

Learning Quantum Computing: An Interaction Protocol for Quantum Computing Interfaces

Rosa M. Gil Iranzo
Computing Science and Industrial Engineering
Universitat de Lleida
Lleida, Spain
rosamaria.gil@udl.cat

Carina González González
Computer and Systems Engineering Department
University of La Laguna
Tenerife, Spain
carina.gonzalez@ull.edu.es

Mercè Teixidó Cairol
Computing Science and Industrial Engineering
Universitat de Lleida
Lleida, Spain
merce.teixido@udl.cat

Roberto García
Computing Science and Industrial Engineering
Universitat de Lleida
Lleida, Spain
roberto.garcia@udl.cat

ABSTRACT

Though research on quantum computing continues at a good pace, learning quantum computing requires a deep and clear level of knowledge of a range of domains that diverge from classical computer science. There are not quantum computing interfaces that facilitate learning this new paradigm, only those to illustrate some quantum computations with some accompanying documentation. Moreover, in this documentation, science is mixed with technology without a clear distinction between the underlying mathematics and physics concepts. We consider that this must be addressed in the first place in order to create proper learning environments that go beyond performing calculations without facilitating the understanding of the core of quantum computing concepts and their implications. This is a new research line, dealing with the design of quantum computing interfaces integrating science and technology.

CCS CONCEPTS

• Human-centered computing → Human computer interaction (HCI) → Interactive systems and tools → User interface toolkits

KEYWORDS

Quantum computing, Learning, Education, Interfaces, Interaction

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.

Interacción '21, September 22–24, 2021, Málaga, Spain
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-7597-9/21/09...\$15.00
<https://doi.org/10.1145/3471391.3471424>

ACM Reference format:

Rosa M. Gil, Mercè Teixidó, Carina Gonzalez and Robeto Garcia. 2021. Learning Quantum Computing: An Interaction Protocol for Quantum Computing Interfaces. In XXI International Conference on Human Computer Interaction (Interacción '21), September 22–24, 2021, Málaga, Spain, 5 pages. <https://doi.org/10.1145/3471391.3471424>

1 Introduction

As Gartner Research points out in a recent report [1], the next decade will see a lot of disruptive innovation coming from quantum computing. We are already in the midst of a second quantum revolution: the transition from quantum theory to its application through quantum engineering [2].

In fact, Quantum computing is a reality and can be used today. Some applications are already available and can be even accessed on-line as shown later. However, new interaction and learning means must be developed to smooth the transition to this new computing paradigm, especially for those programmers that coming from digital computing will be contributing to the future applications of quantum computing.

This article is structured as follows: first, to have a broader perspective, a brief review of programming paradigms is carried out that puts in context the way in which they are used to model and organize the world. Then, some relevant attempts to develop quantum computing programming languages are discussed together with the challenges that they have to face. Finally, there is a first proposal of an interaction protocol for quantum computing interfaces that tries to overcome the identified learning issues.

2 Programming paradigms

Classically, the following main programming paradigms have been identified [3]:

- Imperative
- Logical
- Object-Oriented
- Functional

In the imperative paradigm, commands and procedures are the basis of such languages as Basic, Fortran or Pascal. It was mostly used in the '70s and it is still present in the banking context nowadays.

In the logical paradigm, we define the rules for the world, but not how to process them, thus following a declarative instead of imperative approach. A famous scene in the 1975 Monty's Python film, summarized in Fig. 1, the "Monty Python and the Holy Grail" shows the strange outcome of using logical reasoning "funnily", or not so funny for the witch. It is based on axioms, inference rules, and queries, which operate recursively.

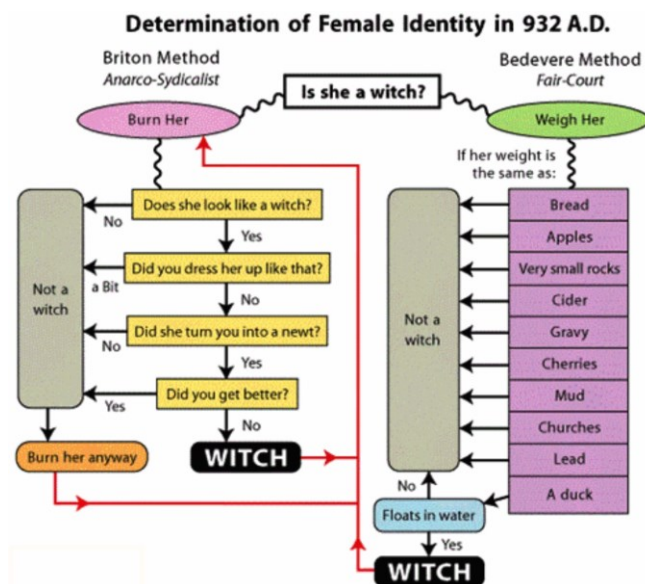


Figure 1: Graphical representation of the logical reasoning used in the 'Monty Python and the Holy Grail' film.
Source: <http://joyreactor.com/post/910757>

