SZX-calculus: Scalable Graphical Quantum Reasoning

Titouan Carette
CNRS, LORIA, Inria Mocqua, Université de Lorraine, F 54000 Nancy, France
titouan.carette@loria.fr

Dominic Horsman
LIG, Université Grenoble Alpes, France
dominic.horsman@univ-grenoble-alpes.fr

Simon Perdrix
CNRS, LORIA, Inria Mocqua, Université de Lorraine, F 54000 Nancy, France
simon.perdrix@loria.fr

Accepted at MFCS2019: https://arxiv.org/pdf/1905.00041.pdf

We introduce the Scalable ZX-calculus (SZX-calculus for short), a formal and compact graphical language for the design and verification of quantum computations. The SZX-calculus is an extension of the ZX-calculus, a powerful framework that captures graphically the fundamental properties of quantum mechanics through its complete set of rewrite rules. The ZX-calculus is, however, a low level language, with each wire representing a single qubit. This limits its ability to handle large and elaborate quantum evolutions. We extend the ZX-calculus to registers of qubits and allow compact representation of sub-diagrams via binary matrices. We show soundness and completeness of the SZX-calculus and provide two examples of applications, for graph states and error correcting codes.

The ZX-calculus is an intuitive and powerful graphical language for quantum computing, introduced by Coecke and Duncan [5]. Quantum processes can be represented by ZX-diagrams, which can be seen intuitively as a generalisation of quantum circuits. The language is also equipped with a set of rewrite rules which preserves the represented quantum evolution. Unlike quantum circuits, the ZX-calculus has been proved to be complete for various universal fragments of pure quantum mechanics [14, 12, 15, 21], and also mixed states quantum mechanics [3]. Completeness means that any equality can be derived in this language: if two diagrams represent the same quantum process then they can be transformed one into the other using the rewriting rules of the language. Completeness opens avenues for various applications of the ZX-calculus in quantum information processing, including circuit optimisation [8, 17] – which outperforms all other technics for T-count reductions [19] – error correcting codes [9, 11, 4], lattice surgery [1], measurement-based quantum computing [10, 7, 18] *etc*. Automated tools for quantum reasoning, e.g. Quantomatic [20] and PyZX [16], are also based on the ZX-calculus. The ZX-calculus is also used as intermediate representation in a commercial quantum compiler [6].

The cornerstone of the ZX-calculus is that fundamental properties of quantum mechanics can be captured graphically. The language remains, however, relatively low level: each wire represents a single qubit, a feature that limits the design of larger-scale and more complex quantum procedures. We address in this paper the problem of scalability of the ZX-calculus. In [4], the authors – including one of the present paper – demonstrated that the ZX-calculus can be used in practice to design and verify quantum error correcting codes. They introduced various shortcuts to deal with the scalability of the language: mainly the use of thick wires to represent registers of qubits and matrices to represent sub-diagrams,

Submitted to:
QPL 2020

and hence reason about families of diagrams in a compact way. However, the approach lacked a general theory and fundamental properties like soundness and completeness.

Contributions. We introduce the Scalable ZX-calculus, SZX calculus for short, to provide theoretical foundations to this approach. We extend the ZX-calculus to deal with registers of qubits by introducing some new generators and rewrite rules. We show soundness – i.e. the new generators can be used in a consistent way – as well as completeness of the SZX-calculus. A simple but key ingredient is the introduction of two generators, not present in [4], for dividing and gathering registers of qubits. A wire representing a register of $(n+m)$-qubits can be divided into two wires representing respectively n and m qubits. Similarly two registers can be gathered into a single larger one. We also extend the generators of the ZX-calculus so that they can act not only on a single qubit but on a register of qubits. The SZX-calculus is then constructed as a combination of the ZX-calculus and the sub-language made of the divider and the gatherer, by adding the necessary rewrite rules describing how these two sub-languages interact. We show that the SZX-calculus is universal, sound, and complete, providing an intuitive and formal language to represent quantum operations on an arbitrarily large finite number of qubits. The use of the divider and the gatherer allows one to derive inductive (graphical) proofs.

Furthermore, the SZX-calculus provides the fundamental structures – namely the (co)comutative Hopf algebras – to develop a graphical theory of binary matrices, following work on graphical linear algebra [2]. As a consequence, we introduce an additional generator parametrized by a binary matrix together with four simple rewrite rules. Note that, while matrices were also used in [4], we introduce here a more elementary generator acting on a single register (1 input/1 output) rather than two registers (2 inputs/2 outputs). We prove completeness of the SZX-calculus augmented with these matrices. The use of matrices allows a compact representation where subdiagrams can be replaced by matrices. Moreover, basic matrix arithmetic can be done graphically. It makes the SZX-calculus with matrices a powerful tool for formal and compact quantum reasoning.

