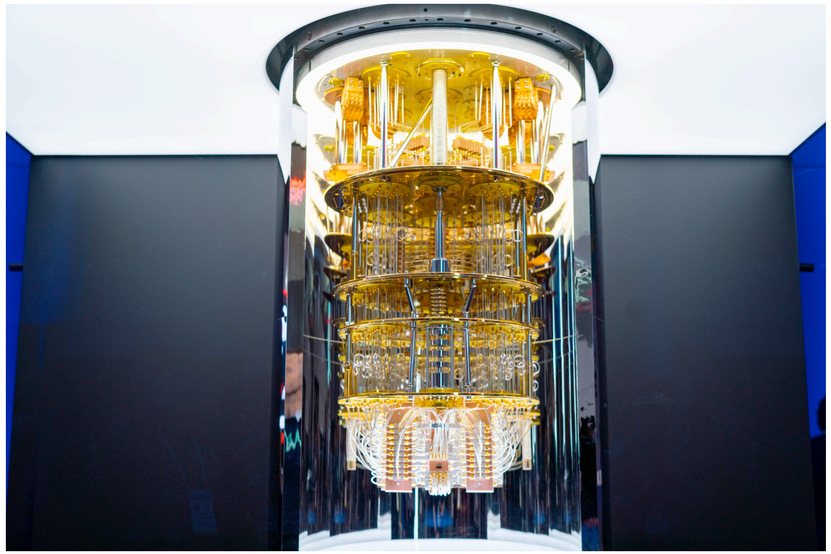
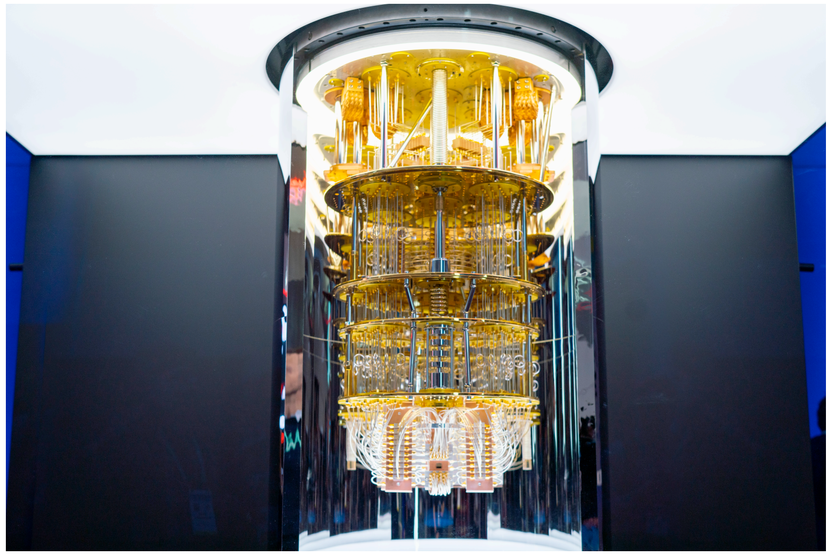
Quantum-Circuit Optimization

Native Basis Gate Set



IBM

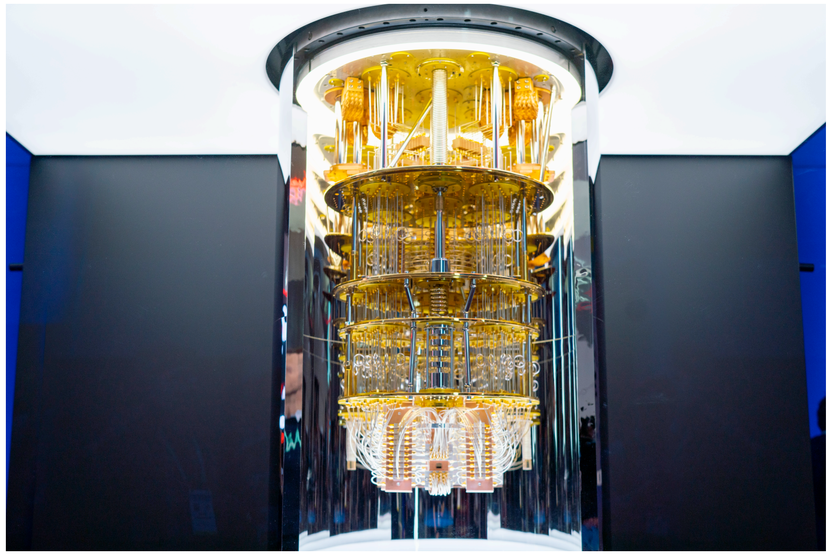
Native Basis Gate Set



IBM

$$R_z^{\theta}$$

Native Basis Gate Set

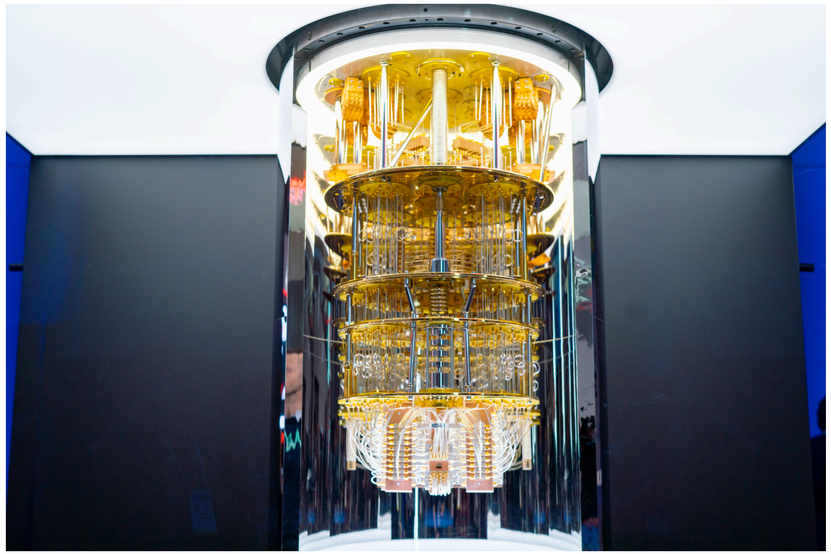


IBM

R_z^θ

SX

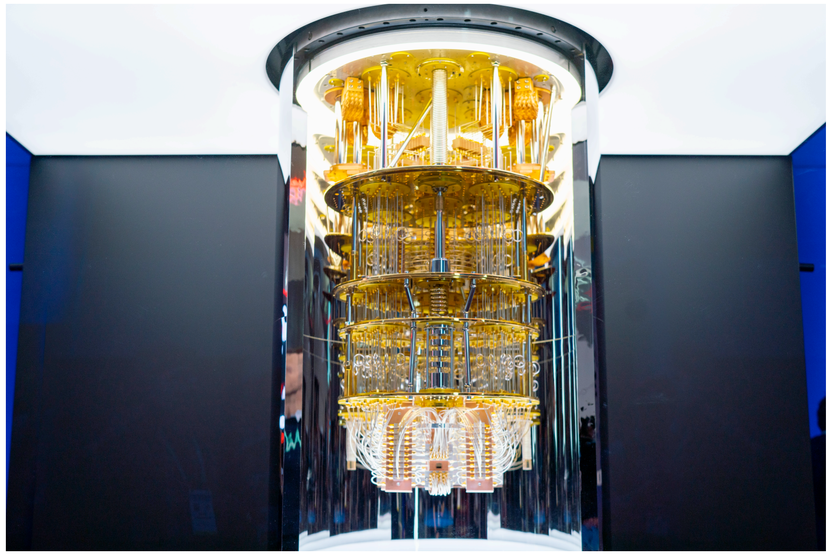
Native Basis Gate Set



IBM

R_z^θ SX X

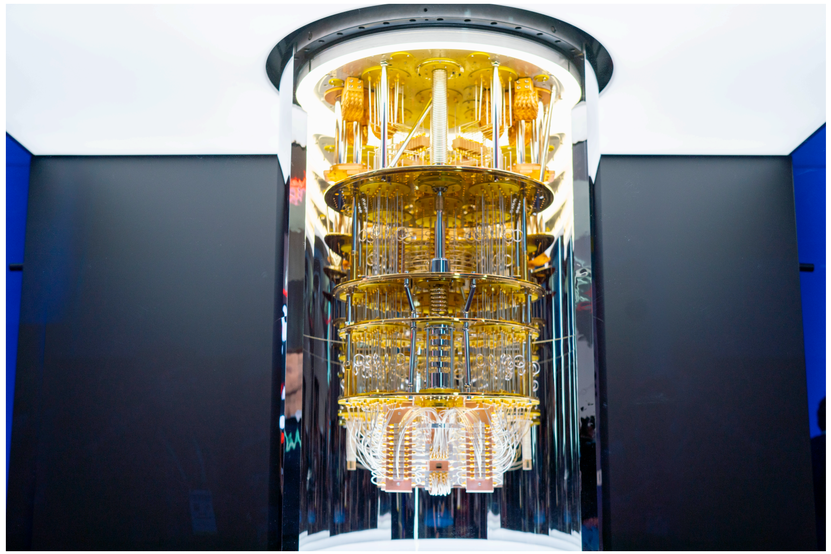
Native Basis Gate Set



IBM

R_z^θ SX X CX

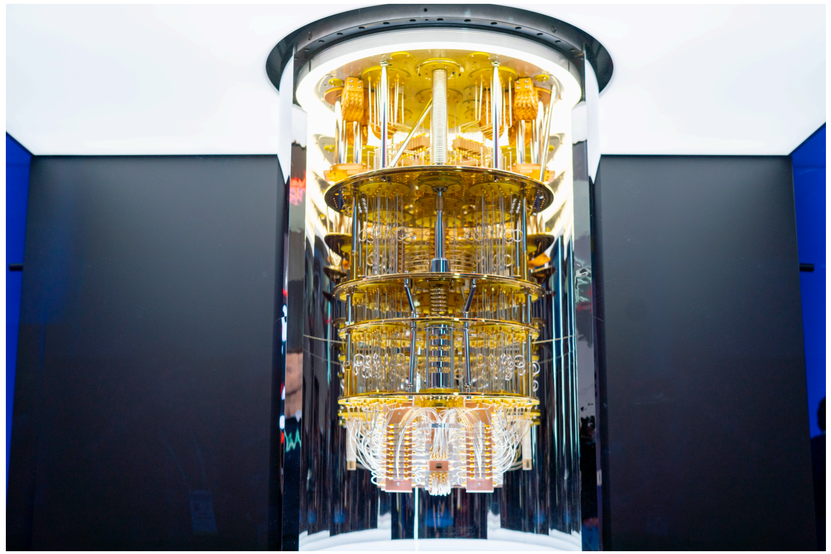
Native Basis Gate Set



IBM

$\{ R_z^\theta , SX , X , CX \}$

Native Basis Gate Set

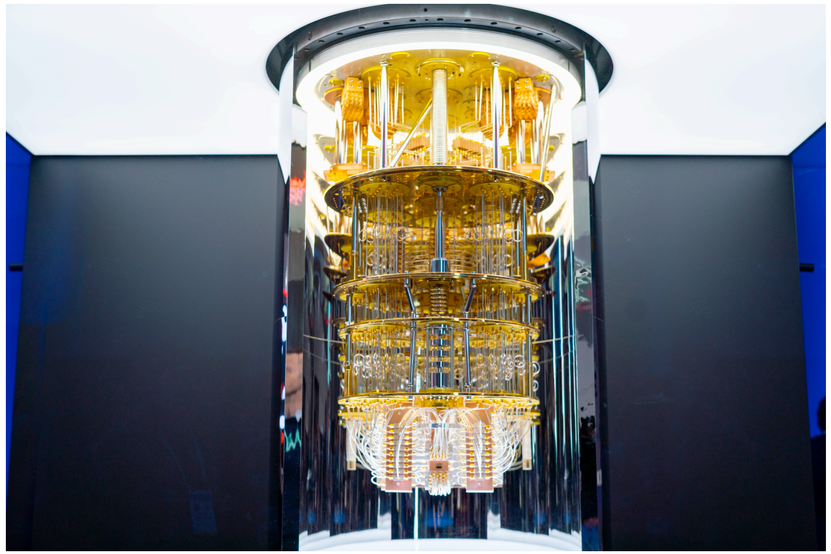


IBM

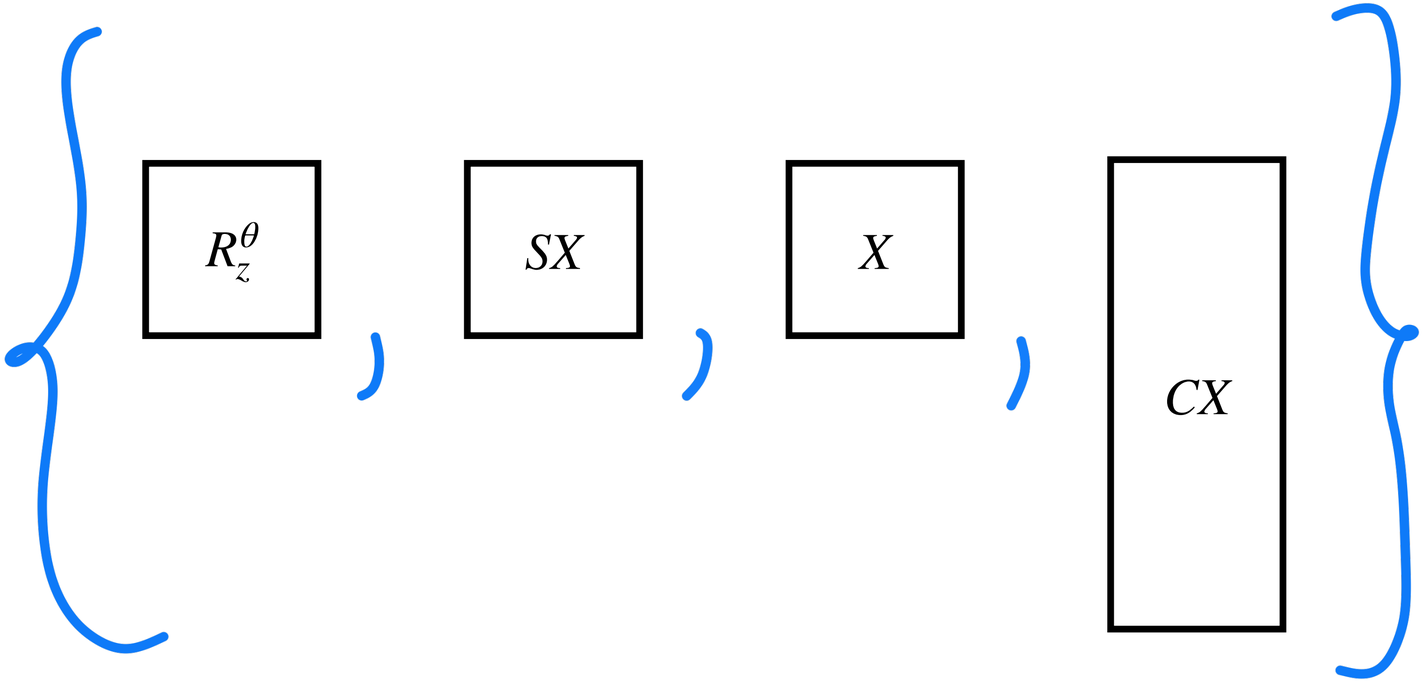
R_z^θ , SX , X , CX

"universal"

Native Basis Gate Set



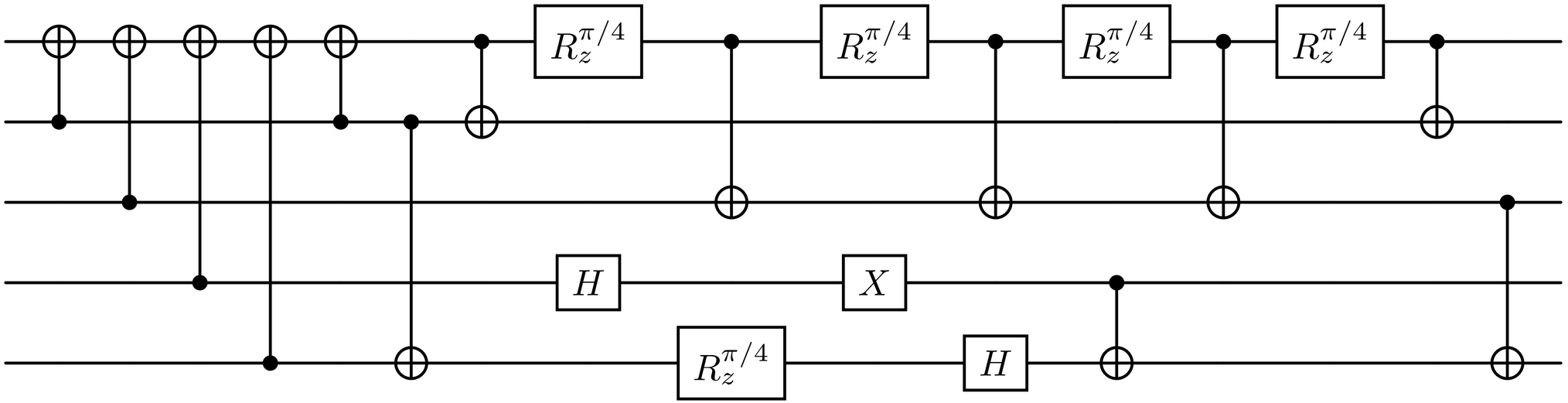
IBM



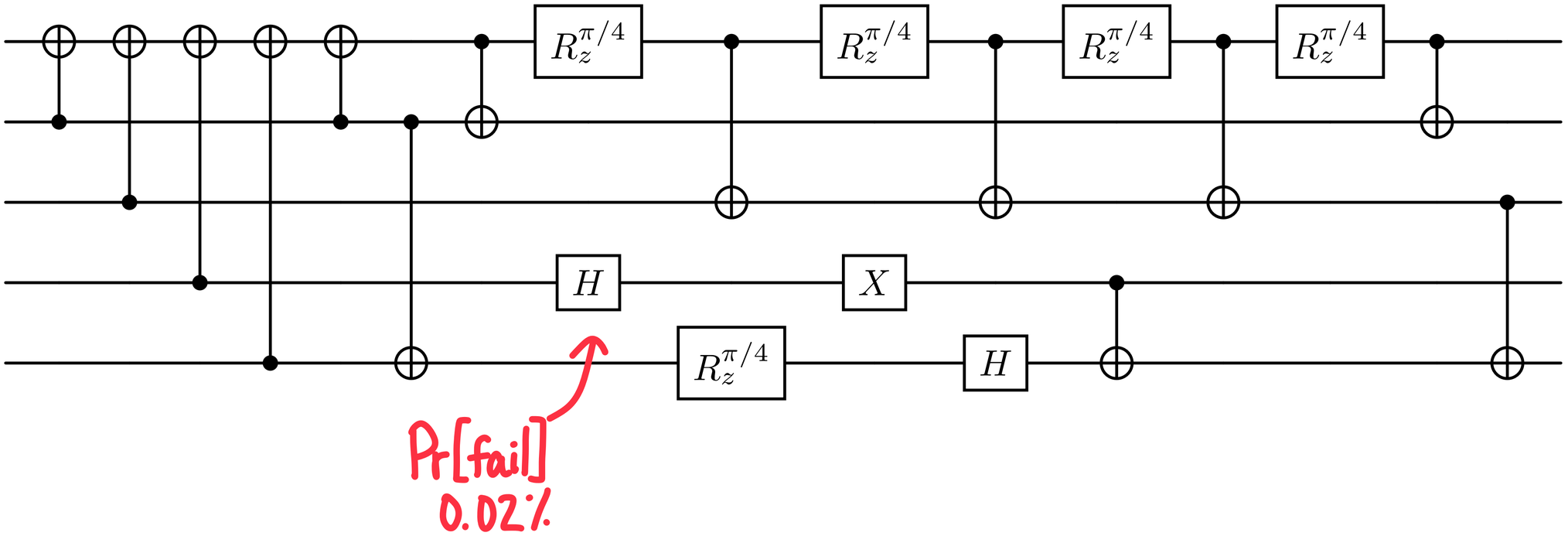
"universal"

