Introduction to Quantum Computing

Module 2 — Part I

Ugo Dal Lago

ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

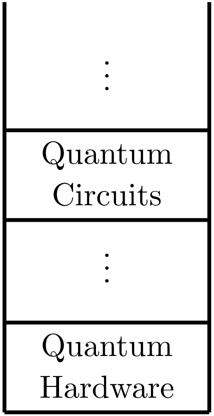
Inria informatiques mathématiques

Academic Year 2024/2025

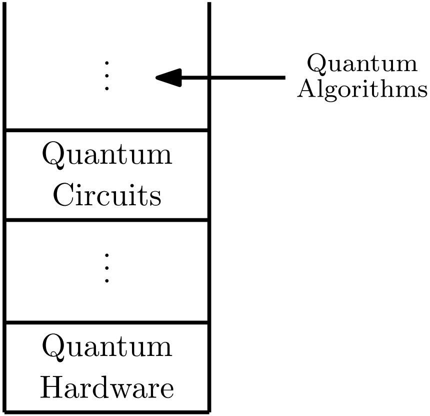
Part I

This Module: Contents and Motivations

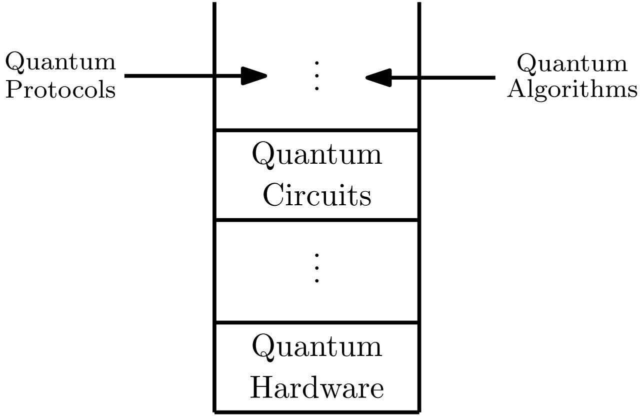
What?



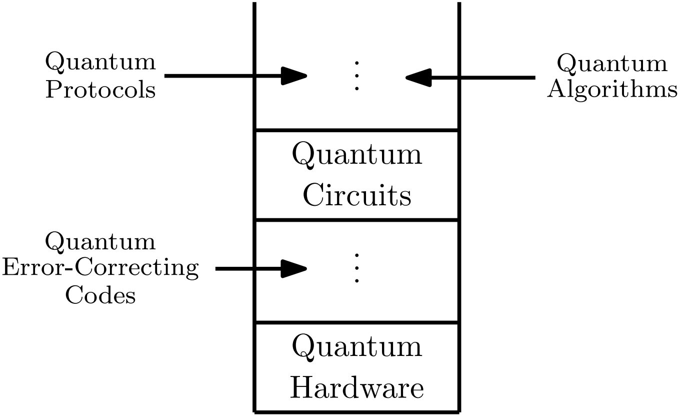
What?



What?


Quantum Protocols
Quantum Algorithms
Quantum Circuits
Quantum Hardware

What?


Quantum Protocols
Quantum Algorithms
Quantum Circuits
Quantum Error-Correcting Codes
Quantum Hardware

How?

- The most basic concepts are described on these **slides**, which will be made available to all students.
  - Please do not expect all details to be present in the slides, which are of course meant to be just a summary.

How?

- The most basic concepts are described on these **slides**, which will be made available to all students.
  - Please do not expect all details to be present in the slides, which are of course meant to be just a summary.
- Some details will be presented at an interactive **whiteboard**, whose transcripts will be made available themselves to all students.
  - That will be extremely useful when giving formal *proofs*, of which we will se just very few.

How?

- The most basic concepts are described on these **slides**, which will be made available to all students.
  - Please do not expect all details to be present in the slides, which are of course meant to be just a summary.
- Some details will be presented at an interactive **whiteboard**, whose transcripts will be made available themselves to all students.
  - That will be extremely useful when giving formal *proofs*, of which we will se just very few.
- Many nice **textbooks** about quantum computing which cover all what we say are available. We will follow the very nice textbook by Mermin:
  - N. David Mermin. *Quantum Computer Science: An Introduction.* Cambridge University Press, New York, NY, USA. 2007.

How?

- The most basic concepts are described on these **slides**, which will be made available to all students.
    - Please do not expect all details to be present in the slides, which are of course meant to be just a summary.
- Some details will be presented at an interactive **whiteboard**, whose transcripts will be made available themselves to all students.
    - That will be extremely useful when giving formal *proofs*, of which we will se just very few.
- Many nice **textbooks** about quantum computing which cover all what we say are available. We will follow the very nice textbook by Mermin:
    - N. David Mermin. *Quantum Computer Science: An Introduction.* Cambridge University Press, New York, NY, USA. 2007.
- Please feel free to ask any question during lectures, or to write me at `ugo.dallago@unibo.it`

Why?

- Quantum computing is considered to have very strong **potential** for revolutionize the way certain computational problems are solved, and this is why there is a strong interest around it.

Why?

- Quantum computing is considered to have very strong **potential** for revolutionize the way certain computational problems are solved, and this is why there is a strong interest around it.
- It's important to understand **where** this potential comes from.
  - You know what a quantum circuit **is**.
  - But perhaps you do not know in which sense quantum circuits can be **helpful**.

Why?

- Quantum computing is considered to have very strong **potential** for revolutionize the way certain computational problems are solved, and this is why there is a strong interest around it.
- It's important to understand **where** this potential comes from.
  - You know what a quantum circuit **is**.
  - But perhaps you do not know in which sense quantum circuits can be **helpful**.
- But **beware**: most of what we are going to say is about the *quantum circuit model*, and nobody currently knows whether this model is realistic when the number of qubits grows.
  - That is why we will also talk about quantum error correcting codes.

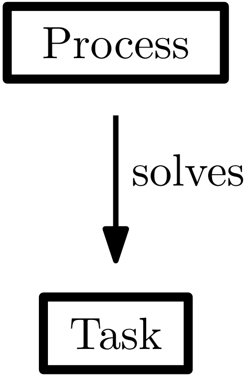
This Slideset

- A glimpse of **circuit complexity**.
  - Classical circuit complexity.
  - Quantum circuit complexity.
  - How all this relates to Turing machines.
- Some basic **quantum algorithmics**.
  - A Recap on Deutsch and Bernstein-Vazirani Algorithms.
  - Simon Algorithm.
  - Shor Algorithm.
    - We will try to give most details, but we will perhaps skip some of them
  - Grover Algorithm

Part II

Basic Circuit Complexity

Tasks and Processes



Tasks

- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.

