PyZX:
Large Scale Automated Diagrammatic Reasoning

Aleks Kissinger

aleks@cs.ru.nl

John van de Wetering

john@vdwetering.name

Institute for Computing and Information Sciences
Radboud University Nijmegen

June 10, 2019

PyZX

- Open source Python library
- `github.com/Quantomatic/pyzx`
- `zxcalculus.com/pyzx`

PyZX

- Open source Python library
- `github.com/Quantomatic/pyzx`
- `zxcalculus.com/pyzx`
- With PyZX you can manipulate large ZX-diagrams

PyZX

- Open source Python library
- `github.com/Quantomatic/pyzx`
- `zxcalculus.com/pyzx`
- With PyZX you can manipulate large ZX-diagrams
- It can be used for:

PyZX

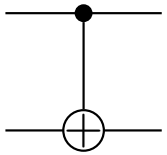
- Open source Python library
- `github.com/Quantomatic/pyzx`
- `zxcalculus.com/pyzx`
- With PyZX you can manipulate large ZX-diagrams
- It can be used for:
  - Quantum circuit optimisation
  - Quantum circuit validation
  - ...

PyZX

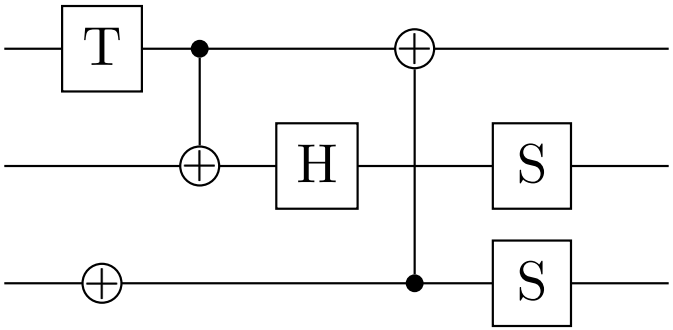
- Open source Python library
- `github.com/Quantomatic/pyzx`
- `zxcalculus.com/pyzx`
- With PyZX you can manipulate large ZX-diagrams
- It can be used for:
  - Quantum circuit optimisation
  - Quantum circuit validation
  - ...
- Why would we use ZX-diagrams for these things?

Circuit diagrams

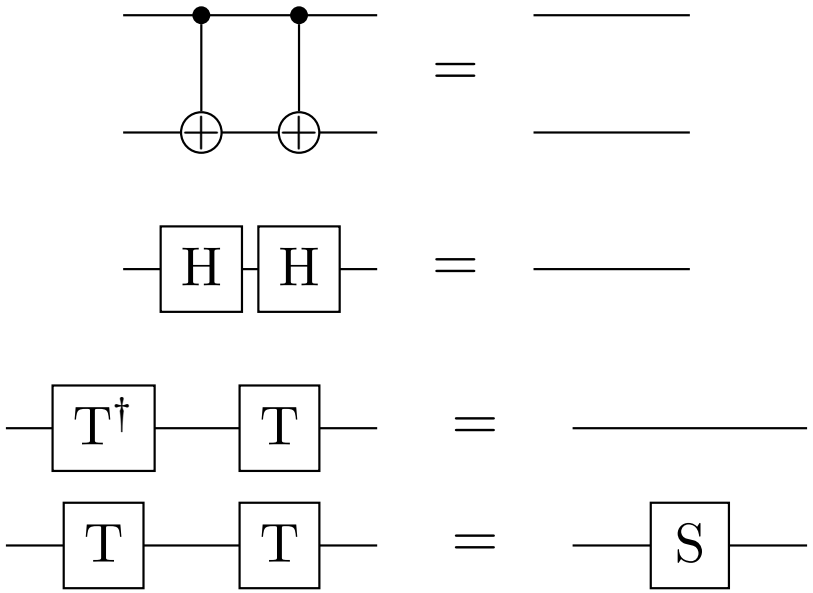
NOT = 

CNOT = 

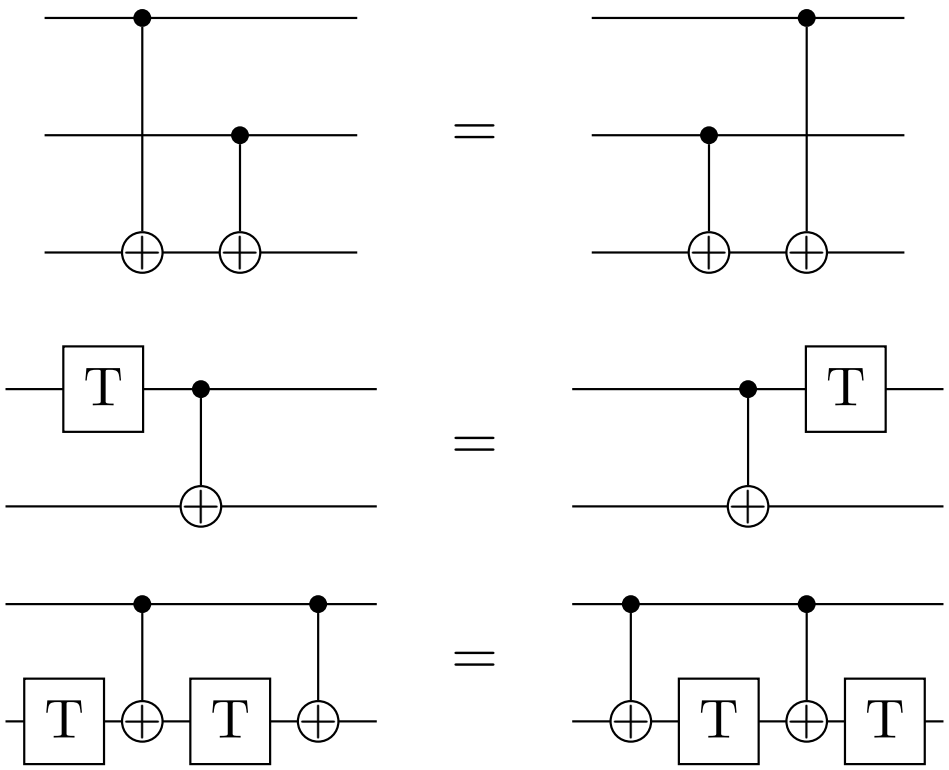
An example quantum circuit:



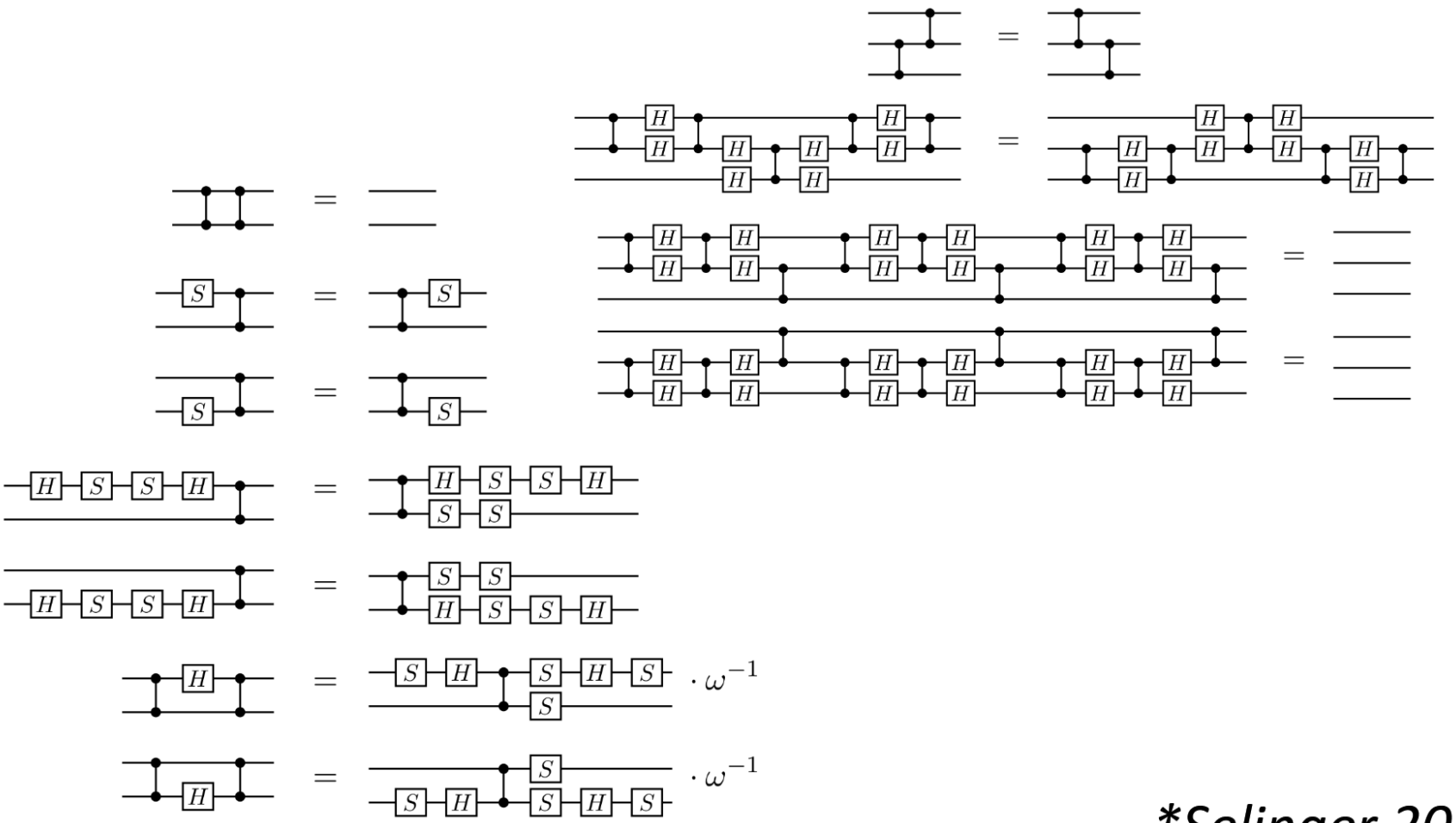
Circuit identities



Gate commutation

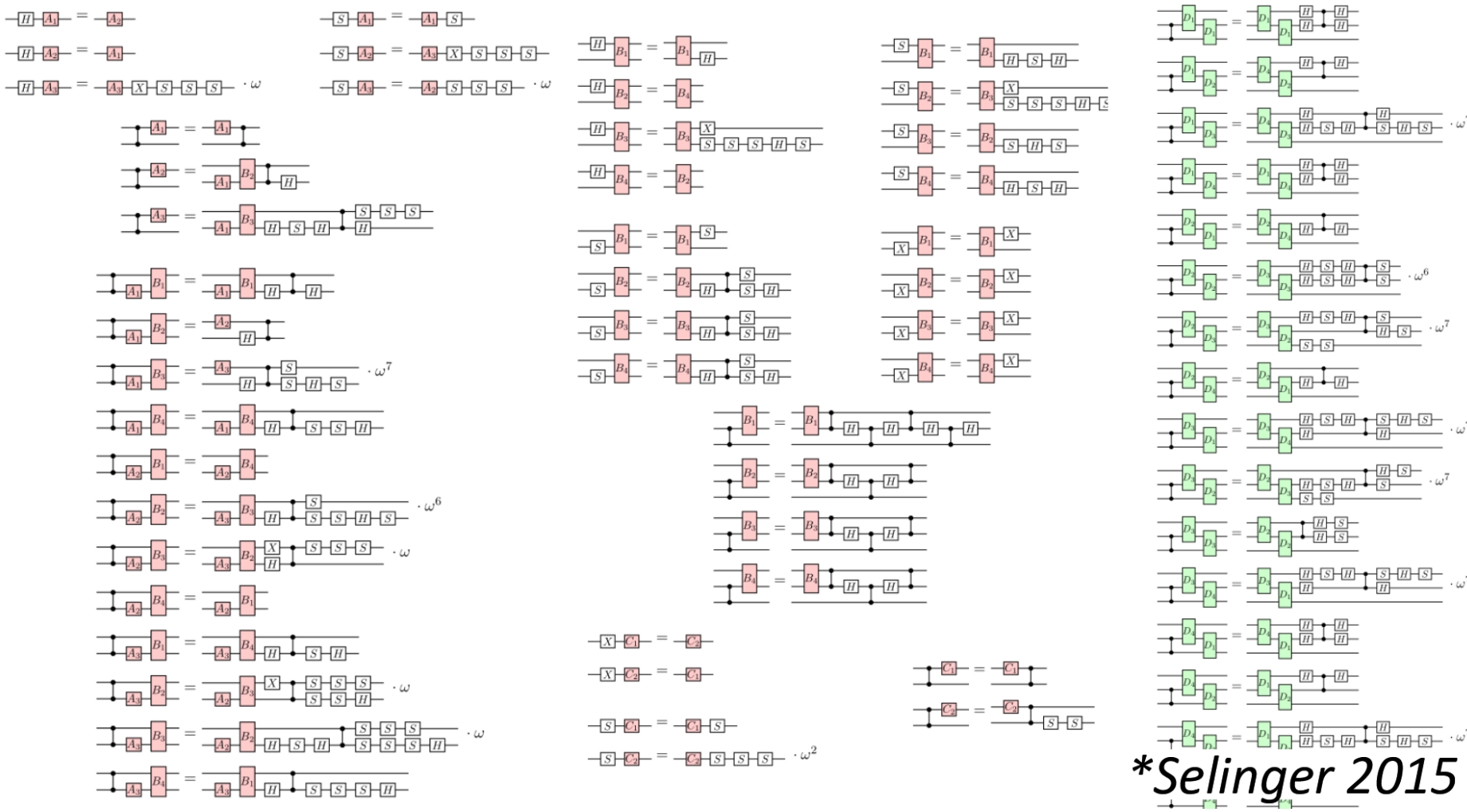


More circuit equalities



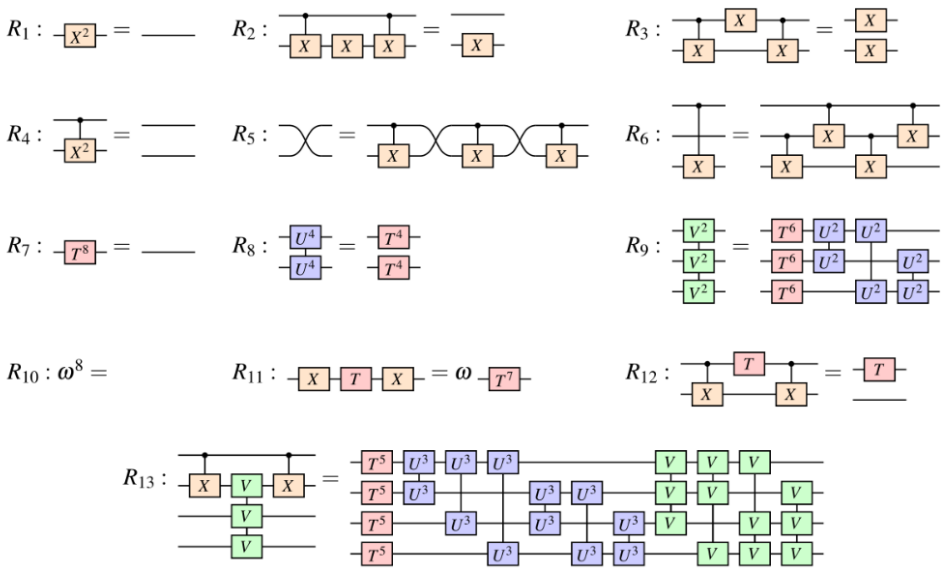
