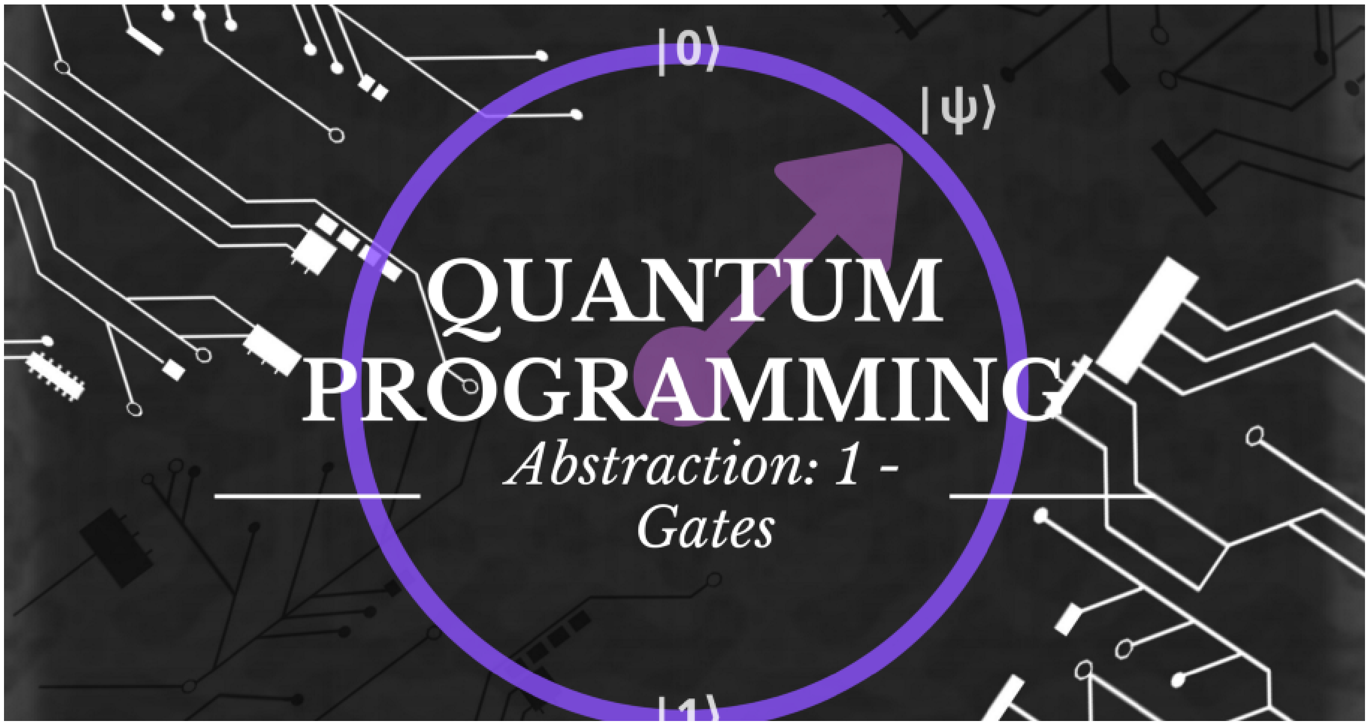
Quantum Programming — Abstraction level 1: Logic Gates

