Diagrammatic Reasoning and Quantum Computation

Aleks Kissinger

ACA, Kalamata

November 4, 2015

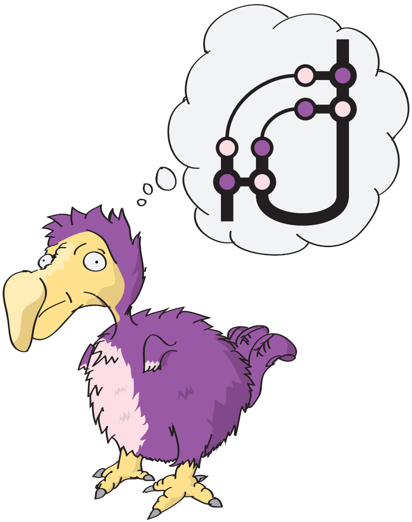
Quantum Group | Department of Computer Science, University of Oxford ⇒ Radboud University

Picturing Quantum Processes

A first course in quantum theory and diagrammatic reasoning

Bob Coecke & Aleks Kissinger

CUP 2015



Algebra and rewriting

- An *algebraic theory* consists of a set of operations and constants, satisfying certain equations

Algebra and rewriting

- An *algebraic theory* consists of a set of operations and constants, satisfying certain equations
- e.g. a monoid consists of a binary operation and constant e such that:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{and} \qquad a \cdot e = a = e \cdot a$$

Algebra and rewriting

- An *algebraic theory* consists of a set of operations and constants, satisfying certain equations
- e.g. a monoid consists of a binary operation and constant e such that:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{and} \qquad a \cdot e = a = e \cdot a$$

- We can apply an equation as a *term rewrite rule*

Algebra and rewriting

- An *algebraic theory* consists of a set of operations and constants, satisfying certain equations
- e.g. a monoid consists of a binary operation and constant e such that:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{and} \qquad a \cdot e = a = e \cdot a$$

- We can apply an equation as a *term rewrite rule*
- Instantiate free variables:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \;\Big/\; \begin{cases} a := x \\ b := (y \cdot e) \\ c := z \end{cases}$$

Algebra and rewriting

- An *algebraic theory* consists of a set of operations and constants, satisfying certain equations
- e.g. a monoid consists of a binary operation and constant e such that:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad \text{and} \qquad a \cdot e = a = e \cdot a$$

- We can apply an equation as a *term rewrite rule*
- Instantiate free variables:

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \Big/ \begin{cases} a := x \\ b := (y \cdot e) \\ c := z \end{cases}$$

then replace a sub-term:

$$w \cdot ((x \cdot (y \cdot e)) \cdot z) \quad \rightsquigarrow \quad w \cdot (x \cdot ((y \cdot e) \cdot z))$$

Algebra and rewriting

- Alternatively, we could write these equations as trees:

Algebra and rewriting

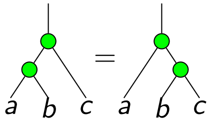
- Alternatively, we could write these equations as trees:

- In which case:

$$w \cdot ((x \cdot (y \cdot e)) \cdot z) \quad \rightsquigarrow \quad w \cdot (x \cdot ((y \cdot e) \cdot z))$$

Algebra and rewriting

- Alternatively, we could write these equations as trees:



- In which case:

$$w \cdot ((x \cdot (y \cdot e)) \cdot z) \quad \rightsquigarrow \quad w \cdot (x \cdot ((y \cdot e) \cdot z))$$

becomes:

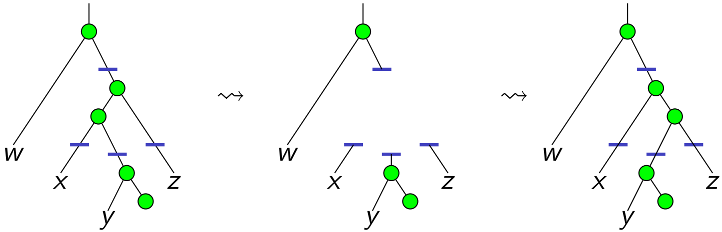


Diagram substitution

- Note we can drop the free variables:

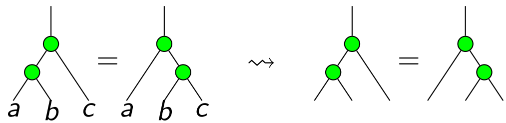
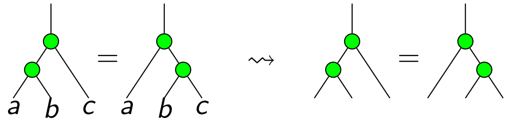


Diagram substitution

- Note we can drop the free variables:



- The role of variables is replaced by the fact that the LHS and RHS have a *shared boundary*:

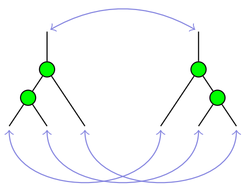
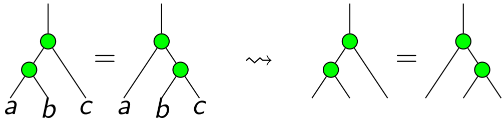
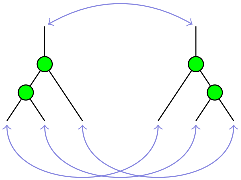


Diagram substitution

- Note we can drop the free variables:



- The role of variables is replaced by the fact that the LHS and RHS have a *shared boundary*:



- This treats inputs and outputs symmetrically

Algebra and coalgebra

- We can consider structures with many *outputs* as well as inputs.

