

Introduction to the Theory of Quantum Computation

The Basics

Ugo Dal Lago



ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Inria
informatiques mathématiques

PhD Program in Computer Science and Engineering, Bologna, July 2020

- ▶ **Quantum computing** refers to a new paradigm, alternative to the Church-Turing one, in which machines behave according to the law of quantum mechanics.

Contents

- ▶ **Quantum computing** refers to a new paradigm, alternative to the Church-Turing one, in which machines behave according to the law of quantum mechanics.
- ▶ It is 35 years old, and has been the object of many investigations, in particular due to the hope of going beyond the limits of the classic model.

Contents

- ▶ **Quantum computing** refers to a new paradigm, alternative to the Church-Turing one, in which machines behave according to the law of quantum mechanics.
- ▶ It is 35 years old, and has been the object of many investigations, in particular due to the hope of going beyond the limits of the classic model.
- ▶ This course aims at being an introduction to the **theory** of quantum computing, and does not aim at describing the many areas of application of this new paradigm.

Contents

- ▶ **Quantum computing** refers to a new paradigm, alternative to the Church-Turing one, in which machines behave according to the law of quantum mechanics.
- ▶ It is 35 years old, and has been the object of many investigations, in particular due to the hope of going beyond the limits of the classic model.
- ▶ This course aims at being an introduction to the **theory** of quantum computing, and does not aim at describing the many areas of application of this new paradigm.
- ▶ The course is divided into three parts:
 1. **Basics.**
 2. **Algorithmics.**
 3. **Programming Languages.**

Schedule and Study Material

- ▶ **Lectures** will take place on Microsoft Teams from 2pm to 4pm *every day* from Monday July 20th to Friday July 24th, 2020.

Schedule and Study Material

- ▶ **Lectures** will take place on Microsoft Teams from 2pm to 4pm *every day* from Monday July 20th to Friday July 24th, 2020.
- ▶ **Slides** will be made available in the course's webpage.

Schedule and Study Material

- ▶ **Lectures** will take place on Microsoft Teams from 2pm to 4pm *every day* from Monday July 20th to Friday July 24th, 2020.
- ▶ **Slides** will be made available in the course's webpage.
- ▶ Many **textbooks** on quantum computation are available. Here, I just mention a few of them:
 - ▶ Michael A. Nielsen, Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge University Press, 2016.
 - ▶ Phillip Kaye, Raymond Laflamme, Michele Mosca. *An Introduction to Quantum Computing*. Oxford University Press, 2006.
 - ▶ Noson S. Yanofsky, Mirco Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008.

Exam

- ▶ The exam consists in reading **a paper** about any aspect of quantum computation which might be interested for your studies.

Exam

- ▶ The exam consists in reading **a paper** about any aspect of quantum computation which might be interesting for your studies.
- ▶ The paper can be *proposed* by the student, or *suggested* by the teacher.
 - ▶ Quantum computing is having a great impact in so many areas of computer science.

Exam

- ▶ The exam consists in reading a **paper** about any aspect of quantum computation which might be interesting for your studies.
- ▶ The paper can be *proposed* by the student, or *suggested* by the teacher.
 - ▶ Quantum computing is having a great impact in so many areas of computer science.
- ▶ After having read the paper, the student is supposed to write a **report** of between 5 and 10 pages in which she describes the results in the paper to non-specialists.

Section 1

Classical and Probabilistic Systems

Notation

- ▶ Sets: A, B, C, \dots ;
- ▶ Cartesian Products: $A \times B$;
- ▶ Natural Numbers: $1, 2, 3, \dots$
- ▶ Real Numbers: \mathbb{R} ;

$$1, \frac{3}{7}, 17463, \pi, \dots$$

- ▶ Complex Numbers: \mathbb{C} ;

$$1 + 5i, \frac{1}{\sqrt{2}}i, \dots$$

- ▶ Vectors: A^n ;
- ▶ Matrices: $B^{m \times n}$.

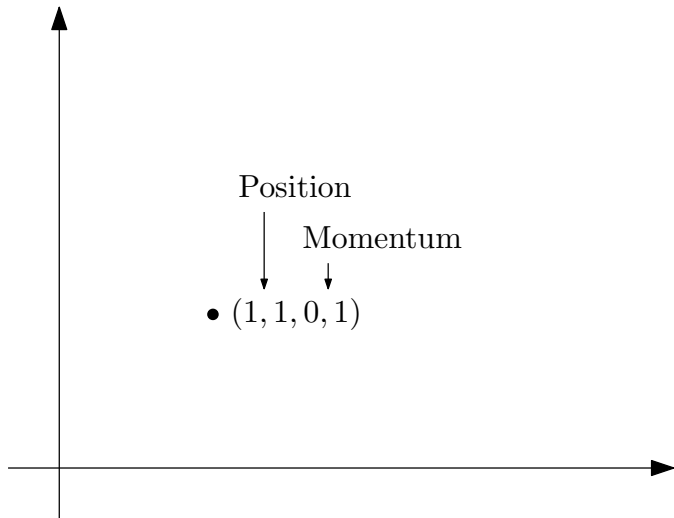
State of a Physical System

- ▶ **Instantaneous** description of the physical entities forming the system;
- ▶ Can change over time;
- ▶ Is very often an element of \mathbb{R}^d ;
- ▶ In classical physics, the state of a system is **uniquely determined**.

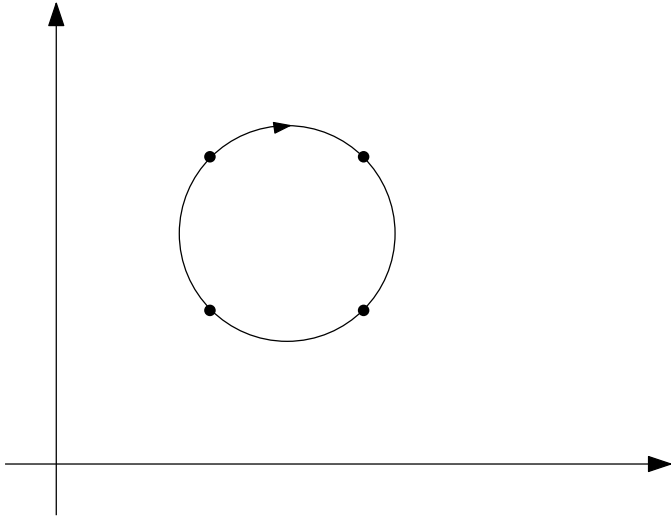
Example: Particle in \mathbb{R}^2 at a Given Time



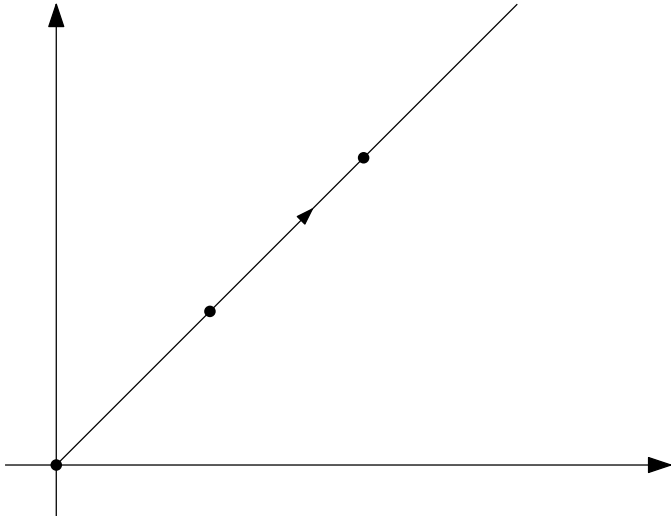
Example: Particle in \mathbb{R}^2 at a Given Time



Example: Particle in \mathbb{R}^2 Dynamically



Example: Particle in \mathbb{R}^2 Dynamically



Composing Physical Systems

- ▶ If the states of two physical systems live in \mathbb{R}^n e \mathbb{R}^m (respectively), how should we represent the physical system obtained by **composing** the two systems?
- ▶ We should keep track of the first **and** the second state.
- ▶ Solution: $\mathbb{R}^n \times \mathbb{R}^m = \mathbb{R}^{n+m}$.

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

0,

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

$0, \alpha,$

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

$$0, \alpha, 2\alpha,$$

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

$$0, \alpha, 2\alpha, 3\alpha,$$

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

$$0, \alpha, 2\alpha, 3\alpha, 4\alpha,$$

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

$$0, \alpha, 2\alpha, 3\alpha, 4\alpha, \dots$$

How to Discretize Time (and the State Space)

- ▶ Traditionally, time is just a number from \mathbb{R} .
- ▶ But sometimes, it suffices to consider the state of the system only at **certain** instants.

