ,$$

which in turn can be implemented by an $(n-1)$-fold controlled operation, i.e. in $O(n^2)$ elementary gates [172].

Overall, we obtain the circuit for the implementation of a $\mathrm{DCT}_{\mathrm{II}}$ in $O(n^2)$ elementary gates shown in Fig. 4.17 (we have set $D_2 := D_1^{-1}$). $\square$

We close this section with the remark that it is possible to derive other decompositions of $\mathrm{DCT}_{\mathrm{II}}$ into a product of elementary quantum gates which have the advantage of being in-place, i.e. the overhead of one qubit that the realization given in the previous section needs can be saved [250].

This factorization of $\mathrm{DCT}_{\mathrm{II}}(2^n)$ of length 2^n can also be computed in $O(n^2)$ elementary operations and does not make use of auxiliary registers.

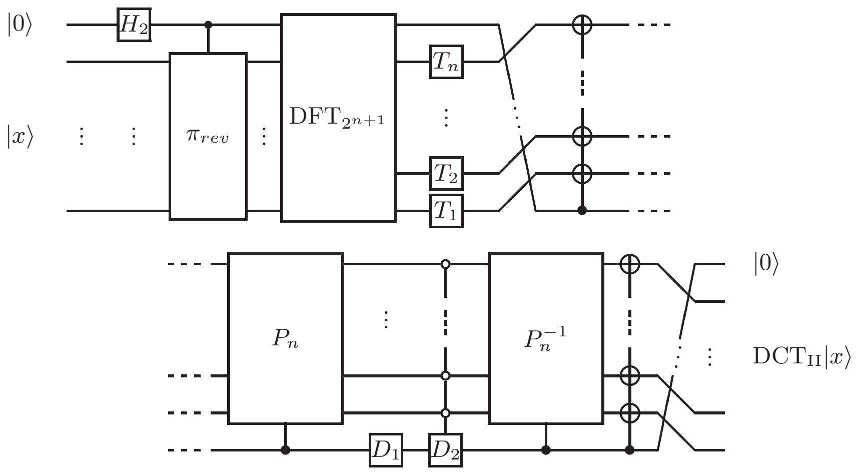

Fig. 4.17. Complete quantum circuit for $\mathrm{DCT}_{\mathrm{II}}$ using one auxiliary qubit

4.7 Quantum Error-Correcting Codes

4.7.1 Introduction

The class of quantum error-correcting codes (QECCs) provides a master example that represents multiple aspects of the features of quantum algorithms. They are quantum algorithms per se, showing the applicability of the algorithmic concepts described so far to an intrinsically important area of quantum computation and quantum information theory, namely, the stabilization of quantum states.

This is achieved by methods which are based on signal-processing techniques at two levels:

- the Hilbert space of the quantum system itself
- the discrete vector space of the underlying combinatorial configurations.

The latter structure of finite-geometric spaces gives, furthermore, a deep insight into the features of entanglement, as the error-correcting codes known from classical coding theory share a particular feature, namely that of generating sets of codewords with maximal minimum distance by constructing suitable geometric configurations. On the other hand, these configurations nicely display the features of maximally entangled quantum states.

4.7.2 Background

We follow the presentation in [104], Chap. 13 for the background on the classical theory. A classical error-correcting code (ECC) consists of a set of binary words

$$\mathbf{x} = (x_0, \dots, x_{n-1})$$

of length n, where each "bit" $x_i \in \{0,1\}$ can take a value $x_i \in GF(2)$ in the finite field of two elements. With this elementary notion, the codewords can be treated as vectors of the n-dimensional $GF(2)$ vector space $GF(2)^n$, where the set of codewords is usually assumed to form a k-dimensional subspace $\mathcal{C} \le GF(2)^n$. This can be obtained canonically as the range $\mathrm{im}(G)$ of a $GF(2)$ linear mapping $G : GF(2)^k \to GF(2)^n$ of the so-called encoder matrix, which maps k-bit messages onto n-bit codewords, adding a redundancy of $r = n - k$ bits in the r so-called parity check bits.