Algebra and coalgebra

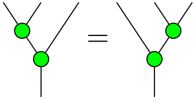
- We can consider structures with many *outputs* as well as inputs.
- *Coalgebraic structures*: algebraic structures “upside-down”

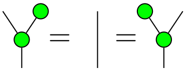
Algebra and coalgebra

- We can consider structures with many *outputs* as well as inputs.
- *Coalgebraic structures*: algebraic structures "upside-down"
- e.g. a *comonoid* satisfies:

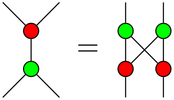
Algebra and coalgebra

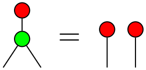
- We can consider structures with many *outputs* as well as inputs.
- *Coalgebraic structures*: algebraic structures "upside-down"
- e.g. a *comonoid* satisfies:

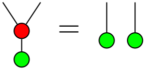




- The most interesting structures consist of algebras *interacting* with coalgebras:

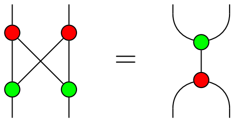






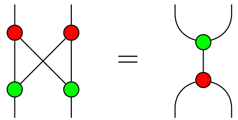
Equational reasoning with diagram substitution

- Again, we use equations to perform substitutions, but on graphs rather than just trees

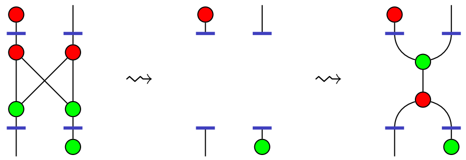


Equational reasoning with diagram substitution

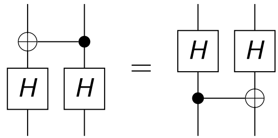
- Again, we use equations to perform substitutions, but on graphs rather than just trees



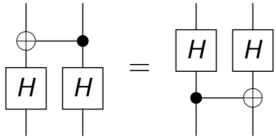
- For example:

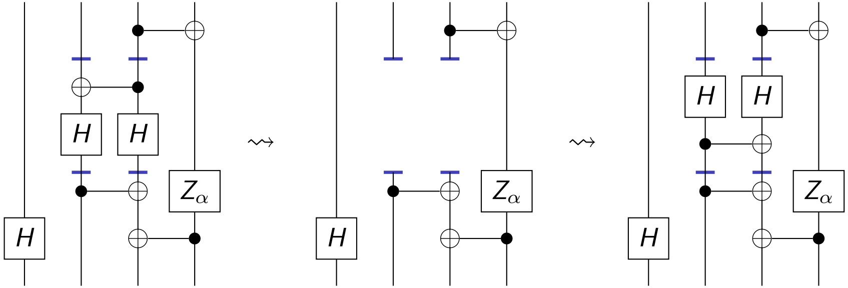


Example: Quantum circuit rewriting

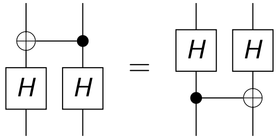


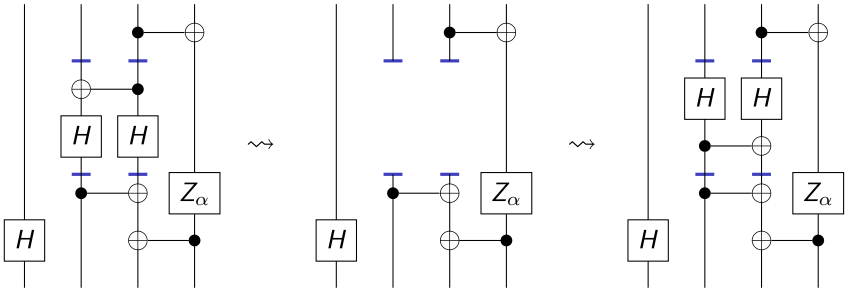
Example: Quantum circuit rewriting





Example: Quantum circuit rewriting



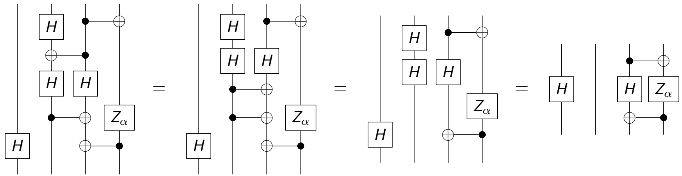


So, we can define an *equational theory* for quantum circuits, using rewriting.

Why an equational theory for quantum circuits?

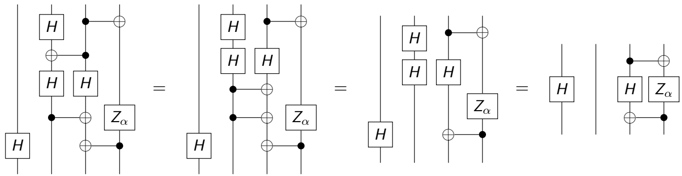
Why an equational theory for quantum circuits?

- circuit optimization:

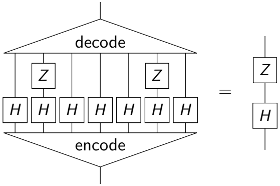


Why an equational theory for quantum circuits?

- circuit optimization:



- verify equivalence (e.g. when adding error-correction)



- (automated) translation to other gate sets and paradigms
- exploit *algebraic invariants* to prove properties about *computations*

A complete set of gate identities

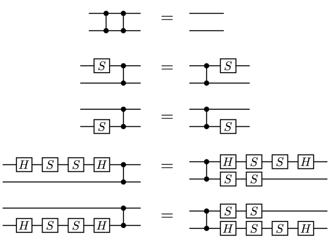
- These equations are complete for *Clifford circuits*:

$$\omega^8 = 1 \qquad \text{(C1)}$$

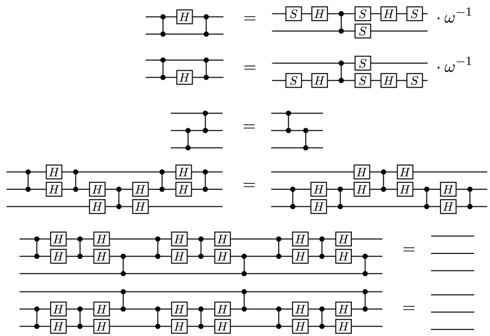
$$H^2 = 1 \qquad \text{(C2)}$$

$$S^4 = 1 \qquad \text{(C3)}$$

$$SHSHSH = \omega \qquad \text{(C4)}$$



(C5) (C6) (C7) (C8) (C9)