Object-Oriented Programming puts classes and inheritance at the center of this programming paradigm, while methods and attributes characterize classes and objects.

On the other hand, functional programming focuses on functions, their inputs and outputs. There are fully functional programming languages like Haskell, but other programming languages based on different paradigms can also include functional operators. For instance, Python provides operators for functional programming using a lambda function, a single-line function declared with no name, which can have any number of arguments, but it can only have one expression.

Functional programming is the paradigm that appears more prepared to be used in the context of quantum computing. It

offers many advantages, for instance, the concept of monads supported by functional programming languages like Haskell [4]. They can be used as metaphors for physical states and measurements. Monads make it possible to wrap and unwrap values in the context of the monad while applying operations to values, which provides a way to model what it is going on in the quantum world when we make a measurement of a quantum state. In fact, monads have been already adopted by non-functional programming language, like Java, which from its 8th version includes the implementation of two monads: Stream and Optional.

3 Physical Interpretations

The following quantum programming languages include the Copenhagen interpretation¹, said that a quantum system remained in this superposition until it interacted with, or was observed by, the external world, at which time the superposition collapses into one or another of the possible definite states. This is the standard accepted theory. However, it is not the only interpretation in physics, many-worlds interpretation and consistent histories are also proposed. There are more theories to be proven that consider consciousness too, however they are not validated yet as Orch OR [5].

Quantum computers rely on qubits meanwhile traditional computers use binary values. Qubits can be 0 or 1, usually we use a description as a vector, however the numbers we can find belong to complex numbers (they are neither binary nor real numbers), the square of the module is the percentage associated with the observation.

Quantum entanglement is a property of the quantum mechanical state of a system containing two or more objects, where the objects that make up the system are linked in such a way that the quantum state of them cannot be adequately described without full mention of the others, even if the individual objects are spatially separated. Thus, the digital logical gates we learnt in engineering degrees are not enough to be able to create quantum circuits.

4 Quantum projects

Some projects and languages for quantum computing have appeared, we list in the following subsections the most notable ones.

4.1 Quipper

Computations in Quipper take the form of functions. Quipper has proven effective and easy to use, and opens the door towards using formal methods to analyse quantum algorithms. The following example shows how we can write a simple quantum function in Quipper [6].

¹ <https://plato.stanford.edu/entries/qm-copenhagen/>

```
plus_minus :: Bool → Circ Qubit
plus_minus b = do
  q ← qinit b
  r ← hadamard q
  return r
```

4.2 Scaffold

Scaffold is a programming language for expressing quantum algorithms. It was created in 2012 in Princeton. Scalability is its most laureated feature [7].

4.3 Imperative Quantum Programming

Under this title, “Imperative Quantum Programming” [8], we can find: Quantum Pseudocode, Quantum Computing Language, Q Language and qGCL. There are also the functional versions QPL and cQPL plus, of course, Quantum Lambda Calculus.

4.4 IBM Qiskit

Qiskit is an open source SDK for working with quantum computers at the level of pulses, circuits and application modules using a graphical interface [9].

5 Challenges in Learning Quantum Computing

In this section we identify the main challenges that we face when we want to learn quantum computing:

- **Insights in the logic gates.** Quantum mechanics force us to face new logic gates as: Hadamard gate, Pauli-X gate, Pauli-Y gate, Pauli-Z gate, Rotation gates or CNOT (A XOR B), plus the digital logic gates that every student has learnt in an engineering degree.
- **Few qubits in the visualization model interface.** The most advanced graphical interfaces like IBM Composer allow composing qubits in a visual way. However, only a few qubits are supported at the moment. It generates a problem to create complex circuits with many qubits. For instance, the logic gate called the Toffoli gate. For those of you who are familiar with Boolean logic gates, it is basically an AND gate. Visually, we build a half-adder, Fig. 2, that in quantum computing becomes Fig. 3. It consumes a lot of visual space as we can see.
- **Documentation mixes disciplines.** The different disciplines, such as physics, mathematics and computer science, are presented in a mixed way in the documentation. Thus, the cognitive load is very high because all the concepts should be processed at the same time.

