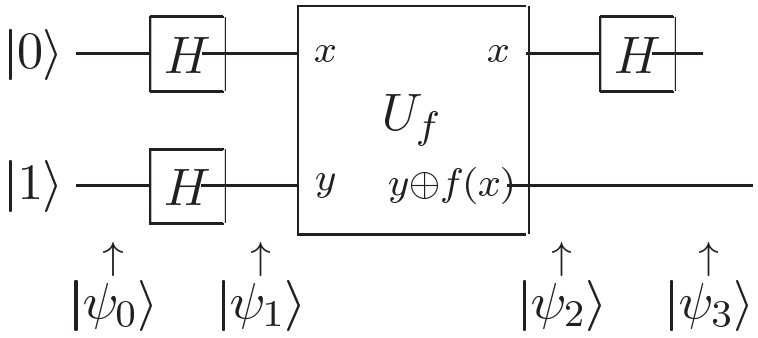

Figure III.22: Quantum circuit for Deutsch algorithm. [fig. from Nielsen & Chuang (2010)]

D Quantum algorithms

D.1 Deutsch-Jozsa algorithm

D.1.a Deutsch's algorithm

In this section you will encounter your first example of a quantum algorithm that can compute faster than a classical algorithm for the same problem. This is a simplified version of Deutsch's original algorithm, which shows how it is possible to extract global information about a function by using quantum parallelism and interference (Fig. III.22).[8]

Suppose we have a function $f : \mathbf{2} \to \mathbf{2}$, as in Sec. C.5. The goal is to determine whether $f(0) = f(1)$ with a *single* function evaluation. This is not a very interesting problem (since there are only four such functions), but it is a warmup for the Deutsch-Jozsa algorithm. Simple as it is, it could be expensive to decide on a classical computer. For example, suppose $f(0) =$ the billionth bit of π and $f(1) =$ the billionth bit of e. Then the problem is to decide if the billionth bits of π and e are the same. It is mathematically simple, but computationally complex.

To see how we might solve this problem, suppose we have a quantum gate array U_f for f; that is, $U_f|x\rangle|y\rangle = |x\rangle|y \oplus f(x)\rangle$. In particular, $U_f|x\rangle|0\rangle =$

[8] This is the 1998 improvement by Cleve et al. to Deutsch's 1985 algorithm (Nielsen & Chuang, 2010, p. 59).

$|x\rangle|f(x)\rangle$ and $U_f|x\rangle|1\rangle = |x\rangle|\neg f(x)\rangle$. Usually we set $y = 0$ to get the result $|f(x)\rangle$, but here you will see an application in which we want $y = 1$.

Now consider the result of applying U_f to $|x\rangle$ in the data register and to the superposition $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ in the target register.

$$U_f|x\rangle|-\rangle = \frac{1}{\sqrt{2}}|x\rangle|f(x)\rangle - \frac{1}{\sqrt{2}}|x\rangle|\neg f(x)\rangle = \frac{1}{\sqrt{2}}|x\rangle[|f(x)\rangle - |\neg f(x)\rangle].$$

Now the rightmost square bracket is $|0\rangle - |1\rangle$ if $f(x) = 0$ or $|1\rangle - |0\rangle$ if $f(x) = 1$. Therefore, we can write

$$U_f|x\rangle|-\rangle = \frac{1}{\sqrt{2}}|x\rangle(-)^{f(x)}(|0\rangle - |1\rangle) = (-)^{f(x)}|x\rangle|-\rangle. \tag{III.21}$$

[Here, $(-)^x$ is an abbreviation for $(-1)^x$ when we want to emphasize that the sign is all that matters.] Since $U_f|x\rangle|-\rangle = (-)^{f(x)}|x\rangle|-\rangle$, the result of applying it to an equal superposition of $x = 0$ and $x = 1$ is:

$$\frac{1}{\sqrt{2}}\sum_{x\in\mathbf{2}} U_f|x\rangle|-\rangle = \frac{1}{\sqrt{2}}\sum_{x\in\mathbf{2}}(-)^{f(x)}|x\rangle|-\rangle.$$

If f is a constant function, then $f(0) = f(1)$, and the summation is $\pm\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|-\rangle = \pm|+\rangle|-\rangle$ because both components have the same sign.. On the other hand, if $f(0) \neq f(1)$, then the summation is $\pm\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|-\rangle = \pm|-\rangle|-\rangle$ because the components have opposite signs. That is, a constant function gives the $|0\rangle$ and $|1\rangle$ components of the data qubit the same phase, and otherwise gives them the opposite phase. Therefore, we can determine whether the function is constant or not by measuring the first qubit in the sign basis; we get $|+\rangle$ if $f(0) = f(1)$ and $|-\rangle$ otherwise. With this background, we can state Deutsch's algorithm.

algorithm Deutsch:

Initial state: Begin with the qubits $|\psi_0\rangle \stackrel{\text{def}}{=} |01\rangle$.

Superposition: Transform it to a pair of superpositions

$$|\psi_1\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |+-\rangle. \tag{III.22}$$

by a pair of Hadamard gates. Recall that $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle$.

Function application: Next apply U_f to $|\psi_1\rangle = |+-\rangle$. As we've seen, $U_f|x\rangle|0\rangle = |x\rangle|0 \oplus f(x)\rangle = |x\rangle|f(x)\rangle$, and $U_f|x\rangle|1\rangle = |x\rangle|1 \oplus f(x)\rangle = |x\rangle|\neg f(x)\rangle$. Therefore, expand Eq. III.22 and apply U_f:

$$\begin{aligned}
|\psi_2\rangle &\stackrel{\text{def}}{=} U_f|\psi_1\rangle \\
&= U_f\left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\right] \\
&= \frac{1}{2}[U_f|00\rangle - U_f|01\rangle + U_f|10\rangle - U_f|11\rangle] \\
&= \frac{1}{2}[|0, f(0)\rangle - |0, \neg f(0)\rangle + |1, f(1)\rangle - |1, \neg f(1)\rangle]
\end{aligned}$$

There are two cases: $f(0) = f(1)$ and $f(0) \neq f(1)$.

Equal (constant function): If $f(0) = f(1)$, then

$$\begin{aligned}
|\psi_2\rangle &= \frac{1}{2}[|0, f(0)\rangle - |0, \neg f(0)\rangle + |1, f(0)\rangle - |1, \neg f(0)\rangle] \\
&= \frac{1}{2}[|0\rangle(|f(0)\rangle - |\neg f(0)\rangle) + |1\rangle(|f(0)\rangle - |\neg f(0)\rangle)] \\
&= \frac{1}{2}(|0\rangle + |1\rangle)(|f(0)\rangle - |\neg f(0)\rangle) \\
&= \pm\frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\
&= \pm\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|-\rangle \\
&= |+-\rangle.
\end{aligned}$$

The last line applies because global phase (including $\pm$) doesn't matter.

Unequal (balanced function): If $f(0) \neq f(1)$, then

$$|\psi_2\rangle = \frac{1}{2}[|0, f(0)\rangle - |0, \neg f(0)\rangle + |1, \neg f(0)\rangle - |1, f(0)\rangle]$$

$$
\begin{aligned}
&= \frac{1}{2}[|0\rangle(|f(0)\rangle - |\neg f(0)\rangle) + |1\rangle(|\neg f(0)\rangle - |f(0)\rangle)] \\
&= \frac{1}{2}[|0\rangle(|f(0)\rangle - |\neg f(0)\rangle) - |1\rangle(|f(0)\rangle - |\neg f(0)\rangle)] \\
&= \frac{1}{2}(|0\rangle - |1\rangle)(|f(0)\rangle - |\neg f(0)\rangle) \\
&= \pm\frac{1}{2}(|0\rangle - |1\rangle)(|0\rangle - |1\rangle) \\
&= \pm\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)|-\rangle \\
&= |--\rangle
\end{aligned}
$$

Clearly we can discriminate between the two cases by measuring the first qubit in the sign basis. In particular, note that in the unequal case, the $|1\rangle$ component has the opposite phase from the $|0\rangle$ component.

Measurement: Therefore we can determine whether $f(0) = f(1)$ or not by measuring the first bit of $|\psi_2\rangle$ in the sign basis, which we can do with the Hadamard gate (recall $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$):

$$
\begin{aligned}
|\psi_3\rangle &\stackrel{\text{def}}{=} (H \otimes I)|\psi_2\rangle \\
&= \begin{cases} \pm|0\rangle|-\rangle, & \text{if } f(0) = f(1) \\ \pm|1\rangle|-\rangle, & \text{if } f(0) \neq f(1) \end{cases} \\
&= \pm|f(0) \oplus f(1)\rangle|-\rangle.
\end{aligned}
$$

□

Notice that the information we need is in the *data* register, not the target register. This technique is called *phase kick-back* (i.e., kicked back into the phase of the data register).

In conclusion, we can determine whether or not $f(0) = f(1)$ with a *single evaluation* of f, which is quite remarkable. In effect, we are evaluating f on a superposition of $|0\rangle$ and $|1\rangle$ and determining how the results interfere with each other. As a result we get a definite (not probabilistic) determination of a global property with a single evaluation.

This is a clear example where a quantum computer can do something faster than a classical computer. However, note that U_f has to uncompute

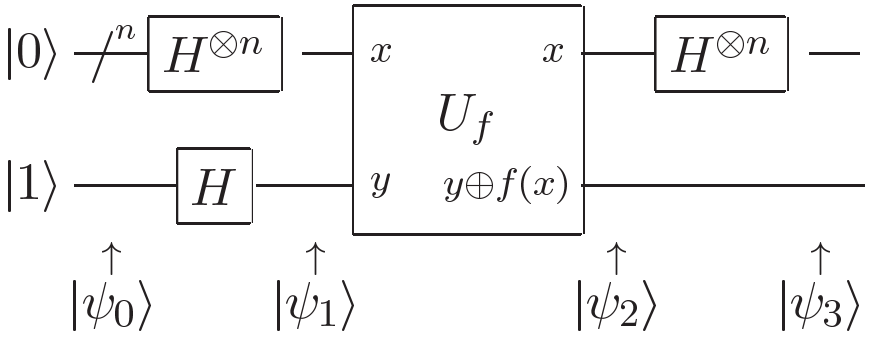

Figure III.23: Quantum circuit for Deutsch-Jozsa algorithm. [fig. from NC]

f, which takes as much time as computing it, but we will see other cases (Deutsch-Jozsa) where the speedup is much more than $2\times$.

D.1.b The Deutsch-Jozsa algorithm

The Deutsch-Jozsa algorithm is a generalization of the Deutsch algorithm to n bits, which they published it in 1992; here we present the improved version of Nielsen & Chuang (2010, p. 59).

This is the problem: Suppose we are given an unknown function $f : \mathbf{2}^n \to \mathbf{2}$ in the form of a unitary transform $U_f \in \mathcal{L}(\mathcal{H}^{n+1}, \mathcal{H})$ (Fig. III.23). We are told only that f is either constant or *balanced*, which means that it is 0 on half its domain and 1 on the other half. Our task is to determine into which class the given f falls.

Consider first the classical situation. We can try different input bit strings $\mathbf{x}$. We might (if we're lucky) discover after the second query of f that it is not constant. But we might require as many as $2^n/2 + 1$ queries to answer the question. So we're facing $\mathcal{O}(2^{n-1})$ function evaluations.

algorithm Deutsch-Jozsa:

Initial state: As in the Deutsch algorithm, prepare the initial state $|\psi_0\rangle \stackrel{\text{def}}{=} |0\rangle^{\otimes n}|1\rangle$.