(C10) (C11) (C12) (C13) (C14) (C15)

(Selinger 2013)

As an equational theory

- The good:

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] \;=\; [\![C_2]\!] \quad\Longrightarrow\quad C_1 \;=_E\; C_2$$

  - unique normal forms

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

  - unique normal forms
  - relatively compact (3 generators, 15 rules)

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

  - unique normal forms
  - relatively compact (3 generators, 15 rules)
- The bad:

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

  - unique normal forms
  - relatively compact (3 generators, 15 rules)
- The bad:
  - rules are large, and don't carry any intuition or algebraic structure

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \quad \Longrightarrow \quad C_1 =_E C_2$$

  - unique normal forms
  - relatively compact (3 generators, 15 rules)
- The bad:
  - rules are large, and don't carry any intuition or algebraic structure
  - rewrite strategy is complicated (17 derived gates, 100 derived rules)

As an equational theory

- The good:
  - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

  - unique normal forms
  - relatively compact (3 generators, 15 rules)
- The bad:
  - rules are large, and don't carry any intuition or algebraic structure
  - rewrite strategy is complicated (17 derived gates, 100 derived rules)
- The ugly:

As an equational theory

- The good:
    - complete for Clifford circuits:

$$[\![C_1]\!] = [\![C_2]\!] \implies C_1 =_E C_2$$

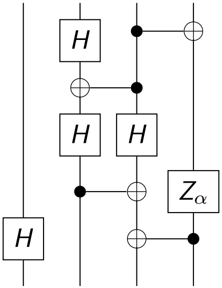
    - unique normal forms
    - relatively compact (3 generators, 15 rules)
- The bad:
    - rules are large, and don't carry any intuition or algebraic structure
    - rewrite strategy is complicated (17 derived gates, 100 derived rules)
- The ugly:
    - proof of completeness is *extremely* complicated ($>$ 100 pages long! though mostly machine-generated)

Can we do better?

- Yes!

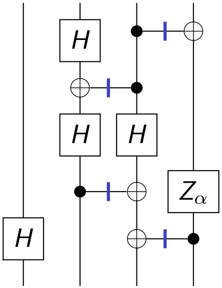
Can we do better?

- Yes!
- We can capture underlying algebraic structure by decomposing gates into smaller pieces

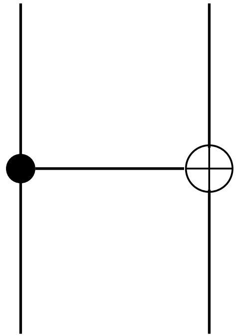


Can we do better?

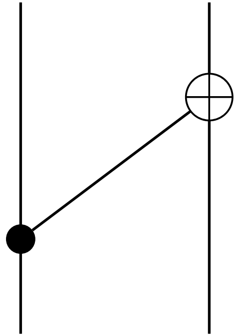
- Yes!
- We can capture underlying algebraic structure by decomposing gates into smaller pieces



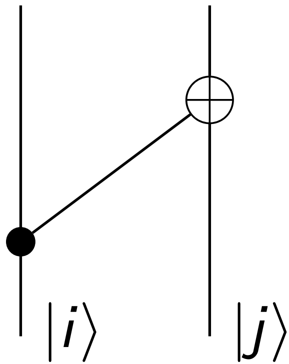
Decomposing CNOT



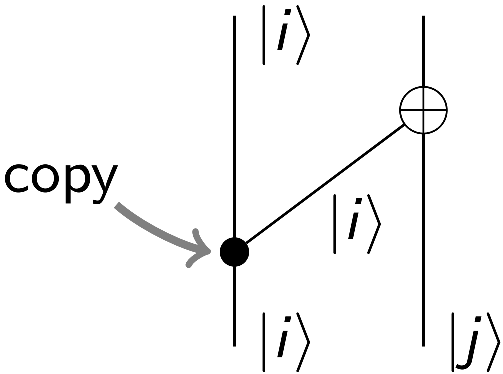
Decomposing CNOT



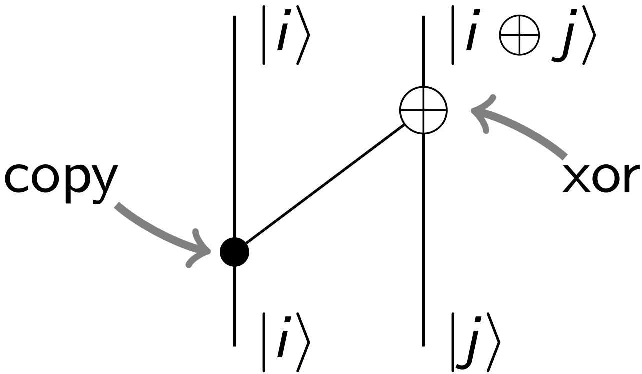
Decomposing CNOT



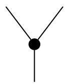
Decomposing CNOT



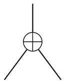
Decomposing CNOT



‘Copy’ maps

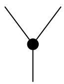


$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

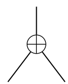


$$\begin{cases} |00\rangle \mapsto |0\rangle \\ |01\rangle \mapsto |1\rangle \\ |10\rangle \mapsto |1\rangle \\ |11\rangle \mapsto |0\rangle \end{cases}$$

