Technische Universität Graz

Bachelorarbeit Technische Physik

Quantum Computation

Tobias Kugel

Mat. Nr: 01530702

July 20, 2018

Contents

1 Introduction

The quantum theory was postulated to explain physical phenomena which could not be explained by classical physics anymore. The new insights enabled scientists to construct computers which use the quantum mechanical properties to operate. These are the so-called quantum computers. In the following subsections there will be a short introduction of the key terms and relevant facts related to quantum computers.

1.1 Qubits

A Quantumbit, or qubit for short, is a 2-dimensional Hilbert space $\mathcal{H}_2$. [1] The states of the bits 0 and 1 of a classical computer correspond to the qubit states $|0\rangle$ and $|1\rangle$. These are quantum mechanical states with only two different outcomes when being measured. There are several opportunities to realize this so-called two state system.[1] Using a quantum mechanical property called superposition principle enables to generate more

[1]See: Einführung in Quantencomputer [2], P.29

states than the classical 0 and 1.[1] A superposition is a linear combination of single states.

$$|\Psi\rangle = \sum_{k\in|0\rangle,|1\rangle}^{n} \alpha_k |k\rangle \tag{1}$$

A condition for every linear combination is the norm of the state: $\sum_k^n |\alpha_k|^2 = 1$. When measuring the state $|\Psi\rangle$ the probability to measure the state $|k\rangle$ is equal to the factor $|\alpha_k|^2$. [1] The qubits $|0\rangle$ and $|1\rangle$ get assigned to the vector form:

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix}$$

1.2 Quantum system

A quantum system is a 2^n-dimensional Hilbert space out of n-qubits. [4] For example, a quantum system with two qubits would be generated by a tensor product of the two Hilberspaces $\mathcal{H}_2 \otimes \mathcal{H}_2 = \mathcal{H}_4$. There are four new basis vectors which form an orthogonal system: $|00\rangle$,$|01\rangle$,$|10\rangle$ and $|11\rangle$. This is an example of how two combined states would look like:

$$|\Theta\rangle_{\in\mathcal{H}_4} = |\Phi\rangle_{\in\mathcal{H}_2} \otimes |\Psi\rangle_{\in\mathcal{H}_2} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{2}$$

The vector form of the new states is the tensor product of the single qubits:

$$|0\rangle\otimes|0\rangle = |00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} \quad |0\rangle\otimes|1\rangle = |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \quad |1\rangle\otimes|0\rangle = |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} \quad |1\rangle\otimes|1\rangle = |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

Taking this 4-dimensional system and creating an operation with a unary operator[2] $\hat{O}$ which only interacts with the second qubit is realized as follows:

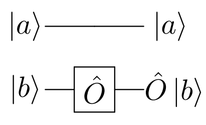


Figure 1: A quantum operation with two qubits

Mathematically, the quantum operation of figure 1 is the tensor product of the states:

$$(\mathbb{1}_2 \otimes \hat{O})\,|ab\rangle = \mathbb{1}_2\,|a\rangle \otimes \hat{O}\,|b\rangle \tag{3}$$

[1] See: Quantum Computation and Quantum Information [3], P.15
[2] A unary operator maps only $\mathcal{H}_2 \rightarrow \mathcal{H}_2$. [5], P.14

2 Quantum Gates

Quantum gates enable operations with qubits. They can be represented as matrices which have to be unitary. ($MM^\dagger = \mathbb{1}$) That makes the operations always reversible which means that there is never a loss of information throughout the whole computation process. [1]

2.1 NOT-Gate

The NOT-gate is simply turning the qubit state from $|0\rangle$ to $|1\rangle$ and vice versa. This is an illustration of how this gate works:

$$|a\rangle - \boxed{N} - |\overline{a}\rangle$$

Figure 2: NOT-Gate ($\overline{a}$ is the inverse of a)

The matrix representation of the NOT operation (figure 2) on a single qubit is:

$$N = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{4}$$

2.2 Hadamard-Gate

The Hadamard-gate is a gate which does not exist in classical computing. It changes a qubit into a superposition of two states by rotating it.[1] This gives many new opportunities for computation processes.

$$|0\rangle - \boxed{H} - \tfrac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|1\rangle - \boxed{H} - \tfrac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

Figure 3: Hadamard-Gates when operating a $|0\rangle$ and a $|1\rangle$ state.

The matrix representation of a Hadamard-gate (figure 3) is:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{5}$$

2.3 CNOT-Gate

All the gates before were gates for operating single qubits. The CNOT-Gate needs a quantum system of two qubits. It negates the second qubit only if there is a $|1\rangle$ at the first qubit. This operation is equal to a modulo 2 ($\oplus$) addition. All quantum circuits can be constructed using only CNOT and unary gates. [5]

[1]See: Quantum Computation and Quantum Information [3], P.19

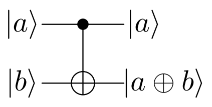


Figure 4: CNOT Gate

The matrix representation of a CNOT gate (figure 4) is:

$$C_{NOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6}$$

2.4 Toffoli-Gate

The Toffoli gate is a three qubit gate which is very important for quantum computation. All quantum circuits can be constructed (in some approximated sense) using only Toffoli gates and Hadamard gates. [6]

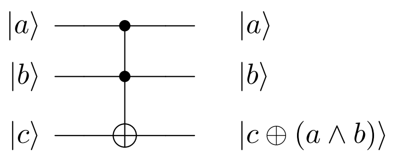


Figure 5: Toffoli-Gate

The Toffoli gate (Figure 5) has three input and output bits. $|a\rangle$ and $|b\rangle$ are the control bits. The third is a target bit that is flipped if both control bits are set to $|1\rangle$, and otherwise is left unchanged.