The characteristic parameters of such a linear code $\mathcal{C}$ are the rate $R = k/n$ and the minimum Hamming weight

$$w_{\mathrm{H}} := \min\{\mathrm{wgt}_{\mathrm{H}}(\mathbf{c}) : \mathbf{c} \in \mathcal{C}, \mathbf{c} \neq 0\} .$$

The Hamming weight for a vector $\mathbf{x} = (x_0, \dots, x_{n-1}) \in GF(2)^n$ is defined by $\mathrm{wgt}_{\mathrm{H}}(\mathbf{x}) := |\mathrm{supp}(\mathbf{x})|$, where $\mathrm{supp}(\mathbf{x}) := \{i \in [0, \dots, n-1] : x_i \neq 0\}$.

It may be noted that the Hamming distance $d_{\mathrm{H}}(\mathbf{u}, \mathbf{v}) := \#\{i \in [0, \dots, n-1] : u_i \neq v_i\}$, which measures the number of bit flips necessary to change $\mathbf{v}$ into $\mathbf{u}$, is given by $d_{\mathrm{H}}(\mathbf{u}, \mathbf{v}) = \mathrm{wgt}_{\mathrm{H}}(\mathbf{u} - \mathbf{v})$. Thus, since the code $\mathcal{C}$ is assumed to be a linear subspace of $GF(2)^n$, the equality for the minimum distance $d_{\mathrm{H}}(\mathcal{C}) = \min_{\mathbf{u} \neq \mathbf{v}} d_{\mathrm{H}}(\mathbf{u}, \mathbf{v}) = w_{\mathrm{H}}$ is easily derived.

In constructing error-correcting codes, besides solving the parametric optimization problem of maximizing the rate of $\mathcal{C}$ and its minimum weight, it is a challenge to provide an efficient decoding algorithm at the same time. A decoder for $\mathcal{C}$ can in principle be built as follows:

Lemma 4.3. *The dual code $\mathcal{C}^\perp$ of $\mathcal{C}$, which is the $(n-k)$-dimensional subspace of $GF(2)^n$ orthogonal to $\mathcal{C}$ with respect to the canonical $GF(2)$ bilinear pairing (see also Sect. 4.4.2), is generated by any matrix $H : GF(2)^{n-k} \to GF(2)^n$ with* $\mathrm{im}(H) = \mathcal{C}^\perp$.

Obviously $\mathcal{C} = \mathrm{Ker}(H^{\mathrm{t}})$, the kernel of the transpose of H, and $G \cdot H^{\mathrm{t}} = 0$. □

For the sake of simplicity we assume that each codeword $\mathbf{c} \in \mathcal{C}$ is transmitted through a binary symmetric channel (BSC) (Fig. 4.18), which is the master model of a discrete memoryless channel [251]. A BSC is assumed to add the error vector $\mathbf{e} \in GF(2)^n$ independently of the codeword so that a noisy vector $\mathbf{u} = \mathbf{c} + \mathbf{e}$ is received with probability $q^{n-\mathrm{wgt}(\mathbf{e})} \cdot p^{\mathrm{wgt}(\mathbf{e})}$, where $q = 1 - p$.

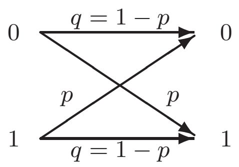

Fig. 4.18. Binary symmetric channel with parameter p

From the *syndrome*

$$\mathbf{s} := \mathbf{u} \cdot H^t = \mathbf{c} \cdot H^t + \mathbf{e} \cdot H^t = \mathbf{e} \cdot H^t$$

we see that the error pattern determines an affine subspace of $GF(2)^n$, namely a coset of $\mathcal{C}$ in $GF(2)^n$. In order to allow error correction, the syndrome has to be linked to the unknown error vector $\mathbf{e}$ uniquely, usually according to the maximum-likelihood decoding principle. Obviously, for the BSC this can be achieved by finding the unique codeword $\mathbf{c}$ that minimizes the Hamming distance to $\mathbf{u}$. For combinatorial reasons this is possible if

$$\mathrm{wgt}(\mathbf{e}) \leq \left\lfloor \frac{w_H - 1}{2} \right\rfloor .$$

Thus a code with minimum Hamming weight $w_{\mathrm{H}} = 2t + 1$ is said to correct t errors per codeword.

4.7.3 A Classic Code

An example which today can be considered a classic in the field of science and technology is the application and design of (first-order) Reed–Müller codes $RM(1, m)$. This is not merely because they were a decisive piece of discrete mathematics in producing the first pictures from the surface of Mars in the early 1970s after the landing of Mariner 9 on 19 January 1972.

