ZX.jl: ZX-calculus for Julia

Chen Zhao

University of Chinese Academy of Sciences

Abstract

ZX-calculus is a graphical language which can characterize quantum circuits. It is a powerful tool which is usually used for quantum circuits simplification. I'm going to develop a pure Julia package `ZX.jl` which implement quantum circuits simplification algorithms based on ZX-calculus. In addition, it will provide interfaces to import and export quantum circuits to the form of YaoIR, an intermediate representation for quantum programs in the `Yao.jl`. So that one can get quantum circuits with higher performance automatically when designing quantum programs with `Yao.jl`.

1 Why this project?

Since Shor's factoring quantum algorithm was found, people believe that quantum computers are more powerful than classical computers and quantum algorithms are more efficient than classical algorithms on some computational tasks. The basic model for describing quantum algorithms is the quantum circuit which can be run on quantum hardwares. The quantum compiler compiles quantum programs to quantum circuits.

On the one hand, as quantum hardware developing is so difficult, only limited number of qubits will be available and only small size of quantum circuits can be run on quantum devices in the near-term future. On the other hand, simulating quantum computer on classical computer needs exponential resources. Simpler circuits will help us developing more efficient quantum simulation algorithms. If we have good quantum circuits simplification algorithms in the quantum compiler, we can get faster quantum algorithms.

ZX-calculus is a powerful tool for quantum circuits simplification. And lots of algorithms are developed based on it. There exists a Python implementation `PyZX`. With `PyZX`, one can simplify quantum circuits with ZX-calculus. Several input and output forms of quantum circuits are provided and visualization of quantum circuits are available. But there is no Julia implementation of these algorithms. In Julia, `Yao.jl` is a high-performance and extensive package for quantum software framework. However, only simple circuits simplification algorithms are available in `Yao.jl`. In order to develop better circuits simplification algorithms for the quantum compiler in `Yao.jl`, a high-performance pure Julia implementation of ZX-calculus is necessary.

2 Technical Details

`ZX.jl` will be a package for ZX-calculus which any Julia user can install. With `ZX.jl`, one will be able to develop quantum circuits simplification algorithms, which are very important for current quantum algorithm designing, in pure Julia.

The implementation of ZX-calculus consist of three levels.

The first level is the backend data structures for representing ZX-diagrams, the basic objects in ZX-calculus. In general, ZX-diagrams are multigraphs with extra information of their edges and vertices, (for example, phases of vertices). Fortunately, there has been `LightGraphs.jl` for simple graphs already. And I have developed `Multigraphs.jl` as a multigraph extension for `LightGraphs.jl`. Hence, problems of representing ZX-diagrams can be solve by using these existing packages.

The second level is the basic rules in ZX-calculus. These basic rules define equivalence relations between ZX-diagrams and can be found in [2]. In ZX-calculus, All simplification algorithms are based on these basic rules. These rules can be implemented by operating the backend multigraphs and extra information of ZX-diagrams.

The third level is the quantum circuits simplification algorithms. These algorithms can be implemented by using basic rules. For example, the state-of-art algorithm for reducing T-count. All these algorithms can be found on ZX-calculus.

As an application, I'm going to develop interfaces for importing and exporting quantum circuits in the form of YaoIR. So that, `Yao.jl` can simplify quantum circuits defined by users when compiling quantum algorithms.

For data visualization, I will provide ways to plot ZX-diagrams and quantum circuits. This will based on the graph visualization tool `GraphPlot.jl`.

3 Schedule of Deliverables

Before Community Bonding Period

April 1, 2020 - April 27, 2020

Read articles on ZX-calculus. Be familiar with the developing tools, working flow, test systems and documentation systems.

Community Bonding Period

April 28, 2020 - May 7, 2020

Discuss about the implementation details with the mentor.

May 8, 2020 - May 17, 2020

Implement data structures for representing ZX-diagrams.

Phase 1

May 18, 2020 - May 27, 2020

Develop algorithms for matching and applying basic rules (including rule $f, h, i_1, i_2, \pi, b, c$) in [2].

May 28, 2020 - June 14, 2020

Implement ZX-diagrams simplification algorithms with basic rules.

Implement circuits to circuits simplification algorithms including [1, 2].

June 15, 2020 - June 19, 2020

Submit Phase 1 evaluations.

Phase 2

June 20, 2020 - July 30, 2020

Add visualization support for ZX-diagram and quantum circuits.

July 1, 2020 - July 12, 2020

Develop transformation between YaoIR and ZX-diagrams.

July 13, 2020 - July 17, 2020

Submit Phase 2 evaluations.

Final Phase

July 18, 2020 - August 1, 2020

Integrate circuits simplification algorithms to the compiler in `Yao.jl`.

August 1, 2020 - August 10, 2020

Make full documentation, refine tests, and show some demonstration for `ZX.jl`.

Final Week

August 11, 2020 - August 17, 2020

Submit final evaluations.

4 About me

I'm Chen Zhao, currently a PhD student majoring applied mathematics in University of Chinese Academy of Sciences. I'm researching on quantum machine learning and quantum algorithms.

- E-mail: `zhaochen17@mails.ucas.ac.cn`
- GitHub: ChenZhao44

Development Experience

- `QDNN.jl`: an implementation of the model QDNN (deep neural networks with quantum layers).
- `Multigraphs.jl`: a multigraph extension for `LightGraphs.jl`

References

[1] Duncan, R., Kissinger, A., Pedrix, S., and Van De Wetering, J. Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus. *arXiv preprint arXiv:1902.03178* (2019).

[2] Kissinger, A., and van de Wetering, J. Reducing T-count with the ZX-calculus. *arXiv preprint arXiv:1903.10477* (2019).