Superposition: Use the Walsh-Hadamard transformation to create a superposition of all possible inputs:

$$|\psi_1\rangle \stackrel{\text{def}}{=} (H^{\otimes n} \otimes H)|\psi_0\rangle = \sum_{\mathbf{x}\in\mathbf{2}^n} \frac{1}{\sqrt{2^n}}|\mathbf{x}, -\rangle.$$

Claim: Similarly to the single qubit case (Eq. III.21), we can see that $U_f|\mathbf{x}, -\rangle = (-)^{f(\mathbf{x})}|\mathbf{x}\rangle|-\rangle$, where $(-)^n$ is an abbreviation for $(-1)^n$. From the definition of $|-\rangle$ and U_f, $U_f|\mathbf{x}, -\rangle = |\mathbf{x}\rangle\frac{1}{\sqrt{2}}(|f(\mathbf{x})\rangle - |\neg f(\mathbf{x})\rangle)$. Since $f(\mathbf{x}) \in \mathbf{2}$, $\frac{1}{\sqrt{2}}(|f(\mathbf{x})\rangle - |\neg f(\mathbf{x})\rangle) = |-\rangle$ if $f(\mathbf{x}) = 0$, and it $= -|-\rangle$ if $f(\mathbf{x}) = 1$. This establishes the claim.

Function application: Therefore, you can see that:

$$|\psi_2\rangle \stackrel{\text{def}}{=} U_f|\psi_1\rangle = \sum_{\mathbf{x}\in\mathbf{2}^n} \frac{1}{\sqrt{2^n}}(-)^{f(\mathbf{x})}|\mathbf{x}\rangle|-\rangle. \tag{III.23}$$

In the case of a constant function, all the components of the data state have the same phase, otherwise they do not.

To see how we can make use of this information, let's consider the state in more detail. For a *single* bit you can show (Exer. III.46):

$$H|x\rangle = \frac{1}{\sqrt{2}}(|0\rangle + (-)^x|1\rangle) = \frac{1}{\sqrt{2}}\sum_{z\in\mathbf{2}}(-)^{xz}|z\rangle = \sum_{z\in\mathbf{2}}\frac{1}{\sqrt{2}}(-)^{xz}|z\rangle.$$

(This is just another way of writing $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.) Therefore, the general formula for the Walsh transform of n bits is:

$$\begin{aligned} H^{\otimes n}|x_1, x_2, \ldots, x_n\rangle &= \frac{1}{\sqrt{2^n}}\sum_{z_1,\ldots,z_n\in\mathbf{2}}(-)^{x_1z_1+\cdots+x_nz_n}|z_1, z_2, \ldots, z_n\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{\mathbf{z}\in\mathbf{2}^n}(-)^{\mathbf{x}\cdot\mathbf{z}}|\mathbf{z}\rangle, \end{aligned} \tag{III.24}$$

where $\mathbf{x}\cdot\mathbf{z}$ is the bitwise inner product. (It doesn't matter if you do addition or $\oplus$ since only the parity of the result is significant.) *Remember this formula!* Combining this and the result in Eq. III.23,

$$|\psi_3\rangle \stackrel{\text{def}}{=} (H^{\otimes n} \otimes I)|\psi_2\rangle = \sum_{\mathbf{z}\in\mathbf{2}^n}\sum_{\mathbf{x}\in\mathbf{2}^n}\frac{1}{2^n}(-)^{\mathbf{x}\cdot\mathbf{z}+f(\mathbf{x})}|\mathbf{z}\rangle|-\rangle.$$

Measurement: Consider the first n qubits and the amplitude of one particular basis state, $\mathbf{z} = |\mathbf{0}\rangle = |0\rangle^{\otimes n}$, which we separate from the rest of the summation:

$$|\psi_3\rangle = \sum_{\mathbf{x}\in\mathbf{2}^n} \frac{1}{2^n}(-)^{f(\mathbf{x})}|\mathbf{0}\rangle|-\rangle + \sum_{\mathbf{z}\in\mathbf{2}^n-\{\mathbf{0}\}} \sum_{\mathbf{x}\in\mathbf{2}^n} \frac{1}{2^n}(-)^{\mathbf{x}\cdot\mathbf{z}+f(\mathbf{x})}|\mathbf{z}\rangle|-\rangle$$

Hence, the amplitude of $|0\rangle^{\otimes n}$, the all-$|0\rangle$ qubit string, is $\sum_{\mathbf{x}\in\mathbf{2}^n} \frac{1}{2^n}(-)^{f(\mathbf{x})}$. Recall how in the basic Deutsch algorithm we got a sum of signs (either all the same or not) for all the function evaluations. The result is similar here, but we have 2^n values rather than just two. We now have two cases:

Constant function: If the function is constant, then all the exponents of -1 will be the same (either all 0 or all 1), and so the amplitude will be ± 1. Therefore all the other amplitudes are 0 and any measurement must yield 0 for all the qubits (since only $|0\rangle^{\otimes n}$ has nonzero amplitude).

Balanced function: If the function is not constant then (*ex hypothesi*) it is balanced, but more specifically, if it is balanced, then there must be an equal number of $+1$ and -1 contributions to the amplitude of $|0\rangle^{\otimes n}$, so its amplitude is 0. Therefore, when we measure the state, at least one qubit must be nonzero (since the all-0s state has amplitude 0).
□

The *good news* about the Deutsch-Jozsa algorithm is that with one quantum function evaluation we have got a result that would require between 2 and $\mathcal{O}(2^{n-1})$ classical function evaluations (exponential speedup!). The *bad news* is that the algorithm has no known applications! Even if it were useful, however, the problem could be solved probabilistically on a classical computer with only a few evaluations of f: for an error probability of ϵ, it takes $\mathcal{O}(\log \epsilon^{-1})$ function evaluations. However, it illustrates principles of quantum computing that can be used in more useful algorithms.

D.2 Simon's algorithm

Simon's algorithm was first presented in 1994 and can be found in Simon, D. (1997), "On the power of quantum computation," *SIAM Journ. Computing*, 26 (5), pp. 1474–83.[9] For breaking RSA we will see that its useful to know the *period* of a function: that r such that $f(x+r) = f(x)$. Simon's problem is a warmup for this.

Simon's Problem: Suppose we are given an unknown function $f : \mathbf{2}^n \to \mathbf{2}^n$ and we are told that it is *two-to-one*. This means $f(\mathbf{x}) = f(\mathbf{y})$ iff $\mathbf{x} \oplus \mathbf{y} = \mathbf{r}$ for some fixed $\mathbf{r} \in \mathbf{2}^n$. The vector $\mathbf{r}$ can be considered the *period* of f, since $f(\mathbf{x} \oplus \mathbf{r}) = f(\mathbf{x})$. The problem is to determine the period $\mathbf{r}$ of a given unknown f.

Consider first the classical solution. Since we don't know anything about f, the best we can do is evaluate it on random inputs. If we are ever lucky enough to find $\mathbf{x}$ and $\mathbf{x}'$ such that $f(\mathbf{x}) = f(\mathbf{x}')$, then we have our answer, $\mathbf{r} = \mathbf{x} \oplus \mathbf{x}'$. After testing m values, you will have eliminated about $m(m-1)/2$ possible $\mathbf{r}$ vectors (namely, $\mathbf{x} \oplus \mathbf{x}'$ for every pair of these m vectors). You will be done when $m^2 \approx 2^n$. Therefore, on the average you need to do $2^{n/2}$ function evaluations, which is exponential in the size of the input. For $n = 100$, it would require about $2^{50} \approx 10^{15}$ evaluations. "At 10 million calls per second it would take about three years" (Mermin, 2007, p. 55). We will see that a quantum computer can determine $\mathbf{r}$ with high probability ($> 1 - 10^{-6}$) in about 120 evaluations. At 10 million calls per second, this would take about 12 microseconds!

algorithm Simon:

Input superposition: As before, start by using the Walsh-Hadamard transform to create a superposition of all possible inputs:

$$|\psi_1\rangle \stackrel{\text{def}}{=} H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{2^{n/2}} \sum_{\mathbf{x} \in \mathbf{2}^n} |\mathbf{x}\rangle.$$

[9] The following presentation follows Mermin's *Quantum Computer Science* (Mermin, 2007, §2.5, pp. 55–8).

Function evaluation: Suppose that U_f is the quantum gate array implementing f and recall $U_f|\mathbf{x}\rangle|\mathbf{y}\rangle = |\mathbf{x}\rangle|\mathbf{y} \oplus f(\mathbf{x})\rangle$. Therefore:

$$|\psi_2\rangle \stackrel{\text{def}}{=} U_f|\psi_1\rangle|0\rangle^{\otimes n} = \frac{1}{2^{n/2}} \sum_{\mathbf{x}\in\mathbf{2}^n} |\mathbf{x}\rangle|f(\mathbf{x})\rangle.$$

Therefore we have an equal superposition of corresponding input-output values.

Output measurement: Measure the output register (in the computational basis) to obtain some $|\mathbf{z}\rangle$. Since the function is two-to-one, the projection will have a superposition of two inputs:

$$\frac{1}{\sqrt{2}}(|\mathbf{x}_0\rangle + |\mathbf{x}_0 \oplus \mathbf{r}\rangle)|\mathbf{z}\rangle,$$

where $f(\mathbf{x}_0) = \mathbf{z} = f(\mathbf{x}_0 \oplus \mathbf{r})$. The information we need is contained in the input register,

$$|\psi_3\rangle \stackrel{\text{def}}{=} \frac{1}{\sqrt{2}}(|\mathbf{x}_0\rangle + |\mathbf{x}_0 \oplus \mathbf{r}\rangle),$$

but it cannot be extracted directly. If we measure it, we will get either $\mathbf{x}_0$ or $\mathbf{x}_0 \oplus \mathbf{r}$, but not both, and we need both to get $\mathbf{r}$. (We cannot make two copies, due to the no-cloning theorem.)

Suppose we apply the Walsh-Hadamard transform to this superposition:

$$\begin{aligned} H^{\otimes n}|\psi_3\rangle &= H^{\otimes n}\frac{1}{\sqrt{2}}(|\mathbf{x}_0\rangle + |\mathbf{x}_0 \oplus \mathbf{r}\rangle) \\ &= \frac{1}{\sqrt{2}}(H^{\otimes n}|\mathbf{x}_0\rangle + H^{\otimes n}|\mathbf{x}_0 \oplus \mathbf{r}\rangle). \end{aligned}$$

Now, recall (D.1.b, p. 129) that

$$H^{\otimes n}|\mathbf{x}\rangle = \frac{1}{2^{n/2}} \sum_{\mathbf{y}\in\mathbf{2}^n} (-1)^{\mathbf{x}\cdot\mathbf{y}}|\mathbf{y}\rangle.$$

(This is the general expression for the Walsh transform of a bit string. The phase depends on the number of common 1-bits.) Therefore,

$$\begin{aligned} H^{\otimes n}|\psi_3\rangle &= \frac{1}{\sqrt{2}}\left[\frac{1}{2^{n/2}} \sum_{\mathbf{y}\in\mathbf{2}^n} (-1)^{\mathbf{x}_0\cdot\mathbf{y}}|\mathbf{y}\rangle + \frac{1}{2^{n/2}} \sum_{\mathbf{y}\in\mathbf{2}^n} (-1)^{(\mathbf{x}_0+\mathbf{r})\cdot\mathbf{y}}|\mathbf{y}\rangle\right] \\ &= \frac{1}{2^{(n+1)/2}} \sum_{\mathbf{y}\in\mathbf{2}^n} \left[(-1)^{\mathbf{x}_0\cdot\mathbf{y}} + (-1)^{(\mathbf{x}_0+\mathbf{r})\cdot\mathbf{y}}\right]|\mathbf{y}\rangle. \end{aligned}$$

Note that

$$(-1)^{(\mathbf{x}_0+\mathbf{r})\cdot\mathbf{y}} = (-1)^{\mathbf{x}_0\cdot\mathbf{y}}(-1)^{\mathbf{r}\cdot\mathbf{y}}.$$

Therefore, if $\mathbf{r}\cdot\mathbf{y} = 1$, then the bracketed expression is 0 (since the terms have opposite sign and cancel). However, if $\mathbf{r}\cdot\mathbf{y} = 0$, then the bracketed expression is $2(-1)^{\mathbf{x}_0\cdot\mathbf{y}}$ (since they don't cancel). Hence the result of the Walsh-Hadamard transform is

$$|\psi_4\rangle = H^{\otimes n}|\psi_3\rangle = \frac{1}{2^{(n-1)/2}} \sum_{\mathbf{y} \text{ s.t. } \mathbf{r}\cdot\mathbf{y}=0} (-1)^{\mathbf{x}_0\cdot\mathbf{y}}|\mathbf{y}\rangle.$$

Measurement: Measuring the input register (in the computational basis) will collapse it with equal probability into a state $|\mathbf{y}^{(1)}\rangle$ such that $\mathbf{r}\cdot\mathbf{y}^{(1)} = 0$.

