Research Statement

Ross Duncan

RESEARCH THEMES: Quantum computing; quantum foundations; theory of computation; verification and correctness; categorical logic and algebra; graph rewriting and graphical methods; visual computer languages.

Overview

The vast majority of current research in quantum computation is extremely low-level, in the sense of being intimately tied to particular implementation or architectural choices, or based on the meagre abstraction from a concrete physical system to the qubit. We rarely view our classical computers as arrays of naked bits, and in the quantum realm the need for a structured, high-level perspective is even more pressing. From this point of view, the usual Hilbert space formulation of quantum mechanics is akin to programming in raw binary code. However, reformulating quantum theory in the language of *category theory* offers a high-level perspective which exposes enough structure to work with, without burying us beneath a mass of irrelevant detail.

In classical computer science, category theory is widely used to give semantics to programs[1]. Such mathematical understanding is a necessary basis for strongly-typed programming languages and other tools to deliver reliable software. *Categorical quantum mechanics*[2] seeks similar categorical foundations for quantum theory, and in particular quantum computation. Remarkably, a great deal of quantum mechanics can be formalised in the language of monoidal categories with only a little extra structure. Adopting this approach allows us to work at an appropriate level of abstraction: hiding low-level details when possible, and exposing the physical specifics when needed.

Past Research Highlights

Interacting Algebras as a foundation for quantum computing Quantum theory is naturally 2-dimensional, since the sequential composition of

[1]Classical examples: functional programming languages [39], side-effects [42] and data types [30].

[2]Initiated by Abramsky and Coecke [1]

linear operators, and their parallel composition by tensor product, interact in a highly non-trivial way. Therefore studying quantum mechanics within the framework of monoidal categories is a natural and profitable choice. With various collaborators, I have pursued the development of quantum theory in terms of algebraic objects which can be expressed purely in the language of monoidal categories, without reference to Hilbert spaces.

Building on earlier work [8] which showed that quantum observables can be formalised as Frobenius algebras, we discovered [9] that the Frobenius algebras corresponding to a *complementary* pair of observables jointly form a Hopf algebra. Behind these words are a few simple axioms whose equational theory is sufficient to derive a large fraction of quantum theory. Even better, all of this can be done in a beautiful pictorial language which makes calculations very simple.[3]

Various connections have been established between these abstract algebraic results and concrete quantum computation. For example, the equivalence of graph states via local complementation [48] is equivalent to the fact that the Hadamard map can be decomposed into more primitive rotations [25]. Later work [11] showed that these algebras are both necessary and sufficient for Mermin non-locality [41]. Most recently we showed that these algebras exist generically for any complementary observables with given dynamics [23]. This opens the door for work proposed in the next section.

Surprisingly, this previously unstudied combination of Frobenius and Hopf algebras has since appeared in range of other application domains including natural language processing [34], control theory [2], and distributed computing [46]. This suggests that these structures may also be a useful handle for quantum algorithms in these domains.

The ZX-calculus: reasoning about quantum processes In the framework of interacting algebras, by restricting attention to qubits and considering the specific case of the Pauli Z and X observables we obtain the ZX-calculus, a formal theory for reasoning about quantum computation [9]. In the ZX-calculus, an algorithm or protocol is presented as a diagram, similar to an electronic circuit diagram; the equations of the theory are then given as *graph rewrite rules.* The graphical notation directly exposes many features of quantum systems and is particularly well adapted to the study of entanglement.

The ZX-calculus is universal, in the sense that any quantum state or operation can be represented, and reproduces common circuit identities and commutation relations [10]. Perhaps unsurprisingly, graph states have a very simple presentation, and the whole framework of measurement-based quantum computing [44] fit very comfortably in the calculus [22]. However

[3]The forthcoming textbook [13] offers a "first course in quantum theory" based on these ideas.