‘Copy’ maps

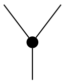


$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

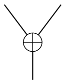


$$\begin{cases} |{+}{+}\rangle \mapsto |{+}\rangle \\ |{-}{-}\rangle \mapsto |{-}\rangle \end{cases}$$

'Copy' maps

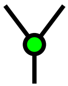


$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

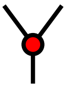


$$\begin{cases} |+\rangle \mapsto |++\rangle \\ |-\rangle \mapsto |--\rangle \end{cases}$$

‘Copy’ maps

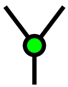


$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$



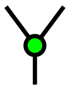
$$\begin{cases} |+\rangle \mapsto |{+}{+}\rangle \\ |-\rangle \mapsto |{-}{-}\rangle \end{cases}$$

'Copy' maps



$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

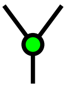
‘Copy’ maps



$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

$$\begin{cases} |0\rangle \mapsto 1 \\ |1\rangle \mapsto 1 \end{cases}$$

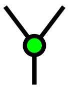
'Copy' maps



$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

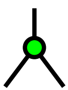
$$\big\{ \langle 0| + \langle 1|$$

‘Copy’ maps



$$\begin{cases} |0\rangle \mapsto |00\rangle \\ |1\rangle \mapsto |11\rangle \end{cases}$$

$$\begin{cases} \langle 0| + \langle 1| \end{cases}$$

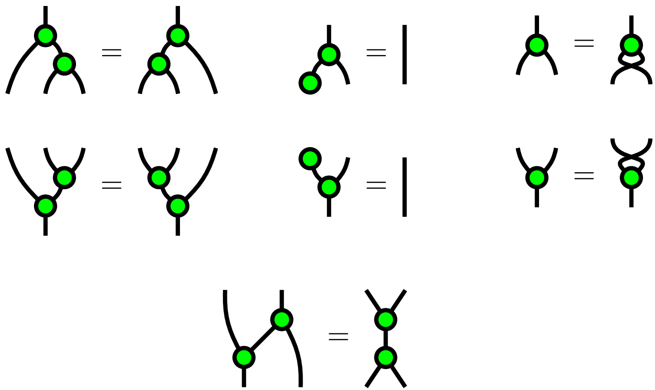


$$\begin{cases} |00\rangle \mapsto |0\rangle \\ |11\rangle \mapsto |1\rangle \end{cases}$$

$$\begin{cases} |0\rangle + |1\rangle \end{cases}$$

Algebraic identities...

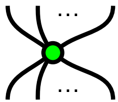
These satisfy 8 identities:



...making them a *commutative Frobenius algebra*.

But luckily...

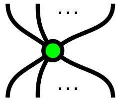
...you don't need to remember all that! The only thing to remember is, for:



$$:= \begin{cases} |0..0\rangle \mapsto |0...0\rangle \\ |1..1\rangle \mapsto |1...1\rangle \end{cases}$$

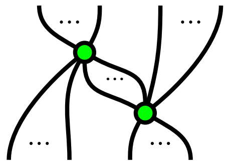
But luckily...

...you don't need to remember all that! The only thing to remember is, for:

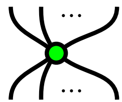


$$:= \begin{cases} |0..0\rangle \mapsto |0...0\rangle \\ |1..1\rangle \mapsto |1...1\rangle \end{cases}$$

we have:

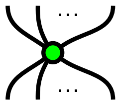


$$=$$



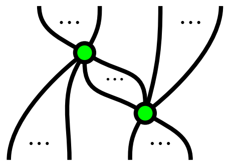
But luckily...

...you don't need to remember all that! The only thing to remember is, for:

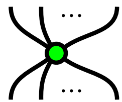


$$:= \begin{cases} |0..0\rangle \mapsto |0...0\rangle \\ |1..1\rangle \mapsto |1...1\rangle \end{cases}$$

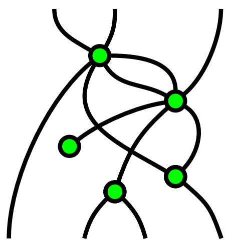
we have:



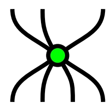
$$=$$



or equivalently:

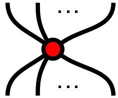


$$=$$



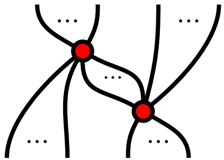
But luckily...

...you don't need to remember all that! The only thing to remember is, for:

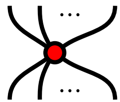


$$:= \begin{cases} |{+}..{+}\rangle \mapsto |{+}...{+}\rangle \\ |{-}..{-}\rangle \mapsto |{-}...{-}\rangle \end{cases}$$

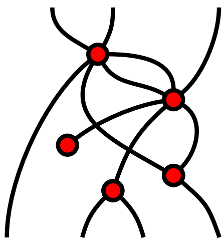
we have:



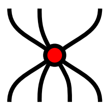
$$=$$



or equivalently:

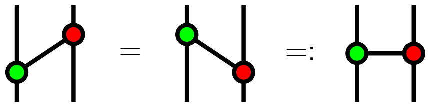


$$=$$



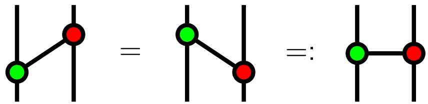
What about 2-colour diagrams?

Direction of edges doesn't matter:

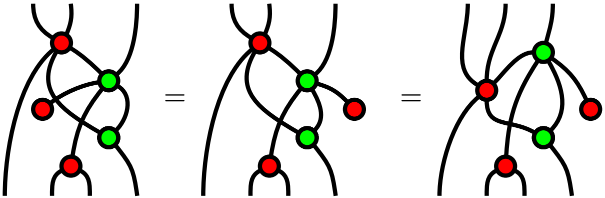


What about 2-colour diagrams?