We then show the SZX-calculus in action. The main application of the SZX-calculus we consider in this paper is the graph state formalism [13]. We show how graph states can be represented using SZX-diagrams and how some fundamental properties like fixpoint properties, local complementation, and pivoting can be derived in the calculus. We also consider error correcting code examples in order to show that the techniques for the design and verification of codes developed in [4] can be performed smoothly in the SZX-calculus.

References

[1] Niel de Beaudrap & Dominic Horsman (2017): *The ZX calculus is a language for surface code lattice surgery*. https://quantum-journal.org/papers/q-2020-01-09-218/.

[2] Filippo Bonchi, Paweł Sobociński & Fabio Zanasi (2017): *Interacting Hopf algebras*. Journal of Pure and Applied Algebra 221(1), pp. 144–184.

[3] Titouan Carette, Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2019): *Completeness of Graphical Languages for Mixed States Quantum Mechanics*. In: International Colloquium on Automata, Languages, and Programming (ICALP'19).

[4] Nicholas Chancellor, Aleks Kissinger, Joschka Roffe, Stefan Zohren & Dominic Horsman (2016): *Graphical structures for design and verification of quantum error correction*. arXiv preprint arXiv:1611.08012.

[5] Bob Coecke & Ross Duncan (2011): *Interacting quantum observables: categorical algebra and diagrammatics*. New Journal of Physics 13(4), p. 043016.

[6] Alexander Cowtan, Silas Dilkes, Ross Duncan, Alexandre Krajenbrink, Will Simmons & Seyon Sivarajah (2019): *On the qubit routing problem*. arXiv preprint arXiv:1902.08091.

[7] Ross Duncan (2012): *A graphical approach to measurement-based quantum computing*. arXiv preprint arXiv:1203.6242.

[8] Ross Duncan, Aleks Kissinger, Simon Pedrix & John van de Wetering (2019): *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*. arXiv preprint arXiv:1902.03178.

[9] Ross Duncan & Maxime Lucas (2014): *Verifying the Steane code with Quantomatic*. Electronic Proceedings in Theoretical Computer Science (171), pp. 33–49. ArXiv preprint arXiv:1902.03178.

[10] Ross Duncan & Simon Perdrix (2010): *Rewriting Measurement-Based Quantum Computations with Generalised Flow*. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide & Paul G. Spirakis, editors: Automata, Languages and Programming, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 285–296.

[11] Craig Gidney & Austin G Fowler (2018): *Efficient magic state factories with a catalyzed —CCZ¿ to 2—T¿ transformation*. arXiv preprint arXiv:1812.01238.

[12] Amar Hadzihasanovic, Kang Feng Ng & Quanlong Wang (2018): *Two Complete Axiomatisations of Pure-state Qubit Quantum Computing*. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '18, ACM, New York, NY, USA, pp. 502–511, doi:10.1145/3209108.3209128. Available at `http://doi.acm.org/10.1145/3209108.3209128`.

[13] Marc Hein, Wolfgang Dür, Jens Eisert, Robert Raussendorf, M Nest & H-J Briegel (2006): *Entanglement in graph states and its applications*. Proceedings of the International School of Physics "Enrico Fermi" on "Quantum Computers, Algorithms and Chaos", arXiv:quantph/0602096;.

[14] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2018): *A complete axiomatisation of the ZX-calculus for Clifford+ T quantum mechanics*. In: Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), ACM, pp. 559–568.

[15] Emmanuel Jeandel, Simon Perdrix & Renaud Vilmart (2019): *A Generic Normal Form for ZX-Diagrams and Application to the Rational Angle Completeness*. In: Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS).

[16] A. Kissinger & John van de Wetering (2018): *PyZX*. Available at `https://github.com/Quantomatic/pyzx`.

[17] Aleks Kissinger & Arianne Meijer-van de Griend (2019): *CNOT circuit extraction for topologically-constrained quantum memories*. arXiv preprint arXiv:1904.00633.

[18] Aleks Kissinger & John van de Wetering (2017): *Universal MBQC with generalised parity-phase interactions and Pauli measurements*. arXiv:1704.06504.

[19] Aleks Kissinger & John van de Wetering (2019): *Reducing T-count with the ZX-calculus*. arXiv preprint arXiv:1903.10477.

[20] Aleks Kissinger & Vladimir Zamdzhiev (2015): *Quantomatic: A proof assistant for diagrammatic reasoning*. In: International Conference on Automated Deduction, Springer, pp. 326–336.

[21] Renaud Vilmart (2019): *A Near-Optimal Axiomatisation of ZX-Calculus for Pure Qubit Quantum Mechanics*. In: Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS). Available at `https://arxiv.org/abs/1812.09114`.