Picturing Quantum Processes II: ZX-calculus and automation

Aleks Kissinger

Radboud University

Coalgebraic Methods in Computer Science
Thessalonaki 2018
The setting: symmetric monoidal categories

\[ f : A \rightarrow B := \begin{array}{c} B \\ \downarrow \\ A \end{array} \]

\[ g \circ f := \begin{array}{c} g \\ \downarrow \\ f \end{array} \]

\[ f \otimes g := \begin{array}{c} g \\ \downarrow \\ f \end{array} \]

\[ 1_A := \begin{array}{c} A \\ \end{array} \]

\[ 1_I := \begin{array}{c} I \\ \end{array} \]

\[ \sigma_{A,B} := \begin{array}{c} B \\ \downarrow \\ A \\ \downarrow \\ A \\ \downarrow \\ B \end{array} \]
Symmetric monoidal theories

- We work with (co)algebraic structures inside an SMC, via \textit{symmetric monoidal theories}.

- Symmetric monoidal theories generalise (universal) algebraic theories.

- They consist of a set of generators with input/output arities e.g.

  \[
  \Sigma := \left\{ \begin{array}{c}
  \bullet : 2 \rightarrow 1, \\
  \circ : 0 \rightarrow 1
  \end{array} \right\}
  \]

- And a set of \textit{relations}, which are pairs of (formal) compositions in the SMC, respecting arities, e.g.

  \[
  E := \left\{ \begin{array}{c}
  \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array} = \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array}, \\
  \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array} = \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array}, \\
  \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array} = \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array}, \\
  \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array} = \begin{array}{c}
  \begin{array}{c}
  \bullet \\
  \circ
  \end{array}
  \end{array}
  \end{array} \right\}
  \]
Symmetric monoidal theories

- Relations \((L = R) \in E\) extend to larger diagrams via substitution.
- In the usual (term-like) picture, we can see this is rewriting modulo the SMC axioms:

\[
L = R \vdash G = H \iff \exists C_1, C_2, X. \begin{cases} 
G \overset{\text{smc}}{=} C_1 \circ (L \otimes 1_X) \circ C_2 \\
H \overset{\text{smc}}{=} C_1 \circ (R \otimes 1_X) \circ C_2
\end{cases}
\]

- Rewriting modulo is hard in general.
- Simpler to perform substitution \textit{directly on string diagrams}. 

Aleks Kissinger

PQP 2

CMCS 2018
Equational reasoning with diagram substitution

• For example:

• and be applied as:
A model of \((\Sigma, E)\) consists of an object \(A \in \mathcal{C}\) and morphisms:

\[
[f] : \underbrace{A \otimes \ldots \otimes A}_m \rightarrow \underbrace{A \otimes \ldots \otimes A}_n
\]

for all \(f : m \rightarrow n \in \Sigma\), such that compositions satisfy the equations in \(E\).

- Equivalently, we can define models of \((\Sigma, E)\) in terms of its associated PROP.

- A PROP is an SMC whose objects are \(\mathbb{N} := (co)\text{arities}\).

- The syntactic PROP \(\text{Syn}(\Sigma, E)\) of a theory \((\Sigma, E)\) has as morphisms string diagrams of \(\Sigma\)-generators, modulo \(E\).

- Models are strong monoidal functors \([−] : \text{Syn}(\Sigma, E) \rightarrow \mathcal{C}\).
Semantic PROPs

- Equality in $\text{Syn}(\Sigma, E)$ is undecidable in general
- However, some theories have particularly nice, ‘semantic’ PROPs with decidable equality.
- e.g. for commutative monoids, $\text{Syn}(\Sigma, E) \simeq (\text{FinSet}, +)$
Example

(Special commutative) Frobenius algebras:

\[
\begin{align*}
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example1.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example2.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example3.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example4.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example5.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example6.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example7.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example8.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example9.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example10.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example11.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example12.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example13.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example14.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example15.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example16.png}}} \\
\vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example17.png}}} & = \vcenter{\hbox{\includegraphics[width=0.2\textwidth]{example18.png}}}
\end{align*}
\]

