Quantum Computing languages landscape

7 min read · Sep 3, 2018

Quantum World Association Follow

Listen Share

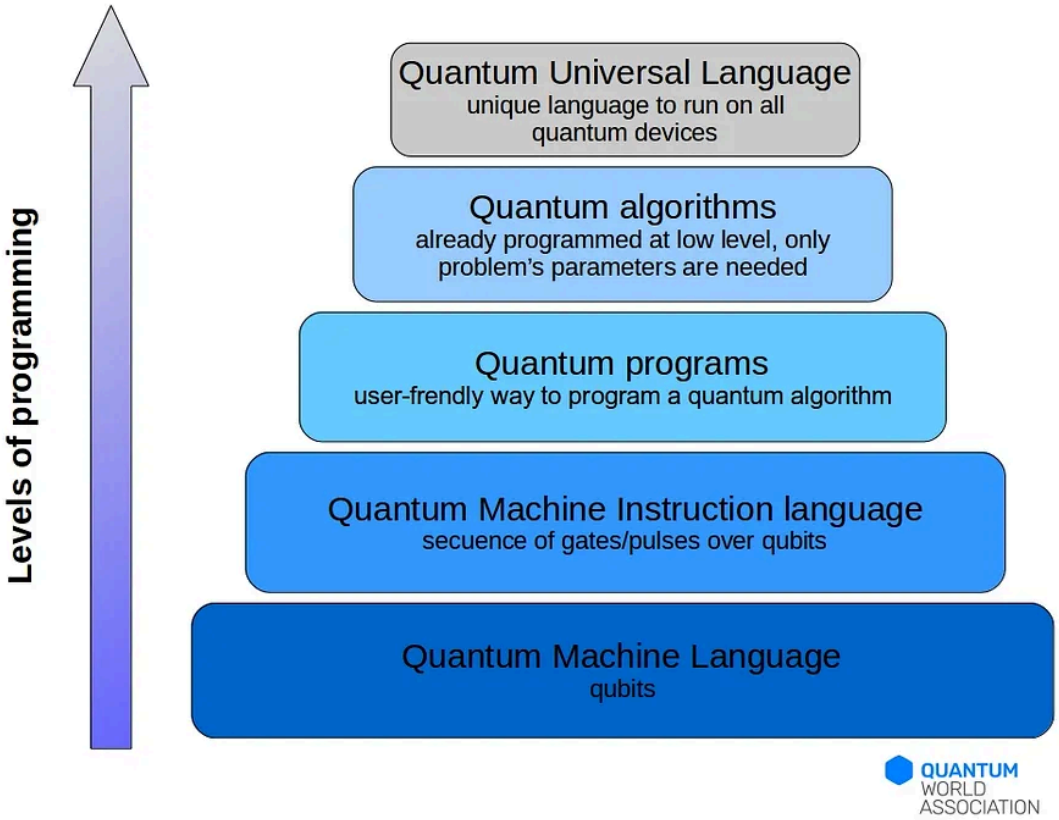
by Alba Cervera-Lierta, QWA team member and researcher at Quantic group.

We have spent the last months introducing the different quantum technologies: communication, computation and sensing. Once we closed these premier chapters, we began to focus on the applications of these technologies, beginning with finance in the last post. Nevertheless, let's take a break, take a breath and talk about a new landscape that is naturally emerging and growing with these new technologies: the quantum languages landscape.

Quantum Computing Languages landscape, by Patricia Segado.

As happens with classical computers, we need a language to send instructions and receive the outputs from our quantum computer — that is, a language to talk with them. There are different levels of programming: from assembly languages (also known as *quantum machine instruction languages*), which give the specific instructions to the computer, to higher level languages, where quantum algorithms are already programmed at low level and we only have to introduce some specific parameters of our problem.

A quantum computer is actually a hybrid machine consisting of a quantum device (hardware) and a classical computer, which sends the instructions to the hardware and receive and process the results (software). Quantum languages correspond with this classical software. Some of them are proper programming languages as are the classical ones. Others are in fact libraries programmed using some well-known language (python, C++, Matlab, ...) that help the user to program quantum algorithms. In this article, and its corresponding figures, we will refer as a "quantum language" both cases indistinctly.



Different levels of programming quantum algorithms.

Quantum languages have been around since before the emergence of real quantum devices. They have been used to simulate quantum algorithms on classical computers. With the construction of real quantum computers, each research group and company have developed their own language to use their machines. This has motivated the creation of even more languages that try to join some elements of the existing ones and whose aim is to be used in any backend: we call them *Universal Quantum Languages*.