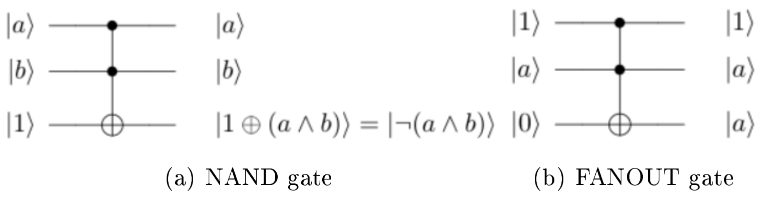


Figure 6: (a) Qubit configuration to perform NAND operations. (b) Qubit configuration to perform FANOUT operations.

In figure 6, two examples of how to use the Toffoli gate can be seen. Having NAND gates (figure 6 (a)) and NOT gates makes it possible to realize all possible logic circuits like in classical computation. The FANOUT gate (figure 6 (b)) is used to duplicate a state.

The matrix representation of a Toffoli gate (figure 5) is:

$$Toffoli = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{7}$$

3 Quantum Algorithms

3.1 Query Operator

The query operator is used to map the solution of a function $f(\mathbf{x}) : \{0,1\}^n \rightarrow \{0,1\}$ onto a qubit.

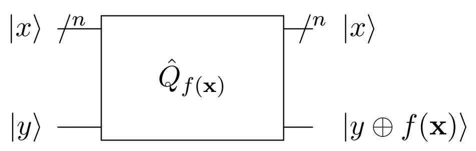


Figure 7: The query operator $\hat{Q}_{f(\mathbf{x})}$ for the function $f(\mathbf{x})$. The '$/^n$' through the wire represents a set of n-qubits.

For example, the query operator acting on the state: $|x\rangle \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{=|y\rangle}$ results in the state:

$$(-1)^{f(\mathbf{x})} |x\rangle \underbrace{\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)}_{=|y\rangle}. \tag{8}$$

3.1.1 Quantum Parallelism

Quantum parallelism is a fundamental feature that is used by quantum algorithms.

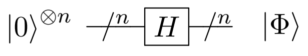


Figure 8: Creation of a parallelized state. The '$/^n$' through the wire represents a set of -qubits.

By applying a Hadamard gate for every qubit a superpositioned state is created. (Figure 8) The state can be written as follows:

$$|\Phi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{9}$$

$|x\rangle$ of equation 9 is the decimal representation of the set of qubits.
Using the state $|\Phi\rangle$ for the query operator gives the result:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(\mathbf{x})} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \tag{10}$$

The superpositioned state $|\Phi\rangle$ has now information about all function values in its amplitudes although the query operator was applied only once. Thus, the function was queried for more than one value simultaneously. Making use of this is called quantum parallelism. [1]

3.2 Deutsch's Algorithm

Deutsch's Algorithm evaluates if the result of a function $f(\mathbf{x}) : \{0,1\} \to \{0,1\}$ is the same for both $\mathbf{x}$ or not.

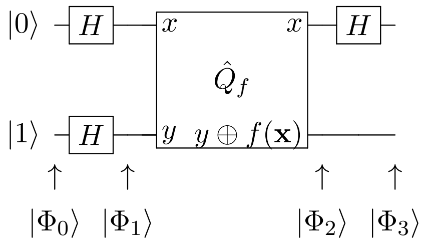


Figure 9: Quantum circuit for Deutsch's algorithm. $\hat{Q}_f$ is the query operator gate. (See figure 7)

Figure 9 shows the quantum circuit for Deutsch's algorithm. The input state

$$|\Phi_0\rangle = |01\rangle \tag{11}$$

is sent through two Hadamard gates and gives

$$|\Phi_1\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \frac{|0\rangle - |1\rangle}{\sqrt{2}}. \tag{12}$$

Equation 8 shows that applying $\hat{Q}_f$ to the state $|x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$ results in $(-1)^{f(\mathbf{x})} |x\rangle \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Applying $\hat{U}_f |\Phi_1\rangle$ gives two possibilities:

$$|\Phi_2\rangle = \begin{cases} \pm \frac{|0\rangle + |1\rangle}{\sqrt{2}} \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(0) = f(1) \\ \pm \frac{|0\rangle - |1\rangle}{\sqrt{2}} \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(0) \neq f(1). \end{cases} \tag{13}$$