i.e. approximate any computation to arbitrary precision

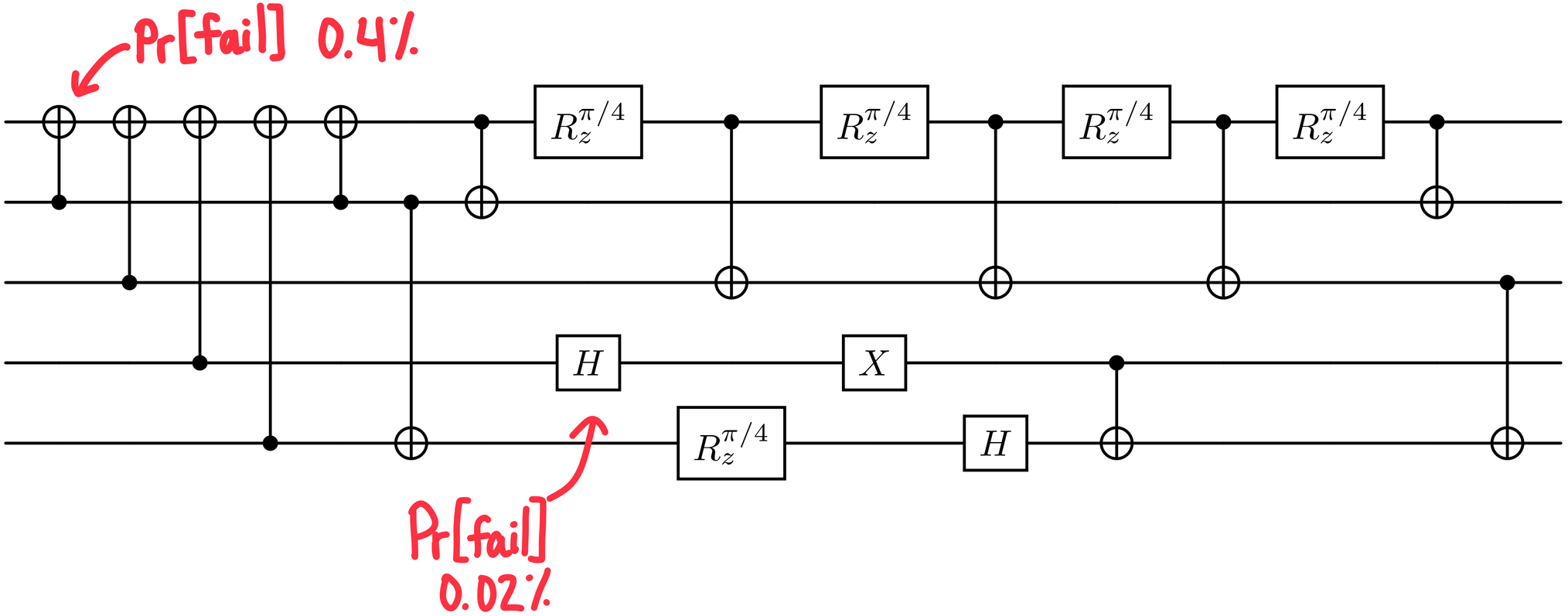
Optimization Objectives



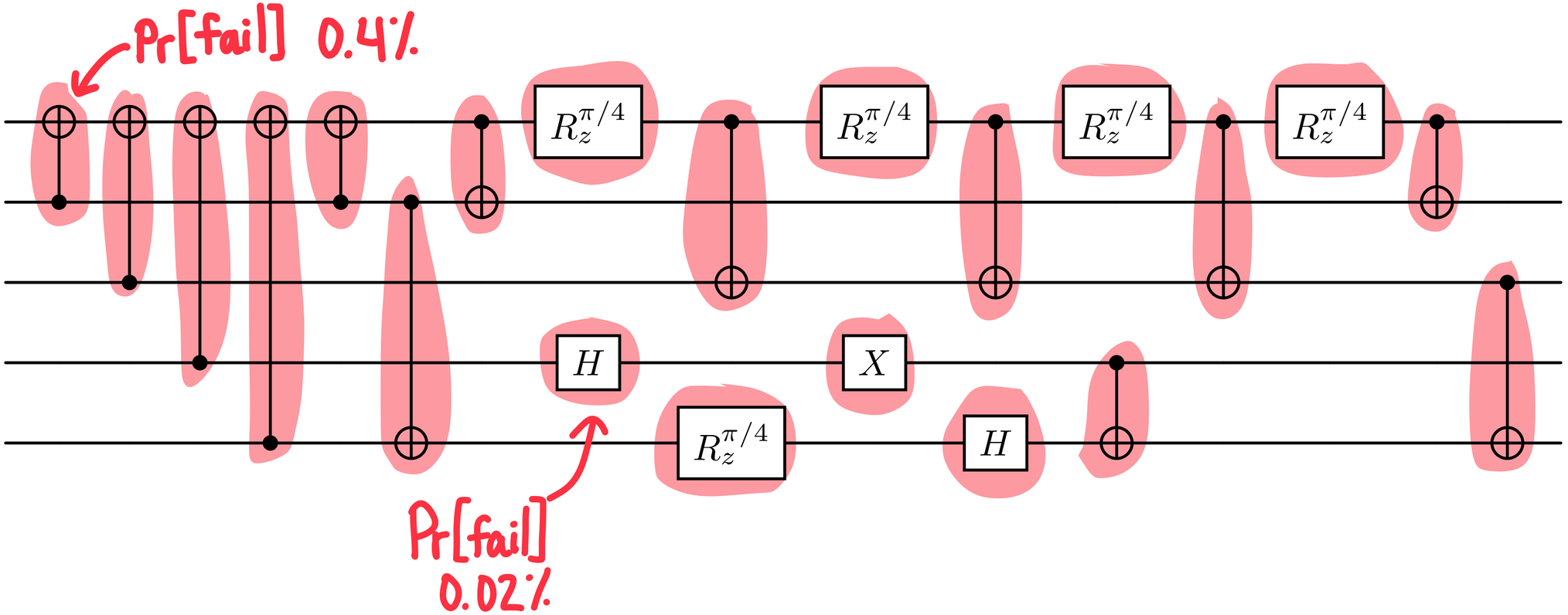
Optimization Objectives



Optimization Objectives



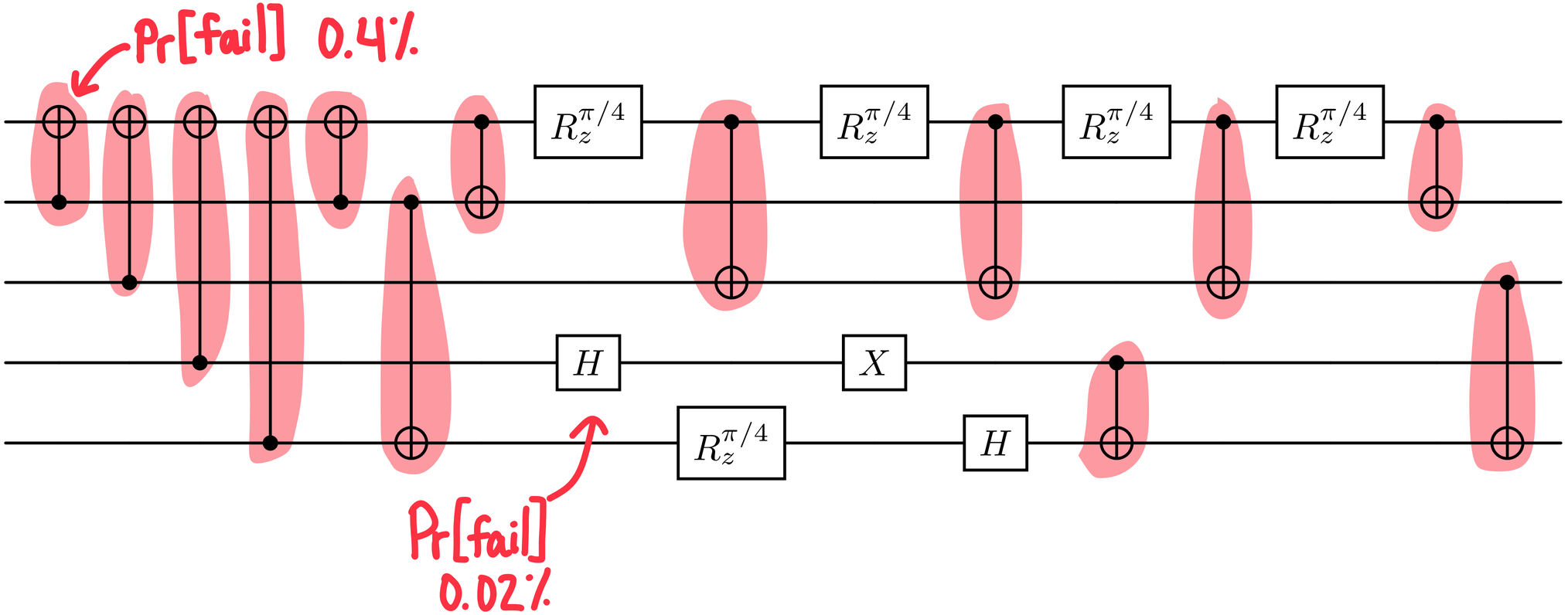
Optimization Objectives



NISQ

- total gate count

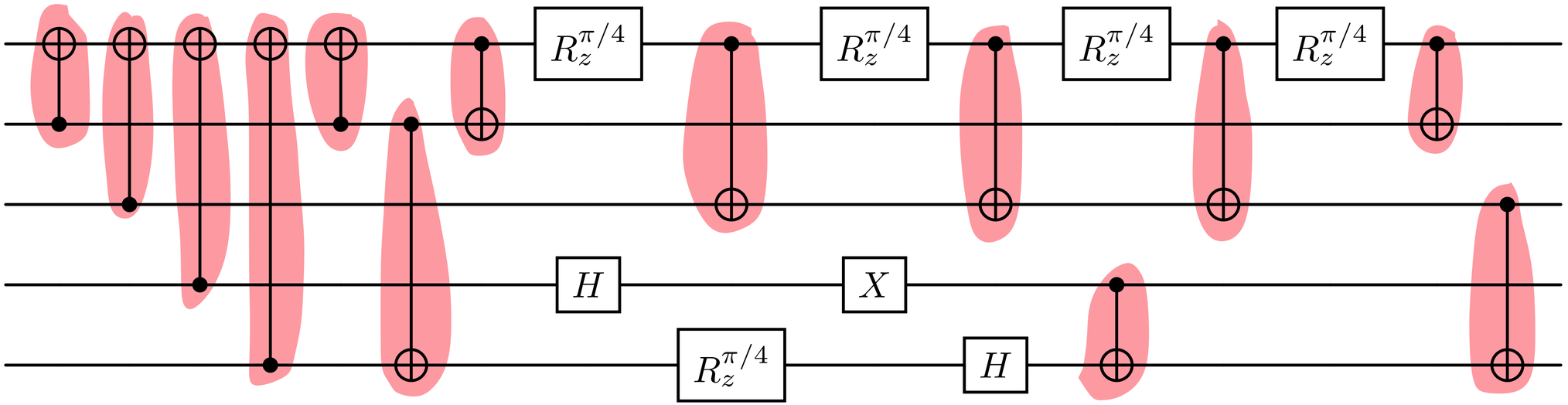
Optimization Objectives



NISQ

- total gate count
- 2q gate count

Optimization Objectives



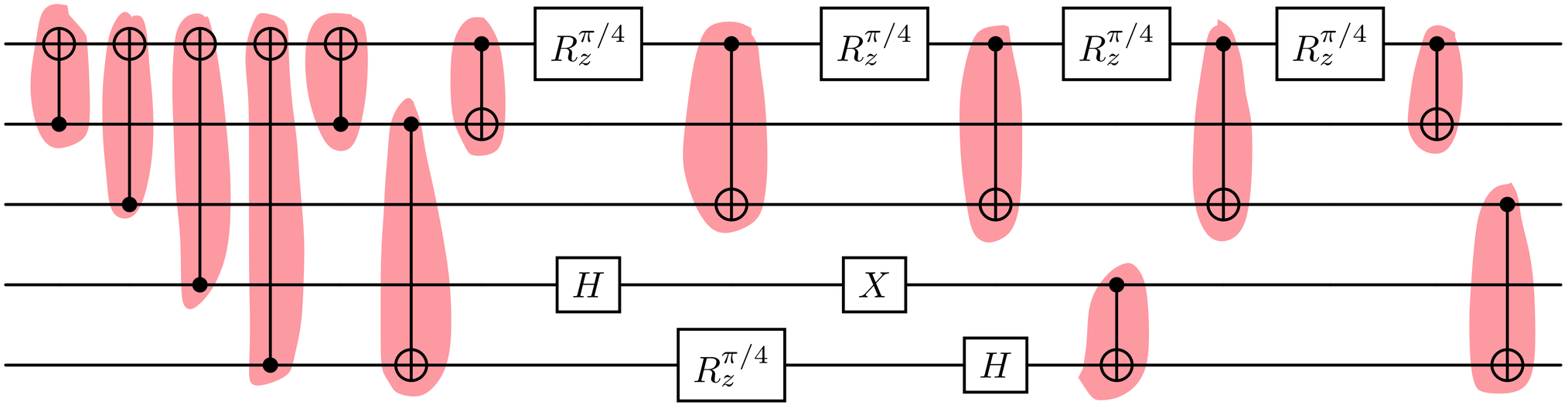
NISQ

- total gate count
- 2q gate count

FTQC

- T count and 2q gate count

Optimization Objectives



$$T = R_z^{\pi/4}$$

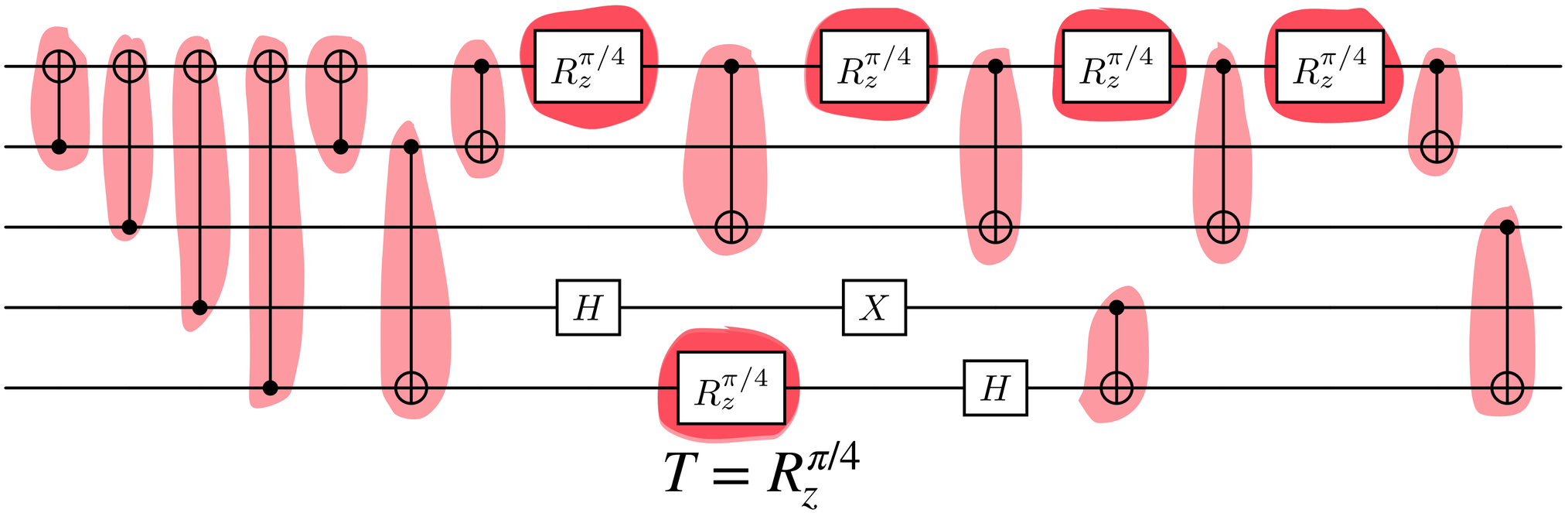
NISQ

- total gate count
- 2q gate count

FTQC

- T count and 2q gate count

Optimization Objectives



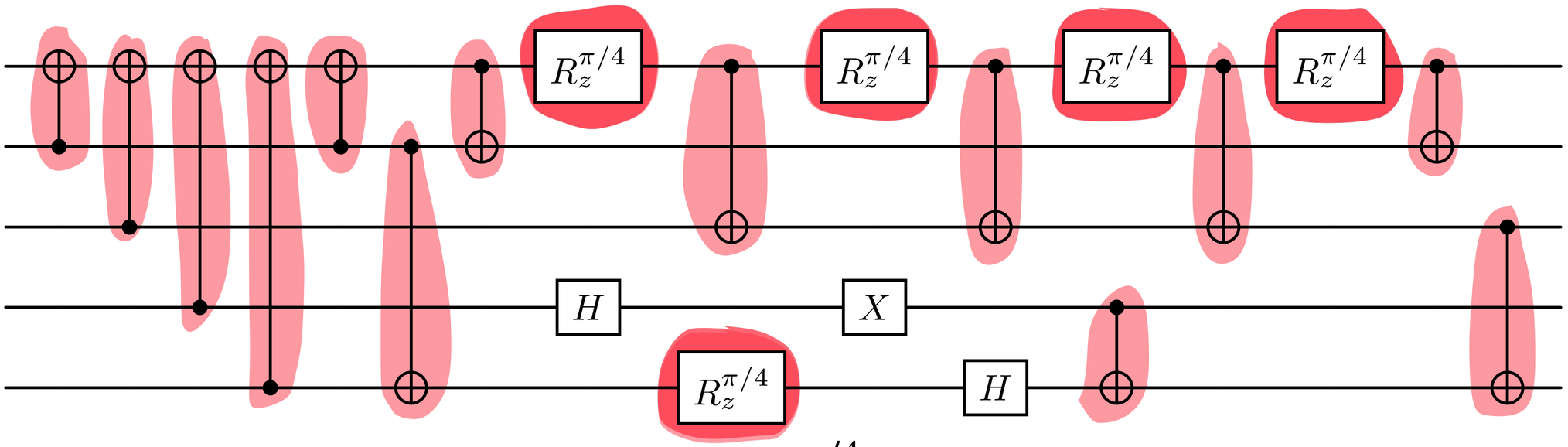
NISQ

- total gate count
- 2q gate count

FTQC

- T count and 2q gate count

Optimization Objectives



NISQ

- total gate count
- 2q gate count

FTQC

- T count and 2q gate count

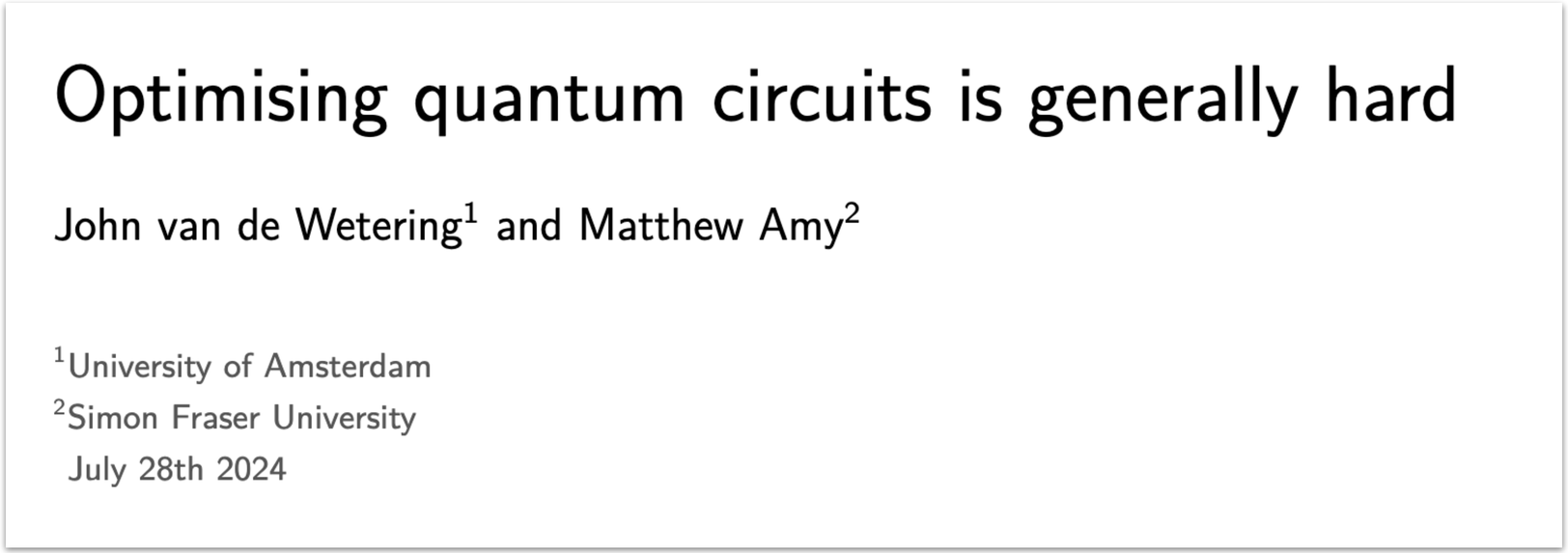
and many more...

High-level problem

Given a circuit and optimization objective, output an equivalent circuit that minimizes the optimization objective.

High-level problem

Given a circuit and optimization objective, output an equivalent circuit that minimizes the optimization objective.

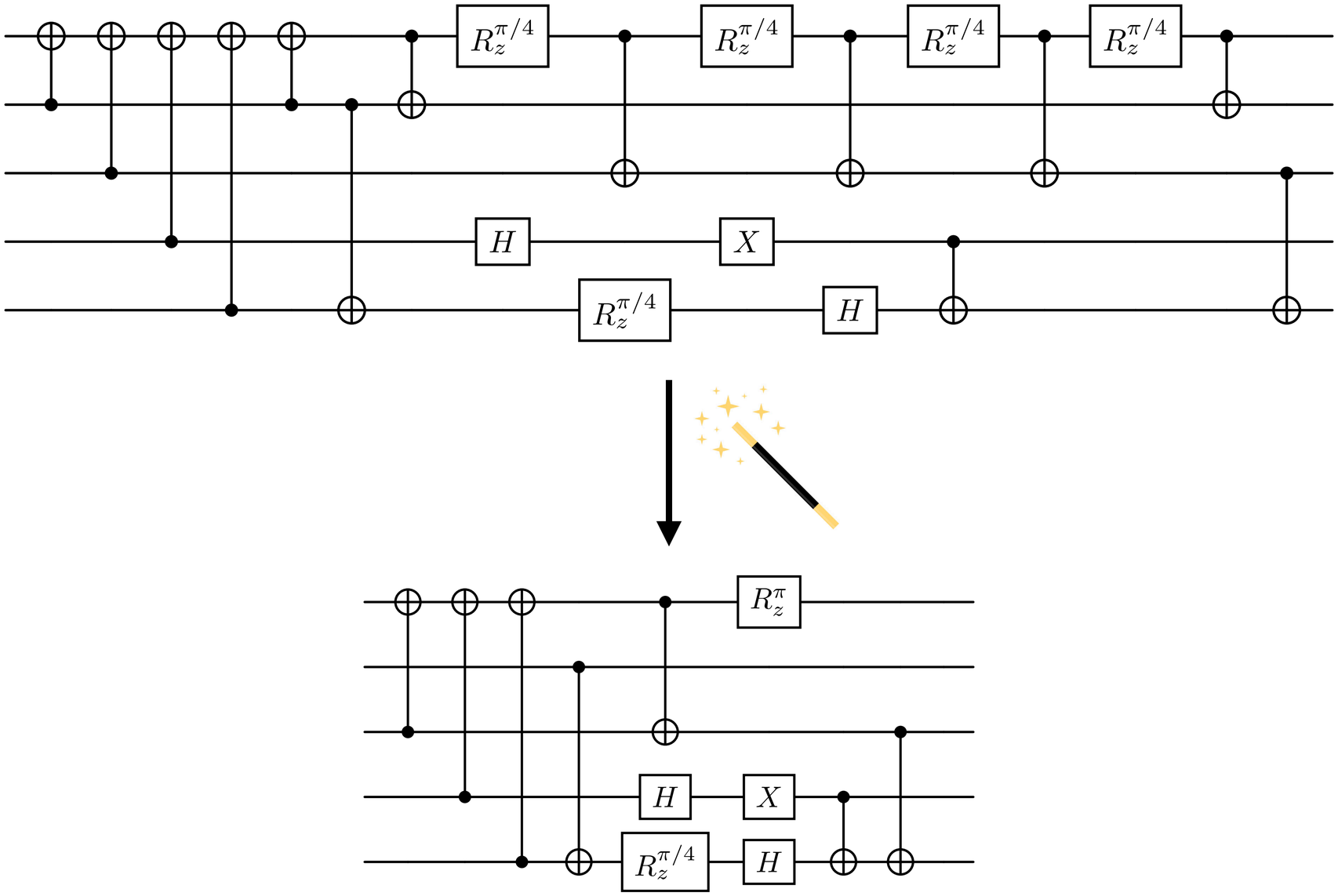


Optimising quantum circuits is generally hard

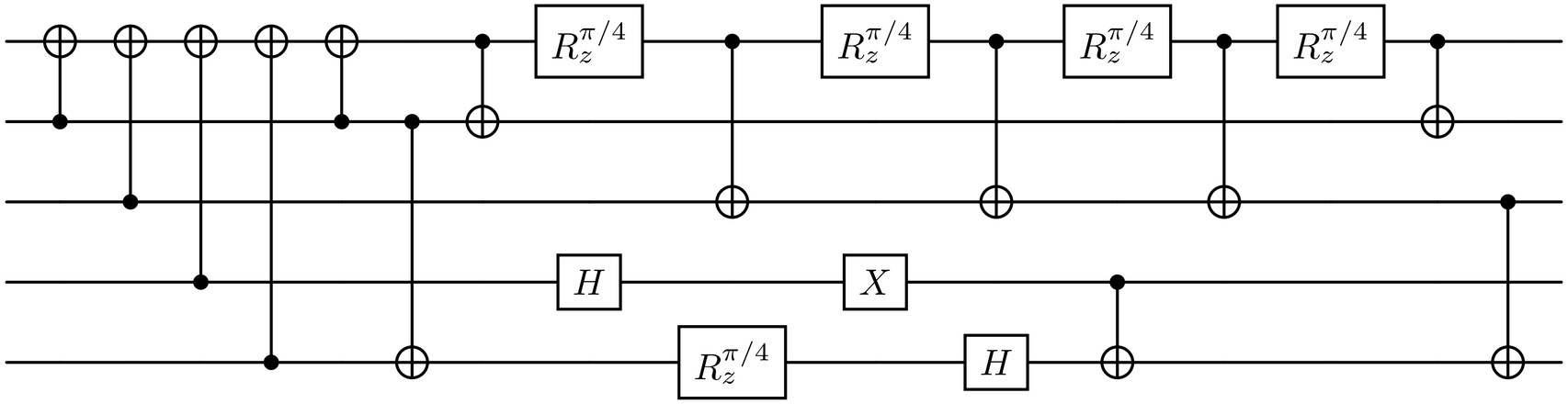
John van de Wetering[1] and Matthew Amy[2]

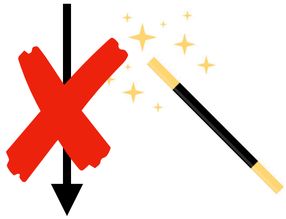
[1]University of Amsterdam
[2]Simon Fraser University
July 28th 2024

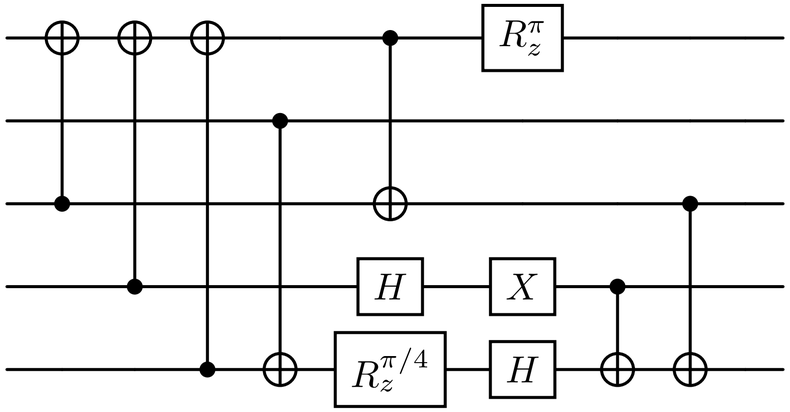
General Approach



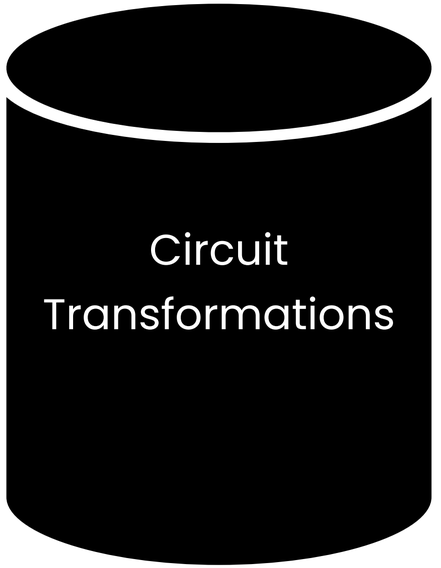
General Approach

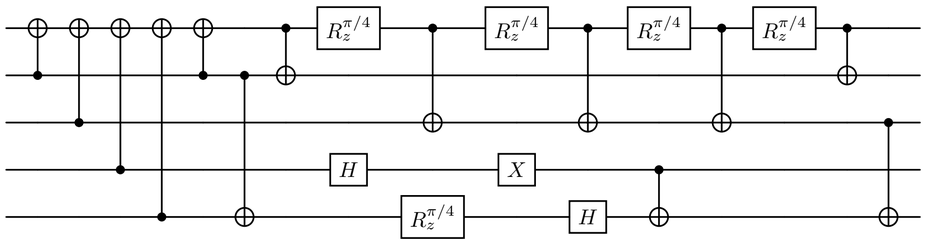




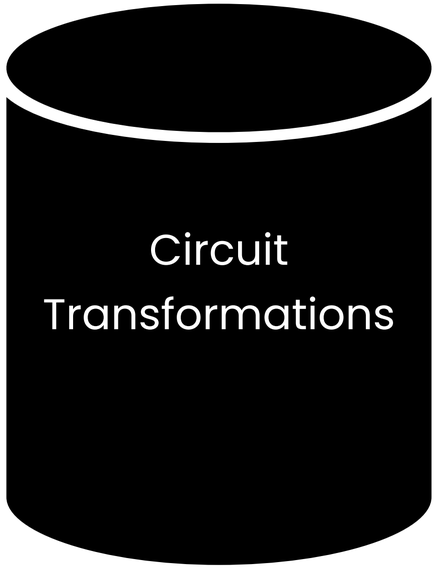


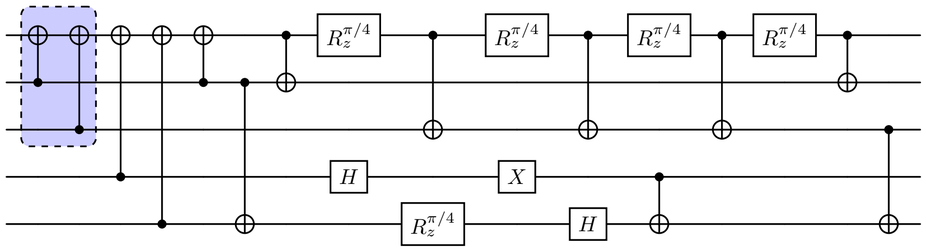
General Approach



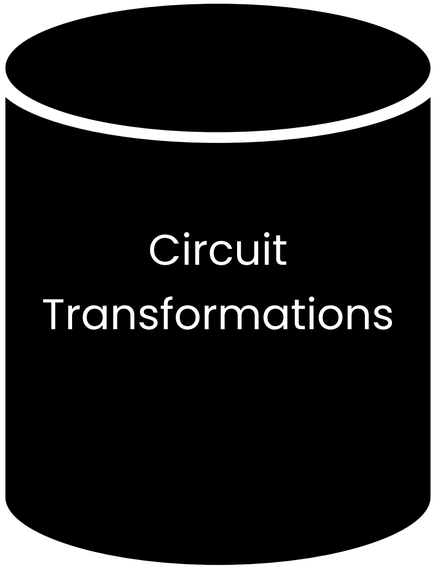


General Approach

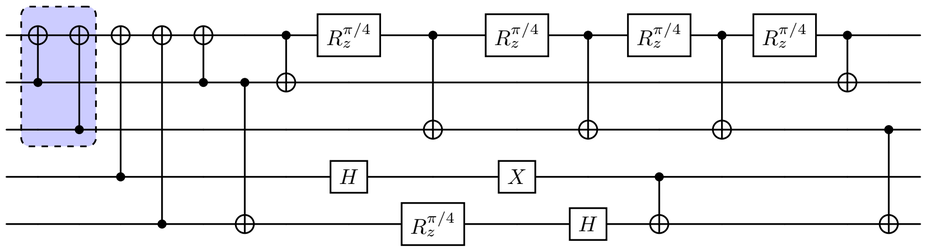




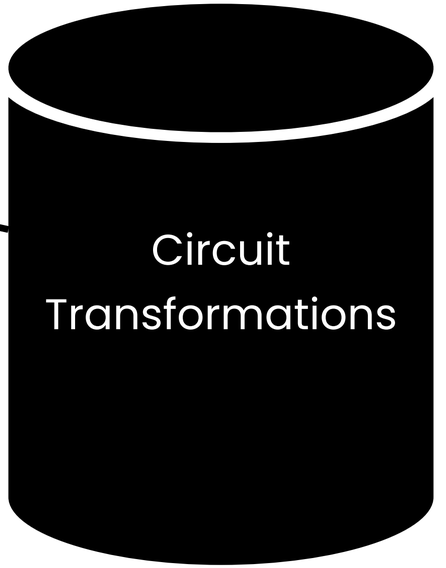
General Approach



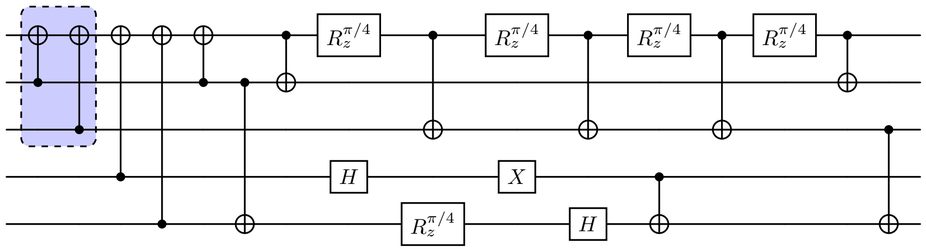
"subcircuit"



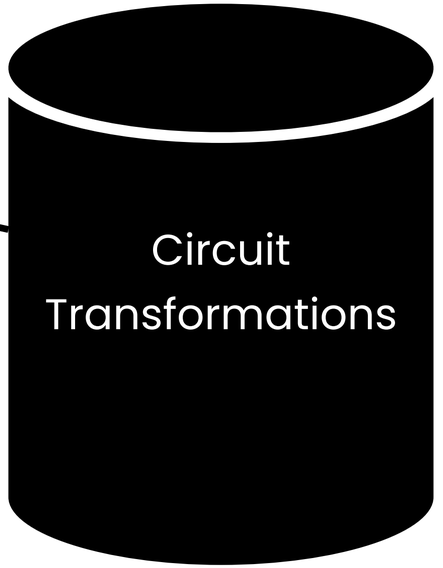
General Approach



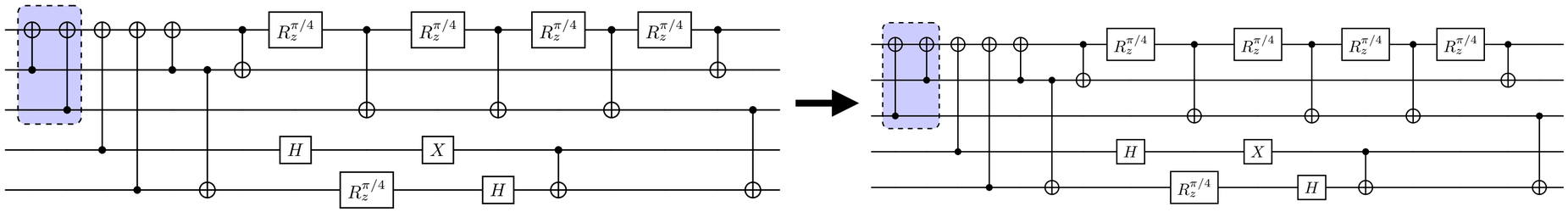
"subcircuit"