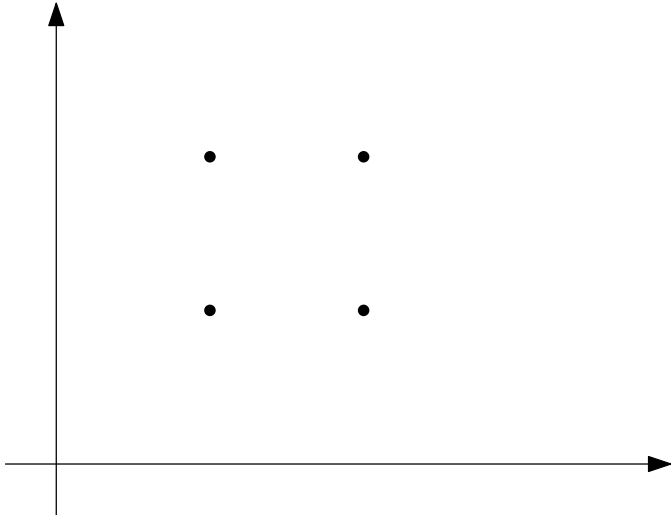
$$0, \alpha, 2\alpha, 3\alpha, 4\alpha, \dots$$

- ▶ Time becomes an element \mathbb{N} , which is **discrete**.
- ▶ Why?
 - ▶ Abstraction;
 - ▶ Simulation.
- ▶ Sometimes also states become **discrete**...
 - ▶ E.g., in Turing Machines.
- ▶ Often one can also assume that there are **finitely many** states.
 - ▶ The system is in one of the n states

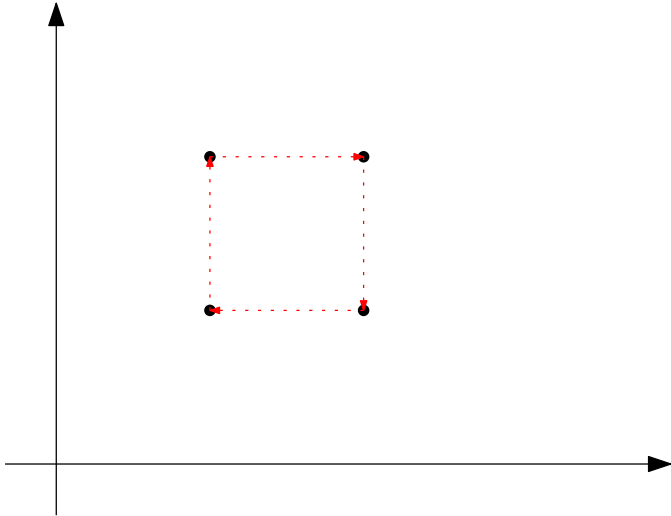
$$\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^n.$$

- ▶ For the sake of simplicity, let us focus our attention on **finite state** systems.

Example: Particle in \mathbb{R}^2 , Dynamically



Example: Particle in \mathbb{R}^2 , Dynamically



Probabilistic Systems, Statically

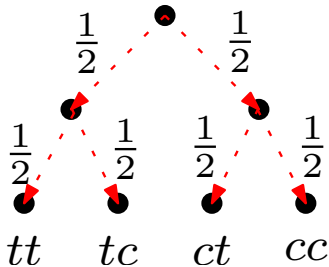
- ▶ In classical physics, the state of any system is **univocally determined**.
- ▶ For reasons of abstraction and complexity, one often uses a **probabilistic** representation of states.
 - ▶ Example: a **dice**.
- ▶ The state, at a given time, is described by a function $\mathbf{P} : \mathcal{S} \rightarrow \mathbb{R}$ such that $\sum_{\mathbf{x} \in \mathcal{S}} \mathbf{P}(\mathbf{x}) = 1$, or by a vector in $\mathbb{R}^{|\mathcal{S}|}$:

$$\begin{pmatrix} \mathbf{P}(\mathbf{x}_1) \\ \vdots \\ \mathbf{P}(\mathbf{x}_n) \end{pmatrix} = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$$

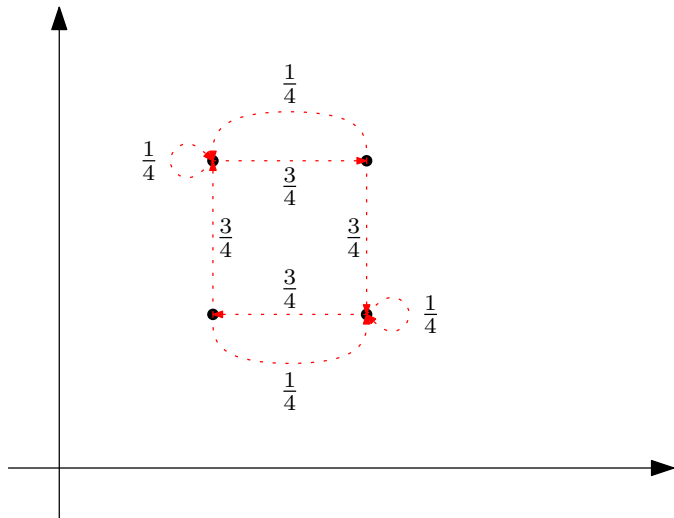
- ▶ This is a **mixed state**.

Probabilistic Systems, Dynamically

- ▶ In any given state \mathbf{x}_i , the system can evolve into **any** state \mathbf{x}_j with probability p_{ij} .
- ▶ As an example:



Example: Particle in \mathbb{R}^2 , Dynamically



Dynamics: Matrix Representation

- ▶ Suppose we work in a system with n states:

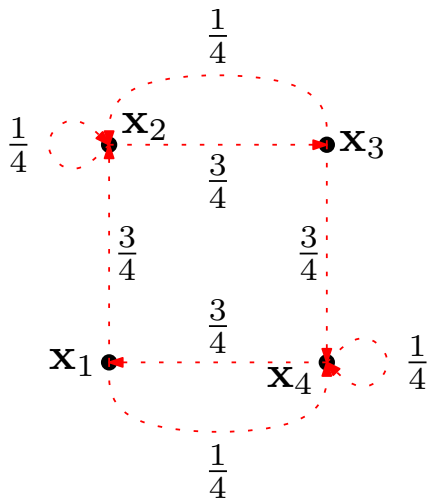
$$\mathcal{S} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^m.$$

- ▶ The dynamic evolution of the system is described, compactly, by the matrix \mathbf{M} in $\mathbb{R}^{n \times n}$:

$$\mathbf{M} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix}$$

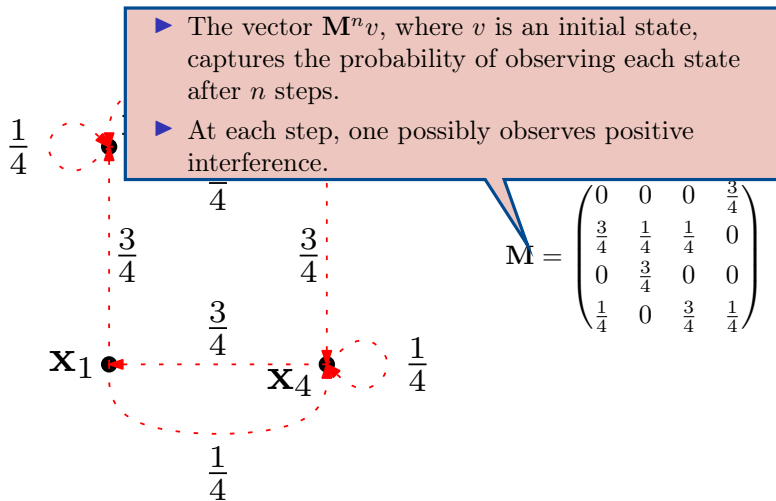
- ▶ A transition step corresponds to **multiplying** \mathbf{M} and \mathbf{P} (the latter seen as a vector).
- ▶ In other words, the system can be seen as a so-called **Markov Chain**.

Example: Particle in \mathbb{R}^2 , Dynamically



$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 0 & \frac{3}{4} \\ \frac{3}{4} & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{3}{4} & 0 & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} & \frac{1}{4} \end{pmatrix}$$

Example: Particle in \mathbb{R}^2 , Dynamically



Linearity

- ▶ Vectors in \mathbb{R}^n support:

- ▶ **Binary Sums:**

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

- ▶ **Products by Scalars in \mathbb{R} :**

$$\alpha \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \alpha x_1 \\ \vdots \\ \alpha x_n \end{pmatrix}$$

- ▶ Multiplication with a matrix gives rise to a **linear map**:

$$\mathbf{M}(\alpha \mathbf{x}) = \alpha(\mathbf{M}\mathbf{x});$$

$$\mathbf{M}(\mathbf{x} + \mathbf{y}) = \mathbf{M}\mathbf{x} + \mathbf{M}\mathbf{y}.$$

Classical Systems as Specific Probabilistic Systems

- ▶ A **pure state** is a function $\mathbf{P} : \mathcal{S} \rightarrow \mathbb{R}$ such that $\mathbf{P}(\mathbf{x}_i) = 1$ for a given \mathbf{x}_i .
 - ▶ The system is in \mathbf{x}_i , without any uncertainty.

Classical Systems as Specific Probabilistic Systems

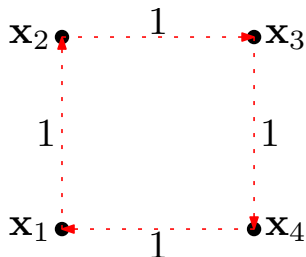
- ▶ A **pure state** is a function $\mathbf{P} : \mathcal{S} \rightarrow \mathbb{R}$ such that $\mathbf{P}(\mathbf{x}_i) = 1$ for a given \mathbf{x}_i .
 - ▶ The system is in \mathbf{x}_i , without any uncertainty.
- ▶ If the matrix $\mathbf{M} = (p_{ij})$ is such that for every $i \in \{1, \dots, n\}$ there is $j \in \{1, \dots, n\}$ such that $p_{ij} = 1$, then \mathbf{M} is **classical**.

Classical Systems as Specific Probabilistic Systems

- ▶ A **pure state** is a function $\mathbf{P} : \mathcal{S} \rightarrow \mathbb{R}$ such that $\mathbf{P}(\mathbf{x}_i) = 1$ for a given \mathbf{x}_i .
 - ▶ The system is in \mathbf{x}_i , without any uncertainty.
- ▶ If the matrix $\mathbf{M} = (p_{ij})$ is such that for every $i \in \{1, \dots, n\}$ there is $j \in \{1, \dots, n\}$ such that $p_{ij} = 1$, then \mathbf{M} is **classical**.
- ▶ A classical matrix turns pure states into pure states!