Direction of edges doesn't matter:

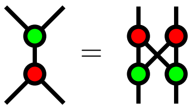


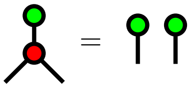
...in fact, **only topology matters**:

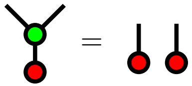


Interaction: Hopf algebra

Red + green spiders also satisfy:

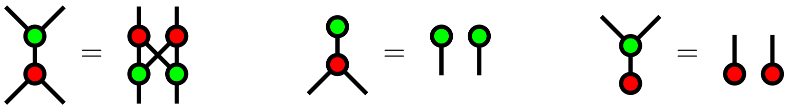




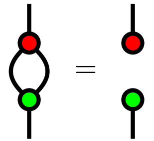


Interaction: Hopf algebra

Red + green spiders also satisfy:

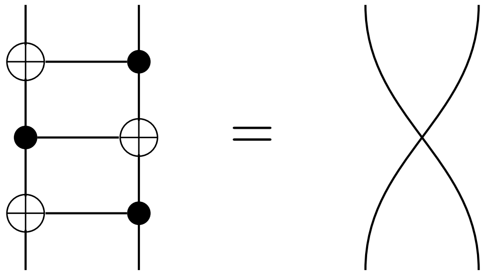


...from which we can derive:

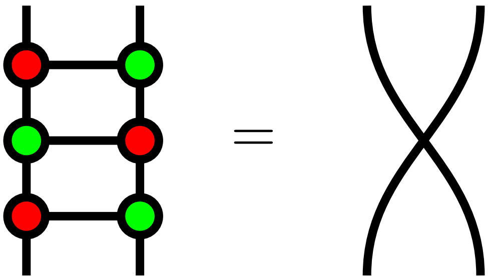


make the overall structure into a *Hopf algebra*

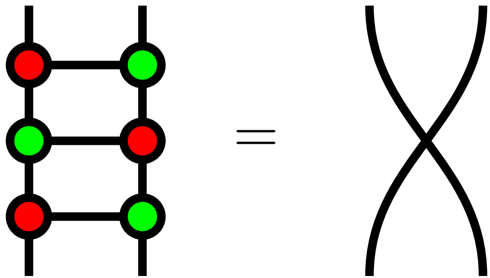
Circuit calculation



Circuit calculation

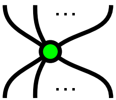


Circuit calculation

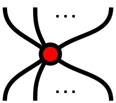


$(\Leftarrow \quad)$

Making spiders universal

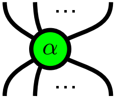


$$:= \begin{cases} |0..0\rangle \mapsto |0...0\rangle \\ |1..1\rangle \mapsto |1...1\rangle \end{cases}$$

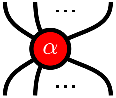


$$:= \begin{cases} |+..+\rangle \mapsto |+...+\rangle \\ |-..-\rangle \mapsto |-...-\rangle \end{cases}$$

Making spiders universal

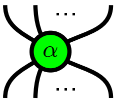


$$:= \begin{cases} |0..0\rangle \mapsto |0...0\rangle \\ |1..1\rangle \mapsto e^{i\alpha} |1...1\rangle \end{cases}$$

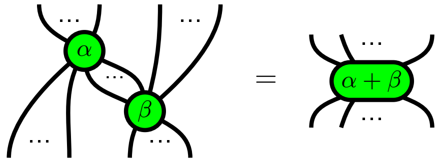


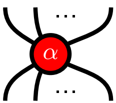
$$:= \begin{cases} |+..+\rangle \mapsto |+...+\rangle \\ |-..-\rangle \mapsto e^{i\alpha} |-...-\rangle \end{cases}$$

Making spiders universal

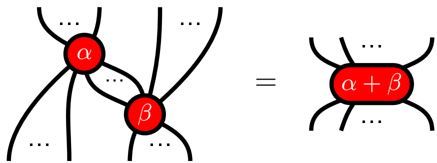


$$:= \begin{cases} |0..0\rangle \mapsto |0\ldots0\rangle \\ |1..1\rangle \mapsto e^{i\alpha}\,|1\ldots1\rangle \end{cases}$$





$$:= \begin{cases} |+..+\rangle \mapsto |+\ldots+\rangle \\ |-..-\rangle \mapsto e^{i\alpha}\,|-\ldots-\rangle \end{cases}$$



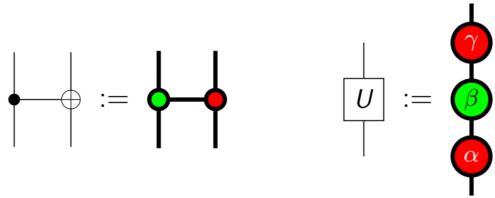
Making spiders universal

Theorem

Phased spiders are universal for qubit quantum computation.

Proof.

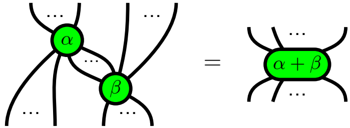
Let:

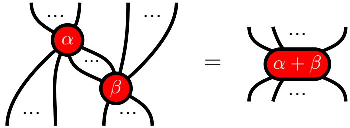


□

The ZX-calculus

The **ZX-calculus** consists of the two spider-fusion rules:

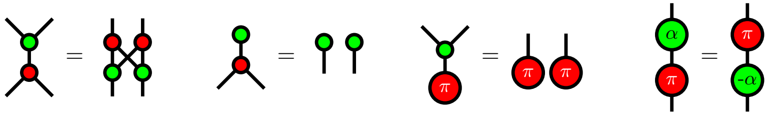




The ZX-calculus

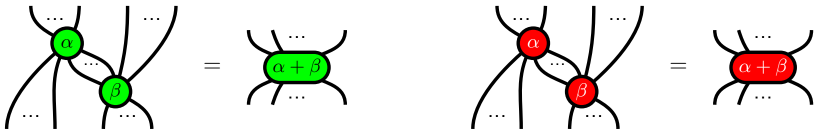
The **ZX-calculus** consists of the two spider-fusion rules:

four Interaction rules:

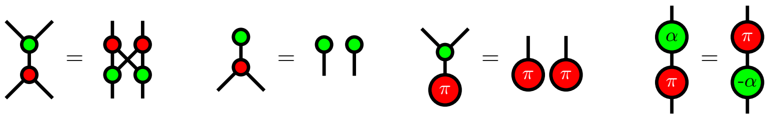


The ZX-calculus

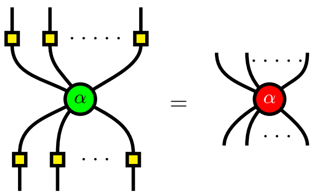
The **ZX-calculus** consists of the two spider-fusion rules:



four Interaction rules:



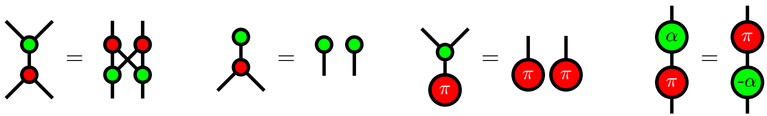
and the Colour Change rule:



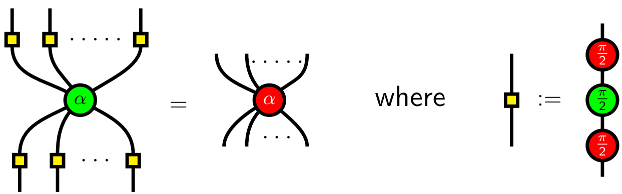
The ZX-calculus

The **ZX-calculus** consists of the two spider-fusion rules:

four Interaction rules:



and the Colour Change rule:

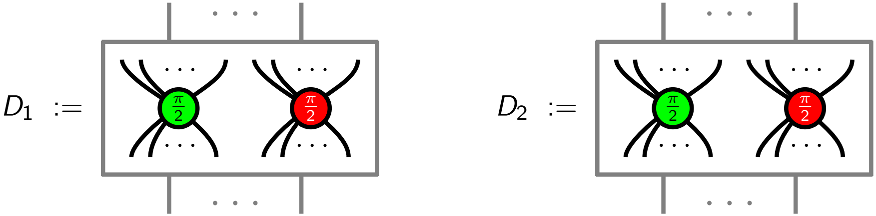


Completeness

Theorem (Backens 2013)

The ZX-calculus is complete for Clifford ZX-diagrams:

$$[\![D_1]\!] = [\![D_2]\!] \implies D_1 =_{ZX} D_2$$



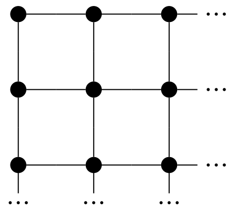
Measurement-based quantum computing

Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model

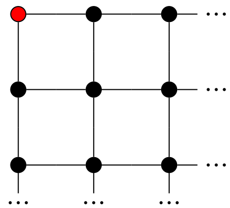
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



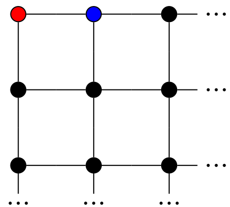
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



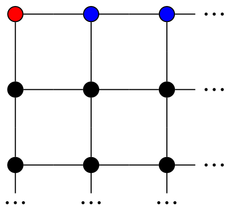
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



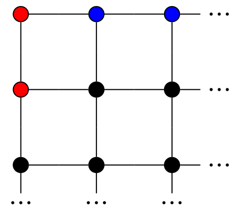
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



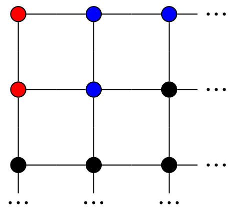
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



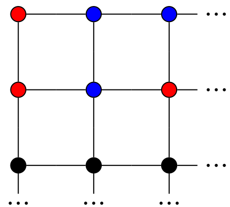
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



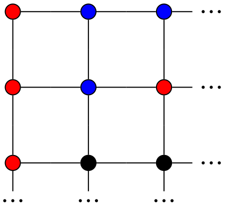
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



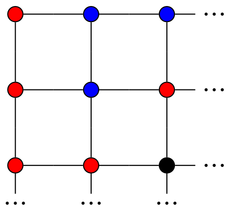
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



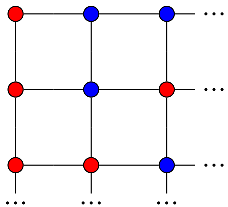
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



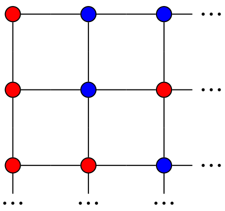
Measurement-based quantum computing

- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



Measurement-based quantum computing

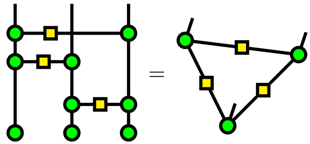
- **Measurement-based quantum computing** is an alternative (and equivalent) paradigm to the circuit model
- Rather than repeatedly applying operations to a small number of systems, start with a big entangled state called a **graph state** and do many **local measurements** in different bases:



- But crucially, the **choices of measurements** can depend on **past measurement outcomes**. This is called **feed-forward**, and it's where all the magic happens.