Applying a Hadamard gate on the first qubit again yields:

$$|\Phi_3\rangle = \begin{cases} \pm |0\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(0) = f(1) \\ \pm |1\rangle \frac{|0\rangle - |1\rangle}{\sqrt{2}} & \text{if } f(0) \neq f(1). \end{cases} \tag{14}$$

Measuring the first qubit state always gives $|0\rangle$ or $|1\rangle$ as an outcome. Through this, it is possible to find out whether or not the function is the same for every $\mathbf{x}$. The quantum algorithm for this evaluation is faster than a classical one which would need at least two evaluations of $f(\mathbf{x})$. [3]

4 Grover's Algorithm

A computer's searching process for elements in a database can be very time-consuming. Grover's algorithm is a fast quantum search algorithm which is also known as quantum search database algorithm. When searching a domain of size N with M solution elements the quantum search requires $\mathcal{O}(\sqrt{\frac{N}{M}})$ steps whereas classical algorithms require in average $\mathcal{O}(0.5\frac{N}{M})$ steps. [7]

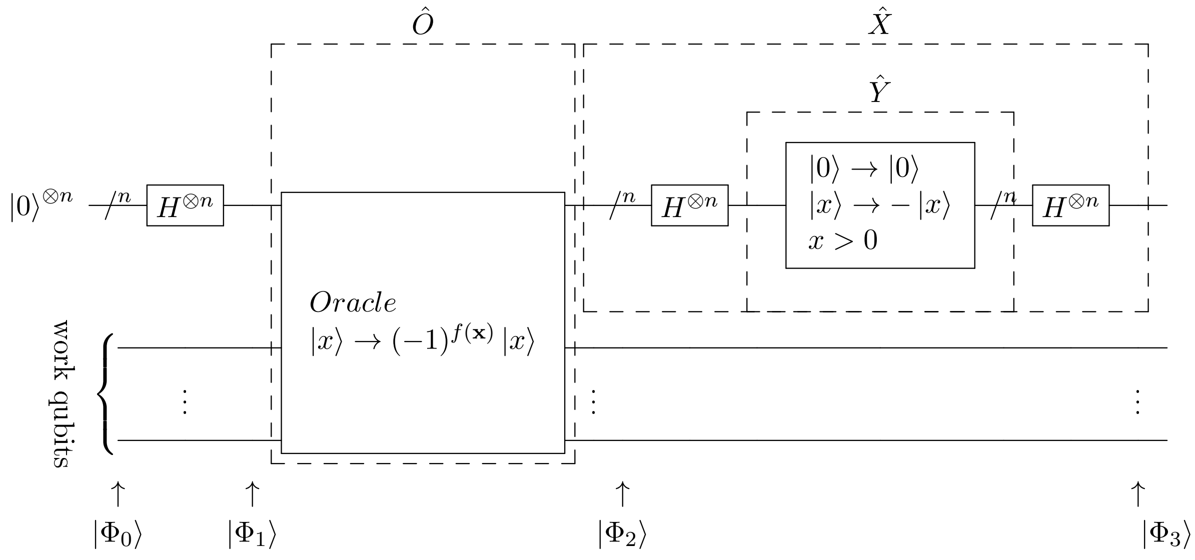


Figure 10: Grover's operator $\hat{G}$ consisting of the $\hat{O}$ and the $\hat{X}$ operator. ($\hat{G} = \hat{X}\hat{O}$) The '$/^n$' through the wire represents a set of n-qubits.

Grover's algorithm consists of two main operations. The Oracle operator $\hat{O}$ and the 'inversion about the mean' operator $\hat{X}$. Figure 10 shows the connection of these two. Basically, there is a wave function with all possible indices of the search domain entries superpositioned. The algorithm transforms the amplitudes of the search solutions in a way that their probability of getting measured increases whilst the amplitude of all non-solution indices decreases. In section 4.2 and 4.1 the Oracle $\hat{O}$ and the operator $\hat{X}$ get described. One Grover iteration is $\hat{G} = \hat{X}\hat{O}$.
Due to the fact that quantum operations always have to be reversible there is always at least one work qubit for this algorithm. (See section 2) An example of a work qubit would be the $|y\rangle$ qubit in figure 7. This qubit is only needed for the query operator to flip the amplitude for solutions of the function f.

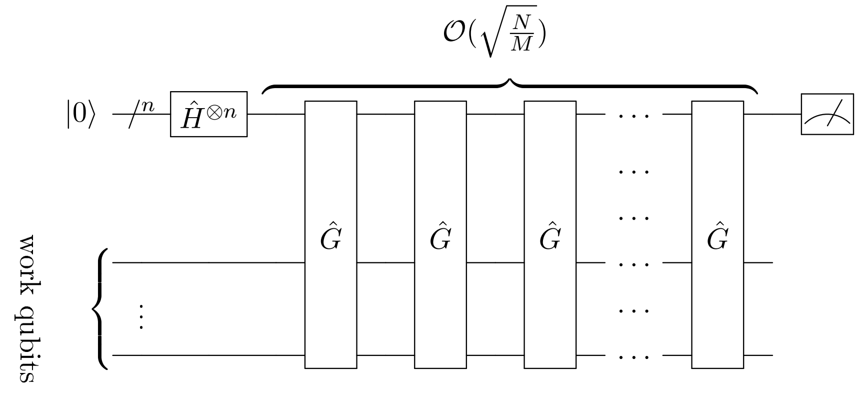


Figure 11: Grover's algorithm. The '$/^{n}$' through the wire represents a set of n-qubits.