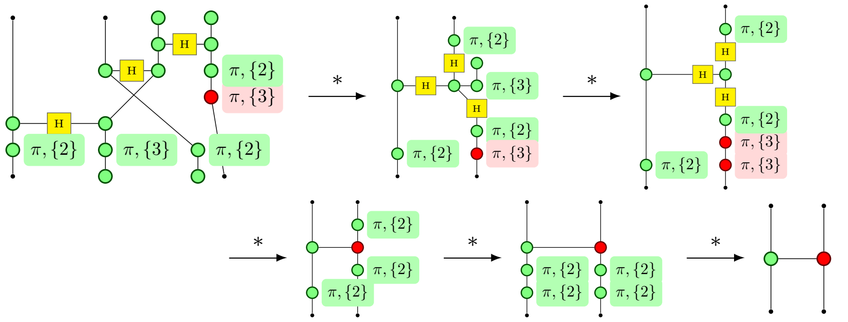

Figure 1: The ZX-calculus in action: an MBQC implementation (top-left) is verified by rewriting it to its specification, a single CNOT-gate (bottom-right)

since the ZX-calculus is about the underlying algebra of observables rather than some gate set or other translation between models is rather easy. In [21], we showed that a quantum computation implemented in the one-way model can be rewritten into an equivalent quantum circuit without any ancillae. Further, since this procedure is given by a rewrite strategy, it can be used to show that the given MBQC program is free of (certain) programming errors; see Figure 1 for an example.

The ZX-calculus has also been used to formalise and verify quantum error correcting codes [24][36], the major algorithms [50, 13], and a variety of communication protocols [35].

Reasoning in 2 Dimensions: string diagrams A distinctive feature of monoidal categories is the use of *string diagrams* in place of conventional mathematical syntax. String diagrams are 2-dimensional notation that offers a huge improvement in terms of simplicity and clarity. Such diagrams are used widely categorical quantum mechanics — for example in the ZX-calculus— but have also been adopted in such diverse areas as quantum thermodynamics [7], functional programming [43], and asynchronous circuits [29].

However the graphical syntax is itself a mathematical object, and to be confident that reasoning by rewriting is sounds it was necessary to formalise it. This was partially achieved in my thesis [20]; the completed theory [18, 19] is closely related to conventional DPO rewriting [26]. However, since the axioms of the ZX-calculus are most usefully presented as an infinite axiom schema, reasoning about concrete graphs is inadequate for practical purposes: we need to represent the whole schema as a single finite object. We introduced the notion of *graph pattern* [17] to address this problem.

When dealing with large or complex systems, manipulating the diagrammatic syntax can become laborious and error-prone. The tool *Quantomatic* [16] allows the user to construct such diagrams and to manipulate them using arbitrary rewrite rules. The program is not restricted to quantum theory: it can work with any graphical language. Quantomatic has been used to verify the correctness of a various quantum protocols [24][35].

Proposed Research

The work described in the previous section is quite abstract. My aim in joining QuSoft is to apply this theoretical approach to concrete quantum devices. My work offers many opportunities for the development of quantum software.

The ZX-calculus and its generalisations represent the algebra of quantum mechanics itself rather any particular physical system or model of quantum computation. Since, for general reasons, the interacting algebras of the theory exist in any quantum system, they offer a universal framework for quantum computation. This means that the ZX-calculus is uniquely well-placed to serve as an *intermediate representation* for all kinds of quantum software, decoupled from the implementation technology, targetable from any programming system. Since we have basically no idea what the eventual hardware is going to look like, a platform independent approach to quantum software is absolutely necessary.

The ZX-calculus has a rich equational theory which makes possible a variety of useful transformations of the program, including optimisation, simulation, and partial evaluation. Since the structures involved are generic, ZX-calculus terms can be faithfully translated not only to the physics implementing the computer, but also to automatically add error correction schemes, or change the gate set.