The description of their geometric construction and the method of error detection/correction of the corresponding wave functions, which we have taken from [104], Chap. 13, is quite similar to the concept of quantum error-correcting codes. In this section we describe Reed–Müller codes in a natural geometrical setting (cf. [103, 104, 252]).

In the *first-order Reed–Müller code* $RM(1,m)$, the codewords $\boldsymbol{f} \in GF(2)^{2^m}$ are $GF(2)$-linear combinations of class functions of the index-2 subgroups $H < \mathbf{Z}_2^m$ and their cosets. From this notion, a natural transform into the orthonormal basis of Walsh–Hadamard functions [253, 254] is given by the characters of $\mathbf{Z}_2^m$, i.e. the Hadamard transformation H_{2^m} (see also Sect. 4.2.1 and [255]). By these means, the $GF(2)$ vector

$$\boldsymbol{f} = (f(\boldsymbol{u}))_{\boldsymbol{u} \in \mathbf{Z}_2^m} \in GF(2)^{2^m}$$

is converted into the real (row) vector

$$\boldsymbol{F} = \left((-1)^{f(\boldsymbol{u})}\right)_{\boldsymbol{u} \in \mathbf{Z}_2^m}$$

by replacing an entry 0 in $\boldsymbol{f}$ by an entry 1 in $\boldsymbol{F}$ and an entry 1 in $\boldsymbol{f}$ by an entry -1 in $\boldsymbol{F}$. Modulation into a wavefunction $F(t)$ is then achieved by transmitting the step function in the interval $[0, 2^m - 1]$ defined by

$$F(t) = \sum_{i=0}^{2^m-1} F_{\mathrm{Binary}(i)} 1_{[i,i+1]}(t) \ .$$

Note that for the first Hadamard coefficient $\hat{F}_0$ of $F(t)$, which is given by

$$\widehat{F}_0 = \int_0^{2^m} F(t)\,\mathrm{dt} \ ,$$

the following identity holds:

$$\widehat{F}_0 = 2^m - 2\ \mathrm{wgt}(\boldsymbol{f}) \ .$$

This provides a beautiful maximum-likelihood decoding device similar to that needed for quantum codes.

We shall illustrate this with the example of the first-order Reed–Müller code $RM(1,3)$.

The codewords of the first-order Reed–Müller code $RM(1,m)$ with $m = 3$, of length 8, can be regarded as incidence vectors of special subsets of points of $AG(3,2)$ (see Fig. 4.19). The following table of Boolean functions (see Sect. 4.2.2) $\boldsymbol{f_i}(\boldsymbol{x}_3, \boldsymbol{x}_2, \boldsymbol{x}_1) = \boldsymbol{1} \oplus \boldsymbol{x}_i$ of incidence vectors,

$$\begin{aligned}
\boldsymbol{1} &= (1,1,1,1,1,1,1,1) \text{ corresponding to the whole space} \ , \\
\boldsymbol{f_1} &= (1,0,1,0,1,0,1,0) \text{ corresponding to the 2-subspace } (*,*,0) \ , \\
\boldsymbol{f_2} &= (1,1,0,0,1,1,0,0) \text{ corresponding to the 2-subspace } (*,0,*) \ , \\
\boldsymbol{f_3} &= (1,1,1,1,0,0,0,0) \text{ corresponding to the 2-subspace } (0,*,*) \ ,
\end{aligned} \tag{4.10}$$

thus defines an $(m+1) \times 2^m$ generator matrix G. Its range is the following set of 16 codewords:

$$\begin{array}{llll}
(0,0,0,0,0,0,0,0) & (1,1,1,1,1,1,1,1) & (0,0,0,0,1,1,1,1) & (1,1,1,1,0,0,0,0) \\
(0,0,1,1,0,0,1,1) & (1,1,0,0,1,1,0,0) & (0,1,0,1,0,1,0,1) & (1,0,1,0,1,0,1,0) \\
(0,0,1,1,1,1,0,0) & (1,1,0,0,0,0,1,1) & (0,1,0,1,1,0,1,0) & (1,0,1,0,0,1,0,1) \\
(0,1,1,0,0,1,1,0) & (1,0,0,1,1,0,0,1) & (0,1,1,0,1,0,0,1) & (1,0,0,1,0,1,1,0) \ .
\end{array}$$

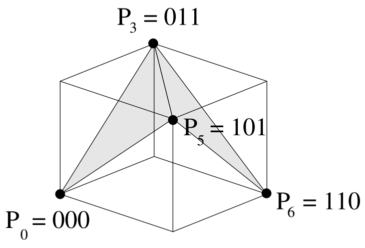

Fig. 4.19. Reed–Müller code $RM(1,3)$ interpreted as a 2-flat in $AG(3,2)$

Thus the subset $S = \{P_0, P_3, P_5, P_6\}$ determines the incidence vector $(1,0,0,1,0,1,1,0)$, which is the codeword $\boldsymbol{f}_1 + \boldsymbol{f}_2 + \boldsymbol{f}_3$ of $RM(1,3)$. The geometric interpretation of this codeword as an $(m-1)$-flat in $AG(3,2)$ is shown in Fig. 4.19.