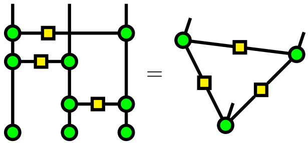
Graph states and cluster states

- Graph states are prepared by starting with many qubits in the $|+\rangle$ state and creating entanglement with controlled-Z operations:

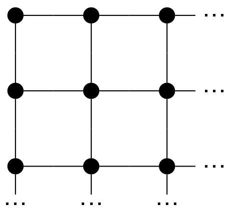


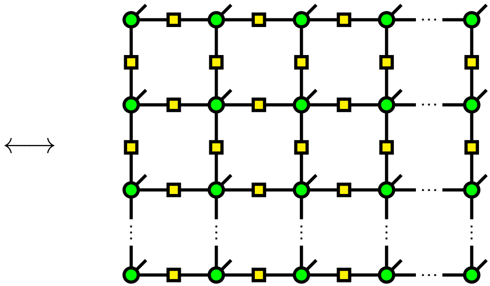
Graph states and cluster states

- Graph states are prepared by starting with many qubits in the $|+\rangle$ state and creating entanglement with controlled-Z operations:



- Since controlled-Z's commute, the only relevant part is the graph:





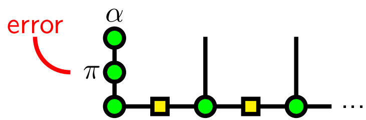
Measurements and feed-forward

- Compute with single qubit ONB measurements of this form:

Measurements and feed-forward

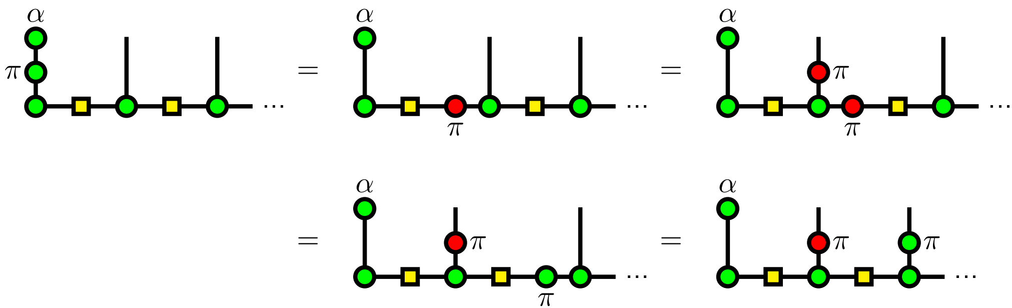
- Compute with single qubit ONB measurements of this form:

- We want to get the first outcome and treat the second outcome as an error:



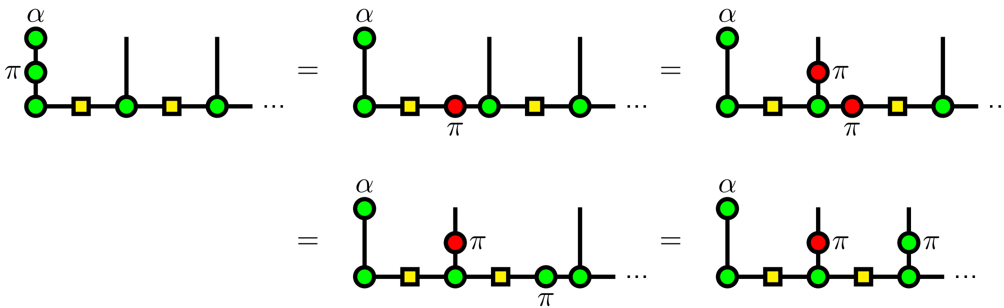
Measurements and feed-forward

- We can propagate the error out using the ZX-rules:

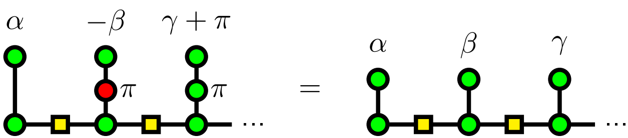


Measurements and feed-forward

- We can propagate the error out using the ZX-rules:

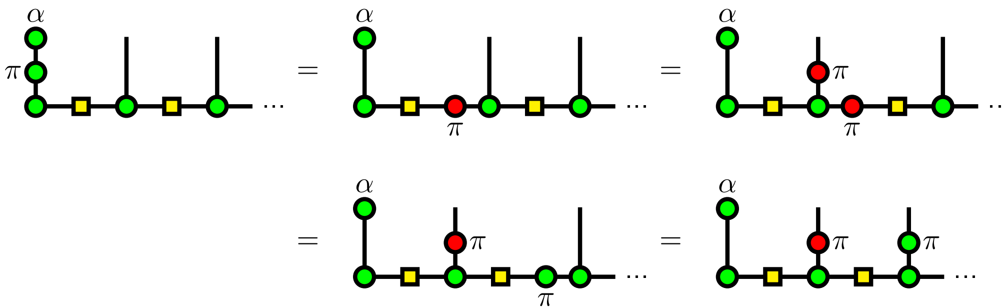


- If we know an error occurred, we can modify our later measurement choices to account for it:

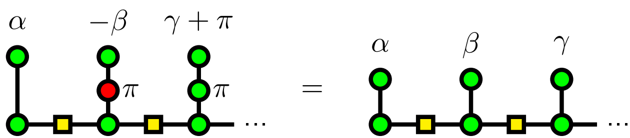


Measurements and feed-forward

- We can propagate the error out using the ZX-rules:



- If we know an error occurred, we can modify our later measurement choices to account for it:

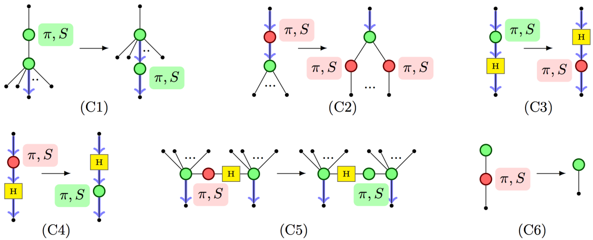


- $(\Leftarrow$ $)$

Notable results

Notable results: MBQC

- Duncan & Perdrix used the ZX-calculus to offer a new technique for transforming MBQC patterns to circuits, which has some advantages over other known methods, e.g. not requiring ancillas.[1]