Figure 11 shows iterations of the Grover operator. The last step is the measurement of the wave function which ideally gives the solutions. In subsection 4.3 there is a step by step analysis of the operations inside a Grover operator.

4.1 Oracle $\hat{O}$

To understand the concept of the algorithm it is possible to treat the Oracle $\hat{O}$ like a black box whose only task is to flip the amplitude of the search solutions by 180° using quantum parallelism. (See section 3.1.1) Nevertheless, this section will specify the Oracle and give some simple examples.

4.1.1 Oracle function f

As described in section 3.1, it is possible to query a function simultaneously for more than one value. For Grover's algorithm it is assumed that there is an unsorted list with a domain of 2^n entries. In addition to this, there is a function $f(\mathbf{x})$ which gives the element of the lists entry with the index $\mathbf{x}$ whereas $\mathbf{x} \in \{x_0, x_1, ..., x_{2^n-1}\}$ is the binary representation of a natural number $\mathcal{N}$. Searching is defined as finding out at which indices $\mathbf{x}$ the list entry is y. For this there needs to be another function f_y that queries $f(\mathbf{x})$ and returns 1 if the list entry equals y and 0 if it does not.

$$f_y(\mathbf{x}) = \begin{cases} 0 & \text{if } f(\mathbf{x}) \neq y \\ 1 & \text{if } f(\mathbf{x}) = y \end{cases} \tag{15}$$

These functions implemented are the preconditions for an Oracle. [1]

4.1.2 Phonebook example

M. Charemza gives an example of how the Oracle functions can be seen.[1] He wrote that the unsorted list can be compared to the entries in a phonebook. Each row on the list represents an entry in the phonebook, the telephone number, with the index $\mathbf{x}_i$. The only way to find a certain telephone number is by checking each line and finding out

[1]See: An Introduction to Quantum Computing [1], P.46

if the telephone number is equal to the wanted one. The current place where checking the line is $\mathbf{x}$ and the number we want to find is y at position $\overline{\mathbf{x}}$. The phonebook itself represents f_y and checking for each line is $f_y(\mathbf{x})$ which results in 1 if the line has the right number, and 0 if it has not.

4.2 $\hat{X}$ Operator

This operator is also called 'inversion about the mean' operator. It consists of the operations

$$\hat{X} = \hat{H}^{\otimes n}\hat{Y}\hat{H}^{\otimes n}. \tag{16}$$

The operator $\hat{Y}$ flips the amplitude for every qubit by 180° except the $|0\rangle$ qubit.

$$\hat{Y} = 2\,|0\rangle\,\langle 0| - \mathbb{1} \tag{17}$$

Applying the $\hat{H}^{\otimes n}$ on both sides gives the $\hat{X}$ operator:

$$\hat{X} = 2\,|\Phi\rangle\,\langle\Phi| - \mathbb{1}. \tag{18}$$

The state $|\Phi\rangle$ is the same as in equation 9. In section 4.4 the geometrical relevance of this operator gets discussed.

4.3 Step by step analysis

The change of the single states of one Grover iteration $\hat{G}$ will be described using the marked states at the bottom line of figure 10. The function of the single operations will be discussed in the next subsections. The starting state is:

$$|\Phi_0\rangle = |0\rangle^{\otimes n} \otimes |\text{work qubits}\rangle\,. \tag{19}$$

For reasons of convenience the $\otimes\,|\text{work qubits}\rangle$ will be left out for the other parts. After applying the Hadamard gate to all n gates there is a superpositioned state of every index.

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}|x\rangle \tag{20}$$

To understand the single steps it is assumed to know that out of $2^n = N$ elements there are M solutions to the search problem. Therefore, the wave function is split into two parts. The sum with the $'$ indicates the elements which are not a solution of the problem and the $''$ indicates the solution elements.

$$|\alpha\rangle = \sum{}' |x\rangle \quad \text{and} \quad |\beta\rangle = \sum{}'' |x\rangle \tag{21}$$

The same wave function of equation 20 with the new variables is:

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^n}}(|\alpha\rangle + |\beta\rangle). \tag{22}$$

In the next step the Oracle $\hat{O}$ gets applied and gives:

$$|\Phi_2\rangle = \frac{1}{\sqrt{2^n}}(|\alpha\rangle - |\beta\rangle). \tag{23}$$

At last the result of the $\hat{X}$ operation yields:

$$|\Phi_3\rangle = \frac{1}{\sqrt{2^n}}\left[\left(\frac{2((N-M)-M)}{N}-1\right)|\alpha\rangle + \left(\frac{2((N-M)-M)}{N}+1\right)|\beta\rangle\right]. \qquad (24)$$

By analyzing the prefactors it is already evident that the $|\beta\rangle$ states prefactor increases whilst the other decreases.