First equation: Since we know $\mathbf{y}^{(1)}$, this gives us some information about $\mathbf{r}$, expressed in the equation:

$$y_1^{(1)}r_1 + y_2^{(1)}r_2 + \cdots + y_n^{(1)}r_n = 0 \pmod 2.$$

Iteration: The quantum computation can be repeated, producing a series of bit strings $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \ldots$ such that $\mathbf{y}^{(k)}\cdot\mathbf{r} = 0$. From them we can build up a system of n linearly-independent equations and solve for $\mathbf{r}$. (If you get a linearly-dependent equation, you have to try again.) Note that each quantum step (involving one evaluation of f) produces an equation (except in the unlikely case $\mathbf{y}^{(k)} = 0$ or that it's linearly dependent), and therefore determines one of the bits in terms of the other bits. That is, each iteration reduced the candidates for $\mathbf{r}$ by approximately one-half.
□

A mathematical analysis (Mermin, 2007, App. G) shows that with $n+m$ iterations the probability of having enough information to determine $\mathbf{r}$ is $> 1 - \frac{1}{2^{m+1}}$. "Thus the odds are more than a million to one that with $n + 20$ invocations of $\mathbf{U}_f$ we will learn $[\mathbf{r}]$, no matter how large n may be."

(Mermin, 2007, p. 57) Note that the "extra" evaluations are independent of n. Therefore Simon's problem can be solved in *linear* time on a quantum computer, but requires *exponential* time on a classical computer.

D.3 Shor's algorithm

> If computers that you build are quantum,
> Then spies everywhere will all want 'em.
> Our codes will all fail,
> And they'll read our email,
> Till we get crypto that's quantum, and daunt 'em.
> — Jennifer and Peter Shor (Nielsen & Chuang, 2010, p. 216)

The widely used RSA public-key cryptography system is based on the difficulty of factoring large numbers.[10] The best classical algorithms are nearly exponential in the size of the input, $m = \ln M$. Specifically, the best current (2006) algorithm (the *number field sieve algorithm*) runs in time $e^{\mathcal{O}(m^{1/3}\ln^{2/3} m)}$. This is subexponential but very inefficient. Shor's quantum algorithm is bounded error-probability quantum polynomial time (BQP), specifically, $\mathcal{O}(m^3)$. Shor's algorithm was invented in 1994, inspired by Simon's algorithm.

Shor's algorithm reduces factoring to finding the period of a function. The connection between factoring and period finding can be understood as follows. Assume you are trying to factor M. Suppose you can find x such that $x^2 = 1 \pmod M$. Then $x^2 - 1 = 0 \pmod M$. Therefore $(x+1)(x-1) = 0 \pmod M$. Therefore both $x+1$ and $x-1$ have common factors with M (except in the trivial case $x = 1$, and so long as neither is a multiple of M). Now pick an a that is coprime (relatively prime) to M. If $a^r = 1 \pmod M$ and r happens to be even, we're done (since we can find a factor of M as explained above). (The smallest such r is called the *order* of a.) This r is the period of $a^x \pmod M$, since $a^{x+r} = a^x a^r = a^x \pmod M$.

In summary, if we can find the order of an appropriate a and it is even, then we can probably factor the number. To accomplish this, we need to find the period of $a^x \pmod M$, which can be determined through a Fourier transform.

Like the classical Fourier transform, the *quantum Fourier transform* puts all the amplitude of the function into multiples of the frequency (reciprocal period). Therefore, measuring the state yields the period with high probability.

[10] These section is based primarily on Rieffel & Polak (2000).

D.3.a Quantum Fourier transform

Before explaining Shor's algorithm, it's necessary to explain the quantum Fourier transform, and to do so it's helpful to begin with a review of the classical Fourier transform.

Let f be a function defined on $[0, 2\pi)$. We know it can be represented as a Fourier series,

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty}(a_k \cos kx + b_k \sin kx) = \frac{A_0}{2} + \sum_{k=1}^{\infty} A_k \cos(kx + \phi_k),$$

where $k = 0, 1, 2, \ldots$ represents the overtone series (natural number multiples of the fundamental frequency). You know also that the Fourier transform can be represented in the *ciscoid* (cosine + i sine) basis, where we define $u_k(x) \stackrel{\text{def}}{=} \text{cis}(-kx) = e^{-ikx}$. (The "$-$" sign is irrelevant, but will be convenient later.) The u_k are orthogonal but not normalized, so we divide them by 2π, since $\int_0^{2\pi} \cos^2 x + \sin^2 x \; \mathrm{d}x = 2\pi$. The Fourier series in this basis is $f(x) = \sum_{k=-\infty}^{\infty} \hat{f}_k \, \text{cis}(-kx)$. The Fourier coefficients are given by $\hat{f}_k = \frac{1}{2\pi}\langle u_k \mid f\rangle = \frac{1}{2\pi}\int_0^{2\pi} e^{ikx} f(x)\mathrm{d}x$. They give the amplitude and phase of the component signals u_k.

For the *discrete Fourier transform* (DFT) we suppose that f is represented by N samples, $f_j \stackrel{\text{def}}{=} f(x_j)$, where $x_j = 2\pi\frac{j}{N}$, with $j \in \mathbf{N} \stackrel{\text{def}}{=} \{0, 1, \ldots, N-1\}$. Let $\mathbf{f} = (f_0, \ldots, f_{N-1})^{\mathrm{T}}$. Note that the x_j are the $1/N$ segments of a circle. (Realistically, N is big.)

Likewise each of the basis functions is represented by a vector of N samples: $\mathbf{u}_k \stackrel{\text{def}}{=} (u_{k,0}, \ldots, u_{k,N-1})^{\mathrm{T}}$. Thus we have a matrix of all the basis samples:

$$u_{kj} \stackrel{\text{def}}{=} \text{cis}(-kx_j) = e^{-2\pi ikj/N}, \; j \in \mathbf{N}.$$

In $e^{-2\pi ikj/N}$, note that $2\pi i$ represents a full cycle, k is the overtone, and j/N represents the fraction of a full cycle.

Recall that every complex number has N principal N^{th}-roots, and in particular the number 1 (unity) has N principal N^{th}-roots. Notice that N samples of the fundamental period correspond to the N principal N^{th}-roots of unity, that is, ω^j where (for a particular N) $\omega = e^{2\pi i/N}$. Hence, $u_{kj} = \omega^{-kj}$. That is, $\mathbf{u}_k = (\omega^{-k\cdot 0}, \omega^{-k\cdot 1} \ldots, \omega^{-k\cdot(N-1)})^{\mathrm{T}}$. It is easy to show that the vectors $\mathbf{u}_k$ are orthogonal, and in fact that $\mathbf{u}_k/\sqrt{N}$ are ON (exercise). Therefore, $\mathbf{f}$

can be represented by a Fourier series,

$$\mathbf{f} = \frac{1}{\sqrt{N}} \sum_{k \in \mathbf{N}} \hat{f}_k \mathbf{u}_k = \frac{1}{N} \sum_{k \in \mathbf{N}} (\mathbf{u}_k^\dagger \mathbf{f}) \mathbf{u}_k.$$

Define the *discrete Fourier transform* of the vector $\mathbf{f}$, $\hat{\mathbf{f}} = \mathrm{F}\mathbf{f}$, to be the vector of Fourier coefficients, $\hat{f}_k = \mathbf{u}_k^\dagger \mathbf{f}/\sqrt{N}$. Determine F as follows:

$$\hat{\mathbf{f}} = \begin{pmatrix} \hat{f}_0 \\ \hat{f}_1 \\ \vdots \\ \hat{f}_{N-1} \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} \mathbf{u}_0^\dagger \mathbf{f} \\ \mathbf{u}_1^\dagger \mathbf{f} \\ \vdots \\ \mathbf{u}_{N-1}^\dagger \mathbf{f} \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} \mathbf{u}_0^\dagger \\ \mathbf{u}_1^\dagger \\ \vdots \\ \mathbf{u}_{N-1}^\dagger \end{pmatrix} \mathbf{f}.$$

Therefore let

$$\mathrm{F} \stackrel{\text{def}}{=} \frac{1}{\sqrt{N}} \begin{pmatrix} \mathbf{u}_0^\dagger \\ \mathbf{u}_1^\dagger \\ \vdots \\ \mathbf{u}_{N-1}^\dagger \end{pmatrix} = \frac{1}{\sqrt{N}} \begin{pmatrix} \omega^{0\cdot 0} & \omega^{0\cdot 1} & \cdots & \omega^{0\cdot(N-1)} \\ \omega^{1\cdot 0} & \omega^{1\cdot 1} & \cdots & \omega^{1\cdot(N-1)} \\ \omega^{2\cdot 0} & \omega^{2\cdot 1} & \cdots & \omega^{2\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{(N-1)\cdot 0} & \omega^{(N-1)\cdot 1} & \cdots & \omega^{(N-1)\cdot(N-1)} \end{pmatrix}. \tag{III.25}$$

That is, $\mathrm{F}_{kj} = \overline{u_{kj}}/\sqrt{N} = \omega^{kj}/\sqrt{N}$ for $k, j \in \mathbf{N}$. Note that the "$-$" signs in the complex exponentials were eliminated by the conjugate transpose. F is unitary transformation (exercise).

The *fast Fourier transform* (FFT) reduces the number of operations required from $\mathcal{O}(N^2)$ to $\mathcal{O}(N \log N)$.[11] It does this with a recursive algorithm that avoids recomputing values. However, it is restricted to powers of two, $N = 2^n$.

The *quantum Fourier transform* (QFT) is even faster, $\mathcal{O}(\log^2 N)$, that is, $\mathcal{O}(n^2)$. However, because the spectrum is encoded in the amplitudes of the quantum state, we cannot get them all. Like the FFT, the QFT is restricted to powers of two, $N = 2^n$. The QFT transforms the amplitudes of a quantum state:

$$U_{\mathrm{QFT}} \sum_{j \in \mathbf{N}} f_j |j\rangle = \sum_{k \in \mathbf{N}} \hat{f}_k |k\rangle,$$

[11] The FFT is $\mathcal{O}(N \log N)$, but $N > M^2 = e^{2m}$. Therefore the FFT is $\mathcal{O}(M^2 \log M^2) = \mathcal{O}(M^2 \log M) = \mathcal{O}(m e^{2m})$

where $\hat{\mathbf{f}} \stackrel{\text{def}}{=} \mathrm{F}\mathbf{f}$.

Suppose f has period r, and suppose that the period evenly divides the number of samples, $r \mid N$. Then all the amplitude of $\hat{f}$ should be at multiples of its fundamental frequency, N/r. If $r \not| N$, then the amplitude will be concentrated *near* multiples of N/r. The approximation is improved by using larger n.

The FFT (and QFT) are implemented in terms of additions and multiplications by various roots of unity (powers of ω). In QFT, these are phase shifts. In fact, the QFT can be implemented with $n(n+1)/2$ gates of two types: (1) One is H_j, the Hadamard transformation of the jth qubit. (2) The other is a controlled phase-shift $S_{j,k}$, which uses qubit x_j to control whether it does a particular phase shift on the $|1\rangle$ component of qubit x_k. That is, $S_{j,k}|x_j x_k\rangle \mapsto |x_j x'_k\rangle$ is defined by

$$S_{j,k} \stackrel{\text{def}}{=} |00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| + e^{i\theta_{k-j}}|11\rangle\langle 11|,$$

where $\theta_{k-j} = \pi/2^{k-j}$. That is, the phase shift depends on the indices j and k.

It can be shown that the QFT can be defined:[12]

$$U_{\mathrm{QFT}} = \prod_{j=0}^{n-1} H_j \prod_{k=j+1}^{n-1} S_{j,k}.$$

This is $\mathcal{O}(n^2)$ gates.

