Quantum Compilation using the ZX-calculus

John van de Wetering
Radboud University Nijmegen
Oxford University

June 22, 2021

The ZX-calculus

The ZX-calculus is a graphical language for quantum computation developed by Bob Coecke and Ross Duncan in 2007.

The ZX-calculus

The ZX-calculus is a graphical language for quantum computation developed by Bob Coecke and Ross Duncan in 2007.

Used in:

- Quantum circuit optimisation and compilation
- Measurement-based quantum computation
- Surface codes and lattice surgery
- ...

The ZX-calculus

The ZX-calculus is a graphical language for quantum computation developed by Bob Coecke and Ross Duncan in 2007.

Used in:

- Quantum circuit optimisation and compilation
- Measurement-based quantum computation
- Surface codes and lattice surgery
- ...

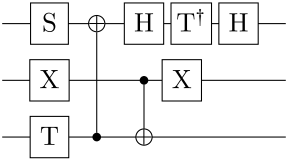
It is also a convenient tool for day-to-day quantum reasoning

Further Reading

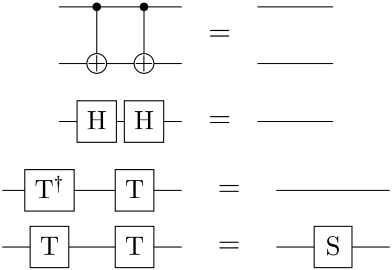
For references and details see:
ZX-calculus for the working quantum computer scientist
`https://arxiv.org/abs/2012.13966`

For a book-length introduction see:
Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning
by Bob Coecke and Aleks Kissinger

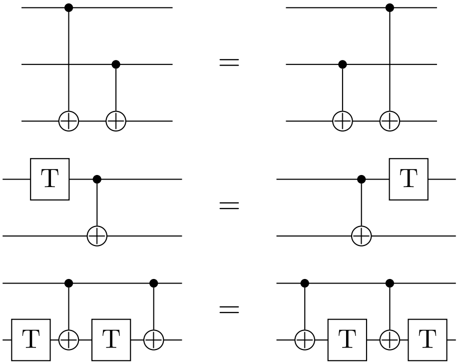
Quantum circuits



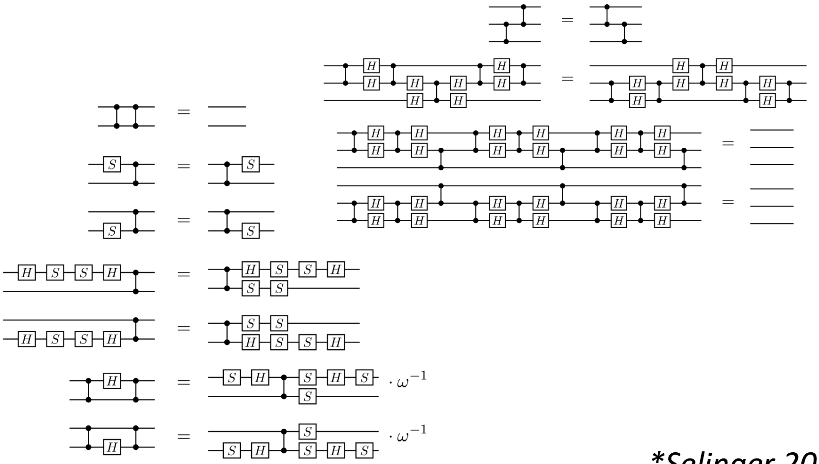
Circuit identities



Gate commutation

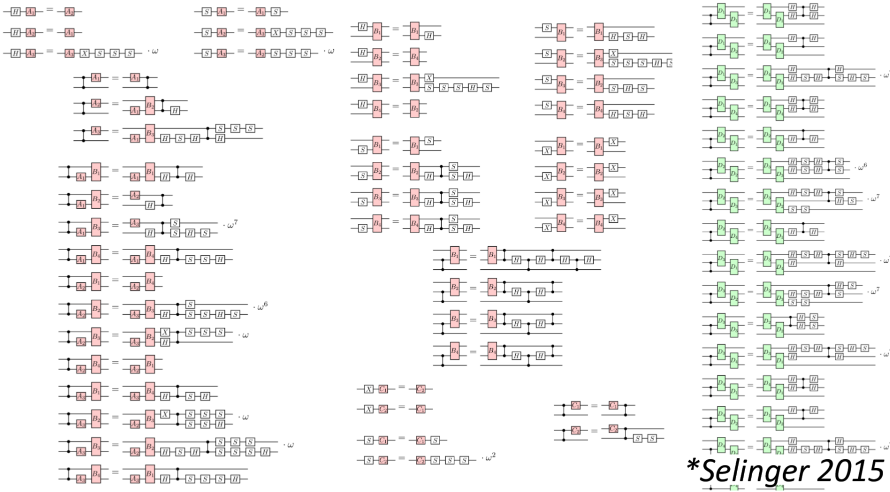


More circuit equalities



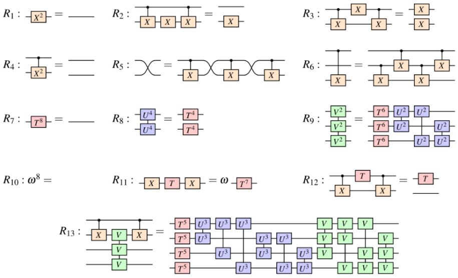
**Selinger 2015*

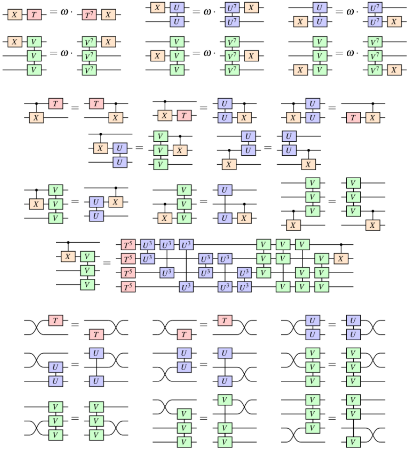
And more circuit equalities



*Selinger 2015

And even more circuit equalities





*Amy, Chen, & Ross 2018

Quantum circuits bad!

Why is this so terrible?

- Choice of gates is a bit arbitrary.
- The notation is not “quantum native”.
- Wires are rigid going from left-to-right.

Quantum circuits bad!