4.4 Geometrical visualization

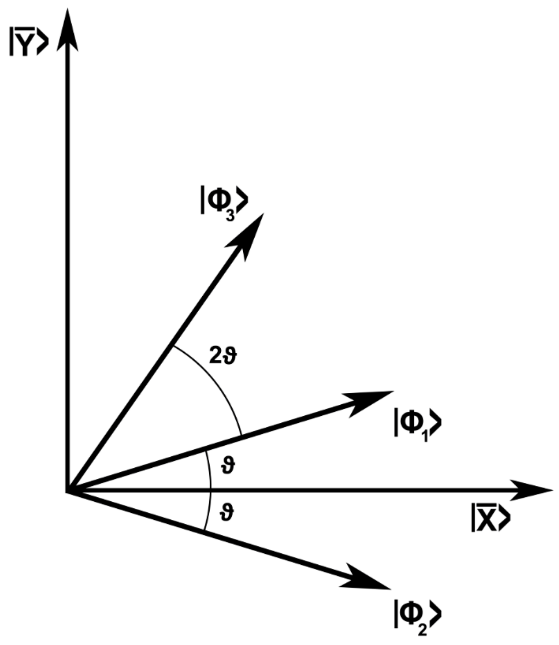


Figure 12: The geometrical visualization of one Grover iteration.

For a geometrical visualization two normalized states get defined. ($|\alpha\rangle$ and $|\beta\rangle$ are defined like in equation 21.)

$$|\overline{x}\rangle = \frac{1}{\sqrt{N-M}}|\alpha\rangle \quad \text{and} \quad |\overline{y}\rangle = \frac{1}{\sqrt{M}}|\beta\rangle \qquad (25)$$

Then $|\Phi_1\rangle$ is

$$|\Phi_1\rangle = \sqrt{\frac{N-M}{N}}|\overline{x}\rangle + \sqrt{\frac{M}{N}}|\overline{y}\rangle \qquad (26)$$

By applying the Oracle $\hat{O}$ this vector is mirrored around the $|\overline{x}\rangle$ axis.

$$|\Phi_2\rangle = \sqrt{\frac{N-M}{N}}|\overline{x}\rangle - \sqrt{\frac{M}{N}}|\overline{y}\rangle \qquad (27)$$

The $\hat{X}$ gate transforms the vector $|\Phi_2\rangle$ into this state:

$$|\Phi_3\rangle = \underbrace{\left(\frac{2N-4M}{N}-1\right)\sqrt{\frac{N-M}{N}}}_{=C_1}|\overline{x}\rangle + \underbrace{\left(\frac{2N-4M}{N}+1\right)\sqrt{\frac{M}{N}}}_{=C_2}|\overline{y}\rangle \qquad (28)$$

Because all of the states $|\Phi_i\rangle$ are normalized it is possible to say that

$$\sqrt{\frac{N-M}{N}} = cos(\vartheta) \quad \text{and} \quad \sqrt{\frac{M}{N}} = sin(\vartheta). \tag{29}$$

That transforms $|\Phi_1\rangle$ into

$$|\Phi_1\rangle = cos(\vartheta)\,|\overline{x}\rangle + sin(\vartheta)\,|\overline{y}\rangle \tag{30}$$

and $|\Phi_3\rangle$ into

$$|\Phi_3\rangle = \underbrace{cos(3\vartheta)}_{=C_1}|\overline{x}\rangle + \underbrace{sin(3\vartheta)}_{=C_2}|\overline{y}\rangle\,. \tag{31}$$

This makes it graphically comprehensible that the $\hat{X}$ gate mirrors the $|\Phi_2\rangle$ state around the $|\Phi_1\rangle$ state. (Figure 12 shows the vector states $|\Phi_i\rangle$) Therefore it can be said that:

$$\hat{G}^k\,|\Phi_1\rangle = cos((2k+1)\vartheta)\,|\overline{x}\rangle + sin((2k+1)\vartheta)\,|\overline{y}\rangle\,. \tag{32}$$

In this base one Grover iteration equals the matrix:

$$\hat{G} = \begin{pmatrix} cos(2\vartheta) & -sin(2\vartheta) \\ sin(2\vartheta) & cos(2\vartheta) \end{pmatrix} \tag{33}$$

The probability to measure one of the solutions after k iterations is:

$$P_{sol} = |sin((2k+1)\vartheta)|^2 \tag{34}$$

4.5 Iteration and time analysis

With equation 32 the number of necessary iterations for reasonable measurement results can be found out. It is the goal to find k the number of iterations where the vectors projection on the $|\overline{y}\rangle$ axis is maximized. Therefore, the angle $(2k+1)\vartheta$ needs to be as close to $\frac{\pi}{2}$ (90°) as possible. Assuming that $M \leq \frac{N}{2}$ the following approximation is used:

$$\vartheta = sin^{-1}\left(\sqrt{\frac{M}{N}}\right) \approx \sqrt{\frac{M}{N}}. \tag{35}$$

Setting $(2k+1)\sqrt{\frac{M}{N}} = \frac{\pi}{2}$ gives an upper bound for k.