D.3.b Shor's algorithm step by step

Shor's algorithm depends on many results from number theory, which are outside of the scope of this course. Since this is not a course in cryptography or number theory, I will just illustrate the ideas.

algorithm Shor:

[12] See Rieffel & Polak (2000) for this, with a detailed explanation in Nielsen & Chuang (2010, §5.1, pp. 517–21).

Input: Suppose we are factoring M (and $M = 21$ will be used for concrete examples, but of course the goal is to factor very large numbers). Let $m \stackrel{\text{def}}{=} \lceil \lg M \rceil = 5$ in the case $M = 21$.

Step 1: Pick a random number $a < M$. If a and M are not coprime (relatively prime), we are done. (Euclid's algorithm is $\mathcal{O}(m^2) = \mathcal{O}(\log^2 M)$.) For our example, suppose we pick $a = 11$, which is relatively prime with 21.

Modular exponentiation: Let $g(x) \stackrel{\text{def}}{=} a^x \pmod M$, for $x \in \mathbf{M} \stackrel{\text{def}}{=} \{0, 1, \ldots, M-1\}$. This takes $\mathcal{O}(m^3)$ gates and is the most complex part of the algorithm! (Reversible circuits typically use m^3 gates for m qubits.) In our example case, $g(x) = 11^x \pmod{21}$, so

$$g(x) = \underbrace{1, 11, 16, 8, 4, 2,}_{\text{period}} 1, 11, 16, 8, 4, \ldots$$

In order to get a good QFT approximation, pick n such that $M^2 \leq 2^n < 2M^2$. Let $N \stackrel{\text{def}}{=} 2^n$. Equivalently, pick n such that $2 \lg M \leq n < 2 \lg M + 1$, that is, roughly twice as many qubits as in M. Note that although the number of samples is $N = 2^n$, we need only n qubits (thanks to the tensor product). This is where quantum computation gets its speedup over classical computation; M is very large, so $N > M^2$ is extremely large. The QFT computes all these in parallel. For our example $M = 21$, and so we pick $n = 9$ for $N = 512$ since $441 \leq 512 < 882$. Therefore $m = 5$.

Step 2 (quantum parallelism): Apply U_g to the superposition

$$|\psi_0\rangle \stackrel{\text{def}}{=} H^{\otimes n}|0\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{x \in \mathbf{N}} |x\rangle$$

to get

$$|\psi_1\rangle \stackrel{\text{def}}{=} U_g|\psi_0\rangle|0\rangle^{\otimes m} = \frac{1}{\sqrt{N}} \sum_{x \in \mathbf{N}} |x, g(x)\rangle.$$

For our example problem, 14 qubits are required [$n = 9$ for x and $m = 5$ for $g(x)$]. The quantum state looks like this (note the periodicity):

$$|\psi_1\rangle = \frac{1}{\sqrt{512}} \quad (\quad |0,1\rangle + |1,11\rangle + |2,16\rangle + |3,8\rangle + |4,4\rangle + |5,2\rangle +$$

$$|6,1\rangle + |7,11\rangle + |8,16\rangle + |9,8\rangle + |10,4\rangle + |11,2\rangle + \cdots)$$

Step 3 (measurement): The function g has a period r, which we want to transfer to the amplitudes of the state so that we can apply the QFT. This is accomplished by measuring (and discarding) the result register (as in Simon's algorithm).[13] Suppose the result register collapses into state g^* (e.g., $g^* = 8$). The input register will collapse into a superposition of all x such that $g(x) = g^*$. We can write it:

$$|\psi_2\rangle \stackrel{\text{def}}{=} \frac{1}{\mathcal{Z}} \sum_{x\in\mathbf{N} \text{ s.t. } g(x)=g^*} |x, g^*\rangle = \frac{1}{\mathcal{Z}} \sum_{x\in\mathbf{N}} f_x |x, g^*\rangle = \left[\frac{1}{\mathcal{Z}} \sum_{x\in\mathbf{N}} f_x |x\rangle\right] |g^*\rangle,$$

where

$$f_x \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if } g(x) = g^* \\ 0, & \text{otherwise} \end{cases},$$

and $\mathcal{Z} \stackrel{\text{def}}{=} \sqrt{|\{x \mid g(x) = g^*\}|}$ is a normalization factor. For example,

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\mathcal{Z}}(|3,8\rangle + |9,8\rangle + |15,8\rangle + \cdots) \\ &= \frac{1}{\mathcal{Z}}(|3\rangle + |9\rangle + |15\rangle + \cdots)|8\rangle \end{aligned}$$

Note that the values x for which $f_x \neq 0$ differ from each other by the period; This produces a function f of very strong periodicity. As in Simon's algorithm, if we could measure two such x, we would have useful information, but we can't. Suppose we measure the result register and get $g^* = 8$. Fig. III.24 shows the corresponding $\mathbf{f} = (0,0,0,0,1,0,0,0,0,0,1,0,\ldots)$.

Step 4 (QFT): Apply the QFT to obtain,

$$\begin{aligned} |\psi_3\rangle &\stackrel{\text{def}}{=} U_{\text{QFT}} \left(\frac{1}{\mathcal{Z}} \sum_{x\in\mathbf{N}} f_x |x\rangle\right) \\ &= \frac{1}{\mathcal{Z}} \sum_{\hat{x}\in\mathbf{N}} \hat{f}_{\hat{x}} |\hat{x}\rangle. \end{aligned}$$

[13] As it turns out, this measurement of the result register can be avoided. This is in general true for "internal" measurement processes in quantum algorithms (Bernstein & Vazirani 1997).

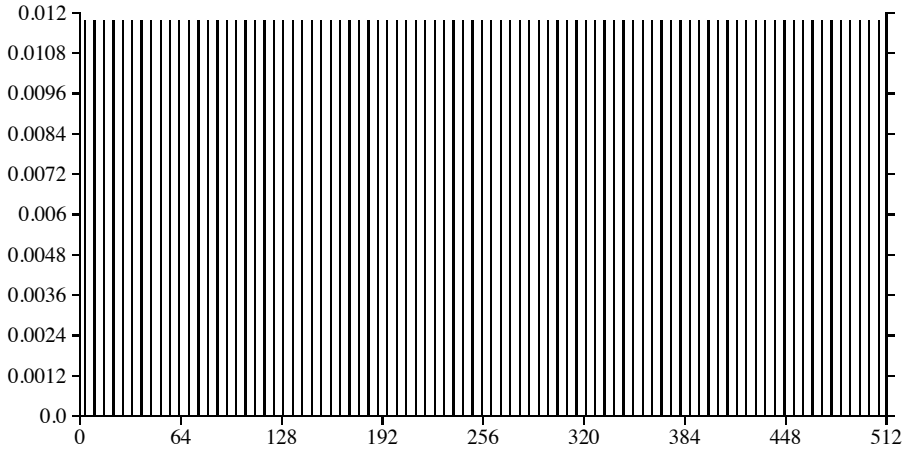

Figure III.24: Example probability distribution $|f_x|^2$ for state $Z^{-1}\sum_{x\in\mathbf{N}} f_x|x, 8\rangle$. In this example the period is $r = 6$ (e.g., at $x = 3, 9, 15, \ldots$). [fig. source: Rieffel & Polak (2000)]

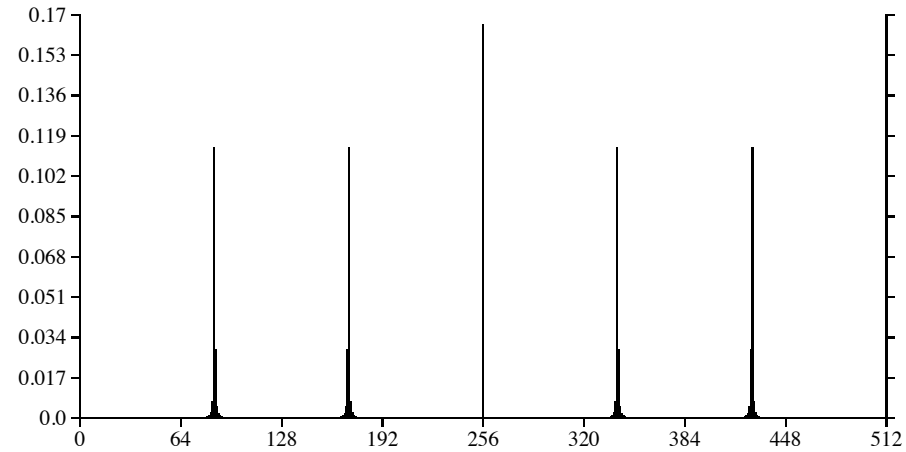

Figure III.25: Example probability distribution $|\hat{f}_{\hat{x}}|^2$ of the quantum Fourier transform of $\mathbf{f}$. The spectrum is concentrated near multiples of $N/6 = 512/6 = 85\ 1/3$, that is $85\ 1/3, 170\ 2/3, 256$, etc. [fig. source: Rieffel & Polak (2000)]

(The collapsed result register $|g^*\rangle$ has been omitted.)

If the period r divides $N = 2^n$, then $\hat{\mathbf{f}}$ will be nonzero only at multiples of the fundamental frequency N/r. That is, the nonzero components will be $|kN/r\rangle$. If it doesn't divide evenly, then the amplitude will be concentrated around these $|kN/r\rangle$. See Fig. III.24 and Fig. III.25 for examples of the probability distributions $|f_x|^2$ and $|\hat{f}_{\hat{x}}|^2$.

Step 5 (period extraction): Measure the state in the computational basis.

Period a power of 2: If $r \mid N$, then the resulting state will be $v \stackrel{\text{def}}{=} |kN/r\rangle$ for some $k \in \mathbf{N}$. Therefore $k/r = v/N$. If k and r are relatively prime, as is likely, then reducing the fraction v/N to lowest terms will produce r in the denominator. In this case the period is discovered.

Period not a power of 2: In the case r does not divide N, it's often possible to guess the period from a continued fraction expansion of v/N.[14] If v/N is sufficiently close to p/q, then a continued fraction expansion of v/N will contain the continued fraction expansion of p/q. For example, suppose the measurement returns $v = 427$, which is not a power of two. This is the result of the continued fraction expansion of v/N (in this case, 427/512) (see IQC):

i	a_i	p_i	q_i	ϵ_i
0	0	0	1	0.8339844
1	1	1	1	0.1990632
2	5	5	6	0.02352941
3	42	211	253	0.5

"which terminates with $6 = q_2 < M \leq q_3$. Thus, $q = 6$ is likely to be the period of f." [IQC]

Step 6 (finding a factor): The following computation applies however we

[14] See Rieffel & Polak (2000, App. B) for an explanation of this procedure and citations for why it works.

got the period q in Step 5. If the guess q is even, then $a^{q/2}+1$ and $a^{q/2}-1$ are likely to have common factors with M. Use the Euclidean algorithm to check this. The reason is as follows. If q is the period of $g(x) = a^x \pmod M$, then $a^q = 1 \pmod M$. This is because, if q is the period, then for all x, $g(x+q) = g(x)$, that is, $a^{q+x} = a^q a^x = a^x \pmod M$ for all x. Therefore $a^q - 1 = 0 \pmod M$. Hence,

$$(a^{q/2}+1)(a^{q/2}-1) = 0 \pmod M.$$

Therefore, unless one of the factors is a multiple of M (and hence $= 0 \bmod M$), one of them has a nontrivial common factor with M.

In the case of our example, the continued fraction gave us a guess $q = 6$, so with $a = 11$ we should consider $11^3+1 = 1332$ and $11^3-1 = 1330$. For $M = 21$ the Euclidean algorithm yields $\gcd(21, 1332) = 3$ and $\gcd(21, 1330) = 7$. We've factored 21!