The ultimate goal of the research proposed below is to construct a complete toolchain for the development of quantum software for use in practical quantum algorithms and realistic devices. This will be built by adapting and generalising the existing ZX-calculus formalism to serve as the intermediate representation of an optimising retargetable compiler. The development of such a platform-agnostic middle layer will bring numerous benefits to the work of QuSoft.

Short Term Goals

ZX for concrete implementations The ZX-calculus is based on an ideal qubit. Real implementations of qubits inevitably deviate from this ideal, by having additional energy levels, non-negligible back action of measurement, or any number of other ways. Further, the quantum logic gates will rarely be atomic operations at the physical level. However, all quantum observables admit a ZX-calculus-like formulation [23], hence it is possible to formalise the implementation of the qubit in the same kind of language as its ideal version. I propose to study concrete qubit implementations and produce "ZX-calculi" for them, incorporating the specifities of their physics into the formalism. Two obvious candidates for this study are superconducting qubits [15, 45] and diamond NV spin qubits [6].

Formalisation of existing protocols and algorithms I propose to formalise and prove the correctness of a wide variety of quantum algorithms and protocols in the ZX-calculus, both in their textbook versions and, where possible, as implemented on real hardware. Many algorithms have already been formalised this way (see e.g. [51]) at least in the textbook version; however error correcting codes and fault tolerant operations have largely been ignored so I will concentrate on these. The aim of doing this is two-fold. Firstly to discover which parts of the algebraic structure the algorithms rely upon, and to discover if something essential is missing from the axioms. Secondly, we will build up a library of verified building blocks which may be combined into larger programs, to facilitate high-level programming.

Homomorphic programming and compilation One of the main advantages of using category theory is that it provides a systematic way of transporting structure from one setting to another[4]. This realisation is at the heart of this proposal. By providing functors from, say, the category of ZX-calculus terms, to the category of codewords and fault-tolerant gates, to the category of observables of our implementing hardware, we implement a compilation procedure from a textbook presentation of an algorithm to the version which can run on the hardware. By functoriality, we don't just transport the program, but also all the program transformations too, so that any rewriting performed can be soundly transported to the next stage (and in certain cases, reflected back to the previous one).

One additional task here is understand which graphical terms are actually runnable on a given architecture. For example, not all ZX-calculus terms are quantum circuits; we provide a graph theoretic characterisation to recognise them [22].

Extending the graphical formalism The graphical notation developed for categorical quantum mechanics is extremely readable for humans and corresponds well to quantum circuits. However graphs are finite objects, whereas we often wish to handle algorithms which are described by a uniform family of circuits rather than a single circuit. To permit this it was necessary to introduce *graph patterns* [18], a simple "regular expression" language for graphs. A key objective is to extend the existing (rudimentary) pattern language to more expressive forms, and to develop techniques for handling recursion and induction inside more expressive graphical languages. Such development would permit the familiar constructs of classical programming

[4]The other main advantage is that everything is compositional.

languages inside the graphical calculus. Work on this problem, based on operads [40, 38, 47], is currently in progress.

Long Term Goals

Quantitative aspects The mainstream quantum information processing literature abounds with numbers: Bell inequalities, entropies, measures of entanglement or dischord, channel capacities, error rates, and many others. In contrast, the existing work in the categorical quantum mechanics programme and is almost entirely qualitative. While the framework allows for the calculation of probabilities, this aspect of quantum mechanics is largely neglected. The addition of an expressive quantitative aspect to the framework is a crucial task.

Various researchers have treated specific aspects of this problem [3], [14, 7, 28, 27]. The most direct route to this goal appears to be via unification of general probabilistic theories [4] and general compositional theories [12]. Such a unification would permit the treatment of device independence within the categorical framework.

Verification of quantum programs via rewriting Verifying that a program meets its specification is famously difficult. Using ZX-calculus quantum programs may be optimised, or their execution simulated, or their correctness checked. To repeat an example described above, a one-way program can be rewritten to an equivalent quantum circuit: the success of this procedure is a correctness proof of the original program, while its failure produces a certificate of a bug [21, 22]. This is pure equational reasoning; however once pattern graphs (see above) enter the picture inductive reasoning is possible [37]. The extended notion of pattern graph we propose above will give rise almost automatically to induction principles, however it should be possible to go further. We aim to develop a rich program logic based on using graph patterns as modal formulae.