Classical Systems as Specific Probabilistic Systems

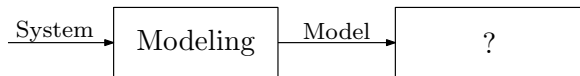
- ▶ A **pure state** is a function $\mathbf{P} : \mathcal{S} \rightarrow \mathbb{R}$ such that $\mathbf{P}(\mathbf{x}_i) = 1$ for a given \mathbf{x}_i .
 - ▶ The system is in \mathbf{x}_i , without any uncertainty.
- ▶ If the matrix $\mathbf{M} = (p_{ij})$ is such that for every $i \in \{1, \dots, n\}$ there is $j \in \{1, \dots, n\}$ such that $p_{ij} = 1$, then \mathbf{M} is **classical**.
- ▶ A classical matrix turns pure states into pure states!
- ▶ As an example:



$$\mathbf{M} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

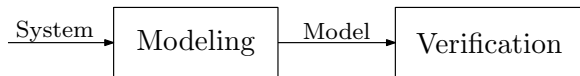
Incepting Probability into Algorithms

- ▶ In what we have seen so far, probability has been seen a way to facilitate **modeling** of existing systems, e.g.



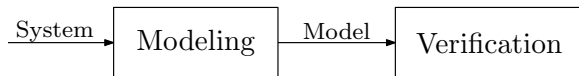
Incepting Probability into Algorithms

- ▶ In what we have seen so far, probability has been seen a way to facilitate **modeling** of existing systems, e.g.



Incepting Probability into Algorithms

- ▶ In what we have seen so far, probability has been seen a way to facilitate **modeling** of existing systems, e.g.



- ▶ Probability, however, can also be seen as a very powerful way to **compute**.
 - ▶ Algorithms can “flip a coin”;
 - ▶ This is somehow a paradigm shift: probabilism is not there for modeling uncertainty, but rather for the purpose of **relaxing determinism** as an algorithm design principle.
 - ▶ This “new” computation paradigm is called **probabilistic computation**.
 - ▶ Your algorithms (then your programs) can be *seen* and *treated* as probabilistic systems.

But Why?

▶ Achieving Better Performances

- ▶ Allowing computation to proceed probabilistically *somehow* allows to exploit different computation paths **at the same time**.
- ▶ Of course, each computation path comes with its own probability.
- ▶ ...but all this can (sometime) be turned into a way to alleviate exponential blowups.

But Why?

▶ Achieving Better Performances

- ▶ Allowing computation to proceed probabilistically *somehow* allows to exploit different computation paths **at the same time**.
- ▶ Of course, each computation path comes with its own probability.
- ▶ ...but all this can (sometime) be turned into a way to alleviate exponential blowups.

▶ Allowing a Form of “Confusion” in Computation

- ▶ Deterministic computation has its own shortcomings.
- ▶ In particular, in some interactive scenarios probabilistic computation is a way to achieve security.

Example

MILLER-RABIN(n)

If $n > 2$ and n is even, return **composite**.

/* Factor $n - 1$ as $2^s t$ where t is odd. */

$s \leftarrow 0$

$t \leftarrow n - 1$

while t is even

$s \leftarrow s + 1$

$t \leftarrow t/2$

end /* Done. $n - 1 = 2^s t$. */

Choose $x \in \{1, 2, \dots, n - 1\}$ uniformly at random.

Compute each of the numbers $x^t, x^{2t}, x^{4t}, \dots, x^{2^s t} = x^{n-1} \pmod n$.

If $x^{n-1} \not\equiv 1 \pmod n$, return **composite**.

for $i = 1, 2, \dots, s$

 If $x^{2^i t} \equiv 1 \pmod n$ and $x^{2^{i-1} t} \not\equiv \pm 1 \pmod n$, return **composite**.

end /* Done checking for fake square roots. */

Return **probably prime**.

Example

MILLER-RABIN(n)

If $n > 2$ and n is even, return **composite**.

/* Factor $n - 1$ as $2^s t$ where t is odd. */

$s \leftarrow 0$

$t \leftarrow n - 1$

while t is even

$s \leftarrow s + 1$

$t \leftarrow t/2$

end /* Done. $n - 1 = 2^s t$. */

Choose $x \in \{1, 2, \dots, n - 1\}$ uniformly at random.

Compute each of the numbers $x^t, x^{2t}, x^{4t}, \dots, x^{2^s t} = x^{n-1} \pmod n$.

If $x^{n-1} \not\equiv 1 \pmod n$, return **composite**.

for $i = 1, 2, \dots, s$

 If $x^{2^i t} \equiv 1 \pmod n$ and $x^{2^{i-1} t} \not\equiv \pm 1 \pmod n$, return **composite**.

end /* Done checking for fake square roots. */

Return **probably prime**.

Another Example

- ▶ Suppose given a public-key encryption scheme $\Pi = (GEN, ENC, DEC)$ and an adversary \mathcal{A} .

Another Example

- ▶ Suppose given a public-key encryption scheme $\Pi = (GEN, ENC, DEC)$ and an adversary \mathcal{A} .
- ▶ Consider the following **experiment**:
 1. $GEN(1^n)$ is run to obtain keys (pk, sk) .
 2. Adversary \mathcal{A} is given pk , and outputs a pair of messages m_0, m_1 **of the same length**.
 3. A bit b is chosen and a ciphertext c is computed as $ENC(pk, m_b)$ and given back to \mathcal{A} .
 4. \mathcal{A} outputs a bit b' .
 5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise.

Another Example

- ▶ Suppose given a public-key encryption scheme $\Pi = (GEN, ENC, DEC)$ and an adversary \mathcal{A} .
- ▶ Consider the following **experiment**:
 1. $GEN(1^n)$ is run to obtain keys (pk, sk) .
 2. Adversary \mathcal{A} is given pk , and outputs a pair of messages m_0, m_1 **of the same length**.
 3. A bit b is chosen and a ciphertext c is computed as $ENC(pk, m_b)$ and given back to \mathcal{A} .
 4. \mathcal{A} outputs a bit b' .
 5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise.
- ▶ If the output is 1, the adversary wins.

Another Example

- ▶ Suppose given a public-key encryption scheme $\Pi = (GEN, ENC, DEC)$ and an adversary \mathcal{A} .
- ▶ Consider the following **experiment**:
 1. $GEN(1^n)$ is run to obtain keys (pk, sk) .
 2. Adversary \mathcal{A} is given pk , and outputs a pair of messages m_0, m_1 **of the same length**.
 3. A bit b is chosen and a ciphertext c is computed as $ENC(pk, m_b)$ and given back to \mathcal{A} .
 4. \mathcal{A} outputs a bit b' .
 5. The output of the experiment is defined to be 1 if $b' = b$ and 0 otherwise.
- ▶ If the output is 1, the adversary wins.
- ▶ Is it possible to design Π in such a way that **any** adversary can win with probability not (too much) bigger than $\frac{1}{2}$?

Another Example

- ▶ Suppose given a public-key encryption scheme $\Pi = (GEN, ENC, DEC)$ and an adversary \mathcal{A} .
 - ▶ Consider the following **experiment**:
 1. $GEN(1^n)$ is run to obtain keys (pk, sk) .
 2. Adversary \mathcal{A} is given pk , and outputs a pair of messages m_0, m_1 **of the same length**.
 3. A bit b is chosen and a ciphertext c is computed as $ENC(pk, m_b)$ and given back to \mathcal{A} .
 4. \mathcal{A} outputs a bit b' .
- Yes, but **only if** ENC is randomized! if $b' = b$ and 0 otherwise.
- ▶ If the output is 1, the adversary wins.
 - ▶ Is it possible to design Π in such a way that **any** adversary can win with probability not (too much) bigger than $\frac{1}{2}$?

Section 2

Quantum Systems

Quanta and Computation

- ▶ Starting from the beginning of the XX century, physical sciences faced a series of **revolutions**.

Quanta and Computation

- ▶ Starting from the beginning of the XX century, physical sciences faced a series of **revolutions**.
- ▶ The principles of classical physics were proved **not** to hold when the involved magnitudes become as little as those in atoms, or as big as those found in astrophysics.

Quanta and Computation

- ▶ Starting from the beginning of the XX century, physical sciences faced a series of **revolutions**.
- ▶ The principles of classical physics were proved **not** to hold when the involved magnitudes become as little as those in atoms, or as big as those found in astrophysics.
- ▶ One of the answers to the subsequent crisis was the introduction of **quantum mechanics**.
 - ▶ It is a **complex, non-intuitive, theory**;
 - ▶ Mathematically, it is highly **not trivial**;

Quanta and Computation

- ▶ Starting from the beginning of the XX century, physical sciences faced a series of **revolutions**.
- ▶ The principles of classical physics were proved **not** to hold when the involved magnitudes become as little as those in atoms, or as big as those found in astrophysics.
- ▶ One of the answers to the subsequent crisis was the introduction of **quantum mechanics**.
 - ▶ It is a **complex, non-intuitive, theory**;
 - ▶ Mathematically, it is highly **not trivial**;
- ▶ In 1982, Richard Feynman suggests, for the first time, that quantum mechanics can form the basis of a **new model of computation**, more efficient than the one introduced by Turing and later generalized to a proper model of computation.

Quantum Mechanical Computers

By Richard P. Feynman

Introduction

This work is a part of an effort to analyze the physical limitations of computers due to the laws of physics. For example, Bennett¹ has made a careful study of the free energy dissipation that must accompany computation. He found it to be virtually zero. He suggested to me the question of the limitations due to quantum mechanics and the uncertainty principle. I have found that, aside from the obvious limitation to size if the working parts are to be made of atoms, there is no fundamental limit from these sources either.