*Selinger 2015

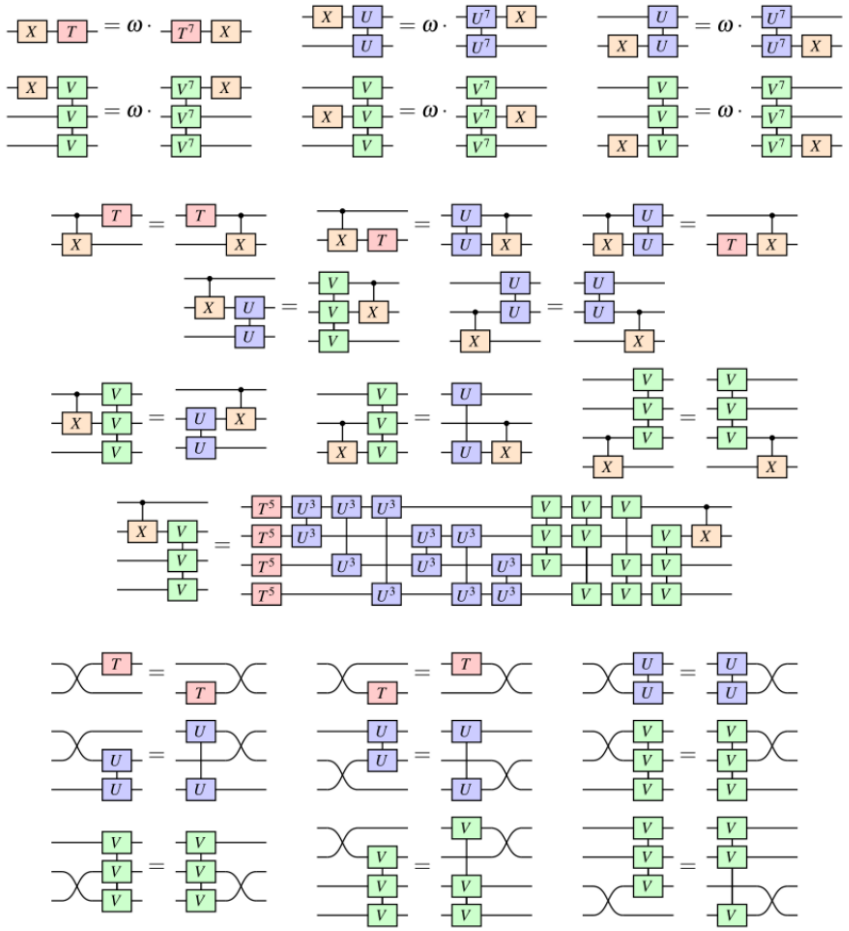
And more circuit equalities



*Selinger 2015

And even more circuit equalities





*Amy, Chen, & Ross 2018

Things get messy
because circuits are very rigid

Things get messy
because circuits are very rigid

Enter ZX-diagrams

ZX-diagrams

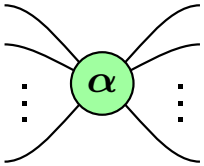
What gates are to circuits, *spiders* are to ZX-diagrams.

