A gentle introduction to Quantum Computing

Massimo Bernaschi

Istituto per le Applicazioni del Calcolo “Mauro Picone”

Consiglio Nazionale delle Ricerche

Via dei Taurini, 19 - 00185 Rome - Italy

e-mail: massimo.bernaschi@cnr.it

Giving credit where credit is due

This class is inspired by

Quantum Computing for Computer Scientists

https://youtu.be/F_Riqjdh2oM

Quantum Computing

- is not simply a “much faster” digital computer;
  - in some sense, it is not *digital* at all!
- does not evaluate “all possibile solutions” at the same time;
- can do whatever digital computing does
  (not more but, hopefully, “faster”).

Classic Computing as a special case of QC (1)

Let us start with classic bits represented as vectors.

- Bit 0 is represented by the $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ vector ($|0\rangle$ in Dirac's notation).
- Bit 1 is represented by the $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ vector ($|1\rangle$ in Dirac's notation).

Computing is carried out by multiplying suitable matrices for vectors representing bits.

Classic Computing as a special case of QC (2)

Matrix $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ inverts the second and third row of a vector:

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

One-bit operations

Operation	**Input**	**Output**
Identity	0 1	0 1
Negation	0 1	1 0
Constant 0	0 1	0 0
Constant 1	0 1	1 1

All these operations can be carried out using matrices

$$\textbf{Identity} \quad \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} ; \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$\textbf{Negation} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} ; \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\textbf{Constant 0} \quad \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} ; \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\textbf{Constant 1} \quad \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} ; \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The first two operations are *reversible*: if you know the output and the operation, you can tell what was the input.

Reversible operations

Roughly...

- operations that just shuffle bits around, like permute them, are reversible.
- operations that erase bits and then overwrite them are not reversible.

A specific feature of QC is that it uses operations which are their own inverses

- by applying them twice, you just get back the original input value.

Tensor product

Not a formal definition! (let's say an operational definition...).

For vectors $\begin{pmatrix} x_0 \\ x_1 \end{pmatrix}$ and $\begin{pmatrix} y_0 \\ y_1 \end{pmatrix}$ is defined as follows

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_0 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \\ x_1 \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} x_0 y_0 \\ x_0 y_1 \\ x_1 y_0 \\ x_1 y_1 \end{pmatrix}$$

You may find it also with the name of *outer* product.

An example with numbers instead of symbols may (hopefully) help:

$$\begin{pmatrix} 1 \\ 2 \end{pmatrix} \otimes \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 4 \\ 6 \\ 8 \end{pmatrix}$$

The tensor product is just an extension of the concept of Cartesian product to make it linear!

The extension to three or more vectors is straightforward:

$$\begin{pmatrix} x_0 \\ x_1 \end{pmatrix} \otimes \begin{pmatrix} y_0 \\ y_1 \end{pmatrix} \otimes \begin{pmatrix} z_0 \\ z_1 \end{pmatrix} = \begin{pmatrix} x_0 y_0 z_0 \\ x_0 y_0 z_1 \\ x_0 y_1 z_0 \\ x_0 y_1 z_1 \\ x_1 y_0 z_0 \\ x_1 y_0 z_1 \\ x_1 y_1 z_0 \\ x_1 y_1 z_1 \end{pmatrix}.$$

Using *classic* bits:

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{1}$$

The structure will be the same regardless of the number of vectors: a single element equal to 1 and all the other elements equal to 0.

Combinations of classic bits

The combination for two bits are:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \; |01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix};$$

$$|10\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}; \; |11\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix};$$

Looking at the four bits combinations: $|00\rangle\,; |01\rangle\,; |10\rangle\,; |11\rangle$ as an ordered sequence $(0, 1, 2, 3)$, the element of the sequence gives the position of the single 1 element in the tensor product.

In general the representation of n bits is a vector of size 2^n called *product state*.

The **CNOT** operation

CNOT or *conditional not* operates on a pair of bits.

One bit plays the role of *control bit*. The other bit is the *target* bit.

- if the control bit it is 1, then the target bit is flipped
- if it is 0, then the target bit is unchanged

The control bit never changes.