\text{n.b.} \; \text{Syn}(\Sigma, E) \simeq (\text{Csp}(\text{FinSet}), +)
Example

Commutative bialgebras:

\[
\begin{align*}
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} &=
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} \\
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} &=
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} \\
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} &=
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} \\
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array} &=
\begin{array}{c}
\begin{tikzpicture}
  \node[circle,draw] (1) at (0,0) {};
  \node[circle,draw] (2) at (1,0) {};
  \node[circle,draw] (3) at (2,0) {};
  \node[circle,draw] (4) at (1,1) {};
  \draw[thick] (1) -- (2);
  \draw[thick] (2) -- (3);
  \draw[thick] (1) -- (4);
\end{tikzpicture}
\end{array}
\end{align*}
\]

n.b. \(\text{Syn}(\Sigma, E) \simeq (\text{Mat}[\mathbb{N}], \oplus)\)
The system $\mathbb{IB}$

- It is interesting to consider multiple, interacting theories
- $\mathbb{IB}$ (for interacting bialgebras) consists of two bialgebras that interact with each other as Frobenius algebras
- equivalently, it’s two Frobenius algebras that interact as bialgebras
- Formally, it consists of:

$$\Sigma_{\mathbb{IB}} = \left\{ \begin{array}{c}
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown, \bigcirc, \bigcirc, \bigcirc, \bigcirc
\end{array}
\end{array} \right\}$$

such that these are Frobenius algebras:

$$\left( \begin{array}{c}
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown
\end{array}
\end{array} \right) \quad \left( \begin{array}{c}
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown
\end{array}
\end{array} \right)$$

these are bialgebras:

$$\left( \begin{array}{c}
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown
\end{array}
\end{array} \right) \quad \left( \begin{array}{c}
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown
\end{array}
\end{array} \right)$$

and caps/cups coincide:

$$\begin{array}{c}
\Upsilon := \quad \Upsilon := \\
\begin{array}{c}
\circlearrowright, \dagger, \bigtriangledown, \bigtriangledown
\end{array}
\end{array}$$
Interacting bialgebras are linear relations

\[ \text{Syn}(\Sigma_{\mathbb{IB}}, E_{\mathbb{IB}}) \cong \text{LinRel}_{\mathbb{Z}_2} \quad [\text{BSZ}'14]^1 \]

- LinRel\(_{\mathbb{Z}_2}\) has:
  - **objects**: \(\mathbb{N}\)
  - **morphisms**: \(R : m \to n\) is a subspace:
    \[ R \subseteq \mathbb{Z}_2^m \times \mathbb{Z}_2^n \cong \mathbb{Z}_2^{m+n} \]
  - **composition** is relation-style. For \(R : m \to n, S : n \to k\):
    \[ (u|w) \in S \circ R \iff \exists v \in \mathbb{Z}_2^n. (u|v) \in R \land (v|w) \in S \]
  - **tensor** is \(\times\) (aka. \(\oplus\))

---

\(^1\)F. Bonchi, P. Sobocinski, F. Zanasi. *Interacting bialgebras are Frobenius*
Semantic picture

- We can define an interpretation $\llbracket - \rrbracket : \text{Syn}(\SigmaIB, EIB) \rightarrow \text{LinRel}_{\mathbb{Z}_2}$ as:

$$
\llbracket \begin{array}{c}
\circlearrowleft \\
\circlearrowright
\end{array} \rrbracket = \left\{ \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right\} \subseteq \mathbb{Z}_2^{1+2} \\
\llbracket \begin{array}{c}
\bigcirc \\
\otimes
\end{array} \rrbracket = \left\{ (0), (1) \right\} \subseteq \mathbb{Z}_2^{1+0}
$$

$$
\llbracket \begin{array}{c}
\downarrow \\
\uparrow
\end{array} \rrbracket = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \right\} \subseteq \mathbb{Z}_2^{2+1} \\
\llbracket \begin{array}{c}
\bigcirc
\end{array} \rrbracket = \left\{ (0) \right\} \subseteq \mathbb{Z}_2^{0+1}
$$