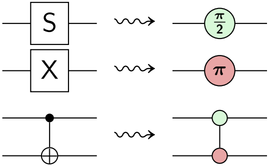
Why is this so terrible?

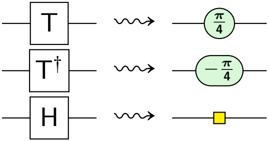
- Choice of gates is a bit arbitrary.
- The notation is not “quantum native”.
- Wires are rigid going from left-to-right.

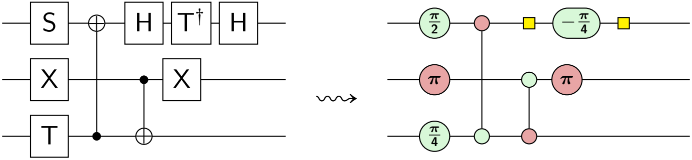
The ZX-calculus essentially gets rid of these problems.

ZX-diagrams

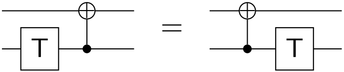
On a surface level, ZX-diagrams are alternative notation to circuits



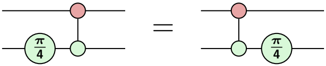




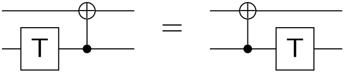
Circuit identity in ZX



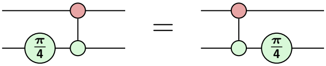
becomes



Circuit identity in ZX

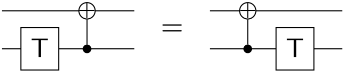


becomes

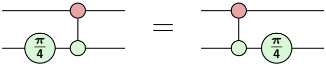


dots of same colour commute through each other

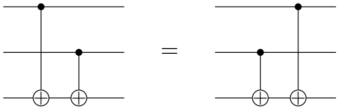
Circuit identity in ZX

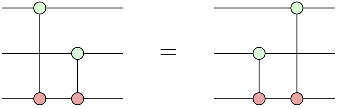


becomes

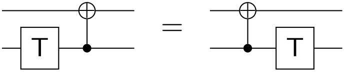


dots of same colour commute through each other

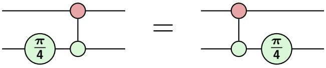




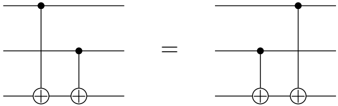
Circuit identity in ZX

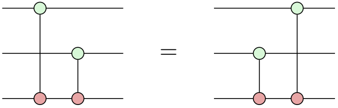


becomes

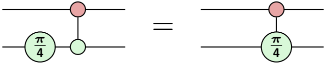


dots of same colour commute through each other





More fundamental rule: *dots of same colour fuse*

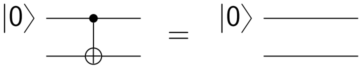


States in ZX

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

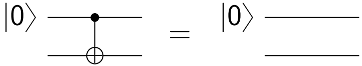
States in ZX

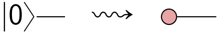
$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \qquad |+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$$



States in ZX

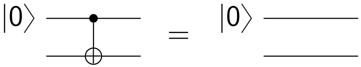
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

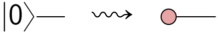




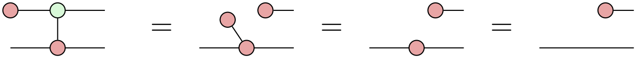
States in ZX

$$|0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \qquad |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \qquad |+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$$



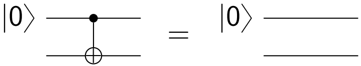


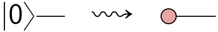
single-wire dot copies through opposite-coloured dot



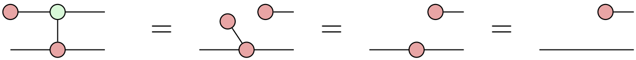
States in ZX

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad |+\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad |-\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

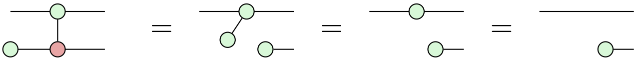




single-wire dot copies through opposite-coloured dot



all rules hold with colours interchanged



Now let's formally introduce ZX-diagrams

Spiders

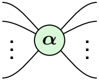
What gates are to circuits, *spiders* are to ZX-diagrams.

Spiders

What gates are to circuits, *spiders* are to ZX-diagrams.

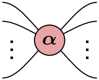
Z-spider

$$|0\cdots0\rangle\langle0\cdots0| + e^{i\alpha}\,|1\cdots1\rangle\langle1\cdots1|$$



X-spider

$$|+\cdots+\rangle\langle+\cdots+| + e^{i\alpha}\,|-\cdots-\rangle\langle-\cdots-|$$

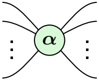


Spiders

What gates are to circuits, *spiders* are to ZX-diagrams.

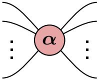
Z-spider

$$|0\cdots 0\rangle\langle 0\cdots 0| + e^{i\alpha}\,|1\cdots 1\rangle\langle 1\cdots 1|$$

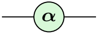


X-spider

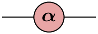
$$|+\cdots +\rangle\langle +\cdots +| + e^{i\alpha}\,|-\cdots -\rangle\langle -\cdots -|$$



For example:



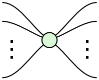
$$= |0\rangle\langle 0| + e^{i\alpha}\,|1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & e^{i\alpha} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\alpha} \end{pmatrix}$$



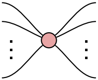
$$= |+\rangle\langle +| + e^{i\alpha}\,|-\rangle\langle -| = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} + \frac{1}{2}e^{i\alpha}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

Spiders cont.

If $\alpha = 0$ we drop the label:



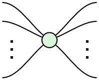
$$= \quad |0\cdots 0\rangle\langle 0\cdots 0| + |1\cdots 1\rangle\langle 1\cdots 1|$$

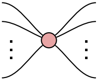


$$= \quad |+\cdots +\rangle\langle +\cdots +| + |-\cdots -\rangle\langle -\cdots -|$$

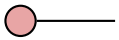
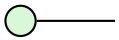
Spiders cont.

