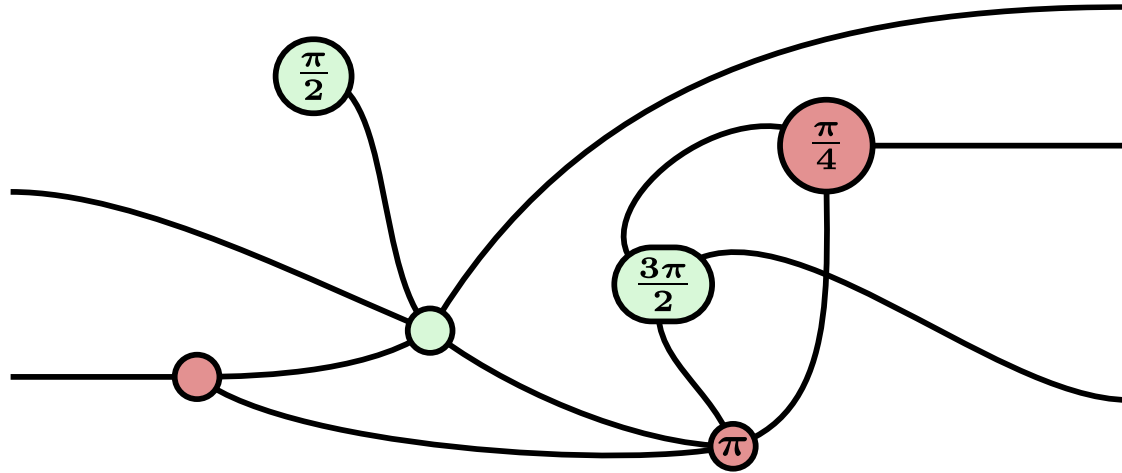


ZX-Calculus



Manuel Lerchner

20.06.2023

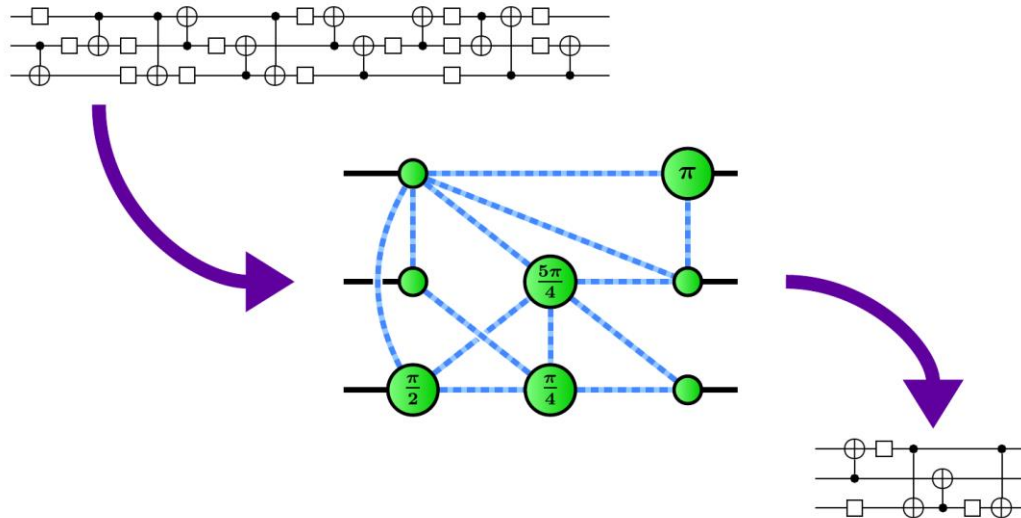
What is ZX-Calculus?

- A graphical language extending circuit diagrams
 - Allows breaking up logic gates into smaller *atoms*
 - The resulting graph is a *low-level* representation of the circuit
- Using transformations on those atoms, we can visually deduce:
 - Properties of circuits
 - Entangled states
 - How quantum protocols work

- Quantum circuit optimization
 - Reduce the complexity of circuits
 - T-count reduction
 - Important metric for performing fault-tolerant computations
- Circuit compilation
 - Reducing high-level quantum algorithms to run on a target architecture

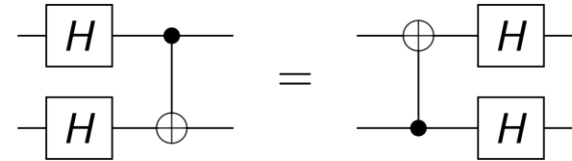
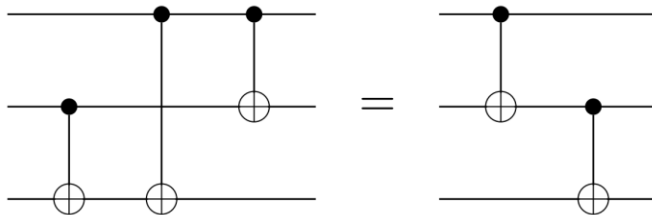
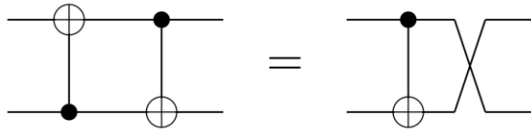
Quantum Circuit optimization

- Goal: Transform circuits into equivalent circuits
 - New Circuit must have fewer / simpler gates



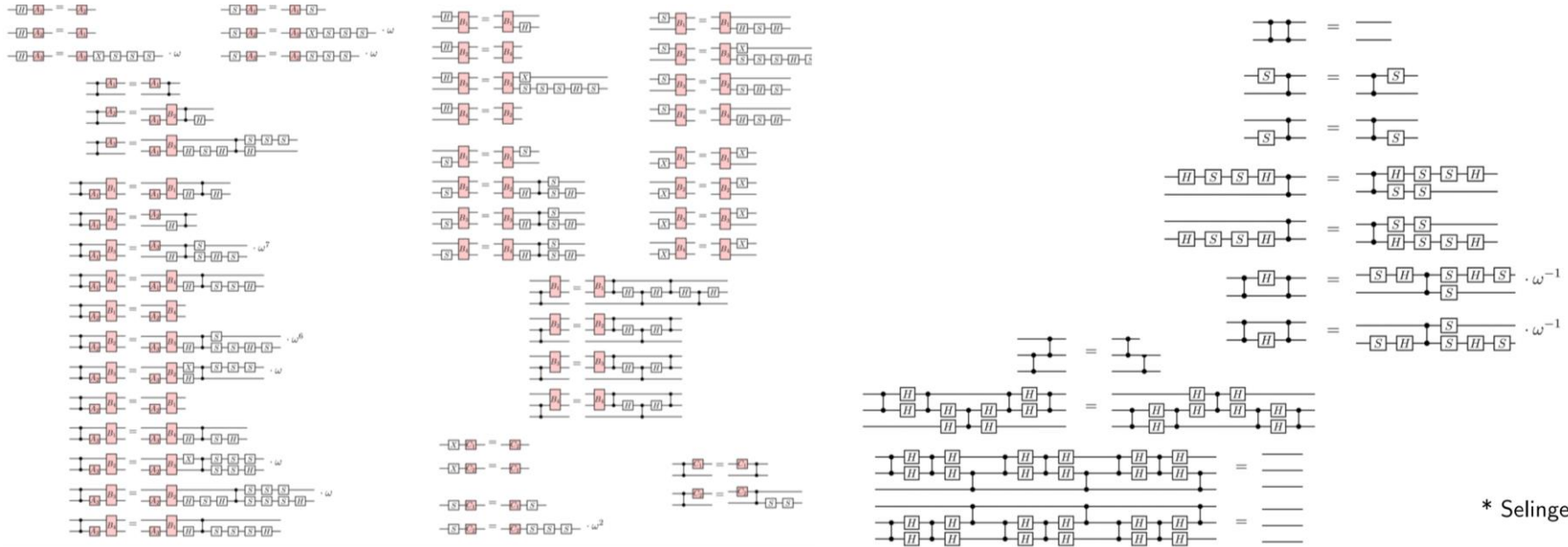
Classical optimization

- Why use ZX-Calculus for this?
- Circuit simplifications also exist for logic gates:



Classical optimization

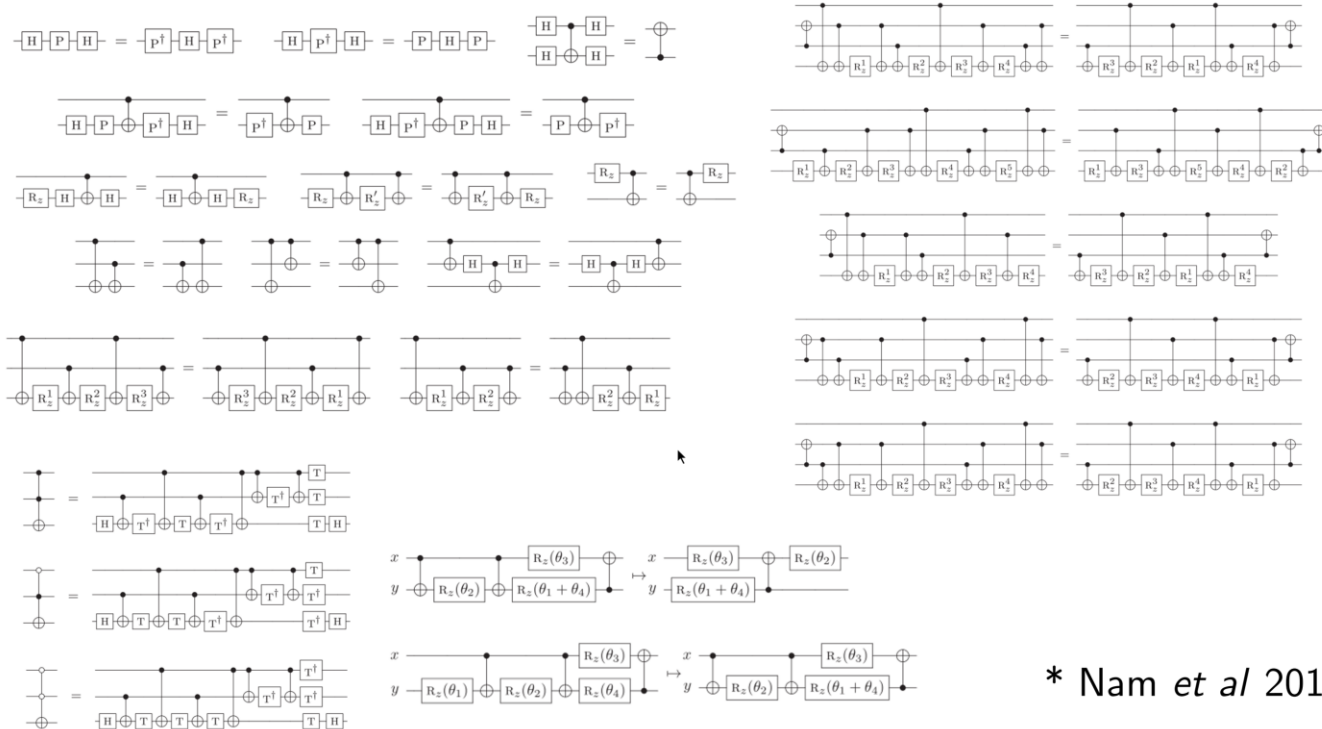
- Unfortunately, there are more ...