Kiril P. Blagoev 1 day ago · 11 min read ★

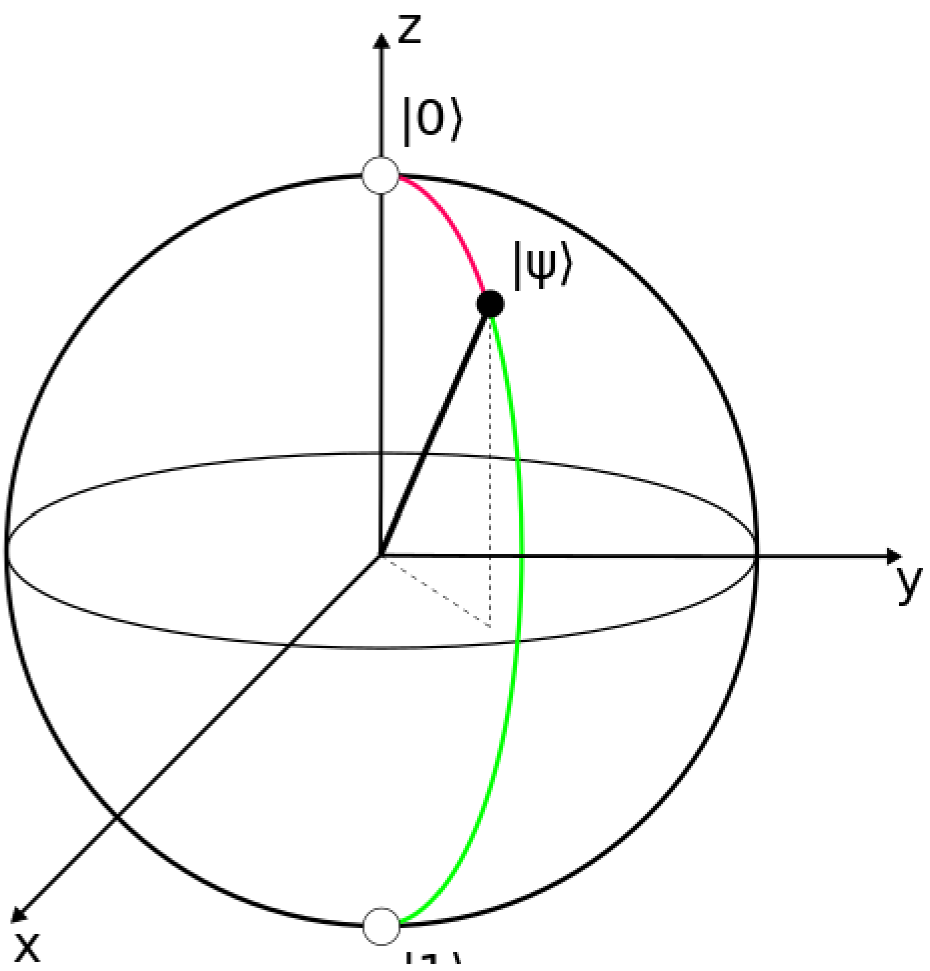
For the majority of our time spent programming on the quantum computer, we will be thinking of the qubits in their state of superposition. What would be useful for our intuition now, is to have a way to visualise superposition, so we can analyse the effects each logical gate has on it.



1. Quantum Programming by abstracting ourselves from Quantum Mechanics: Abstraction level 0
2. Quantum Programming — Abstraction level 1: Logic Gates (this)

We have previously worked out that a quantum computer operates logically on qubits, which are the quantum counterparts of classical bits. We've learned that a qubit can stay in a superposition while we operate on it, and then we can collapse it into a definite state upon measurement. So, for the majority of our time spent programming on the quantum computer, we will be thinking of the qubits in their state of superposition.

What would be extremely useful for our intuition now, would be to have a way to visualise this superposition. Enter the **Bloch Sphere**. This tool allows us to represent the whole state-space of a single qubit in one (very simple) geometric shape — the unit sphere. Think of it as the unit circle in trigonometry.



Let's discuss the things that we are interested in. The two poles of the sphere (along the z axis) we arbitrarily define as the qubit being in the 0 state in the north pole, and in the 1 state in the south pole. The current state of superposition of the qubit is defined by the black vector $|\psi\rangle$. Our current state is in such a superposition, that it's closer to the o state, than the 1 state. This simply means that when we measure this qubit, we will be more likely to observe it collapsing into a zero, than into a one.