If we have the most significant bit of a 2-bit system as control, and the least significant bit as target, we can use the following matrix

$C = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$ to implement the **CNOT** operation as follows:

$$C\,|10\rangle = C\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} =$$

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |11\rangle\,;$$

$$C|11\rangle = C\left(\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}0\\0\\0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix} =$$

$$\begin{pmatrix}0\\1\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix} = |10\rangle\ ;$$

$$C|00\rangle = C\left(\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}1\\0\\0\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} =$$

$$\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}1\\0\end{pmatrix} = |00\rangle\ ;$$

$$C\,|01\rangle = C\left(\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix}\right) = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix} =$$

$$\begin{pmatrix}1\\0\end{pmatrix}\otimes\begin{pmatrix}0\\1\end{pmatrix} = |01\rangle\,;$$

Classical computers are all built on the **NAND** gate.

CNOT is the analogous NAND for reversible computing.

It is used to build up larger and more complicated logical statements.

Actually, it is **not** possible to build every logical function with the CNOT gate.

It is necessary to use a Toffoli gate that has **two** control bits.

Qubits and QC

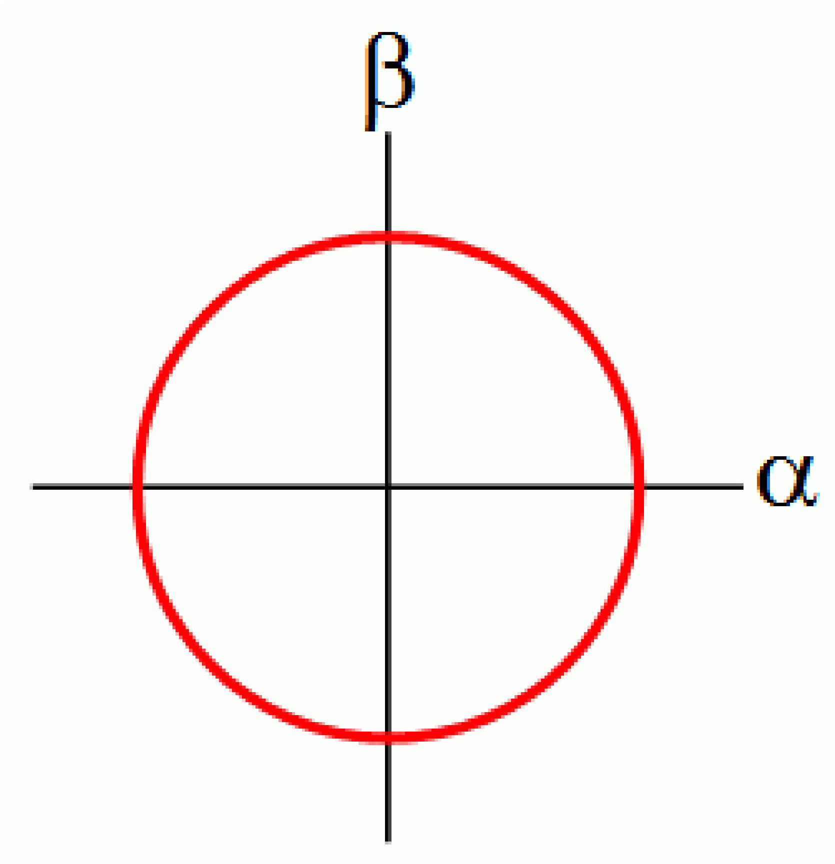
The classic bits are nothing else than a special case of *qubits*.