Tasks

- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.
- The set of *all* strings of length n from Σ indicated as Σ^n.

Tasks

- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.
- The set of *all* strings of length n from Σ indicated as Σ^n.
- A particularly interesting alphabet is $\Sigma = \{0, 1\}$.

Tasks

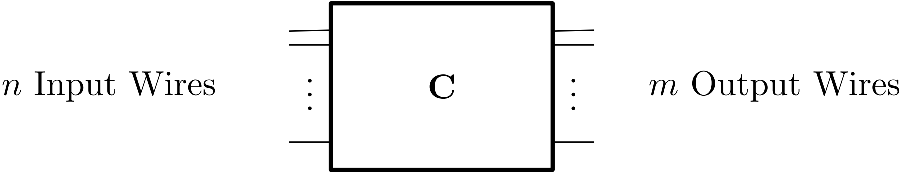
- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.
- The set of *all* strings of length n from Σ indicated as Σ^n.
- A particularly interesting alphabet is $\Sigma = \{0, 1\}$.
- We are interested in tasks formulated as mathematical functions from $\{0, 1\}^n$ to $\{0, 1\}^m$. The set of all such **boolean functions** is indicated as $\mathcal{F}_m^n$.

Tasks

- A string from an alphabet Σ is any ordered, finite sequence of symbols from Σ.
- The set of *all* strings of length n from Σ indicated as Σ^n.
- A particularly interesting alphabet is $\Sigma = \{0, 1\}$.
- We are interested in tasks formulated as mathematical functions from $\{0, 1\}^n$ to $\{0, 1\}^m$. The set of all such **boolean functions** is indicated as $\mathcal{F}^n_m$.
- Sometimes, we are also interested in **parametric boolean functions** namely in functionals which turn a function in $\mathcal{F}^p_q$ to a function in $\mathcal{F}^n_m$.
  - An example when $p = q = 2$, $n = 0$ and $m = 1$ is the task of checking whether the input function $f \in \mathcal{F}^p_q$ returns 00 for all possible inputs.

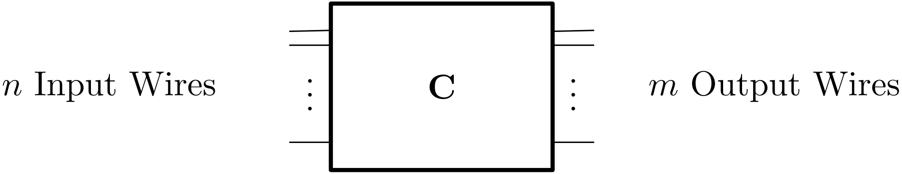
Classical Processes

- A *classical process* will be a **boolean circuit** :



Classical Processes

- A *classical process* will be a **boolean circuit** :

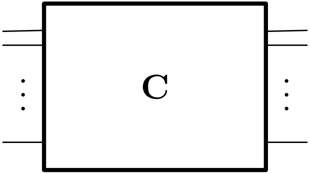


- One such circuit $\mathbf{C}$ is said to **compute** the boolean function $f_{\mathbf{C}}$ in $\mathcal{F}^n_m$ such that $f_{\mathbf{C}}(x) = y$ precisely when $\mathbf{C}$ when executed on x on the n input wires produces y on its m output wires.

Classical Processes

- A *classical process* will be a **boolean circuit** :

n Input Wires

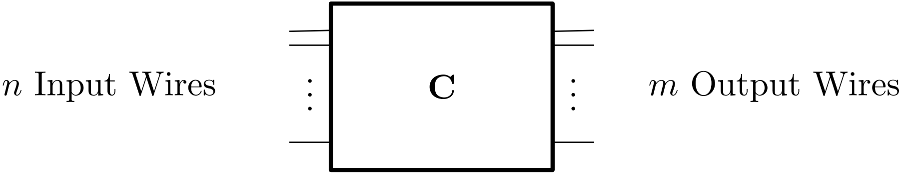


m Output Wires

- One such circuit $\mathbf{C}$ is said to **compute** the boolean function $f_{\mathbf{C}}$ in $\mathcal{F}_m^n$ such that $f_{\mathbf{C}}(x) = y$ precisely when $\mathbf{C}$ when executed on x on the n input wires produces y on its m output wires.
- We can generalize the view above to parametric boolean functions by considering boolean circuits with a special gate corresponding to the parameter function in $\mathcal{F}_q^p$, called a parametric boolean circuits.

Classical Processes

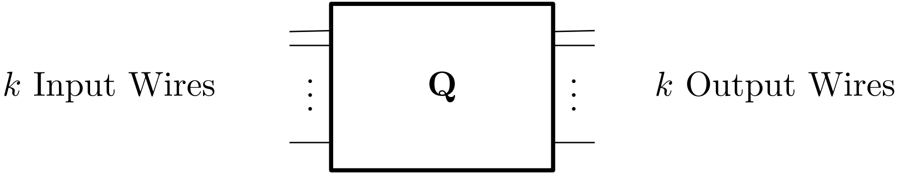
- A *classical process* will be a **boolean circuit** :



- One such circuit $\mathbf{C}$ is said to **compute** the boolean function $f_{\mathbf{C}}$ in $\mathcal{F}_m^n$ such that $f_{\mathbf{C}}(x) = y$ precisely when $\mathbf{C}$ when executed on x on the n input wires produces y on its m output wires.
- We can generalize the view above to parametric boolean functions by considering boolean circuits with a special gate corresponding to the parameter function in $\mathcal{F}_q^p$, called a parametric boolean circuits.
- The *size* of $\mathbf{C}$ is a measure of how many resources $\mathbf{C}$ would consume when executed.
  - The size of $\mathbf{C}$, that we write as $|\mathbf{C}|$, can be measured in different ways, e.g., as the *number of gates*, as the *width*, or as the *depth*.
  - An interesting size parameter of any parametric boolean circuit $\mathbf{C}$ is the number of calls to the special gate computing the parameter function, which we indicate as $|\mathbf{C}|_{\mathsf{P}}$.