ZX-diagrams

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

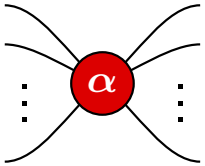
Z-spider

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



X-spider

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

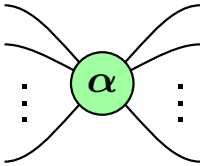


ZX-diagrams

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

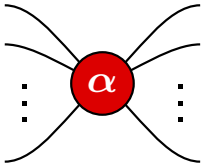
Z-spider

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

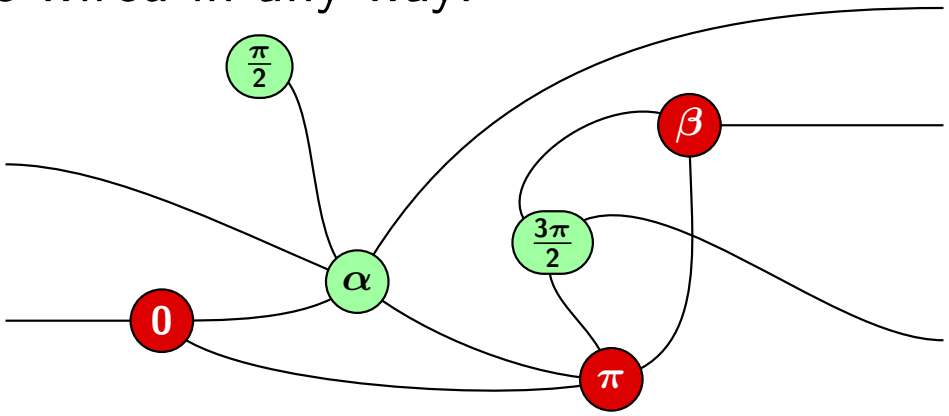


X-spider

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

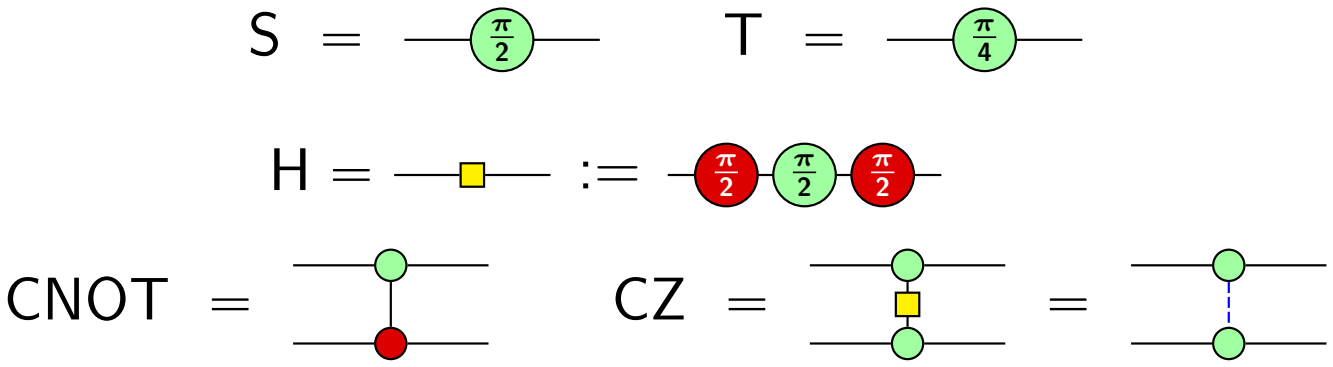


Spiders can be wired in any way:



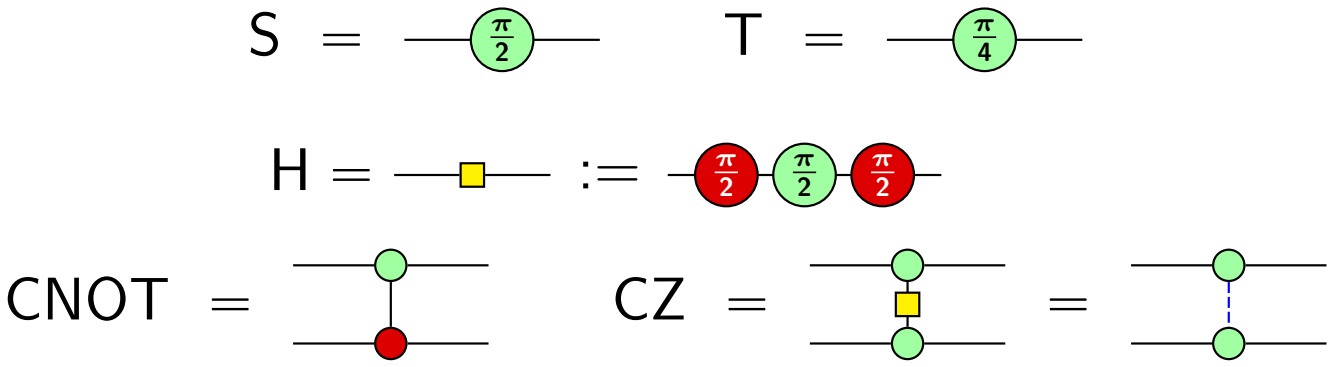
Quantum gates as ZX-diagrams

Every quantum gate can be written as a ZX-diagram:



Quantum gates as ZX-diagrams

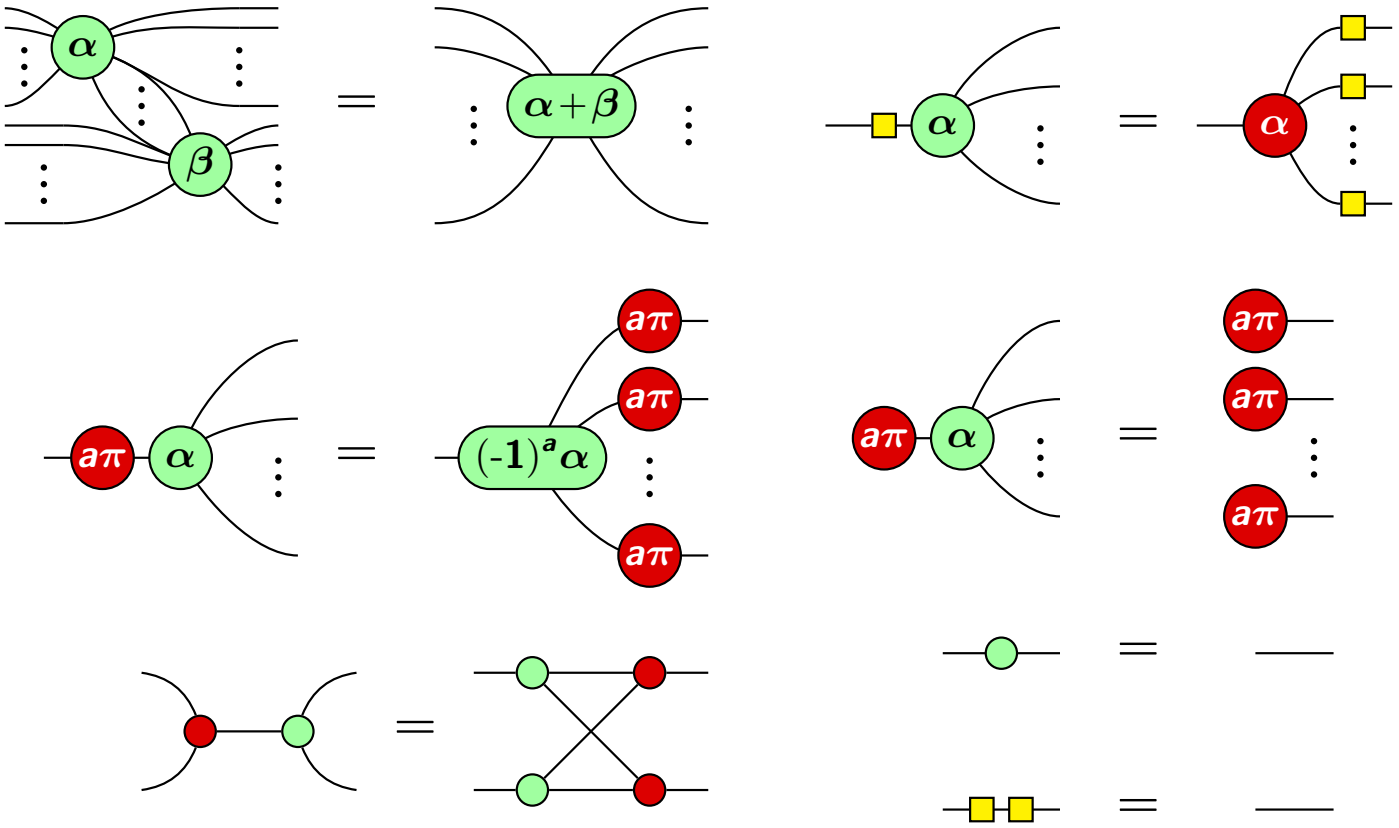
Every quantum gate can be written as a ZX-diagram:



Universality

Any linear map between qubits can be represented as a ZX-diagram.

Rules for ZX-diagrams: The ZX-calculus



$\alpha, \beta \in [0, 2\pi]$, $a \in \{0, 1\}$

Completeness of the ZX-calculus

Theorem

If two ZX-diagrams represent the same linear map, then they can be transformed into one another using the previous rules (and some additional ones).

Completeness of the ZX-calculus

Theorem

If two ZX-diagrams represent the same linear map, then they can be transformed into one another using the previous rules (and some additional ones).

So instead of dozens of circuit equalities, we just need a few simple rules.

Architecture of PyZX

Two main datastructures in PyZX: `Circuits` and `Graphs`.

Architecture of PyZX

Two main datastructures in PyZX: `Circuits` and `Graphs`.

`Circuits` are just lists of gates.

Architecture of PyZX

Two main datastructures in PyZX: `Circuits` and `Graphs`.

`Circuits` are just lists of gates.

A `Graph` represents a ZX-diagram:

- Three types of vertices: *boundary*, X and Z.
- Phases are stored as rational fractions of π.

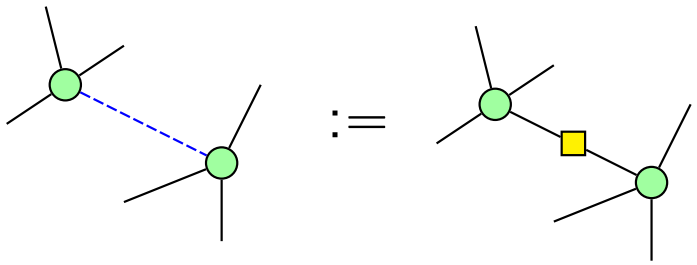
Architecture of PyZX

Two main datastructures in PyZX: `Circuits` and `Graphs`.

`Circuits` are just lists of gates.

A `Graph` represents a ZX-diagram:

- Three types of vertices: *boundary*, X and Z.
- Phases are stored as rational fractions of π.
- Two types of edges: *regular* and *Hadamard*:



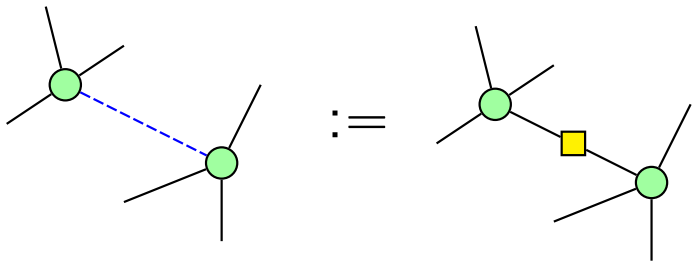
Architecture of PyZX

Two main datastructures in PyZX: `Circuits` and `Graphs`.

`Circuits` are just lists of gates.