We are here considering ideal machines; the effects of small imperfections will be considered later. This study is one of principle; our aim is to exhibit some Hamiltonian for a system which could serve as a computer. We are not concerned with whether we have the most efficient system, nor how we could best implement it.

such. We see we really have two more logical primitives, FAN OUT when two wires are connected to one, and EXCHANGE, when wires are crossed. In the usual computer the NOT and NAND primitives are implemented by transistors, possibly as in Fig. 2.

What is the minimum free energy that must be expended to operate an ideal computer made of such primitives? Since, for example, when the AND operates the output line, c' is being determined to be one of two values no matter what it was before the entropy change is $\ln(2)$ units. This represents a heat generation of $kT \ln(2)$ at temperature T . For many years it was thought that this represented an absolute minimum to the quantity of heat per primitive step that had to be dissipated in making a calculation.

The question is academic at this time. In actual machines we are quite concerned with the heat dissipation question, but the transistor system used actually dissipates about $10^{10}kT$! As

could be stored in an inductance, or other reactive element.

However, it is apparently very difficult to make inductive elements on silicon wafers with present techniques. Even Nature, in her DNA copying machine, dissipates about $100 kT$ per bit copied. Being, at present, so very far from this $kT \ln(2)$ figure, it seems ridiculous to argue that even this is too high and the minimum is really essentially zero. But, we are going to be even more ridiculous later and consider bits written on one atom instead of the present 10^{11} atoms. Such nonsense is very entertaining to professors like me. I hope you will find it interesting and entertaining also.

What Bennett pointed out was that this former limit was wrong because it is not necessary to use irreversible primitives. Calculations can be done with reversible machines containing only reversible primitives. If this is done the minimum free energy required is independent of the complexity or num-

RICHARD FEYNMAN

State of a Quantum System H_n

- ▶ It is a vector in \mathbb{C}^n :

$$\mathbf{x} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

- ▶ The following condition must hold:

$$\sum_{i=1}^n \|c_i\|^2 = 1$$

- ▶ We can write

$$\mathbf{x} = c_1 \mathbf{x}_1 + \dots + c_n \mathbf{x}_n,$$

where the \mathbf{x}_i are always 0 except in the i th coordinate, which is 1 ($\mathbf{x}_1, \dots, \mathbf{x}_n$ is the **computational base** of the system).

- ▶ What does \mathbf{x}_i represent? It is an **observable** of the system.
 - ▶ Every **orthonormal base** for \mathbb{C}^n corresponds to one such observable property.

Evolution of a Quantum System H_n

- ▶ The system evolves **linearly**, thus its dynamics can be described by a matrix:

$$\begin{pmatrix} d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

- ▶ It must be that $\sum_{i=1}^n \|d_i\|^2 = 1$.
- ▶ A necessary and sufficient condition is that $\mathbf{Q} = (a_{ij})$ must be **unitary**: $\mathbf{Q}^* = \mathbf{Q}^{-1}$.
 - ▶ \mathbf{Q}^* is the conjugate transpose of \mathbf{Q} . As an example,

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}^* = \begin{pmatrix} a_{11}^* & \cdots & a_{n1}^* \\ \vdots & & \vdots \\ a_{1n}^* & \cdots & a_{nn}^* \end{pmatrix}$$

- ▶ The product of two unitary matrices is again unitary.

Measurement in a Quantum System H_n

- ▶ Given a Quantum System $\mathbf{x} = (c_i)$, the quantity $\|c_i\|^2$ **should not** be interpreted as the probability of being in state \mathbf{x}_i .
- ▶ The only way to observe the state of a quantum system is by way of **measurements**:
 - ▶ When measuring the state of a system, it goes from $\mathbf{x} = (c_i)$ to each of the states \mathbf{x}_j with probability equals to $\|c_j\|^2$.
 - ▶ At that point the system is in a **classical state**, i.e., it can be observed without being altered.
 - ▶ Everything depends on the chosen base, which corresponds to the desired observable.
- ▶ Again: measuring a quantum system **can alter** its state!

Measurement in a Quantum System H_n

$$\mathbf{x} = \sum_{i=1}^n c_i \mathbf{x}_i$$

The diagram illustrates the decomposition of a quantum state \mathbf{x} into a superposition of basis states $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$. The state \mathbf{x} is represented as a sum of terms $c_i \mathbf{x}_i$. Arrows point from the coefficients c_i to the corresponding basis states \mathbf{x}_i , with the squared magnitude of the coefficient, $\|c_i\|^2$, indicating the probability of measuring the system in state \mathbf{x}_i .

Composing two Quantum Systems H_n and H_m

- ▶ Given two quantum systems with computational bases

$$\mathbf{x}_1, \dots, \mathbf{x}_n;$$

$$\mathbf{y}_1, \dots, \mathbf{y}_m;$$

their **composition** has computational basis

$$(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_m).$$

- ▶ Thus, we get the system

$$H_{nm} = H_n \otimes H_m.$$

- ▶ Let us indicate the pair $(\mathbf{x}_i, \mathbf{x}_j)$ by $\mathbf{x}_i \otimes \mathbf{x}_j$;

Composing two Quantum Systems H_n and H_m

- ▶ We can extend this operation to all states of H_n and H_m :

$$\left(\sum_{i=1}^n c_i \mathbf{x}_i \right) \otimes \left(\sum_{j=1}^m d_j \mathbf{y}_j \right) = \sum_{i=1}^n \sum_{j=1}^m c_i d_j \mathbf{x}_i \otimes \mathbf{y}_j.$$

- ▶ A quantum state \mathbf{x} is sometimes indicated as $|\mathbf{x}\rangle$.
- ▶ Another notational abuse:

$$|\mathbf{x}\rangle \otimes |\mathbf{y}\rangle = |\mathbf{x}\rangle|\mathbf{y}\rangle = |\mathbf{xy}\rangle.$$

- ▶ If a state \mathbf{x} of $H_n \otimes H_m$ is such that $\mathbf{x} = \mathbf{y} \otimes \mathbf{z}$, then \mathbf{x} is said to be **decomposable**, and otherwise **entangled**.
- ▶ Given two unitary transforms $\mathbf{Q}_n, \mathbf{Q}_m$ on H_n and H_m (respectively), we can build $\mathbf{Q}_n \otimes \mathbf{Q}_m \dots$

Example: H_2

- ▶ The states of the system are the unitary vectors in \mathbb{C}^2 .
 - ▶ The states of the orthonormal base $|\mathbf{x}_1\rangle, |\mathbf{x}_2\rangle$ are sometime denoted with $|\mathbf{0}\rangle$ e $|\mathbf{1}\rangle$.
- ▶ Unitary transforms: examples:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

$$\mathbf{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \mathbf{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

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

- ▶ States of H_2 are said to be **qubits**.

Example: $H_2 \otimes H_2$

- ▶ States of the obtained system are the unitary vectors in $\mathbb{C}^4 \cong \mathbb{C}^2 \otimes \mathbb{C}^2$.
- ▶ Let us consider two example states in $H_2 \otimes H_2$.

- ▶ The state $|01\rangle$ is decomposable: $|01\rangle = |0\rangle \otimes |1\rangle$.
- ▶ The state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

is entangled

- ▶ A unitary transform on $H_2 \otimes H_2$ which cannot be decomposed as two unitary transforms on H_2 is:

$$\mathbf{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

An Example

$$(\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes |0\rangle) \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes |0\rangle) \\ &= \mathbf{CNOT} \left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \otimes |0\rangle \right) \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes |0\rangle) \\ &= \mathbf{CNOT} \left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \otimes |0\rangle \right) \\ &= \mathbf{CNOT} \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle \right) \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes |0\rangle) \\ &= \mathbf{CNOT} \left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \otimes |0\rangle \right) \\ &= \mathbf{CNOT} \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle \right) \\ &= \frac{1}{\sqrt{2}}\mathbf{CNOT}|00\rangle + \frac{1}{\sqrt{2}}\mathbf{CNOT}|10\rangle \end{aligned}$$

An Example

$$\begin{aligned} & (\mathbf{CNOT} \circ (\mathbf{H} \otimes \mathbf{I}))|00\rangle \\ &= \mathbf{CNOT}((\mathbf{H} \otimes \mathbf{I})(|00\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes \mathbf{I}(|0\rangle)) \\ &= \mathbf{CNOT}(\mathbf{H}(|0\rangle) \otimes |0\rangle) \\ &= \mathbf{CNOT} \left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \otimes |0\rangle \right) \\ &= \mathbf{CNOT} \left(\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|10\rangle \right) \\ &= \frac{1}{\sqrt{2}}\mathbf{CNOT}|00\rangle + \frac{1}{\sqrt{2}}\mathbf{CNOT}|10\rangle \\ &= \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \end{aligned}$$

About Measurements, Again

- ▶ Suppose we want to work with the system $H_n \otimes H_m$. A state of the system can be expressed as:

$$\mathcal{Q} = \sum_{i=1}^n \sum_{j=1}^m c_{ij} |\mathbf{x}_i\rangle |\mathbf{x}_j\rangle.$$