With respect to G, the signal function $\boldsymbol{F}$ transmitted (see Fig. 4.20) thus corresponds to the $(m+1)$-bit input vector $(0,1,1,1)$.

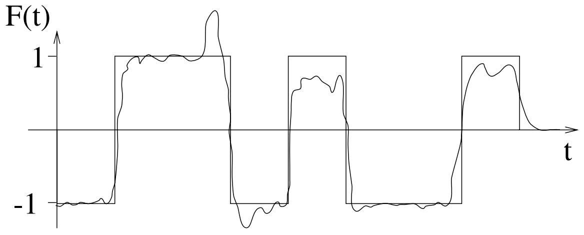

Fig. 4.20. Transmitted signal corresponding to the codeword $\boldsymbol{f}_0 + \boldsymbol{f}_1 + \boldsymbol{f}_2 = (1,0,0,1,0,1,1,0)$ with and without noise

In the more general case of Reed–Müller codes $RM(1,m)$ and their modulated signal functions, noise $e(t)$ is added to the signal function by the channel during transmission, so that the receiver will detect only a signal

$$\psi(t) = F(t) + e(t) \ .$$

The behavior of the $RM(1,m)$ demodulator–decoder device can be sketched on the basis of the underlying geometry as follows.

Transformation of the received signal $\psi(t)$ into the orthonormal basis of Walsh–Hadamard functions $W_{\boldsymbol{u}}(t)$ of order m is achieved by computing the 2^m scalar products

$$\widehat{\psi}(\boldsymbol{u}) = \langle W_{\boldsymbol{u}} \mid \psi \rangle \ ,$$

where, for $\boldsymbol{u} \in GF(2)^m$, $W_{\boldsymbol{u}}(t)$ is given by

$$W_{\boldsymbol{u}}(t) = \sum_{j=1}^{2^m} (-1)^{\boldsymbol{u}\cdot \mathrm{Binary}(j-1)} \cdot 1_{[j-1,j]}(t) \ ,$$

corresponding to the uth row of the Hadamard matrix H_{2^n}.

As the received signal function is represented by a sampling vector $\boldsymbol{\psi} = \boldsymbol{F} + \boldsymbol{e}$, after the Walsh–Hadamard transformation one obtains $\widehat{\boldsymbol{\psi}} = \boldsymbol{\psi} \cdot H_{2^m} = \boldsymbol{F} H_{2^m} + \boldsymbol{e} H_{2^m}$. We estimate a maximum-likelihood signal function as follows. From $\widehat{\boldsymbol{F}} = \boldsymbol{F} \cdot H_{2^m}$, where

$$\widehat{F}(\boldsymbol{u}) = \sum_{\boldsymbol{v}} (-1)^{\boldsymbol{u}\cdot\boldsymbol{v}} F(\boldsymbol{v}) = \sum_{\boldsymbol{v}} (-1)^{\boldsymbol{u}\cdot\boldsymbol{v} + f(\boldsymbol{v}) \bmod 2},$$

we obtain the identity

$$|2^m - \widehat{F}(\boldsymbol{u})| = 2\,\mathrm{wgt}(\boldsymbol{u}^{\perp} \oplus \boldsymbol{f}) \ ,$$

since $(\boldsymbol{u}\cdot\boldsymbol{v})_{\boldsymbol{v}\in GF(2)^m}$ is the incidence vector $\boldsymbol{u}^{\perp}$ of the hyperspace of $AG(2,m)$ orthogonal to the vector $\boldsymbol{u} \in GF(2)^m$. As the $RM(1,m)$ codes consist of all incidence vectors of such hyperspaces or their cosets (i.e. the complement), a minimum distance decoder is realized as shown in Fig. 4.21.[6]