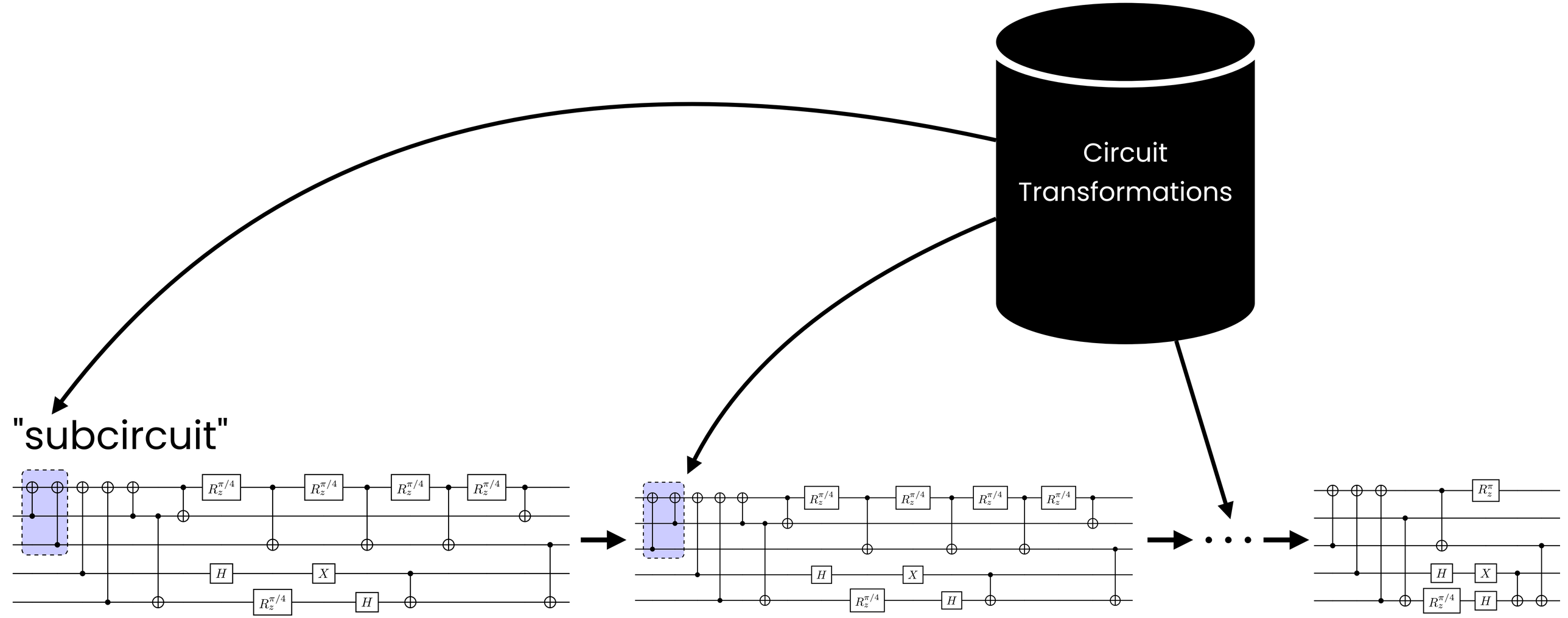
General Approach



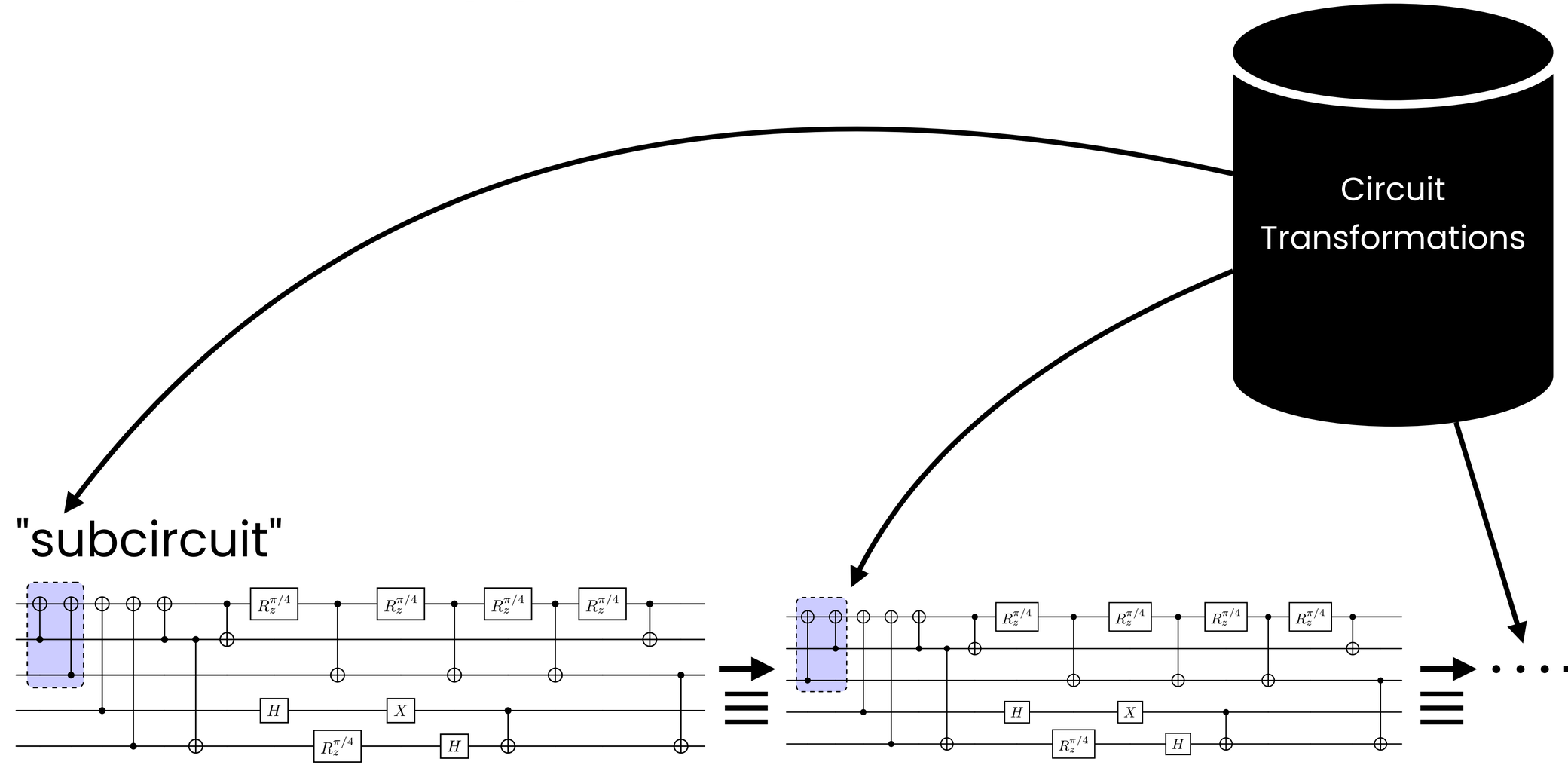
"subcircuit"



General Approach



General Approach



Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

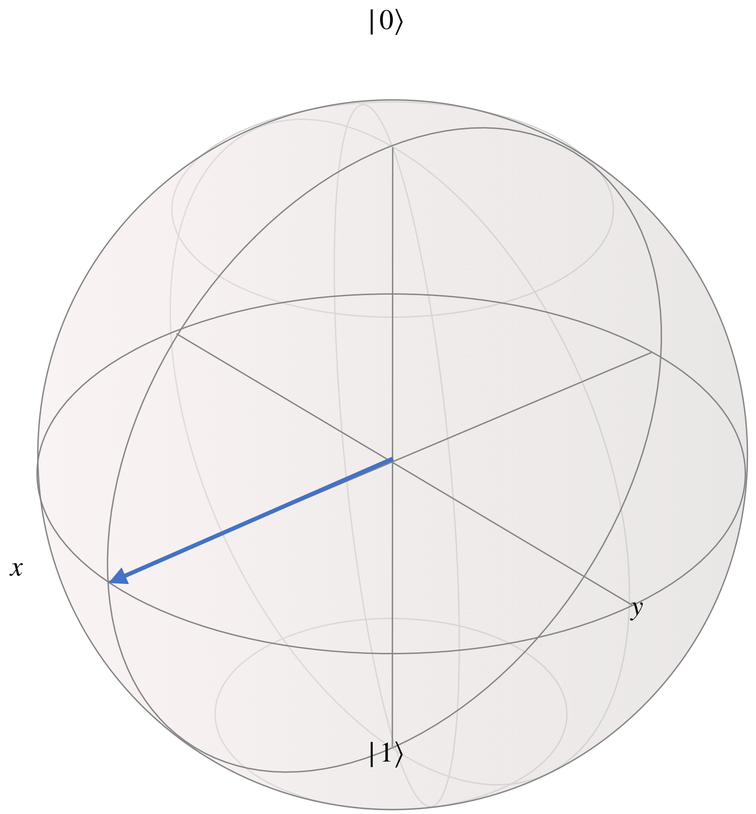
"global phase"

Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

"global phase"

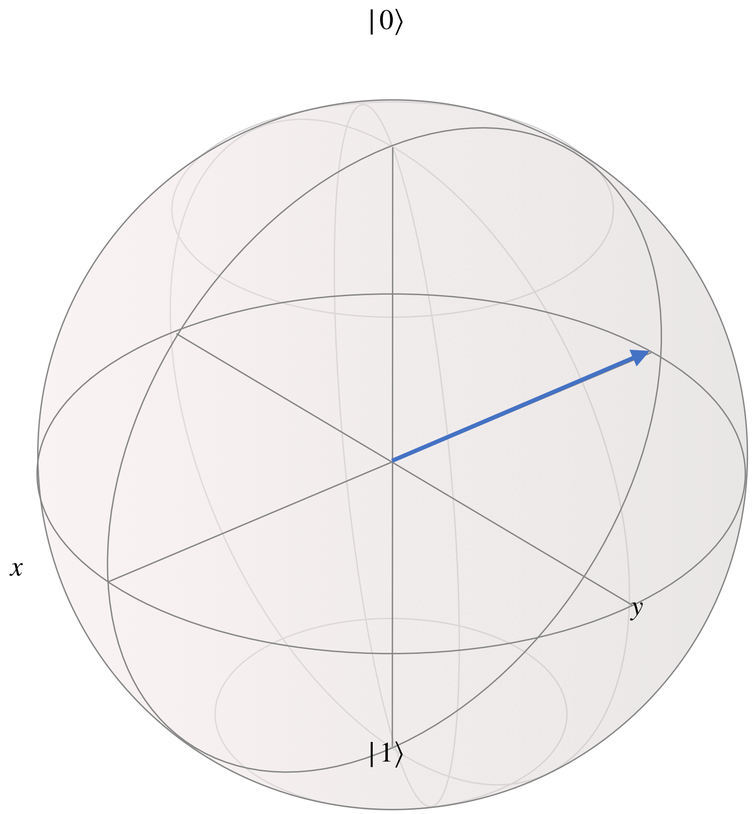


Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

"global phase"

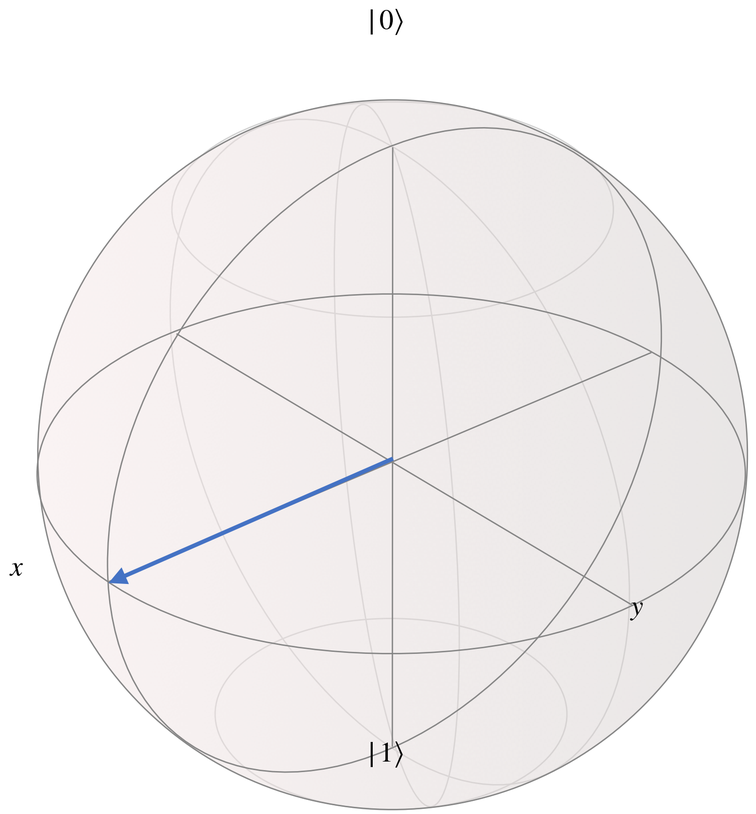


Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

"global phase"



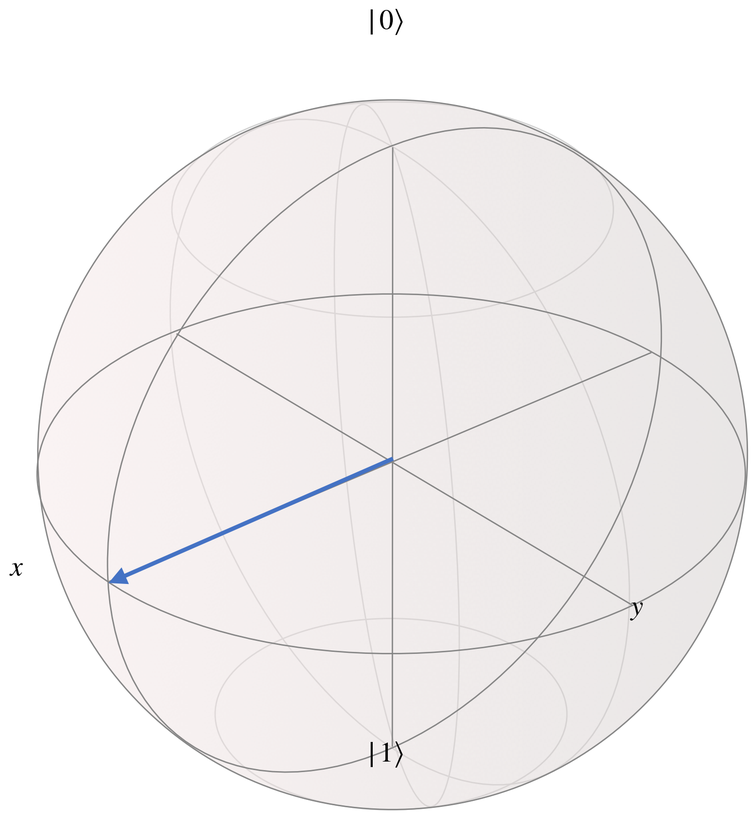
Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

"global phase"

$$C_1 \equiv C_2 \iff \exists \varphi \,.\, U_{C_1} = e^{i\varphi} U_{C_2}$$



Circuit Equivalence

Circuits are unitary matrices: $U^\dagger U = UU^\dagger = I$

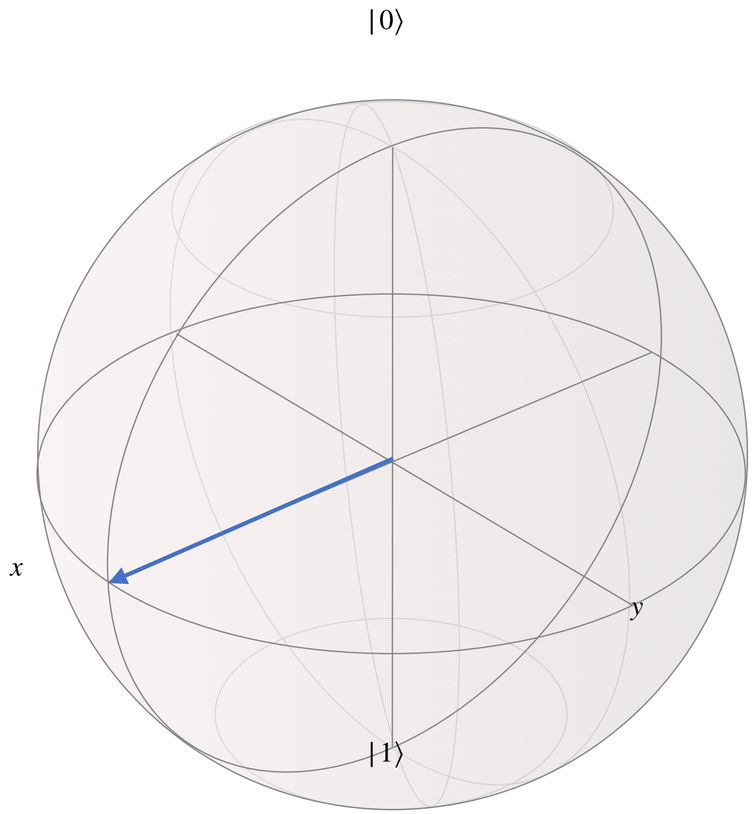
$$C_1 \equiv C_2 \iff U_{C_1} = U_{C_2}$$

"global phase"

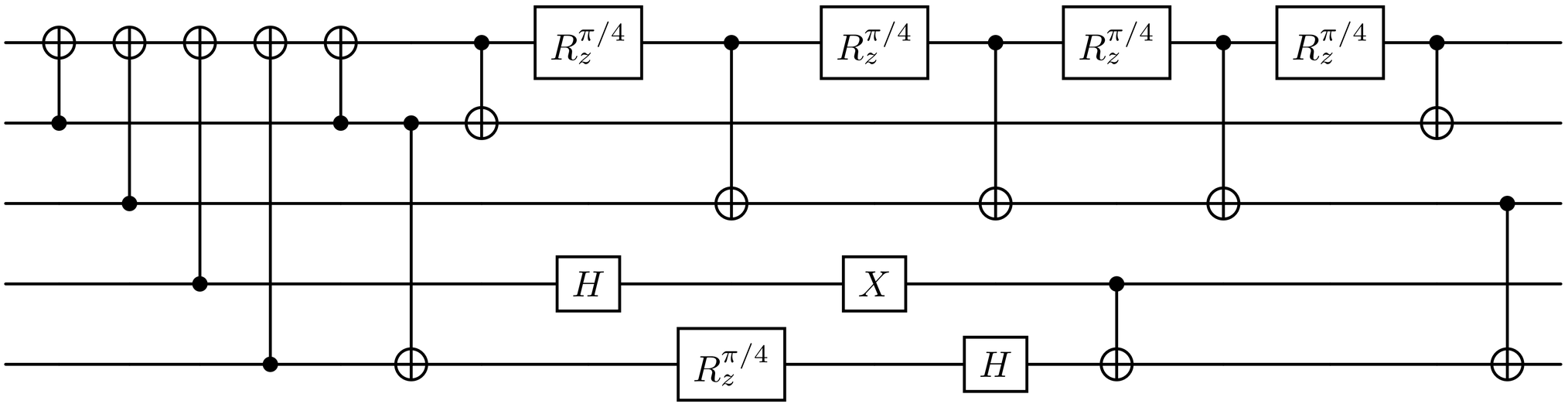
$$C_1 \equiv C_2 \iff \exists \varphi \,.\, U_{C_1} = e^{i\varphi} U_{C_2}$$

Hard for even a quantum computer to verify

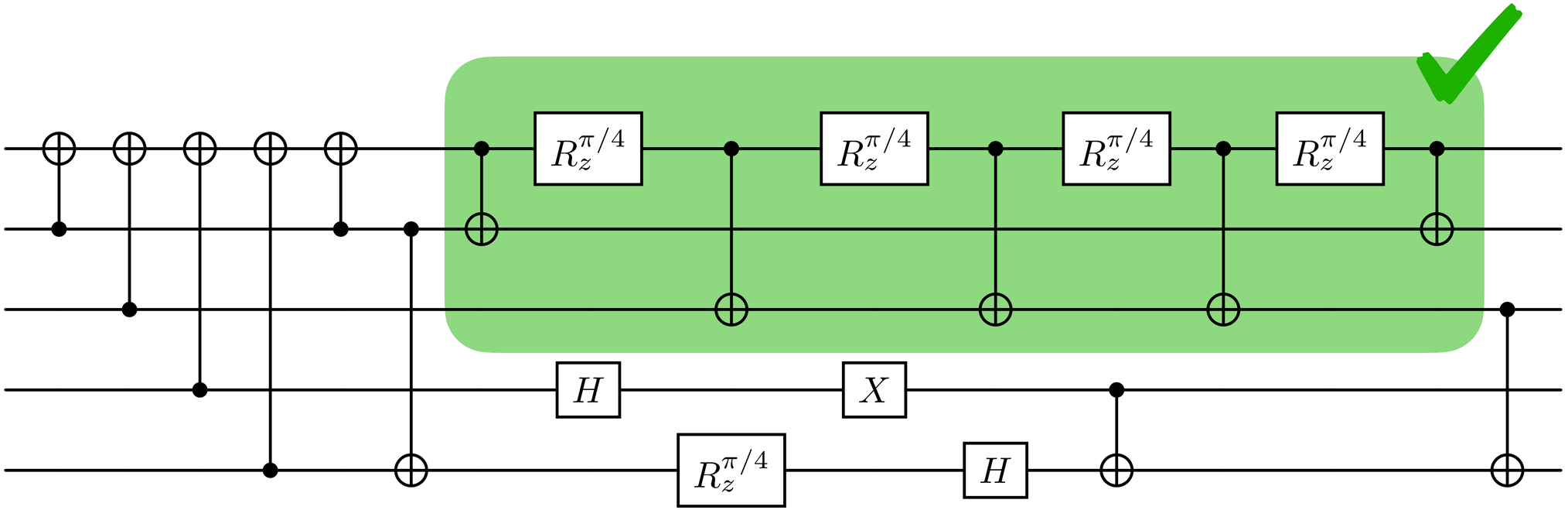
(QMA-Complete)



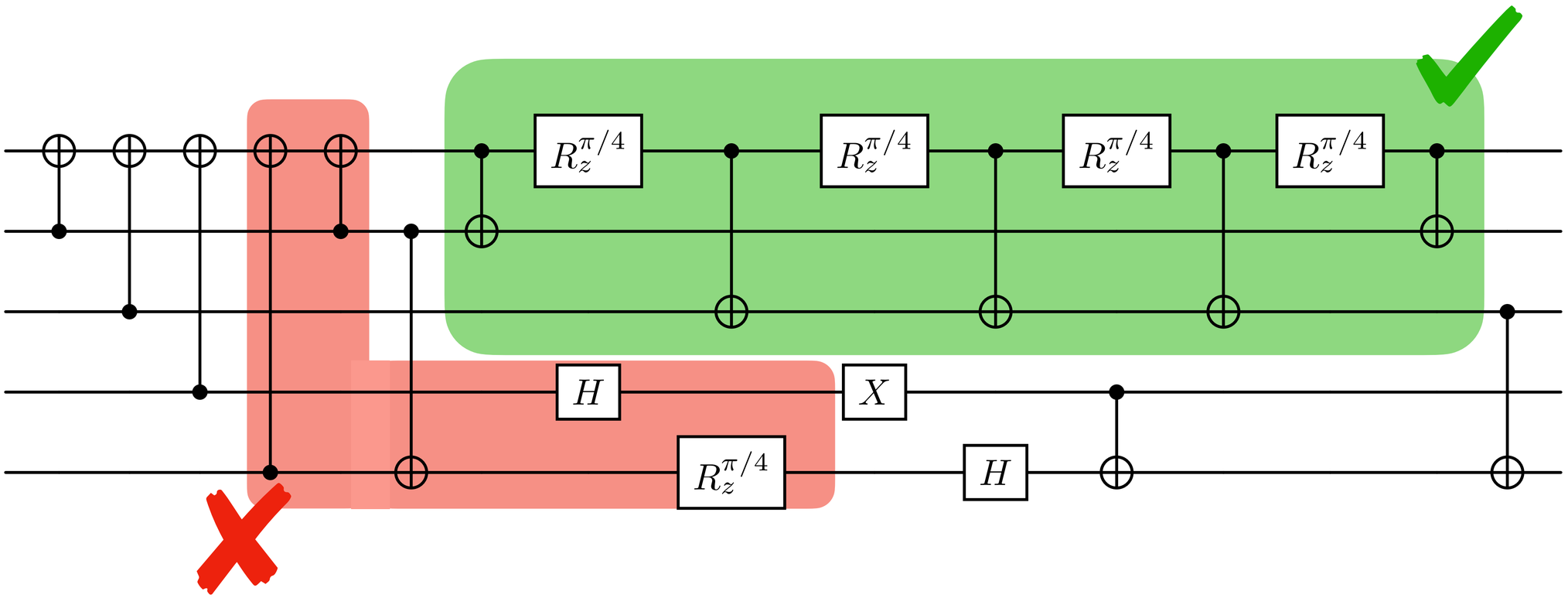
Subcircuit



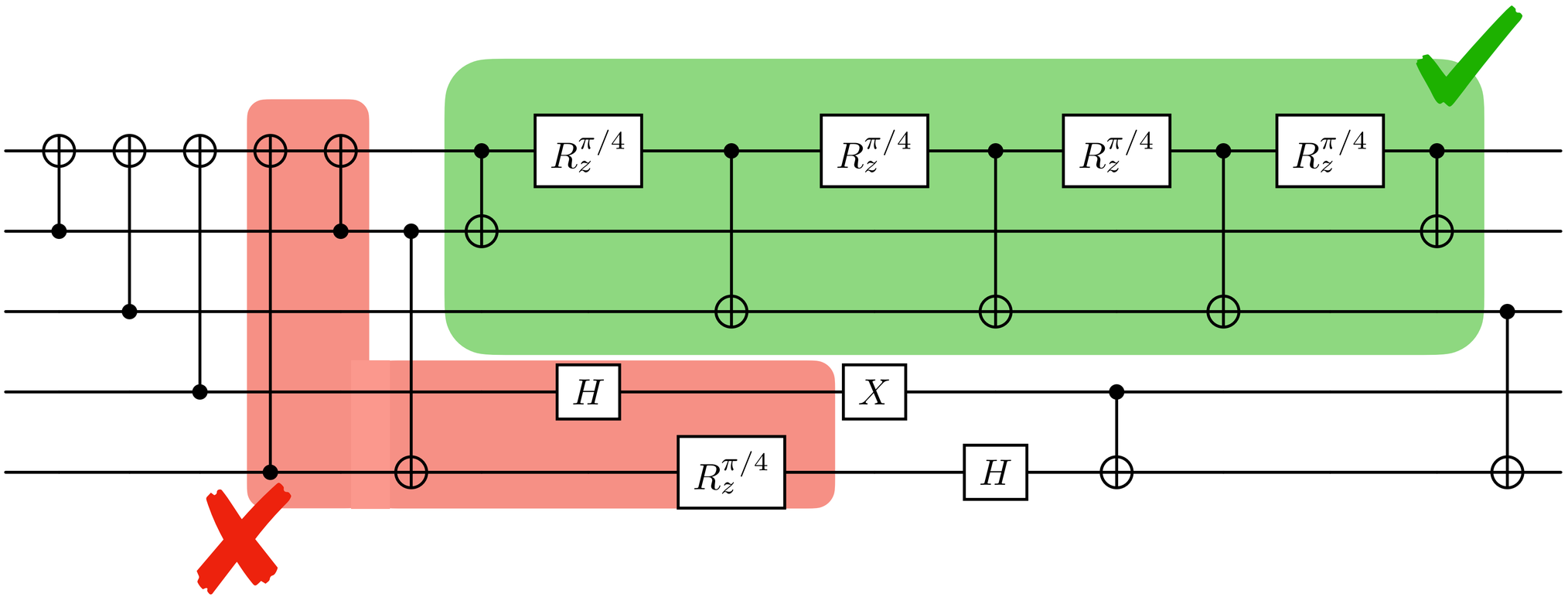
Subcircuit



Subcircuit

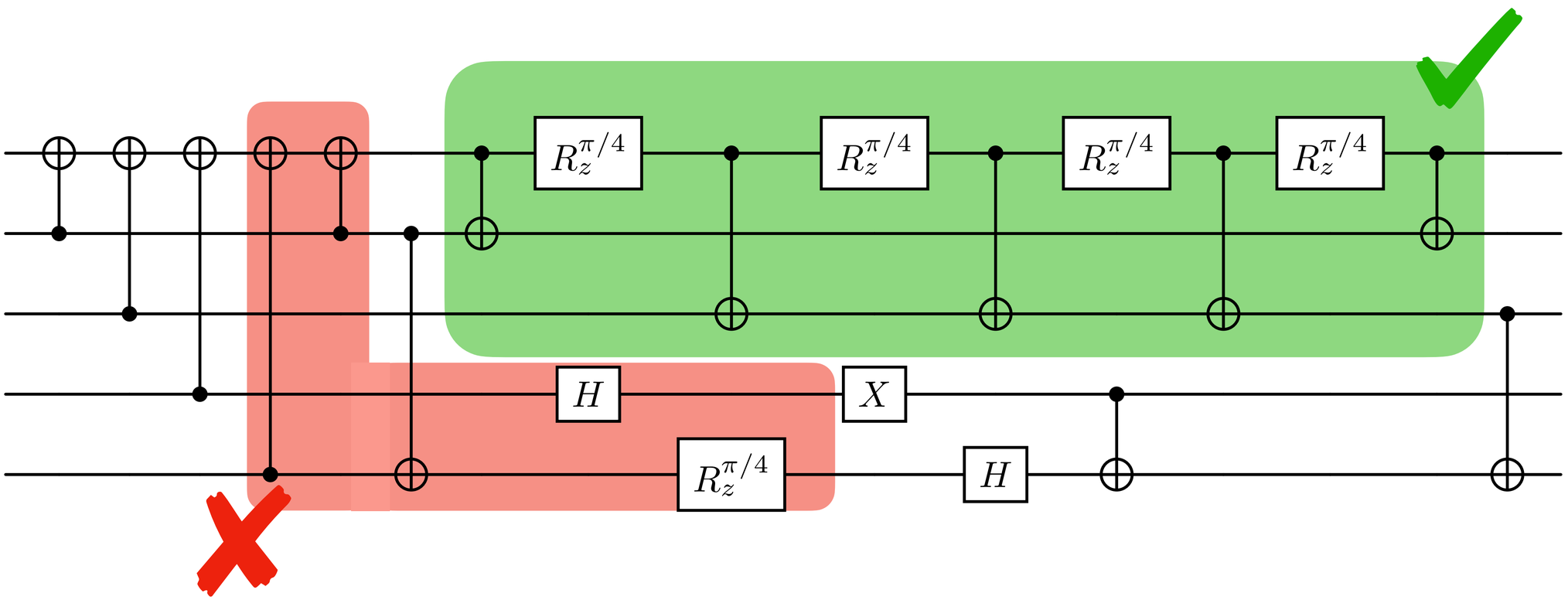


Subcircuit



"convex subgraph"