Now, where it gets confusing, is that this is a three-dimensional sphere, and yet our state can collapse only into a single one-dimensional number — 1 or 0. Why do we even need a sphere, why not just a line with $|0\rangle$ at one end, and $|1\rangle$ at the other? Well, to make a long story very short, our state is actually defined by a two-dimensional vector, for example for the state $|1\rangle$[1 2],

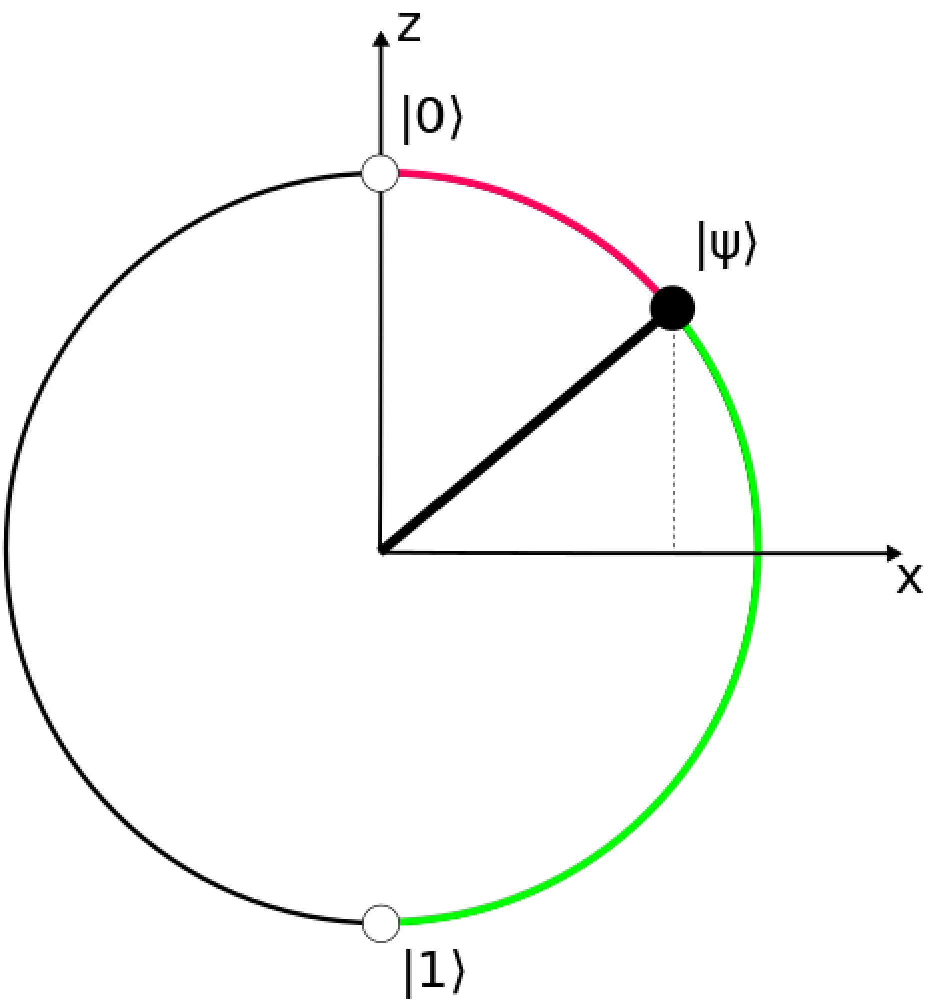
$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

and the components in that two-dimensional vector (called **complex amplitudes**) are actually complex numbers, as in for example

$$|1\rangle = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} + \frac{i}{\sqrt{2}} \end{bmatrix}.$$

All in all that needs 4 coordinates to be geometrically described (2 for each complex number), but for practical reasons we can ignore one of the dimensions, so we are left with 3, which we can map to the Bloch sphere.

imaginary part of the complex components, so we can temporarily get rid of one dimension, and be left with a circle instead. Told you, you can imagine a unit circle.



For the sake of just showing the probability of collapsing, we can for now get rid of the third dimension

The X-Gate

So why did we need a circle in the end, why not a line?