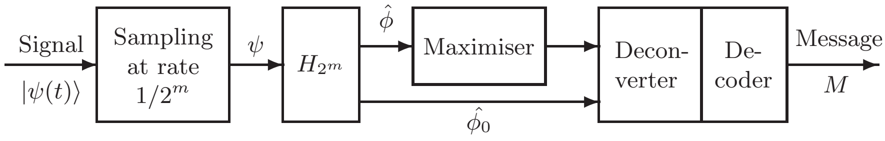

Fig. 4.21. Minimum-distance decoder for Reed–Müller codes (cf. [104], Chap. 13)

Here the maximizer computes

$$\boldsymbol{x} \in GF(2)^m \quad \text{such that } |\psi(\boldsymbol{x})| = \max_{\boldsymbol{u}\in GF(2)^m} |\psi(\boldsymbol{u})| \ .$$

With the overall $\pm$ parity given by the sign $(\widehat{\psi}_0) = (-1)^{\varepsilon(\psi)}$, the deconverter produces a most likely codeword $\boldsymbol{f} = \varepsilon(\psi) \cdot \mathbf{1} + \boldsymbol{x}^{\perp}$. If the enumeration of the generating hyperspaces $\boldsymbol{u_i}$ (cf. (4.10)) is chosen suitably, the decoder therefore reproduces the $(m+1)$-bit message $\boldsymbol{n} = (\varepsilon, x_1, \ldots, x_m)$, which is a maximum-likelihood estimator of the word originally encoded into $\boldsymbol{f} = \boldsymbol{M} \cdot G$. The reader is urged to compare this decoding algorithm with the quantum decoding algorithm described in Sect. 4.7.4.

[6] With the fast Hadamard transform algorithm (see Sect. 4.2.1), this decoder requires $O(m2^m)$ computational steps. Note that this is one of the earliest communication applications of generalized FFT algorithms (cf. [171]).

4.7.4 Quantum Channels and Codes

One of the basic features described so far in this contribution has been the use of quantum mechanical properties (entanglement, superposition) to speed up the solution of classical problems. Most notably, this basic idea of quantum computing has been present in the complex of hidden-subgroup problems (see Sect. 4.3, 4.5.2 and 5), where the problem was to identity an unknown subgroup of a given group out of exponentially many candidates, given a superposition over a coset of the unknown subgroup. In this section we shall take the dual point of view: we construct states which are simultaneous eigenstates of a suitably chosen subgroup of a fixed error group and have the additional property that an element of an unknown coset of this group – this models an error which happens to the states – can be identified and also corrected.

To construct these states, we rely on the powerful theory of classical ECC [184] introduced in the preceding sections. In what follows, we shall introduce the class of binary QECCs, the so-called CSS codes, referring to the elaborate article [255] by Beth and Grassl. These codes were independently discovered by Calderbank and Shor [82] and Steane [62].

Quantum Channels. Before describing the construction of these codes, we shall loosely describe the similarities and differences between a classical binary symmetric channel (BSC) (see Sect. 4.7.2, Fig. 4.18) and a quantum channel (QC).

Much as in the idealized case of a BSC, where vectors $\mathbf{c} \in GF(2)^n$ are transmitted, we shall consider the QC to be a carrier of kets $|\psi\rangle \in \mathcal{H}_{2^n} = \mathcal{C}^{2^n}$ spanned by the basis kets $|\mathbf{x}\rangle$, where $\mathbf{x} \in GF(2)^n$. In this system, so-called error operations, generated by local errors as bit-flip errors, and sign-flip errors can occur in superposition. Similarly to the case of a BSC, where the error group is isomorphic to $(GF(2)^n, \oplus) = \langle (e_1, \ldots, e_n) : e_i \in \{0,1\}, i = 1, \ldots, n\rangle$, in QC the error group

$$E = \langle e_1 \otimes \ldots \otimes e_n : e_i \in \{\mathrm{id}, \sigma_x, \sigma_z\}, i = 1, \ldots, n\}\rangle ,$$

generated by the local bit flips and phase flips, represents all possible error operators in the quantum channel by the transition diagram described below.

Initially, the input wavefunction $|\psi\rangle$ will interact with the environment via $|\psi\rangle \mapsto |\psi\rangle|\epsilon\rangle$ through a modulator; within the channel, this waveform evolves under the error group according to the channel characteristics,

$$|\psi\rangle|\epsilon\rangle \quad \xrightarrow[\text{error}]{\text{channel -}} \quad \sum_{\gamma\in E} (\gamma|\psi\rangle)|\epsilon_\gamma\rangle \ , \tag{4.11}$$

whereas, in the "environment", ancillae tacitly contain probability amplitudes for the occurrence of group elements.