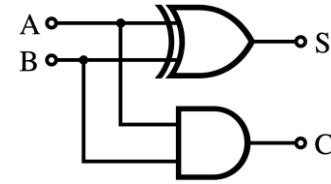


Figure 2: Traditional half adder

- **Lack of background knowledge.** The most fundamental reason to propose a new protocol and scheme to learn quantum computing is because we must face the counterintuitive principles of quantum physics, which are:
 1. A physical system in a definite state can still behave randomly.
 2. Two systems that are too far apart to influence each other can nevertheless behave in ways that, though individually random, are somehow strongly correlated.

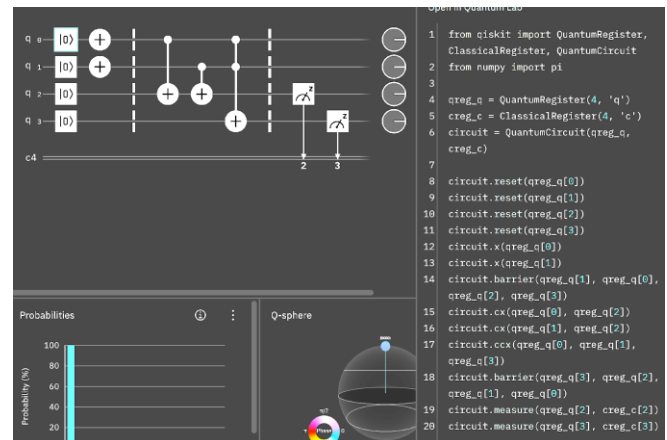


Figure 3: Half adder using Qiskit in IBM composer

6 Proposal

The proposed layered structure to introduce quantum computing concepts can be used at different learning levels, from high school to university and the research level, performing activities for the appropriate age.

6.1 Interaction Protocol for Quantum Computer Interfaces

The following layers are proposed as the foundations of a layered approach to leaning quantum computing. These layers help keeping the different disciplines separate and the mix of concepts that causes most of the cognitive overload:

- **Physical Layer:** this layer is about the underlying technologies used to build the quantum computer [10]. The main are **Trapped Ion Qubits** and **Superconducting Qubits**. These are not the only ones; however, it is argued that the others are not purely quantum. What these technologies have in common is that there are no electrical signals that are 0 or 1 as inputs and outputs. We have **quantum** states as **inputs** and measurement results as **outputs**.
- **Abstract Layer:** this layer deals with the logic, protocols and algorithms that constitute the applications:
 - **Logical Sublayer:** the quantum computing language program with a programming interface (e.g. Composer by IBM).
 - **Quantum Protocols and Quantum Algorithms:** this layers with the corresponding protocols and algorithms, among others Deutsch-Jozsa, Bernstein-Vazirani & Simon's Algorithm, Quantum Fourier Transform, Quantum Phase Estimation, Shor's Algorithm, Grover's Algorithm, Quantum Counting, Quantum Teleportation, Superdense Coding, or Quantum Key Distribution.
 - **Apps and Applications level:** this layer deals with concrete applications of quantum computing like solving linear systems of equations, simulating molecules, solving combinatorial optimization problems, solving satisfiability problems or hybrid quantum-classical neural networks.
- **Mathematical layer:** linear algebra used like vectors, matrices or tensors, with the corresponding mathematical operations.
- **Theoretical Physical layer:** linear algebra is a useful tool for calculations, however, there is more information about the physical process that is going on that is not captured by the Mathematical layer. For instance, the basic idea of matrix mechanics is to replace the wave function with a vector. Moreover, integrals are replaced with dot products (overlap between any two wave functions) and operators are represented by matrices and average values are mapped into numbers.

Many times, the **Mathematical layer** is confused with the **Theoretical Physical layer** (see Table 1).

Each layer has its own research (i.e. in the physical layer, we can find that Microwave Pulses or Quantum Circuits are used for investigating Quantum Hardware).

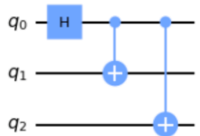
For children, functions can be learned with Scratch [11], even recursive functions [12], and we can work with physical or virtual boxes to work to enhance the concept of measurement. Which is a critical point for quantum computing programming to be understood.

