The Implementation of a QC

The idea is to give insights about what the Hardware and Logic Andrew and Pranav have been talking about in their respective part can accomplish

Implementation mean in Object Oriented Programming (OOP) that a class provide the methods specified in the interface. However, implementing is not specific to OOP and concerns more generally the realisation of algorithms; algorithms which once carefully programmed should output a result that fits a given problem.

To study the quantum implementation, it’s therefore important to work out the logic and algorithms that drive these machines and make them so powerful. We are used to multi-purpose computers that are easily reprogrammable, going to lower level of details to higher order programming language is therefore a key in the development of QC and their implementation.

# The Quantum Algorithms

Algorithm exists from the dawn of computing. They are a sequence of steps that should lead to a logical, generally immutable, and meaningful outcome. We find computers better than humans to follow most algorithm as their capacity to calculate and treat data exceed us by far. However, it is now easy to spot the limits of Moore’s law in today computers and the computation time is still too slow to explore multiple solutions at once. For example, the analysis of multiple factor that impact a phenomenon requires the analysis of each one of them varying separately (weather predictions, deciphering communication, etc.). The algorithms for QC take advantage of the fact QC can explore multiple solutions at once.

## Algorithms on amplitude amplification

“Quantum mechanics can speed up a range of search applications over unsorted data.”, (Groover, 1997). Groover studied in his paper the problem of accessing an unsorted database of *N* states. Logic would tell you that on average you would need to scan half of the database on average before finding the required entry, or in the worst case all *N* entries of the database (*O(N)*). The Quantum algorithm uses properties of the amplitude of qubits. The algorithm can be described as follow:

Only one item (*I*) in the database satisfies the property *P(I)=1*, for all the others states *S* we have *P(S)=0*.

Initialise the system so each state has the same amplitude (superposition ).

Apply the transformation and if a state reaches *P(S)* =1, rotate the phase by 180 degrees.

Note that the transformation affects the amplitude of the desired state by and order of , the order of the loop, considering its initial condition is therefore *O()*, which is a gain of polynomial order compared to the classic algorithm.

## Fourier’s transform

The Discrete Fourier’s transform (DFT) represents the frequency domain of the given input. It can be applied to quantum algorithms and implemented very efficiently, making it a very robust tool to compute the Deutsch–Jozsa algorithm, finding eigenvalue of unitary operators and probably its most famous application is the use in Shor’s algorithm.

### Boson sampling

(Bryan T. Gard, 2015)

Reference

### Shor’s Algorithm

(Shor, 1997)

Shor’s algorithm has become very famous over the years as its realisation, if it should come, will affect radically the domain of communication and cryptography that protects everyone’s privacy. Decrypting an RSA transaction to a bank to retrieve the credit cards info could be done in polynomial time –in terms of log n- instead of exponential time leaving many system at risk.

The factorisation algorithm of a number *N* consists like many Quantum algorithms of a classic routine run on a regular computer, that routine triggers at some point a quantum routine, but the process is originally controlled by a regular computer. The main advantage of this method is that it allows to save resources on both machines as they are used to what they are the most efficient for. The Quantum routine is very specialised and only serves as finding the order of an element; its efficiency results in the ability to search a remaining square root of different from 1 and -1. We are sure it exists, as stated by the Chinese remainder theorem>>>>>>. Finding such a number and taking then will be a ‘proper’ factor of *N*.

Indeed is proper because:

* If , then by Bézout’s theorem,

Leading by a multiplication by on both sides that which contradicts how we obtained .

* If which also contradicts how we obtained S.
* Therefore we only have and d is a proper factor for N

The algorithm to find such an S is as follows:

1. Pick a random number
2. If return a
3. Else find the order of in the group and if or goto step 1
4. Else we’ve found our S!

Researchers have for now mostly limited themselves to factorising the number 15 (the smallest possible number that can test Shor’s algorithm. The runtime of Fourier’s transformation is still high and a field of improvement. But it’s probably one of the biggest success for QC which demonstrate their capacity on one the most complex Quantum algorithm that exists until these days.

## Quantum walks

### Element distinctness problem

### Triangle finding algorithm

# The needs of a general-purpose programming language

What are the differences between QC and CC?

Why do we need to code in the first place?

## A Pseudo-code

## Imperative Programming Language

### Omer

### Extending Omer’s work, Betelli’s imperative program

### Find a good recent one, because previous examples are out of date

## Functional Programming languages

### Peter Sellinger’s work

### Quipper