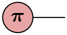
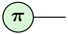
If $\alpha = 0$ we drop the label:

 $= \quad |0\cdots0\rangle\langle0\cdots0| + |1\cdots1\rangle\langle1\cdots1|$

 $= \quad |+\cdots+\rangle\langle+\cdots+| + |-\cdots-\rangle\langle-\cdots-|$

Example:

 $= \quad |+\rangle + |-\rangle \; = \sqrt{2}\,|0\rangle$  $= \quad |0\rangle + |1\rangle \; = \sqrt{2}\,|+\rangle$

 $= \quad |+\rangle - |-\rangle \; = \sqrt{2}\,|1\rangle$  $= \quad |0\rangle - |1\rangle \; = \sqrt{2}\,|-\rangle$

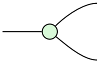
We ignore these non-zero scalar factors

Formal composition

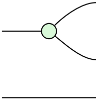
Spiders can be composed in two ways.

Formal composition

Spiders can be composed in two ways.
Horizontal composition gives tensor product:



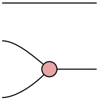
$$= \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}$$



$$= \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Formal composition

Other tensor product:

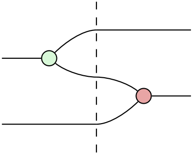


$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

Formal composition

Horizontal composition is regular composition of linear maps:

 $=$

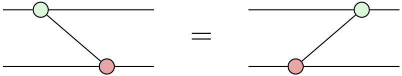
$$\frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&1&0&0&0&0\\0&1&1&0&0&0&0&0\\0&0&0&0&1&0&0&1\\0&0&0&0&0&1&1&0\end{pmatrix}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$$

Building ZX-diagrams

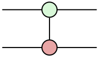
Any ZX-diagram is built by simply iterating these vertical and horizontal compositions

Symmetries

Note:

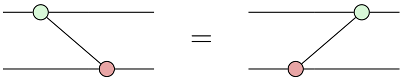


Hence, we may write

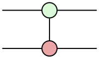


Symmetries

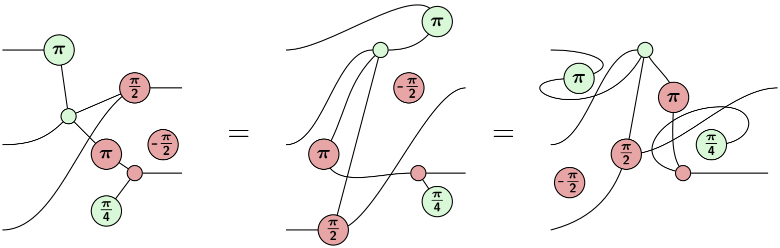
Note:



Hence, we may write



In general: *only connectivity matters*



ZX-diagrams summary

- Two types of generators: Z-spiders and X-spiders
- Can compose both horizontally and vertically
- Wires can connect every which way

ZX-diagrams summary

- Two types of generators: Z-spiders and X-spiders
- Can compose both horizontally and vertically
- Wires can connect every which way

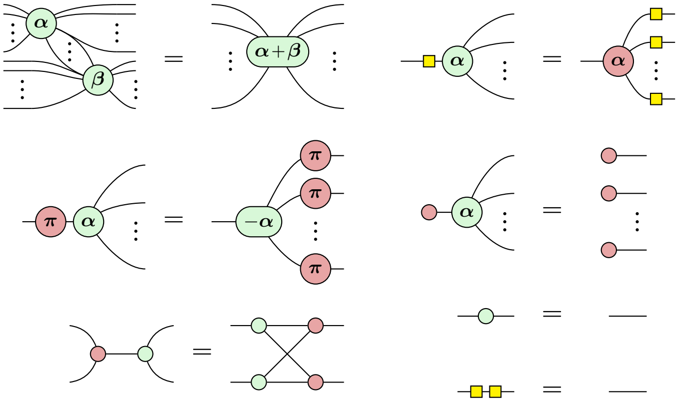
How powerful are ZX-diagrams as a representation?

Theorem

ZX-diagrams are *universal*: any linear map between qubits can be represented as a ZX-diagram.

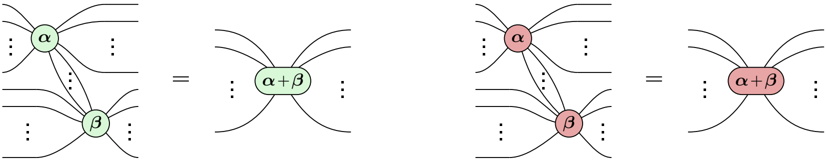
So far it's just notation. What can we do with it?

Rules for ZX-diagrams: The ZX-calculus



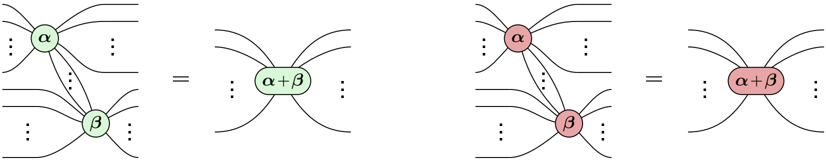
$\alpha, \beta \in [0, 2\pi]$

Spider fusion

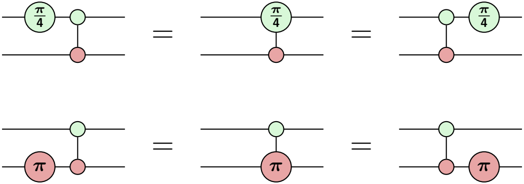


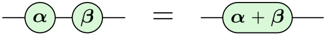
Connected spiders of same colour fuse

Spider fusion

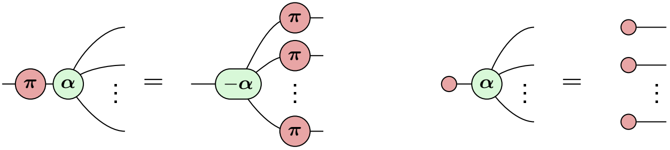


Connected spiders of same colour fuse



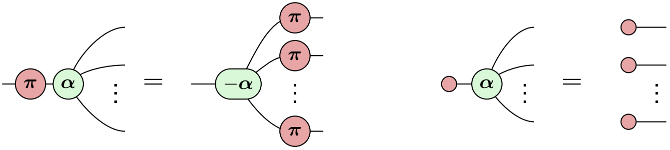


State and pi-copy



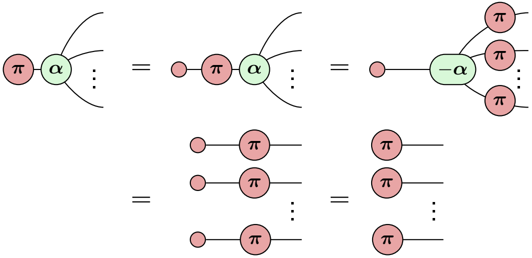
π's and states copy through the other colour