7 Conclusions and future work

Quantum computing is beginning to be a relevant and impactful topic. An effort has to be made by UX professionals to aid people to understand what, how and why they code in the way it is necessary to program this kind of systems.

The focus should be placed on trying to lower the cognitive load. To address this, we have used the divide and conquer strategy and cut and organised contents as well as examples of how it can be achieved to ensure that there is an adequate interaction during the learning process.

Table 1: Three-qubit GHZ state, comparison among 3 layers.

Theoretical Physical layer Quantum state three-qubit. $ \psi\rangle = (000\rangle + 111\rangle)/\sqrt{2}$	
Mathematical layer Quantum mechanics only allows linear operations to be applied to quantum states. $U \begin{pmatrix} \alpha_{000} \\ \vdots \\ \alpha_{111} \end{pmatrix} = \begin{pmatrix} \beta_{000} \\ \vdots \\ \beta_{111} \end{pmatrix}$ The states $ 000\rangle \dots 111\rangle$ form an orthonormal basis of an N-dimensional Hilbert space (i.e., an N-dimensional vector space equipped with an inner product) of dimension N, and a quantum state $ \psi\rangle$ is a vector in this space where $\beta_{000} = \frac{1}{\sqrt{2}}$ and $\beta_{111} = \frac{1}{\sqrt{2}}$	
Abstract Layer <pre># Add a H gate on qubit 0, putting this qubit in superposition. circ.h(0) # Add a CX (CNOT) gate on control qubit 0 and target qubit 1, putting the qubits in a Bell state. circ.cx(0, 1) # Add a CX (CNOT) gate on control qubit 0 and target qubit 2, putting the qubits in a GHZ state. circ.cx(0, 2)</pre> 	

ACKNOWLEDGMENTS

Alfabetización digital y STEAM en edades tempranas: propuesta co-educativa inclusiva. Ref. 2020EDU08. 2021-2023. 57.000€. Fundación Caja Canarias y Fundación La Caixa.

REFERENCES

- [1] Gao, A. and Dekate, C. 2021. Predicts 2021: Disruptive Potential During the Next Decade of Quantum Computing. *Gartner Research*.
- [2] Ying, M., Feng, Y., Duan, R., Li, Y., and Yu, N., 2012. Quantum programming: From theories to implementations. *Chinese science bulletin* 57(16), 1903-1909. DOI: <http://dx.doi.org/10.1007/s11434-012-5147-6>
- [3] Gabbrielli, M., and Martini, S., 2010. Programming languages: principles and paradigms. *Series: Undergraduate Topics in Computer Science*. 1st Edition.
- [4] O'Sullivan, B., Goerzen, J. and Stewart, D.B., 2008. Real world haskell: Code you can believe in. O'Reilly Media, Inc.
- [5] Stuart Hameroff, 2021. 'Orch OR' is the most complete, and most easily falsifiable theory of consciousness. *Cognitive Neuroscience* 12(2), 74-76. DOI: 10.1080/17588928.2020.1839037
- [6] Green, A. S., Lumsdaine, P. L., Ross, N. J., Selinger, P., and Valiron, B. 2013. Quipper: a scalable quantum programming language. In *Proceedings of the 34th ACM SIGPLAN conference on Programming language design and implementation*, 333-342.
- [7] Abhari, A. J., Faruque, A., Dousti, M. J., Svec, L., Catu, O., Chakrabati, A., Chiang, C.-F., Vanderwilt, S., Black, J. and Chong, F., 2012. Scaffold: Quantum programming language. DTIC Document, Tech. Rep.
- [8] Quantiki Quantum Information Portal and Wiki, <https://www.quantiki.org/wiki/quantum-programming-language>, Accessed: April 30, 2021.
- [9] Qiskit documentation, <https://qiskit.org/documentation/>, Accessed: April 30, 2021.
- [10] National Academies of Sciences, Engineering, and Medicine 2018. *Quantum Computing: Progress and Prospects*. Washington, DC: The National Academies Press. <https://doi.org/10.17226/25196>
- [11] Scratch, <http://opensask.ca/VisualProgrammingEnv/MakingOwnBlocks.html>, Accessed: April 30, 2021.
- [12] Recursive Functions - Scratch Tutorial - Finding Factorial with Scratch, https://www.youtube.com/watch?v=T7kSI8GEeME&ab_channel=LearningWithJeff, Accessed: April 30, 2021.