Subcircuit



"convex subgraph"

i.e. any path between two vertices in the DAG also in subgraph

Rewrite Rules

Rewrite Rules

"peephole optimization"

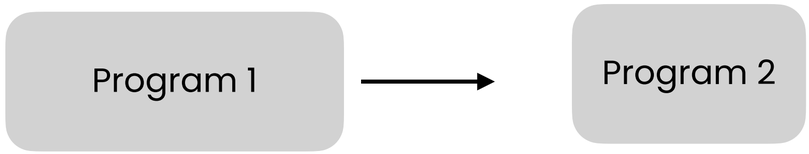
Rewrite Rules

"peephole optimization"

Program 1

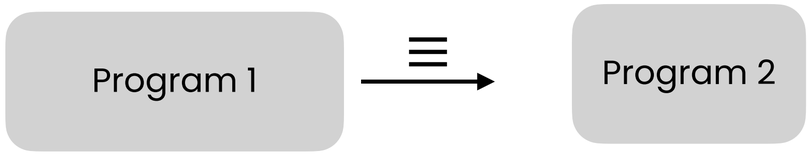
Rewrite Rules

"peephole optimization"



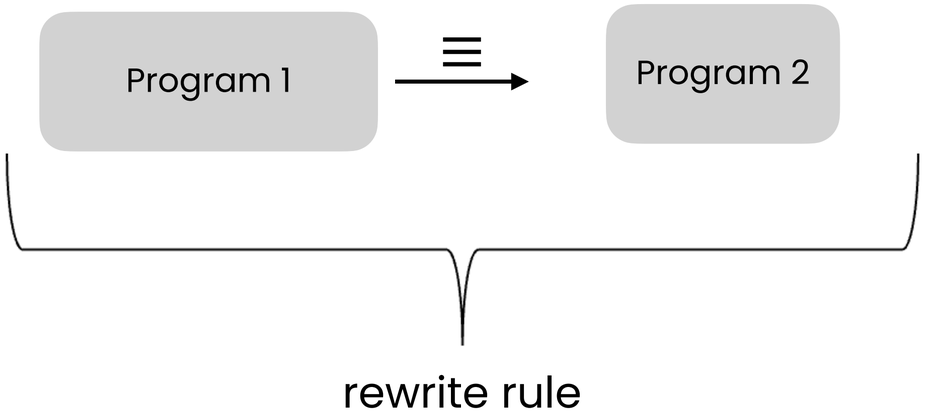
Rewrite Rules

"peephole optimization"

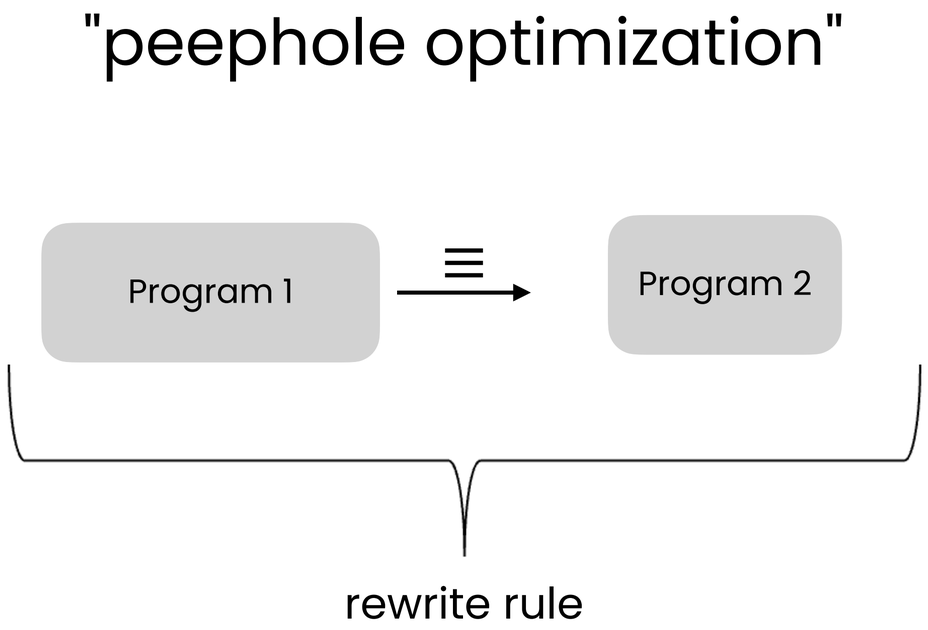


Rewrite Rules

"peephole optimization"

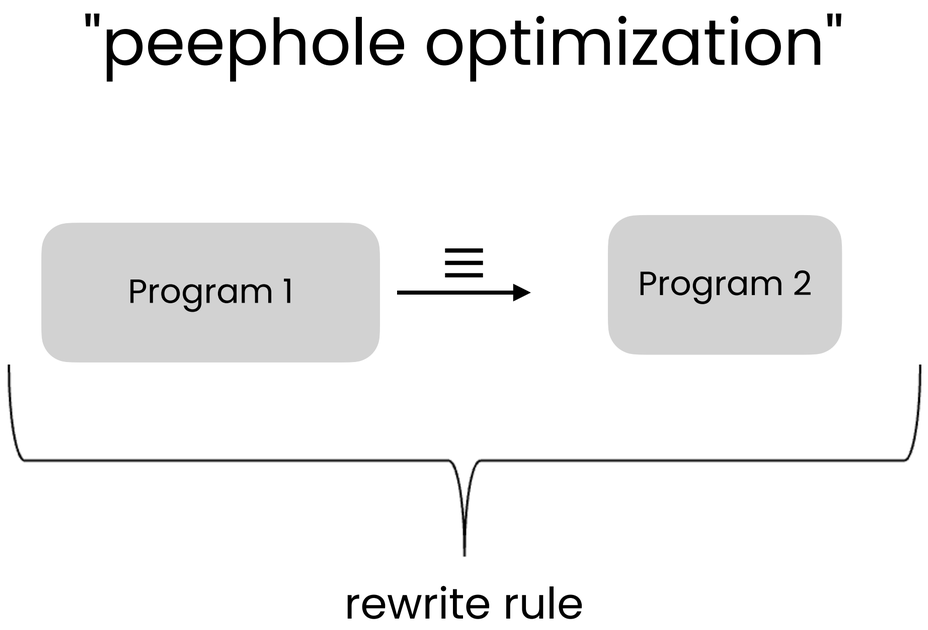


Rewrite Rules



- Optimization: LHS and RHS same gate set

Rewrite Rules



- Optimization: LHS and RHS same gate set
- Decomposition: LHS higher level than RHS

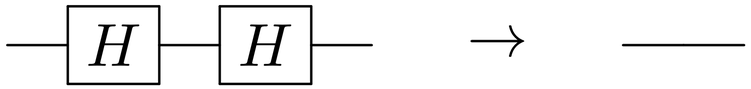
Simple Rewrite Rules

Simple Rewrite Rules

Cancel

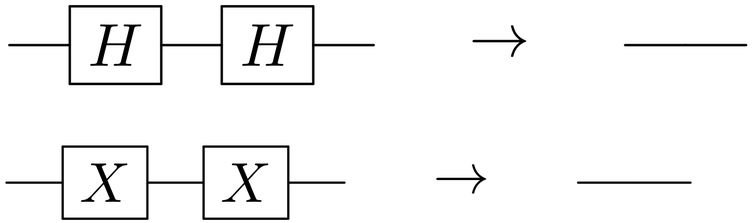
Simple Rewrite Rules

Cancel



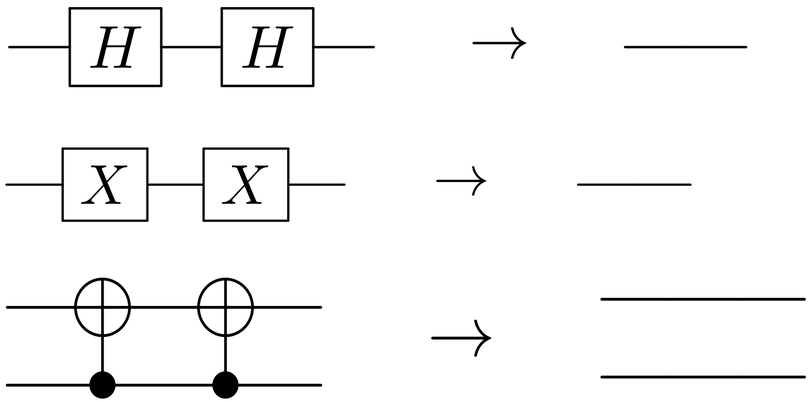
Simple Rewrite Rules

Cancel



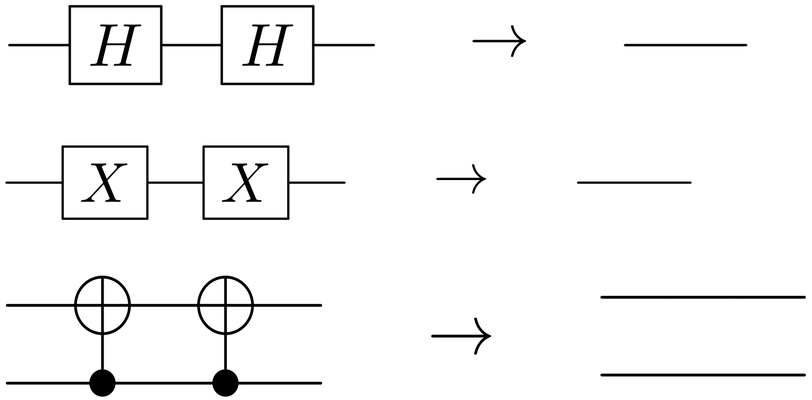
Simple Rewrite Rules

Cancel



Simple Rewrite Rules

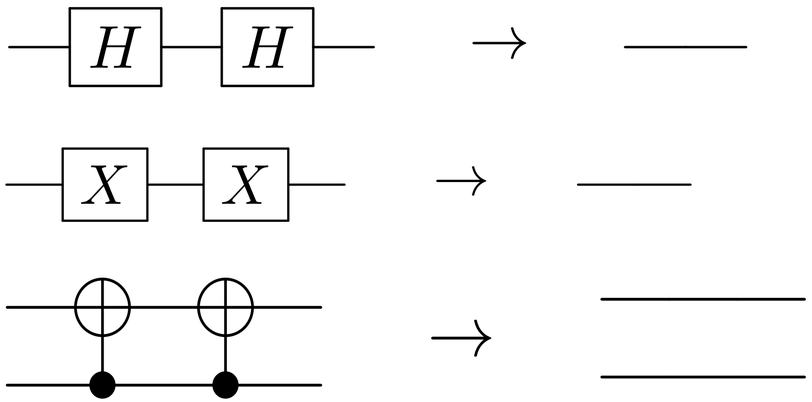
Cancel



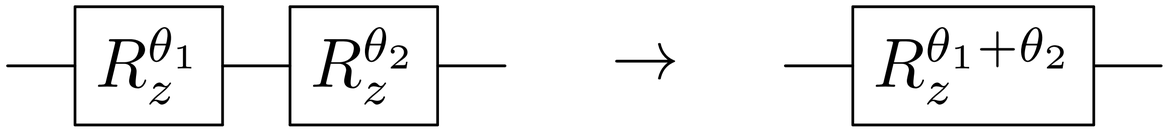
Merge

Simple Rewrite Rules

Cancel

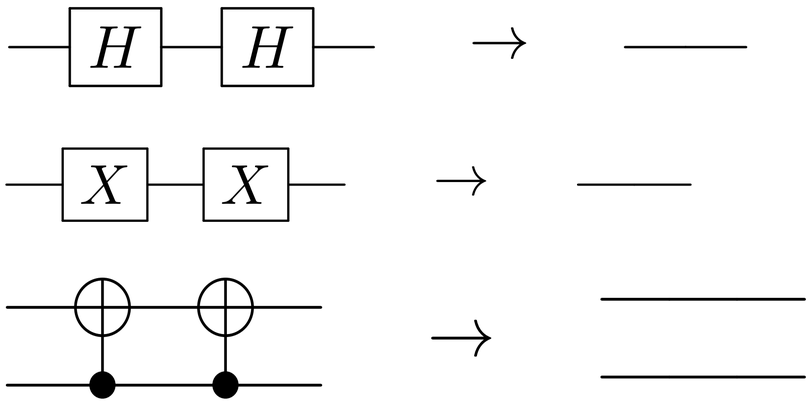


Merge

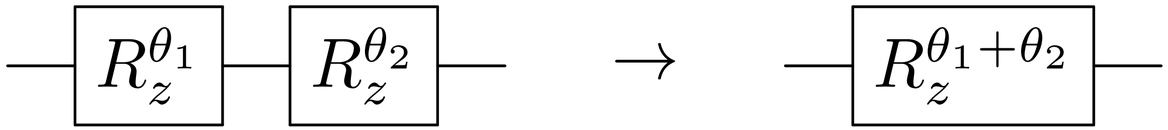


Simple Rewrite Rules

Cancel



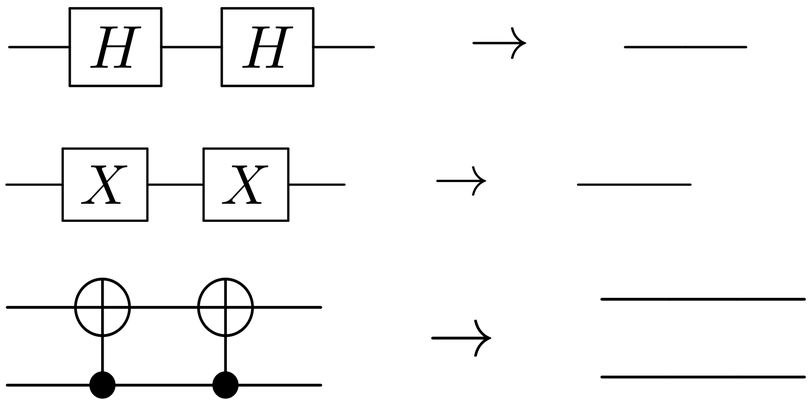
Merge



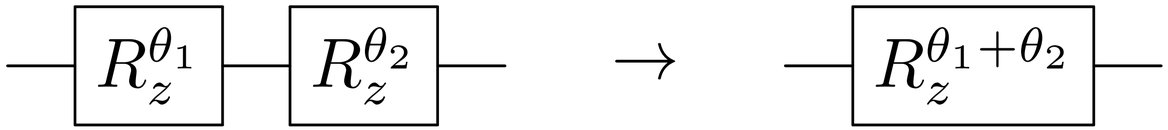
Commute

Simple Rewrite Rules

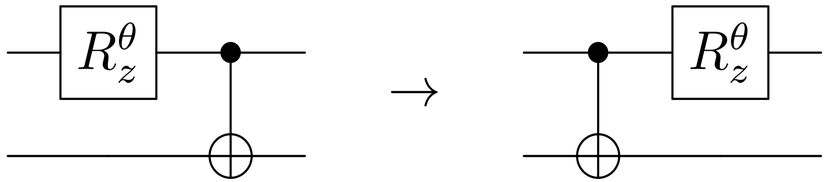
Cancel



Merge

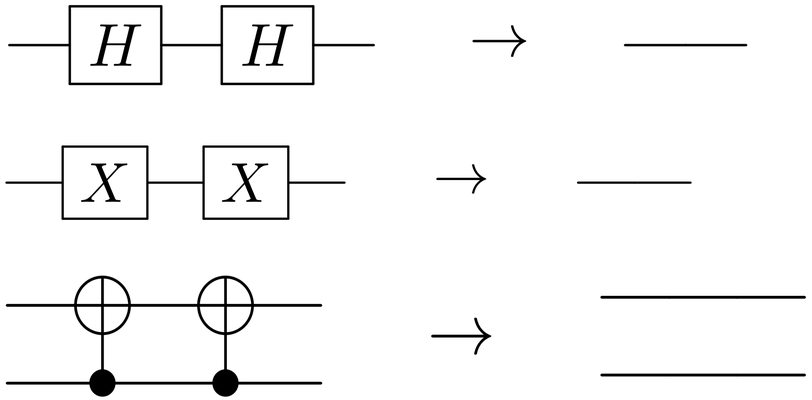


Commute

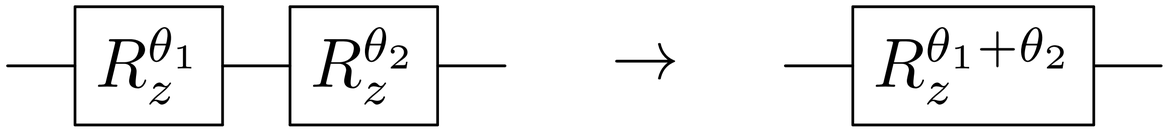


Simple Rewrite Rules

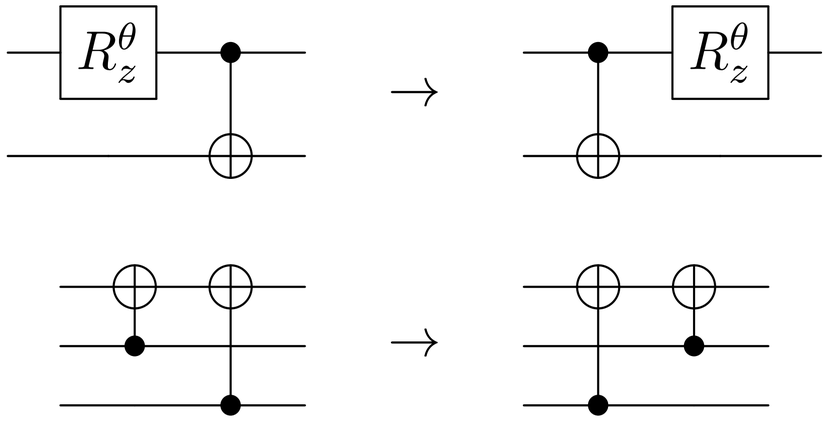
Cancel



Merge

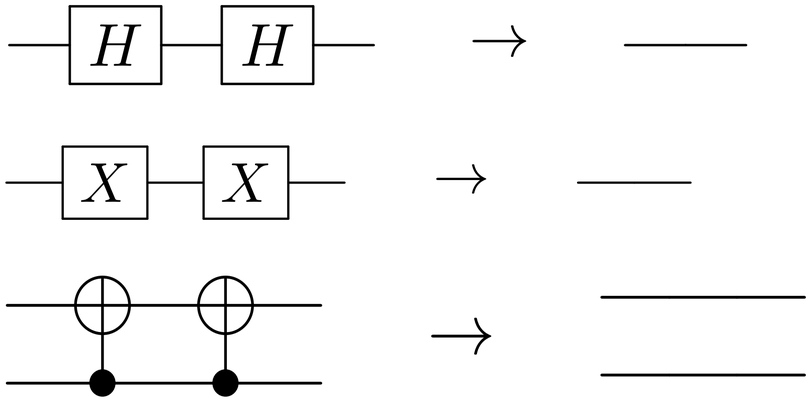


Commute

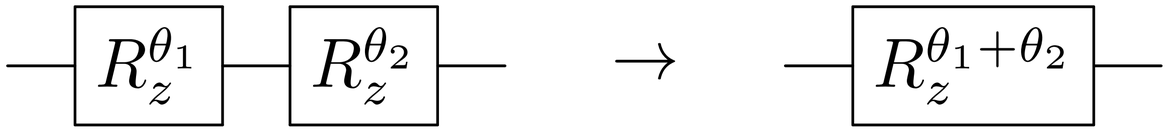


Simple Rewrite Rules

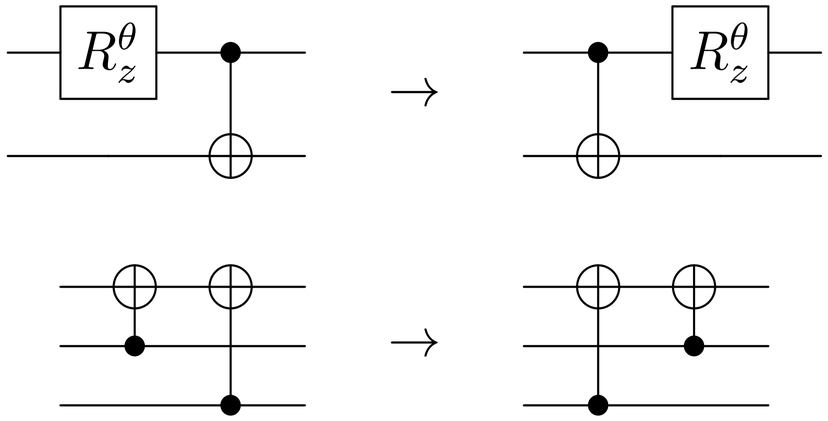
Cancel



Merge



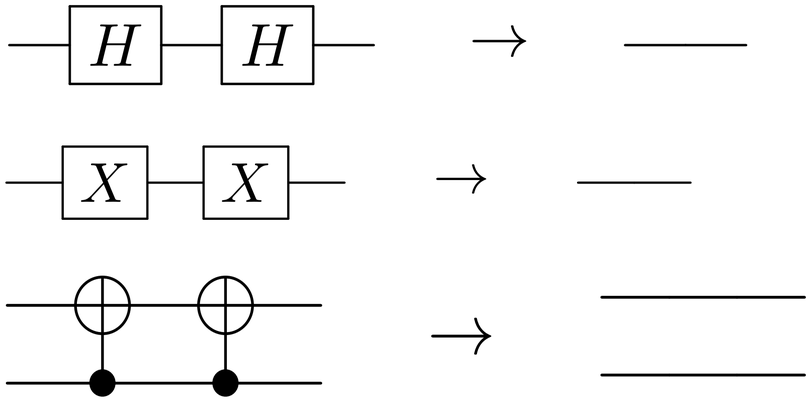
Commute



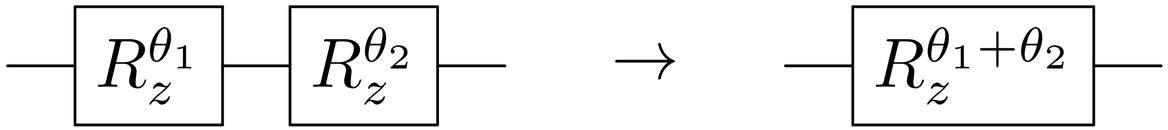
Global Phase

Simple Rewrite Rules

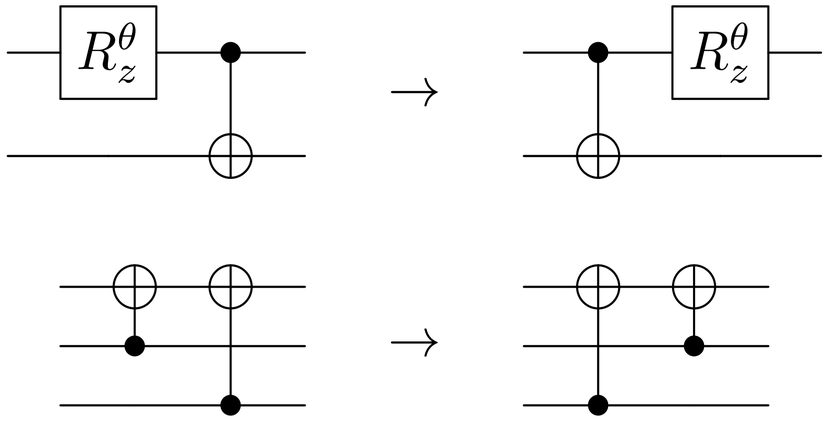
Cancel



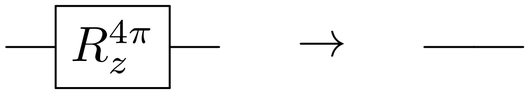
Merge



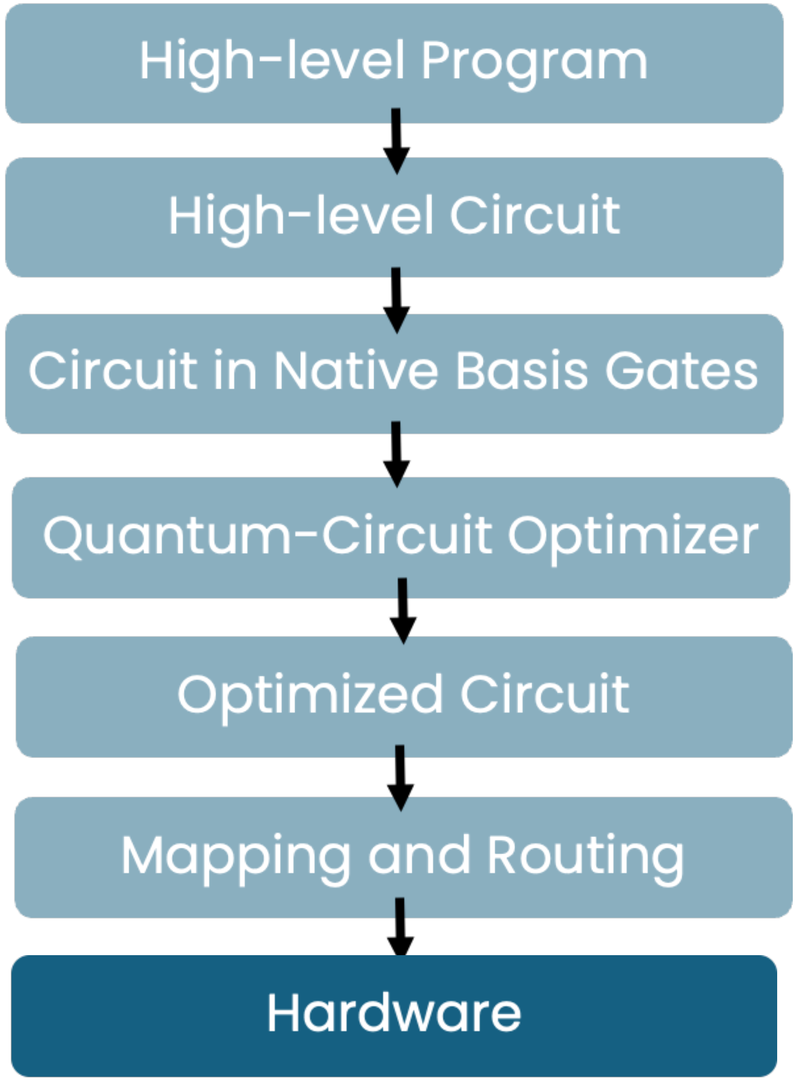
Commute



Global Phase



Diverse Hardware



Diverse Hardware

Hardware

Superconducting

Google
Quantum AI

Diverse Hardware

Hardware

Superconducting

Google Quantum AI

Neutral Atom

Diverse Hardware

Ion Trap

Hardware

Superconducting

Neutral Atom

Diverse Hardware

Ion Trap

IONQ

Neutral Atom

rigetti

OQC

QuEra
COMPUTING INC.