State and pi-copy



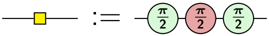
π's and states copy through the other colour

Combining rules:



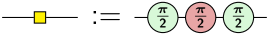
Hadamards and colour-changing

Definition of Hadamard in ZX:

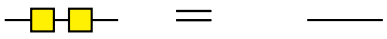


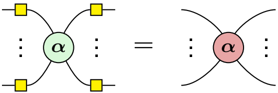
Hadamards and colour-changing

Definition of Hadamard in ZX:



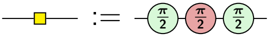
Rules:



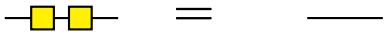


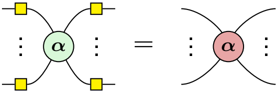
Hadamards and colour-changing

Definition of Hadamard in ZX:

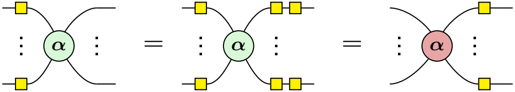


Rules:



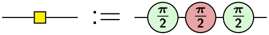


Derived rule: *commuting Hadamards changes colour*

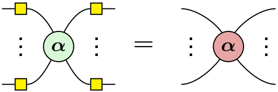


Hadamards and colour-changing

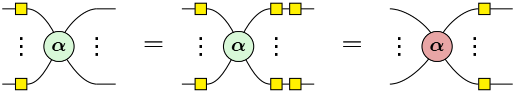
Definition of Hadamard in ZX:



Rules:



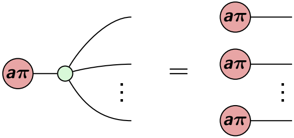
Derived rule: *commuting Hadamards changes colour*

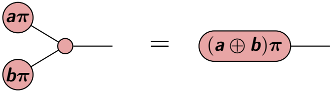


Consequence: *Everything in ZX holds with colours reversed*

Bialgebra

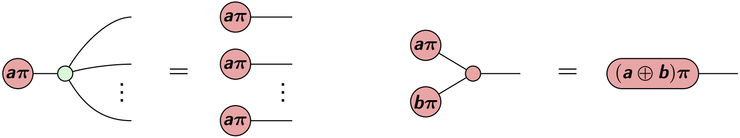
Z-spiders act like COPY; X-spiders act like XOR:



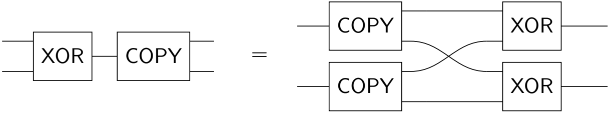


Bialgebra

Z-spiders act like COPY; X-spiders act like XOR:

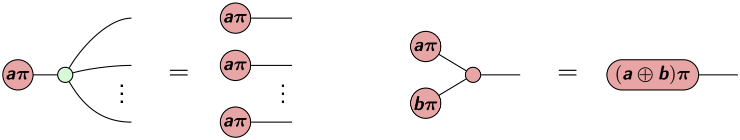


Classically we have:

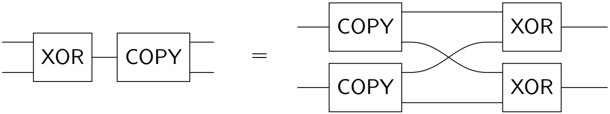


Bialgebra

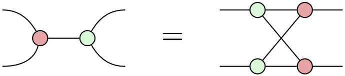
Z-spiders act like COPY; X-spiders act like XOR:



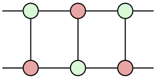
Classically we have:



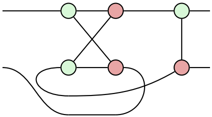
Hence:



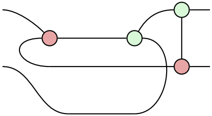
Three CNOTs make SWAP



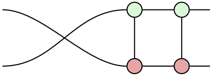
Three CNOTs make SWAP



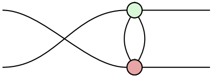
Three CNOTs make SWAP



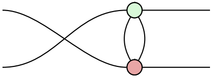
Three CNOTs make SWAP

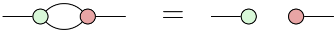


Three CNOTs make SWAP

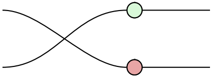


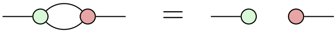
Three CNOTs make SWAP



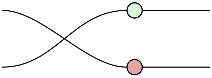


Three CNOTs make SWAP

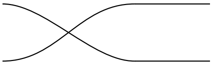




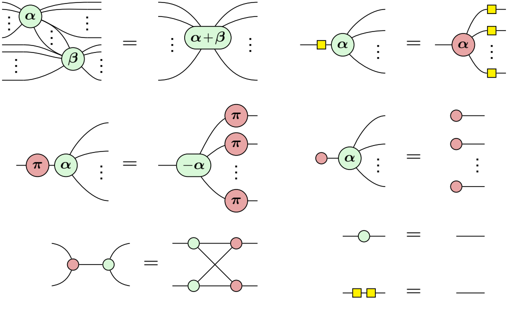
Three CNOTs make SWAP



Three CNOTs make SWAP



Rules for ZX-diagrams: The ZX-calculus

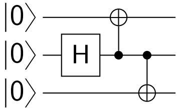


$\alpha, \beta \in [0, 2\pi]$

- All derivations hold in any orientation
- All derivations hold with colours interchanged
- All derivations hold with phases negated

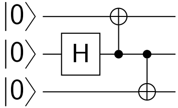
Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

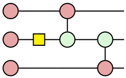


Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

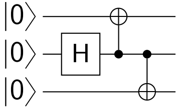


Proof:

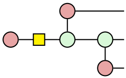


Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

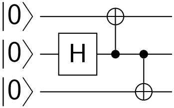


Proof:

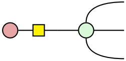


Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

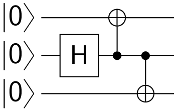


Proof:

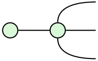


Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

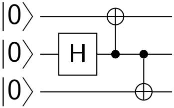


Proof:

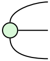


Example 1: GHZ-preparation circuit

Recall that the GHZ-state is $|000\rangle + |111\rangle$.
The following circuit creates a GHZ-state:

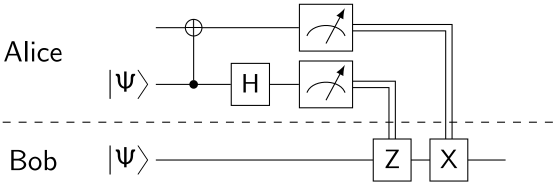


Proof:



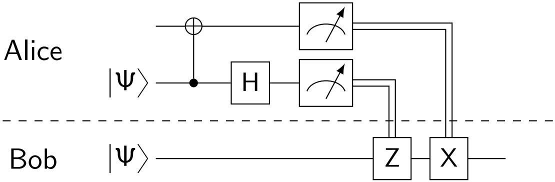
Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

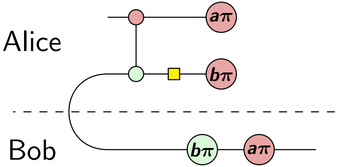


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

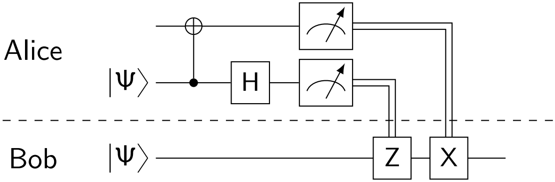


Proof:

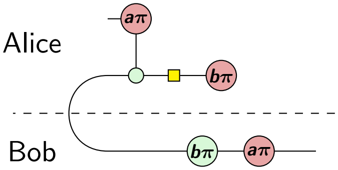


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

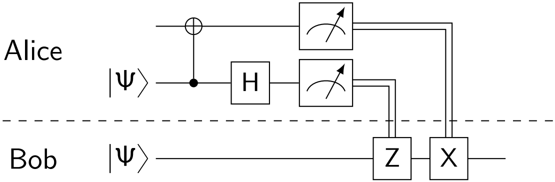


Proof:

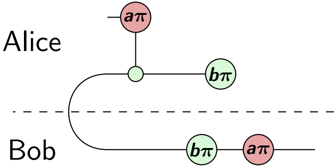


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

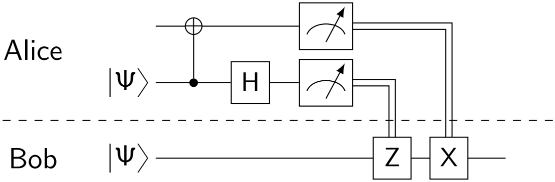


Proof:

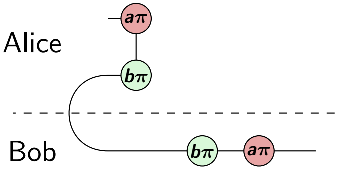


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

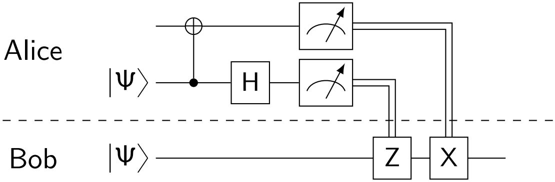


Proof:

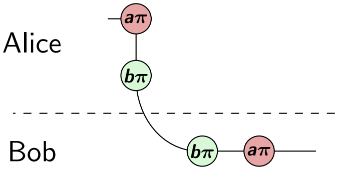


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

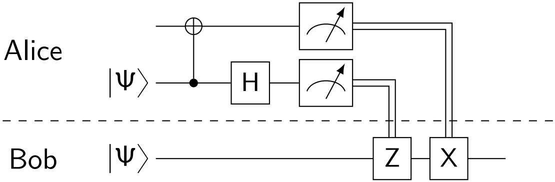


Proof:

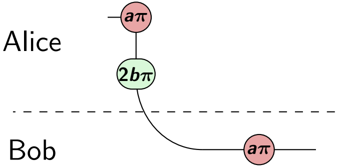


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

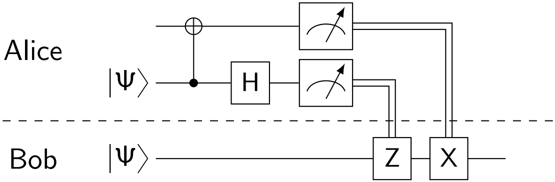


Proof:

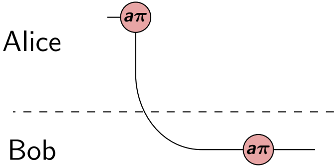


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

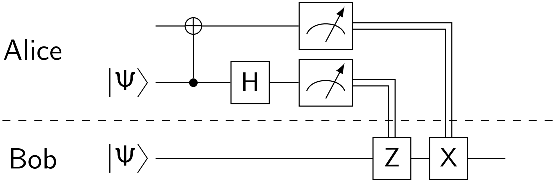


Proof:

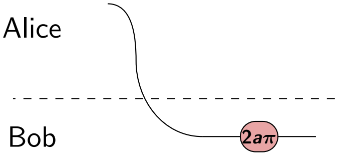


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:

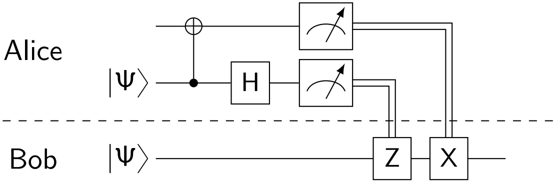


Proof:

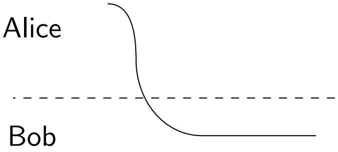


Example 2: Teleportation

Let $|\Psi\rangle$ represent a side of a Bell state.
Then this is the standard quantum teleportation protocol:



Proof:



Now let's look at specific use-cases of ZX

Use-case of ZX #1: Clifford computation

Gottesman-Knill theorem

A quantum circuit of Cliffords can be efficiently classically simulated.

Use-case of ZX #1: Clifford computation

Gottesman-Knill theorem

A quantum circuit of Cliffords can be efficiently classically simulated.