Automated Reasoning Rewriting diagrams is time consuming and error-prone when done by hand. Machine support is therefore essential to large-scale use of these techniques. The existing interactive graphical proof-assistant Quantomatic [16] provides a good starting point for mechanised implementation of the compilation procedure described above. It will be necessary to develop rewriting strategies to effect the desired program transformations, and theorem proving tactics to establish their properties with minimal human intervention. This is less fanciful than it sounds: Quantomatic is the software underlying the PSGraph/Tinker toolset [32, 33], which is used for the verification of safety critical software for the automotive industry.

Quantum programming with dependent types The story so far has focussed on the back end of the compiler, but of course there should also be a language to compile. The technology developed for the previous objectives would amount to a mechanisation of a significant chunk of quantum theory. It is my final proposal to design a high-level quantum programming language with dependent types in the style of Agda [5]: that is, a language whose type system is able to enforce strong guarantees about the behaviour of the programs. Unlike extant quantum programming languages [31, 49], the proposed language could take advantage of the ZX-calculus to reason about quantum mechanics as part of its typing judgements. This might enable, for example, the compiler to automatically insert the appropriate number of rounds of distillation into an entanglement using protocol. Evidently, this requires the earlier phases of this proposal to be at least partially completed, and is by far the most speculative thing listed here.

References

[1] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science: LICS 2004*, pages 415–425. IEEE Computer Society, 2004.

[2] J. C. Baez and J. Erbele. Categories in control. *Theory and Applications of Categories*, 30(24):836–881, 2015, arXiv:1405.6881.

[3] H. Barnum, R. Duncan, and A. Wilce. Convexity, categorical semantics and the foundations of physics. In B. Coecke, P. Panangaden, and P. Selinger, editors, *Proceedings of 7th Workshop on Quantum Physics and Logic (QPL 2010)*, 2010.

[4] J. Barrett. Information processing in generalized probabilistic theories. *Phys. Rev. A*, 75(032304), 2007, arXiv:quant-ph/0508211v3.

[5] A. Bove, P. Dybjer, and U. Norell. A brief overview of agda – a functional language with dependent types. In *proceedings of TPHOLs*. Springer, 2009.

[6] L. Childress and R. Hanson. Diamond nv centers for quantum computing and quantum networks. *MRS Bulletin*, 38(2):134–138, 002 2013.

[7] G. Chiribella and C. M. Scandolo. Entanglement and thermodynamics in general probabilistic theories. *New Journal of Physics*, 17(10):103027, 2015.

[8] B. Coecke, D. Pavlovic, and J. Vicary. A new description of orthogonal bases. *Math. Structures in Comp. Sci.*, 23(3):555–567, 2013, arxiv:0810.0812.

[9] B. Coecke and R. Duncan. Interacting quantum observables: Categorical algebra and diagrammatics. *New J. Phys*, 13(043016), 2011, arXiv:0906.4725.

[10] B. Coecke and R. Duncan. Tutorial: Graphical calculus for quantum circuits. In *Reversible Computation, 4th International Workshop, RC 2012, Copenhagen, Denmark, July 2-3, 2012.*, vol 7581 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2013.

[11] B. Coecke, R. Duncan, A. Kissinger, and Q. Wang. Strong complementarity and non-locality in categorical quantum mechanics. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science. (LiCS2012)*, pages 245–254. IEEE Computer Society Press, 2012, arXiv:1203.4988.

[12] B. Coecke, R. Duncan, A. Kissinger, and Q. Wang. Generalised compositional theories and diagrammatic reasoning. In G. Chirabella and R. Spekkens, editors, *Quantum Theory: Informational Foundations and Foils*. Springer, 2015, arXiv:1506.03632.