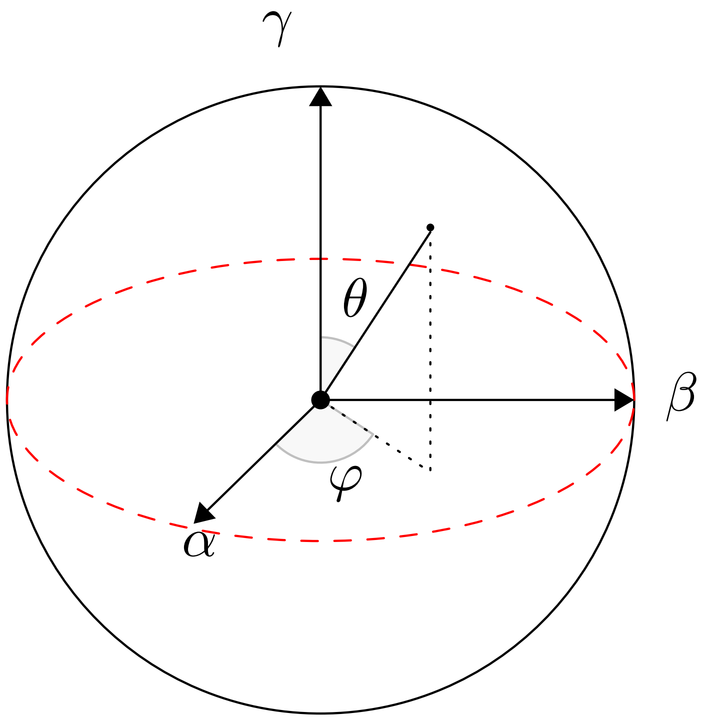
The general *qubit* is represented by a vector $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ of two elements α and β with α and β complex numbers such that $||\alpha||^2 + ||\beta||^2 = 1$.

For the sake of simplicity, we consider only *real* numbers ($\mathbb{R}$).

$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$ are examples of qubits.

A ($\mathbb{R}$) valued qubit can be represented as a point on an unitary circle.
In general a qubit is represented on the *Bloch*'s sphere.





Quantum *vs.* Classic probabilities (1)

- The fact that α and β may assume negative values plays an important role in the *tricky* inner workings of QC;
- a qubit has a value which is actually both zero and one at the *same* time
  - this is the property called *superposition* in a qualitative description of QC.
- Very roughly, since the qubit is in superposition we “compute” with the values of both zero and one at the same time;
- this does **not** mean we compute more than one solution at the same time!
- These two statements sounds contradictory but they are both true in the *Quantum* realm!

Quantum *vs.* Classic probabilities (2)

The crucial point is: when we *measure* a qubit, it collapses to the familiar values of either zero or one with some probability

- The probability of $\begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ to collapse to 0 is $||\alpha||^2$

  the probability to collapse to 1 is $||\beta||^2$

The *measure* is carried out at the end of a quantum computation to get the final result.

For instance $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ has a $||1/\sqrt{2}||^2 = \frac{1}{2}$ chance of collapsing to 0, and $\frac{1}{2}$ chance of collapsing into one. It's like a coin-flip.

- A *classical qubit*, $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ has a 100% chance of collapsing to zero
- $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ has a 100% chance of collapsing into one

Multiple qubits

Multiple qubits are also represented by the tensor product.

For instance: $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \otimes \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$ so there is a $\frac{1}{4}$ chance each of collapsing to $|00\rangle, |01\rangle, |10\rangle, |11\rangle$.

Operations on qubits

Also for qubits operations are carried out by using matrices.

For instance (applying *negation*):

The qubit $\begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix}$ has a 25% chance of collapsing to 0 and 75% chance of collapsing collapse into 1.

If we apply the bit flip operator $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{pmatrix}$

now there is a 75% chance of collapsing into 0 and 25% chance of collapsing into 1.

Quantum *circuits*

QC can be defined as "the art of manipulating those probabilities" (actually called *amplitudes* in the quantum jargon), with *quantum* gates (each one corresponding to a suitable matrix).

- a combination of quantum gates defines a quantum circuit;
- the qubits must be manipulated with a gentle touch...
  - not enough to collapse them
  - enough to change their state.

  this is one of the reasons why QC is still in its infancy!

The Hadamard gate (1)

The Hadamard gate corresponds to the $\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}$ matrix.

It takes a 0 or 1 qubit, and transforms it in the coin-flip state where it is in exactly equal superposition

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} ; H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

Note the $-$ (minus sign) in the lower right corner of the Hadamard matrix. Can you tell the reason of having a $-$ there?

The Hadamard gate (2)

The Hadamard gate puts the qubit in superposition.

Both $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$ and $\begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$ have the same probabilities from the classic viewpoint but they correspond to two *distinct* quantum states.