* Selinger 2015

Classical optimization

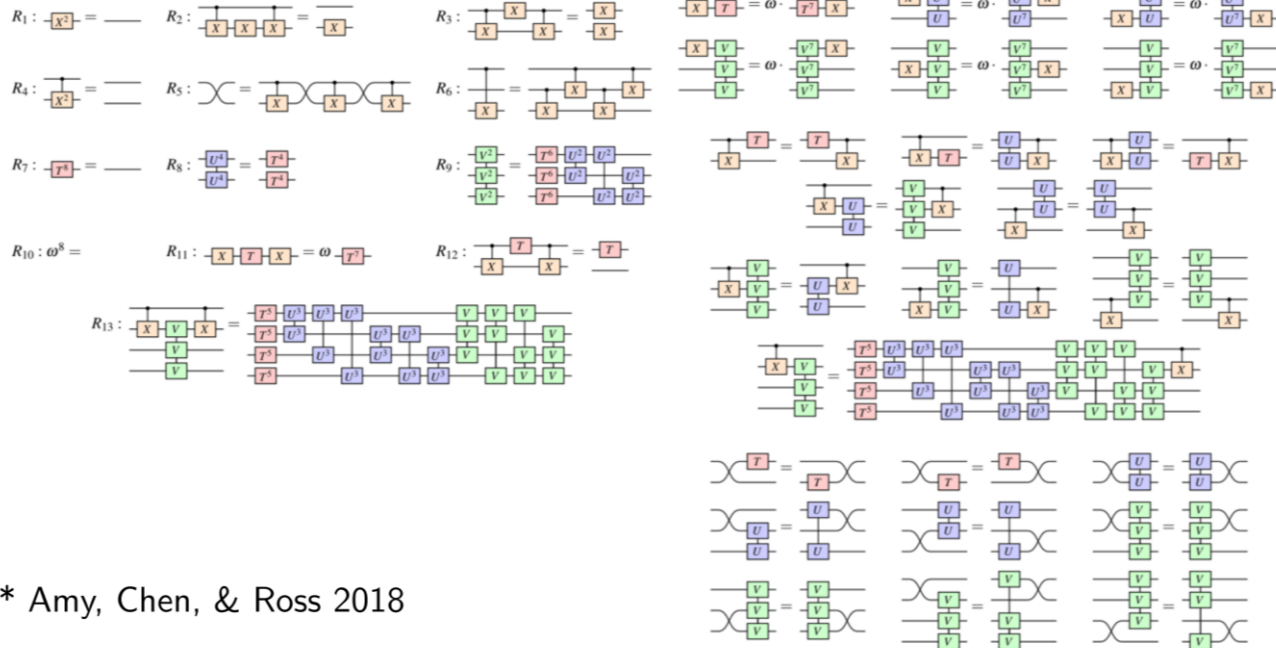
- And more ...



* Nam et al 2018

Classical optimization

- And even more ...

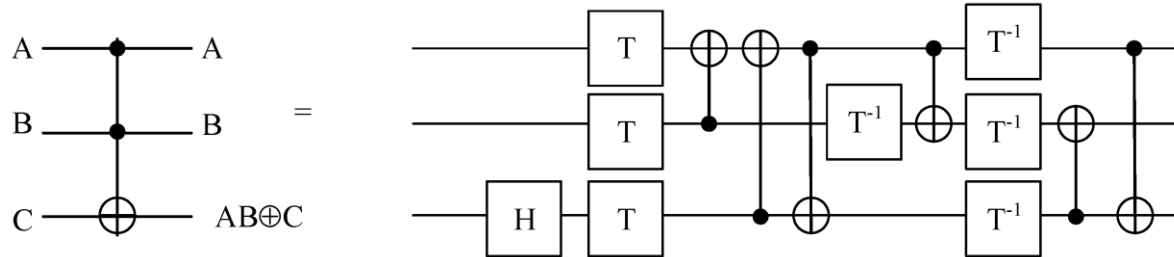


* Amy, Chen, & Ross 2018

- Optimizing a circuit classically is unpractical
 - Finding an optimal simplification path is hard
 - Huge number of identities to consider
- ZX-Calculus solves this problem
 - It has a massively reduced set of simplification rules
 - Finding the optimal path is still hard, but it's manageable

T-Count Optimization

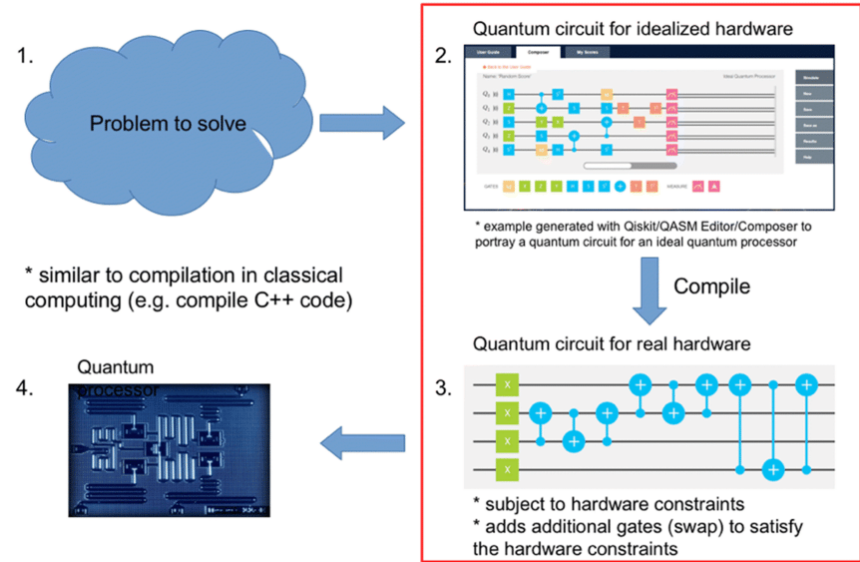
- Quantum computers are affected by noise
- Clifford+T Circuits can be made tolerant to noise (Error correcting codes)
 - Problem: Many new T-Gates need to be introduced
 - Difficult to simulate (Hardware Limits, high latency)
- ZX-Calculus can simplify such circuits



Decomposition of a Toffoli Gate: TCount = 7

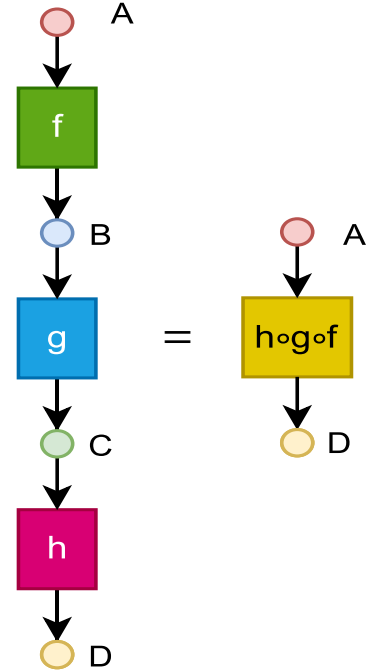
Quantum Circuit Compilation

- Circuits use many abstract gates
- Problems of real quantum computers:
 - Limited set of gates
 - Limited connectivity between qubits
 - Noise
- Special circuit transformations
 - To swap qubits, so they can be entangled
 - Reduce the depth of circuits (noise issues)
 - Allow only some sets of gates
- ZX-Calculus can solve this
 - Can try to stick to those rules during the reconstruction phase



Mathematical Background: Category Theory

- Category \mathcal{C} consists of **objects** and **arrows** (/morphisms)
 - Objects: $\{A, B, C, D\}$
 - Morphisms: $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$
- Every object A has an identity function
 - $id_A: A \rightarrow A$
 - Represent *loops* in the diagram
- Naturally there exists an associative Composition (\circ)
 - Which is just the application of f and then g in a sequence
 - Example: $g \circ f: A \rightarrow C$
 - Composition with id does nothing
 - $id_B \circ f = f \circ id_A = f: A \rightarrow B$



$$h \circ g \circ f: A \rightarrow D$$

Extension: Monoidal Category

- Extend a Category \mathcal{C} with:

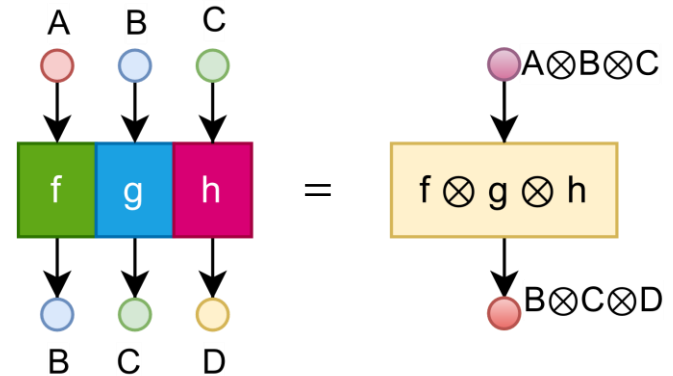
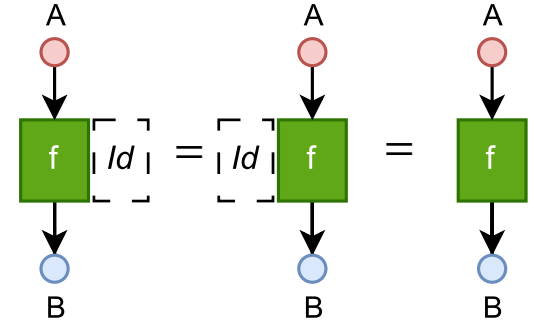
- An associative Bifunctor: $\otimes: \mathcal{C} \times \mathcal{C} \rightarrow \mathcal{C}$
- A special object $I \in \text{obj}(\mathcal{C})$ with
 - $A \otimes I = A \otimes I = A$ for all objects A

- Example:

- For the morphisms: $f: A \rightarrow B$, $g: B \rightarrow C$, $h: C \rightarrow D$
- We can *apply* them in *parallel*
 - $f \otimes g \otimes h: A \otimes B \otimes C \rightarrow B \otimes C \otimes D$

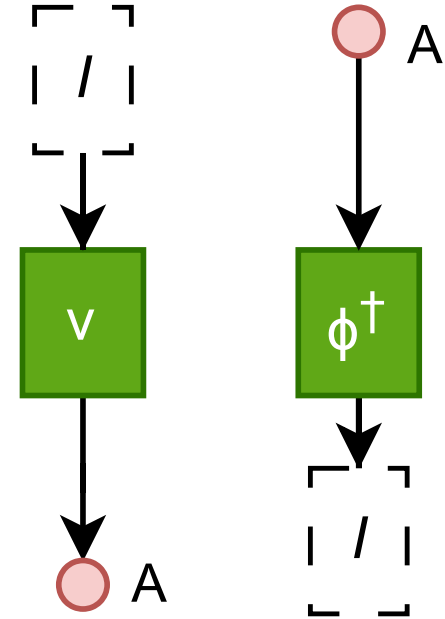
- So far: Very similar to regular circuit calculations

- Matrix product for sequential gates
- Tensor product for parallel gates



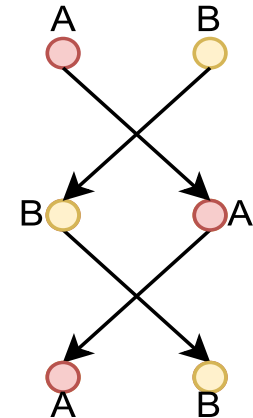
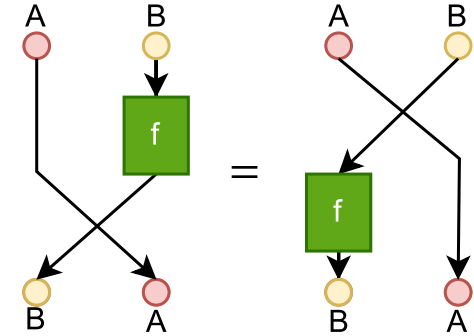
Special morphisms in Monoidal Categories

- Preparing States:
 - $v: I \rightarrow A$ (Ket)
 - Creates State A out of nothing
- Erasing States:
 - $\phi^\dagger: A \rightarrow I$ (Bra)
 - Deletes state A



Extension: Symmetric Monoidal Category

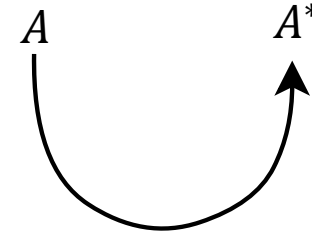
- Extend a Monoidal Category \mathcal{C} with:
 - Special tensor product: Swap isomorphism
 - $\sigma_{A,B} : A \otimes B \rightarrow B \otimes A$
 - For all $A, B \in \text{obj}(\mathcal{C})$
- Some other rules must hold:
 - “Push through” rule
 - “Double Swap = Identity” rule



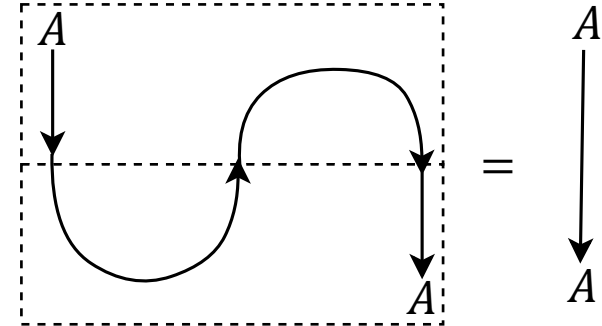
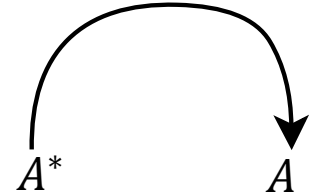
Extension: Compact Monoidal Category

- If a monoidal category \mathcal{C} has:
 - A dual object A^* for every object $A \in \text{obj}(\mathcal{C})$
 - Special morphisms:
 - *Unit*: $\eta_A : I \rightarrow A^* \otimes A$
 - *Counit*: $\epsilon_A : A \otimes A^* \rightarrow I$
- The combining rule must hold:
 - Meaning: $(\epsilon_A \otimes id_A) \circ (id_A \otimes \eta_A) = id_A$ for all $A \in \text{obj}(\mathcal{C})$
 - „Push through“ rule also holds
- Why do all this mathematical mess?
 - It can just be seen graphically!
 - Just follow the lines
 - “Only Topology Matters”

$$\epsilon_A : A \otimes A^* \rightarrow I$$

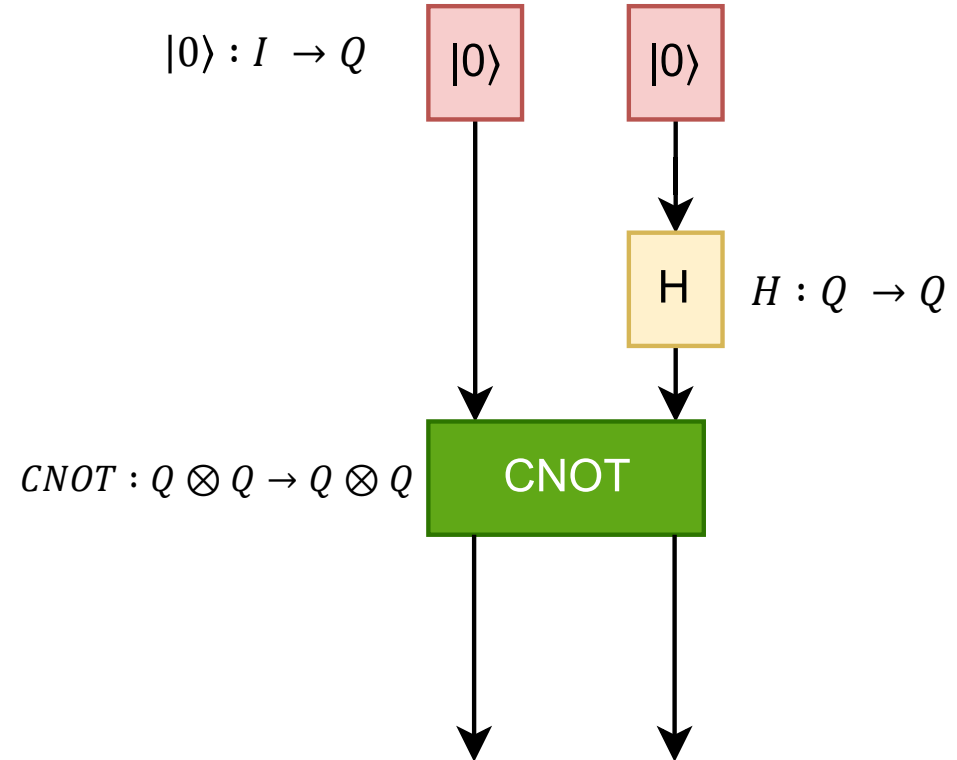


$$\eta_A : I \rightarrow A^* \otimes A$$

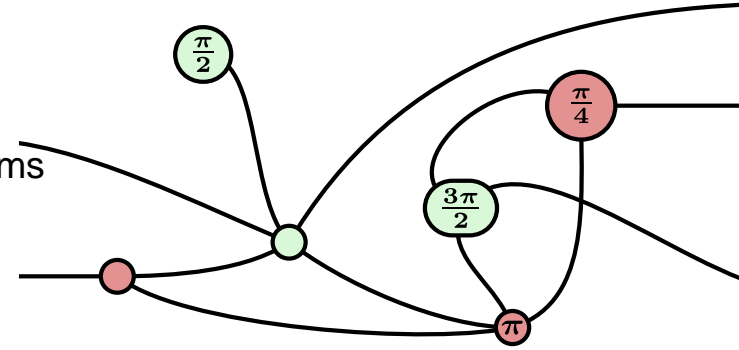


Example Network

- Concrete Category: **FDHilb**
 - Objects: Hilbert Spaces / (Vector Spaces)
 - Concrete $I = \mathbb{C}, Q = \mathbb{C}^n$
 - Morphisms: Linear Maps
 - $m = \mathbb{C}^{m \times n}$
 - Tensor Product: Kronecker Product
 - Composition: Matrix Product
- Skeleton of Entanglement Circuit
 1. Create two $|0\rangle$ States
 2. Apply Hadamard
 3. Apply CNOT