Quantum Processes

- Similarly, a *quantum process* is just a **quantum circuit** :



- We can generalize all what we have said about boolean circuits, keeping in mind that
  - Quantum circuits, due to measurements, can have a probabilistic behaviour. We then say that one such circuit *computes* a function $f_{\mathbf{Q}}$ in $\mathcal{F}_m^n$ if it does so up to a *small* probability of error.
  - The circuit $\mathbf{Q}$, if we do not consider measurements, must be reversible. It is thus convenient to assume that for such a circuit to compute $f \in \mathcal{F}_m^n$, it must be that $\mathbf{Q}$ has $n+m+r$ inputs and $n+m+r$ outputs an that, with high probability, on input $|x\rangle \otimes |y\rangle \otimes \left|0^{\otimes r}\right\rangle$ it produces (with high probability) the value $|x\rangle \otimes |f(x) \oplus y\rangle \otimes |\psi\rangle$ in output.
    - The r qubits are auxiliary, and their output value is not relevant.

The Fundamental Question

- The question now is the following: Is there any (parametric) function f such that there is a quantum circuit $\mathbf{Q}$ computing f such that all (known) boolean circuits $\mathbf{C}$ computing f are such that $|\mathbf{Q}| < |\mathbf{C}|$?

The Fundamental Question

- The question now is the following: Is there any (parametric) function f such that there is a quantum circuit $\mathbf{Q}$ computing f such that all (known) boolean circuits $\mathbf{C}$ computing f are such that $|\mathbf{Q}| < |\mathbf{C}|$?
- We will give some examples of such functions:
  - Starting from **toy** functions, for which you have already seen the corresponding algorithms.
  - And ending in **relevant** functions.
  - Many of them will be **parametric** functions.

A Bridge to "Uniform" Complexity Theory

- In classical computation, a *circuit family* is a family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ such that for every $n \in \mathbf{N}$, the circuit $\mathbf{C}_n$ computes a function in $\mathcal{F}_1^n$.
  - Such a circuit family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ can thus be seen as computing a function $f_{\{\mathbf{C}_n\}} : \{0,1\}^* \to \{0,1\}$ where $\{0,1\}^* = \cup_{n=0}^{\infty}\{0,1\}^n$.

A Bridge to "Uniform" Complexity Theory

- In classical computation, a *circuit family* is a family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ such that for every $n \in \mathbf{N}$, the circuit $\mathbf{C}_n$ computes a function in $\mathcal{F}_1^n$.
  - Such a circuit family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ can thus be seen as computing a function $f_{\{\mathbf{C}_n\}} : \{0,1\}^* \to \{0,1\}$ where $\{0,1\}^* = \cup_{n=0}^{\infty}\{0,1\}^n$.
- Again, something very similar can be said for quantum circuits.

A Bridge to "Uniform" Complexity Theory

- In classical computation, a *circuit family* is a family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ such that for every $n \in \mathbf{N}$, the circuit $\mathbf{C}_n$ computes a function in $\mathcal{F}_1^n$.
  - Such a circuit family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ can thus be seen as computing a function $f_{\{\mathbf{C}_n\}} : \{0,1\}^* \to \{0,1\}$ where $\{0,1\}^* = \cup_{n=0}^{\infty}\{0,1\}^n$.
- Again, something very similar can be said for quantum circuits.
- The class of languages (i.e., subsets of $\{0,1\}^*$) which can be computed by classic circuit families having polynomially bounded size *and* being generated in polynomial time through algorithms is precisely $\mathtt{P}$.

A Bridge to "Uniform" Complexity Theory

- In classical computation, a *circuit family* is a family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ such that for every $n \in \mathbf{N}$, the circuit $\mathbf{C}_n$ computes a function in $\mathcal{F}_1^n$.
  - Such a circuit family $\{\mathbf{C}_n\}_{n\in\mathbf{N}}$ can thus be seen as computing a function $f_{\{\mathbf{C}_n\}} : \{0,1\}^* \to \{0,1\}$ where $\{0,1\}^* = \cup_{n=0}^{\infty}\{0,1\}^n$.
- Again, something very similar can be said for quantum circuits.
- The class of languages (i.e., subsets of $\{0,1\}^*$) which can be computed by classic circuit families having polynomially bounded size *and* being generated in polynomial time through algorithms is precisely $\mathtt{P}$.
- A very similar result holds for polysize *quantum* circuit families generated in *classical* polynomial time and a class called $\mathtt{BQP}$

Part III

Basic Quantum Algorithmics

The Deutsch Problem

- In the Deutsch Problem, we are interested in determining whether a parameter function $f \in \mathcal{F}_1^1$ is such that $f(0) = f(1)$. In doing so, we can use a gate computing f.

The Deutsch Problem

- In the Deutsch Problem, we are interested in determining whether a parameter function $f \in \mathcal{F}_1^1$ is such that $f(0) = f(1)$. In doing so, we can use a gate computing f.
- Classically, we do not have any hope of getting the job done without invoking the f *twice*.

The Deutsch Problem

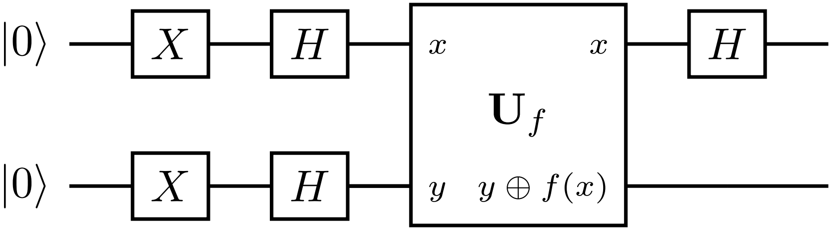
- In the Deutsch Problem, we are interested in determining whether a parameter function $f \in \mathcal{F}_1^1$ is such that $f(0) = f(1)$. In doing so, we can use a gate computing f.
- Classically, we do not have any hope of getting the job done without invoking the f *twice*.
- In quantum computing, one can do strictly better, and invoke f just *once*!

The Deutsch Problem

- In the Deutsch Problem, we are interested in determining whether a parameter function $f \in \mathcal{F}_1^1$ is such that $f(0) = f(1)$. In doing so, we can use a gate computing f.
- Classically, we do not have any hope of getting the job done without invoking the f *twice*.
- In quantum computing, one can do strictly better, and invoke f just *once*!
- In doing so, we will assume that there is a gate $\mathbf{U}_f$ computing f without using any auxiliary qubit:

$$\mathbf{U}_f(|x\rangle\,|y\rangle) = |x\rangle\,|y \oplus f(x)\rangle$$

The Deutsch Circuit $\mathbf{C}_{\mathsf{DEUTSCH}}$

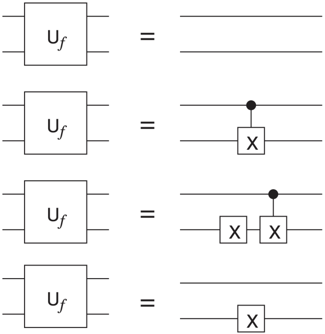


Correctness

$$(\mathbf{H} \otimes \mathbf{1})\mathbf{U}_f(\mathbf{H} \otimes \mathbf{H})(\mathbf{X} \otimes \mathbf{X})(|0\rangle\,|0\rangle)$$