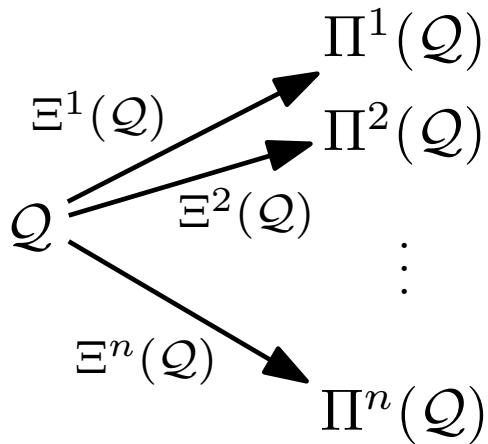
- ▶ If we observe the **first** component of the system (the “rightmost” component of the tensor product), we get that:
 - ▶ We will end up in a state $|\mathbf{x}_k\rangle \otimes |\mathbf{y}\rangle$ where \mathbf{x}_k is an element of the computational base, while \mathbf{y} is **not necessarily one such**.
 - ▶ The probability of reaching $|\mathbf{x}_k\rangle \otimes |\mathbf{y}\rangle$ is equal to

$$\Xi^k(\mathcal{Q}) = \sum_{j=1}^m \|c_{kj}\|^2.$$

- ▶ The state $|\mathbf{x}_k\rangle \otimes |\mathbf{y}\rangle$ can be expressed as follows:

$$\Pi^k(\mathcal{Q}) = \frac{1}{\sqrt{\mathbf{P}(k)}} \sum_{j=1}^m c_{kj} |\mathbf{x}_k\rangle |\mathbf{y}_j\rangle = \frac{|\mathbf{x}_k\rangle}{\sqrt{\mathbf{P}(k)}} \sum_{j=1}^m c_{kj} |\mathbf{y}_j\rangle$$

About Measurements, Again

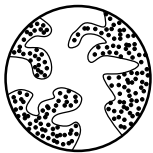


About Measurements, Again

$$\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$$

The diagram illustrates the decomposition of the quantum state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$ into two basis states. Two arrows originate from the state expression. The upper arrow points to the state $|00\rangle$ and is labeled with the coefficient $\frac{1}{2}$. The lower arrow points to the state $|11\rangle$ and is also labeled with the coefficient $\frac{1}{2}$.

A Game



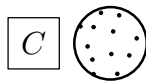
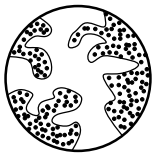
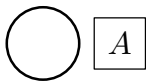
A

B

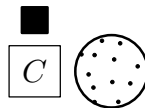
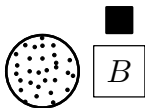
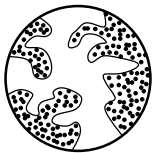
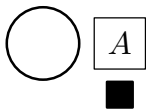
C



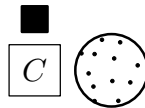
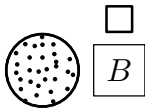
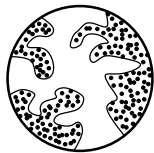
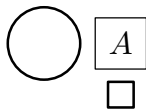
A Game



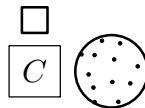
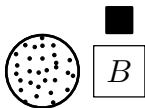
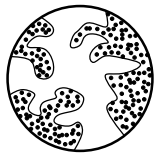
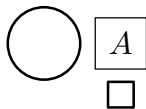
A Game



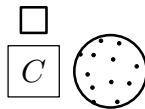
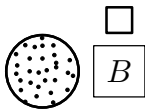
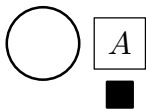
A Game



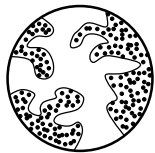
A Game



A Game



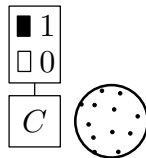
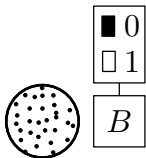
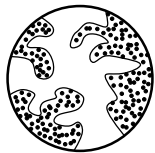
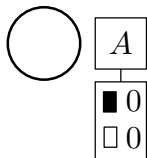
A Game



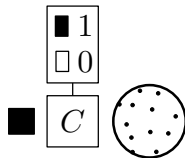
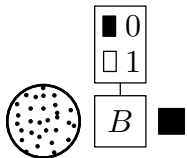
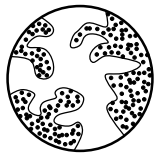
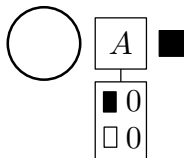
A	B	C
<input checked="" type="checkbox"/> 0	<input checked="" type="checkbox"/> 0	<input checked="" type="checkbox"/> 1
<input type="checkbox"/> 0	<input type="checkbox"/> 1	<input type="checkbox"/> 0



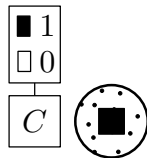
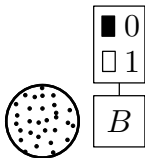
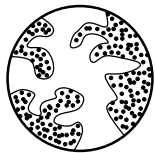
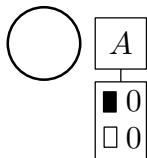
A Game



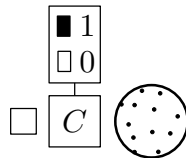
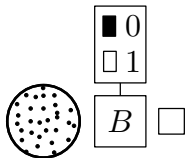
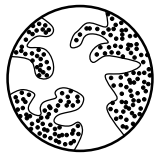
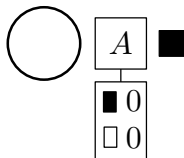
A Game



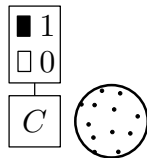
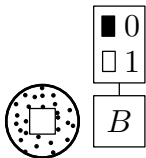
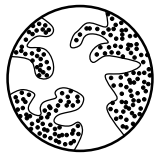
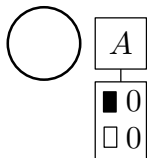
A Game



A Game



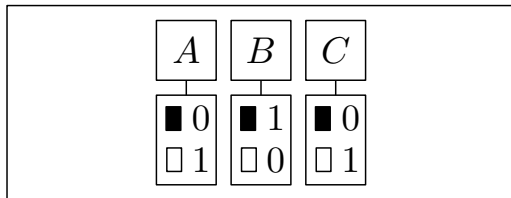
A Game



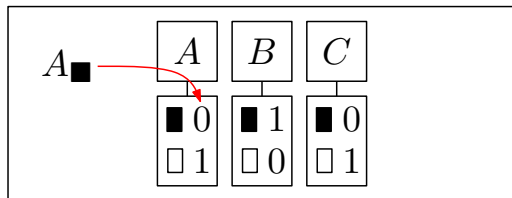
When do A , B and C Win?

A	B	C	
■	■	■	Odd
■	□	□	} Even
□	■	□	
□	□	■	

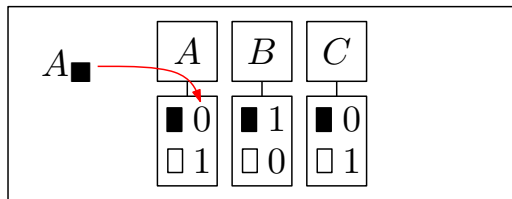
When do A , B and C Win?



When do A , B and C Win?



When do A , B and C Win?



$$A_{\blacksquare} + B_{\blacksquare} + C_{\blacksquare} = \text{Odd}$$

$$A_{\blacksquare} + B_{\square} + C_{\square} = \text{Even}$$

$$A_{\square} + B_{\blacksquare} + C_{\square} = \text{Even}$$

$$A_{\square} + B_{\square} + C_{\blacksquare} = \text{Even}$$

A, B e C Cannot Win!

$$2A_{\blacksquare} + 2A_{\square} + 2B_{\blacksquare} + 2B_{\square} + 2C_{\blacksquare} + 2C_{\square} = \text{Odd}$$

A, B e C Cannot Win!

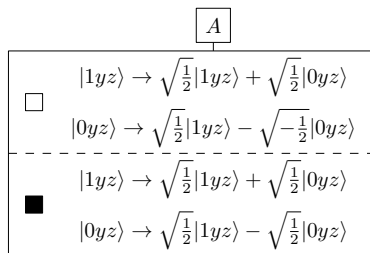
$$2A_{\blacksquare} + 2A_{\square} + 2B_{\blacksquare} + 2B_{\square} + 2C_{\blacksquare} + 2C_{\square} = \text{Odd}$$


Quantum Strategies

A	
\square	$ 1yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle + \sqrt{\frac{1}{2}} 0yz\rangle$
	$ 0yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle - \sqrt{\frac{1}{2}} 0yz\rangle$

\blacksquare	$ 1yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle + \sqrt{\frac{1}{2}} 0yz\rangle$
	$ 0yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle - \sqrt{\frac{1}{2}} 0yz\rangle$

Quantum Strategies



If ■ arrives...

Quantum Strategies

	A
□	$ 1yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle + \sqrt{\frac{1}{2}} 0yz\rangle$
	$ 0yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle - \sqrt{-\frac{1}{2}} 0yz\rangle$

■	$ 1yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle + \sqrt{\frac{1}{2}} 0yz\rangle$
	$ 0yz\rangle \rightarrow \sqrt{\frac{1}{2}} 1yz\rangle - \sqrt{\frac{1}{2}} 0yz\rangle$

If ■ arrives...

$$\sqrt{\frac{1}{2}}|111\rangle + \sqrt{\frac{1}{2}}|000\rangle \rightarrow \frac{1}{2}|111\rangle + \frac{1}{2}|011\rangle + \frac{1}{2}|100\rangle - \frac{1}{2}|000\rangle$$

Suppose that A, B, C Receive ■ ...

$$\begin{aligned}\sqrt{\frac{1}{2}}|111\rangle + \sqrt{\frac{1}{2}}|000\rangle \rightarrow & \frac{1}{4}|111\rangle + \frac{1}{4}|110\rangle + \frac{1}{4}|111\rangle - \frac{1}{4}|110\rangle \\ & + \frac{1}{4}|101\rangle + \frac{1}{4}|100\rangle - \frac{1}{4}|101\rangle + \frac{1}{4}|100\rangle \\ & + \frac{1}{4}|011\rangle + \frac{1}{4}|010\rangle - \frac{1}{4}|011\rangle + \frac{1}{4}|010\rangle \\ & + \frac{1}{4}|001\rangle + \frac{1}{4}|000\rangle + \frac{1}{4}|001\rangle - \frac{1}{4}|000\rangle\end{aligned}$$