[13] B. Coecke and A. Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Pressm, 2017.

[14] B. Coecke and R. W. Spekkens. Picturing classical and quantum bayesian inference. *Synthese*, 186(3):651–696, 2012, arXiv:1102.2368v1.

[15] L. DiCarlo, J. M. Chow, J. M. Gambetta, L. S. Bishop, B. R. Johnson, D. I. Schuster, J. Majer, A. Blais, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf. Demonstration of two-qubit algorithms with a superconducting quantum processor. *Nature*, 460(7252):240–244, 07 2009.

[16] L. Dixon, R. Duncan, B. Frot, A. Kissinger, A. Merry, D. Quick, M. Soloviev, and V. Zamdzhiev. Quantomatic. https://sites.google.com/site/quantomatic/.

[17] L. Dixon and R. Duncan. Extending graphical representations for compact closed categories with applications to symbolic quantum computation. In S. Autexier, J. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 - August 1, 2008. Proceedings*, vol 5144 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2008.

[18] L. Dixon and R. Duncan. Graphical reasoning in compact closed categories for quantum computation. *Annals of Mathematics and Artificial Intelligence*, 56(1):23–42, 2009, arXiv:0902.0514.

[19] L. Dixon, R. Duncan, and A. Kissinger. Open graphs and computational reasoning. In *Proceedings DCM 2010*, vol 26 of *Electronic Proceedings in Theoretical Computer Science*, pages 169–180, 2010, arXiv:1006.1937.

[20] R. Duncan. *Types for Quantum Computing*. PhD thesis, Oxford University, 2006.

[21] R. Duncan and S. Perdrix. Rewriting measurement-based quantum computations with generalised flow. In S. Abramsky, C. Gavoille, C. Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Proceedings Part II*, vol 6199 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2010.

[22] R. Duncan. A graphical approach to measurement-based quantum computing. In C. Heunen, M. Sadrzadeh, and E. Grefenstette, editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*, chapter 3. Oxford University Press, 2013, arxiv:1203.6242.

[23] R. Duncan and K. Dunne. Interacting Frobenius algebras are Hopf. In M. Grohe, E. Koskinen, and N. Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, LICS '16, pages 535–544. ACM, 2016, arXiv:1601.04964.

[24] R. Duncan and M. Lucas. Verifying the Steane code with Quantomatic. In B. Coecke and M. Hoban, editors, *Proceedings 10th International Workshop on Quantum Physics and Logic (QPL 2013)*, vol 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–49, 2014, arXiv:1306.4532.

[25] R. Duncan and S. Perdrix. Graph states and the necessity of Euler decomposition. In K. Ambos-Spies, B. Löwe, and W. Merkle, editors, *Computability in Europe: Mathematical Theory and Computational Practice (CiE'09)*, vol 5635 of *Lecture Notes in Computer Science*, pages 167–177. Springer, 2009, arXiv:0902.0500.

[26] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. Springer Berlin Heidelberg, 2006.

[27] B. Fong and H. Nava-Kopp. Additive monotones for resource theories of parallel-combinable processes with discarding. In *Proceedings 12th International Workshop on Quantum Physics and Logic, QPL 2015, Oxford, UK, July 15-17, 2015.*, pages 170–178, 2015.

[28] T. Fritz. Resource convertibility and ordered commutative monoids. *Mathematical Structures in Computer Science*, pages 1–89, 10 2015, arXiv:1504.03661.

[29] D. R. Ghica. Diagrammatic reasoning for delay-insensitive asynchronous circuits. In *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky*, pages 52–68. Springer, 2013.

[30] J. A. Goguen and J. W. Thatcher. Initial algebra semantics. *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, 0:63–77, 1974.

[31] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron. Quipper: A scalable quantum programming language. In *Programming language design and implementation (PLDI'13)*, vol 48 of *ACM SIGPLAN Notices*, pages 333–342, 2013, arXiv:1304.3390.