- BSZ showed that $\llbracket - \rrbracket$ extends to a PROP iso
- They used the technique of *composing PROPs* via distributive laws
Syntactic picture

- Can also show this syntactically, by string diagram rewriting
- For this, it's useful to switch to *unbiased* presentations of Frobenius algebras and bialgebras, via *spiders*:

\[ \begin{array}{c}
\quad \vdash \quad \\
\quad \vdash \quad \\
\end{array} \]
Unbiased Frobenius algebras

All Frobenius equations are subsumed by ‘spider fusion’:

\[ \ldots \quad \ldots \quad = \quad \ldots \]
Unbiased bialgebras

All bialgebra laws are subsumed by replacing two connected spiders by a complete bipartite graph:

\[
\begin{array}{c}
\ldots \\
\ldots \\
\end{array}
\quad =
\begin{array}{c}
\ldots \\
\ldots \\
\ldots \\
\ldots \\
\end{array}
\]

The three basic laws are special cases:

\[
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
\end{array}
\end{array}
= 
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
\end{array}
\]
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
\end{array}
= 
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
\]
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
= 
\begin{array}{c}
\begin{array}{c}
\begin{array}{c}
\end{array}
\end{array}
\end{array}
\]
Cups and caps

Coincidence of cups and caps:

\[ \sqcup := \begin{array}{c}
\text{Diagram 1} \\
\text{Diagram 2}
\end{array} = \begin{array}{c}
\text{Diagram 3} \\
\text{Diagram 4}
\end{array} \]

...can be subsumed by treating string diagrams as undirected, i.e. we can flip wires at will:

\[ \begin{array}{c}
\text{Diagram 5} \\
\text{Diagram 6}
\end{array} = \begin{array}{c}
\text{Diagram 7} \\
\text{Diagram 8}
\end{array} \]
A simple rewriting strategy

An arbitrary IB diagram has many alternating layers of $\circ/\circ$:

![IB diagram]

**GOAL:** make just 3 layers $\circ\rightarrow\circ\rightarrow\circ$.

---

\(^2\)F. Bonchi, F. Gadducci, A. Kissinger, P. Sobocinski, F. Zanasi. *Rewriting with Frobenius*
A simple rewriting strategy

**STRATEGY:** find an *interior* $\odot$-spider, apply generalised bialgebra, then fuse as much as possible.
A simple rewriting strategy

Every iteration removes at least one interior \( \bigcirc \), and doesn’t introduce any new ones, so it terminates, with just three layers, e.g.

We can read off the subspace from this pseudo-NF:

- \( \bigcirc \)-spiders are ‘placeholders’
- \( \bigcirc \)-spiders are ‘basis vectors’
- edges represent 1’s in the basis vectors at a given place.
Pseudo-normal forms

- Subspaces can be represented as:

- The 1’s indicate where edges appear for each vector.
• Not unique! We can always add or remove a vector that is the sum of two other spanning vectors and get the same space:
The dual normal form

- We can also pass to the dual normal form (grey-white-grey), using the colour-reversed strategy.
- This describes the subspace dually, as a system of linear equations:

\[
\begin{align*}
x_1 &= y_1 \oplus y_2 \\
x_1 \oplus x_2 &= y_2 \\
x_3 &= 0
\end{align*}
\]
We would like to *automated* simplification for $\mathbb{IB}$:

...by turning equations $L = R$ into (directed) rules $L \Rightarrow R$. 
but we have a problem

- (Biased) AC rules are not terminating:

\[
\begin{align*}
\begin{array}{c}
\text{input:} \\
\text{output:}
\end{array}
\end{align*}
\]