Suppose that A, B, C Receive ■ ...

$$\begin{aligned}\sqrt{\frac{1}{2}}|111\rangle + \sqrt{\frac{1}{2}}|000\rangle &\rightarrow \frac{1}{4}|111\rangle + \frac{1}{4}|110\rangle + \frac{1}{4}|111\rangle - \frac{1}{4}|110\rangle \\ &+ \frac{1}{4}|101\rangle + \frac{1}{4}|100\rangle - \frac{1}{4}|101\rangle + \frac{1}{4}|100\rangle \\ &+ \frac{1}{4}|011\rangle + \frac{1}{4}|010\rangle - \frac{1}{4}|011\rangle + \frac{1}{4}|010\rangle \\ &+ \frac{1}{4}|001\rangle + \frac{1}{4}|000\rangle + \frac{1}{4}|001\rangle - \frac{1}{4}|000\rangle \\ &= \frac{1}{2}|111\rangle + \frac{1}{2}|100\rangle + \frac{1}{2}|010\rangle - \frac{1}{2}|001\rangle\end{aligned}$$

Suppose that A, B, C Receive ■ ...

$$\begin{aligned} \sqrt{\frac{1}{2}}|111\rangle + \sqrt{\frac{1}{2}}|000\rangle &\rightarrow \frac{1}{4}|111\rangle + \frac{1}{4}|110\rangle + \frac{1}{4}|111\rangle - \frac{1}{4}|110\rangle \\ &\quad + \frac{1}{4}|001\rangle + \frac{1}{4}|000\rangle + \frac{1}{4}|001\rangle - \frac{1}{4}|000\rangle \\ &= \frac{1}{2}|111\rangle + \frac{1}{2}|100\rangle + \frac{1}{2}|010\rangle - \frac{1}{2}|001\rangle \end{aligned}$$

The key ingredients are

- ▶ entanglement;
- ▶ a negative form of interference.

No-Cloning Theorem

- ▶ Let us consider a quantum system H_n with computational base $\mathbf{x}_1, \dots, \mathbf{x}_n$.
- ▶ A unitary transformation \mathbf{Q} on $H_n \otimes H_n$ is a **cloning map** if $\mathbf{Q} |\mathbf{x}\rangle |\mathbf{x}_1\rangle = |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle$.

No-Cloning Theorem

- ▶ Let us consider a quantum system H_n with computational base $\mathbf{x}_1, \dots, \mathbf{x}_n$.
- ▶ A unitary transformation \mathbf{Q} on $H_n \otimes H_n$ is a **cloning map** if $\mathbf{Q} |\mathbf{x}\rangle |\mathbf{x}_1\rangle = |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle$.

Theorem (No-Cloning)

For $n \geq 2$, there is no cloning map.

No-Cloning Theorem

- ▶ Let us consider a quantum system H_n with computational base $\mathbf{x}_1, \dots, \mathbf{x}_n$.
- ▶ A unitary transformation \mathbf{Q} on $H_n \otimes H_n$ is a **cloning map** if $\mathbf{Q} |\mathbf{x}\rangle |\mathbf{x}_1\rangle = |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle$.

Theorem (No-Cloning)

For $n \geq 2$, there is no cloning map.

- ▶ This does *not* mean that states **of the computational base** cannot be cloned.

No-Cloning Theorem

- ▶ Let us consider a quantum system H_n with computational base $\mathbf{x}_1, \dots, \mathbf{x}_n$.
- ▶ A unitary transformation \mathbf{Q} on $H_n \otimes H_n$ is a **cloning map** if $\mathbf{Q} |\mathbf{x}\rangle |\mathbf{x}_1\rangle = |\mathbf{x}\rangle \otimes |\mathbf{x}\rangle$.

Theorem (No-Cloning)

For $n \geq 2$, there is no cloning map.

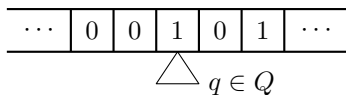
- ▶ This does *not* mean that states **of the computational base** cannot be cloned.
- ▶ There is an analogous theorem about **erasing maps**.

Section 3

Deterministic and Probabilistic Computational Models: a Quick Overview

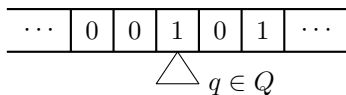
(Probabilistic) Turing Machines

- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.



(Probabilistic) Turing Machines

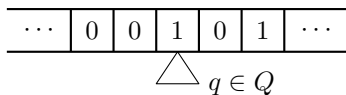
- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.



- ▶ Generalizing ordinary Turing Machines to the nondeterministic or probabilistic setting is a relatively easy task.

(Probabilistic) Turing Machines

- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.

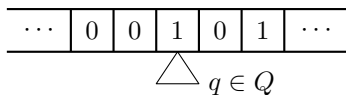


- ▶ Generalizing ordinary Turing Machines to the nondeterministic or probabilistic setting is a relatively easy task.
- ▶ A **Turing Machine** is a quadruple (Q, Σ, δ, F) .
 - ▶ In presence of **determinism**,

$$\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}.$$

(Probabilistic) Turing Machines

- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.



- ▶ Generalizing ordinary Turing Machines to the nondeterministic or probabilistic setting is a relatively easy task.
- ▶ A **Turing Machine** is a quadruple (Q, Σ, δ, F) .
 - ▶ In presence of **determinism**,

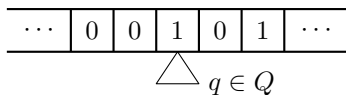
$$\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}.$$

- ▶ When evolution is **nondeterministic**,

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}).$$

(Probabilistic) Turing Machines

- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.



- ▶ Generalizing ordinary Turing Machines to the nondeterministic or probabilistic setting is a relatively easy task.
- ▶ A **Turing Machine** is a quadruple (Q, Σ, δ, F) .
 - ▶ In presence of **determinism**,

$$\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}.$$

- ▶ When evolution is **nondeterministic**,

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}).$$

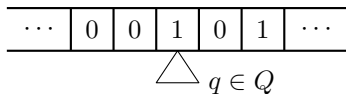
- ▶ When evolution is **probabilistic**,

$$\delta : Q \times \Sigma \rightarrow \mathcal{D}(Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}).$$

where $\mathcal{D}(A)$ is the set of all functions f from A to \mathbb{R} such that $\sum_{a \in A} f(a) = 1$.

(Probabilistic) Turing Machines

- ▶ Turing machines are one of the (many equivalent) ways to formalize the notion of universal computation.



- ▶ Generalizing ordinary Turing Machines to the nondeterministic or probabilistic setting is a relatively easy task.
- ▶ A **Turing Machine** is a quadruple (Q, Σ, δ, F) .
 - ▶ In presence of **determinism**,

$$\delta : Q \times \Sigma \rightarrow Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}.$$

- ▶ When evolution is **nondeterministic**,

$$\delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}).$$

- ▶ When evolution is **probabilistic**,

$$\delta : Q \times \Sigma \rightarrow \mathcal{D}(Q \times \Sigma \times \{\leftarrow, \downarrow, \rightarrow\}).$$

where $\mathcal{D}(A)$ is the set of all functions f from A to \mathbb{R} such that $\sum_{a \in A} f(a) = 1$.

- ▶ In PTMs, any computation $\rho : C_1 \rightarrow C_2 \rightarrow \dots \rightarrow C_n$ has a probability.

PTMs, Languages and Functions

- ▶ In PTM, **termination** and **acceptance of a string** become probabilistic events.
- ▶ A PTM can terminate with probability 1 even if there is the *possibility* of nontermination.
- ▶ A given string $s \in \Sigma^*$ is accepted *with a probability* between 0 and 1, included.
- ▶ How can we define language recognition?
 - ▶ One can consider different, alternative, constraints on the **error** probability:
 - ▶ It must be 0.
 - ▶ It must be smaller than $\frac{1}{2}$;
 - ▶ It must be smaller than a given fixed constant ε , itself smaller than $\frac{1}{2}$.
 - ▶ One can stipulate that the given result is the one with **highest probability**.

Computing Functions and Solve Problems

- ▶ How could we formalize that a function $f : \mathbb{F}^* \rightarrow \mathbb{F}^*$ can be computed **mechanically**?

Computing Functions and Solve Problems

- ▶ How could we formalize that a function $f : \mathbb{F}^* \rightarrow \mathbb{F}^*$ can be computed **mechanically**?
 - ▶ The answer is simple: is there a TM computing f ?
 - ▶ A TM **computes** f iff whenever $s \in \mathbb{F}^*$ the machine always halts producing $f(s) \in \mathbb{F}^*$ in output.
 - ▶ \mathcal{FR} is the corresponding class of functions.

Computing Functions and Solve Problems

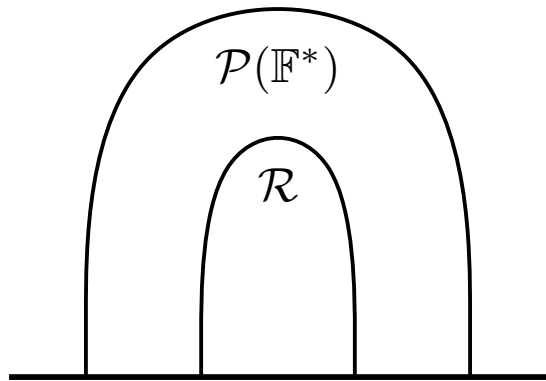
- ▶ How could we formalize that a function $f : \mathbb{F}^* \rightarrow \mathbb{F}^*$ can be computed **mechanically**?
 - ▶ The answer is simple: is there a TM computing f ?
 - ▶ A TM **computes** f iff whenever $s \in \mathbb{F}^*$ the machine always halts producing $f(s) \in \mathbb{F}^*$ in output.
 - ▶ \mathcal{FR} is the corresponding class of functions.
- ▶ What could we say about **computational problems**, i.e., about any $L \subseteq \mathbb{F}^*$?