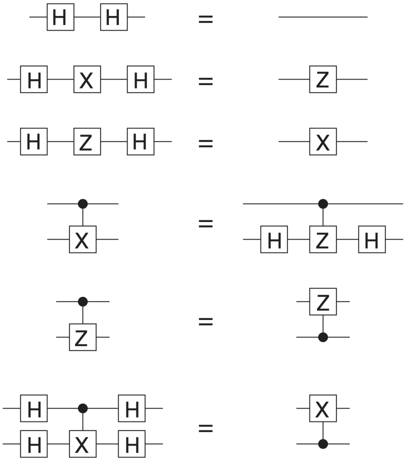
$$= \begin{cases} |1\rangle \frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\tilde{f}(0)\rangle\Big), & f(0) = f(1), \\ |0\rangle \frac{1}{\sqrt{2}}\Big(|f(0)\rangle - |\tilde{f}(0)\rangle\Big), & f(0) \neq f(1). \end{cases}$$

Equational Interpretation

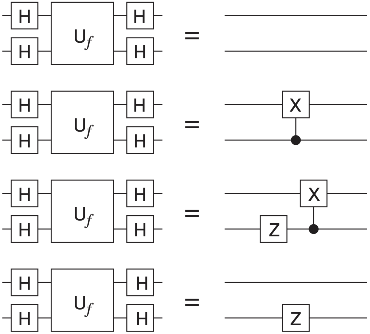


$f(0)$	$f(1)$
0	0
0	1
1	0
1	1

Equational Interpretation



Equational Interpretation



$f(0)$	$f(1)$
0	0
0	1
1	0
1	1

The Bernstein-Vazirani Problem

- Given $a, x \in \{0,1\}^n$. We write $a \cdot x$ for the bitwise inner-product of a and x:

$$a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n$$

The Bernstein-Vazirani Problem

- Given $a, x \in \{0,1\}^n$. We write $a \cdot x$ for the bitwise inner-product of a and x:

$$a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n$$

- The Bernstein-Vazirani problem has to do with parametric functions whose input function $f_a \in \mathcal{F}_1^n$ is such that $f_a(x) = a \cdot x$.

The Bernstein-Vazirani Problem

- Given $a, x \in \{0,1\}^n$. We write $a \cdot x$ for the bitwise inner-product of a and x:

$$a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n$$

- The Bernstein-Vazirani problem has to do with parametric functions whose input function $f_a \in \mathcal{F}_1^n$ is such that $f_a(x) = a \cdot x$.
- Simply, we want to determine a invoking f_a, but trying to minimize the amount of times we invoke the function.

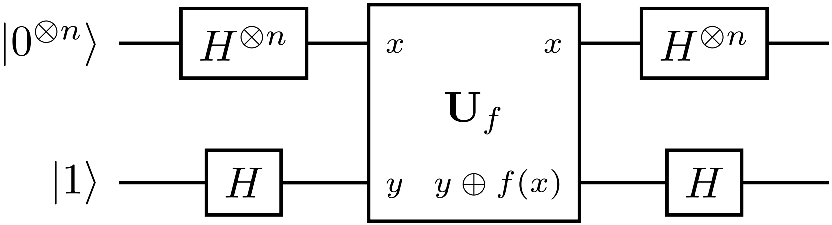
The Bernstein-Vazirani Problem

- Given $a, x \in \{0,1\}^n$. We write $a \cdot x$ for the bitwise inner-product of a and x:

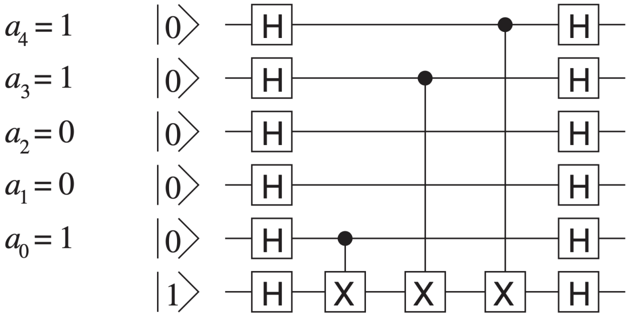
$$a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n$$

- The Bernstein-Vazirani problem has to do with parametric functions whose input function $f_a \in \mathcal{F}_1^n$ is such that $f_a(x) = a \cdot x$.
- Simply, we want to determine a invoking f_a, but trying to minimize the amount of times we invoke the function.
- Classically, we cannot do that by invoking f_a less than n times.
- There is a quantum circuit which invokes f_a **just once**.

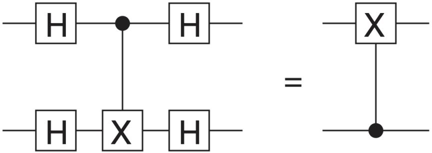
The Bernstein-Vazirani Circuit $\mathbf{C}_{\mathsf{BV}}$



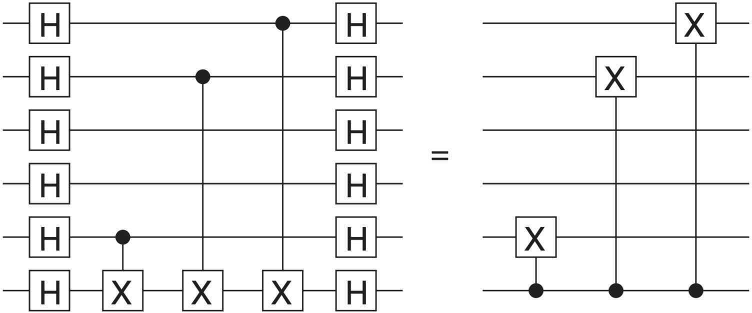
Equational Interpretation



Equational Interpretation



Equational Interpretation



Part IV

The Simon Problem

The Simon Problem

- Like in the Bernstein-Vazirani problem, in the Simon Problem we deal with a function f which depends on $a \in \{0,1\}^n$.
- The function f is defined differently, and is in $\mathcal{F}_n^n$. There are many such functions for the same $a \in \{0,1\}^n$.