[32] G. Grov, A. Kissinger, and Y. Lin. A graphical language for proof strategies. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-19)*, vol 8312 of *Lecture Notes in Computer Science*, pages 324–339. Springer, 2013, arXiv:1302.6890.

[33] G. Grov, A. Kissinger, and Y. Lin. Tinker, tailor, solver, proof. In C. Benzmüller and B. W. Paleo, editors, *Proceedings Eleventh Workshop on User Interfaces for Theorem Provers (UITP 2014)*, vol 167 of *EPTCS*, pages 23–34, 2014, arXiv:1410.8217.

[34] J. Hedges and M. Sadrzadeh. A generalised quantifier theory of natural language in categorical compositional distributional semantics with bialgebras. *CoRR*, abs/1602.01635, 2016, arxiv:1602.01635.

[35] A. Hillebrand. Quantum protocols involving multiparticle entanglement and their representations in the ZX-calculus. Master's thesis, Oxford University, 2011.

[36] C. Horsman. Quantum picturalism for topological cluster-state computing. *New J. Phys.*, 13(095011), September 2011.

[37] A. Kissinger and D. Quick. A first-order logic for string diagrams. In L. S. Moss and P. Sobocinski, editors, *6th Conference on Algebra and Coalgebra in Computer Science (CALCO 2015)*, vol 35 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 171–189. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2015, arXiv:1505.00343.

[38] Y. Lafont. Diagram rewriting and operads. Lecture given at the Thematic school : Operads CIRM, Luminy (Marseille), 20-25 April 2009, February 2010.

[39] J. Lambek and P. J. Scott. *Introduction to higher order categorical logic*, vol 7 of *Cambridge studies in advanced mathematics*. Cambridge University Press, 1986.

[40] T. Leinster. *Higher Operads, Higher Categories*. London Mathematical Society. Cambridge University Press, 2003, http://arxiv.org/abs/math.CT/0305049.

[41] N. D. Mermin. Quantum mysteries revisited. *American Journal of Physics*, 58(8):731–734, 1990.

[42] E. Moggi. Notions of computation and monads. *Inf. Comput.*, 93(1):55–92, July 1991.

[43] M. Piróg and N. Wu. String diagrams for free monads (functional pearl). In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, ICFP 2016, pages 490–501, New York, NY, USA, 2016. ACM.

[44] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, 2001.

[45] D. Riste, M. Dukalski, C. A. Watson, G. de Lange, M. J. Tiggelman, Y. M. Blanter, K. W. Lehnert, R. N. Schouten, and L. DiCarlo. Deterministic entanglement of superconducting qubits by parity measurement and feedback. *Nature*, 502(7471):350–354, 10 2013.

[46] P. Sobociński. Nets, relations and linking diagrams. In R. Heckel and S. Milius, editors, *Algebra and Coalgebra in Computer Science*, vol 8089 of *Lecture Notes in Computer Science*, pages 282–298. Springer Berlin Heidelberg, 2013.

[47] D. Vagner, D. I. Spivak, and E. Lerman. Algebras of open dynamical systems on the operad of wiring diagrams. *Theory and Applications of Categories*, 30(51):1793–1822, 2015, arXiv:1408.1598.

[48] M. Van den Nest, J. Dehaene, and B. De Moor. Graphical description of the action of local clifford transformations on graph states. *Physical Review A*, 69:022316, 2004.

[49] D. Wecker and K. M. Svore. Liqui|>: A software design architecture and domain-specific language for quantum computing. February 2014, arXiv:1402.4467.

[50] W. Zeng. *The Abstract Structure of Quantum Algorithms*. PhD thesis, Oxford University, 2015, arXiv:1512.08062.

[51] W. Zeng and J. Vicary. Abstract structure of unitary oracles for quantum algorithms. In *Proceedings QPL 2014*, vol 172 of *EPTCS*, pages 270–284, 2014, arXiv:1406.1278.