- **Solution:** use *unbiased* simplifications, like spider-fusion:

\[
\begin{align*}
\begin{array}{c}
\text{input:} \\
\text{output:}
\end{array}
\end{align*}
\]

- need infinitely many rules, or *rule schemas*
!-boxes: simple diagram schemas

![Diagram with ![boxes and diagrams]](image-url)

\[
\begin{bmatrix}
\text{...}
\end{bmatrix} = \{\text{circles, squares, triangles, ...}\}
!-boxes: simple diagram rule schemas

\[ \ldots \quad = \quad \ldots \quad \Rightarrow \quad \begin{array}{c} \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \end{array} \quad = \quad \begin{array}{c} \begin{array}{c} \end{array} \\ \begin{array}{c} \end{array} \end{array} \]
!-boxes

\[
\begin{align*}
\begin{array}{c}
\text{blue} \\
\text{red}
\end{array}
\end{align*}
\]
Unbiased rules with !-boxes
Unbiased rules with !-boxes
Unbiased rules with !-boxes
A !-box presentation of $\mathbb{IB}$

30 rules $\sim$ 7 rules
A !-box presentation of $\mathbb{IB}$

Time to fire up Quantomatic.
Quantum computation: the circuit model

- Focus on:
  \[ U : \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_m \rightarrow \underbrace{\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2}_n \]

  where \( U \) is a *unitary* linear map \( (U^\dagger = U^{-1}) \) in \( \text{Vect}_\mathbb{C} \).

- We can decompose \( U \) into smaller unitaries, called *gates*, which we know how to implement on a quantum computer.
The ‘quantum trick’: unitary oracles

- We have:

\[
\mathbb{C}^2 \otimes \ldots \otimes \mathbb{C}^2 \cong \mathbb{C}^{2^n}
\]

- So, we fix a basis of bitstrings called the **computational basis**:

\[
\{|00..0\rangle, |0..01\rangle, \ldots, |11..1\rangle\} \subseteq \mathbb{C}^{2^n}
\]

- This lets us encode classical functions \( F : \{0, 1\}^N \rightarrow \{0, 1\} \) as linear maps:

\[
f(|b_1..b_N\rangle) := |F(b_1, .., b_N)\rangle
\]
The ‘quantum trick’: unitary oracles

- The $\mathbb{1b}$ generators have an interpretation into $(\text{Vect}_\mathbb{C}, \otimes)$
- ...which we can use make the linear map $f$ into a unitary with one weird trick:

\[
U_f := \ldots \quad \begin{array}{c}
\text{f}
\end{array}
\]

which is called the quantum oracle of $f$. 
Q: How much does an oracle know?

- If we plug in the right state, quite a bit!

- By a good choice of measurements, we can extract **global properties** of $f$.
- Main trick behind Grover search, Shor’s factoring algorithm, etc.
Efficient classical simulation

• The 'quantum hardness' is in $U : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^n}$.

• Some $U$ can be represented/computed efficiently, depending on the choice of gates.

• Simplest non-trivial 2-qubit gate: CNOT

$$\begin{align*}
|00\rangle &\mapsto |00\rangle \\
|01\rangle &\mapsto |01\rangle \\
|10\rangle &\mapsto |11\rangle \\
|11\rangle &\mapsto |10\rangle
\end{align*}$$

• The $\mathbb{IB}$ generators have a model in $(\text{Vect}_\mathbb{C}, \otimes)$ where:

$$\text{CNOT} = \begin{array}{c}
\text{CNOT} \\
\end{array} = \begin{array}{c}
\text{CNOT} \\
\end{array} = \begin{array}{c}
\text{CNOT} \\
\end{array}$$

• CNOT circuits are efficiently classically simulable.
Adding single-qubit gates

- A single system is a *qubit*, which can be pictured on a sphere:

\[
\psi = \begin{pmatrix} c \\ d \end{pmatrix} \propto \left( \cos \left( \frac{\theta}{2} \right) \right) \left( e^{i\alpha} \sin \left( \frac{\theta}{2} \right) \right)
\]

- Unitaries on single qubits ↔ rotations of the sphere
Adding single-qubit gates

• Adding NOT (180° around X-axis) still gives efficient classical simulation

\[ \text{LinRel}_{\mathbb{Z}_2} \sim \text{AffRel}_{\mathbb{Z}_2} \]

• More interesting/quantum: add all rotations preserving this octahedron:

• Gives interesting quantum behaviour (quantum uncertainty/complemetarity, non-locality, ...)