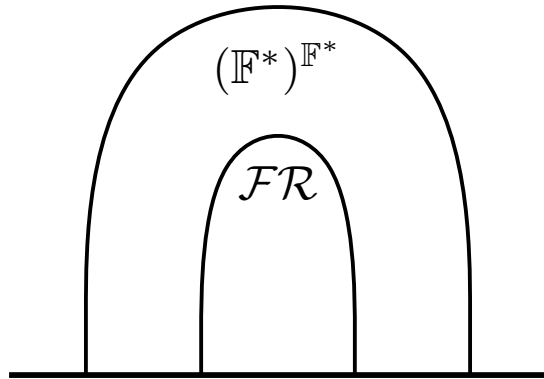
Computing Functions and Solve Problems

- ▶ How could we formalize that a function $f : \mathbb{F}^* \rightarrow \mathbb{F}^*$ can be computed **mechanically**?
 - ▶ The answer is simple: is there a TM computing f ?
 - ▶ A TM **computes** f iff whenever $s \in \mathbb{F}^*$ the machine always halts producing $f(s) \in \mathbb{F}^*$ in output.
 - ▶ \mathcal{FR} is the corresponding class of functions.
- ▶ What could we say about **computational problems**, i.e., about any $L \subseteq \mathbb{F}^*$?
 - ▶ L is said to be **recursive** iff the function f_L defined as follows is computable:

$$f_L(x) = \begin{cases} 0 & \text{if } x \in L \\ 1 & \text{otherwise} \end{cases}$$

- ▶ \mathcal{R} is the corresponding class of problems.





Computing with Limited Resources

- ▶ Some computable functions seem to require too much **time** (and/or **space**) to be computed.
- ▶ Let us consider the function $A : \mathbb{N}^2 \rightarrow \mathbb{N}$ defined as follows:

$$A(m, n) = \begin{cases} n + 1 & \text{if } m = 0; \\ A(m - 1, 1) & \text{if } m > 0 \text{ and } n = 0; \\ A(m - 1, A(m, n - 1)) & \text{if } m > 0 \text{ and } n > 0. \end{cases}$$

- ▶ The value $A(m, n)$ increases **too fast** when m and n grow, e.g.,

$$A(4, 3) \approx 10^{6031 \cdot 10^{19727}}.$$

- ▶ Ideally, we would like the time necessary to compute a function on $s \in \mathbb{F}^*$ to be bounded by a **fixed** polynomial on $|s|$.

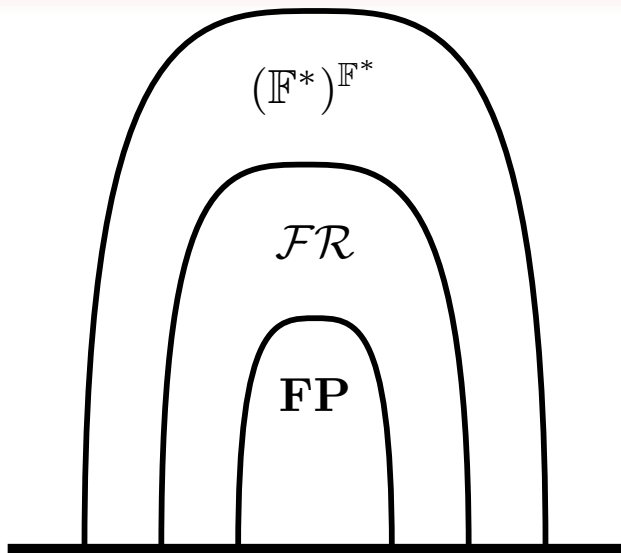
Polynomial Time

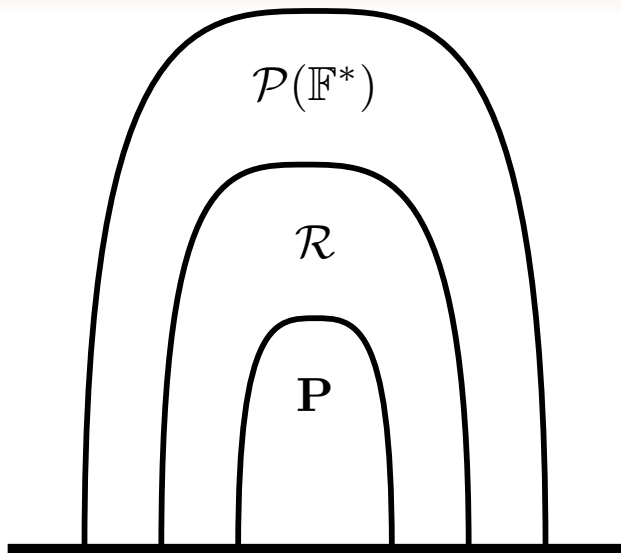
- ▶ A TM M **works** in time $g : \mathbb{N} \rightarrow \mathbb{N}$ iff for every $s \in \mathbb{F}^*$, the computation of M on input s takes at most $g(s)$ steps. In this case, we will write that $M \in \text{Time}(g)$.
- ▶ The complexity class **FP** is the collection of all functions from \mathbb{F}^* in \mathbb{F}^* which are computable by a TM M in

$$\bigcup_{g \in \text{POLY}} \text{Time}(g).$$

where POLY is the space of all polynomials from \mathbb{N} to \mathbb{N} .

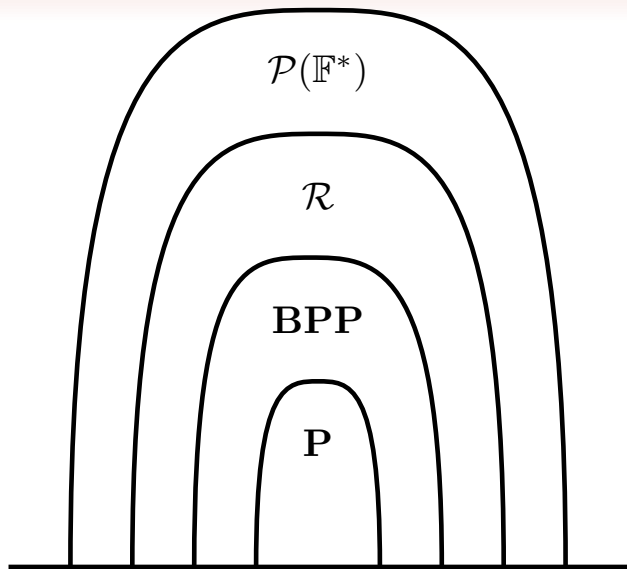
- ▶ Knowing that a function is **not** in **FP** can sometime help a lot.....
 - ▶ Proving such negative results can be very hard.

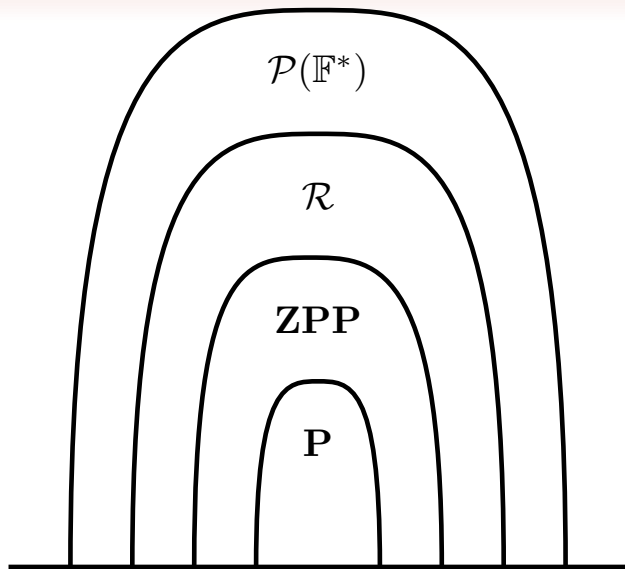




Probabilistic Polynomial Time

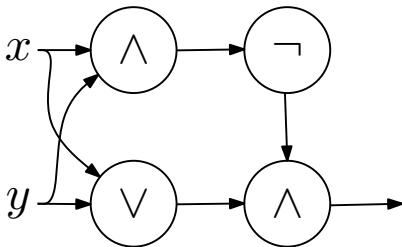
- ▶ How should we capture the concept of feasibility in a probabilistic setting?
- ▶ There is a tradeoff between bounding running times and bounding the error...
 - ▶ We can insist on having **no** error, but allow computation to take polynomial time only on the average, obtaining the class **ZPP**.
 - ▶ We can allow the error to be smaller than a constant $\varepsilon < \frac{1}{2}$, but insist on polynomial time bounds to hold independently on the random choices, getting the class **BPP**.
- ▶ There are many other possibilities, but these are somehow less interesting, giving rise to classes which are too large.





Boolean Circuits

- ▶ As we all know, they are **networks** of gates which compute very simple functions on $\mathbb{F} = \{0, 1\}$.
- ▶ Example:



- ▶ *Every* function $f : \mathbb{F}^n \rightarrow \mathbb{F}^m$ is computable by a boolean circuit C_f .
- ▶ Could we have finite, **universal**, set of gates?
 - ▶ Yes, e.g., $\{\wedge, \vee, \neg\}$.

But... How About Turing Machines?

- ▶ Boolean circuits can be seen as a **minimal** computational model, whose expressive power turns out to be **very poor**.

But... How About Turing Machines?