The creation of these more general languages is possible because almost everybody is developing open-source software, that is, anybody can contribute to the improvement and extension of these languages by using platforms like GitHub.

To avoid possible confusions, let's point out the differences between "open-source" and "free" software. Both terminologies are used indistinctly to refer to software with fewer restrictions on how can be used than proprietary software. However, the philosophy of each movement is different. Taking the words of Richard Stallman, one of the founders of the Free Software Fundations:

"*The two terms describe almost the same category of software, but they stand for views based on fundamentally different values. Open source is a development methodology; free software is a social movement.*"

So, free-software allows the users "have the freedom to run, copy, distribute, study, change, and improve the software", not necessarily at zero cost, whereas open-source software is more focussed on the value of a collaborative and community-driven development model. The open-source software movement was born from the original free software community and, in general, a free software code is also open-source but not necessarily the opposite. A good short review about the history and differences between these two movements can be found in the article "The Difference Between Free and Open-Source Software" by Mark Drake.

The ideological discussion about "free" or "open-source" software is not the aim of this article. What is undoubted is the advantage that the open-source methodology introduces in the development and improvement of software codes. This advantage is even clearer if the community is trying to create a programming language whose applications are quite new. Quantum computation is a relatively new field that introduces a new paradigm in computation. There are totally different technologies

to run quantum algorithms and the ultimate programming language should satisfy the needs of all possible users.

It's not always easy to establish a parallelism between classical and quantum programming levels. We will try to classify some of the most well known quantum programming languages in different abstraction levels, especially the ones that are developed by private companies and start-ups. This will not be an exhaustive list. As with the quantum computing landscape introduced in our first quantum computation article, the list is growing day by day. A more complete classification that is updated periodically can be found on the Quantum Computing Reports webpage or in the Open-Source Quantum Software Projects list.

Some Quantum Programing Languages

We will focus on quantum languages used by companies that are developing quantum computers. Not all of them use this open-source strategy to create their codes, so the language used by some companies is not public, not even the name.

- *IBM:* they have developed the Quantum Information Software Kit (QISKit), which is a full-stack library to write, simulate and run quantum programs. Recently, they have divided it in four pieces: *Terra*, which allows to program at the level of quantum gates and pulses (quantum gates are implemented with sequences of pulses); *Aqua*, a higher level of programming to run algorithms used in quantum chemistry, optimization problems and A.I.; *Ignis*, to characterize errors and improve gate implementation; *Aer*, to study the limits of quantum computation using simulations in classical devices. QISKit translates quantum programs into a lower level language called *QASM*, which is its quantum instruction language.

- *Rigetti Computing:* similarly, they have created Forest, a developer environment to write and run quantum programs. Their quantum devices are programmed using a python library called *pyquil*, which translates programs written in terms of quantum gates into its lower level language called *quil*. They have also developed a library, *Grove*, which includes algorithms such as Variational Quantum Eigensolver, used in quantum chemistry, or QAOA, used for optimization problems.

- *Microsoft:* this company offers the Quantum Development Kit which comprises a quantum simulator, libraries to implement quantum algorithms and a full-stack

quantum programming language called *Q#*, which is available as a separately downloaded extension for Visual Studio. Microsoft is still developing their quantum computer with topological qubits, so they currently offer a quantum simulator to run the quantum programs.

- *D-Wave:* their software environment comprises qbsolv, a quantum language which helps users to map their QUBO problem (the type of problems to be solved with their quantum annealers) to the qubits connectivity of D-Wave devices and translates the program to its quantum instruction language. They also offer some higher level libraries to implement some optimization problems called *QSage* and *ToQ*. Los Alamos National Laboratory has developed a quantum macro-assembler language called QMASM, which fills the gap in D-Wave's software ecosystem.