$$k \leq \frac{\pi}{4}\sqrt{\frac{N}{M}} \tag{36}$$

This shows that Grover's algorithm needs to perform $\mathcal{O}(\sqrt{\frac{N}{M}})$ iterations to yield a solution of the search with high probability. [8]
If $M \geq \frac{N}{2}$ the number of iterations needed by the search algorithm increases with M. There are search tasks where it is not known whether the number of solutions M is greater or smaller than $\frac{N}{2}$. There is an approach which helps to prevent the case that the iterations increase. The idea is to double the number of elements in the search space by adding an extra qubit. ($2N = 2^{n+1}$) As a consequence of that, always less than half of

the elements are solutions. [3] Running the search algorithm with that augmented input register gives a new upper bound for the iterations k:

$$k \leq \frac{\pi}{4}\sqrt{\frac{2N}{M}} \tag{37}$$

This approach still increases the number of iterations but helps keeping track of how many iterations are needed.

4.6 Logic circuit

A boolean function of a logic circuit has the same constraints and properties as the Oracle function f. A boolean function maps: $f_B : B^n \rightarrow B$, where $B = \{0, 1\}$. When implementing a boolean function as the Oracle function, the truth table of the boolean function can be seen as the list with all the 2^n binary combinations as an index. The list entry would be 0 if the index is not a solution of the boolean function and 1 if the index is a solution. Due to the outcome which is only 1 and 0 the boolean function does not need a further constraint like f_y. It always searches for the entry 1. As an example, suppose to have the boolean function $f_B = (a \wedge b) \wedge c$.

Table 1: The truth table of the boolean function $f_B = (a \wedge b) \wedge c$.

a	b	c	f_B
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Implementing this function into an Oracle looks like that:

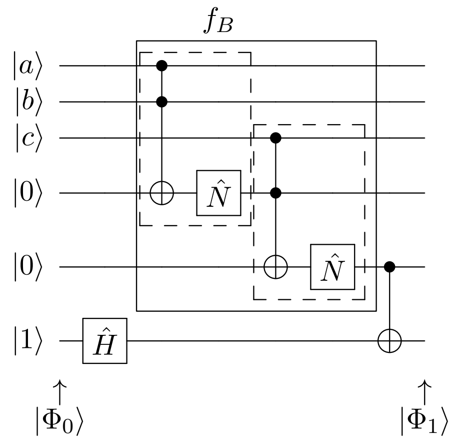


Figure 13: Boolean function f_B implemented into the Oracle operator. The dashed line boxes are the AND-gates. (See figure 6 a)

The starting state is:

$$|\Phi_0\rangle = |abc\rangle \otimes \underbrace{|001\rangle}_{\text{work qubits } = |x\rangle} \tag{38}$$

whereas the last three qubits are work qubits. After the application of the new Oracle gate the state is:

$$|\Phi_1\rangle = \underbrace{(-1)^{f_B(a,b,c)}\,|a,b,c\rangle}_{\text{query qubits}}\,|x'\rangle\,. \tag{39}$$

This is a valid Oracle wavefunction, whereas only the query qubits are relevant for the consecutive algorithm.

Matlab simulation:
The evolution of the wavefunction after each Grover iteration can be simulated with a Matlab program.[1] The Oracle, as it is in figure 13, is used for this Grover's algorithm simulation. In this case it is known that there is a search space of 8 elements with one solution. (See truth table 1) Through this it is possible to predict the number of Grover iterations needed to find a solution with the help of equation 36 which was derived in section 4.5.

$$k \leq \frac{\pi}{4}\sqrt{\frac{N}{M}} = 2.22 \Rightarrow 2 \text{ iterations} \tag{40}$$

There will be listings of the code output for each iteration. The starting state is equivalent to $|\Phi_1\rangle$ of figure 10 and the last three qubits are the work qubits.

```
|phi1> = 0.35355|000001> + 0.35355|001001> + 0.35355|010001>
+ 0.35355|011001> + 0.35355|100001> + 0.35355|101001>
+ 0.35355|110001> + 0.35355|111001>
```

[1]E-Mail tobiaskugel@student.tugraz.at for the source code and simulation graphs

Applying the Grover operator gives:

```
G∗|phi1> = 0.17678|000001> + 0.17678|001001> + 0.17678|010001>
+ 0.17678|011001> + 0.17678|100001> + 0.17678|101001>
+ 0.17678|110001> + 0.88388|111001>
```

The probability to measure the already known solution $|111\rangle |x'\rangle$ is already $P_{sol} = |0.88388|^2 \approx 0.78$ increased after one operation. The second operation gives:

```
G∗G∗|phi1> = − 0.088388|000001> − 0.088388|001001> − 0.088388|010001>
− 0.088388|011001> − 0.088388|100001> − 0.088388|101001>
− 0.088388|110001> + 0.97227|111001>
```