- ▶ Boolean circuits can be seen as a **minimal** computational model, whose expressive power turns out to be **very poor**.
- ▶ There are fundamental differences with Turing Machines:
 - ▶ In boolean circuits, everything is (even potentially) finite.
 - ▶ Functions computed by circuits and by TM are functions from different domains:
 $\mathbb{F}^n \rightarrow \mathbb{F}^m$ vs. $\mathbb{F}^* \rightarrow \mathbb{F}^*$.
 - ▶ Boolean circuits are somehow **complete**, while TM are not, as we all know.

But... How About Turing Machines?

- ▶ Boolean circuits can be seen as a **minimal** computational model, whose expressive power turns out to be **very poor**.
- ▶ There are fundamental differences with Turing Machines:
 - ▶ In boolean circuits, everything is (even potentially) finite.
 - ▶ Functions computed by circuits and by TM are functions from different domains: $\mathbb{F}^n \rightarrow \mathbb{F}^m$ vs. $\mathbb{F}^* \rightarrow \mathbb{F}^*$.
 - ▶ Boolean circuits are somehow **complete**, while TM are not, as we all know.
- ▶ Can we somehow reconcile the two worlds?
 - ▶ A **circuit family** is a collection $\{C_n\}_{n \in \mathbb{N}}$.
 - ▶ A circuit family **computes** a function from \mathbb{F}^* to \mathbb{F}^* .
 - ▶ A circuit family is said to be **uniform** if there is a TM which, on input n , outputs (an encoding) of C_n .

But... How About Turing Machines?

- ▶ Boolean circuits can be seen as a **minimal** computational model, whose expressive power turns out to be **very poor**.
- ▶ There are fundamental differences with Turing Machines:
 - ▶ In boolean circuits, everything is (even potentially) finite.
 - ▶ Functions computed by circuits and by TM are functions from different domains: $\mathbb{F}^n \rightarrow \mathbb{F}^m$ vs. $\mathbb{F}^* \rightarrow \mathbb{F}^*$.
 - ▶ Boolean circuits are somehow **complete**, while TM are not, as we all know.
- ▶ Can we somehow reconcile the two worlds?
 - ▶ A **circuit family** is a collection $\{C_n\}_{n \in \mathbb{N}}$.
 - ▶ A circuit family **computes** a function from \mathbb{F}^* to \mathbb{F}^* .
 - ▶ A circuit family is said to be **uniform** if there is a TM which, on input n , outputs (an encoding) of C_n .

Theorem

The class of functions computed by TMs equals the class of functions computed by uniform circuit families.

Section 4

Quantum Computational Models

Quantum Circuits

- ▶ They themselves consist in **gate networks**, which can be of three different kinds:
 - ▶ **Quantum gates**, which compute unitary transforms on $H_2 \otimes \dots \otimes H_2$.
 - ▶ **Measurement gates**, which observe the value of one qubit.

Quantum Circuits

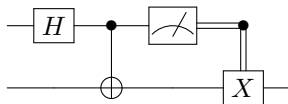
- ▶ They themselves consist in **gate networks**, which can be of three different kinds:
 - ▶ **Quantum gates**, which compute unitary transforms on $H_2 \otimes \dots \otimes H_2$.
 - ▶ **Measurement gates**, which observe the value of one qubit.
- ▶ The wires connecting gates are themselves of two types:
 - ▶ **Quantum channels**, which connect quantum gates to other quantum gates or to measurement gates.
 - ▶ **Classic channels**, which are the output wires of measurement gates and which control the application of quantum gates.

Quantum Circuits

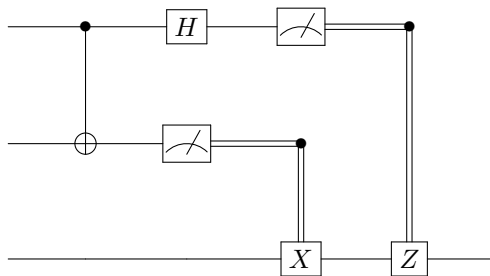
- ▶ They themselves consist in **gate networks**, which can be of three different kinds:
 - ▶ **Quantum gates**, which compute unitary transforms on $H_2 \otimes \dots \otimes H_2$.
 - ▶ **Measurement gates**, which observe the value of one qubit.
- ▶ The wires connecting gates are themselves of two types:
 - ▶ **Quantum channels**, which connect quantum gates to other quantum gates or to measurement gates.
 - ▶ **Classic channels**, which are the output wires of measurement gates and which control the application of quantum gates.
- ▶ Qubits cannot be duplicated.

Quantum Circuits

- ▶ They themselves consist in **gate networks**, which can be of three different kinds:
 - ▶ **Quantum gates**, which compute unitary transforms on $H_2 \otimes \dots \otimes H_2$.
 - ▶ **Measurement gates**, which observe the value of one qubit.
- ▶ The wires connecting gates are themselves of two types:
 - ▶ **Quantum channels**, which connect quantum gates to other quantum gates or to measurement gates.
 - ▶ **Classic channels**, which are the output wires of measurement gates and which control the application of quantum gates.
- ▶ Qubits cannot be duplicated.
- ▶ Example:



Quantum Teleportation



Boolean Circuits as Quantum Circuits

- ▶ When can a boolean gate be considered a quantum gate?
- ▶ How to simulate copying?

Boolean Circuits as Quantum Circuits

- ▶ When can a boolean gate be considered a quantum gate?
- ▶ How to simulate copying?
- ▶ First of all, we need to allow boolean gates to have **more than one** outputs.
- ▶ Then, we need Restrict to **invertible** gates.
 - ▶ Indeed, every unitary transform \mathbf{Q} has an inverse!
 - ▶ We can however go through the so called Toffoli gate, which is invertible:



Input			Output		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1

Input			Output		
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

Boolean Circuits as Quantum Circuits

- ▶ When can a boolean gate be considered a quantum gate?
- ▶ How to simulate copying?
- ▶ First of all, we need to allow boolean gates to have **more than one** outputs.
- ▶ Then, we need Restrict to **invertible** gates.
 - ▶ Indeed, every unitary transform \mathbf{Q} has an inverse!
 - ▶ We can however go through the so called Toffoli gate, which is invertible:



Input			Output		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1

Input			Output		
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

- ▶ Are we losing anything?

Boolean Circuits as Quantum Circuits

- ▶ When can a boolean gate be considered a quantum gate?
- ▶ How to simulate copying?
- ▶ First of all, we need to allow boolean gates to have **more than one** outputs.
- ▶ Then, we need Restrict to **invertible** gates.
 - ▶ Indeed, every unitary transform \mathbf{Q} has an inverse!
 - ▶ We can however go through the so called Toffoli gate, which is invertible:



Input			Output		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1

Input			Output		
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

- ▶ Are we losing anything?

Theorem

Every boolean function can be computed by a quantum circuit.

Universal Collections of Gates

- ▶ Is it possible to single out a **finite** set of quantum gates such that **all** quantum circuits can be expressed by way of these ports?

Universal Collections of Gates

- ▶ Is it possible to single out a **finite** set of quantum gates such that **all** quantum circuits can be expressed by way of these ports?

Theorem

*Every quantum circuit is equivalent to a circuit built from **unary** quantum gates and the **CNOT** port.*

Universal Collections of Gates

- ▶ Is it possible to single out a **finite** set of quantum gates such that **all** quantum circuits can be expressed by way of these ports?

Theorem

*Every quantum circuit is equivalent to a circuit built from **unary** quantum gates and the **CNOT** port.*

Theorem

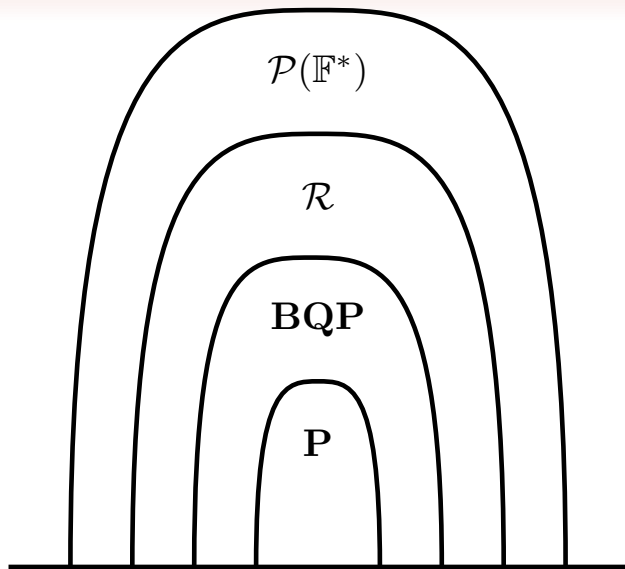
*Every quantum circuit is approximately equivalent to a circuit built from **CNOT** and from four specific quantum gates.*

Quantum Polynomial Time

- ▶ If, when defining **FP**, we replace the TM M with a **quantum** TM N , then the underlying complexity class is, again different, at least in principle.

Quantum Polynomial Time

- ▶ If, when defining **FP**, we replace the TM M with a **quantum** TM N , then the underlying complexity class is, again different, at least in principle.
- ▶ It is very convenient, however, to consider **quantum circuit families** which are generated by deterministic Turing Machines.
 - ▶ Timing constraints are captured by imposing that the generating TM works in polynomial time.
- ▶ The result of a quantum computation, however, is not a string in \mathbb{F}^* but, instead, a **probability distribution** on \mathbb{F}^* .
- ▶ But of course, we should also take the **error** into account
 - ▶ It suffices to impose that the error is strictly smaller than a constant $\varepsilon < \frac{1}{2}$, independently on the size of the input.
 - ▶ This way we obtain the complexity class **BQP**.



Thank You!

Questions?