The Simon Problem

- Like in the Bernstein-Vazirani problem, in the Simon Problem we deal with a function f which depends on $a \in \{0,1\}^n$.
- The function f is defined differently, and is in $\mathcal{F}_n^n$. There are many such functions for the same $a \in \{0,1\}^n$.
- We only know that f is *periodic modulo* a, namely that

$$f(x) = f(y) \iff y = x \oplus a \iff x \oplus y = a$$

for every $x, y \in \{0,1\}^n$.

The Simon Problem

- Like in the Bernstein-Vazirani problem, in the Simon Problem we deal with a function f which depends on $a \in \{0,1\}^n$.
- The function f is defined differently, and is in $\mathcal{F}_n^n$. There are many such functions for the same $a \in \{0,1\}^n$.
- We only know that f is *periodic modulo* a, namely that

$$f(x) = f(y) \iff y = x \oplus a \iff x \oplus y = a$$

  for every $x, y \in \{0,1\}^n$.
- Like in the Bernstein-Vazirani problem, we want to compute parametric functions whose input function $f \in \mathcal{F}_n^n$ is periodic modulo a in the sense above, and we want to determine a *minimizing* the amount of times one invokes f.

The Simon Problem

- Like in the Bernstein-Vazirani problem, in the Simon Problem we deal with a function f which depends on $a \in \{0,1\}^n$.
- The function f is defined differently, and is in $\mathcal{F}_n^n$. There are many such functions for the same $a \in \{0,1\}^n$.
- We only know that f is *periodic modulo* a, namely that

$$f(x) = f(y) \iff y = x \oplus a \iff x \oplus y = a$$

  for every $x, y \in \{0,1\}^n$.
- Like in the Bernstein-Vazirani problem, we want to compute parametric functions whose input function $f \in \mathcal{F}_n^n$ is periodic modulo a in the sense above, and we want to determine a *minimizing* the amount of times one invokes f.
- Classically, one cannot succeed (with high probability) unless one invokes f around 2^n times, so **exponentially many** times.
- There are, instead, quantum circuits which succeed in determining a by invoking f an amount of time which is **linear** in n.

Classical Strategy

- Any classical algorithm querying a function $f \in \mathcal{F}_n^n$ which is periodic modulo a *accumulates knowledge* as follows:
  - If the queries done so far, namely

    $$y_1 = f(x_1), \ldots, y_k = f(x_k)$$

    are such that $y_i \neq y_j$ for every distinct $i, j \in \{1, \ldots, k\}$, then the only thing the algorithm knows is that for every distinct $i, j \in \{1, \ldots, k\}$, it holds that $a \neq x_i \oplus x_j$.
  - If, instead, $y_i = y_j$, then a can be easily computed as $x_i \oplus x_j$.

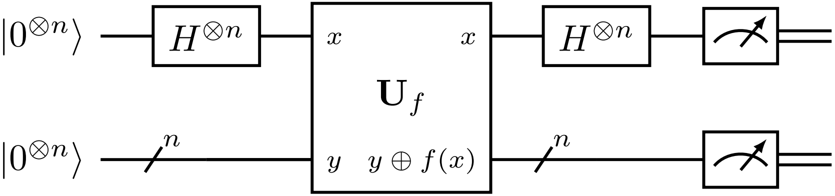
Classical Strategy

- Any classical algorithm querying a function $f \in \mathcal{F}_n^n$ which is periodic modulo a *accumulates knowledge* as follows:
  - If the queries done so far, namely

    $$y_1 = f(x_1), \ldots, y_k = f(x_k)$$

    are such that $y_i \neq y_j$ for every distinct $i, j \in \{1, \ldots, k\}$, then the only thing the algorithm knows is that for every distinct $i, j \in \{1, \ldots, k\}$, it holds that $a \neq x_i \oplus x_j$.
  - If, instead, $y_i = y_j$, then a can be easily computed as $x_i \oplus x_j$.
- Since there are only $\frac{k(k-1)}{2}$ pairs of distinct indices in $\{1, \ldots, k\}$, in the worst case the number of queries is has to be exponential.

The Simon Circuit $\mathbf{C}_{\mathsf{SIMON}}$



The Simon Algorithm

- The actual Simon's Algorithm has the following structure:
  1. $i \leftarrow 1$.
  2. Apply $\mathbf{C}_{\mathsf{SIMON}}$ and obtain in the first n qubits the value $|x_i\rangle$
  3. If $i = n + 3$, then go to step 4, otherwise, increment i by 1 and go back to 2.
  4. Solve the linear system of equations

$$\begin{cases} a \cdot x_1 = 0 \\ \vdots \\ a \cdot x_{n+3} = 0 \end{cases}$$

  5. If there is a unique nonnull solution a, then return it, otherwise fail.

The Simon Algorithm

- The actual Simon's Algorithm has the following structure:
  1. $i \leftarrow 1$.
  2. Apply $\mathbf{C}_{\mathsf{SIMON}}$ and obtain in the first n qubits the value $|x_i\rangle$
  3. If $i = n + 3$, then go to step 4, otherwise, increment i by 1 and go back to 2.
  4. Solve the linear system of equations

$$\begin{cases} a \cdot x_1 = 0 \\ \vdots \\ a \cdot x_{n+3} = 0 \end{cases}$$

  5. If there is a unique nonnull solution a, then return it, otherwise fail.
- The system of equation above has a unique solution with probability at least $\frac{2}{3}$.

Part V

Shor's Algorithm

A Groundbreaking Result

- Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties.

A Groundbreaking Result

- Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties.
- It is a witnesses of the fact that some useful problems can be solved by quantum circuits of polynomial size in the input n.

A Groundbreaking Result

- Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties.
- It is a witnesses of the fact that some useful problems can be solved by quantum circuits of polynomial size in the input n.
- This includes:
  - Breaking **RSA**, one of the most widely used public-key encryption scheme.
  - Integer **Factorization**;
  - Discrete **Logarithm**;

A Groundbreaking Result

- Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties.
- It is a witnesses of the fact that some useful problems can be solved by quantum circuits of polynomial size in the input n.
- This includes:
  - Breaking **RSA**, one of the most widely used public-key encryption scheme.
  - Integer **Factorization**;
  - Discrete **Logarithm**;
- A whole sub-field of cryptography, called **post-quantum cryptography**, grew out of all this.

A Groundbreaking Result

- Shor's Algorithm is considered as **one of the major results** in the theory of computation since the inception of the discipline, in the sixties.
- It is a witnesses of the fact that some useful problems can be solved by quantum circuits of polynomial size in the input n.
- This includes:
  - Breaking **RSA**, one of the most widely used public-key encryption scheme.
  - Integer **Factorization**;
  - Discrete **Logarithm**;
- A whole sub-field of cryptography, called **post-quantum cryptography**, grew out of all this.
- Ultimately, however, Shor's Algorithm is an algorithm about **period finding**.