Much as in the BSC, where only errors $\mathbf{e} \in GF(2)^n$ with a given maximal weight $\mathrm{wgt}(\mathbf{e}) \le t$ are allowed or assumed, in the QC the sum of the received ket

$$|\mu\rangle = \sum_{\substack{\gamma\in E\\ \mathrm{wgt}(\gamma)\leq t}} \gamma|\psi\rangle|\epsilon_\gamma\rangle \tag{4.12}$$

will range only over those group elements γ which are products of at most t local (single-bit) errors. Much in accordance with the classical case, for $\gamma = e_1 \otimes \ldots \otimes e_n \in E$ we define $\mathrm{wgt}(\gamma)$ to be the number of occurrences of σ_x and σ_z needed to generate γ.

In order to protect quantum states against errors of this kind in a quantum channel, a so-called Pauli channel, a quantum error-correcting code must be constructed to map original quantum states into certain "protected" orthogonal subspaces, so that errors of the type of (4.12) cannot in practice damage the original state seriously. For this purpose, the theory of classical codes for the BSC can be successfully applied, as we describe below.

Quantum Codes. The basic principle is to consider encoded quantum states which are superpositions of basis vectors belonging to classical codes, e.g.

$$|\mathcal{C}\rangle := \frac{1}{\sqrt{|\mathcal{C}|}}\sum_{\mathbf{c}\in\mathcal{C}}|\mathbf{c}\rangle\ .$$

First we note a surprising fact expressing an old result of error-correcting codes directly in the language of quantum theory.

Lemma 4.4 (MacWilliams). *Let $\mathcal{C} < GF(2)^n$ be an error-correcting code with its dual as usual. Let A be the elementary abelian group $(GF(2)^n, \oplus)$. Then DFT_A is the Hadamard transformation (see Sect. 4.4.2) $H_{2^n} := H_2 \otimes \ldots \otimes H_2$ (n factors). For each error vector $\mathbf{e} \in GF(2)^n$,*

$$H_{2^n}|\mathcal{C}\rangle = H_{2^n}\left(\frac{1}{\sqrt{|\mathcal{C}|}}\sum_{\mathbf{c}\in\mathcal{C}}|\mathbf{c}\oplus\mathbf{e}\rangle\right) = \frac{1}{\sqrt{|\mathcal{C}^\perp|}}\sum_{\mathbf{c}\in\mathcal{C}^\perp}(-1)^{\mathbf{c}\cdot\mathbf{e}}|\mathbf{c}\rangle\ . \tag{4.13}$$

Proof. This result is due to the identity

$$\mathrm{DFT}_A\left(\frac{1}{\sqrt{|U|}}\sum_{\mathbf{c}\in U}|\mathbf{c}\oplus\mathbf{e}\rangle\right) = \frac{1}{\sqrt{|U^\perp|}}\sum_{\mathbf{c}\in U^\perp}\beta(\mathbf{c},\mathbf{e})|\mathbf{c}\rangle\ , \tag{4.14}$$

which holds for all subgroups U of an abelian group A (see Sect. 4.4.2 and 4.4.3). □

The lemma says that any bit-flip error applied to the state $|\mathcal{C}\rangle$ will give a state whose support $\mathcal{C}^\perp$ is translation invariant, the shift being expressed only in the phases of the elements of $\mathcal{C}^\perp$. Dually, a phase-flip error in $|\mathcal{C}\rangle$ will occur as a bit-flip error in $|\mathcal{C}^\perp\rangle$.

From this, we deduce the following basic coding principle ([255], p. 462): given a classical binary linear (n, k) code $\mathcal{C}$ of length n and dimension k, the states of the related quantum code are given by

$$|\psi_{\boldsymbol{w}_i}\rangle = \frac{1}{\sqrt{|\mathcal{C}|}} \sum_{\boldsymbol{c}\in\mathcal{C}} (-1)^{\boldsymbol{c}\cdot\boldsymbol{w}_i} |\boldsymbol{c}\rangle \ , \tag{4.15}$$

with suitable $\boldsymbol{w}_i \in GF(2)^n$ encoding the ith basis ket of the initial state to be protected. In addition to the choice of the classical code $\mathcal{C}$, for the construction of a binary quantum code a subset $\mathcal{W} \subseteq GF(2)^n/\mathcal{C}^\perp$ has to be given to define the vectors $\boldsymbol{w}_i$.