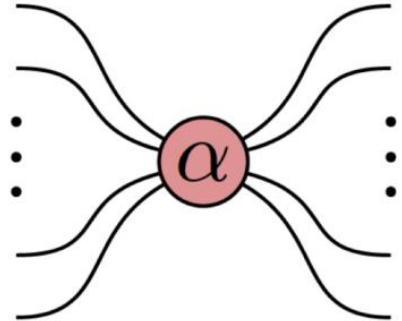
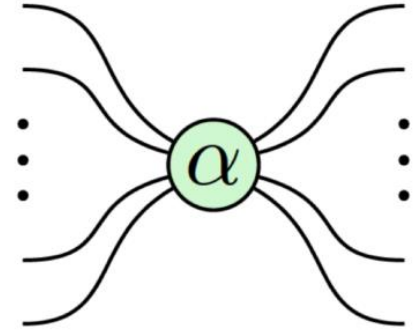


- Main idea of ZX-Calculus
 - Represent quantum circuits visually as a network of morphisms
 - Apply simplifications on the network
 - “Only topology Matters”
 - If it looks like the same graph, it’s the same thing
 - Guaranteed by the rules of the underlying Category
- The circuit consists of the *atoms* of logic gates
 - We represent these atoms using spiders
 - Spiders represent the morphisms from before



Spiders

- Spiders are the nodes in the graph
 - Arbitrary number of inputs/outputs
 - Represent linear maps
- Spider variations:
 - Green
 - Defined using eigenbasis of Z matrix
 - Red
 - Defined using eigenbasis of X matrix
 - Can have a phase angle α



Spiders as linear maps

- Each spider is a linear map

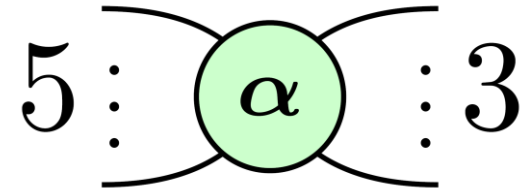
- $$\text{GreenSpider}(n, m)_{\alpha} = \underbrace{|0 \dots 0\rangle}_{m} \underbrace{\langle 0 \dots 0|}_{n} + e^{i\alpha} \underbrace{|1 \dots 1\rangle}_{m} \underbrace{\langle 1 \dots 1|}_{n}$$

- $$\text{RedSpider}(n, m)_{\alpha} = \underbrace{|+\dots+\rangle}_{m} \underbrace{\langle +\dots+|}_{n} + e^{i\alpha} \underbrace{|-\dots-\rangle}_{m} \underbrace{\langle -\dots-|}_{n}$$

- Example Spider:

- $\text{GreenSpider}(5, 3)_{\alpha}$ represents a $2^3 \times 2^5 = 8 \times 32$ matrix
 - Not unitary
 - Not even square
- Spiders extend circuits into non-dimensional matrices

$$\left[\begin{array}{c} \vdots \\ \text{Spider } \alpha \\ \vdots \end{array} \right] = \begin{pmatrix} 1 & 0 & & \\ 0 & 0 & & \\ & & \ddots & \\ & & & 0 & 0 \\ & & & 0 & e^{i\alpha} \end{pmatrix}$$



Spiders for Basis States

- Spiders can represent States **and** Gates

- Global scalars are omitted

- **X-Basis:** $GreenSpider(n, m)_\alpha = \underbrace{|0 \dots 0\rangle}_m \underbrace{\langle 0 \dots 0|}_n + e^{i\alpha} \underbrace{|1 \dots 1\rangle}_m \underbrace{\langle 1 \dots 1|}_n$

- $GreenSpider(0, 1)_0 = |0\rangle \cdot 1 + e^{i \cdot 0} |1\rangle \cdot 1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \propto |+\rangle$

- $GreenSpider(0, 1)_\pi = |0\rangle \cdot 1 + e^{i \cdot \pi} |1\rangle \cdot 1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \propto |-\rangle$

- **Z-Basis:** $RedSpider(n, m)_\alpha = \underbrace{|+\dots+\rangle}_m \underbrace{\langle +\dots+|}_n + e^{i\alpha} \underbrace{|-\dots-\rangle}_m \underbrace{\langle -\dots-|}_n$

- $RedSpider(0, 1)_0 = |+\rangle \cdot 1 + e^{i \cdot 0} |-\rangle \cdot 1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \propto |0\rangle$

- $RedSpider(0, 1)_\pi = |+\rangle \cdot 1 + e^{i \cdot \pi} |-\rangle \cdot 1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \propto |1\rangle$

$$\textcircled{0} = |+\rangle$$

$$\textcircled{\pi} = |-\rangle$$

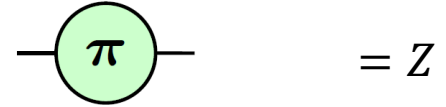
$$\textcircled{0} = |0\rangle$$

$$\textcircled{\pi} = |1\rangle$$

Spiders for Pauli Matrices

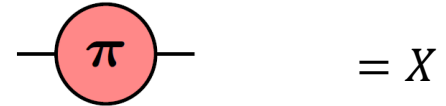
- **Pauli-Z:** $GreenSpider(n, m)_\alpha = \underbrace{|0 \dots 0\rangle}_m \underbrace{\langle 0 \dots 0|}_n + e^{i\alpha} \underbrace{|1 \dots 1\rangle}_m \underbrace{\langle 1 \dots 1|}_n$

- $GreenSpider(1, 1)_\pi = |0\rangle\langle 0| + e^{i\pi}|1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = Z$



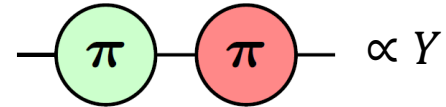
- **Pauli-X:** $RedSpider(n, m)_\alpha = \underbrace{|+\dots+\rangle}_m \underbrace{\langle +\dots+|}_n + e^{i\alpha} \underbrace{|-\dots-\rangle}_m \underbrace{\langle -\dots-|}_n$

- $RedSpider(1, 1)_\pi = |+\rangle\langle +| + e^{i\pi}|-\rangle\langle -| = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = X$



- **Pauli-Y:** (Use commutation rules of Pauli Gates)

- $RedSpider(1, 1)_\pi \circ GreenSpider(1, 1)_\pi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \circ \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \propto Y$



Spider for Identity Matrix

- Every spider of arity 2 and phase angle $\alpha = 0$ is an identity matrix
 - If spider has no angle, it implicitly means $\alpha = 0$
 - $GreenSpider(1,1)_0 = |0\rangle\langle 0| + e^{i \cdot 0} |1\rangle\langle 1| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = id_2$
 - $RedSpider(1,1)_0 = |+\rangle\langle +| + e^{i \cdot 0} |-\rangle\langle -| = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = id_2$
- This is an important rewrite rule!
 - Identity spiders can be removed from the diagram

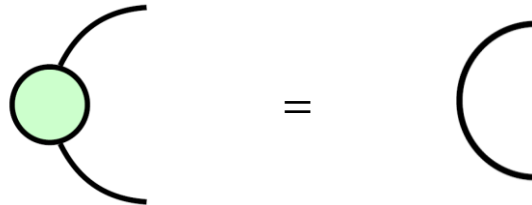


Spider for Bell State

- Spiders can generate entangled States

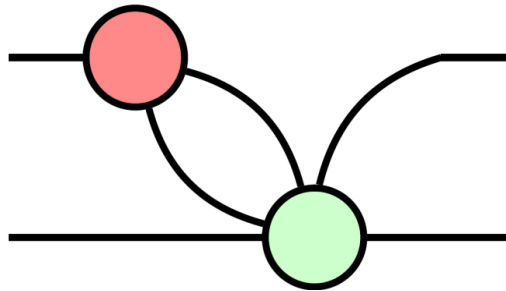
$$- \text{GreenSpider}(0, 2)_0 = |00\rangle \cdot 1 + e^{i \cdot 0} |11\rangle \cdot 1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \propto |\Phi^+\rangle$$

- Using the identity rule from before we get a CAP
 - $|\Phi^+\rangle$ represent the Unit-Morphism from the compact monoidal category



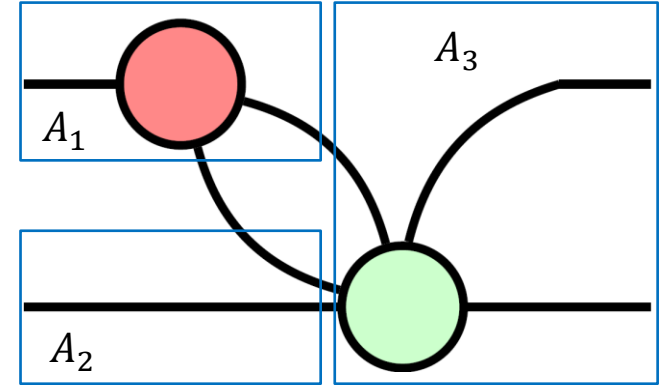
Combining Spiders

- Bigger Graphs can be constructed by connecting spiders
 - Every output of a spider gets connected to the input of another spider
 - Again: Only Topology matters
- The resulting graph can represent a quantum circuit
 - But not every graph is a valid circuit



Calculating a Graph

- To calculate the matrix representation
 - Divide the graph into regions
 - Each region must contain **exactly** one spider
- Simplification works just like normal circuits
 - “parallel” parts are combined using the tensor product
 - “sequential” parts are combined using the matrix product
- The resulting matrix represents the circuit
 - It is not necessary square / unitary