The probability of measuring the solution $|111\rangle |x'\rangle$ is now $P_{sol} = |0.97227|^2 \approx 0.95$ which is already a reasonable value for the predicted two operations. Applying the operator one more time gives:

```
G∗G∗G∗|phi1> = −0.30936|000001> − 0.30936|001001> − 0.30936|010001>
− 0.30936|011001> − 0.30936|100001> − 0.30936|101001>
− 0.30936|110001> + 0.57452|111001>
```

The probability to measure the solution $P_{sol} = |0.57452|^2 \approx 0.33$ decreases again which means that doing more operations does not give a better result. By knowing the number of solutions and the size of the search space it is possible to find the solutions of the boolean function even without any information about its composition before.
A more complex problem occurs if there is an Oracle with a completely unknown boolean function implemented. The goal is to find all the queries which are solutions to the, at this point unknown, function. With the help of Grover's algorithm it is possible to find the queries which are a solution of the problem with less operations than in a classical way. For this application of the algorithm the number of solutions is not known in advance. The so-called phase estimation technique gives a prediction about the number of it. It will be discussed in section 4.7.
Another application for Grover's algorithm is finding the solutions of a 3-SAT[1] problem. Finding solutions to these problems within a reasonable amount of computation time is still one of the most challenging problems in computer science. S.T. Cheng and M. H. Tao wrote an article about making Grover's algorithm work for solving 3-SAT problems. [9]
A big disadvantage of using boolean functions in quantum algorithms is that the number of total qubits grows very fast. For example, the simple circuit of figure 13 needs already two work qubits to be realized. The more qubits there are in a quantum system the harder it is to realize it in an experiment.

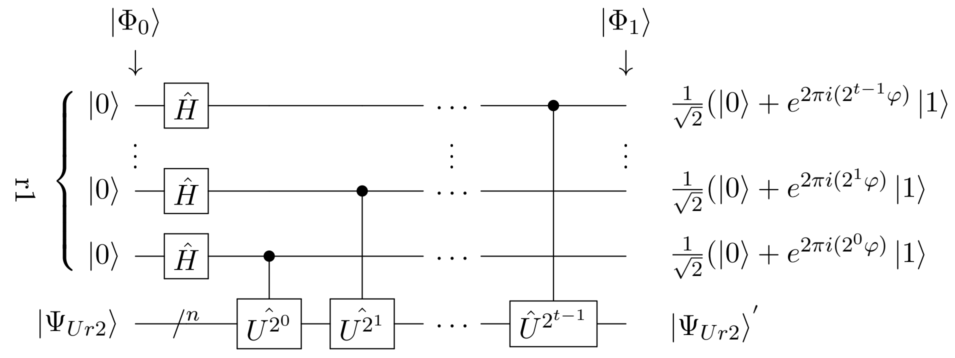
4.7 Quantum counting

There are searching tasks where the number of solutions M is not known in advance. Combining Grover's algorithm with the phase estimation technique, which is based upon the quantum Fourier transformation, gives us the possibility to estimate the number of solutions M.

[1]A complex boolean function. SAT is an abbreviation for satisfiability.

4.7.1 Phase estimation

Assume there is an unitary operator $\hat{U}$ with the eigenvector $|u\rangle$ and the eigenvalue $e^{2\pi i\varphi}$. The goal of the phase estimation algorithm is to estimate φ. The algorithm uses two registers. The first contains t qubits with the initial state $|0\rangle$. The size of this register depends on the number of digits of accuracy we want to estimate φ and the probability we wish the phase estimation to be successful. The second register needs as many qubits as the operator $\hat{U}$ needs to operate. The phase estimation is performed in three stages. The first contains out of these operations:



$|\Phi_0\rangle$
$|\Phi_1\rangle$
r1
$|0\rangle$
$\hat{H}$
$|\Psi_{Ur2}\rangle$
n
$\hat{U^{2^0}}$
$\hat{U^{2^1}}$
$\hat{U}^{2^{t-1}}$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{t-1}\varphi)}|1\rangle$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{1}\varphi)}|1\rangle$
$\frac{1}{\sqrt{2}}(|0\rangle + e^{2\pi i(2^{0}\varphi)}|1\rangle$
$|\Psi_{Ur2}\rangle'$

Figure 14: The first stage of the phase estimation circuit. $r1$ marks the first register with t qubits. The state $|\Psi_{Ur2}\rangle$ is the state of the second register. The '$/^n$' through the wire represents a set of n-qubits.

As seen in section 4.4, it is possible to write every state of the second register of figure 14 as a superpositioned state of the eigenvectors of the operator $\hat{U}$. The first stage of this algorithm maps the phasefactors of the new state onto the qubits of the first register. The first state $|\Phi_0\rangle$ of figure 14 is:

$$|\Phi_0\rangle = |000\rangle \otimes |\Psi_{Ur2}\rangle \tag{41}$$

and after applying the gates we yield:

$$|\Phi_1\rangle = \frac{1}{\sqrt{2^t}} \sum_{x=0}^{2^t-1} e^{2\pi i\varphi k} |x\rangle \otimes |\Psi_{Ur2}\rangle' \tag{42}$$

$|\Psi_{Ur2}\rangle'$ is the state of the second register after the application of the gates. The second stage is an application of an inverse Fourier transform on the first register. The third and final stage is to measure the state of the first register and evaluate φ. [3]