Example 4.8. Let $\mathcal{C} := \{(0,0,0),(1,1,1)\} \subseteq GF(2)^3$ be the dual of the Reed–Müller code $\mathcal{R} = RM(1,3)$ shown in Fig. 4.19. Here $\mathcal{W} := GF(2)^3/\mathcal{C}^\perp$ provides an appropriate choice,

$$\boldsymbol{w}_0 = (0,0,0),\ \boldsymbol{w}_1 = (1,1,1) \ ,$$

for the following encoding:

$$\begin{aligned} |\mathbf{0}\rangle &\mapsto |\psi_{\boldsymbol{w}_0}\rangle = |0,0,0\rangle + |1,1,1\rangle \ , \\ |\mathbf{1}\rangle &\mapsto |\psi_{\boldsymbol{w}_1}\rangle = |0,0,0\rangle - |1,1,1\rangle \ . \end{aligned} \tag{4.16}$$

Note that the state $|0,0,0\rangle + |1,1,1\rangle$ is the maximally entangled GHZ state. We remark that the GHZ state is, up to local unitary transformations, the unique maximally entangled state [256]. The "protected" subspace of code vectors is, by definition,

$$\mathcal{H}_0 = \{|\psi\rangle = \alpha|\psi_{\boldsymbol{w}_0}\rangle\beta|\psi_{\boldsymbol{w}_1}\rangle \mid |\alpha|^2 + |\beta|^2 = 1\} \ .$$

Since $\mathcal{C}$ is a one-error-correcting binary code, the quantum code is endowed with this property with respect to single bit-flip errors, i.e. it is protected against the error operators

$$\varepsilon_3 = id \otimes id \otimes \sigma_x,\ \varepsilon_2 = id \otimes \sigma_x \otimes id,\ \varepsilon_1 = \sigma_x \otimes id \otimes id \in \mathcal{U}(8) \ .$$

Obviously, the subspaces $\mathcal{H}_0$ and $\mathcal{H}_i = \varepsilon_i\mathcal{H}_0$ $(i = 1, 2, 3)$ are mutually orthogonal, so that

$$\mathcal{H}^{2^3} = \bigoplus_{i=0}^{3} \mathcal{H}_i$$

is the direct sum of the four orthogonal spaces

$$\begin{aligned} \mathcal{H}_0 &= \langle |0,0,0\rangle + |1,1,1\rangle, |0,0,0\rangle - |1,1,1\rangle \rangle \ , \\ \mathcal{H}_1 &= \langle |1,0,0\rangle + |0,1,1\rangle, |1,0,0\rangle - |0,1,1\rangle \rangle \ , \\ \mathcal{H}_2 &= \langle |0,1,0\rangle + |1,0,1\rangle, |0,1,0\rangle - |1,0,1\rangle \rangle \ , \\ \mathcal{H}_3 &= \langle |0,0,1\rangle + |1,1,0\rangle, |0,0,1\rangle - |1,1,0\rangle \rangle \ . \end{aligned}$$

Thus, for any linear combination $\mathcal{E}$ of these single bit-flip errors ϵ_i, the encoded state $|\psi\rangle = \alpha|\psi_{\boldsymbol{w}_0}\rangle + \beta|\psi_{\boldsymbol{w}_1}\rangle$ is represented by

$$\mathcal{E}|\psi\rangle = \sum_{i=0}^{3} \lambda_i(\alpha\epsilon_i|\psi_{\boldsymbol{w}_0}\rangle + \beta\epsilon_i|\psi_{\boldsymbol{w}_1}\rangle) \ ,$$

as a direct sum of four orthogonal vectors, each being "proportional" to $|\psi\rangle$. So, by a measurement, i.e. a random projection onto any of the spaces $\mathcal{H}_i$, or by applying a conditional gate $U = \mathrm{diag}(\epsilon_i^\dagger : i = 0, \ldots, 3)$, the original state can be reconstituted, thus correcting up to one bit-flip error.

But note that if, instead of $\mathcal{C}$, the code $\mathcal{R} = \mathcal{C}^\perp$ had been selected, this code construction could not have been successful, as $\mathcal{R} = RM(1,3)$ can detect one error but not correct it. It can be seen (see [255], p. 463) that the corresponding quantum code inherits this property of *detecting* one phase-error but not being capable of *correcting* it.