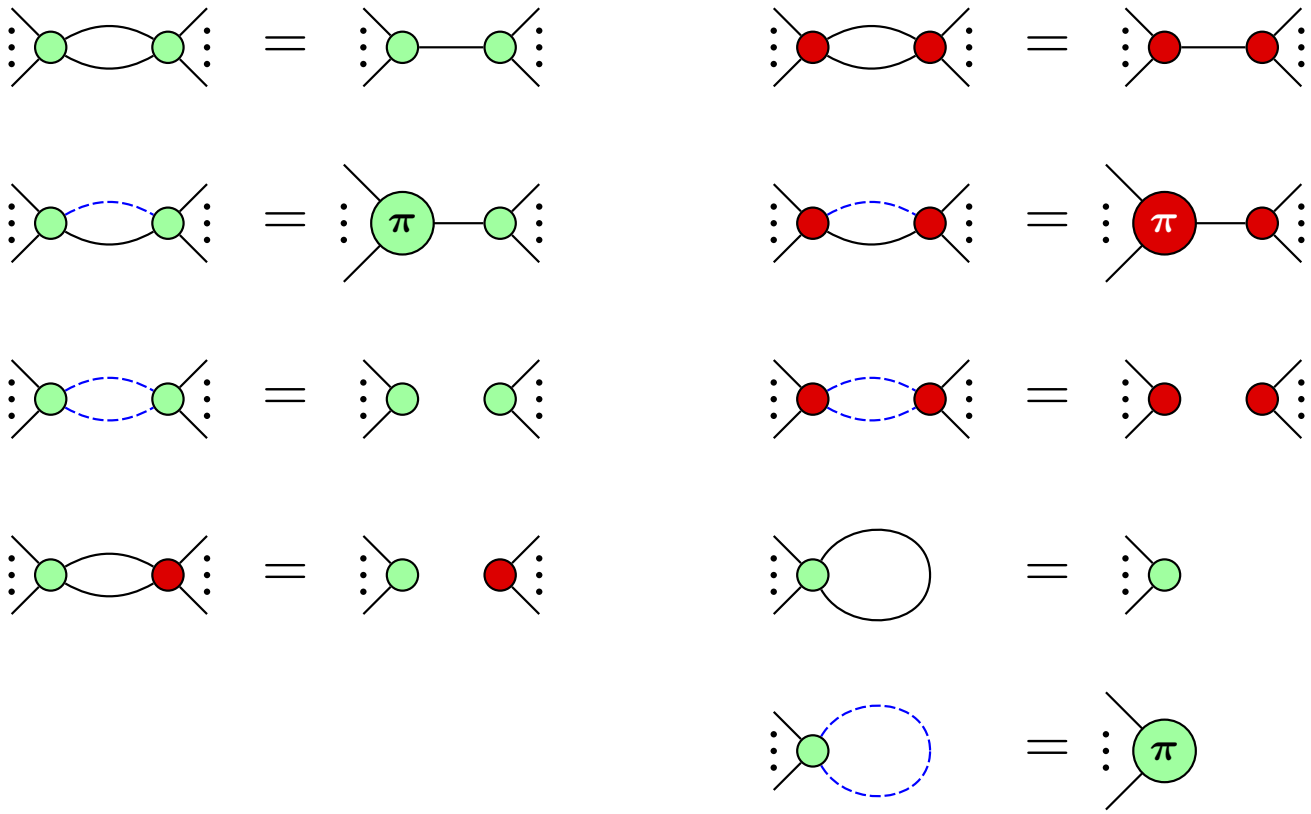
A `Graph` represents a ZX-diagram:

- Three types of vertices: *boundary*, X and Z.
- Phases are stored as rational fractions of π.
- Two types of edges: *regular* and *Hadamard*:



- It is *undirected* and *simple*.

Dealing with parallel edges



Rewrite engine

Hierarchy of rewriting:

- First level: *parallel matcher* and *rewriter*.

Rewrite engine

Hierarchy of rewriting:

- First level: *parallel matcher* and *rewriter*.
- Second: *basic simplifiers* recursively apply such rewrites.

Rewrite engine

Hierarchy of rewriting:

- First level: *parallel matcher* and *rewriter*.
- Second: *basic simplifiers* recursively apply such rewrites.
- Third: these are combined for a more powerful effect.

Rewrite engine

Hierarchy of rewriting:

- First level: *parallel matcher* and *rewriter*.
- Second: *basic simplifiers* recursively apply such rewrites.
- Third: these are combined for a more powerful effect.

Important: simplifiers should be terminating.

Optimization using ZX-diagrams

Duncan, Kissinger, Perdrix, vdW 2019, arXiv:1902.03178
Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus

Kissinger, vdW 2019, arXiv:1903.10477
Reducing T-count with the ZX-calculus

Optimization using ZX-diagrams

Duncan, Kissinger, Perdrix, vdW 2019, arXiv:1902.03178
Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus

Kissinger, vdW 2019, arXiv:1903.10477
Reducing T-count with the ZX-calculus

Summary:

- Write circuit as ZX-diagram.

Optimization using ZX-diagrams

Duncan, Kissinger, Perdrix, vdW 2019, arXiv:1902.03178
Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus

Kissinger, vdW 2019, arXiv:1903.10477
Reducing T-count with the ZX-calculus

Summary:

- Write circuit as ZX-diagram.
- Simplify the diagram (in a smart way).