Iteration: There are several reasons that the preceding steps might not have succeeded: (1) The value v projected from the spectrum might not be close enough to a multiple of N/r (D.3.b). (2) In D.3.b, k and r might not be relatively prime, so that the denominator is only a factor of the period, but not the period itself. (3) In D.3.b, one of the two factors turns out to be a multiple of M. (4) In D.3.b, q was odd. In these cases, a few repetitions of the preceding steps yields a factor of M.
□

D.3.c Recent progress

> To read our E-mail, how mean
> of the spies and their quantum machine;
> be comforted though,
> they do not yet know
> how to factorize twelve or fifteen.
> — Volker Strassen (Nielsen & Chuang, 2010, p. 216)

In this section we review recent progress in hardware implementation of

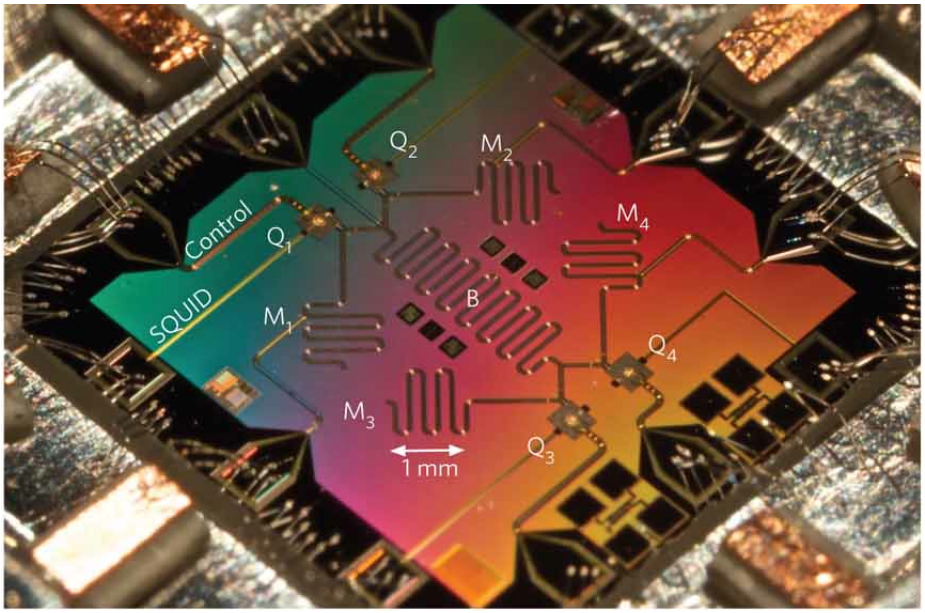

Figure III.26: Hardware implementation of Shor's algorithm developed at UCSB (2012). The M_j are quantum memory elements, B is a quantum "bus," and the Q_j are phase qubits that can be used to implement qubit operations between the bus and memory elements. [source: CPF]

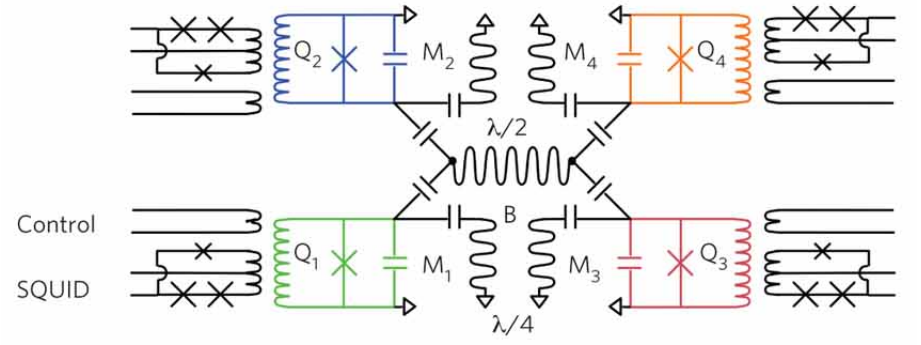

Figure III.27: Circuit of hardware implementation of Shor's algorithm developed at UCSB. [source: CPF]

Shor's algorithm.[15] In Aug. 2012 a group at UC Santa Barbara described a quantum implementation of Shor's algorithm that correctly factored 15 about 48% of the time (50% being the theoretical success rate). (There have been NMR hardware factorizations of 15 since 2001, but there is some doubt if entanglement was involved.) This is a 3-qubit compiled version of Shor's algorithm, where "compiled" means that the implementation of modular exponentiation is for fixed M and a. This compiled version used $a = 4$ as the coprime to $M = 15$. In this case the correct period $r = 2$. The device (Fig. III.26) has nine quantum devices, including four phase qubits and five superconducing co-planar waveguide (CPW) microwave resonators. The four CPWs (M_j) can be used as memory elements and fifth (B) can be used as a "bus" to mediate entangling operations. In effect the qubits Q_j can be read and written. Radio frequency pulses in the bias coil can be used to adjust the qubit's frequency, and gigahertz pulses can be used to manipulate and measure the qubit's state. SQUIDs are used for one-shot readout of the qubits. The qubits Q_j can be tuned into resonance with the bus B or memory elements M_j. The quantum processor can be used to implement the single-qubit gates X, Y, Z, H, and the two-qubit swap (iSWAP) and controlled-phase (C_ϕ) gates. The entanglement protocol can be scaled to an arbitrary number of qubits. The relaxation and dephasing times are about 200ns.

Another group has reported the quantum factoring of 21.[16] Their procedure operates by using one qubit instead of the n qubits in the (upper) control qubits. It does this by doing all the unitaries associated with the lowest-order control qubit, then for the next control qubit, updating the work register after each step, for n interations.

[15]This section is based primarily on Erik Lucero, R. Barends, Y. Chen, J. Kelly, M. Mariantoni, A. Megrant, P. O'Malley, D. Sank, A. Vainsencher, J. Wenner, T. White, Y. Yin, A. N. Cleland & John M. Martinis, "Computing prime factors with a Josephson phase qubit quantum processor." *Nature Physics* **8**, 719–723 (2012) `doi:10.1038/nphys2385` [CPF].

[16]See Martin-Lópex et al., "Experimental realization of Shor's quantum factoring algorithm using qubit recycling," *Nature Photonics* 6, 773–6 (2012).

D.4 Search problems

D.4.a Overview

Many problems can be formulated as search problems over a solution space $\mathcal{S}$.[17] That is, find the $x \in \mathcal{S}$ such that some predicate $P(x)$ is true. For example, hard problems such as the Hamiltonian path problem and Boolean satisfiability can be formulated this way. An *unstructured search problem* is a problem that makes no assumptions about the *structure* of the search space, or for which there is no known way to make use of it (also called a *needle in a haystack problem*). That is, information about a particular value $P(x_0)$ does not give us usable information about another value $P(x_1)$. In contrast, a *structured search problem* is a problem in which the structure of the solution space can be used to guide the search, for example, searching an alphabetized array. In general, unstructured search takes $\mathcal{O}(M)$ evaluations, where $M = |\mathcal{S}|$ is the size of the solution space (which is often exponential in the size of the problem). On the average it will be $M/2$ (think of searching an unordered array) to find a solution with 50% probability.

We will see that Grover's algorithm can do unstructured search on a quantum computer with bounded probability in $\mathcal{O}(\sqrt{M})$ time, that is, quadratic speedup. This is provably more efficient than any algorithm on a classical computer, which is good (but not great). Unfortunately, it has been proved that Grover's algorithm is optimal for unstructured search. Therefore, to do better requires exploiting the structure of the solution space. *Quantum computers do not exempt us from understanding the problems we are trying to solve!* Shor's algorithm is an excellent example of exploiting the structure of a problem domain. Later we will take a look at *heuristic quantum search algorithms* that do make use of problem structure.

D.4.b Grover's Algorithm

algorithm Grover:

Input: Let M be the size of the solution space and pick n such that $2^n \geq M$.

[17] This section is based primarily on Rieffel & Polak (2000).

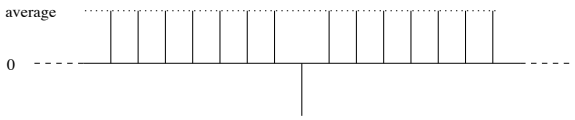

Figure III.28: Depiction of the result of phase rotation (changing the sign) of solutions in Grover's algorithm. [source: Rieffel & Polak (2000)]

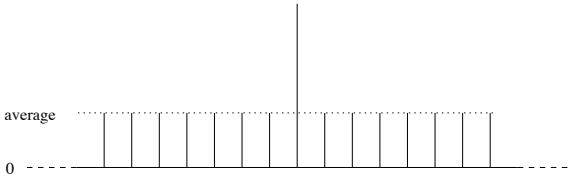

Figure III.29: Depiction of result of inversion about the mean in Grover's algorithm. [source: Rieffel & Polak (2000)]

Let $N \stackrel{\text{def}}{=} 2^n$ and let $\mathbf{N} \stackrel{\text{def}}{=} \mathbf{2}^n = \{0, 1, \ldots, N-1\}$, the set of n-bit strings. We are given a computable predicate $P : \mathbf{N} \to \mathbf{2}$. Suppose we have a quantum gate array U_P (an *oracle*) that computes the predicate:

$$U_P|x, y\rangle = |x, y \oplus P(x)\rangle.$$

Application: Consider what happens if, as usual, we apply the function to a superposition of all possible inputs $|\psi_0\rangle$:

$$U_P|\psi_0\rangle|0\rangle = U_P\left[\frac{1}{\sqrt{N}}\sum_{x\in\mathbf{N}}|x, 0\rangle\right] = \frac{1}{\sqrt{N}}\sum_{x\in\mathbf{N}}|x, P(x)\rangle.$$

Notice that the components we want, $|x, 1\rangle$, and the components we don't want, $|x, 0\rangle$, all have the same amplitude, $\frac{1}{\sqrt{N}}$. So if we measure the state, the chances of getting a hit are very small, $\mathcal{O}(2^{-n})$. The trick, therefore, is to amplify the components that we want at the expense of the ones we don't want; this is what Grover's algorithm accomplishes.

Sign-change: To do this, first we change the sign of every solution (a phase rotation of π). That is, if the state is $\sum_j a_j|x_j, P(x_j)\rangle$, then we want to change a_j to $-a_j$ whenever $P(x_j) = 1$. See Fig. III.28. I'll get to the technique in a moment.

Inversion about mean: Next, we invert all the components around their mean amplitude (which is a little less than the amplitudes of the non-solutions); the result is shown in Fig. III.29. As a result of this operation, amplitudes of non-solutions go from a little above the mean to a little below it, but amplitudes of solutions go from far below the mean to far above it. This amplifies the solutions.

Iteration: This *Grover iteration* (the sign change and inversion about the mean) is repeated $\frac{\pi\sqrt{N}}{4}$ times. Thus the algorithm is $\mathcal{O}(\sqrt{N})$.

Measurement: Measurement yields an x_0 for which $P(x_0) = 1$ with high probability. Specifically, if there is exactly one solution $x_0 \in \mathcal{S}$, then $\frac{\pi\sqrt{N}}{8}$ iterations will yield it with probability 1/2. With $\frac{\pi\sqrt{N}}{4}$ iterations, the probability of failure drops to $1/N = 2^{-n}$. Unlike with most classical algorithms, additional iterations will give a *worse* result! This is because Grover iterations are unitary rotations, and so excessive rotations can rotate past the solution. Therefore it is critical to know when to stop. Fortunately there is a quantum technique (Brassard et al. 1998) for determining the optimal stopping point. Grover's iteration can be used for a wide variety of problems as a part of other quantum algorithms.
□

In the following geometric analysis, I will suppose that there is just one answer α such that $P(\alpha) = 1$; then $|\alpha\rangle$ is the desired answer vector. Let $|\omega\rangle$ be a uniform superposition of all the other (non-answer) states, and observe that $|\alpha\rangle$ and $|\omega\rangle$ are orthonormal. Therefore, initially the state is $|\psi_0\rangle = \frac{1}{\sqrt{N}}|\alpha\rangle + \sqrt{\frac{N-1}{N}}|\omega\rangle$. In general, after k iterations the state is $|\psi_k\rangle =$

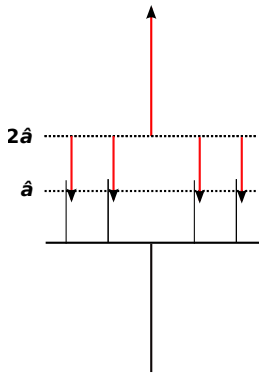

Figure III.30: Process of inversion about the mean in Grover's algorithm. The black lines represent the original amplitudes a_j. The red lines represent $2\bar{a} - a_j$, with the arrow heads indicating the new amplitudes a'_j.

