Curry: A quantum programming language

Lucas Saldyt, *Arizona State University*

***Abstract*—Programming quantum computers currently requires specialized knowledge. This project aims to prototype a higher-level, quantum-hybrid programming language for expressing probabilistic computations easily. The MIT cognitive science community has produced many developments using probabilistic programming languages [cite]. Classical probabilistic programming builds models from building blocks called exchangeable random primitives. Similarly, quantum programming uses qubits, which are have complex probability amplitudes and may similarly be building blocks for models. This paper presents the prototype for a quantum programming language which offers novel abstractions not yet available in existing quantum programming languages.**

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

I. Introduction

QUANTUM computing is still in a nascent form, especially when it comes to programming interfaces. Many acclaimed quantum programming languages are simply circuit description languages that add superfluous syntax to have the look and feel of a traditional programming language. These interfaces generally lack abstraction, which is an absolutely essential feature of any programming language intended for humans. For instance, the world of classical programming languages was revolutionized with the invention of the C programming language, which is now ubiquitous in today's software either in either its original or derivative forms [Cite].

Classical probabilistic programming languages are a recent innovation from the MIT cognitive science community. Essentially, they create a way for non-expert programmers to access the power of Bayesian inference. Users can create simple probabilistic models in standard code, and then run them through an expert-created inference backend. Famously, this has resulted in dramatically reduced code complexity, with a famous case where a 50-line probabilistic program could compete with traditional approaches to face recognition [TODO: Cite]. Highly abstract and specialized languages such as this will be useful both in industry and in scientific research.

Both quantum computing and classical probabilistic programming have variables which are probability distributions. In quantum computing, measuring a single qubit results in 0 or 1, which represents a bernoulli trial, even though the qubits pre-measurement state is richer (because its true representation is a complex probability amplitude). If this is repeated multiple times, it creates a binomial distribution. Similarly, measuring multiple (n) qubits gives a bitstring $(0, 1)^n$. If this is repeated multiple times, it creates a multinomial distribution.

Quantum states are richer than standard probability distributions. The simplest case is a Bell state, where, for instance, one may measure 00 or 11 with equal probability…[TODO: Extend]

mds
August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

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.

Manuscript received April 19, 2005; revised August 26, 2015.