Google Quantum AI

IBM

aws

Diverse Hardware

Ion Trap

Topological

Hardware

Superconducting

Neutral Atom

Diverse Hardware

Ion Trap

Topological

Hardware

Superconducting

Google Quantum AI

U1, U2, U3, CX

Neutral Atom

Diverse Hardware

Ion Trap

Topological

Hardware

Superconducting

Google Quantum AI

Neutral Atom

~~U1, U2, U3, CX~~ → Rz, SX, X, CX

Diverse Hardware

Ion Trap

Topological

Hardware

Superconducting

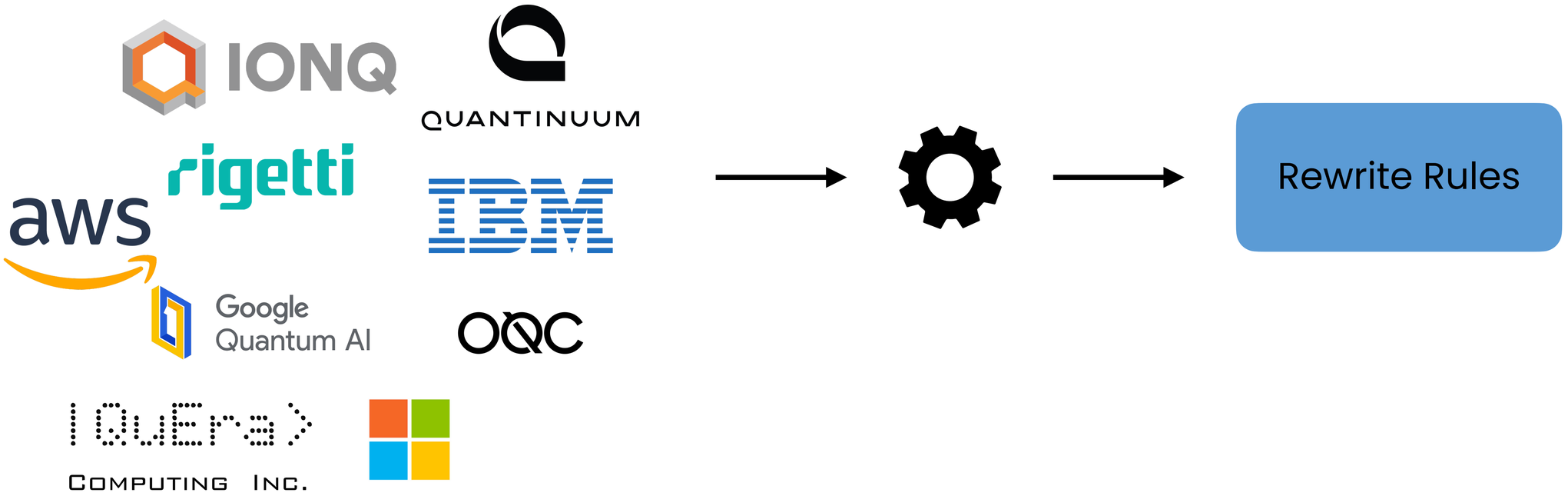
Neutral Atom

~~U1, U2, U3, CX~~ → ~~Rz, SX, X, CX~~

→ Rz, Rx, SX, X, CZ, Rzz

Synthesis!

Arbitrary gate set



Naïve Synthesis

```
rules = []
```

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
```

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
```

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      rules.append(c1 → c2)
```

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      rules.append(c1 → c2)
```

big

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      rules.append(c1 → c2)
```

big

expensive

Naïve Synthesis

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      rules.append(c1 → c2)
```

big

uh oh...

expensive

QUESO

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

(Symbolic)
Circuit

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

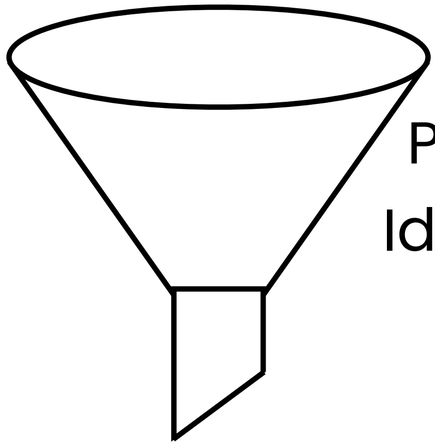
(Symbolic) Circuit → Polynomial

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

(Symbolic) Circuit → Polynomial →  Polynomial Identity Filter (PIF) → Circuit Equivalence Classes

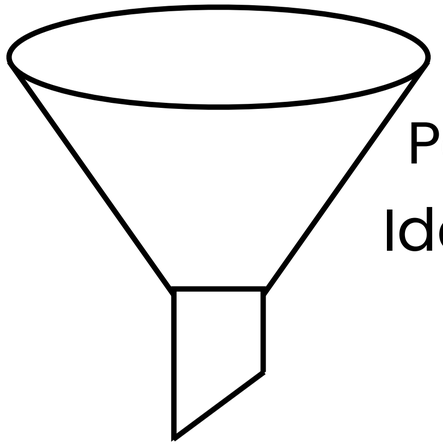
Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

(Symbolic) Circuit



Polynomial Identity Filter (PIF)

Polynomial

Circuit Equivalence Classes

Synthesizing Quantum-Circuit Optimizers

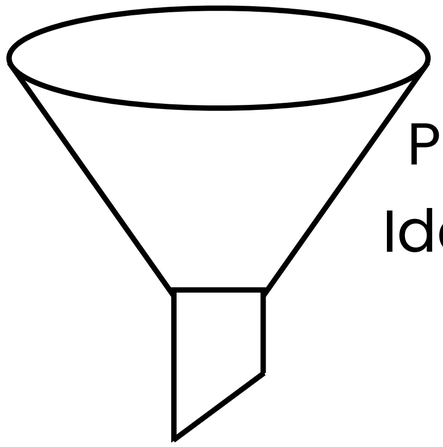
AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

Feynman's path integral formulation for quantum mechanics

(Symbolic) Circuit



Polynomial Identity Filter (PIF)

Polynomial

Circuit Equivalence Classes

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

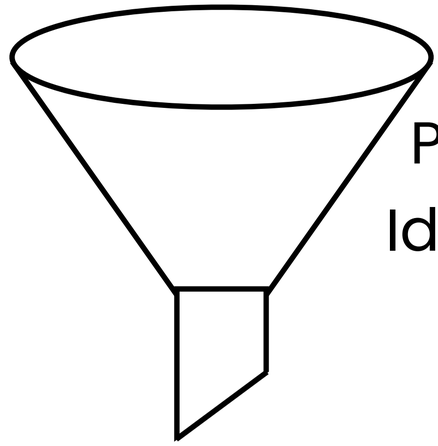
@ PLDI'23

QUESO

Schwartz-Zippel Lemma for polynomial identity testing (PIT)

Feynman's path integral formulation for quantum mechanics

(Symbolic) Circuit



Polynomial Identity Filter (PIF)

Polynomial

Circuit Equivalence Classes

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

QUESO

Feynman's path integral formulation

for quantum mechanics

(Symbolic) Circuit

Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \to e^{i(2x-1)\theta}|x\rangle$$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

$$CNOT : |x_1 x_2\rangle \rightarrow |x_1(x_1 \oplus x_2)\rangle$$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

$$CNOT : |x_1 x_2\rangle \rightarrow |x_1(x_1 \oplus x_2)\rangle$$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

$$CNOT : |x_1 x_2\rangle \rightarrow |x_1(x_1 \oplus x_2)\rangle$$

More generally: $|\mathbf{x}\rangle \rightarrow \phi(\mathbf{x}, \rho)\,|f(\mathbf{x})\rangle$

Path-sum Semantics

Algebraically represent semantics of quantum gates

$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta}|x\rangle$$

$$CNOT : |x_1 x_2\rangle \rightarrow |x_1(x_1 \oplus x_2)\rangle$$

amplitude state

More generally: $|\mathbf{x}\rangle \rightarrow \phi(\mathbf{x}, \rho)\,|f(\mathbf{x})\rangle$

Simple Circuit as Polynomial

Circuit	Polynomial
`Rz(`θ`) q1;`	$e^{i(2(0)-1)\theta}\|0\rangle$

Simple Circuit as Polynomial

Circuit	Polynomial
`Rz(`θ`) q1;`	$e^{-i\theta}\|0\rangle$

Simple Circuit as Polynomial

Circuit	Polynomial
`Rz(`θ`)  q1;`	$e^{-i\theta}\|0\rangle + e^{i\theta}\|1\rangle$

Simple Circuit as Polynomial

Circuit

`Rz(`θ`)  q1;`

Polynomial

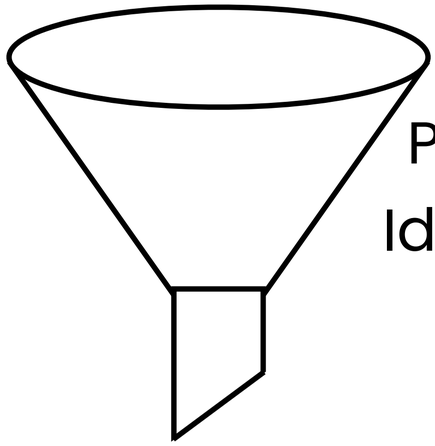
$e^{-i\theta}|0\rangle + e^{i\theta}|1\rangle$

QUESO

Schwartz-Zippel Lemma for polynomial identity testing (PIT)

Feynman's path integral formulation for quantum mechanics

(Symbolic) Circuit → Polynomial →



Polynomial Identity Filter (PIF)

Synthesizing Quantum-Circuit Optimizers

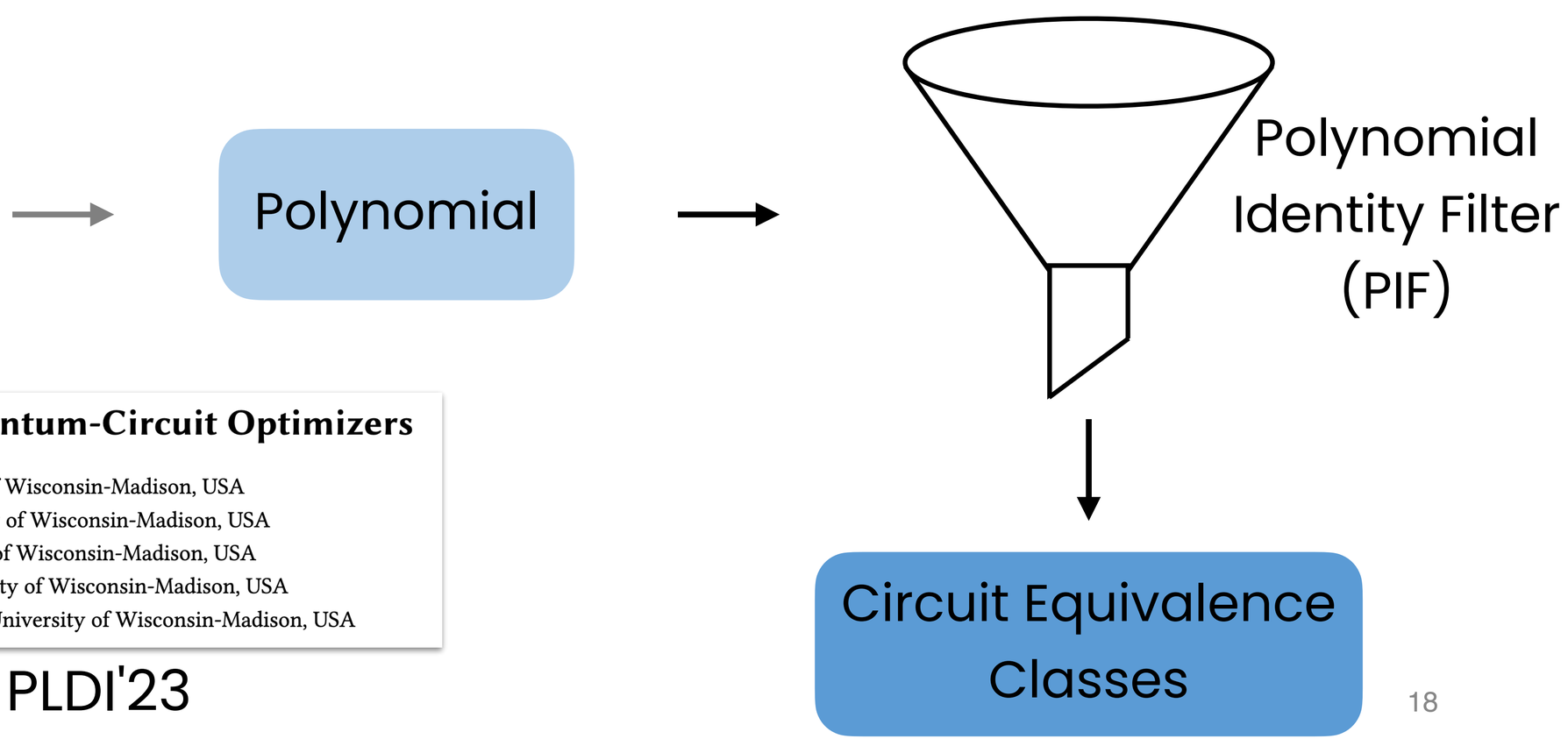
AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

Circuit Equivalence Classes

QUESO

Schwartz-Zippel Lemma for polynomial identity testing (PIT)



Synthesizing Quantum-Circuit Optimizers

AMANDA XU, University of Wisconsin-Madison, USA
ABTIN MOLAVI, University of Wisconsin-Madison, USA
LAUREN PICK, University of Wisconsin-Madison, USA
SWAMIT TANNU, University of Wisconsin-Madison, USA
AWS ALBARGHOUTHI, University of Wisconsin-Madison, USA

@ PLDI'23

Polynomial Identity Testing (PIT)

$$\mathbf{x} \in \mathbb{C}^3 \qquad p_1(\mathbf{x}) = x_1 x_2$$

Polynomial Identity Testing (PIT)

$$\mathbf{x} \in \mathbb{C}^3 \qquad p_1(\mathbf{x}) = x_1 x_2$$

$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$

Polynomial Identity Testing (PIT)

$$\mathbf{x} \in \mathbb{C}^3 \qquad p_1(\mathbf{x}) = x_1 x_2$$
$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$

How can we check $p_1 \equiv p_2$?

Schwartz-Zippel Lemma

$$p_1(\mathbf{x}) = x_1 x_2$$

$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$

Schwartz-Zippel Lemma

$$p_1(\mathbf{x}) = x_1 x_2$$

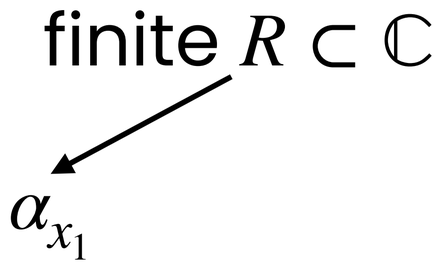
$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$

finite $R \subset \mathbb{C}$

Schwartz-Zippel Lemma

$$p_1(\mathbf{x}) = x_1 x_2$$

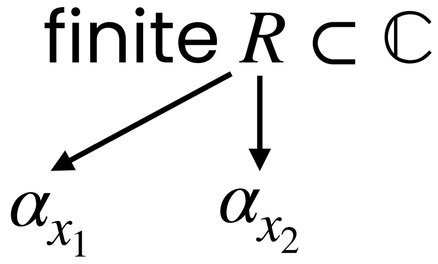
$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$



Schwartz-Zippel Lemma

$$p_1(\mathbf{x}) = x_1 x_2$$

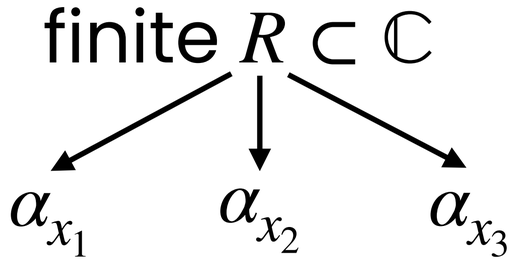
$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$



Schwartz-Zippel Lemma

$$p_1(\mathbf{x}) = x_1 x_2$$

$$p_2(\mathbf{x}) = x_1 x_2 + x_3 x_2$$



Schwartz-Zippel Lemma

$$p_1(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2}$$
$$p_2(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2} + \alpha_{x_3}\alpha_{x_2}$$

Schwartz-Zippel Lemma

$$p_1(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2}$$
$$p_2(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2} + \alpha_{x_3}\alpha_{x_2}$$

Check $p_1(\boldsymbol{\alpha}) = p_2(\boldsymbol{\alpha})$

Schwartz-Zippel Lemma

$$p_1(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2}$$
$$p_2(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2} + \alpha_{x_3}\alpha_{x_2}$$

Check $p_1(\boldsymbol{\alpha}) = p_2(\boldsymbol{\alpha})$

If $p_1 \neq p_2$, then the probability that $p_1(\boldsymbol{\alpha}) = p_2(\boldsymbol{\alpha})$ is at most $\frac{d}{|R|}$ (d (max total degree))

Schwartz-Zippel Lemma

$$p_1(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2}$$
$$p_2(\boldsymbol{\alpha}) = \alpha_{x_1}\alpha_{x_2} + \alpha_{x_3}\alpha_{x_2}$$

Check $p_1(\boldsymbol{\alpha}) = p_2(\boldsymbol{\alpha})$

If $p_1 \neq p_2$, then the probability that $p_1(\boldsymbol{\alpha}) = p_2(\boldsymbol{\alpha})$ is at most $\frac{d}{|R|}$

small

d (max total degree)

very large

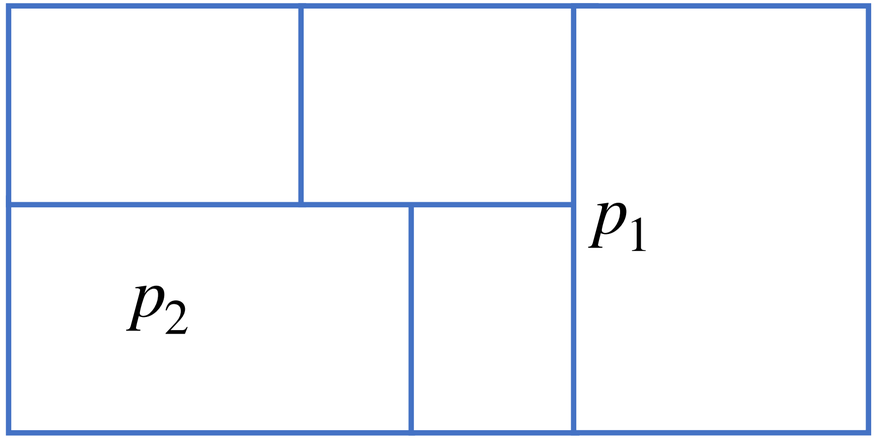
Polynomial Identity Filter (PIF)

Insert p_3

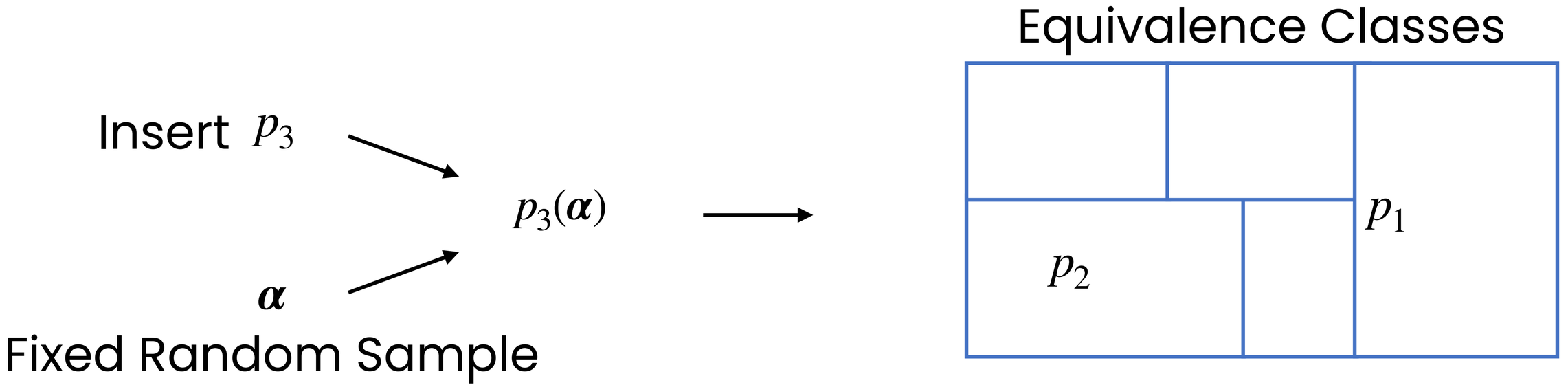
$\boldsymbol{\alpha}$

Fixed Random Sample

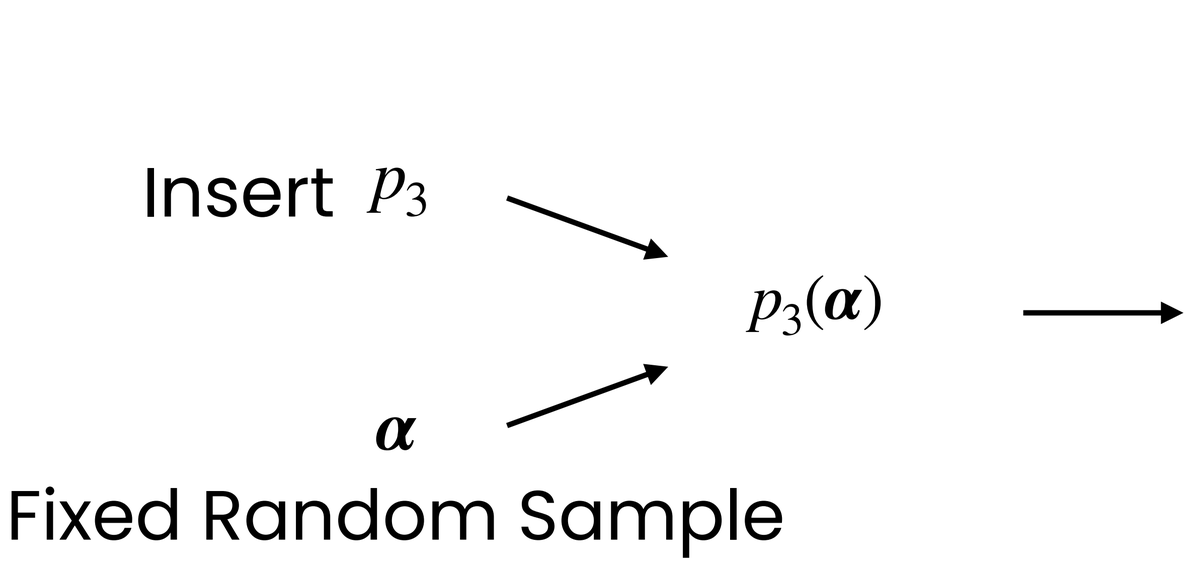
Equivalence Classes

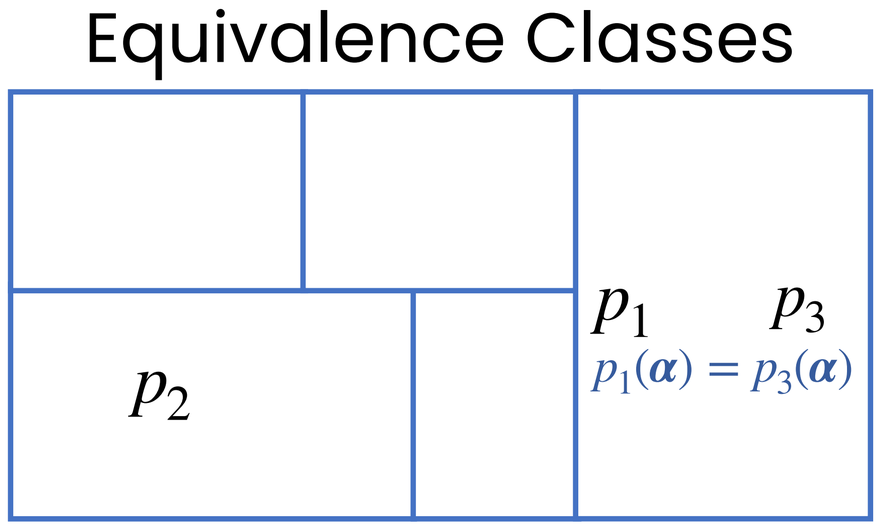


Polynomial Identity Filter (PIF)



Polynomial Identity Filter (PIF)





QUESO's Improvements

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      rules.append(c1 → c2)
```

big

uh oh...

expensive

QUESO's Improvements

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      insert into PIF
      rules.append(c1 → c2)
```

big

uh oh...

expensive

QUESO's Improvements

```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
  for c2 in circuits:
    if verify(c1, c2):
      insert into PIF
      rules.append(c1 → c2)
```

big

uh oh...

expensive

Efficient equivalence check between circuit pairs!

QUESO's Improvements

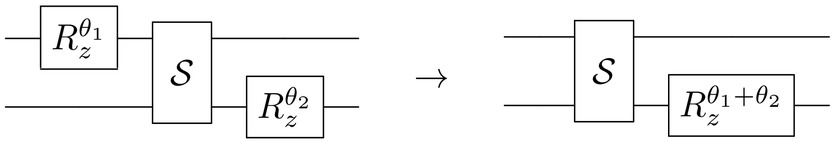
```
rules = []
circuits = enumerate(max_qubits, max_size)
for c1 in circuits:
    for c2 in circuits:
        if verify(c1, c2):
            insert into PIF
            rules.append(c1 → c2)
```

big

uh oh...

expensive

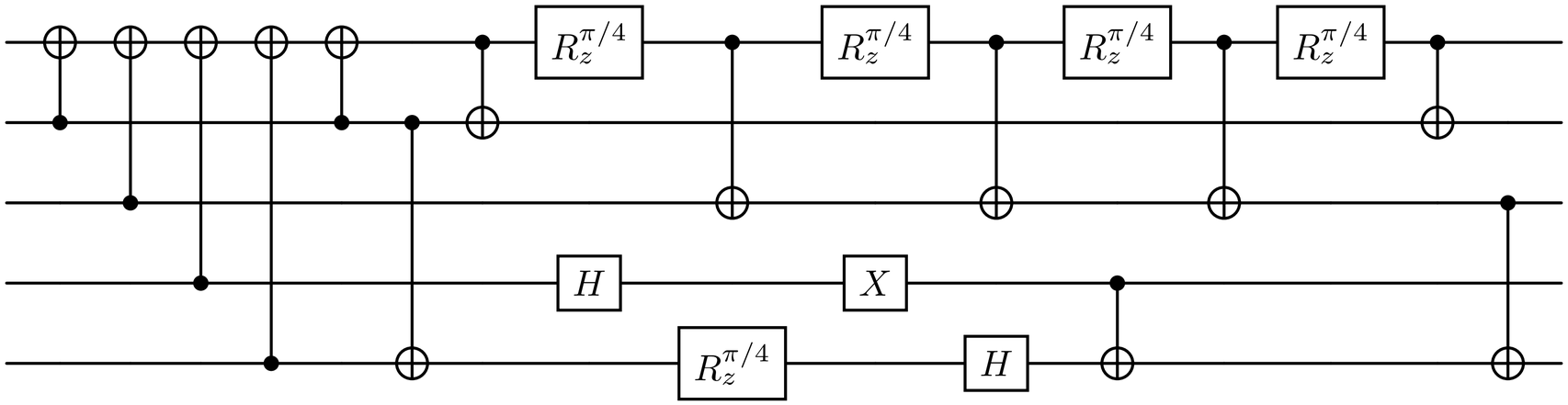
Efficient equivalence check between circuit pairs!