• But still classically simulable (by Gottesman-Knill theorem)
A complete set of gate identities

- Octahedron rotations are 2-generated (call generators $H$ and $S$). Adding to CNOT gives *Clifford circuits*.
- The following is a complete set of equations of Clifford circuits:

\[
\begin{align*}
\omega^8 &= 1 \\
H^2 &= 1 \\
S^4 &= 1 \\
SHSHSH &= \omega \\
\text{CNOT} &= \text{CNOT} \\
S^2 &= S \\
S^2 &= S \\
HSHS &= HSHS \\
HSHS &= HSHS
\end{align*}
\]

(Selinger 2013)
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 \textit{extremely} complicated (> 100 pages long! though mostly machine-generated)

- Can we do better by extending \( \boxplus \)?
ZX-calculus, presentation 1

Generators:

$$\Sigma_{IB} + \{S, H\}$$

Equations:

$$H^2 = H = S^3 = (S^2)^4 = |$$
**ZX-calculus, unbiased presentation**

Generators:

These are related to the other generators by:

\[
\frac{\pi}{2} = S \\
\frac{\pi}{2} = S H
\]
ZX-calculus, presentation 2
Completeness

Theorem (Backens’10)

The ZX-calculus is complete for Clifford quantum computation.
T gates and universality

• Adding one more generator:

\[
\begin{array}{c}
T \\
\frac{\pi}{4}
\end{array}
\]

gives us (approximately) everything.

• For any unitary \( U \), we can find \( U' \) built with our generators such that for any \( \epsilon > 0 \), we have:

\[
U \approx U'
\]
Completeness, take 2

Theorem (JPV'17\textsuperscript{3})

The ZX-calculus is complete for Clifford+$T$ quantum computation.

The rules are ZX + 3 more:

---

\textsuperscript{3}Jeandel, Perdrix, and Vilmart. A Complete Axiomatisation of the ZX-Calculus for Clifford+$T$

Quantum Mechanics
...and even more completeness

<table>
<thead>
<tr>
<th>Calculus</th>
<th>Family</th>
<th>Num. Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backens’10</td>
<td>ZX</td>
<td>Clifford</td>
</tr>
<tr>
<td>Backens’14</td>
<td>ZX</td>
<td>1-qubit Clifford+T</td>
</tr>
<tr>
<td>Hadzihasanavic’15</td>
<td>ZW</td>
<td>Z-matrices</td>
</tr>
<tr>
<td>JPV’17</td>
<td>Y</td>
<td>CNOT + (Y(\pi/2))</td>
</tr>
<tr>
<td>JPV’17</td>
<td>ZX+</td>
<td>Clifford+T</td>
</tr>
<tr>
<td>Wang &amp; Ng’17</td>
<td>ZX+</td>
<td>ALL</td>
</tr>
<tr>
<td>JPV’18</td>
<td>ZX+</td>
<td>ALL</td>
</tr>
<tr>
<td>Wang &amp; Ng’18</td>
<td>ZX+</td>
<td>2-qubit Clifford+T</td>
</tr>
<tr>
<td>AK &amp; Backens’18</td>
<td>ZH</td>
<td>ALL</td>
</tr>
</tbody>
</table>
TODO NOW:

Completeness theorems $\implies$ (efficient) simplification

Application of techniques beyond QT$^4$

\footnote{Signal flow diagrams, classical circuits, Petri nets, ...}
Thanks for your attention

http://quantomatic.github.io

http://cambridge.org/pqp