- The Hadamard gate may be used also to take out of superposition into the classical bits.
- If we apply the Hadamard gate to the coin-flip state, we obtain

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}; \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

A first insight to the “logic” of QC

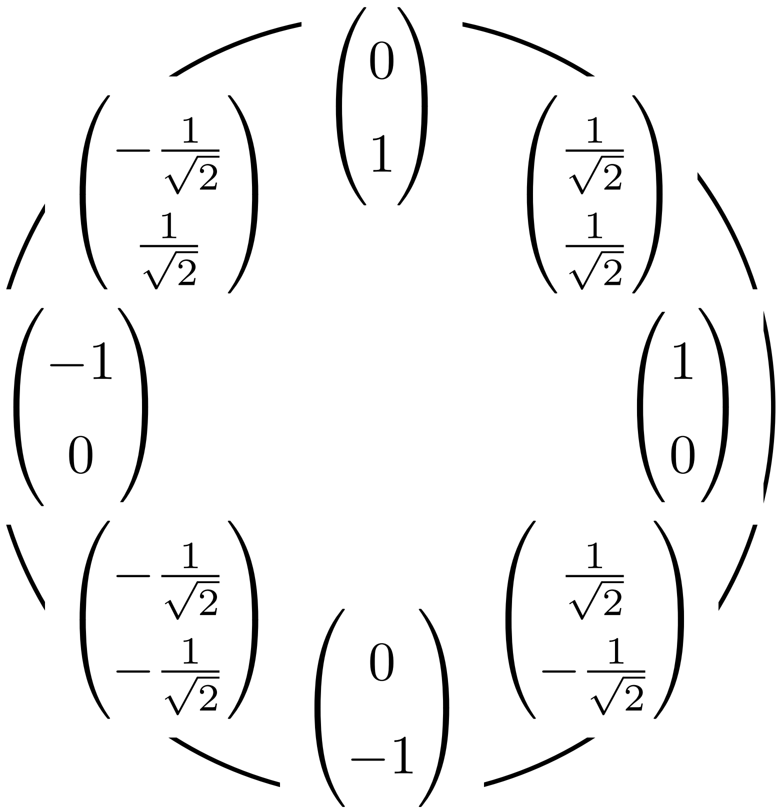
1. start with classical bit values;
2. put them into superposition;
3. carry out quantum transformations;
4. at the end, if we are clever, it is possible to move them back to zero or one so that there is a “high” chance of getting a “right” answer.

In general a QC algorithm is not deterministic!

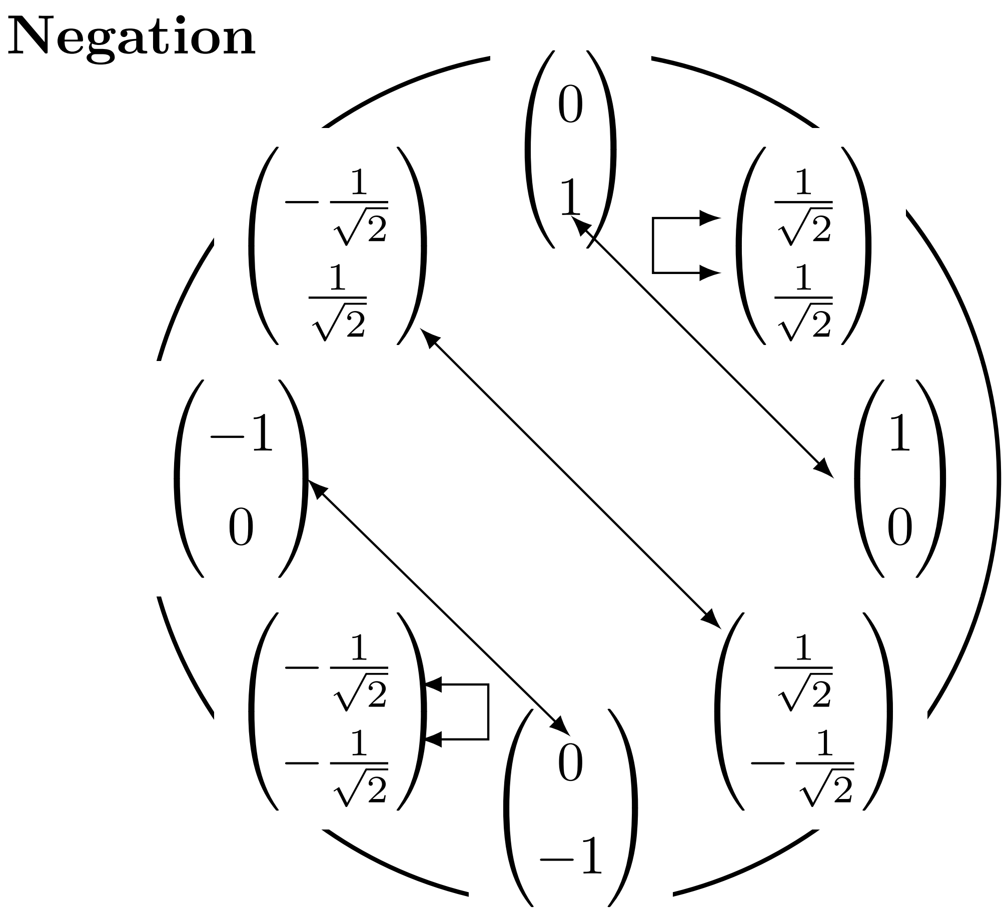
- the famous Shor’s algorithm for factoring large numbers only gives the right answer 50% of the times.