Optimization using ZX-diagrams

Duncan, Kissinger, Perdrix, vdW 2019, arXiv:1902.03178
Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus

Kissinger, vdW 2019, arXiv:1903.10477
Reducing T-count with the ZX-calculus

Summary:

- Write circuit as ZX-diagram.
- Simplify the diagram (in a smart way).
- *Extract* a circuit from the diagram.

Optimization using ZX-diagrams

Duncan, Kissinger, Perdrix, vdW 2019, arXiv:1902.03178
Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus

Kissinger, vdW 2019, arXiv:1903.10477
Reducing T-count with the ZX-calculus

Summary:

- Write circuit as ZX-diagram.
- Simplify the diagram (in a smart way).
- *Extract* a circuit from the diagram.

Goal: Minimize amount of T gates in circuit.

Demonstration time

Circuit	n	T	Best	Method	PyZX	PyZX +TODD
adder$_8$	**24**	**399**	**213**	**RM$_m$**	**173**	**167**
Adder8	23	266	56	NRSCM	56	56
Adder16	47	602	120	NRSCM	120	120
Adder32	95	1274	248	NRSCM	248	248
Adder64	191	2618	504	NRSCM	504	504
csla-mux$_3$	*15*	*70*	*58*	*RM$_r$*	*62*	**45**
csum-mux$_9$	*30*	*196*	*76*	*RM$_r$*	*84*	**72**
cycle17$_3$	**35**	**4739**	**1944**	**RM$_m$**	**1797**	**1797**
gf(2^4)-mult	*12*	*112*	*56*	*TODD*	*68*	**52**
gf(2^5)-mult	*15*	*175*	*90*	*TODD*	*115*	**86**
gf(2^6)-mult	*18*	*252*	*132*	*TODD*	*150*	**122**
gf(2^7)-mult	*21*	*343*	*185*	*TODD*	*217*	**173**
gf(2^8)-mult	*24*	*448*	*216*	*TODD*	*264*	**214**
ham15-low	17	161	97	Tpar	97	97
ham15-med	**17**	**574**	**230**	**Tpar**	**212**	**212**
ham15-high	**20**	**2457**	**1019**	**Tpar**	**1019**	**1013**
hwb$_6$	**7**	**105**	**75**	**Tpar**	**75**	**72**
hwb$_8$	**12**	**5887**	**3531**	**RM$_{m\&r}$**	**3517**	**3501**
mod-mult-55	*9*	*49*	*28*	*TODD*	*35*	**20**
mod-red-21	11	119	73	Tpar	73	73
mod5$_4$	**5**	**28**	**16**	**Tpar**	**8**	**7**
nth-prime$_6$	**9**	**567**	**400**	**RM$_{m\&r}$**	**279**	**279**
nth-prime$_8$	*12*	*6671*	*4045*	*RM$_{m\&r}$*	*4047*	**3958**
qcla-adder$_{10}$	36	589	162	Tpar	162	**158**
qcla-com$_7$	*24*	*203*	*94*	*RM$_m$*	*95*	**91**
qcla-mod$_7$	26	413	237	NRSCM	237	**216**
rc-adder$_6$	14	77	47	RM$_{m\&r}$	47	47
vbe-adder$_3$	10	70	24	Tpar	24	24

Validation

Problem: How do we know our optimized circuits are correct?

Validation

Problem: How do we know our optimized circuits are correct?

- Direct tensor calculation (using NumPy)

Validation

Problem: How do we know our optimized circuits are correct?

- Direct tensor calculation (using NumPy)
- Use the rewrite engine!

Validation

Problem: How do we know our optimized circuits are correct?

- Direct tensor calculation (using NumPy)
- Use the rewrite engine!

Using the latter, correctness was verified for all our circuits.

Current & Future work

- Qubit routing for restricted topologies
  cf. Arianne Meijer-van de Griend's talk Thursday!

Current & Future work

- Qubit routing for restricted topologies
  cf. Arianne Meijer-van de Griend's talk Thursday!
- ZH-diagrammatic rewriting to reason about Toffoli circuits.

Current & Future work

- Qubit routing for restricted topologies
  cf. Arianne Meijer-van de Griend's talk Thursday!
- ZH-diagrammatic rewriting to reason about Toffoli circuits.
- Quantum circuit simulation with ZX-diagrams.

Current & Future work

- Qubit routing for restricted topologies
  cf. Arianne Meijer-van de Griend's talk Thursday!
- ZH-diagrammatic rewriting to reason about Toffoli circuits.
- Quantum circuit simulation with ZX-diagrams.

Interesting challenges:

- Optimization using auxiliary qubits

Current & Future work

- Qubit routing for restricted topologies
  cf. Arianne Meijer-van de Griend's talk Thursday!
- ZH-diagrammatic rewriting to reason about Toffoli circuits.
- Quantum circuit simulation with ZX-diagrams.

Interesting challenges:

- Optimization using auxiliary qubits
- Compiling directly to lattice surgery procedure
  (cf. previous talk)

Thank you for your attention!

github.com/Quantomatic/pyzx zxcalculus.com/pyzx