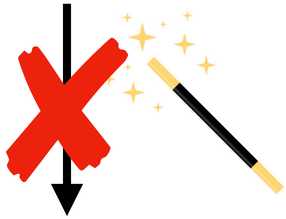
bonus: 

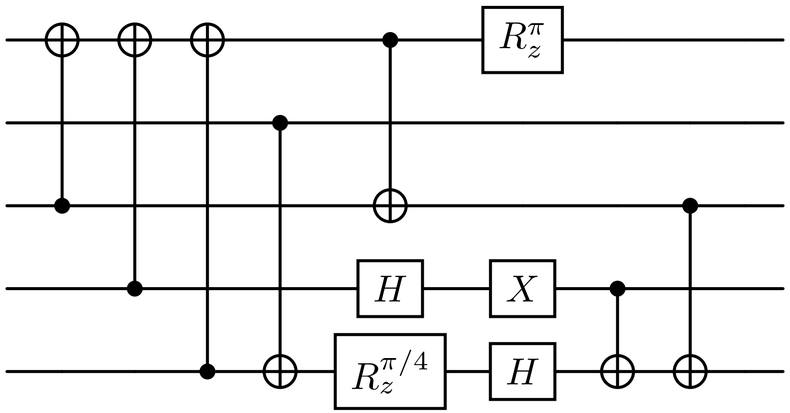
(advanced symbolic rules)

Circuit Resynthesis

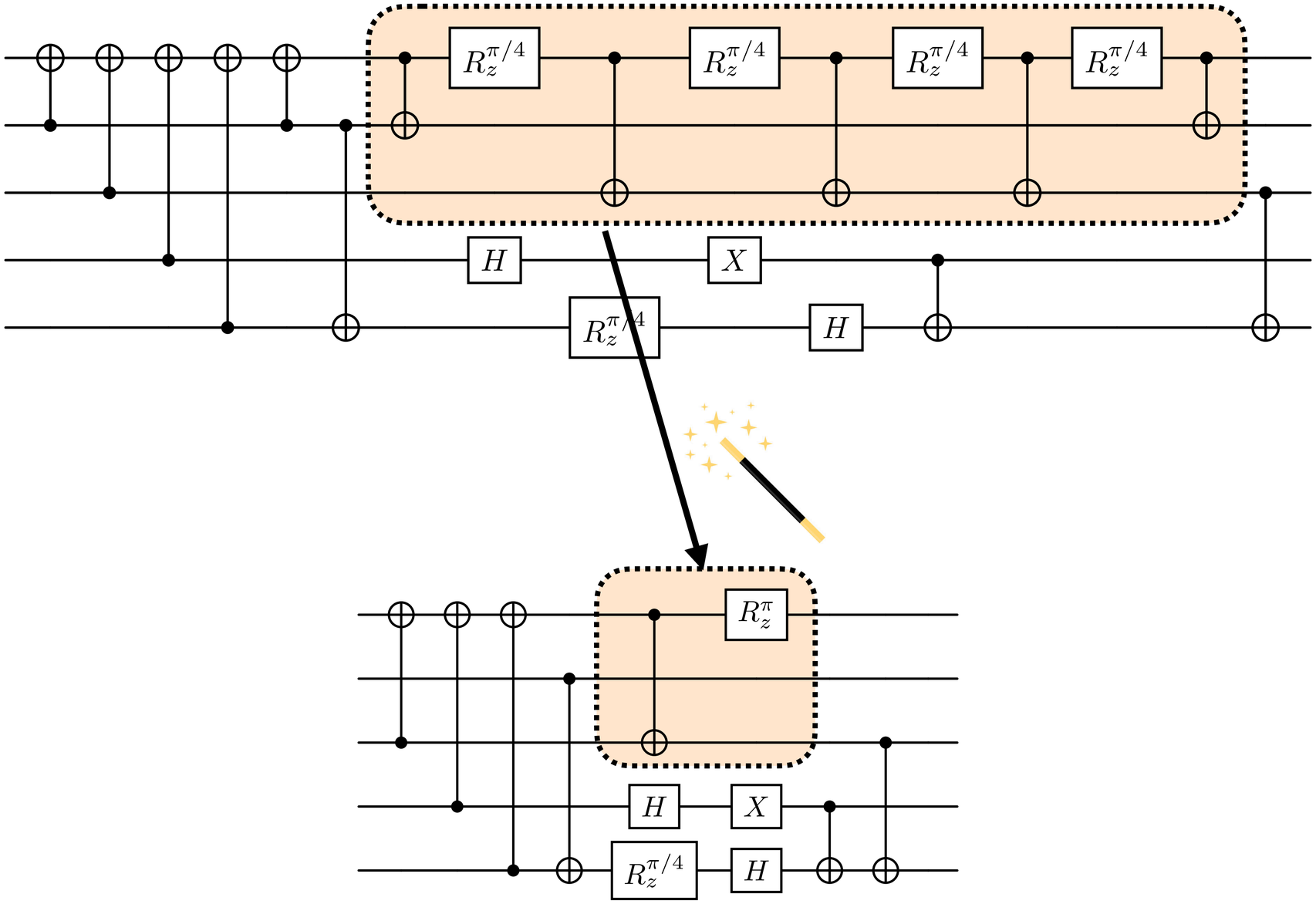
Circuit Resynthesis



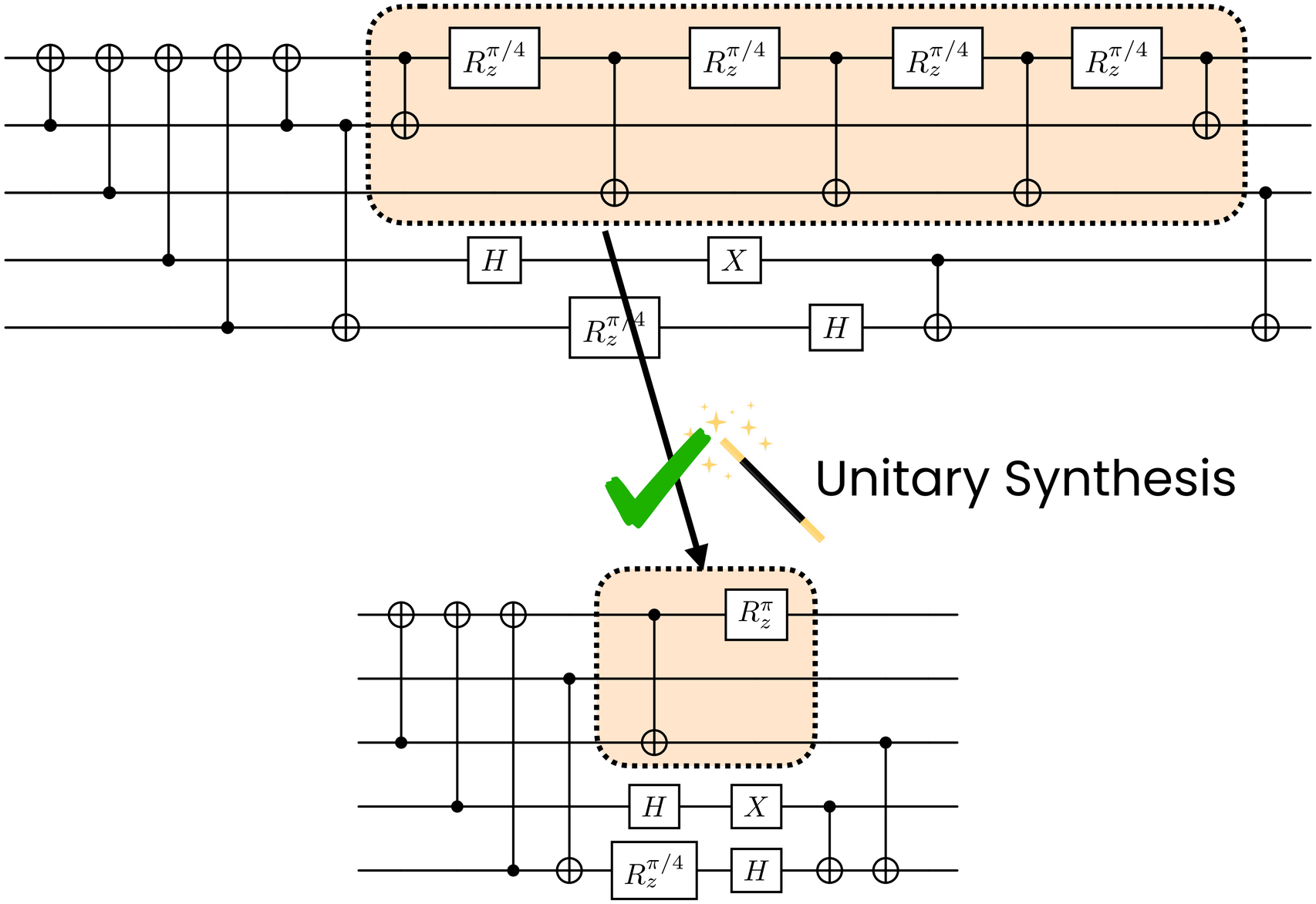




Circuit Resynthesis

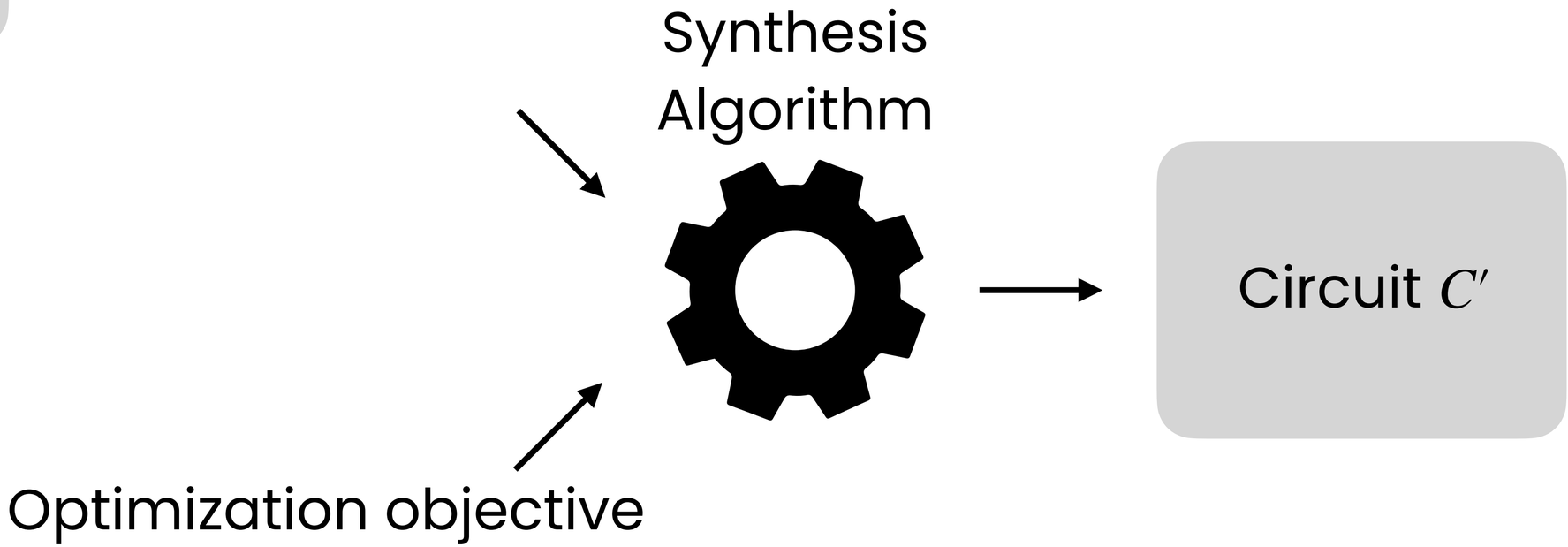


Circuit Resynthesis

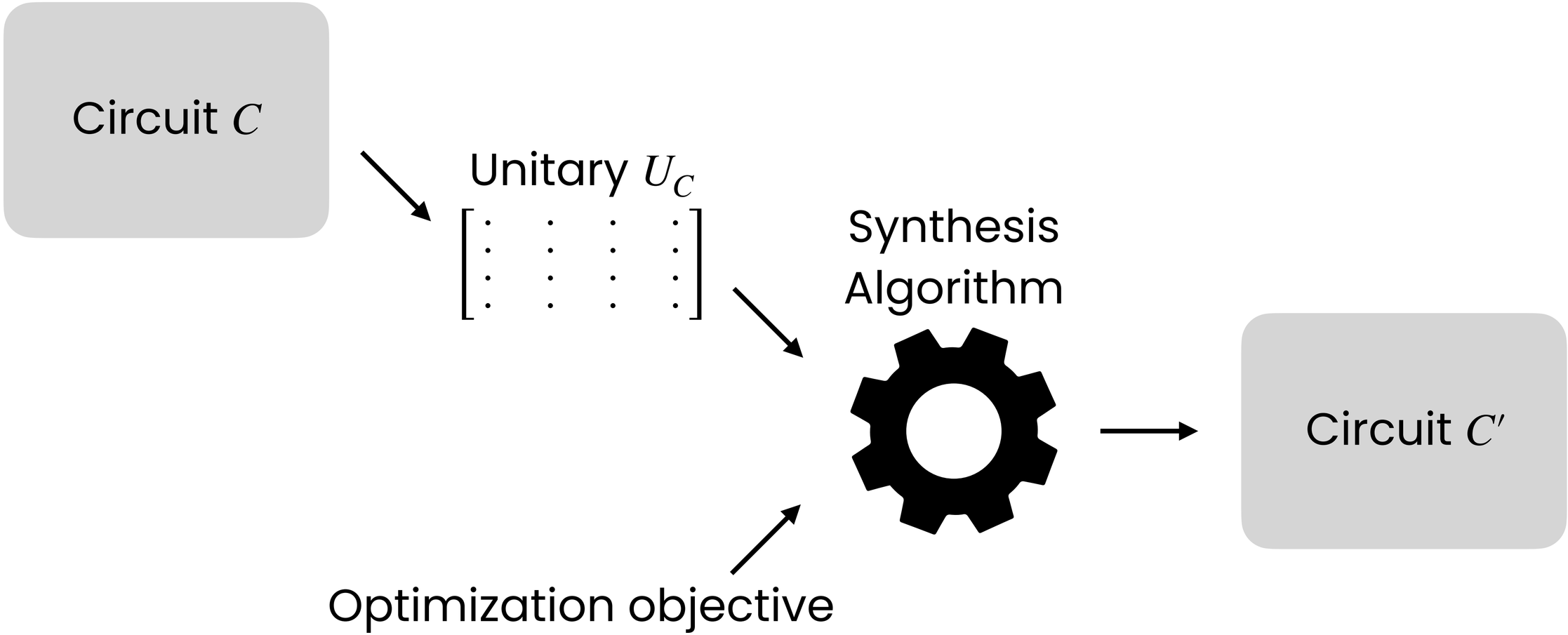


Unitary Synthesis

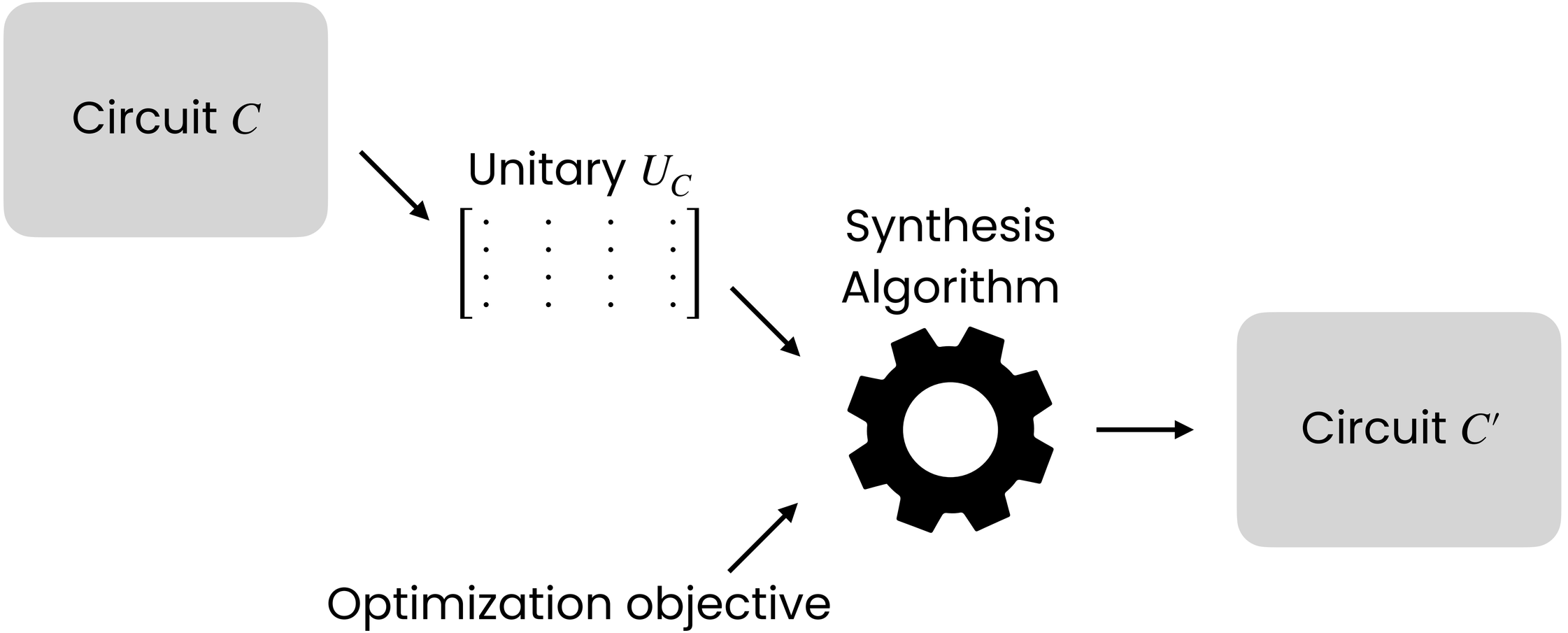
Circuit C



Unitary Synthesis



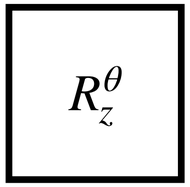
Unitary Synthesis



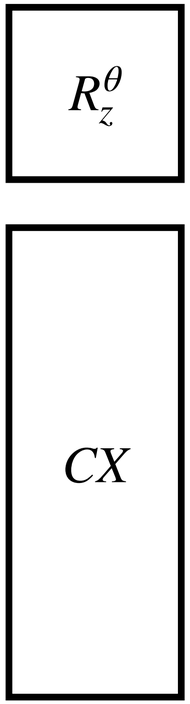
Guarantee $C \equiv C'$

Circuit to Unitary

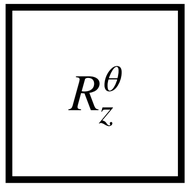
Circuit to Unitary



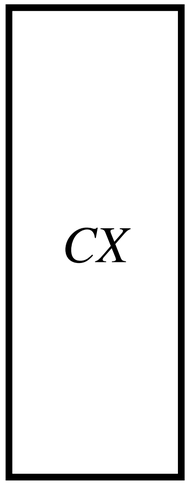
Circuit to Unitary



Circuit to Unitary



$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$



Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

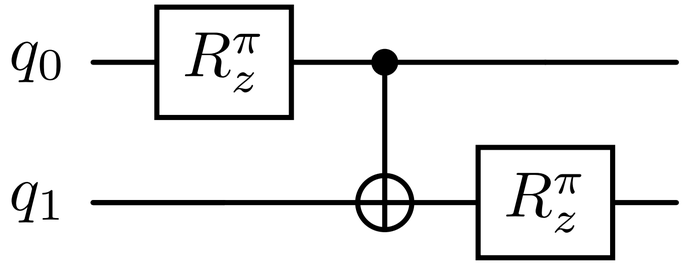
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$C :=$



Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

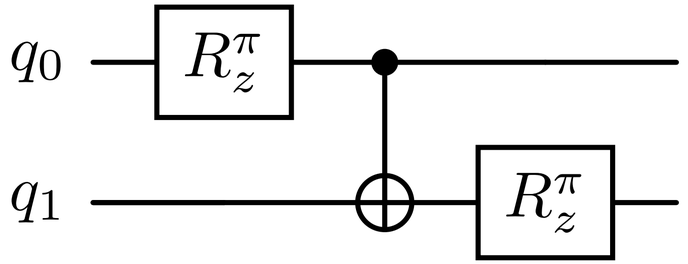
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$C :=$



$U_C =$ $U_{R_z^\pi}$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

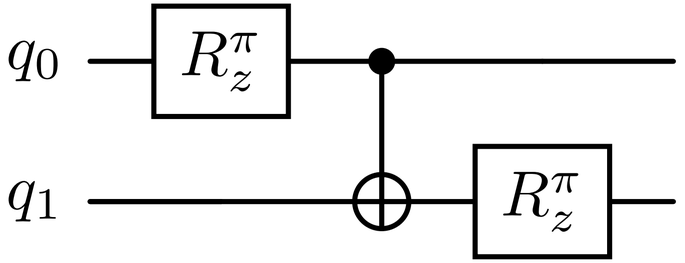
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C := \quad q_0 \quad q_1$$



$$U_C = \quad U_{R_z^\pi}$$

$2^2 \times 2^2$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

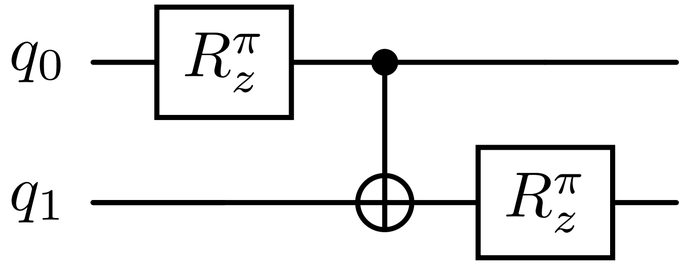
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$C :=$



$U_C =$
$2^2 \times 2^2$

$U_{R_z^\pi}$
2×2

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

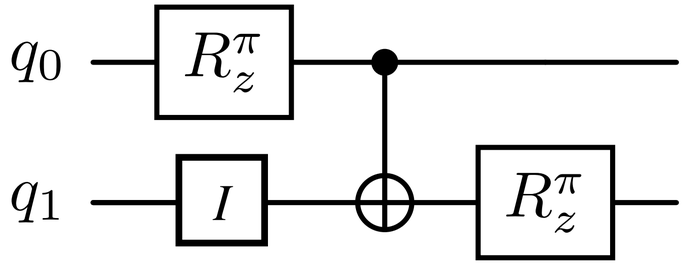
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$C :=$



$U_C =$
$2^2 \times 2^2$

$U_{R_z^\pi}$
2×2

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C := \quad q_0 \text{ — } R_z^\pi \text{ — } \bullet \text{ — }$$
$$\qquad\quad q_1 \text{ — } I \text{ — } \oplus \text{ — } R_z^\pi \text{ — }$$

$$U_C = (U_{R_z^\pi} \otimes U_I)$$
$$2^2 \times 2^2 \qquad 2 \times 2$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

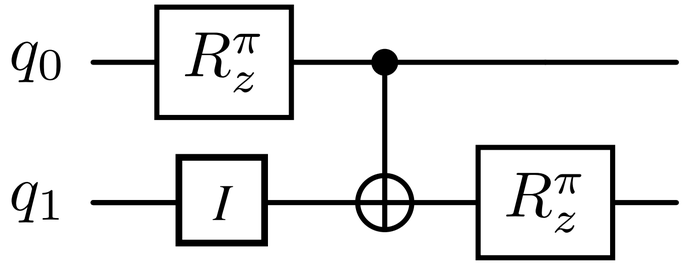
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C :=$$



$$U_C = \underbrace{(U_{R_z^\pi} \otimes U_I)}_{2^2 \times 2^2}$$

$$\underbrace{2 \times 2 \quad 2 \times 2}_{2^2 \times 2^2}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

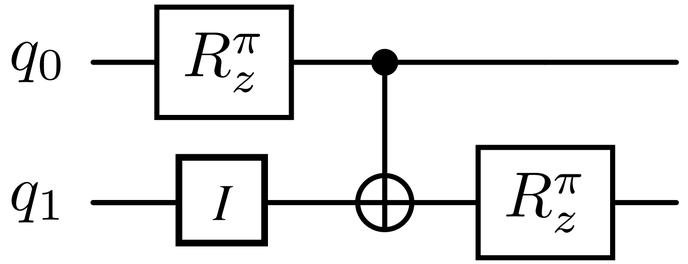
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C :=$$



$$U_C = \underbrace{(U_{R_z^\pi} \otimes U_I)}_{2^2 \times 2^2}$$
$$\underbrace{2 \times 2 \quad 2 \times 2}$$

$$(U_{R_z^\pi} \otimes U_I) = \begin{bmatrix} e^{-i\frac{\pi}{2}} \cdot U_I & 0 \cdot U_I \\ 0 \cdot U_I & e^{i\frac{\pi}{2}} \cdot U_I \end{bmatrix}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

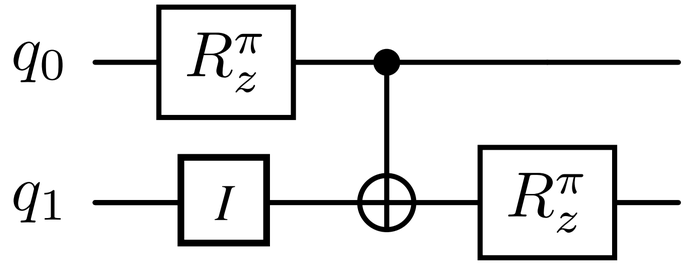
CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C := \begin{array}{l} q_0 \\ q_1 \end{array} \quad \Qquad$$



$$U_C = \underset{2^2 \times 2^2}{(U_{R_z^\pi} \otimes U_I)}$$

$$\underbrace{\overset{2\times 2}{\quad} \quad \overset{2\times 2}{\quad}}_{2^2 \times 2^2}$$

$$(U_{R_z^\pi} \otimes U_I) = \begin{bmatrix} \overset{2\times 2}{e^{-i\frac{\pi}{2}} \cdot U_I} & \overset{2\times 2}{0 \cdot U_I} \\ \underset{2\times 2}{0 \cdot U_I} & \underset{2\times 2}{e^{i\frac{\pi}{2}} \cdot U_I} \end{bmatrix}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C := \begin{array}{l} q_0 \\ q_1 \end{array} \quad \begin{array}{c} R_z^\pi \\ I \end{array} \quad \bullet \quad \oplus \quad R_z^\pi$$

$$U_C = U_{CX} \cdot (U_{R_z^\pi} \otimes U_I)$$

$$2^2 \times 2^2 \qquad \underbrace{2 \times 2 \quad 2 \times 2}_{2^2 \times 2^2}$$

$$(U_{R_z^\pi} \otimes U_I) = \begin{bmatrix} \overset{2\times 2}{e^{-i\frac{\pi}{2}} \cdot U_I} & \overset{2\times 2}{0 \cdot U_I} \\ \underset{2\times 2}{0 \cdot U_I} & \underset{2\times 2}{e^{i\frac{\pi}{2}} \cdot U_I} \end{bmatrix}$$

Circuit to Unitary

R_z^θ

$$U_{R_z^\theta} := \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}$$

CX

$$U_{CX} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

I

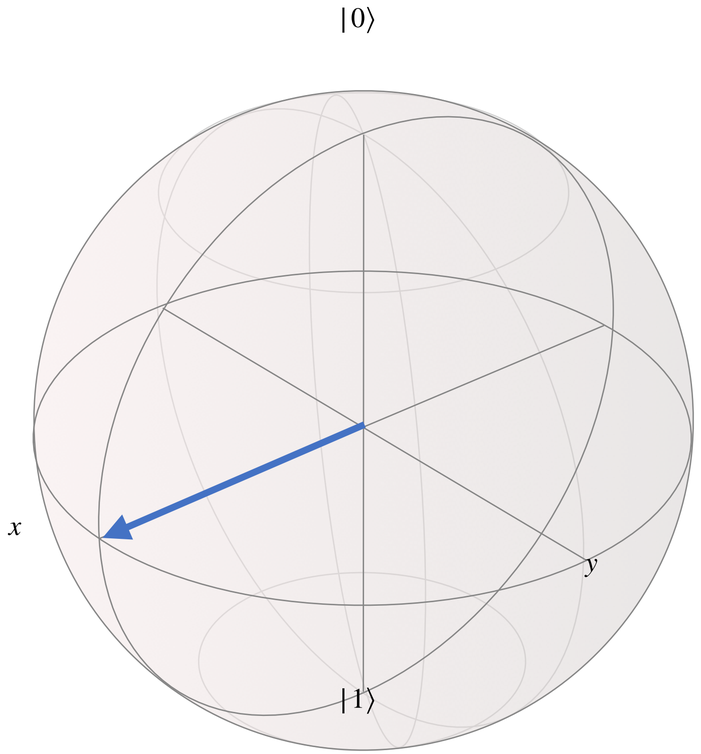
$$U_I := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$C := \begin{array}{l} q_0 \\ q_1 \end{array} \quad \Qcircuit @C=1em @R=1em { & \gate{R_z^\pi} & \ctrl{1} & \gate{I} \\ & \gate{I} & \targ & \gate{R_z^\pi} }$$

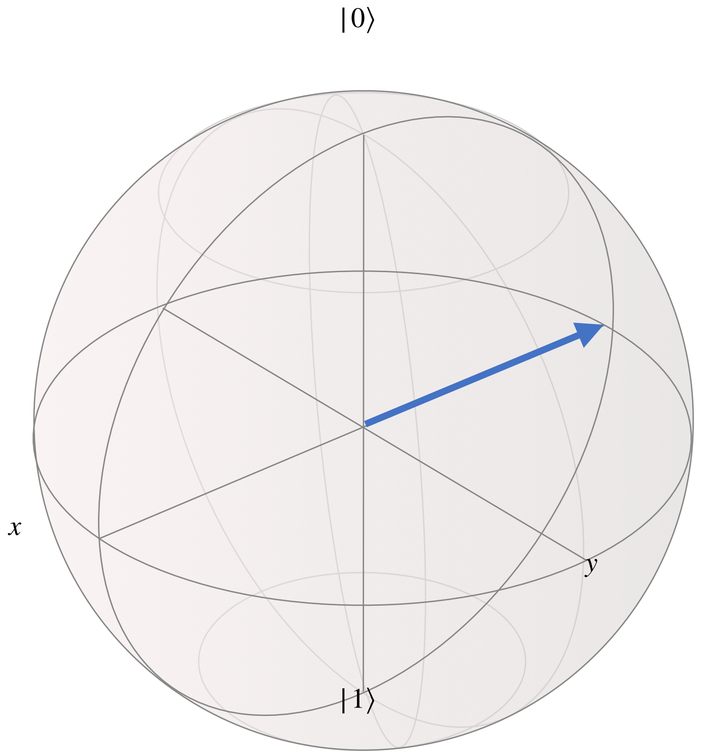
$$\underset{2^2 \times 2^2}{U_C} = \underbrace{(U_I \otimes U_{R_z^\pi}) \cdot U_{CX} \cdot (U_{R_z^\pi} \otimes U_I)}_{2^2 \times 2^2}$$

$$(U_{R_z^\pi} \otimes U_I) = \begin{bmatrix} \overset{2\times 2}{e^{-i\frac{\pi}{2}} \cdot U_I} & \overset{2\times 2}{0 \cdot U_I} \\ \underset{2\times 2}{0 \cdot U_I} & \underset{2\times 2}{e^{i\frac{\pi}{2}} \cdot U_I} \end{bmatrix}$$