A Quantum Finite State Machine

A Finite State Machine (FSM) represents a system that can, at any point in time, be in a specific state from a *finite* set of possible states.

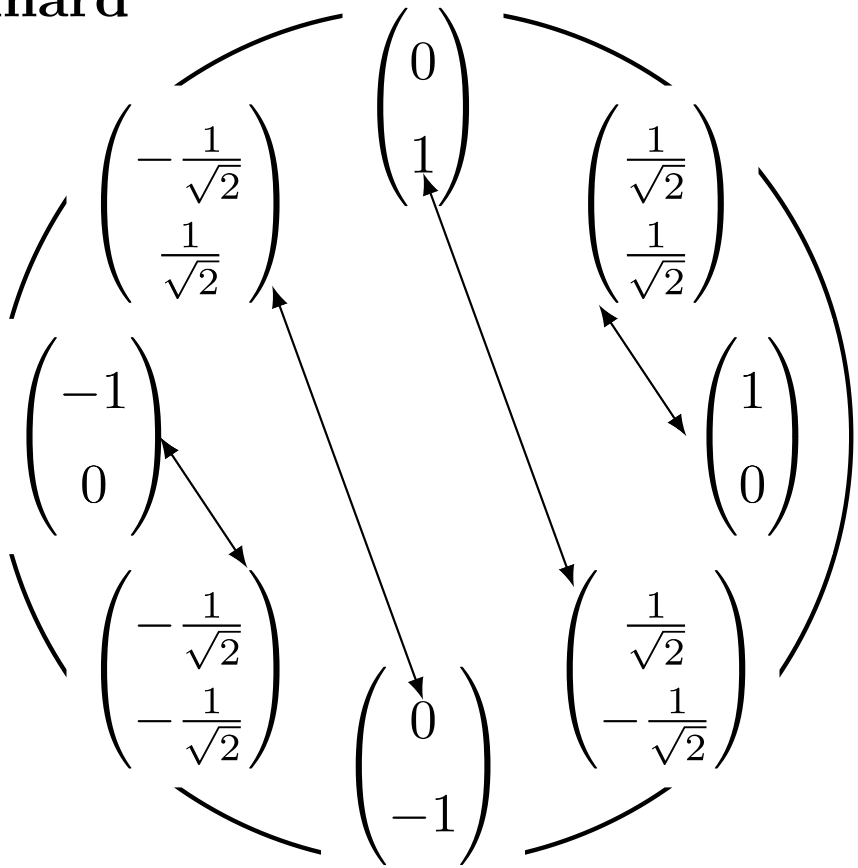


If we apply the **Negation** (bit flip) operator, we change the state (*i.e.*, position along the unit circle)