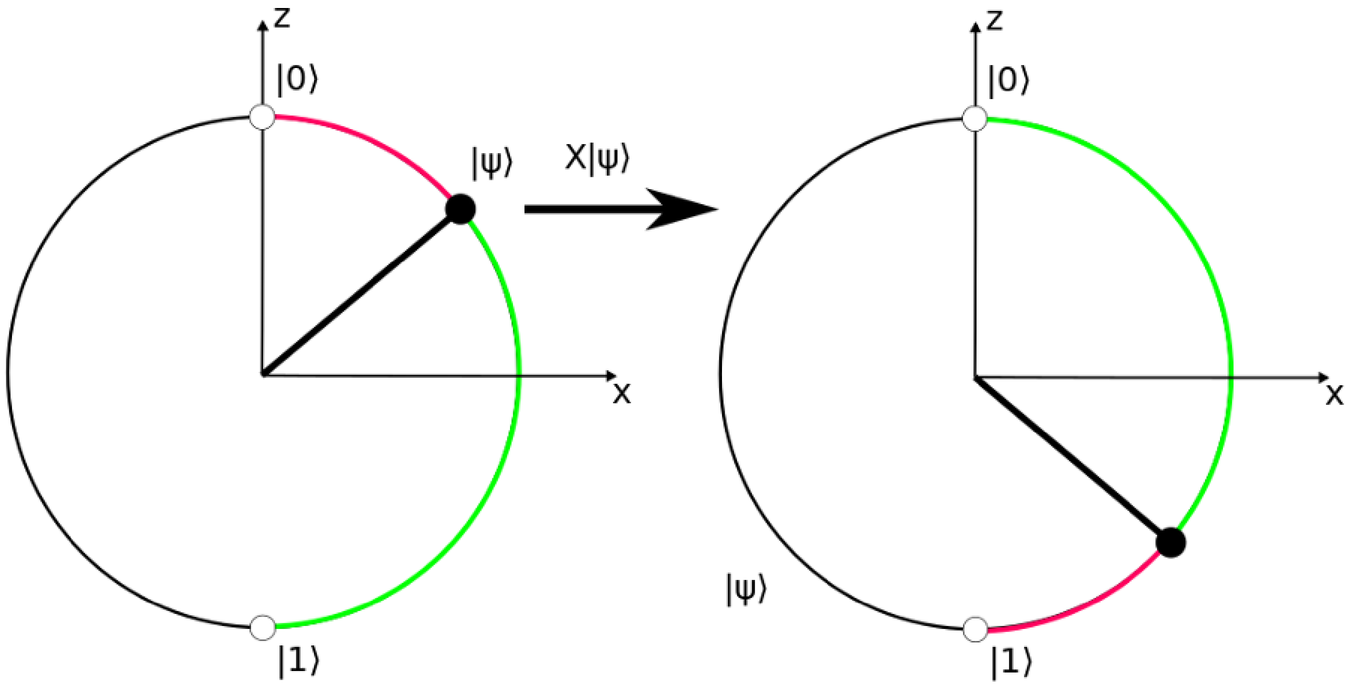
The X-Gate is an equivalent of the classical NOT gate — it negates the current value of our qubit. But the way it does it is by swapping the two components of our state-vector. For example if we apply it to the definite state of 0

$$X\left|0\right\rangle = X\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\1\end{bmatrix} = \left|1\right\rangle,$$

we get the definite state of 1. Or when acting on a superposition state,

$$X|\psi\rangle = X\begin{bmatrix}\frac{\sqrt{3}}{2}\\ \frac{1}{2}\end{bmatrix} = \begin{bmatrix}\frac{1}{2}\\ \frac{\sqrt{3}}{2}\end{bmatrix},$$