$a|\alpha\rangle + w|\omega\rangle$, for some a, w with $|a|^2 + |w|^2 = 1$.

The sign change operation transforms the state as follows:

$$|\psi_k\rangle = a|\alpha\rangle + w|\omega\rangle \quad \mapsto \quad -a|\alpha\rangle + w|\omega\rangle = |\psi'_k\rangle,$$

where I've called the result $|\psi'_k\rangle$. This is a reflection across the $|\omega\rangle$ vector, which means that it will be useful to look at reflections more generally.

Suppose that $|\phi\rangle$ and $|\phi^\perp\rangle$ are orthonormal vectors and that $|\psi\rangle = a|\phi^\perp\rangle + b|\phi\rangle$ is an arbitrary vector in the space they span. The reflection of $|\psi\rangle$ across $|\phi\rangle$ is $|\psi'\rangle = -a|\phi^\perp\rangle + b|\phi\rangle$. Since $a = \langle\phi^\perp \mid \psi\rangle$ and $b = \langle\phi \mid \psi\rangle$, we know $|\psi\rangle = |\phi\rangle\langle\phi \mid \psi\rangle + |\phi^\perp\rangle\langle\phi^\perp \mid \psi\rangle$, and you can see that $|\psi'\rangle = |\phi\rangle\langle\phi \mid \psi\rangle - |\phi^\perp\rangle\langle\phi^\perp \mid \psi\rangle$. Hence the operator to reflect across $|\phi\rangle$ is $R_\phi \stackrel{\text{def}}{=} |\phi\rangle\langle\phi| - |\phi^\perp\rangle\langle\phi^\perp|$. Alternate forms of this operator are $2|\phi\rangle\langle\phi| - I$ and $I - 2|\phi^\perp\rangle\langle\phi^\perp|$, that is, subtract twice the perpendicular component.

The sign change can be expressed as a reflection:

$$R_\omega = |\omega\rangle\langle\omega| - |\alpha\rangle\langle\alpha| = I - 2|\alpha\rangle\langle\alpha|,$$

which expresses the sign-change of the answer vector clearly. Of course we don't know $|\alpha\rangle$, which is why we will have to use a different process to accomplish this reflection (see p. 150). We also will see that the inversion about the mean is equivalent to reflecting that state vector across $|\psi_0\rangle$.

But first, taking this for granted, let's see the effect of the Grover iteration (Fig. III.31). Let θ be the angle between $|\psi_0\rangle$ and $|\omega\rangle$. It's given by the inner

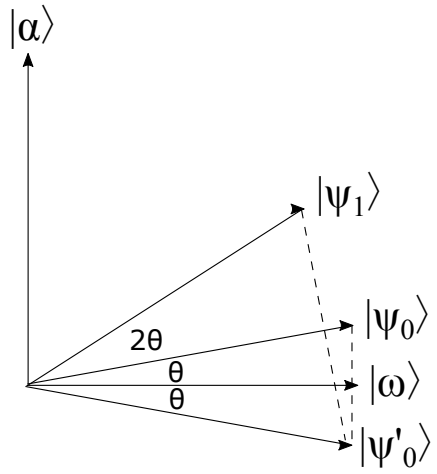

Figure III.31: Geometry of first Grover iteration.

---

product $\cos\theta = \langle\psi_0 \mid \omega\rangle = \sqrt{\frac{N-1}{N}}$. Therefore the sign change reflects $|\psi_0\rangle$ from θ above $|\omega\rangle$ into $|\psi_0'\rangle$, which is θ below it. Inversion about the mean reflects $|\psi_0'\rangle$ from 2θ below $|\psi_0\rangle$ into a state we call $|\psi_1\rangle$, which is 2θ above it. Therefore, in going from $|\psi_0\rangle$ to $|\psi_1\rangle$ the state vector has rotated 2θ closer to $|\alpha\rangle$.

You can see that after k iterations, the state vector $|\psi_k\rangle$ will be $(2k+1)\theta$ above $|\omega\rangle$. We can solve $(2k+1)\theta = \pi/2$ to get the required number of iterations to bring $|\psi_k\rangle$ to $|\alpha\rangle$. Note that for small θ, $\theta \approx \sin\theta = \frac{1}{\sqrt{N}}$ (which is certainly small). Hence, we want $(2k+1)/\sqrt{N} \approx \pi/2$, or $2k+1 \approx \pi\sqrt{N}/2$. That is, $k \approx \pi\sqrt{N}/4$ is the required number of iterations. Note that after $\pi\sqrt{N}/8$ iterations, we are about halfway there (i.e., $\pi/4$), so the probability of success is 50%. In general, the probability of success is about $\sin^2\frac{2k+1}{\sqrt{N}}$.

Now for the techniques for changing the sign and inversion about the mean. Let $|\psi_k\rangle$ be the state after k iterations ($k \geq 0$). To change the sign, simply apply U_P to $|\psi_k\rangle|-\rangle$. To see the result, let $X_0 = \{x \mid P(x) = 0\}$ and $X_1 = \{x \mid P(x) = 1\}$, the solution set. Then:

$$\begin{aligned} & U_P|\psi_k\rangle|-\rangle \\ &= U_P\left[\sum_{x\in\mathbf{N}} a_x|x,-\rangle\right] \end{aligned}$$

$$
\begin{aligned}
&= U_P \left[\frac{1}{\sqrt{2}} \sum_{x \in \mathbf{N}} a_x |x,0\rangle - a_x |x,1\rangle \right] \\
&= \frac{1}{\sqrt{2}} U_P \left[\sum_{x \in X_0} a_x |x,0\rangle + \sum_{x \in X_1} a_x |x,0\rangle - \sum_{x \in X_0} a_x |x,1\rangle - \sum_{x \in X_1} a_x |x,1\rangle \right] \\
&= \frac{1}{\sqrt{2}} \left[\sum_{x \in X_0} a_x U_P |x,0\rangle + \sum_{x \in X_1} a_x U_P |x,0\rangle \right. \\
&\qquad \left. - \sum_{x \in X_0} a_x U_P |x,1\rangle - \sum_{x \in X_1} a_x U_P |x,1\rangle \right] \\
&= \frac{1}{\sqrt{2}} \left[\sum_{x \in X_0} a_x |x,0\rangle + \sum_{x \in X_1} a_x |x,1\rangle \right. \\
&\qquad \left. - \sum_{x \in X_0} a_x |x,1 \oplus 0\rangle - \sum_{x \in X_1} a_x |x,1 \oplus 1\rangle \right] \\
&= \frac{1}{\sqrt{2}} \left[\sum_{x \in X_0} a_x |x\rangle|0\rangle + \sum_{x \in X_1} a_x |x\rangle|1\rangle - \sum_{x \in X_0} a_x |x\rangle|1\rangle - \sum_{x \in X_1} a_x |x\rangle|0\rangle \right] \\
&= \frac{1}{\sqrt{2}} \left(\sum_{x \in X_0} a_x |x\rangle - \sum_{x \in X_1} a_x |x\rangle \right) (|0\rangle - |1\rangle) \\
&= \left(\sum_{x \in X_0} a_x |x\rangle - \sum_{x \in X_1} a_x |x\rangle \right) |-\rangle.
\end{aligned}
$$

Therefore the signs of the solutions have been reversed (they have been rotated by π). Notice how $|-\rangle$ in the target register can be used to separate the 0 and 1 results by rotation. This is a useful idea!

It remains to show the connection between inversion about the mean and reflection across $|\psi_0\rangle$. This reflection is given by $R_{\psi_0} = 2|\psi_0\rangle\langle\psi_0| - I$. Note that:

$$
|\psi_0\rangle\langle\psi_0| = \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{x} \in \mathbf{N}} |\mathbf{x}\rangle \right) \left(\frac{1}{\sqrt{N}} \sum_{\mathbf{y} \in \mathbf{N}} \langle \mathbf{y}| \right) = \frac{1}{N} \sum_{\mathbf{x} \in \mathbf{N}} \sum_{\mathbf{y} \in \mathbf{N}} |\mathbf{x}\rangle\langle\mathbf{y}|.
$$

This is the *diffusion matrix*:

$$\begin{pmatrix} \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \cdots & \frac{1}{N} \end{pmatrix},$$

which, as we will see, does the averaging.

To perform inversion about the mean, let $\bar{a}$ be the average of the a_j (see Fig. III.30). Inversion about the mean is accomplished by the transformation:

$$\sum_{j\in\mathbf{N}} a_j|x_j\rangle \mapsto \sum_{j\in\mathbf{N}}(2\bar{a}-a_j)|x_j\rangle.$$

To see this, write $a_j = \bar{a} \pm \delta_j$, that is, as a difference from the mean. Then $2\bar{a} - a_j = 2\bar{a} - (\bar{a} \pm \delta_j) = \bar{a} \mp \delta_j$. Therefore an amplitude δ_j below the mean will be transformed to δ_j above, and vice verse. But an amplitude that is negative, and thus very far below the mean, will be transformed to an amplitude much above the mean. This is exactly what we want in order to amplify the negative components, which correspond to solutions.

Inversion about the mean is accomplished by a "Grover diffusion transformation" D. To derive the matrix D, consider the new amplitude a'_j as a function of all the others:

$$a'_j \stackrel{\text{def}}{=} 2\bar{a} - a_j = 2\left(\frac{1}{N}\sum_{k=0}^{N-1} a_k\right) - a_j = \sum_{k\neq j}\frac{2}{N}a_k + \left(\frac{2}{N}-1\right)a_j.$$

This matrix has $\frac{2}{N} - 1$ on the main diagonal and $\frac{2}{N}$ in the off-diagonal elements:

$$D = \begin{pmatrix} \frac{2}{N}-1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N}-1 \end{pmatrix}.$$

Note that $D = 2|\psi_0\rangle\langle\psi_0| - I = R_{\psi_0}$. It is easy to confirm that $DD^\dagger = I$ (Exer. III.49), so the matrix is unitary and therefore a possible quantum operation, but it remains to be seen if it can be implemented efficiently.

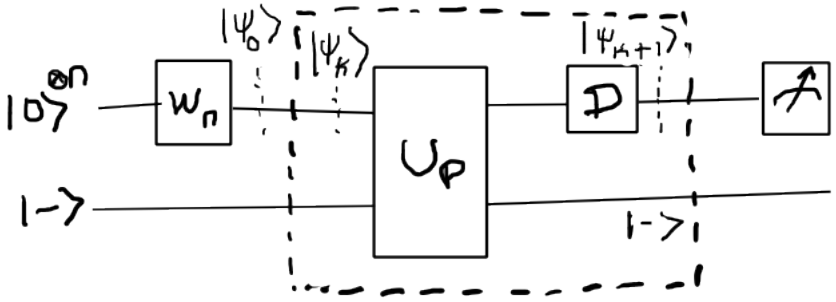

Figure III.32: Circuit for Grover's algorithm. The Grover iteration in the dashed box is repeated $\frac{\pi\sqrt{N}}{4}$ times.

We claim $D = WRW$, where $W = H^{\otimes n}$ is the n-qubit Walsh-Hadamard transform and R is the *phase rotation matrix*:

$$R \stackrel{\text{def}}{=} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{pmatrix}.$$

To see this, let

$$R' \stackrel{\text{def}}{=} R + I = \begin{pmatrix} 2 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}.$$

Then $WRW = W(R' - I)W = WR'W - WW = WR'W - I$. It is easy to show (Exer. III.50) that:

$$WR'W = \begin{pmatrix} \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{2}{N} & \frac{2}{N} & \cdots & \frac{2}{N} \end{pmatrix}.$$

It therefore follows that $D = WR'W - I = WRW$. See Fig. III.32 for a diagram of Grover's algorithm.

It remains to consider the possibility that there may be several solutions to the problem. If there are s solutions, then run the Grover iteration $\frac{\pi\sqrt{N/s}}{4}$ times, which is optimal (Exer. III.51). It can be done in $\sqrt{N/s}$ iterations even if s is unknown.