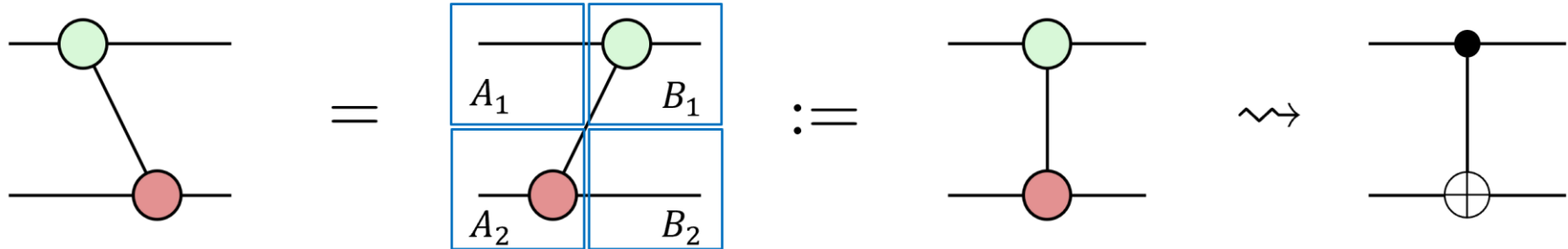
Example: CNOT

1. Evaluate parallel Sections

- $A = A_1 \otimes A_2 = id_2 \otimes RedSpider(1,2)$
- $B = B_1 \otimes B_2 = GreenSpider(2,1) \otimes id_2$

2. Combine sequential Regions

- $CNOT = B \circ A$



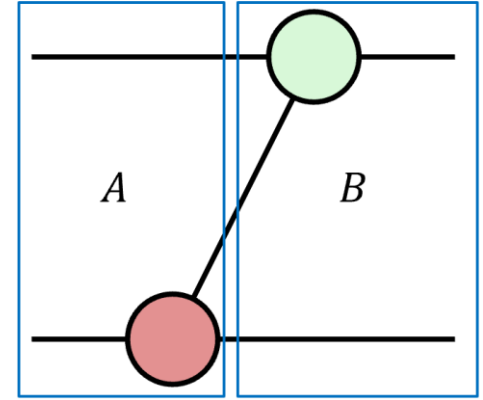
Example: CNOT Parallel Sections

- $A = id_2 \otimes RedSpider(1,2) = id_2 \otimes (|+\rangle\langle+| + |-\rangle\langle-|)$

$$- A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- $B = GreenSpider(2,1) \otimes id_2 = (|00\rangle\langle 0| + |11\rangle\langle 1|) \otimes id_2$

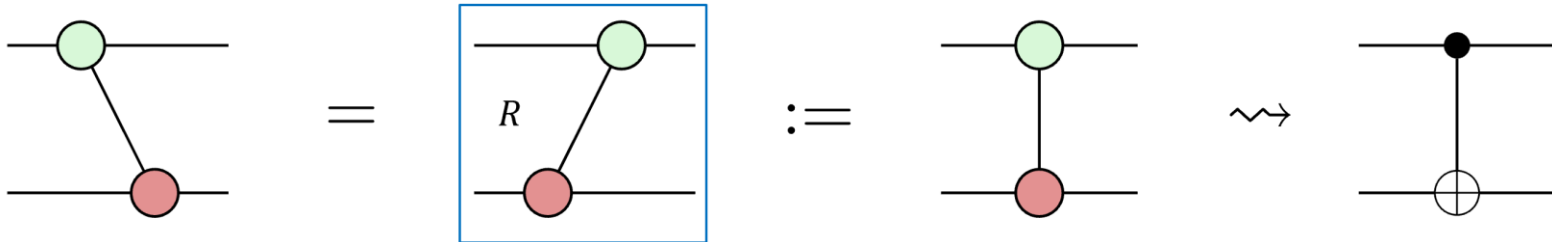
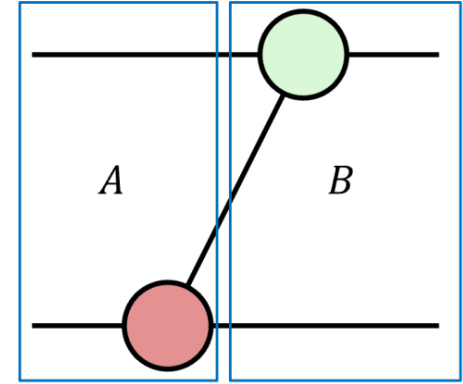
$$- B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$



Example: CNOT Sequential Sections

- $R = B \circ A$

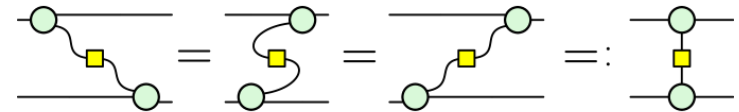
$$- R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \circ \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \propto CNOT$$



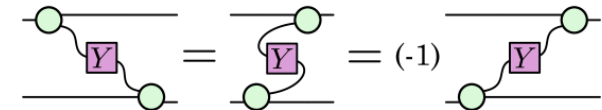
Where to draw the Regions?

- There may exist multiple ways of drawing the regions
 - Obvious, as you are allowed to move components around
 - “Only topology matters”
- This leads to different matrices in the calculation process
 - But the final matrices are always equivalent
 - (neglecting the global scalar factor)
- But why do it this way?
 - It is just as bad as the classical matrix approach

(a) Equivalent layouts of a CZ gate

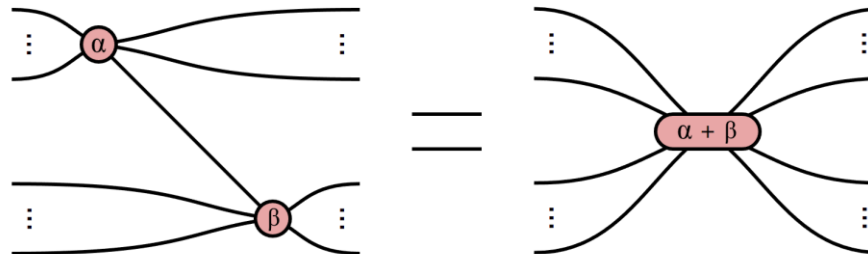


(b) Sign introduced by the transposition of a Y gate



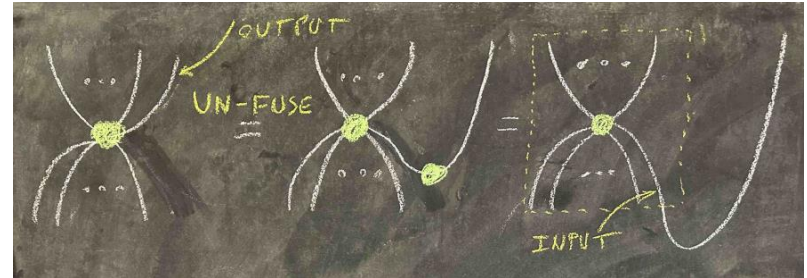
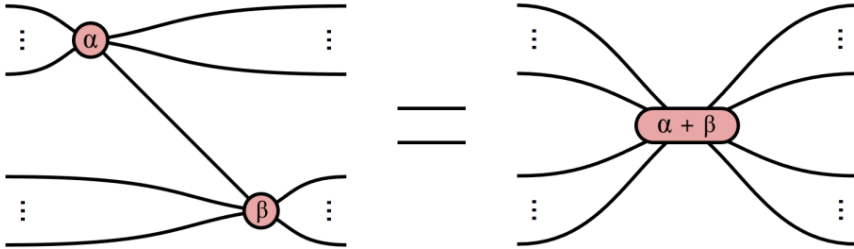
Simplification Rules

- We don't want to calculate the graph using its matrix form
- There exist many rules to simplify ZX-Graphs
 - But far fewer rules as for classical circuits
- We can apply the rules anywhere in the graph if:
 - The pattern for the substitution matches
 - The order of the input/output wires of regions is unchanged
 - All rule still holds if the spiders flip colors



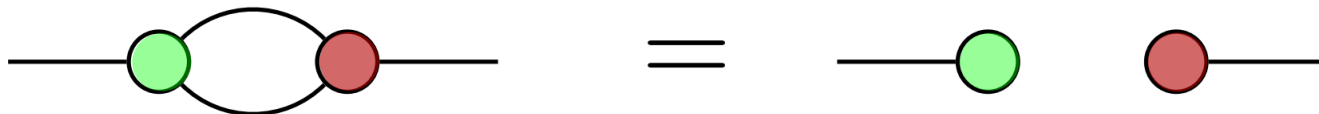
Spider Fusion

- Idea: Two spiders of the same color fuse together
 - Phase angles add up
 - Can simplify the graph a lot
- This rule allows to freely change input and outputs!
 - Only topology matters



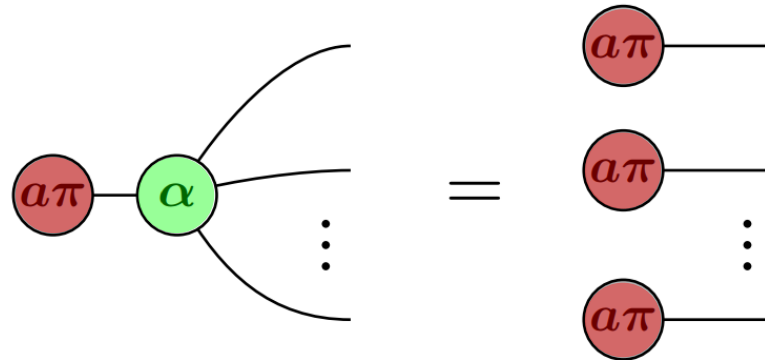
Hopf Rule

- Idea: 2 lines connecting 2 different colored spiders, can be deleted



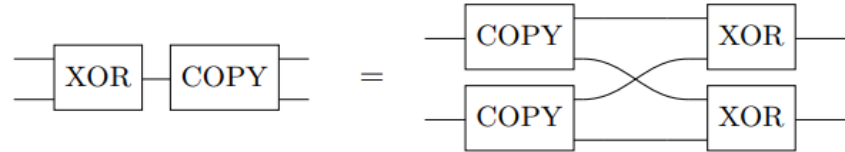
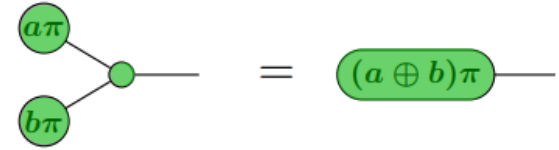
Basis-State Copy Rules

