Qurry: A quantum programming language

Lucas Saldyt, *Arizona State University*

***Abstract*—Quantum programming languages are largely still in early development. However, many lack true *abstraction* and are simply proxies to circuit languages. There is reason for this, primarily because the desired semantics of a quantum programming language are not yet completely crystalized. This paper focuses on the creation of "lightweight abstractions," which allow human-level understanding without sacrificing low level control. Additionally, this paper describes a testbed, which is meant to catalyze the development of quantum programming languages.**

***Index Terms*—Quantum Computing, Programming Languages**

I. Introduction

QUANTUM programming …

Essentially, innovation in near-term quantum programming requires the use of lightweight abstractions. Because the field is (comparatively) nascent, some transparency is crucial. However, there are already plenty of abstractions in quantum programming. Theorists will be familiar with a linear algebra matrix-operator model, which is an abstract scaffold over true hardware implementations, such as [Citation: superconducting implementations using microwave pulses etc]. Given these facts, current term quantum programming languages are analogous to classical assembly languages, and the next stage of development is to create the equivalent of C for quantum programming. When C was invented for classical programming, it revolutionized [Summarize the effects]. Even in modern times, C++ is used for a [large part] of software development. The term "lightweight abstraction" comes from the C++ community, in particular [that one paper by stroustrup that I read like 1/8 of].

Additionally, it is necessary to rapidly test new ideas in quantum programming from the bottom up and simply collect data on them, as opposed to architecting a top-down "perfect" language. As can be seen in the natural language attempt to create Esperanto as a universal language, this is unlikely to work. In general, evolution will create a much more robust system. Thus, Qurry offers a framework in which programmers can easily create new syntax features. This framework works similarly to macros in lisp, but has a simplified interface as well.

Since quantum computers are simply special probabilistic computers, Qurry also attempts to create a statistical library for high-level modeling. This is particularly useful in the same way that a classical probabilistic programming language is, namely for modeling anything statistical, and especially for bayesian machine learning. For instance, the R. Tucci and H. group have shown uses for this through their software, Bayesforge.

TODO

II. Conclusion

The conclusion goes here.

Appendix A

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgment

The author would like to thank Will Zeng of Rigetti computing, who sponsered this project through the unitary fund [LINK].

References

[1] H. Kopka and P. W. Daly, *A Guide to $\LaTeX$*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

Lucas Saldyt is currently a student researcher at Arizona State University, and has previously worked for Sandia National Laboratories and Los Alamos National Laboratories as a student intern in the quantum computing department.