D.4.c Hogg's heuristic search algorithms

Many important problems can be formulated as *constraint satisfaction problems*, in which we try to find a set of assignments to variables that satisfy specified *constraints*. More specifically let $V \stackrel{\text{def}}{=} \{v_1, \ldots, v_n\}$ be a set of variables, and let $X \stackrel{\text{def}}{=} \{x_1, \ldots, x_n\}$ be a set of values that can be assigned to the variables, and let $C_1, \ldots, C_l$ be the constraints. The set of all possible assignments of values to variables is $V \times X$. Subsets of this set correspond to full or partial assignments, including inconsistent assignments. The set of all such assignments is $\mathcal{P}(V \times X)$.

The sets of assignments form a *lattice* under the $\subseteq$ partial order (Fig. III.33). By assigning bits to the elements of $V \times X$, elements of $\mathcal{P}(V \times X)$ can be represented by mn-element bit strings (i.e., integers in the set $\mathbf{MN} = \{0, \ldots, 2^{mn} - 1\}$); see Fig. III.34. Hogg's algorithms are based on the observation that if an assignment violates the constraints, then so do all those assignments above it in the lattice.

algorithm Hogg:

Initialization: The algorithm begins with all the amplitude concentrated in the bottom of the lattice, $|0 \cdots 0\rangle$ (i.e., the empty set of assignments).

Movement: The algorithm proceeds by moving amplitude up the lattice, while avoiding assignments that violate the constraints; that is, we want to move amplitude from a set to its supersets. For example, we want to redistribute the amplitude from $|1010\rangle$ to $|1110\rangle$ and $|1011\rangle$. Hogg has developed several methods. One method is based on the assumption that the transformation has the form WDW, where $W = H^{\otimes mn}$, the mn-dimensional Walsh-Hadamard transformation, and D is diagonal. The elements of D

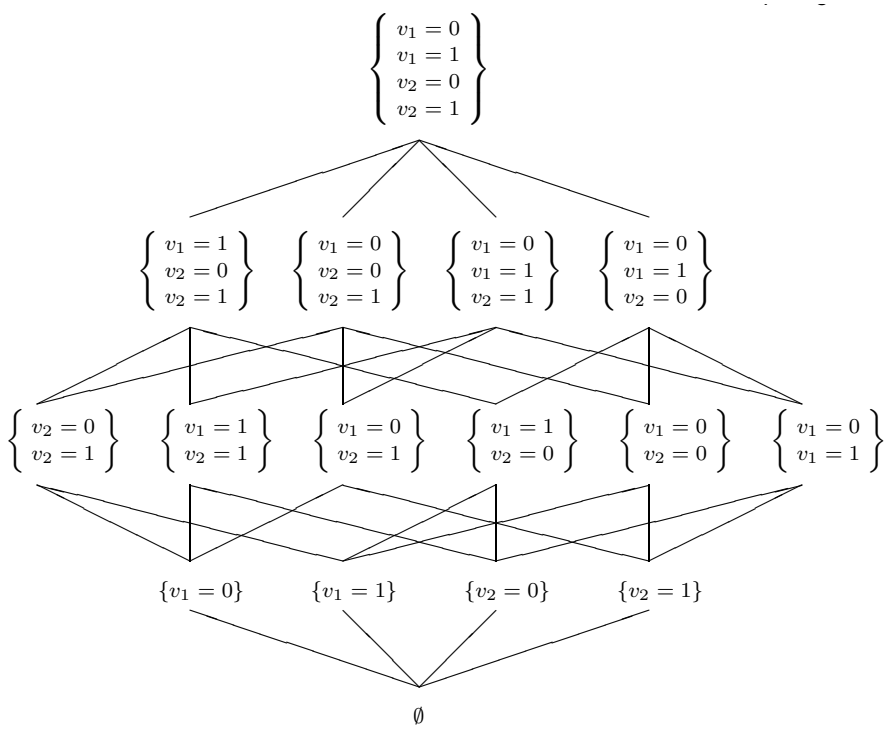

Figure III.33: Lattice of variable assignments. [source: Rieffel & Polak (2000)]

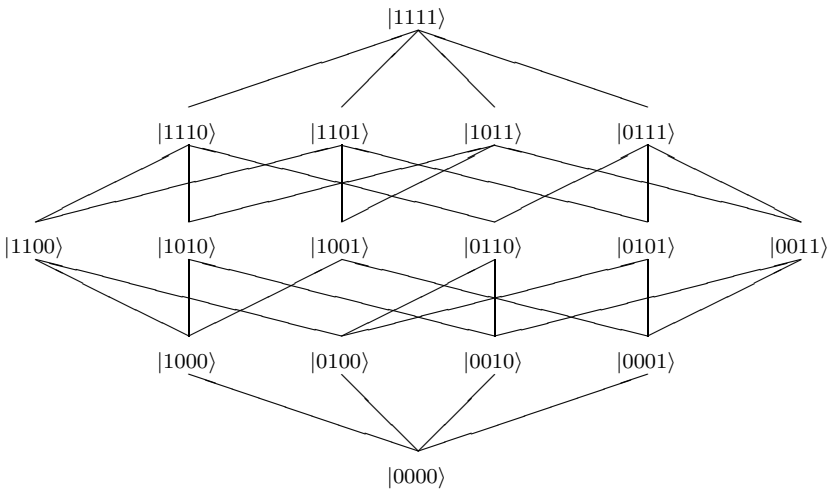

Figure III.34: Lattice of binary strings corresponding to all subsets of a 4-element set. [source: Rieffel & Polak (2000)]

---

depend on the size of the sets. Recall (D.1.b, p. 129) that

$$W|x\rangle = \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{x \cdot z} |z\rangle.$$

As shown in Sec. A.2.c (p. 70), we can derive a matrix representation for W:

$$\begin{aligned} W_{jk} &= \langle j \mid W \mid k \rangle \\ &= \langle j | \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{k \cdot z} |z\rangle \\ &= \frac{1}{\sqrt{2^{mn}}} \sum_{z \in \mathbf{MN}} (-)^{k \cdot z} \langle j \mid z \rangle \\ &= \frac{1}{\sqrt{2^{mn}}} (-1)^{k \cdot j}. \end{aligned}$$

Note that $k \cdot j = |k \cap j|$, where on the right-hand side we interpret the bit strings as sets.
□

The general approach is to try to steer amplitude away from sets that violate the constraints, but the best technique depends on the particular problem. One technique is to invert the phase on bad subsets so that they tend to cancel the contribution of good subsets to supersets. This could be done by a process like Grover's algorithm using a predicate that tests for violation of constraints. Another approach is to assign random phases to bad sets.

It is difficult to analyze the probability that an iteration of a heuristic algorithm will produce a solution, and so its efficiency is usually evaluated empirically, but empirical tests will be difficult to apply to quantum heuristic search until larger quantum computers are available, since classical computers require exponential time to simulate quantum systems. Small simulations, however, indicate that Hogg's algorithms may provide polynomial speedup over Grover's algorithm.

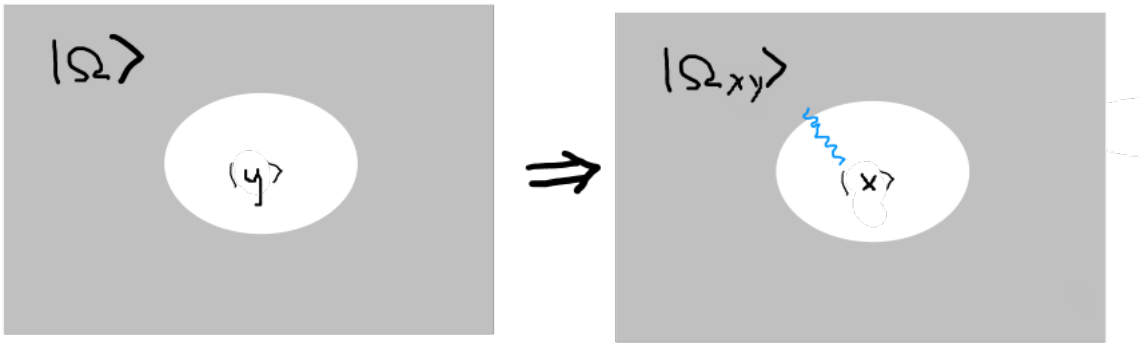

Figure III.35: Effects of decoherence on a qubit. On the left is a qubit $|y\rangle$ that is mostly isoloated from its environment $|\Omega\rangle$. On the right, a weak interaction between the qubit and the environment has led to a possibly altered qubit $|x\rangle$ and a correspondingly (slightly) altered environment $|\Omega_{xy}\rangle$.

D.5 Quantum error correction

D.5.a Motivation

Quantum coherence is very difficult to maintain for long.[18] Even weak interactions with the environment can affect the quantum state, and we've seen that the amplitudes of the quantum state are critical to quantum algorithms. On classical computers, bits are represented by very large numbers of particles (but that is changing). On quantum computers, qubits are represented by atomic-scale states or objects (photons, nuclear spins, electrons, trapped ions, etc.). They are very likely to become entangled with computationally irrelevant states of the computer and its environment, which are out of our control. Quantum error correction is similar to classical error correction in that additional bits are introduced, creating redundancy that can be used to correct errors. It is different from classical error correction in that: (a) We want to restore the entire quantum state (i.e., the continuous amplitudes), not just 0s and 1s. Further, errors are continuous and can accumulate. (b) It must obey the no-cloning theorem. (c) Measurement destroys quantum information.

[18] This section follows Rieffel & Polak (2000).

D.5.b Effect of decoherence

Ideally the environment $|\Omega\rangle$, considered as a quantum system, does not interact with the computational state. But if it does, the effect can be categorized as a unitary transformation on the environment-qubit system. Consider decoherence operator D describing a bit flip error in a single qubit (Fig. III.35):

$$D : \begin{cases} |\Omega\rangle|0\rangle & \Longrightarrow & |\Omega_{00}\rangle|0\rangle + |\Omega_{10}\rangle|1\rangle \\ |\Omega\rangle|1\rangle & \Longrightarrow & |\Omega_{01}\rangle|0\rangle + |\Omega_{11}\rangle|1\rangle \end{cases} .$$

In this notation the state vectors $|\Omega_{xy}\rangle$ are not normalized, but incorporate the amplitudes of the various outcomes. In the case of no error, $|\Omega_{00}\rangle = |\Omega_{11}\rangle = |\Omega\rangle$ and $|\Omega_{01}\rangle = |\Omega_{10}\rangle = \mathbf{0}$. If the entanglement with the environment is small, then $\|\Omega_{01}\|, \|\Omega_{10}\| \ll 1$ (small exchange of amplitude).

Define *decoherence operators* $D_{xy}|\Omega\rangle \stackrel{\text{def}}{=} |\Omega_{xy}\rangle$, for $x, y \in \mathbf{2}$, which describe the effect of the decoherence on the environment. (These are not unitary, but are the products of scalar amplitudes and unitary operators for the various outcomes.) Then the evolution of the joint system is defined by the equations:

$$\begin{aligned} D|\Omega\rangle|0\rangle &= (D_{00} \otimes I + D_{10} \otimes X)|\Omega\rangle|0\rangle, \\ D|\Omega\rangle|1\rangle &= (D_{01} \otimes X + D_{11} \otimes I)|\Omega\rangle|1\rangle. \end{aligned}$$

Alternately, we can define it:

$$D = D_{00} \otimes |0\rangle\langle 0| + D_{10} \otimes |1\rangle\langle 0| + D_{01} \otimes |0\rangle\langle 1| + D_{11} \otimes |1\rangle\langle 1|.$$

Now, it's easy to show (Exer. III.19):

$$|0\rangle\langle 0| = \frac{1}{2}(I + Z), |0\rangle\langle 1| = \frac{1}{2}(X - Y), |1\rangle\langle 0| = \frac{1}{2}(X + Y), |1\rangle\langle 1| = \frac{1}{2}(I - Z),$$

where $Y = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$. Therefore

$$\begin{aligned} D &= \frac{1}{2}[D_{00} \otimes (I + Z) + D_{01} \otimes (X - Y) + \\ &\quad D_{10} \otimes (X + Y) + D_{11} \otimes (I - Z)] \\ &= \frac{1}{2}[(D_{00} + D_{11}) \otimes I + (D_{10} + D_{01}) \otimes X + \\ &\quad (D_{10} - D_{01}) \otimes Y + (D_{00} - D_{11}) \otimes Z]. \end{aligned}$$

Therefore the bit flip error can be described as a linear combination of the Pauli matrices. It is generally the case that the effect of decoherence on a single qubit can be described by a linear combination of the Pauli matrices, which is important, since qubits are subject to various errors beside bit flips. This is a distinctive feature about quantum errors: they have a finite basis, and because they are unitary, they are therefore invertible. In other words, single-qubit errors can be characterized in terms of a linear combination of the Pauli matrices (which span the space of 2×2 self-adjoint unitary matrices: C.2.a, p. 105): I (no error), X (bit flip error), Y (phase error), and $Z = YX$ (bit flip phase error). Therefore a single qubit error is represented by $e_0\sigma_0 + e_1\sigma_1 + e_2\sigma_2 + e_3\sigma_3 = \sum_{j=0}^{3} e_j\sigma_j$, where the σ_j are the Pauli matrices (Sec. C.2.a, p. 105).