which when applied to our circle looks like so

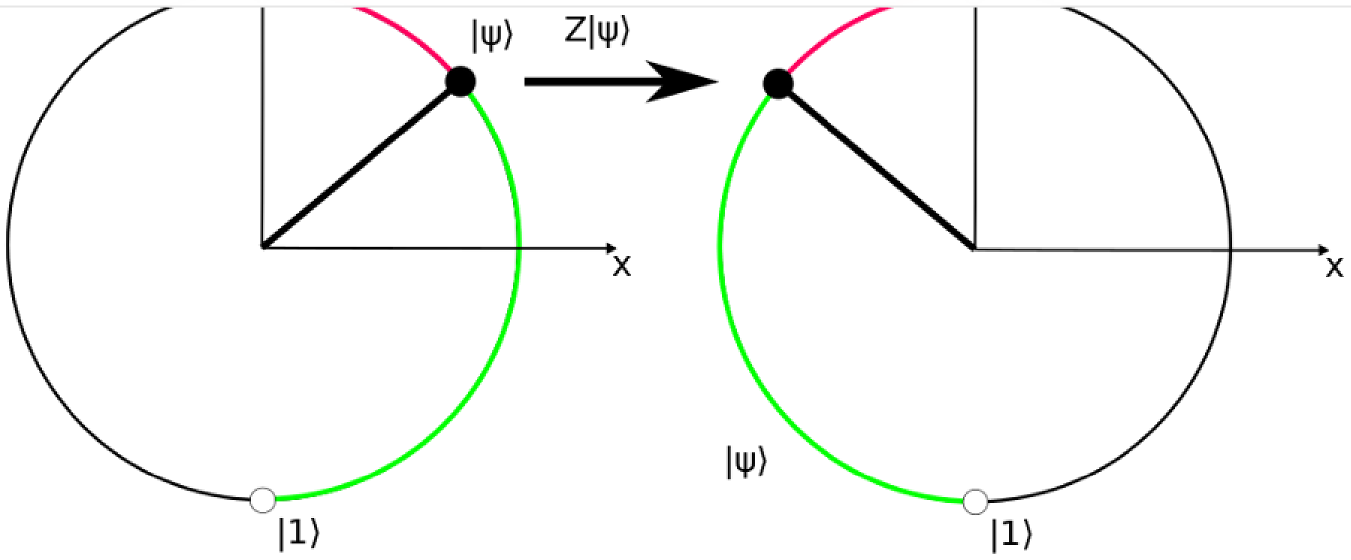


The new state is now opposite of the old one — the two components of its vector have been swapped.

> *So, in our case after applying the X-Gate our state-vector has the same probability of collapsing to |1⟩ as it did before of collapsing into |0⟩ (the state was negated).*

But why did we need a circle in the end, why not a line?

Mind you, we also have a Y-Gate and a Z-Gate, which do the same thing, but flip the position of the vector with respect to the other 2 coordinates. Let's take a look at the **Z-Gate** for example.



The Z-Gate did nothing to change the probability of our qubit

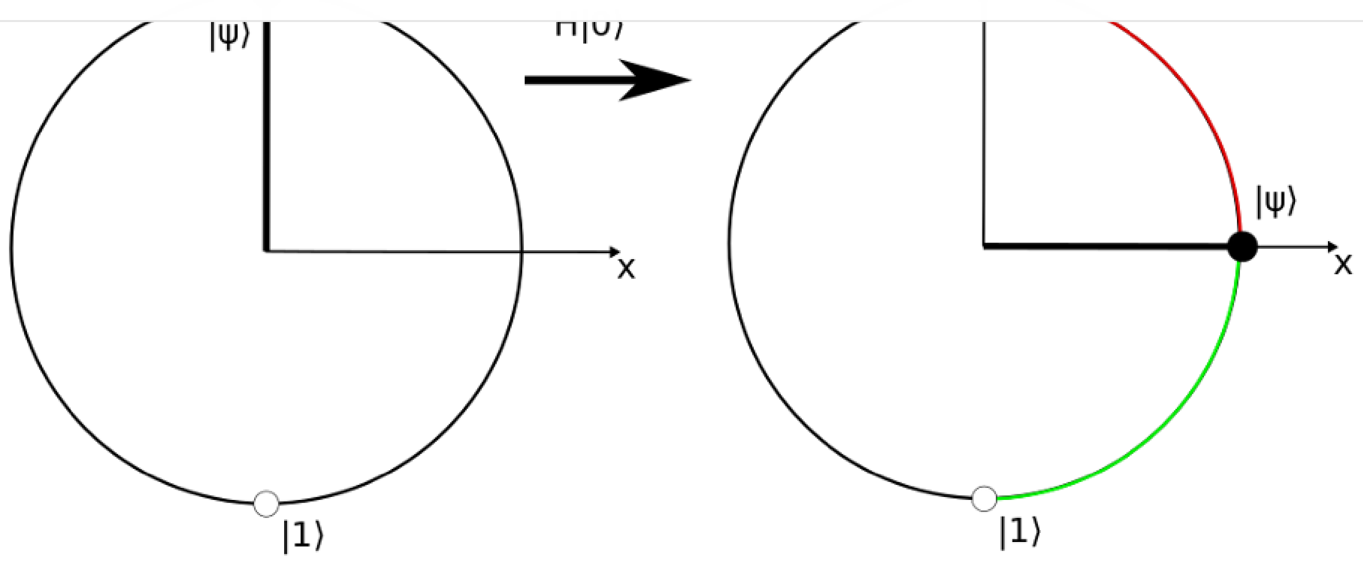
In our particular setup, by applying the Z-Gate (negation around the *z* axis) *did nothing* to change the *probability distrubution* of our qubit! But in order to represent what actually happened to the qubit, we needed the extra dimension provided to us by the circle relative to simply having a line.