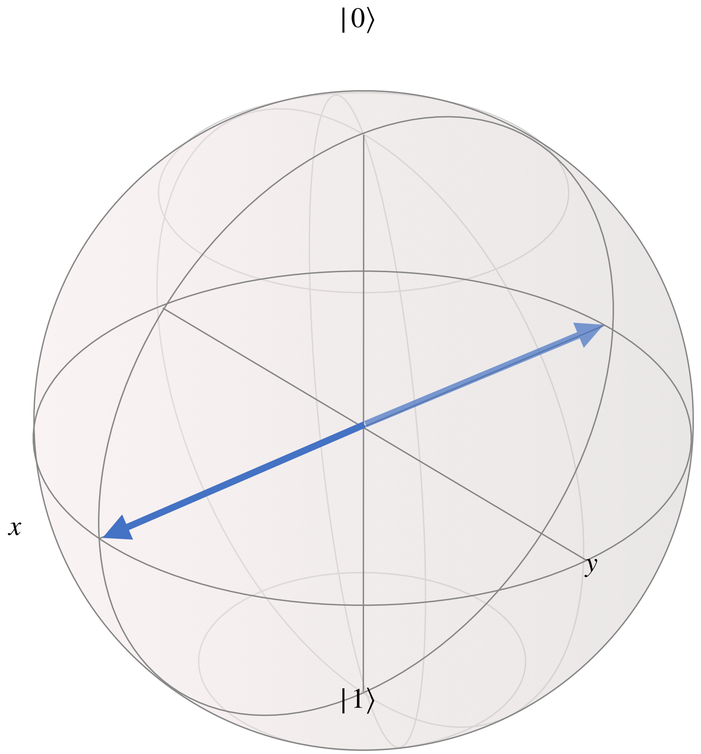
Approximate Circuit Equivalence



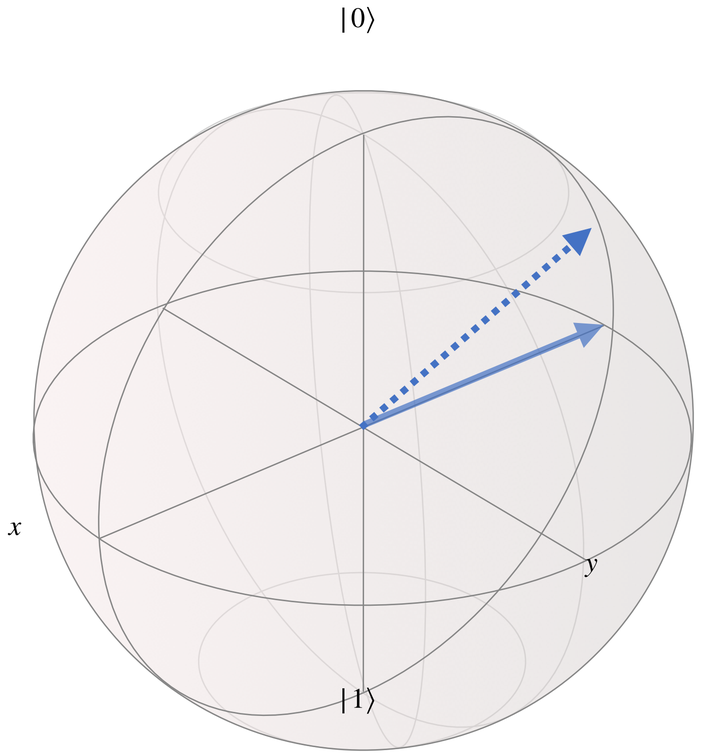
Approximate Circuit Equivalence



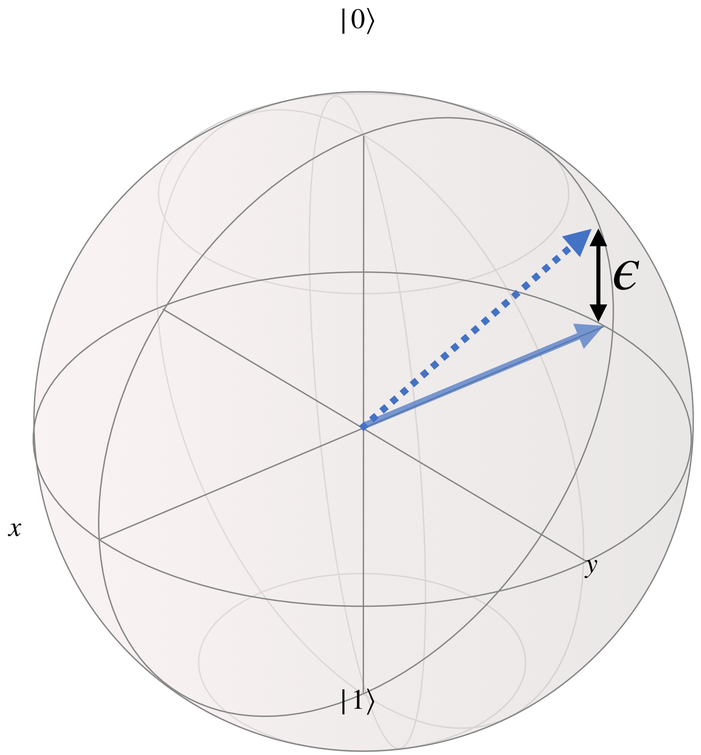
Approximate Circuit Equivalence



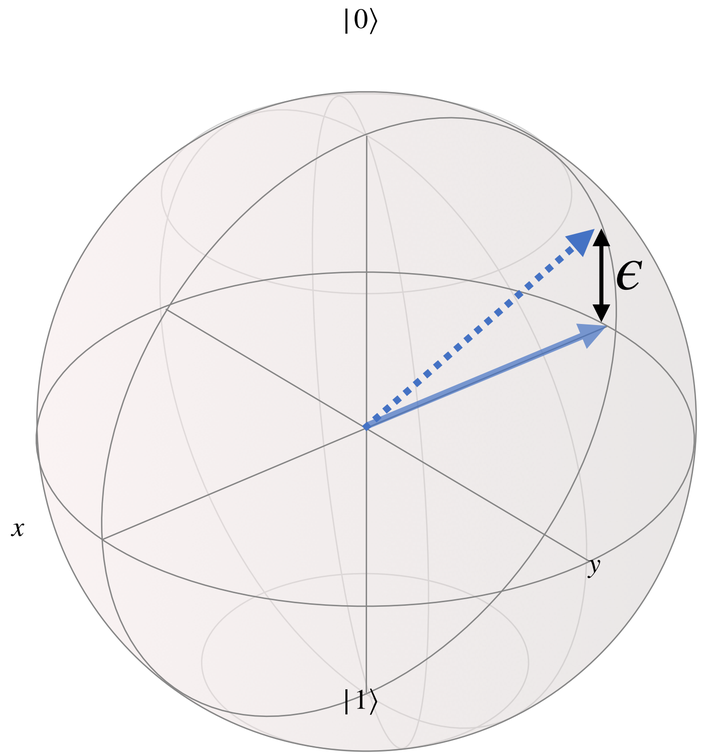
Approximate Circuit Equivalence



Approximate Circuit Equivalence



Approximate Circuit Equivalence



$$\Delta(U_C, U_{C'}) \leq \epsilon \iff C \equiv_\epsilon C'$$

Hilbert-Schmidt Distance

"easy" to compute

Hilbert-Schmidt Distance

"easy" to compute

$$\Delta_{HS}(U, U') := \sqrt{1 - \frac{|\,Tr(U^\dagger U')\,|^2}{2^{2n}}}$$

Hilbert-Schmidt Distance

"easy" to compute

$$\Delta_{HS}(U, U') := \sqrt{1 - \frac{|\,Tr(U^\dagger U')\,|^2}{2^{2n}}}$$

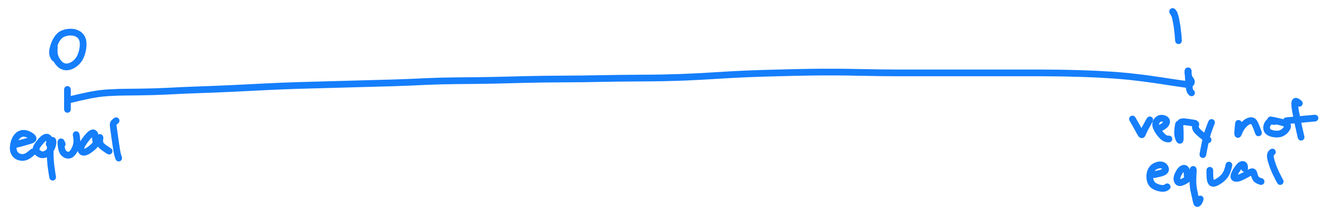
qubits

Hilbert-Schmidt Distance

"easy" to compute

$$\Delta_{HS}(U, U') := \sqrt{1 - \frac{|\,Tr(U^\dagger U')\,|^2}{2^{2n}}}$$

qubits

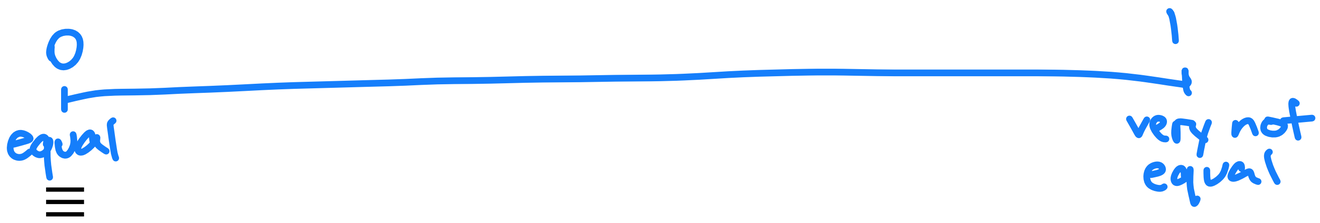


Hilbert-Schmidt Distance

"easy" to compute

$$\Delta_{HS}(U, U') := \sqrt{1 - \frac{|Tr(U^\dagger U')|^2}{2^{2n}}}$$

qubits



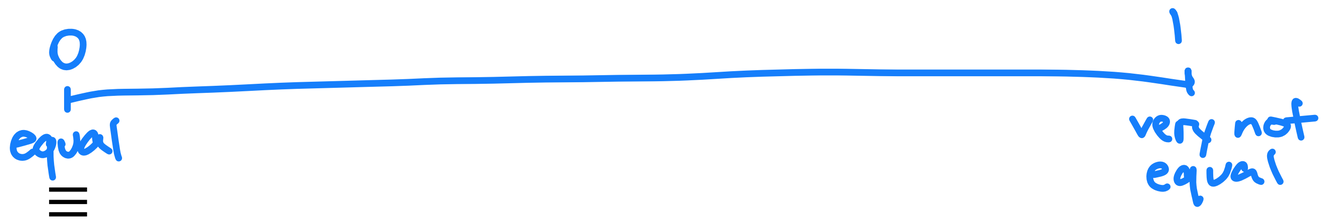
≡

Hilbert-Schmidt Distance

"easy" to compute

$$\Delta_{HS}(U, U') := \sqrt{1 - \frac{|\,Tr(U^\dagger U')\,|^2}{2^{2n}}}$$

qubits



Upper bound on total variation distance

Why Approximate Circuits?

Ideal

Circuit C

Why Approximate Circuits?

Ideal

Circuit C + No noise

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

Circuit $C' \equiv_\epsilon C$

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

Circuit $C' \equiv_\epsilon C$

k fewer gates than C

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

Circuit $C' \equiv_\epsilon C$

k fewer gates than C

The hope: $U_{C'_{noisy}}$ closer to U_C than $U_{C_{noisy}}$

Why Approximate Circuits?

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

Circuit $C' \equiv_\epsilon C$

k fewer gates than C

The hope: $U_{C'_{noisy}}$ closer to U_C than $U_{C_{noisy}}$

Noise from k gates $> \epsilon$

Why Approximate Circuits?

FTQC **requires** approximations for arbitrary angle rotations

Ideal

Circuit C + No noise = U_C

Reality

Circuit C + Noise = $U_{C_{noisy}}$

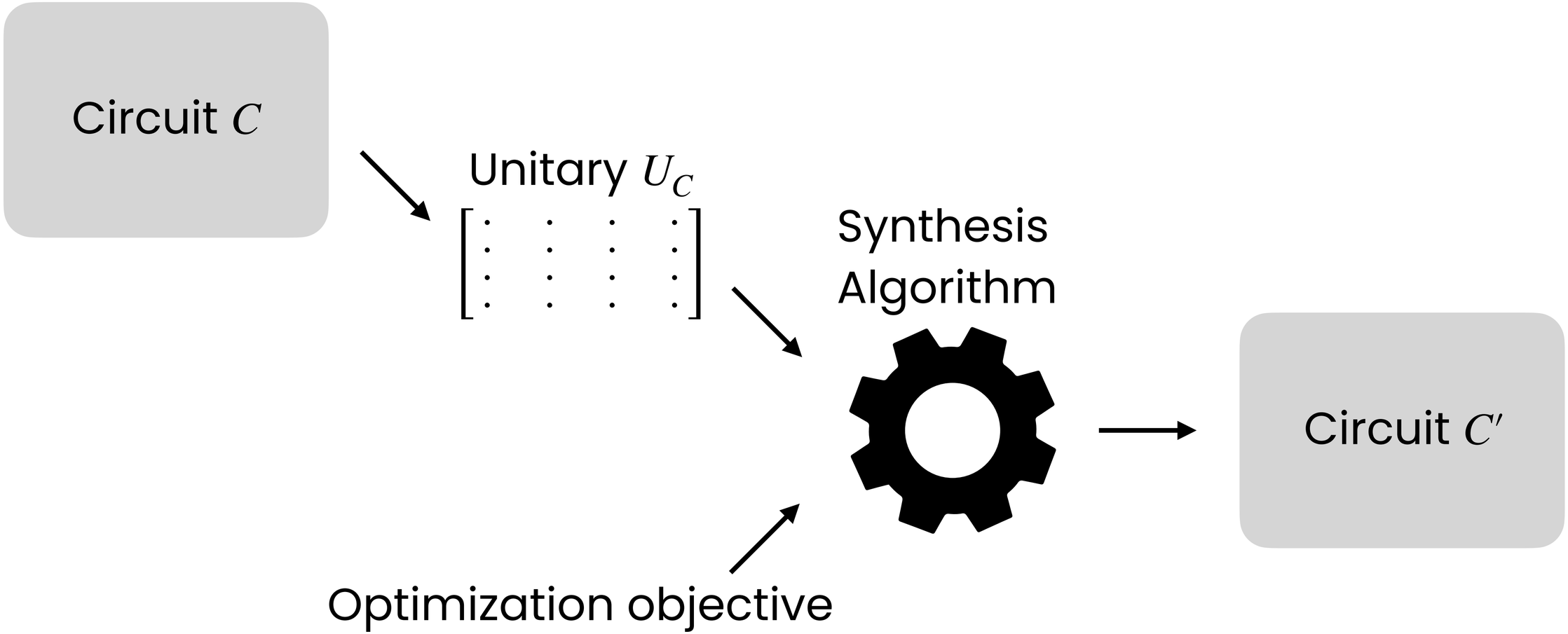
Circuit $C' \equiv_\epsilon C$

k fewer gates than C

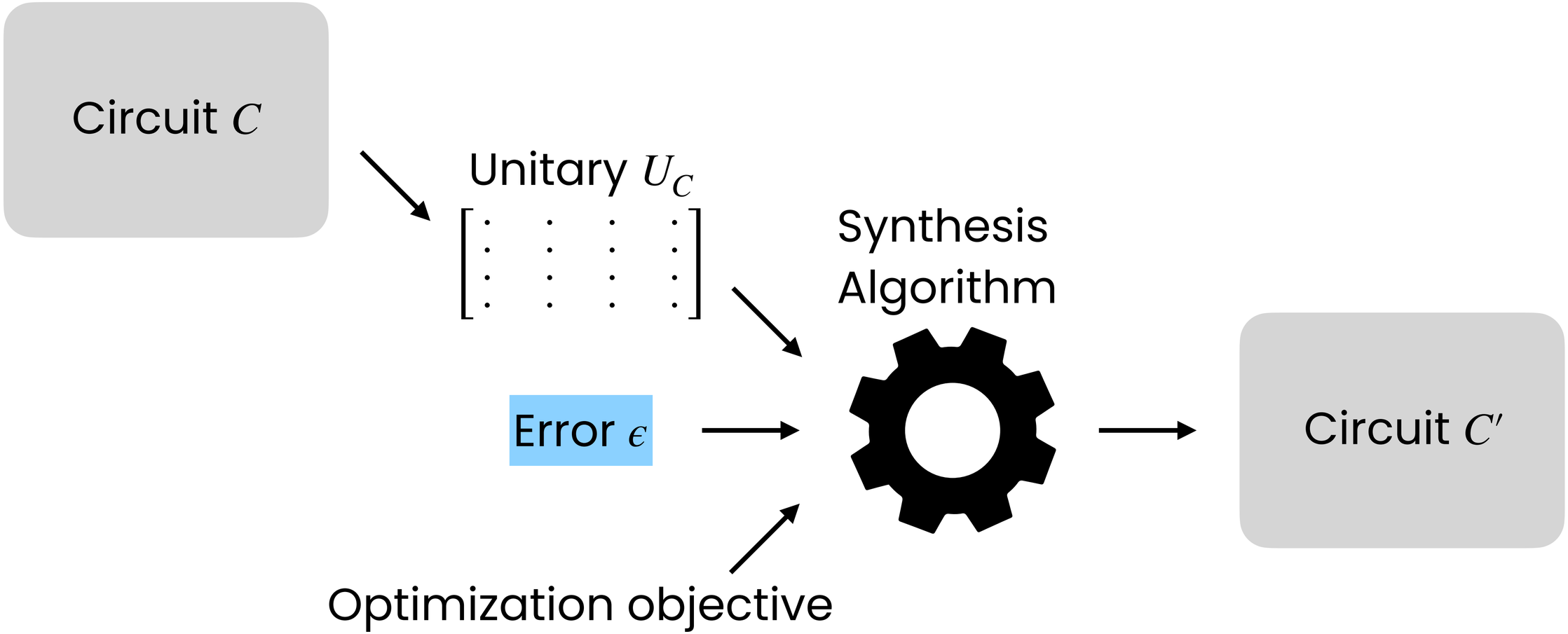
The hope: $U_{C'_{noisy}}$ closer to U_C than $U_{C_{noisy}}$

Noise from k gates $> \epsilon$

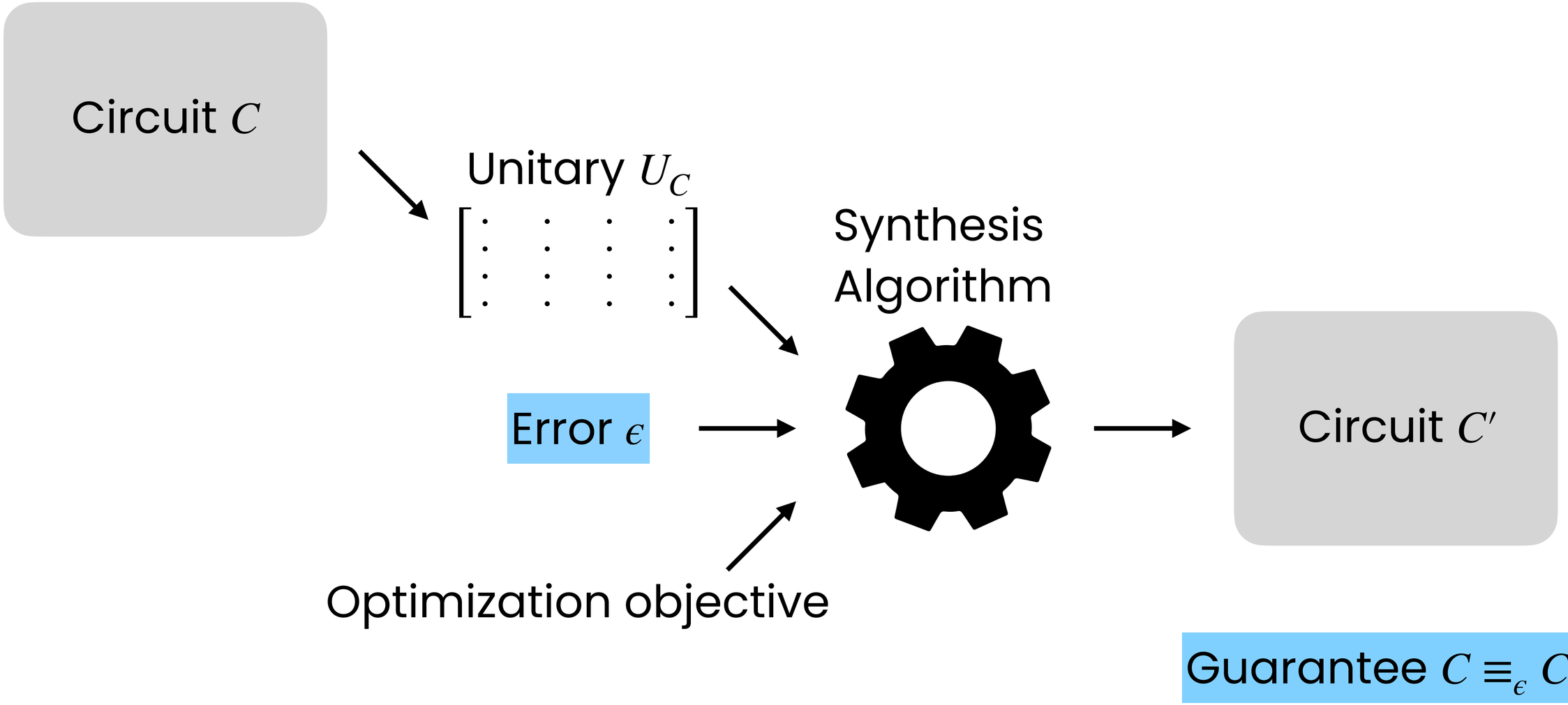
Unitary Synthesis



Unitary Synthesis

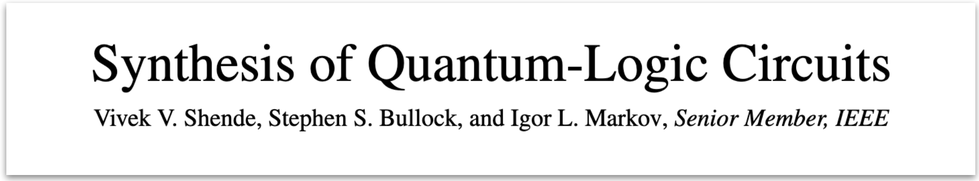


Unitary Synthesis



Rich Research Area

Rich Research Area


Synthesis of Quantum-Logic Circuits

Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov, *Senior Member, IEEE*

Rich Research Area

Synthesis of Quantum-Logic Circuits

Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov, *Senior Member, IEEE*

Towards Optimal Topology Aware Quantum Circuit Synthesis

Marc G. Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi
University of California Berkeley
{marc.davis,ethanhs,anamtudor,ksen,irfan_siddiqi}@berkeley.edu

Costin Iancu
Lawrence Berkeley National Laboratory
cciancu@lbl.gov

Rich Research Area

Synthesis of Quantum-Logic Circuits

Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov, *Senior Member, IEEE*

Towards Optimal Topology Aware Quantum Circuit Synthesis

Marc G. Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi
University of California Berkeley
{marc.davis,ethanhs,anamtudor,ksen,irfan_sidd

Costin Iancu
Lawrence Berkeley National Laboratory

LEAP: Scaling Numerical Optimization Based Synthesis Using an Incremental Approach

ETHAN SMITH and MARC GRAU DAVIS, University of California, Berkeley
JEFFREY LARSON, Argonne National Laboratory
ED YOUNIS, LINDSAY BASSMAN OFTELIE, WIM LAVRIJSEN, and COSTIN IANCU,
Lawrence Berkeley National Laboratory

Rich Research Area

Synthesis of Quantum-Logic Circuits

Vivek V. Shende, Stephen S. Bullock, and Igor L. Markov, *Senior Member, IEEE*

Towards Optimal Topology Aware Quantum Circuit Synthesis

Marc G. Davis, Ethan Smith, Ana Tudor, Koushik Sen, Irfan Siddiqi
University of California Berkeley
{marc.davis,ethanhs,anamtudor,ksen,irfan_sidd

Costin Iancu
Lawrence Berkeley National Laboratory

LEAP: Scaling Numerical Optimization Based Synthesis Using an Incremental Approach

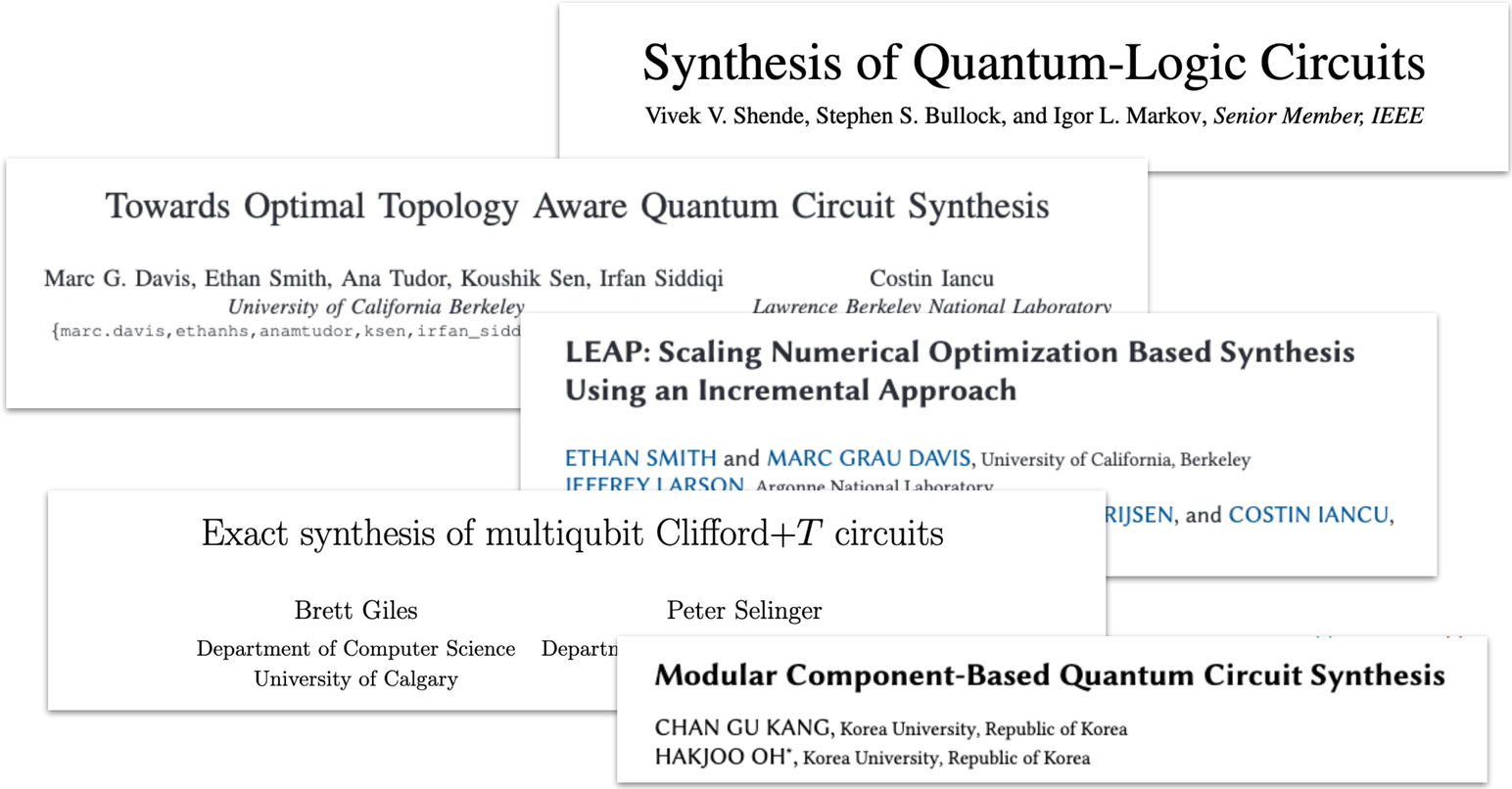
ETHAN SMITH and MARC GRAU DAVIS, University of California, Berkeley
JEFFREY LARSON, Argonne National Laboratory
RIJSEN, and COSTIN IANCU,