D.5.c Correcting the quantum state

We consider a basis set of unitary "error operators" E_j, so that the error transformation is a superposition $E \stackrel{\text{def}}{=} \sum_j e_j E_j$. In the more general case of quantum registers, the E_j affect the entire register, not just a single qubit.

algorithm quantum error correction:

Encoding: An n-bit register is encoded in $n + m$ bits, where the extra bits are used for error correction. Let $y \stackrel{\text{def}}{=} C(x) \in \mathbf{2}^{m+n}$ be the $n + m$ bit code for $x \in \mathbf{2}^n$. As in classical error correcting codes, we embed the message in a space of higher dimension.

Error process: Suppose $\tilde{y} \in \mathbf{2}^{m+n}$ is the result of error type k, $\tilde{y} = E_k(y)$.

Syndrome: Let $k = S(\tilde{y})$ be a function that determines the error syndrome, which identifies the error E_k from the corrupted code. That is, $S(E_k(y)) = k$.

Correction: Since the errors are unitary, and the syndrome is known, we can invert the error and thereby correct it: $y = E^{-1}_{S(\tilde{y})}(\tilde{y})$.

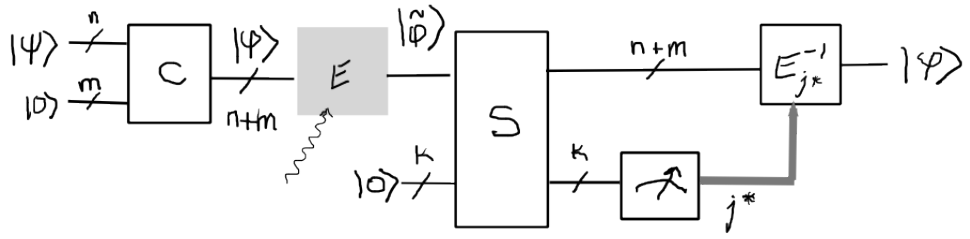

Figure III.36: Circuit for quantum error correction. $|\psi\rangle$ is the n-qubit quantum state to be encoded by C, which adds m error-correction qubits to yield the encoded state $|\phi\rangle$. E is a unitary superposition of error operators E_j, which alter the quantum state to $|\tilde{\phi}\rangle$. S is the syndrome extraction operator, which computes a superposition of codes for the errors E. The syndrome register is measured, to yield a particular syndrome code j^*, which is used to select a corresponding inverse error transformation $E_{j^*}^{-1}$ to correct the error.

Quantum case: Now consider the quantum case, in which the state $|\psi\rangle$ is a superposition of basis vectors, and the error is a superposition of error types, $E = \sum_j e_j E_j$. This is an orthogonal decomposition of E (see Fig. III.36).

Encoding: The encoded state is $|\phi\rangle \stackrel{\text{def}}{=} C|\psi\rangle|\mathbf{0}\rangle$. There are several requirements for a useful quantum error correcting code. Obviously, the codes for orthogonal inputs must be orthogonal; that is, if $\langle\psi \mid \psi'\rangle = 0$, then $C|\psi, \mathbf{0}\rangle$ and $C|\psi', \mathbf{0}\rangle$ are orthogonal: $\langle\psi, \mathbf{0}|C^\dagger C|\psi', \mathbf{0}\rangle = 0$. Next, if $|\phi\rangle$ and $|\phi'\rangle$ are the codes of distinct inputs, we do not want them to be confused by the error processes, so $\langle\phi|E_j^\dagger E_k|\phi'\rangle = 0$ for all i, j. Finally, we require that for each pair of error indices j, k, there is a number m_{jk} such that $\langle\phi|E_j^\dagger E_k|\phi\rangle = m_{jk}$ for every code $|\phi\rangle$. This means that the error syndromes are independent of the codes, and therefore the syndromes can be measured without collapsing superpositions in the codes, which would make them useless for quantum computation.

Error process: Let $|\tilde{\phi}\rangle \stackrel{\text{def}}{=} E|\phi\rangle$ be the code corrupted by error.

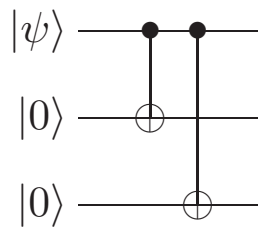

Figure III.37: Quantum encoding circuit for triple repetition code. [source: NC]

Syndrome extraction: Apply the syndrome extraction operator to the encoded state, augmented with enough extra qubits to represent the set of syndromes. This yields a superposition of syndromes:

$$S|\tilde{\phi}, \mathbf{0}\rangle = S\left(\sum_j e_j E_j|\phi\rangle\right) \otimes |\mathbf{0}\rangle = \sum_j e_j (SE_j|\phi\rangle|\mathbf{0}\rangle) = \sum_j e_j (E_j|\phi\rangle|j\rangle).$$

Measurement: Measure the syndrome register to obtain some j^* and the collapsed state $E_{j^*}|\phi\rangle|j^*\rangle$.[19]

Correction: Apply $E_{j^*}^{-1}$ to correct the error.
□

Note the remarkable fact that although there was a superposition of errors, we only have to correct one of them to get the original state back. This is because measurement of the error syndrome collapses into a state affected by just that one error.

D.5.d Example

We'll work through an example to illustrate the error correction process. For an example, suppose we use a simple triple redundancy code that assigns

[19] As we mentioned the discussion of in Shor's algorithm (p. 140), it is not necessary to actually perform the measurement; the same effect can be obtained by unitary operations.

$|0\rangle \mapsto |000\rangle$ and $|1\rangle \mapsto |111\rangle$. This is accomplished by a simple quantum gate array:

$$C|0\rangle|00\rangle = |000\rangle, \quad C|1\rangle|00\rangle = |111\rangle.$$

This is not a sophisticated code! It's called a *repetition code*. The three-qubit codes are called *logical zero* and *logical one* (See Fig. III.37). This code can correct single bit flips (*by majority voting*); the errors are represented by the operators:

$$\begin{aligned} E_0 &= I \otimes I \otimes I \\ E_1 &= I \otimes I \otimes X \\ E_2 &= I \otimes X \otimes I \\ E_3 &= X \otimes I \otimes I. \end{aligned}$$

The following works as a syndrome extraction operator:

$$S|x_3, x_2, x_1, 0, 0, 0\rangle \stackrel{\text{def}}{=} |x_3, x_2, x_1, x_1 \oplus x_2, x_1 \oplus x_3, x_2 \oplus x_3\rangle.$$

The $\oplus$s compare each pair of bits, and so the $\oplus$ will be zero if the two bits are the same (the majority). The following table shows the bit flipped (if any), the corresponding syndrome, and the operator to correct it (which is the same as the operator that caused the error):

bit flipped	syndrome	error correction
none	$\|000\rangle$	$I \otimes I \otimes I$
1	$\|110\rangle$	$I \otimes I \otimes X$
2	$\|101\rangle$	$I \otimes X \otimes I$
3	$\|011\rangle$	$X \otimes I \otimes I$

(Note that the correction operators need not be the same as the error operators, although they are in this case.)

For an example, suppose we want to encode the state $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Its code is $|\phi\rangle = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)$. Suppose the following error occurs: $E = \frac{4}{5}X \otimes I \otimes I + \frac{3}{5}I \otimes X \otimes I$ (that is, the bit 3 flips with probability 16/25, and bit 2 with probability 9/25). The resulting error state is

$$\begin{aligned} |\tilde{\phi}\rangle &= E|\phi\rangle \\ &= \left(\frac{4}{5}X \otimes I \otimes I + \frac{3}{5}I \otimes X \otimes I\right) \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) \end{aligned}$$

$$\begin{aligned} &= \frac{4}{5\sqrt{2}} X \otimes I \otimes I(|000\rangle - |111\rangle) + \frac{3}{5\sqrt{2}} I \otimes X \otimes I(|000\rangle - |111\rangle) \\ &= \frac{4}{5\sqrt{2}}(|100\rangle - |011\rangle) + \frac{3}{5\sqrt{2}}(|010\rangle - |101\rangle). \end{aligned}$$

Applying the syndrome extraction operator yields:

$$\begin{aligned} S|\tilde{\phi}, 000\rangle &= S\left[\frac{4}{5\sqrt{2}}(|100000\rangle - |011000\rangle) + \frac{3}{5\sqrt{2}}(|010000\rangle - |101000\rangle)\right] \\ &= \frac{4}{5\sqrt{2}}(|100011\rangle - |011011\rangle) + \frac{3}{5\sqrt{2}}(|010101\rangle - |101101\rangle) \\ &= \frac{4}{5\sqrt{2}}(|100\rangle - |011\rangle) \otimes |011\rangle + \frac{3}{5\sqrt{2}}(|010\rangle - |101\rangle) \otimes |101\rangle \end{aligned}$$

Measuring the syndrome register yields either $|011\rangle$ (representing an error in bit 3) or $|101\rangle$ (representing an error in bit 2). Suppose we get $|011\rangle$. The state collapses into:

$$\frac{1}{\sqrt{2}}(|100\rangle - |011\rangle) \otimes |011\rangle.$$

Note that we have projected into a subspace for just one of the two bit-flip errors that occurred (the flip in bit 3). The measured syndrome $|011\rangle$ tells us to apply $X \otimes I \otimes I$ to the first three bits, which restores $|\phi\rangle$:

$$(X \otimes I \otimes I)\frac{1}{\sqrt{2}}(|100\rangle - |011\rangle) = \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle) = |\phi\rangle.$$

We can do something similar to correct single phase flip (Z) errors by using the encoding $|0\rangle \mapsto |{+}{+}{+}\rangle$, $|1\rangle \mapsto |{-}{-}{-}\rangle$ (Exer. III.54). To see this, recall that Z in the sign basis is the analog of X is the computational basis.

D.5.e Discussion

There is a nine-qubit code, called the *Shor code*, that can correct arbitrary errors on a single qubit, even replacing the entire qubit by garbage (Nielsen & Chuang, 2010, §10.2). The essence of this code is that triple redundancy is used to correct X errors, and triple redundancy again to correct Z errors, thus requiring nine code qubits for each logical qubut. Since $Y = ZX$ and the Pauli matrices are a basis, this code is able to correct all errors.

Quantum error correction is remarkable in that an entire continuum of errors can be corrected by correcting only a discrete set of errors. This

works in quantum computation, but not classical analog computing. The general goal in syndrome extraction is to separate the syndrome information from the computational information in such a way that the syndrome can be measured without collapsing any of the computational information. Since the syndrome is unitary, it can be inverted to correct the error.

What do we do about noise in the gates that do the encoding and decoding? It is possible to do *fault-tolerant quantum computation*. "Even more impressively, fault-tolerance allow us to perform logical operations on encoded quantum states, in a manner which tolerates faults in the underlying gate operations." (Nielsen & Chuang, 2010, p. 425) Indeed, "provided the noise in individual quantum gates is below a certain constant threshold it is possible to efficiently perform an arbitrarily large quantum computation." (Nielsen & Chuang, 2010, p. 425)[20]

[20]See Nielsen & Chuang (2010, §10.6.4).