Number-Theoretic Preliminaries — I

- let $\mathbb{G}_N$ be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.

Number-Theoretic Preliminaries — I

- let $\mathbb{G}_N$ be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.
- On $\mathbb{G}_N$, one can define a binary operation, namely multiplication modulo N:

$$(x, y) \mapsto x \cdot y \mod N$$

- Multiplication modulo N is well-defined, associative, and has an identity, namely 1, and has inverses. It is thus a (finite) **group**.

Number-Theoretic Preliminaries — I

- let $\mathbb{G}_N$ be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.
- On $\mathbb{G}_N$, one can define a binary operation, namely multiplication modulo N:

$$(x, y) \mapsto x \cdot y \mod N$$

- Multiplication modulo N is well-defined, associative, and has an identity, namely 1, and has inverses. It is thus a (finite) **group**.
- The cardinality $\Phi(N) \in \mathbb{N}$ of $\mathbb{G}_N$ is at most equal to $N - 1$.
  - If N is a prime, then $\Phi(N) = N - 1$.

Number-Theoretic Preliminaries — I

- let $\mathbb{G}_N$ be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.
- On $\mathbb{G}_N$, one can define a binary operation, namely multiplication modulo N:

$$(x, y) \mapsto x \cdot y \mod N$$

- Multiplication modulo N is well-defined, associative, and has an identity, namely 1, and has inverses. It is thus a (finite) **group**.
- The cardinality $\Phi(N) \in \mathbb{N}$ of $\mathbb{G}_N$ is at most equal to $N - 1$.
  - If N is a prime, then $\Phi(N) = N - 1$.
  - If N is the product of two prime numbers p and q, then $\Phi(N) = (p - 1)(q - 1)$.

Number-Theoretic Preliminaries — I

- let $\mathbb{G}_N$ be the set of all positive integers (including 1) which are strictly less than N and which has no prime factor in common with N.
- On $\mathbb{G}_N$, one can define a binary operation, namely multiplication modulo N:

$$(x, y) \mapsto x \cdot y \mod N$$

- Multiplication modulo N is well-defined, associative, and has an identity, namely 1, and has inverses. It is thus a (finite) **group**.
- The cardinality $\Phi(N) \in \mathbb{N}$ of $\mathbb{G}_N$ is at most equal to $N - 1$.
  - If N is a prime, then $\Phi(N) = N - 1$.
  - If N is the product of two prime numbers p and q, then $\Phi(N) = (p - 1)(q - 1)$.
  - Euler's theorem tells us that every $a \in \mathbb{G}_N$, it holds that

$$a^{\Phi(N)} \equiv 1 \mod N$$

Number-Theoretic Preliminaries — II

- The **order** of $n \in \mathbb{G}_N$ is the smallest number r such that

$$n^r \equiv 1 \mod N$$

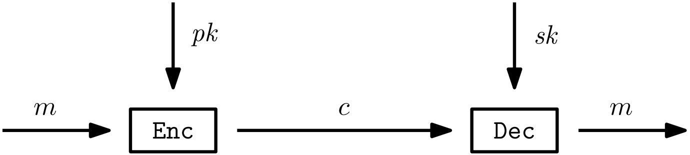
  - The order of n always divides $\Phi(N)$.
- A **subgroup** of $\mathbb{G}_N$ generated by $n \in \mathbb{G}_N$ is the subset $\langle n \rangle$ of $\mathbb{G}_N$ of those elements of $\mathbb{G}_N$ which can be written as $n^k \mod N$, where $k \in \mathbb{Z}$.
- If $\langle n \rangle = \langle m \rangle$ then n and m have the same order.

RSA

- RSA is one of the most common **public-key** encryption schemes.

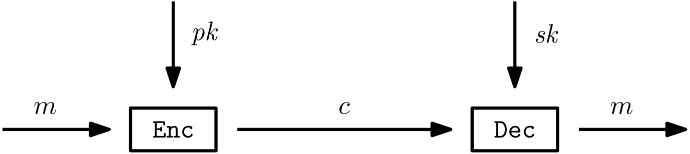
RSA

- RSA is one of the most common **public-key** encryption schemes.



RSA

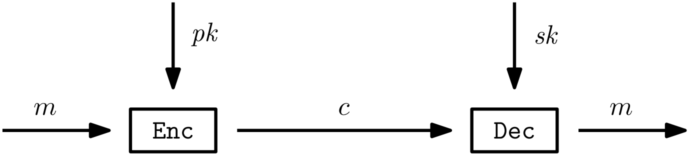
- RSA is one of the most common **public-key** encryption schemes.



- Given $N = pq$ product of two (large) primes and e, d such that $ed \equiv 1 \mod \Phi(N)$, the public key will be (N, e), while the secret key is (N, d).

RSA

- RSA is one of the most common **public-key** encryption schemes.



- Given $N = pq$ product of two (large) primes and e, d such that $ed \equiv 1 \mod \Phi(N)$, the public key will be (N, e), while the secret key is (N, d).
- Messages are simply elements of $\mathbb{G}_N$.
- Given a message $m \in \mathbb{G}_N$ and a public key (N, e), its encryption is $\mathtt{Enc}(m, (N, e)) = m^e \mod N$.
- Given a ciphertext $c \in \mathbb{G}_N$ and a private key (N, d), its decryption is $\mathtt{Dec}(c, (N, d)) = c^d \mod N$.

RSA

- RSA is one of the most common **public-key** encryption schemes.

$$\begin{tikzcd} & {} \arrow[d, "pk"] & & {} \arrow[d, "sk"] & \\ {} \arrow[r, "m"] & \boxed{\texttt{Enc}} \arrow[rr, "c"] & & \boxed{\texttt{Dec}} \arrow[r, "m"] & {} \end{tikzcd}$$

- Given $N = pq$ product of two (large) primes and e, d such that $ed \equiv 1 \mod \Phi(N)$, the public key will be (N, e), while the secret key is (N, d).
- Messages are simply elements of $\mathbb{G}_N$.
- Given a message $m \in \mathbb{G}_N$ and a public key (N, e), its encryption is $\texttt{Enc}(m, (N, e)) = m^e \mod N$.
- Given a ciphertext $c \in \mathbb{G}_N$ and a private key (N, d), its decryption is $\texttt{Dec}(c, (N, d)) = c^d \mod N$.
- Noticeably, $\texttt{Dec}(\texttt{Enc}(m, (N, e)), (N, d)) = m$.