> *We say that two states with the same probability distribution, but different state-vectors differ only by* ***phase****.*

The Hadamard Gate

This is a very important gate, fundamentally quantum, with no counterpart in classical computing. You could probably have guessed it, the H-gate is the one which puts a qubit from a definite state into a state of superposition. Say we had a qubit with a hundred percent chance of being a zero. When we apply the H-gate it would turn it into a 50/50 superposition state.

$$H|0\rangle = H\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\\frac{1}{\sqrt{2}}\end{bmatrix} = |+\rangle$$



The Hadamard Gate turned a definite 0 state into a superposition. It now has equal probability of collapsing into either state.

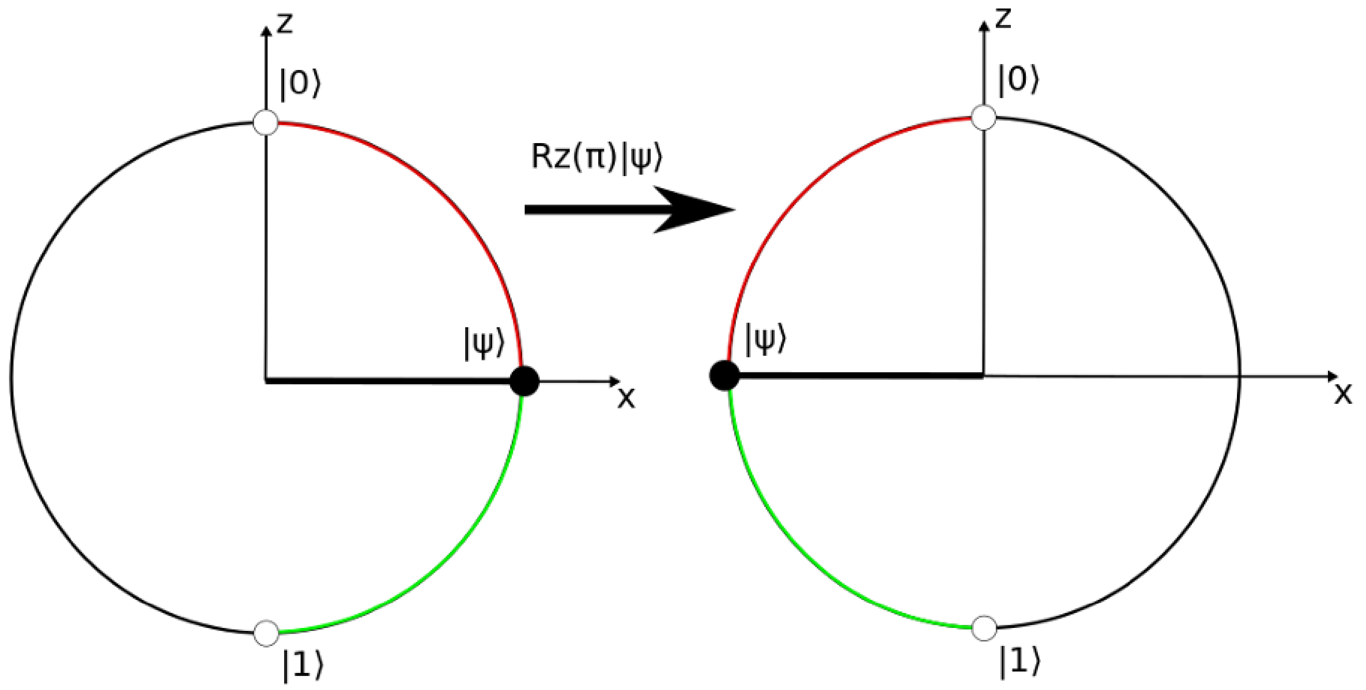
The Rz Gate

This is a gate which I want you to take notice of. This is a rotation around the z-axis gate. It has two things going for it — first of all it's a parametrised gate. This means that it takes a value, namely some angle, with which to rotate the vector. Second, I want to argue, that this is the most versitile gate there is.

Think about it, both the X-Gate and the Hadamard gate do one thing which is not unique to any one of them — they rotate the vector for the state to some position on the sphere. And the Rz gate does exactly that, but parametrised — it rotates the vector to any degree that we want it to rotate to. Of course, since we have 3 dimensions, we can rotate around any of these axes to make full use of the vector's capabilities. For example the Hadamard Gate is actually a rotation around the *y* axis by 90 degrees or π/2 radians, followed by a rotation around the *x* axis by 180 degrees or π radians. And the X-Gate is a rotation around the *z* axis by π. So arguably,

> all the gates are special cases of a linear combination of rotations around the Bloch sphere.

$$R_z(\pi)\,|+\rangle = R_z(\pi)\begin{bmatrix}\frac{1}{\sqrt{2}}\\ 1\end{bmatrix} = \begin{bmatrix}\frac{1}{\sqrt{2}}\\ -1\end{bmatrix} = |-\rangle$$



Rotating the 50/50 state called |+⟩ by π radians around the z axis results in the opposite 50/50 state |−⟩

The I Gate

If you've done anything with either computer science or linear algebra, you will know that there exists the so-called identity gate (matrix). This operator simply does nothing, it leaves the state-vector as it is. Its two main purposes are

1. Proving that the X-Gate is its own inverse.