- Idea: “Green copies Red”, “Red copies Green”
 - Only works for computational basis states
 - $|0\rangle, |1\rangle / |+\rangle, |-\rangle$ depending on colors



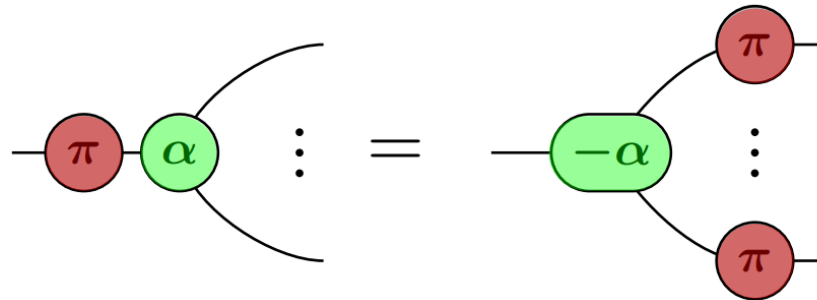
Bi-Algebra Rule

- Analog to digital logic
 - Create XOR using spider fusion
 - Use Copy-Rule from before



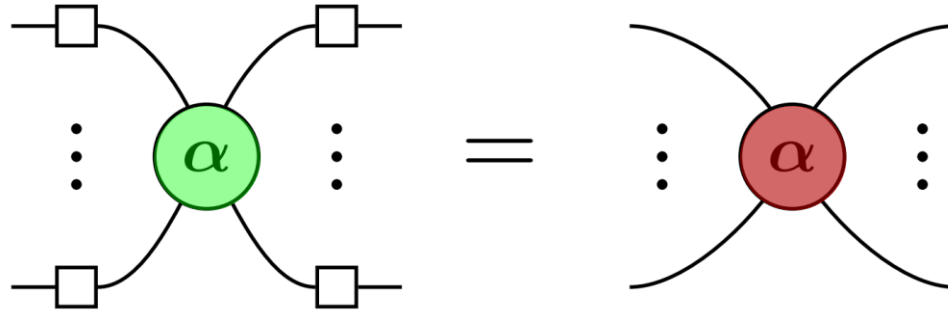
π -Commutation

- Idea: A spider with angle π copy through a spider of opposite color
 - The phase of the other spider gets flipped



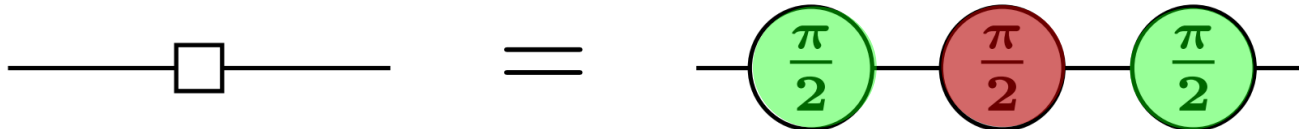
Color Change Rule

- Idea: We can change the color of a single spider
 - But we need to add Hadamard gates to all its inputs / outputs to compensate
- Analog: A Hadamard Gate can push through a spider
 - By copying itself to every other output



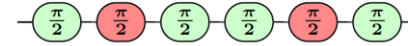
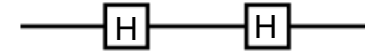
Hadamard Gate

- Hadamard Gates can be constructed using euler angles
 - Set up Hadamard as sequence of rotations

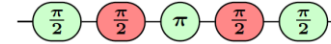


Hadamard Gate Cancelation

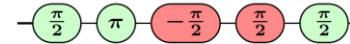
- Two Hadamard Gates should cancel!
 - Using a visual proof, we see that the rule holds



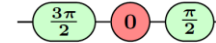
(Translate to ZX)



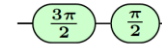
(Spider-Fusion)



(π -Commutation)



(Spider-Fusion)



(Identity-Removal)



(Spider-Fusion)



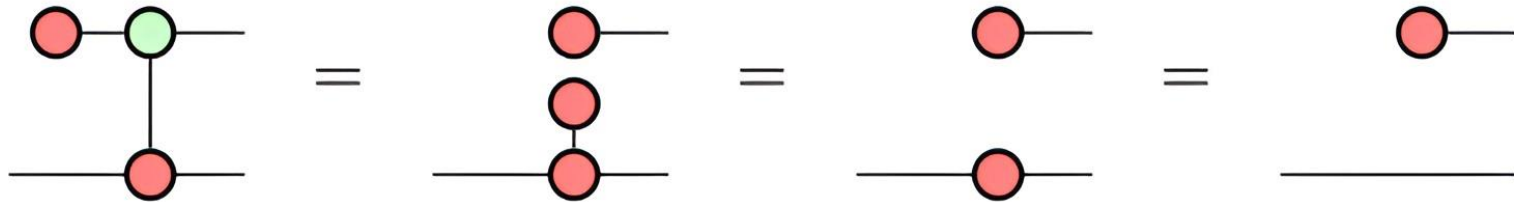
(Modulo)



(Identity-Removal)

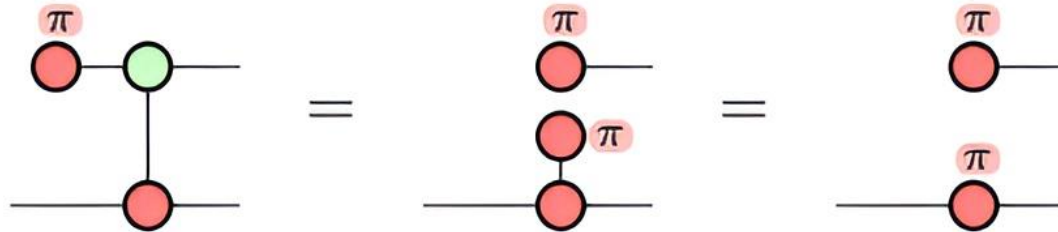
Example CNOT Application: Control = $|0\rangle$

- As expected:
 - $|0\rangle$ Qubit passes through
 - Target qubit is untouched



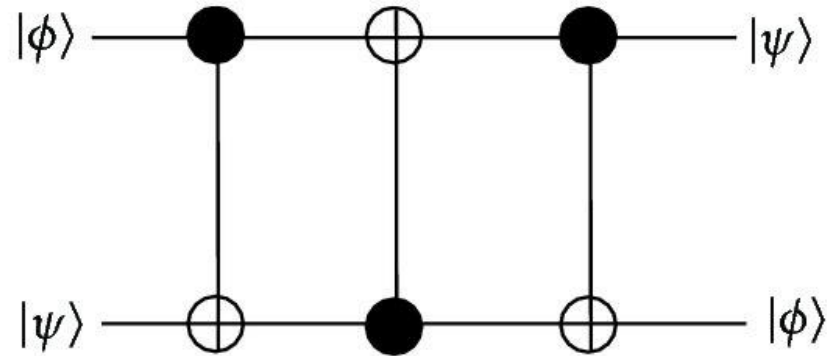
Example CNOT Application: Control = $|1\rangle$

- As expected:
 - $|1\rangle$ Qubit passes through
 - Pauli X gate is applied to target qubit



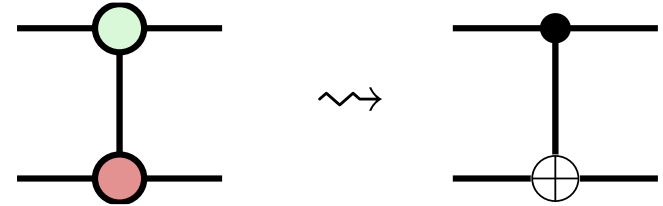
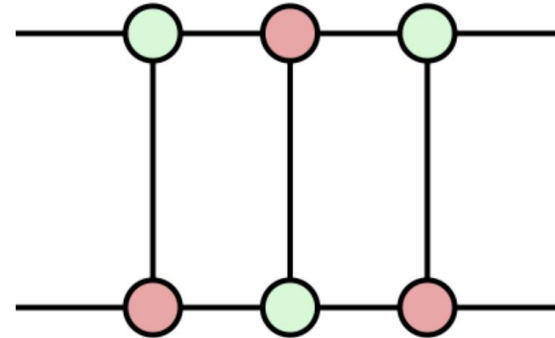
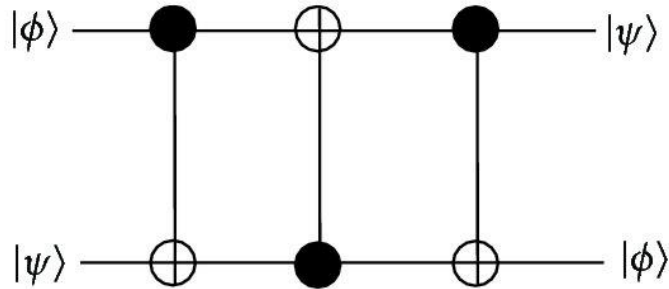
Simplification: Three CNOTs

- Circuit containing 3 alternating CNOTs



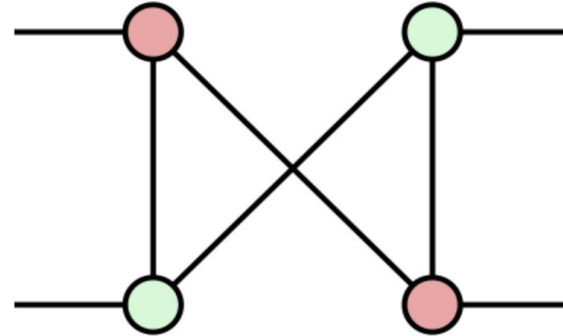
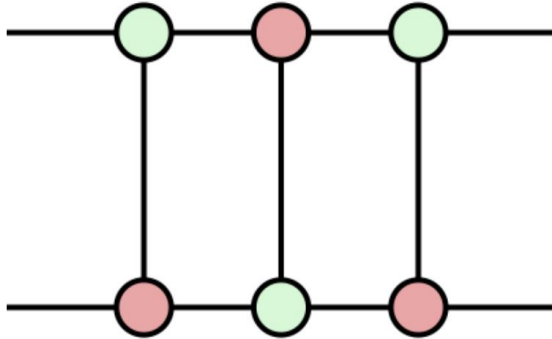
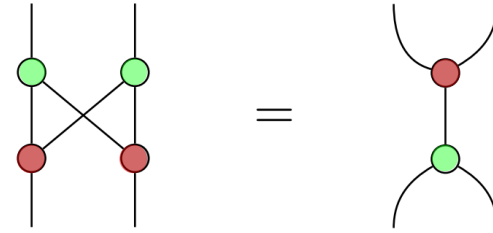
Simplification: Three CNOTs