Breaking RSA

- Clearly, **one way** of breaking RSA consists in, given $N = pq$ and $e \in \mathbb{G}_{(p-1)(q-1)}$, determine d such that
$$ed \equiv 1 \mod (p-1)(q-1)$$
  - This way one can determine the private key from the public key.
  - We do not even know $(p-1)(q-1)$, however.
- If we can find, given a and N, the smallest number $r \geq 1$ such that
$$c^r \equiv 1 \mod N$$
(where c is the ciphertext) then we can retrieve the message from the cryptogram in a natural way.
  - Such an r is nothing more and nothing less than the period of the function $x \mapsto c^x \mod N$.

Again About Period-Finding

- Shor's Algorithm is precisely about finding periods of functions in the form

$$x \mapsto a^x \mod N$$

Again About Period-Finding

- Shor's Algorithm is precisely about finding periods of functions in the form

$$x \mapsto a^x \mod N$$

- We can thus proceed as in Simon's Algorithm, thus applying a gate for the function above to the result of calling Hadamard functions, and subsequently measuring the output qubits, obtaining $f(x_0)$. This way we get the state

$$|\Psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle \, .$$

  where m is the smallest integer such that $mr + x_0 \geq 2^n$

Again About Period-Finding

- Shor's Algorithm is precisely about finding periods of functions in the form

$$x \mapsto a^x \mod N$$

- We can thus proceed as in Simon's Algorithm, thus applying a gate for the function above to the result of calling Hadamard functions, and subsequently measuring the output qubits, obtaining $f(x_0)$. This way we get the state

$$|\Psi\rangle_n = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |x_0 + kr\rangle \,.$$

  where m is the smallest integer such that $mr + x_0 \geq 2^n$
- There are two problems, however, namely:
  - The presence of x_0
  - The fact that f is not necessarily easy to be computed.

QFT

- The **Quantum Fourier Transform** is a way to get rid of the term x_0 in the sum above.

QFT

- The **Quantum Fourier Transform** is a way to get rid of the term x_0 in the sum above.
- It is defined as a unitary transform $\mathbf{U}_{FT}$ such that

$$\mathbf{U}_{FT}^{n} \left|x\right\rangle = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y 2^n} \left|y\right\rangle$$

QFT

- The **Quantum Fourier Transform** is a way to get rid of the term x_0 in the sum above.
- It is defined as a unitary transform $\mathbf{U}_{FT}$ such that

$$\mathbf{U}_{FT}^{n} |x\rangle = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y 2^n} |y\rangle$$

- Remarkably, $\mathbf{U}_{FT}^{n}$ can be implemented with a circuit of quadratic size consisting of unary gates, only.

QFT

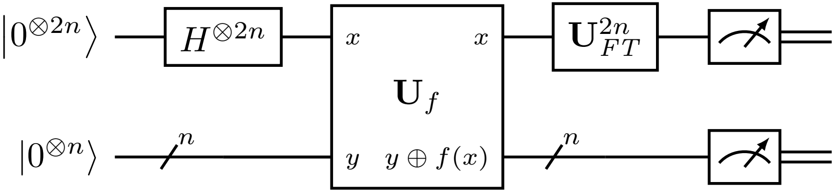
- The **Quantum Fourier Transform** is a way to get rid of the term x_0 in the sum above.
- It is defined as a unitary transform $\mathbf{U}_{FT}$ such that

$$\mathbf{U}_{FT}^{n} \left| x \right\rangle = \frac{1}{2^{n/2}} \sum_{y=0}^{2^n-1} e^{2\pi i x y 2^n} \left| y \right\rangle$$

- Remarkably, $\mathbf{U}_{FT}^{n}$ can be implemented with a circuit of quadratic size consisting of unary gates, only.
- If one applies $\mathbf{U}_{FT}^{n}$ to $\left|\Psi\right\rangle_n$, after that measuring *all bits*, then the probability of observing y when measuring the input qubits is

$$p(y) = \frac{1}{2^n m} \left| \sum_{k=0}^{m-1} e^{2\pi i k r y/2^n} \right|$$

Shor's Circuit $\mathbf{C}_{\mathsf{SHOR}}$



About Post-Processing and Replication

- By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.

About Post-Processing and Replication

- By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.
- The probability $p(y)$ has maxima when y is close to integral multiples of $2^{2n}/r$.

About Post-Processing and Replication

- By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.
- The probability $p(y)$ has maxima when y is close to integral multiples of $2^{2n}/r$.
- If y is within $\frac{1}{2}$ of $j2^{2n}/r$ for some integer j, we have that

$$\left|\frac{y}{2^{2n}} - \frac{j}{r}\right| \leq \frac{1}{2^{2n+1}}$$

About Post-Processing and Replication

- By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.
- The probability $p(y)$ has maxima when y is close to integral multiples of $2^{2n}/r$.
- If y is within $\frac{1}{2}$ of $j2^{2n}/r$ for some integer j, we have that

$$\left|\frac{y}{2^{2n}} - \frac{j}{r}\right| \leq \frac{1}{2^{2n+1}}$$

- We can thus learn terms in the form $\frac{j}{r}$ by repeating the process not so many times.

About Post-Processing and Replication

- By measuring y, however, we do not get exacctly what we need, namely the period r. We can get it, however, by way of some **post-processing**, perhaps having to **redo** the quantum computation from scratch.
- The probability $p(y)$ has maxima when y is close to integral multiples of $2^{2n}/r$.
- If y is within $\frac{1}{2}$ of $j2^{2n}/r$ for some integer j, we have that

$$\left|\frac{y}{2^{2n}} - \frac{j}{r}\right| \leq \frac{1}{2^{2n+1}}$$

- We can thus learn terms in the form $\frac{j}{r}$ by repeating the process not so many times.
- Out of them, we can finally find r.

Fast Exponentiation

- Computing the function $x \mapsto b^x \mod N$ is of course necessary for Shor's algorithm to work. In particular, we need a (quantum) circuit doing that and not so big.

Fast Exponentiation

- Computing the function $x \mapsto b^x \mod N$ is of course necessary for Shor's algorithm to work. In particular, we need a (quantum) circuit doing that and not so big.
- Just iterating the multiplication in the obvious way does *not* work, because the number of iterations will be in the worst case equal to N.