Motivated by and starting from this example and the properties and problems derived from it, we now quote the following theorem, which provides a method to obtain quantum codes from classical codes.

Theorem 4.17 (CSS Codes). *Suppose $\mathcal{C}_1, \mathcal{C}_2 \subseteq GF(2)^n$ are classical binary codes with parameters (n_1, k_1, d_1) and (n_2, k_2, d_2), respectively, fulfilling the additional requirement $\mathcal{C}_1^\perp \subseteq \mathcal{C}_2$. Let $W := \{\mathbf{w}_i : i = 1, \ldots, [\mathcal{C}_2 : \mathcal{C}_1^\perp]\}$ be a system of representatives of the cosets $\mathcal{C}_2^\perp/\mathcal{C}_1$. Then the set of states*

$$|\phi_{\mathbf{w}_i}\rangle := \frac{1}{\sqrt{|\mathcal{C}_1|}} \sum_{\mathbf{c}\in\mathcal{C}_1} (-1)^{\mathbf{c}\cdot\mathbf{w}_i}|\mathbf{c}\rangle$$

forms a quantum code $\mathcal{Q}$ which can correct $(d_1-1)/2$ bit errors and $(d_2-1)/2$ phase errors.

In practice, we have the following corollary in the case of weakly self-dual codes $\mathcal{C} \subseteq \mathcal{C}^\perp$, which were shown to be as good asymptotically in [82]:

Theorem 4.18. *Let $\mathcal{C}$ be a weakly self-dual binary code with dual distance d. Then the corresponding quantum code is capable of correcting up to $(d-1)/2$ errors.*

The construction in this theorem can be made more general, as Beth and Grassl have shown in [255]:

Theorem 4.19. *Let $\mathcal{C}^\perp$ be a weakly self-dual binary code. If for*

$$\mathcal{M}_0 := \{\boldsymbol{w} \in GF(2)^n/\mathcal{C}^\perp \mid d_{\mathrm{H}}(\mathcal{C}^\perp, \mathcal{C}^\perp + \boldsymbol{w}) \leq t\}$$

the following condition,

$$\forall \boldsymbol{w}_i, \boldsymbol{w}_j : i \neq j \Rightarrow \mathcal{M}_0 \cap (\mathcal{M}_0 + (\boldsymbol{w}_i - \boldsymbol{w}_j)) = \emptyset \ , \tag{4.17}$$

is satisfied, the quantum code can correct t errors, i.e. any error operator $\varepsilon \in E$ where the total number of positions exposed to bit flips or sign flips is at most t.

This leads to the following decoding algorithm.

Algorithm 6 (Quantum Decoding Algorithm) *Let $|\phi\rangle$ be encoded by a QECC according to Theorems 4.17–4.19. The received vector $\mathcal{E}|\phi\rangle$ will be decoded and reconstructed by the following steps:*

- *Perform a measurement to determine the bit-flip errors, i.e. project onto the code space $\mathcal{H}_{\mathcal{C}}$ or one of its orthogonal images $\mathcal{H}_{\mathcal{C}+\boldsymbol{e}}$ under a bit-flip error $\boldsymbol{e}$.*
- *Correct this bit-flip error ("subtract" $\boldsymbol{e}$) by applying the corresponding tensor product of σ_x operators.*
- *Perform a Hadamard transformation.*
- *Perform a measurement to determine the sign-flip errors which corresponds to a bit-flip error in the actual bases.*
- *Correct this error.*
- *Reencode the final state.*

This decoding algorithm is easily understood from the point of view of binary codes, which have been designed as general constructions for the BSC (see Sect. 4.7.2). The reader should also observe the stunning analogy between this quantum decoder and the Green-machine decoder described in Sect. 4.7.3, Fig. 4.21.

4.8 Conclusions

We have presented an introduction to a computational model of quantum computers from a computer science point of view. Discrete Fourier transforms have been introduced as important subroutines used in several quantum algorithms. Throughout, unitary transformations which can be implemented in terms of elementary gates using a quantum circuit of polylogarithmic size have been of special interest; they yield an exponential speedup compared with the classical situation in many cases.

The quantum algorithms presented here exploit the fundamental principles of interference, superposition and entanglement that quantum physics offers. We have explored these principles in various algorithms, ranging from Shor's algorithms to algorithms for quantum error-correcting codes.