2. Useful for specifying a "do-nothing" operation, i.e. *noop*.

The remainder of gates that can operate on one qubit are either special cases of the Rotation gate, or the general U-Gate which is the most generalised gate acting on one qubit.

Now let's take a look at an example of a **gate acting on two qubits**, namely the

CNOT Gate

The CNOT gate takes two qubits as parameters, one as the control $q0$, and one as the target $q1$. What the gate does is conditionally perform an **X-Gate** (negation (along the x

When our qubits are in definite states (and by extension, if we do this on a classical computer), this becomes very simple to grasp through a *truth table*

input (control,target)	output (control,target)
0,0	0,0
0,1	0,1
1,0	1,1
1,1	1,0

But if we have superposition involved, let's remind ourselves of two things:

1. In our quantum circuit, when a qubit is in superposition, it's simultaneously in both the 0 and the 1 state.
2. When we change the probability distribution of 1 qubit, we ultimately change the probability distribution of the states of the whole system of qubits.

What does this tell us? Well,

1. if say the control bit is in a superposition, then it will be in both the one and zero state.
2. So for certain states of the whole system it will be a one.
3. In those system states the target bit will be negated by the X-Gate as per the definition of the CNOT gate.
4. The probability distribution of the whole system will change.

This is where it really starts get mind-bogglingly difficult to follow the logic by simply imagining it. So let's analyse a few situations of using the CNOT with superposition involved.

Both the control and the target are Hadamard-ed

along *x* and |−⟩ when pointing towards *-x*. So we can initialise our qubits to be in either one of them, and then apply the CNOT gate on them using *q0* as the control, and *q1* as the target. After quite a bit of calculation we will see the following truth table

input $\lvert control, target\rangle$	**output** $\lvert control, target\rangle$
$\lvert ++\rangle$	$\lvert ++\rangle$
$\lvert +-\rangle$	$\lvert --\rangle$
$\lvert -+\rangle$	$\lvert -+\rangle$
$\lvert --\rangle$	$\lvert +-\rangle$

If we look closely we will notice something unexpected. Remember, in the CNOT gate the control bit is supposed to remain unchanged, and the target one to flip whenever it's supposed to. But in our scenario, the target qubit remains always unchanged, but the control qubit flips according to the state of the target bit.