Fast Exponentiation

- Computing the function $x \mapsto b^x \mod N$ is of course necessary for Shor's algorithm to work. In particular, we need a (quantum) circuit doing that and not so big.
- Just iterating the multiplication in the obvious way does *not* work, because the number of iterations will be in the worst case equal to N.
- We need a more clever algorithm, called **fast exponentiation**, working on the input register x, the output register y and a work register w.
  - Initially, $y = 1$ and $w = b$.
  - We then update y and w in an iterative way, squaring w and multiplying y by w only if the corresponding bit of x is 1.

Factoring and Discrete Logarithm

- We saw Shor as an algorithm for computing the period of a function in the form

$$x \mapsto b^x \mod N,$$

  this way breaking RSA in polynomial time.

Factoring and Discrete Logarithm

- We saw Shor as an algorithm for computing the period of a function in the form

$$x \mapsto b^x \mod N,$$

this way breaking RSA in polynomial time.

- However, the same algorithm can be used to solve other problems of interest:
  - **Integer Factorization**.
    - After checking if the number N in input is prime, generate $1 < a < N$ at random, and if a is prime with N, compute the period r of $x \mapsto a^x \mod N$. If r is even, then any factor of N is either a factor of $a^{\frac{r}{2}} - 1$ or a factor of $a^{\frac{r}{2}} + 1$.
  - **Discrete Logarithm**.

Part VI

Grover Algorithm

Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$ such that there exists $a \in \{0,1\}^n$ with $f(x) = 1$ iff $x = a$.
  - *Output*: the (unique) string $a \in \{0,1\}^n$ such that $f(a) = 1$.

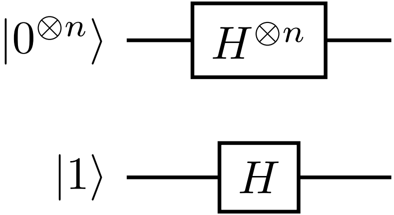
Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$ such that there exists $a \in \{0,1\}^n$ with $f(x) = 1$ iff $x = a$.
  - *Output*: the (unique) string $a \in \{0,1\}^n$ such that $f(a) = 1$.
- This is a *search problem*. In the worst case, any classic algorithm must evaluate f on all the 2^n coordinates.

Grover's Algorithm

- Grover's algorithm solves the following computational problem:
  - *Input*: a boolean circuit (whose structure is not accessible) computing an unknown function $f : \{0,1\}^n \to \{0,1\}$ such that there exists $a \in \{0,1\}^n$ with $f(x) = 1$ iff $x = a$.
  - *Output*: the (unique) string $a \in \{0,1\}^n$ such that $f(a) = 1$.
- This is a *search problem*. In the worst case, any classic algorithm must evaluate f on all the 2^n coordinates.
- Grover's Algorithm, by way of a technique called **amplitude amplification**, manages to solve the same problem in time at most $O(\sqrt{2^n})$.

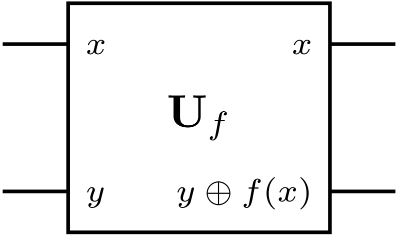
First Component



The resulting state is

$$|\phi\rangle \otimes H(|1\rangle) = \left(\frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle \right) \otimes H(|1\rangle)$$

Second Component

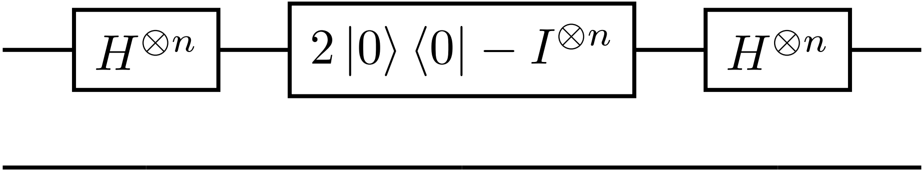


On an input $|x\rangle \otimes H(|1\rangle)$, the output is

$$(-1)^{f(x)}\,|x\rangle \otimes H(|1\rangle)$$

and so there must be a unitary transformation $\mathbf{V}$ on n qubits capturing its behaviour.

Third Component



The action of the component on the first n qubits can be seen as that of a unitary transformation, call it $\mathbf{W}$

A Geometric Analysis

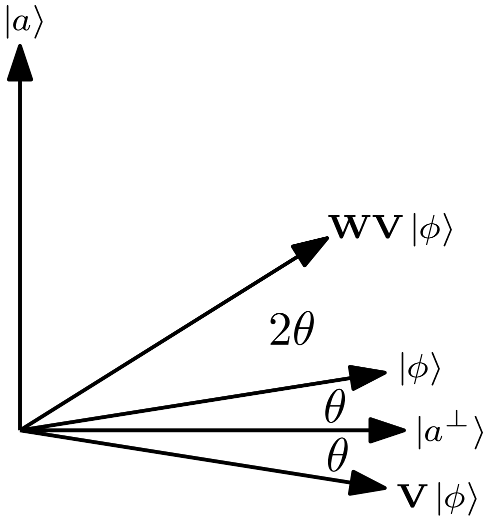
- Let us take a look at how $\mathbf{V}$ and $\mathbf{W}$ behave on the states $|a\rangle$ and $|\phi\rangle$:

$$\mathbf{V}\,|a\rangle = -\,|a\rangle\,; \qquad \mathbf{V}\,|\phi\rangle = |\phi\rangle - \frac{2}{2^{n/2}}\,|a\rangle\,;$$

$$\mathbf{W}\,|a\rangle = \frac{2}{2^{n/2}}\,|\phi\rangle - |a\rangle \qquad \mathbf{W}\,|\phi\rangle = |\phi\rangle$$

- Applying $\mathbf{V}$ and $\mathbf{W}$, then keep the following invariant true: the state is in the form $|\psi\rangle \otimes H(|1\rangle)$, where $|\psi\rangle$ lives in the two-dimensional space of all linear combinations $\alpha\,|\phi\rangle + \beta\,|a\rangle$, where $\alpha, \beta \in \mathbb{R}$ and $\alpha^2 + \beta^2 = 1$.
- The states $|a\rangle$ and $|\phi\rangle$ are almost orthogonal, since $\theta = \langle a|\phi\rangle = 2^{-n/2}$. There is in particular a vector forming a small angle with $|\phi\rangle$ and being orthogonal with $|a\rangle$, call it $\left|a^{\perp}\right\rangle$.
- Both $\mathbf{V}$ and $\mathbf{W}$ are reflections, and their product $\mathbf{WV}$ is thus a rotation.

A Geometric Analysis



Grover's Circuit $\mathbf{C}_{\mathsf{GROVER}}$