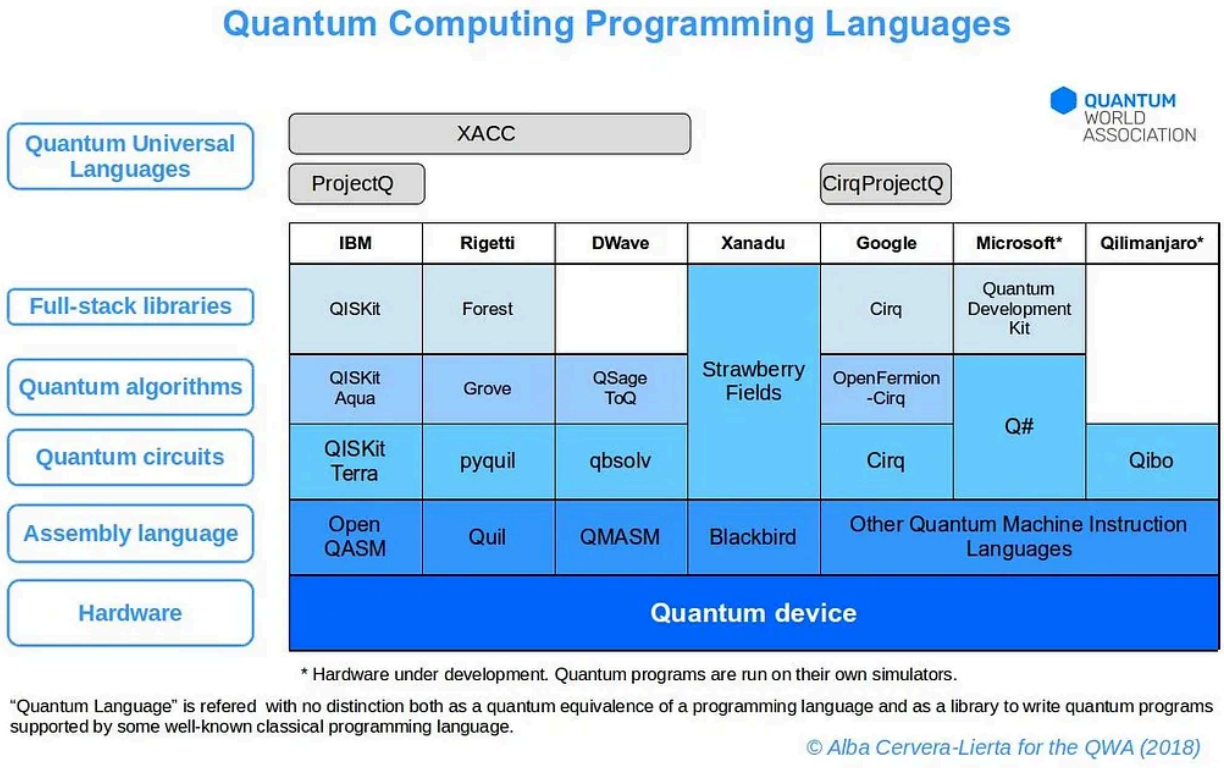
- *Google:* although is not a Google official product, its AI group has developed Cirq, a quantum language which consists of a python library for writing, manipulating and optimizing circuits and running them again quantum computers and simulators. It's currently been used by some companies and groups to run Google quantum devices. There is a library to run OpenFermion — an open-source library to obtain and manipulate representations of fermionic systems, very useful for quantum chemistry, for simulation on quantum computers — programmed for Cirq codes called OpenFermion-Cirq.

- *Xanadu:* they have developed Strawberry Fields, a full-stack library specifically targeted to continuous-variable quantum computation, which is the photon-based quantum computer they are designing and constructing. Quantum circuits are written using *Blackbird* quantum programming language, which is the assembly language for their quantum devices.

- *Qilimanjaro:* this start-up has launched the alpha version of Qibo, the language that will be used to program its quantum annealer. They have just started to construct the quantum device, so they offer a quantum simulator to run the programs. It's also the aim of this company to use this language as a quantum universal language to any other backend.

As you may notice, there are plenty of quantum computing languages. Each company is making a proposal to program their devices. Which of them will be better and be imposed over the rest? Who knows. If we look back, this is not a new discussion. We have a similar one with classical programing languages. In the end,

they suffer a kind of natural selection; only the most used survive. In the meantime, some people are trying to simplify researchers' work by developing languages which translate quantum programs into the above ones depending on which device would you like to use. One of this quantum universal language is ProjectQ, which can be used to program IBM devices and Google's with its extension CirqProjectQ. Other is XACC, which is a full-stack library to program IBM, Rigetti and D-Wave quantum computers.



Classification of some of the existing quantum computing languages according to the level of programming.

Quantum languages are still under development as are quantum devices. They evolve together with the hardware which they have to talk with. As you know, we have some prototypes of quantum computers, composed by few *noisy* qubits, but we aspire to construct *fault-tolerant* quantum computers, that is, quantum devices that can perform sophisticated algorithms with no errors. In the next Medium article, we will talk about this issue: near term noisy vs fault-tolerant quantum computation.

QWA is helping to bridge the gap between quantum and business. The go-to for providing suggestions, feedback and questions is our email address info@quantumwa.org.

Quantum Computing | Entrepreneurship | Computer Science