- Transform into ZX-Diagram
 - Remember CNOT Gate from earlier



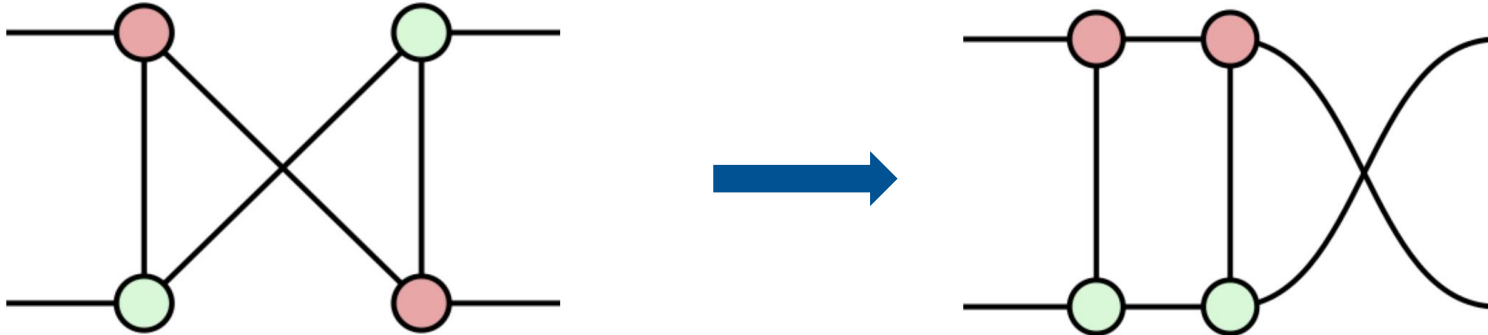
Simplification: Three CNOTs

- Apply the bi-algebra rule
 - We need to morph the graph first!



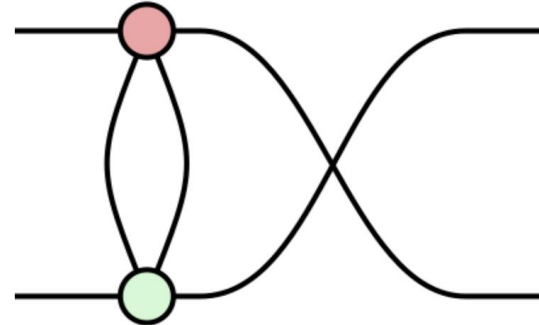
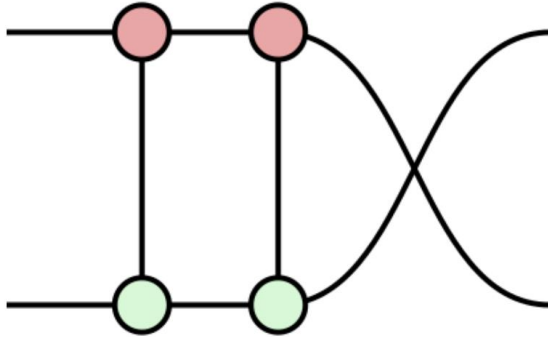
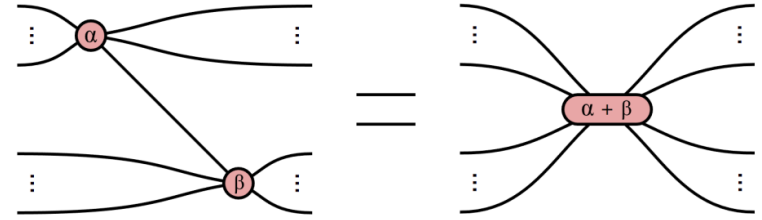
Simplification: Three CNOTs

- Morph the ZX-Diagram
 - Allowed, since we are allowed to move stuff freely
 - Guaranteed by underlying category



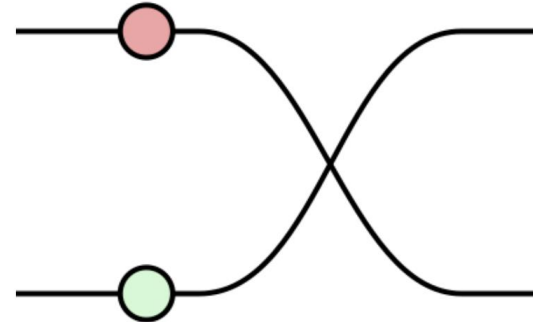
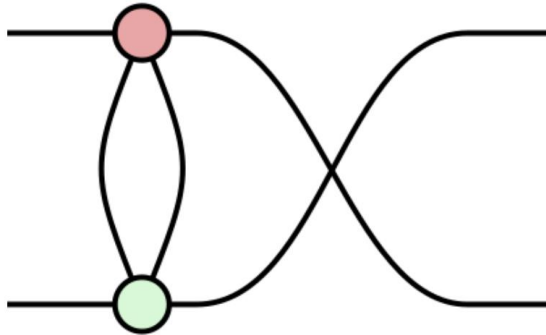
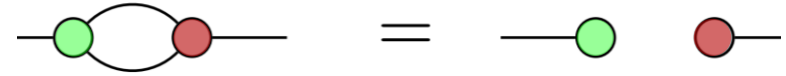
Simplification: Three CNOTs

- Apply spider fusion rule
 - Attention, two wires remain between the first nodes!



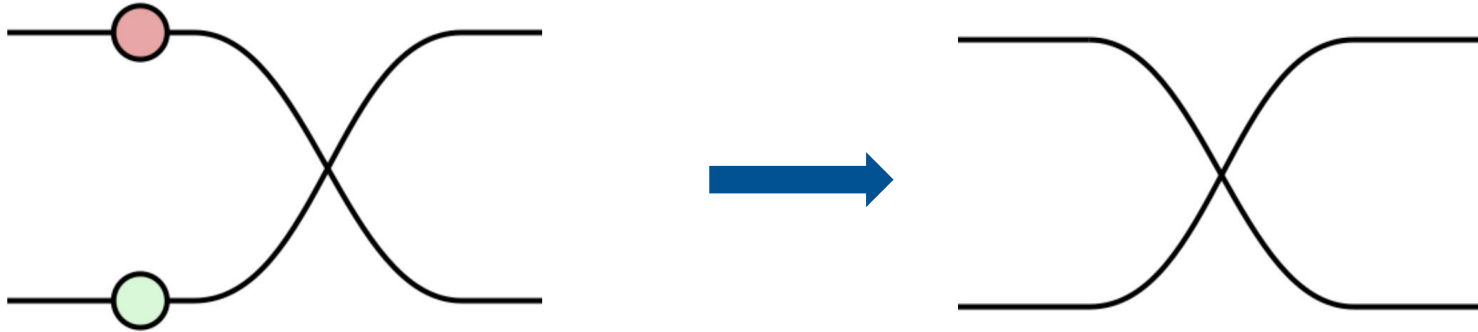
Simplification: Three CNOTs

- Apply the Hopf-Law



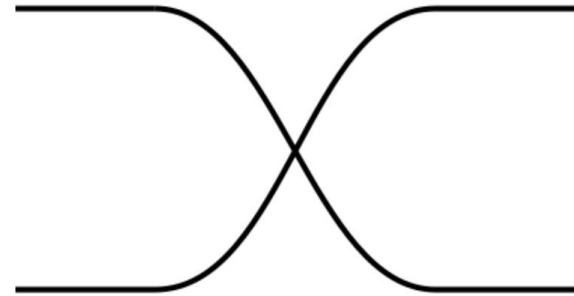
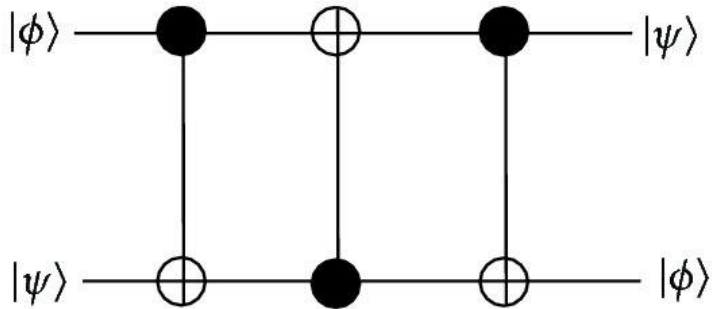
Simplification: Three CNOTs

- Identity Removal



Simplification: Three CNOTs

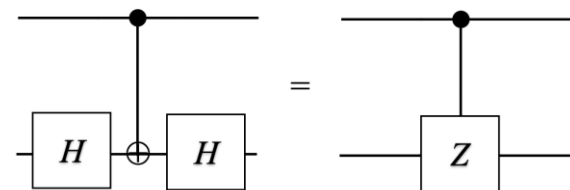
- Complicated CNOT circuit is actually a swap operation
 - Since we are working at the *atom*-level many other circuits can be simplified like this
 - Extracting back quantum circuits is hard
 - But it's possible for simple circuits
 - (In general: Graph needs to have “gFlow”)