> All of a sudden the sense of *control* and *target* bits has reversed.

This fact can be extended further, and used when building quantum algorithms. Suddenly new phenomena start appearing, such as *CNOT Circuit Identity* and *Phase Kickback*. We will explore these as we ascend one more step of abstraction next time.

Building a Bell State

When it comes to the physics of the qubits, there's one more weird phenomenon to explore, which comes directly from quantum mechanics — entanglement.

Let's take a pure definite state of two qubits such that |00⟩, or in other words both qubits are definitely zeroes. We can write that as a global state-vector[3]

$$\text{State-vector} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Or more clearly, in a table of probabilities[4]

probability	system after measured (q1,q0)
100%	00
0%	01
0%	10
0%	11

If we apply a Hadamard gate to the control bit making it into state $|+\rangle$, we get

$$\text{State-vector} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ 0 \\ 0 \end{bmatrix},$$

which we can read as " *q0* is in superposition, and *q1* is in definite state 0". In this scenario, our system probability table looks like so

probability	system after measured (q1,q0)
50%	00
50%	01
0%	10
0%	11

and we can still reason for each qubit separately, i.e. *q1* can't be a 1, and *q0* can be either.

$$\text{State-vector} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}.$$

Or in our more readable format of probability tables

probability	system after measured (q1,q0)
50%	00
0%	01
0%	10
50%	11

We can see, that the system now has 50% chance of being 00, and 50 of being 11.

But if we try reading this by splitting the state-vector into state-vectors of two separate qubits, we will be unpleasantly surprised.

- Let's say we look at state 00. It has 50% chance, which would mean that if we try to take only *q1*, we would expect it to have at least some probability of being 0 on its own.
- But if we look at state 01, it has 0% chance. This can maybe hint us into thinking that *q0* **can't** be a 1, and therefore 01 is impossible.
- But state 11 also has probability 50%. So *q0* **can** be a 1.

This no longer makes sense (as the probabilities don't add up to 1), and has interesting implications. We can no longer consider these two qubits as separate. Measuring one will affect the other.

Entanglement is the final purely-quantum concept that we needed to tackle. Now we have built an abstraction machine, which allows us to manipulate both single and multiple qubits at once, and started to see how dealing with multiple qubits can have extremely weird effects on our computational techniques. From now on we can concentrate on using this weirdness to build our next level of abstraction on our way to creating actual code — circuits and algorithms. So let's do that next time!

1. I guess it would be important here to note, that the probabilities that we talked so much about last time are the squares of the components (complex amplitudes) of the vector. So for example if our state is

$$\begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{-1}{2} \end{bmatrix},$$

then the qubit has (sqrt(3)/2)^2= 3/4 chance of collapsing into a 0, and (-1/2)^2 = 1/4 chance of collapsing into a 1. When we sum them up, we get a 1, which is what we would expect from probabilities.

2. Additionally, it's not a coincidence that the states are represented by vectors. All the gates that we can have on qubits are actually linear operators, which we can represent using matrices. You can never know too much linear algebra…

3. System state-vectors of the whole qubit circuit are created by placing the probabilities of all system states in a 4D matrix. The collective state of a number of qubits on the other hand is calculated using a tensor product of the state-vectors of each qubit.

4. Both notations are really showing the same thing, it's just that the state-vector shows the complex amplitude of that system state, while the probability table squares it to get the probability associated with that system state

$$\text{State-vector} = \begin{bmatrix} a_{01} \\ a_{10} \\ a_{11} \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 0 \\ 0 \end{bmatrix},$$

where *a* is the complex amplitude of each state of the two qubits *q1,q0*.

probability	system after measured (q1,q0)
$50\% = (\frac{1}{\sqrt{2}})^2$	00
$50\% = (\frac{1}{\sqrt{2}})^2$	01
$0\% = 0^2$	10
$0\% = 0^2$	11

Originally published at https://kblagoev.com on March 28, 2021.