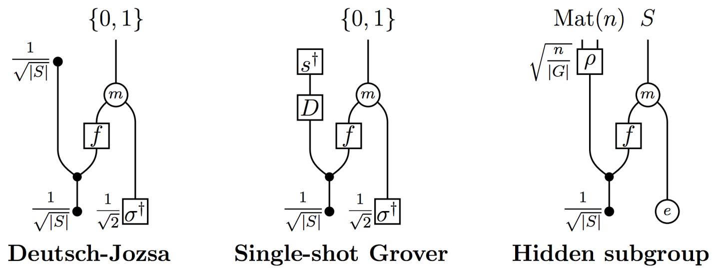
- For more details, Ducan has written a self-contained introduction to MBQC from the diagrammatic/ZX point of view, which is available on the arXiv.[2]

[1] Rewriting measurement-based quantum computations with generalised flow. R. Duncan, S. Perdrix, ICALP 2010.
personal.strath.ac.uk/ross.duncan/papers/gflow.pdf

[2] A graphical approach to measurement-based quantum computing. R. Duncan. arXiv:1203.6242

Notable results: quantum algorithms

- Vicary gave graphical characterisations of standard quantum algorithms[3]



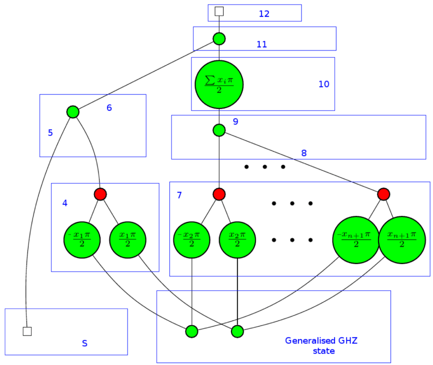
- ...a framework since used by Vicary & Zeng to develop *new* algorithms as generalisations[4]

[3] The Topology of Quantum Algorithms. LICS 2013, J. Vicary. arXiv:1209.3917

[4] Abstract structure of unitary oracles for quantum algorithms. J.Vicary, W. Zeng. arXiv:1406.1278

Notable results: quantum protocols

- Coecke, along with 3 Wangs and a Zhang give graphical proof of QKD[5]
- Hillebrand gave rewriting proofs of many ($\sim$ 25) quantum protocols.[6]
- Zamdzhiev used ZX-calculus to verify 3 kinds of quantum secret sharing.[7]



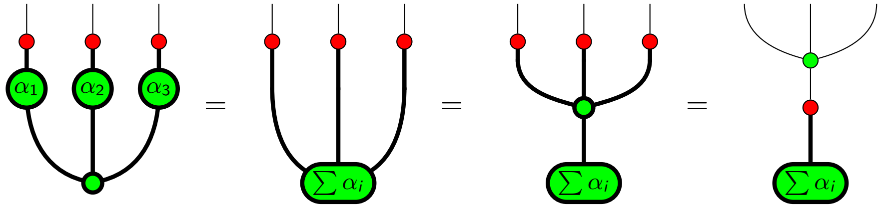
[5] Graphical Calculus for Quantum Key Distribution. B. Coecke, Q. Wang, B. Wang, Y. Wang, and Q. Zhang. QPL 2011.

[6] Quantum Protocols involving Multiparticle Entanglement and their Representations in the zx-calculus. A. Hillebrand. Masters thesis, Oxford 2011. `www.cs.ox.ac.uk/people/bob.coecke/Anne.pdf`

[7] An Abstract Approach towards Quantum Secret Sharing. Masters thesis, Oxford 2012. `www.cs.ox.ac.uk/people/bob.coecke/VladimirZamdzhievThesis.pdf`

Notable results: quantum non-locality

- AK, Coecke, Duncan, and Wang gave diagrammatic presentation of GHZ/Mermin non-locality argument[8]



- ...which has since been generalised to arbitrary dimensions and quantum-like theories[9]

[8] Strong Complementarity and Non-locality in Categorical Quantum Mechanics. B. Coecke, R. Duncan, A. Kissinger, Q. Wang. LICS 2012.

[9] Mermin Non-Locality in Abstract Process Theories. QPL 2015

Where do we go from here?

- Completeness (Clifford + T, full)

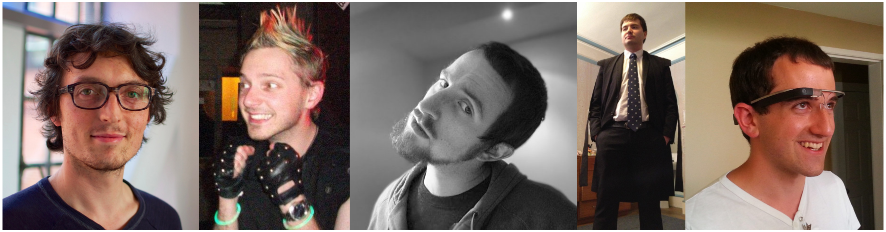
Where do we go from here?

- Completeness (Clifford + T, full)
- Automation: implementation of Clifford decision procedure, theory synthesis

Where do we go from here?

- Completeness (Clifford + T, full)
- Automation: implementation of Clifford decision procedure, theory synthesis
- **Bigger algorithms**, more **sophisticated protocols**, and generally more **expressiveness** of the diagrammatic language

Thanks!



- Quantomatic is joint work with Lucas Dixon, Alex Merry, Ross Duncan, Vladimir Zamdzhiev, and David Quick
- See: `quantomatic.github.io`