Translating Circuits

identity		$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
Pauli Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Pauli X NOT gate		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli Y	i	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Hadamard gate		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
S gate		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
V gate		$\frac{1}{2} \begin{pmatrix} 1+i & 1-i \\ 1-i & 1+i \end{pmatrix}$
T gate		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$

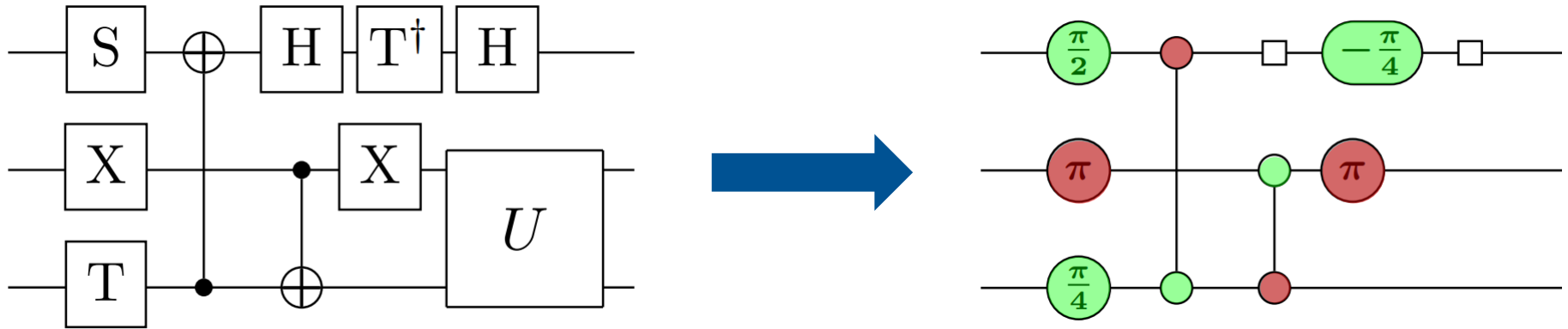
CNOT gate CX gate	$\sqrt{2}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CZ gate	$\sqrt{2}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$



Challenge: Try to prove this identity!

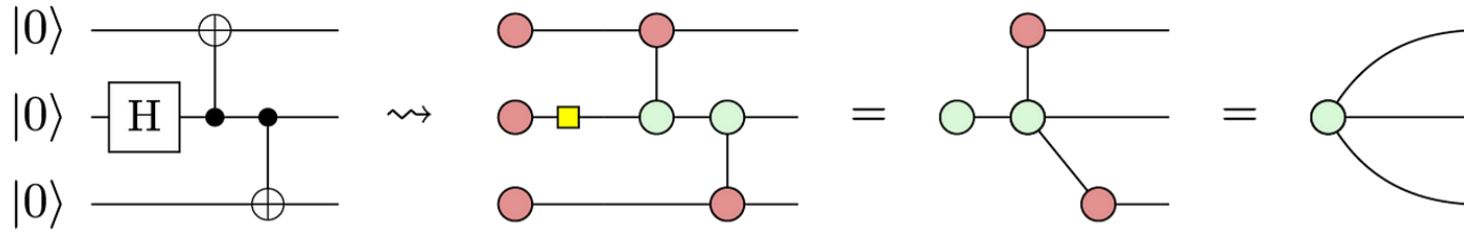
From Quantum Circuits to ZX-Diagrams

- Example circuit:
 - Custom gates need to be handled separately
- Backwards direction also possible

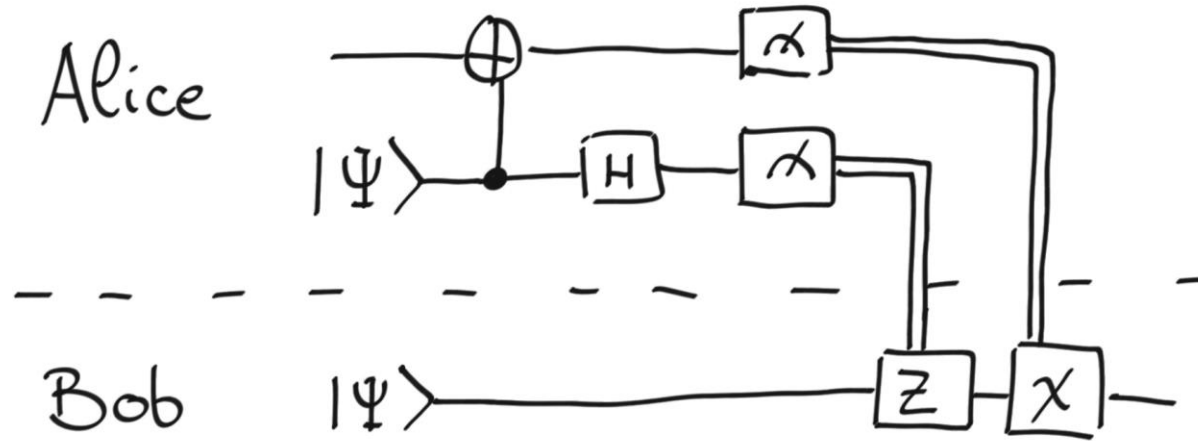


Example Circuit Simplification GHZ-State

- $|GHZ\rangle = \frac{|000\rangle + |111\rangle}{\sqrt{2}}$
 - Calculating the circuit with ZX seems trivial compared to classical reasoning
 - $GreenSpider(0, 3) = 1 \cdot |000\rangle + e^{i \cdot 0} \cdot 1 \cdot |111\rangle = |000\rangle + |111\rangle$

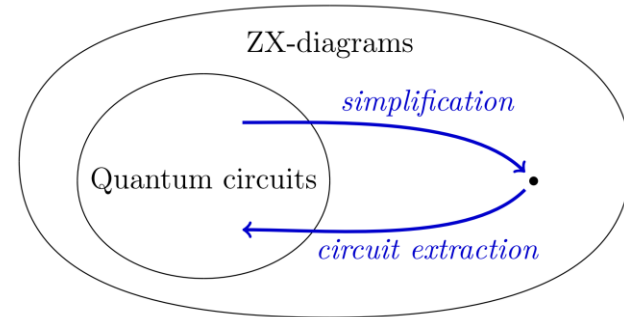
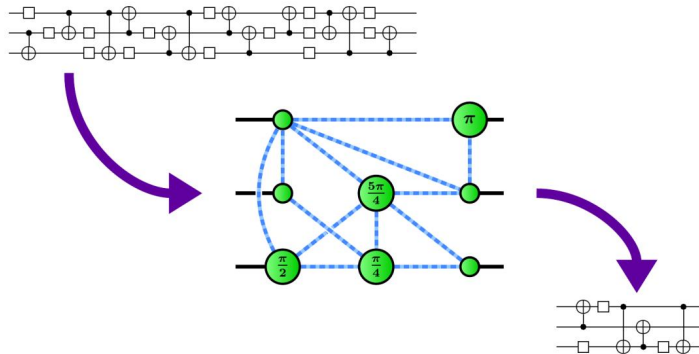


Quantum Teleportation



-
- Diagram illustrating the simplification of a quantum circuit:
- Initial Circuit:** Alice's qubit has a phase gate $\alpha\pi$ and a Hadamard gate. Bob's qubit has a phase gate $\alpha\pi$ and a Hadamard gate. The circuit is controlled by a $1/\sqrt{2}$ gate.
 - Simplification Step 1:** The Hadamard gates are moved to the end of the qubits, resulting in a phase gate $\beta\pi$ on Alice's qubit and a phase gate $\beta\pi$ on Bob's qubit.
 - Simplification Step 2:** The phase gates are combined, resulting in a phase gate $2\beta\pi$ on Alice's qubit and a phase gate $\alpha\pi$ on Bob's qubit.
 - Simplification Step 3:** The phase gates are further simplified, resulting in a phase gate $2\alpha\pi$ on Alice's qubit and a phase gate $\alpha\pi$ on Bob's qubit.
 - Simplification Step 4:** The final simplified circuit shows a phase gate $2\alpha\pi$ on Alice's qubit and a phase gate $\alpha\pi$ on Bob's qubit.

- ZX Calculus is complete for Clifford+T gates
 - Any two ZX-Graphs for the same Quantum circuit can be transformed into each other
 - Since Clifford+T is approximately universal, ZX-Calculus is too
- This means that if a simpler circuit exists, it can be found using ZX-calculus
 - But the path between those transformations may traverse invalid quantum circuits



External Image Sources

- Quantum compilation [P.4]: <https://quantum-journal.org/papers/q-2020-06-04-279/>
- Circuit optimization [P.5]: <https://quantum-journal.org/papers/q-2020-06-04-279/>
- Circuit Identities: [P.6 - P.9]: <https://arxiv.org/pdf/1310.6813.pdf>
- Spiders as Matrix [P.20]: <https://zxcalculus.com/tutorial>
- CNOT ZX-Graph [P.27]: https://commons.wikimedia.org/wiki/File:ZX-calculus_cNOT-example.svg
- Example Circuit Identities [P.30]: <https://www.arxiv-vanity.com/papers/2303.08829/>
- Spider Fusion [P.32]: https://www.cs.ox.ac.uk/people/bob.coecke/ZX-lectures_JPG.pdf
- CNOT Application [P.40 – P.41]: <https://quantumcomputing.stackexchange.com/questions/10235/explain-the-representation-of-the-cnot-gate-in-zx-calculus>
- CNOT Simplification [P.43 – P.48]: <https://zxcalculus.com/>
- Translating from Gates to ZX [P.50], Example Circuit [P.51]: <https://arxiv.org/pdf/2012.13966.pdf>
- GHZ State Simplification [P.52]: <https://medium.com/quantinuum/how-zx-calculus-reveals-the-logic-and-processes-of-quantum-mechanics-to-everyone-944fc3bbb2c>
- Teleportation, Toffoli Gate Examples [P.53 – P.56]: https://pennylane.ai/qml/demos/tutorial_zx_calculus
- Completeness [P.57]: <https://arxiv.org/pdf/1902.03178.pdf>