4.7.2 Grover quantum counting

One Grover iteration in the basis of $|\overline{x}\rangle$ and $|\overline{y}\rangle$ which was used for the geometrical visualization in section 4.4 is equal to the matrix (equation 33):

$$\hat{G} = \begin{pmatrix} cos(2\vartheta) & -sin(2\vartheta) \\ sin(2\vartheta) & cos(2\vartheta) \end{pmatrix}$$

The eigenvalues of this matrix are

$$\lambda_1 = e^{i2\vartheta} \quad \text{and} \quad \lambda_2 = e^{i(2\pi - 2\vartheta)}. \tag{43}$$

For an easier analysis of the next calculations the elements of the second register are doubled by adding one qubit. Using equation 29: $sin^2(\vartheta) = \frac{M}{2N}$ gives the correlation between the angle and the search elements M and N. Applying the phase estimation circuit of figure 14 gives an estimate of ϑ to an accuracy of m bits, with a probability of success of at least $1 - \epsilon$. To achieve this there need to be $t = m + [log(2 + \frac{1}{2\epsilon})]$ qubits in the first register.

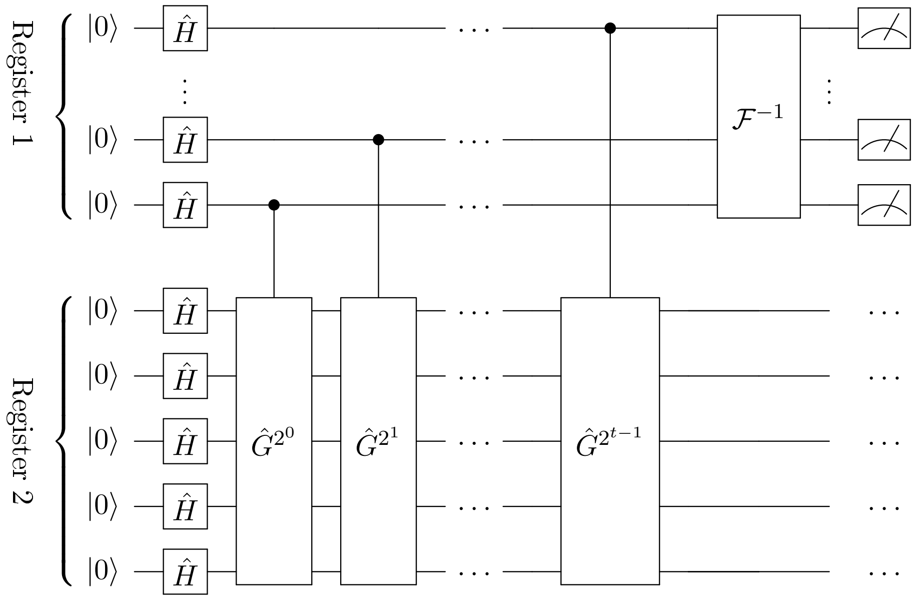


Figure 15: The circuit for finding the number of solutions of a search problem. *Register* 1 marks the first register with t qubits. *Register* 2 marks the Grover algorithm register with n+1 qubits. $\mathcal{F}^{-1}$ is the inverse Fourier transform.

Figure 15 shows the use of Grover operations in a phase estimation algorithm like figure 14. The measurement gives an estimate of the angle ϑ. Using the correlation $sin^2(\vartheta) = \frac{M}{2N}$ the number of answers is:

$$M = sin^2(\vartheta)2N. \tag{44}$$

The absolute error of that value has the upper bound

$$|\Delta M| < \left(\sqrt{2MN} + \frac{N}{2^{m+1}}\right) 2^{-m}. \text{ [3]} \tag{45}$$

References

[1] M. Charemza. *An introduction to quantum computing.* - (2005).

[2] *Prof. Dr. Arrigoni, Einführung in Quantencomputer.* - (2010).

[3] M. A. Nielsen, I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010. doi:10.1017/CBO9780511976667

[4] R. B. Griffiths. *Hilbert Space Quantum Mechanics.* - (2014).

[5] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, H. Weinfurter. *Elementary gates for quantum computation.* Phys. Rev. A **52** (1995) 3457. doi:10.1103/PhysRevA.52.3457

[6] Y. Shi. *Both Toffoli and controlled-NOT Need Little Help to Do Universal Quantum Computing.* Quantum Info. Comput. **3** (2003) 84.

[7] J. A. Jones, M. Mosca, R. H. Hansen. *Implementation of a quantum search algorithm on a quantum computer.* Nature **393** (1998) 344. doi:10.1038/30687

[8] M. Boyer, G. Brassard, P. Høyer, A. Tapp. *Tight Bounds on Quantum Searching.* Fortschritte der Physik **46** (1998) 493. doi:10.1002/(SICI)1521-3978(199806)46:4

[9] S.-T. Cheng, M.-H. Tao. *Quantum cooperative search algorithm for 3-SAT.* Journal of Computer and System Sciences **73** (2007) 123 . doi:10.1016/j.jcss.2006.09.003