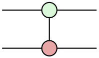
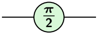
Can we prove this using ZX?

Cliffords in ZX

- A *Clifford map* is any linear map produced from combining Clifford unitaries, states and post-selections $\langle 0|$.

Cliffords in ZX

- A *Clifford map* is any linear map produced from combining Clifford unitaries, states and post-selections $\langle 0|$.
- As a ZX-diagram, a Clifford map only has phases multiple of $\frac{\pi}{2}$.

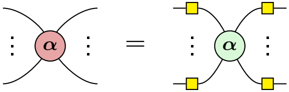
CNOT =  S = 

H = 

- Conversely, ZX-diagrams with phases multiple of $\frac{\pi}{2}$ are Clifford.

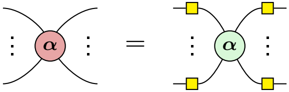
Graph-like diagrams

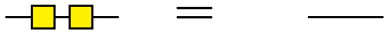
Every ZX-diagram can be reduced to a *graph-like* diagram:

- Only Z-spiders and Hadamards: 

Graph-like diagrams

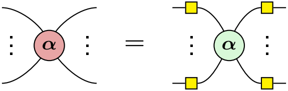
Every ZX-diagram can be reduced to a *graph-like* diagram:

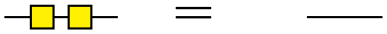
- Only Z-spiders and Hadamards: 

- Cancel adjacent hadamards: 

Graph-like diagrams

Every ZX-diagram can be reduced to a *graph-like* diagram:

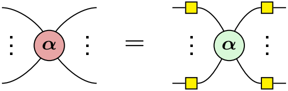
- Only Z-spiders and Hadamards: 

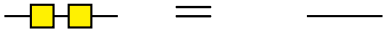
- Cancel adjacent hadamards: 

- Fuse all spiders.

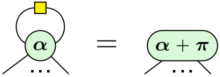
Graph-like diagrams

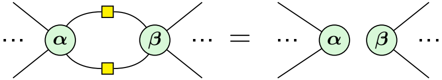
Every ZX-diagram can be reduced to a *graph-like* diagram:

- Only Z-spiders and Hadamards: 

- Cancel adjacent hadamards: 

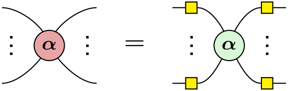
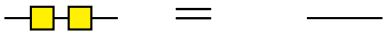
- Fuse all spiders.
- No self-loops or multiple edges:

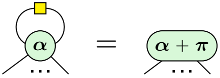


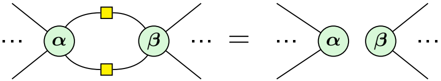


Graph-like diagrams

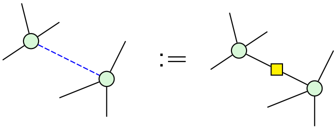
Every ZX-diagram can be reduced to a *graph-like* diagram:

- Only Z-spiders and Hadamards: 
- Cancel adjacent hadamards: 
- Fuse all spiders.
- No self-loops or multiple edges:





- View all Hadamards as a type of *edge*:

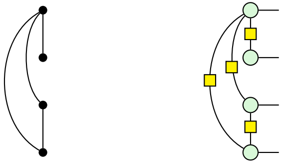


Graph states

A graph-like diagram is a *graph state* when

- it has no inputs,
- every spider is connected to a unique output,
- all phases are zero.

Example:



G $\qquad$ $|G\rangle$

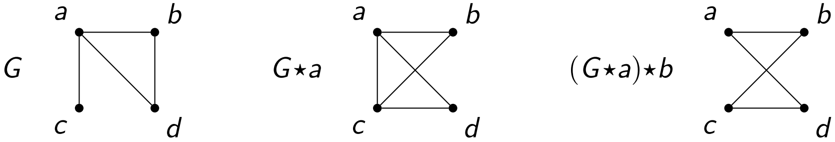
Graph-theoretic rewriting

We've transformed ZX-diagrams into simple undirected graphs so we can view rewrites graph-theoretically.

Graph-theoretic rewriting

We've transformed ZX-diagrams into simple undirected graphs so we can view rewrites graph-theoretically.

Local complementation



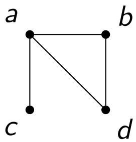
$G \star a := G$, but with connectivity of neighbours of a complemented.

Graph-theoretic rewriting

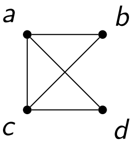
We've transformed ZX-diagrams into simple undirected graphs so we can view rewrites graph-theoretically.

Local complementation

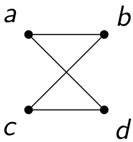
G



$G\star a$



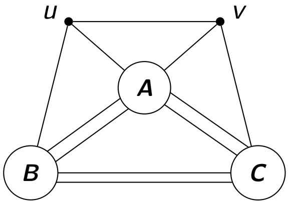
$(G\star a)\star b$



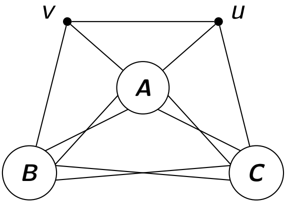
$G \star a := G$, but with connectivity of neighbours of a complemented.

A *pivot* on edge uv is $G \wedge uv := G \star u \star v \star u$.

G

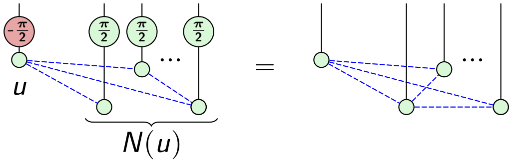


$G \wedge uv$



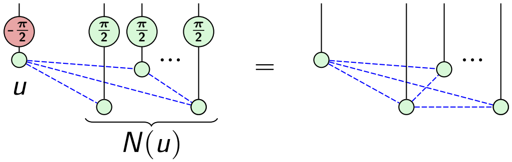
Local complementation on graph states

An lcomp on graph state can be implemented using *local Cliffords*:

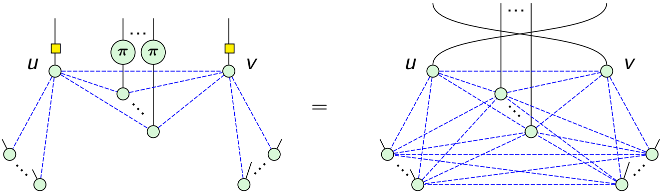


Local complementation on graph states

An lcomp on graph state can be implemented using *local Cliffords*:

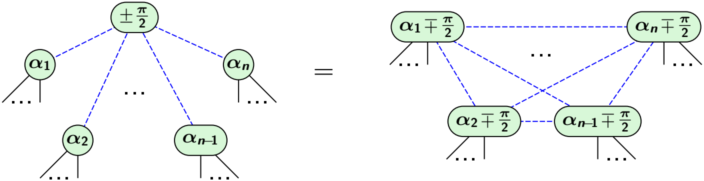


Same goes for pivot:

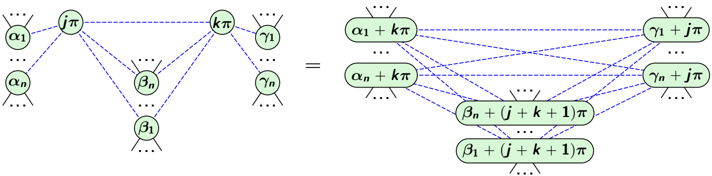


Removing vertices

Remove vertex by lcomp:



Similarly, using pivot:



Clifford simplification

- With lcomp can remove all internal vertices with $\pm\frac{\pi}{2}$ phase.
- With pivot can remove all internal vertices with 0 or π phase.

Clifford simplification

- With lcomp can remove all internal vertices with $\pm\frac{\pi}{2}$ phase.
- With pivot can remove all internal vertices with 0 or π phase.
- ...But Clifford ZX only has phases multiple of $\frac{\pi}{2}$.

Clifford simplification

- With lcomp can remove all internal vertices with $\pm\frac{\pi}{2}$ phase.
- With pivot can remove all internal vertices with 0 or π phase.
- ...But Clifford ZX only has phases multiple of $\frac{\pi}{2}$.
- So Clifford diagram without in- and outputs just disappears!

Clifford simplification

- With lcomp can remove all internal vertices with $\pm\frac{\pi}{2}$ phase.
- With pivot can remove all internal vertices with 0 or π phase.
- ...But Clifford ZX only has phases multiple of $\frac{\pi}{2}$.
- So Clifford diagram without in- and outputs just disappears!
- Hence: efficient calculation of amplitudes.

Clifford simplification

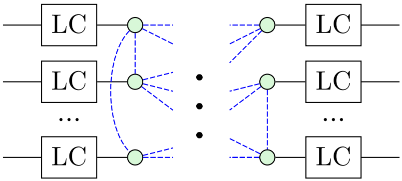
- With lcomp can remove all internal vertices with $\pm\frac{\pi}{2}$ phase.
- With pivot can remove all internal vertices with 0 or π phase.
- ...But Clifford ZX only has phases multiple of $\frac{\pi}{2}$.
- So Clifford diagram without in- and outputs just disappears!
- Hence: efficient calculation of amplitudes.

Gottesman-Knill theorem

A Clifford computation can be efficiently classically simulated.

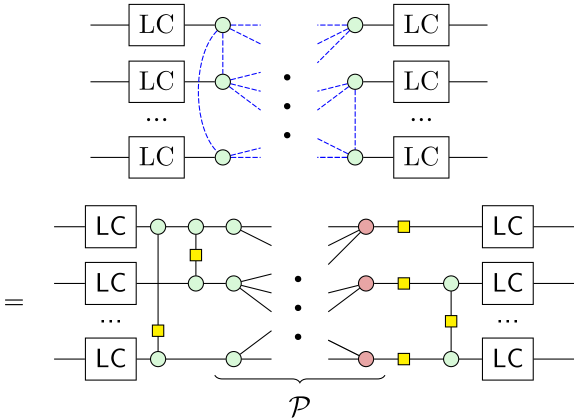
Clifford normal form

Other consequence: Clifford circuit reduced to



Clifford normal form

Other consequence: Clifford circuit reduced to

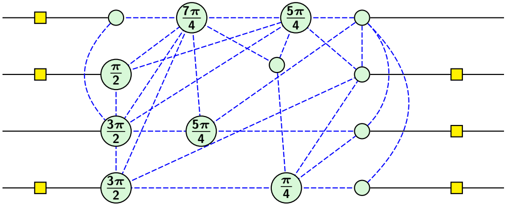


Normal form of layers:

H+S+CZ+CNOT+H+CZ+S+H

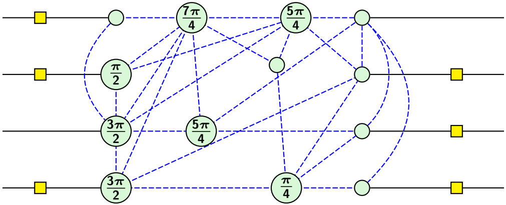
Simplifying general circuits

Example result after simplification:



Simplifying general circuits

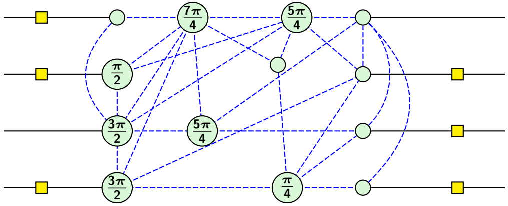
Example result after simplification:



Problem: does not look a circuit.

Simplifying general circuits

Example result after simplification:



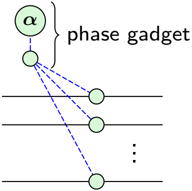
Problem: does not look a circuit.
Solution: all rewrites preserve *gflow*.

- Duncan, Perdrix, Kissinger, vdW (2019). *Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus*.
- Backens, Miller-Bakewell, de Felice, Lobski, vdW (2020). *There and back again: A circuit extraction tale*.