For the **Hadamard** operator

Hadamard



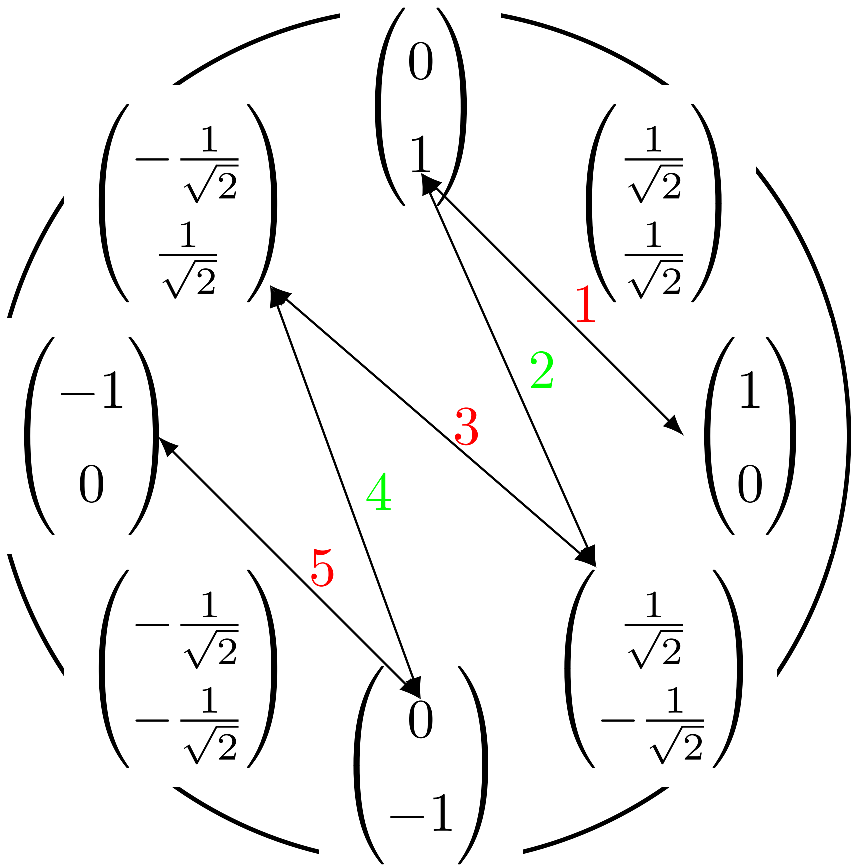
FSM as an alternative to matrix-vector products

For instance, we can draw a quantum circuit

$$\begin{pmatrix}1\\0\end{pmatrix} \;—\; \boxed{X} \;—\; \boxed{H} \;—\; \boxed{X} \;—\; \boxed{H} \;—\; \boxed{X} \;—\; \begin{pmatrix}?\\?\end{pmatrix}$$

where $\boxed{X}$ is a Negation gate and $\boxed{H}$ is a Hadamard gate.

What is $\begin{pmatrix}?\\?\end{pmatrix}$ at the end of the computation?



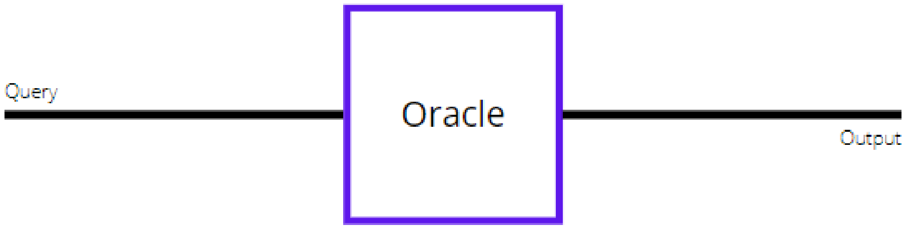
The result of applying the circuit to $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$ is $\begin{pmatrix} -1 \\ 0 \end{pmatrix}$

note that the computation is *reversible*.

The Deutsch's oracle problem

Problem is

- there is a *black-box* (*i.e.*, a device that can not be inspected) that takes in input one bit and returns one bit;
- it is possible to use the device, giving in input one bit and looking at the output bit;



1. how many queries would it take to determine the function on a classic computer?
2. how many on a quantum computer?