Non-Clifford optimisation

Non-Clifford optimisation

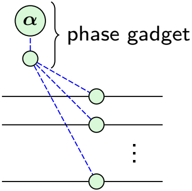
Additional rules for *phase gadgets*:



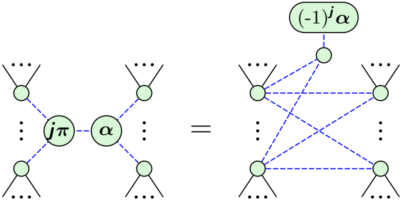
$$:: \quad \left|x_1, \ldots, x_n\right\rangle \mapsto e^{i\alpha(x_1 \oplus \ldots \oplus x_n)} \left|x_1, \ldots, x_n\right\rangle$$

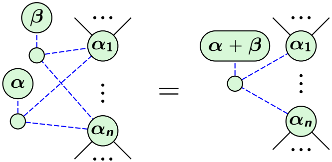
Non-Clifford optimisation

Additional rules for *phase gadgets*:



$$:: \quad |x_1, \ldots, x_n\rangle \mapsto e^{i\alpha(x_1 \oplus \ldots \oplus x_n)} |x_1, \ldots, x_n\rangle$$





Kissinger, vdW 2019: *Reducing T-count with the ZX-calculus*

T-count optimisation

- Phase gadget optimisation allows us to kill non-Clifford phases.

T-count optimisation

- Phase gadget optimisation allows us to kill non-Clifford phases.
- At time of publishing, our method improved upon previous best T-counts for 6/36 benchmark circuits — in one case by 50%.

T-count optimisation

- Phase gadget optimisation allows us to kill non-Clifford phases.
- At time of publishing, our method improved upon previous best T-counts for 6/36 benchmark circuits — in one case by 50%.
- Combining with TODD [Heyfron & Campbell 2018] we improved T-counts for 20/36 circuits.

T-count optimisation

- Phase gadget optimisation allows us to kill non-Clifford phases.
- At time of publishing, our method improved upon previous best T-counts for 6/36 benchmark circuits — in one case by 50%.
- Combining with TODD [Heyfron & Campbell 2018] we improved T-counts for 20/36 circuits.
- Note: [Zhang & Chen 2019] use a different method that achieves nearly identical T-counts.

Circuit equality verification

Can we verify correctness of optimisations?

Circuit equality verification

Can we verify correctness of optimisations?

- Compose optimised circuit with adjoint of original circuit
- Simplify
- If reduced to identity: optimisation was correct
- If not: inconclusive

Circuit equality verification

Can we verify correctness of optimisations?

- Compose optimised circuit with adjoint of original circuit
- Simplify
- If reduced to identity: optimisation was correct
- If not: inconclusive

Using this method found mistake in other peer-reviewed optimiser.

Classical Quantum circuit optimisation

Two ways for ZX to classically simulate circuits:

- Treat ZX-diagram as tensor network and contract.

Classical Quantum circuit optimisation

Two ways for ZX to classically simulate circuits:

- Treat ZX-diagram as tensor network and contract.
- Use stabiliser decomposition of magic states to write as sum of simpler diagrams.

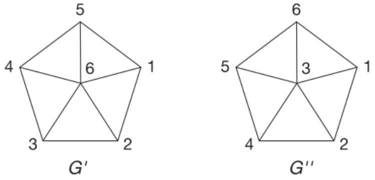


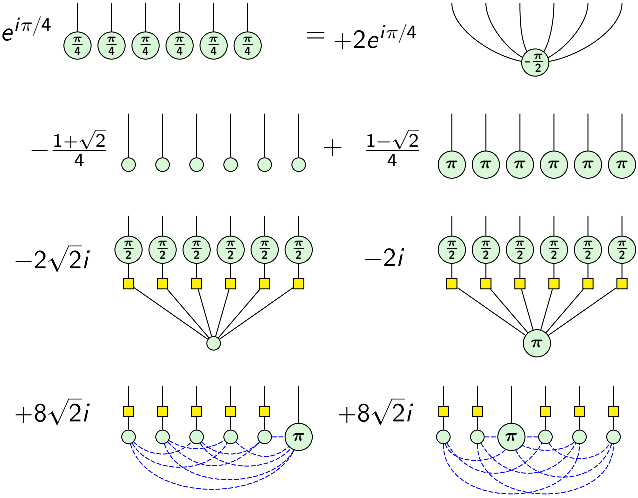
FIG. 3. Graphs G' and G'' used in the definition of stabilizer states ϕ' and ϕ''; see Eq. (11).

$$
\begin{aligned}
|H^{\otimes 6}\rangle = (-16 + 12\sqrt{2})|B_{6,0}\rangle + (96 - 68\sqrt{2})|B_{6,6}\rangle \\
+ (10 - 7\sqrt{2})|E_6\rangle + (-14 + 10\sqrt{2})|O_6\rangle \\
+ (7 - 5\sqrt{2})Z^{\otimes 6}|K_6\rangle + (10 - 7\sqrt{2})|\phi'\rangle \\
+ (10 - 7\sqrt{2})|\phi''\rangle,
\end{aligned} \tag{11}
$$

where

$$
|\phi'\rangle = \prod_{(i,j)\in E'} \Lambda(Z)_{i,j}|O_6\rangle \quad \text{and} \quad |\phi''\rangle = \prod_{(i,j)\in E''} \Lambda(Z)_{i,j}|O_6\rangle.
$$

Source: Sergey Bravyi, Graeme Smith, and John A Smolin. *Trading classical and quantum computational resources* (2016).


$e^{i\pi/4}$
$\frac{\pi}{4}$ $\frac{\pi}{4}$ $\frac{\pi}{4}$ $\frac{\pi}{4}$ $\frac{\pi}{4}$ $\frac{\pi}{4}$
$= +2e^{i\pi/4}$
$-\frac{\pi}{2}$
$-\frac{1+\sqrt{2}}{4}$
$+$
$\frac{1-\sqrt{2}}{4}$
π π π π π π
$-2\sqrt{2}i$
$\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$
$-2i$
$\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$ $\frac{\pi}{2}$
π
$+8\sqrt{2}i$
π
$+8\sqrt{2}i$
π

Circuit simulation with ZX-calculus

1. Write circuit+state as ZX-diagram.
2. Simplify using ZX-calculus rules.
3. Replace magic states by stabilizer decomposition.
4. Repeat.
5. ...
6. Profit!

Early results looks like this could give major benefit

Stuff I didn't talk about

- CNOT optimisation
- Relationship to MBQC and lattice surgery
- Circuit routing
- Applications in tensor networks

Conclusion

- ZX-calculus is a better representation of quantum circuits
- It allows you to graphically do many things

Conclusion

- ZX-calculus is a better representation of quantum circuits
- It allows you to graphically do many things

Thank you for your attention

vdW 2020, arXiv:2012.13966.
ZX-calculus for the working quantum computer scientist