Exact synthesis of multiqubit Clifford+T circuits

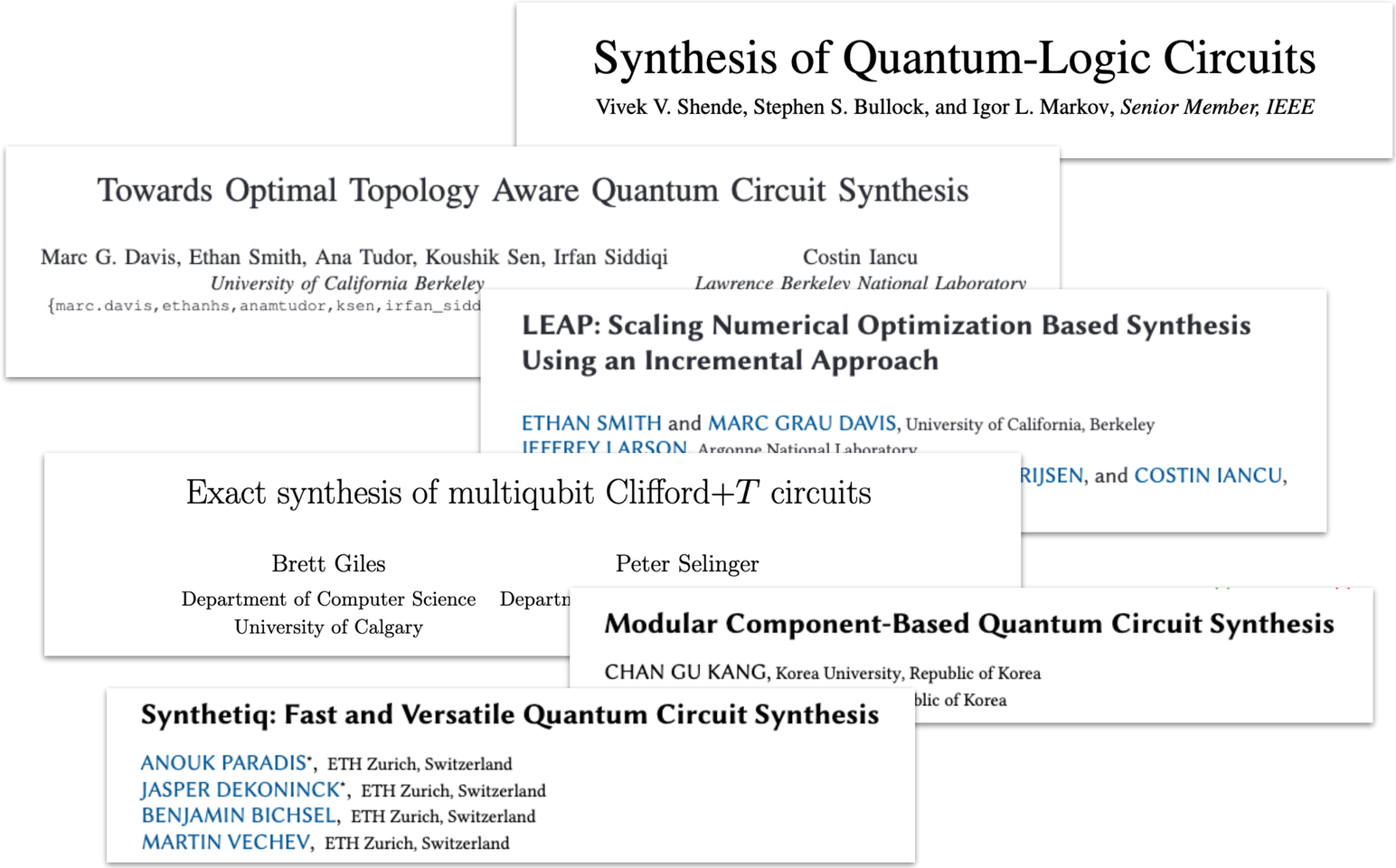
Brett Giles
Department of Computer Science
University of Calgary

Peter Selinger
Department of Mathematics and Statistics
Dalhousie University

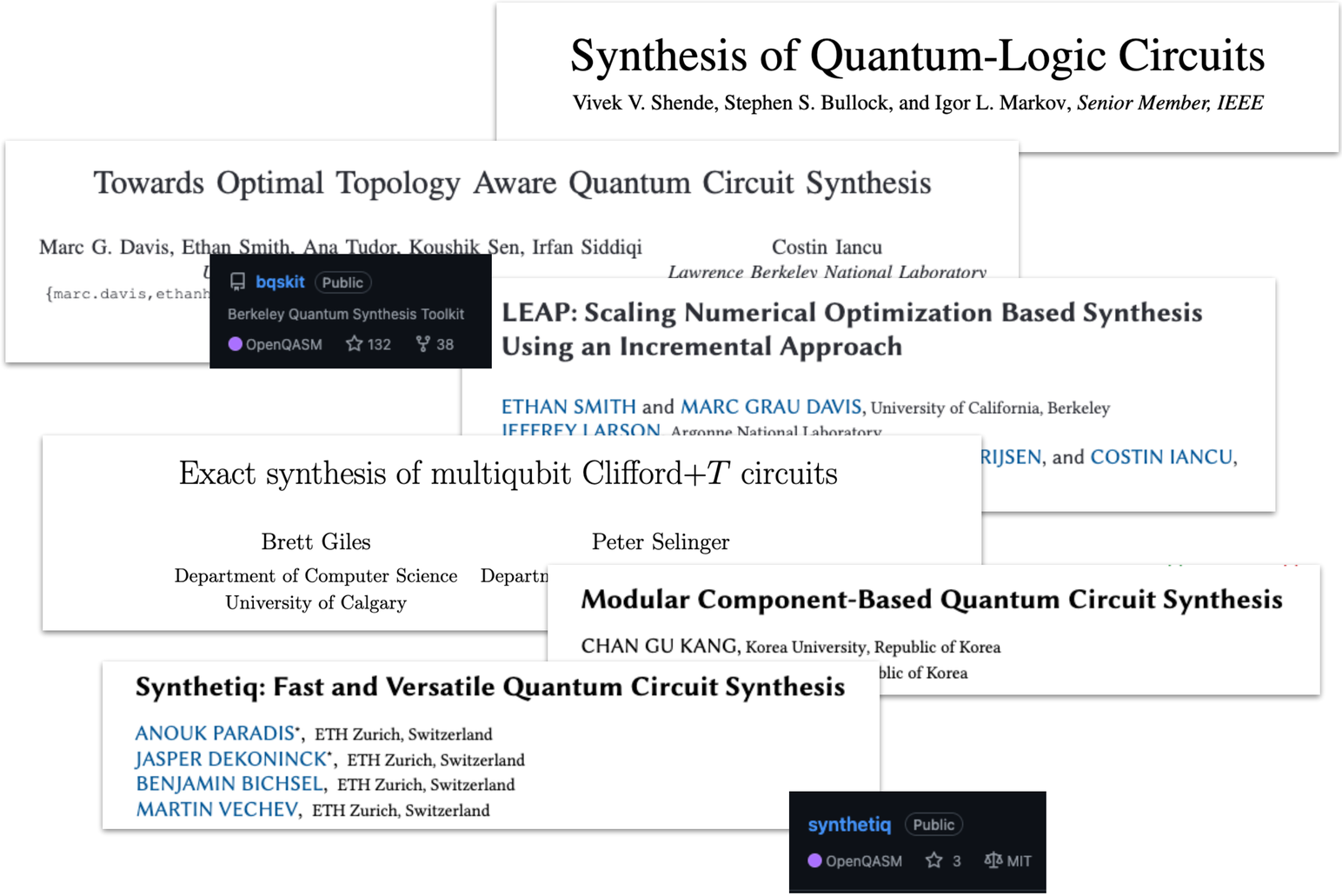
Rich Research Area



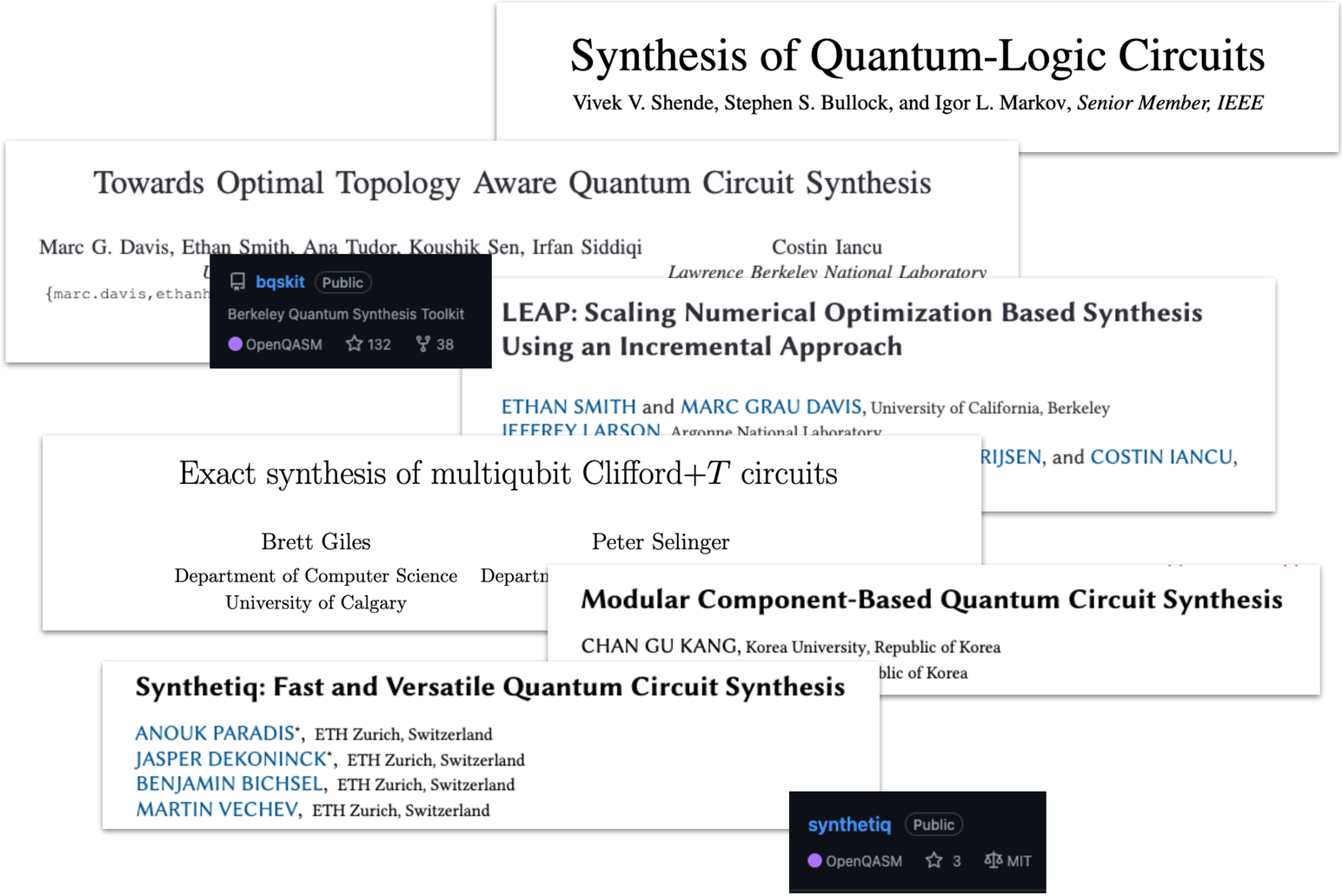
Rich Research Area



Rich Research Area



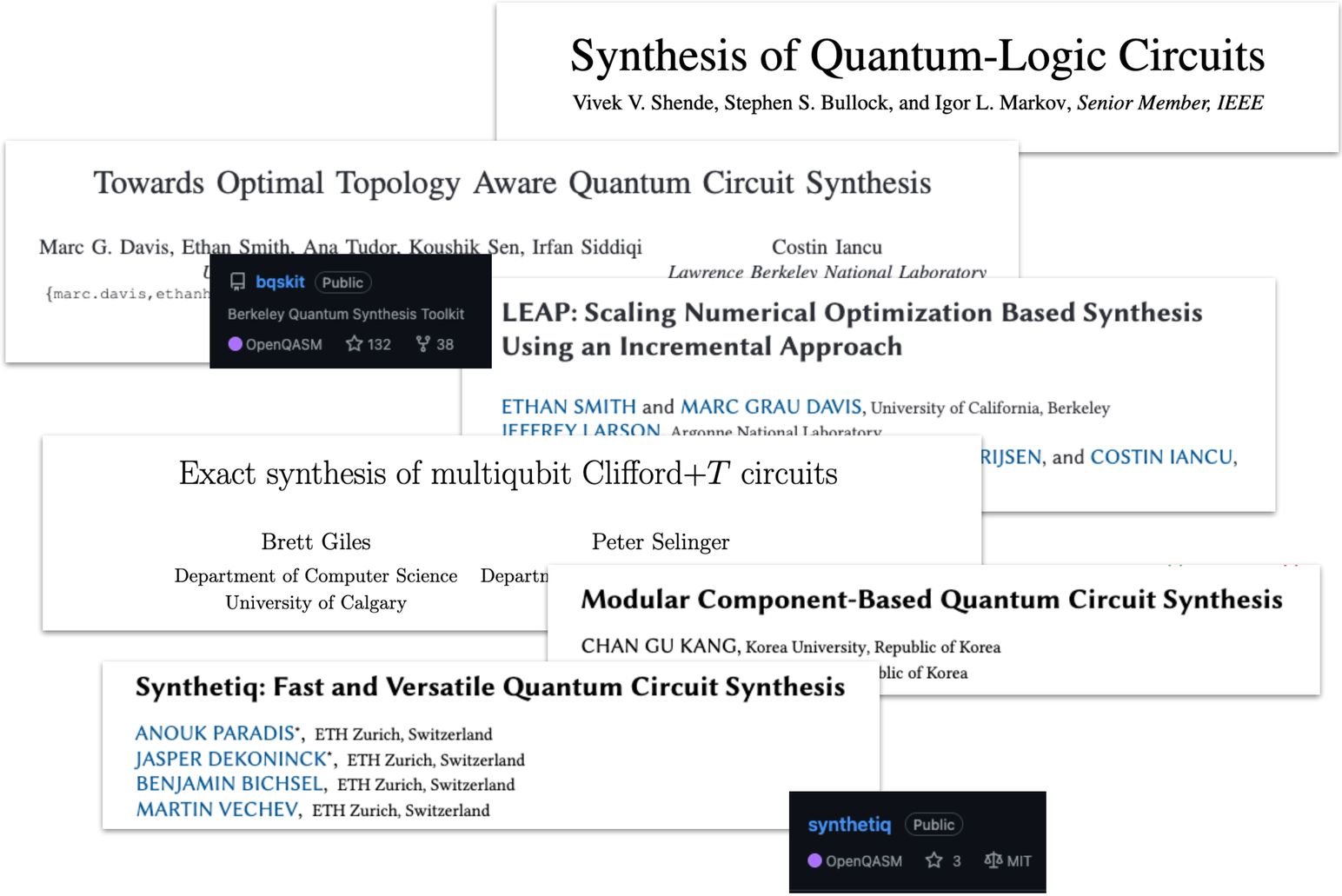
Rich Research Area



Techniques:

- analytical

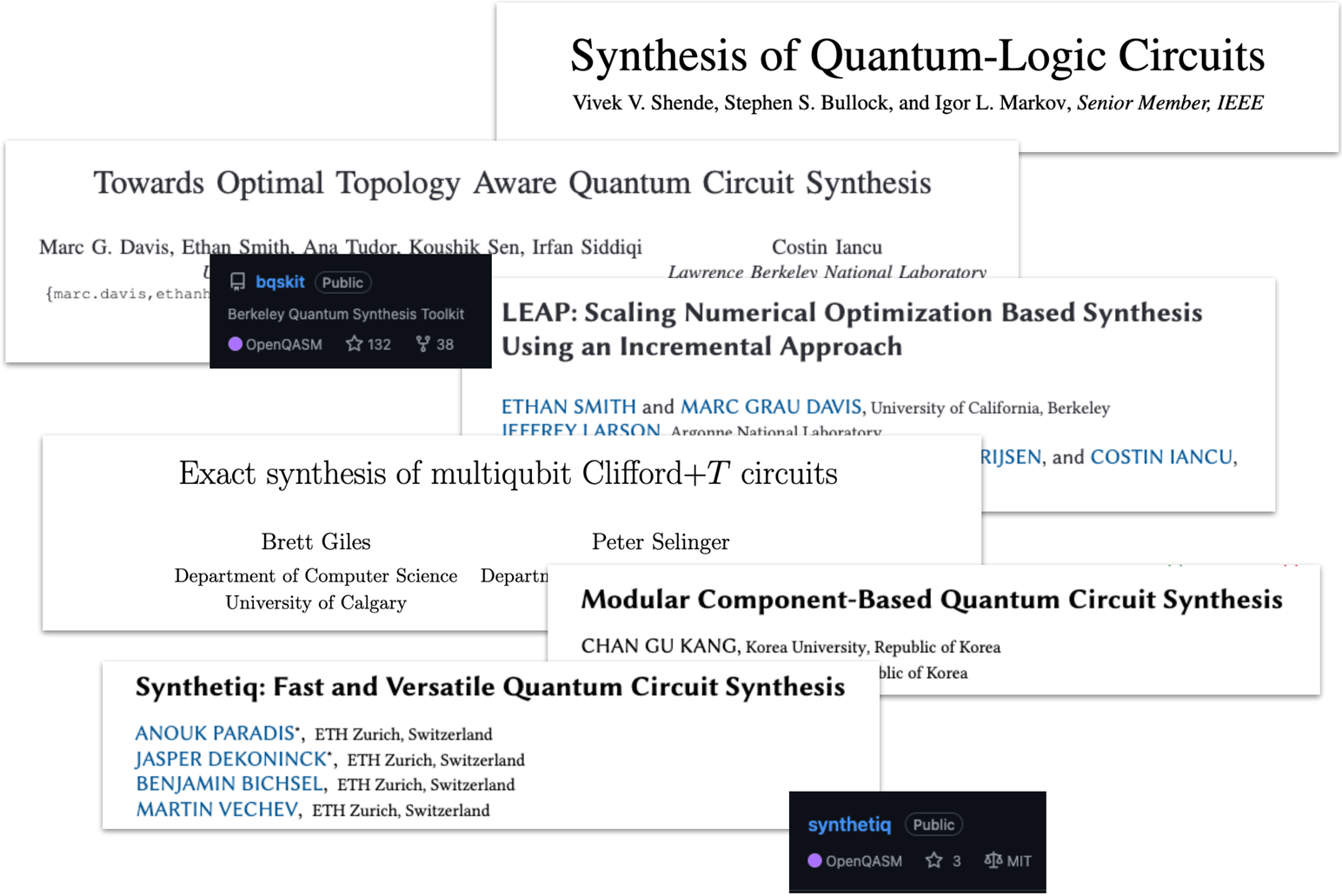
Rich Research Area



Techniques:

- analytical
- numerical optimization

Rich Research Area

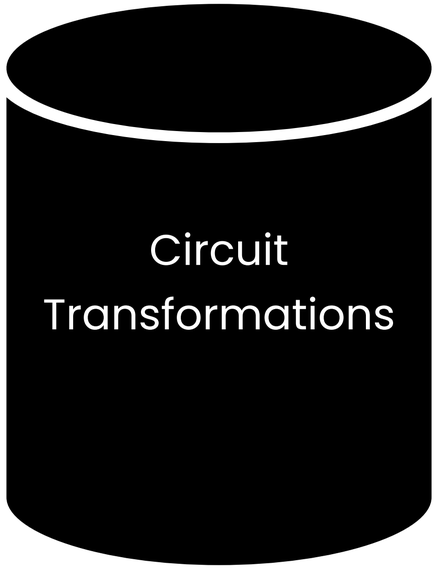


Techniques:

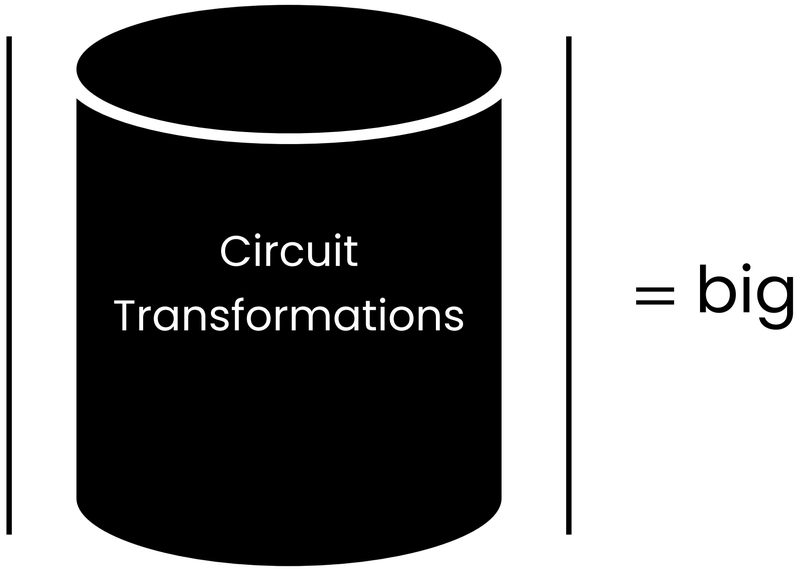
- analytical
- numerical optimization
- explicit search

Scheduling Transformations

Scheduling is hard

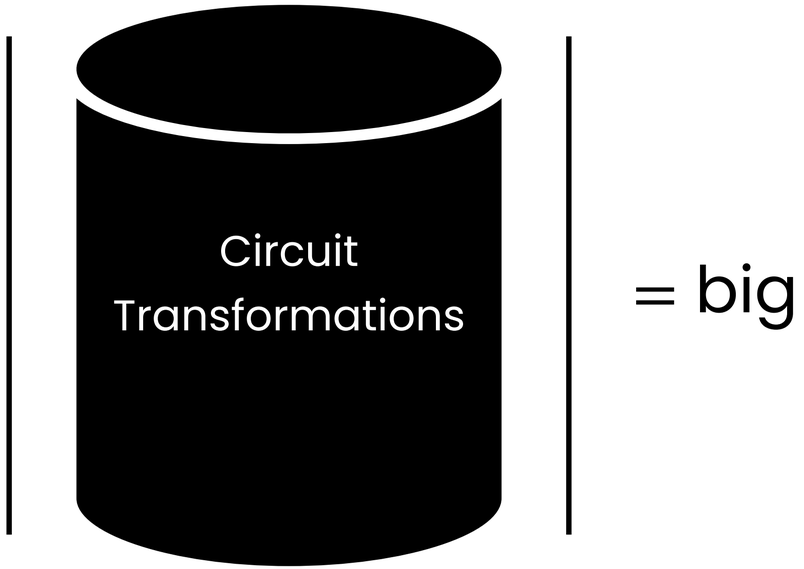


Scheduling is hard



Scheduling is hard

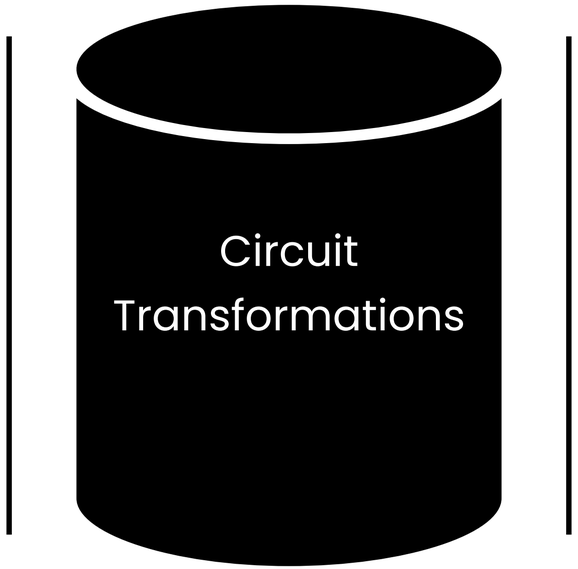
In what order to apply???



= big

Scheduling is hard

In what order to apply???



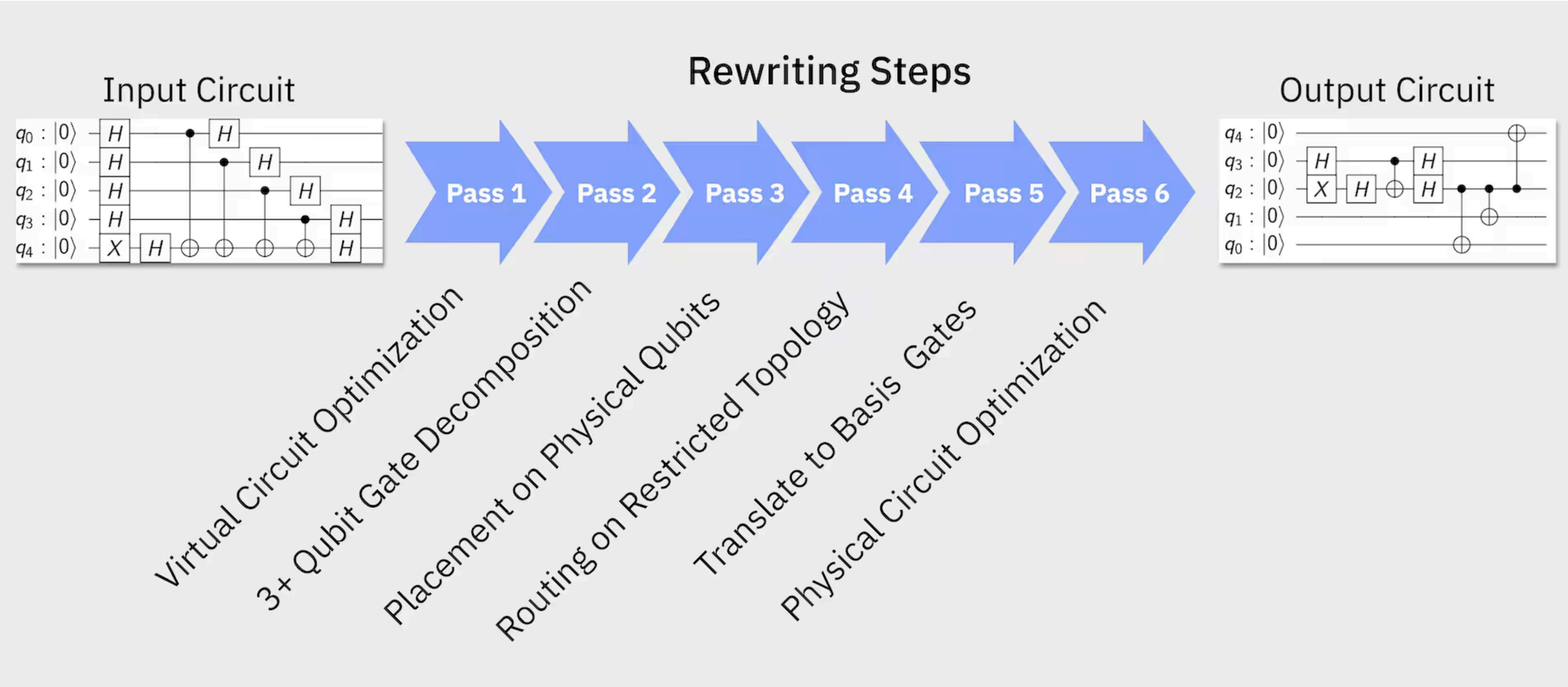
= big

"phase-ordering problem"

Qiskit

https://docs.quantum.ibm.com/api/qiskit/transpiler#optimization-stage

https://github.com/Qiskit/qiskit/blob/stable/1.4/qiskit/transpiler/preset_passmanagers/level3.py



Qiskit

https://docs.quantum.ibm.com/api/qiskit/transpiler#optimization-stage

https://github.com/Qiskit/qiskit/blob/stable/1.4/qiskit/transpiler/preset_passmanagers/level3.py

qiskit / qiskit / transpiler / preset_passmanagers / **level3.py** ↑ Top

Code | Blame | 119 lines (104 loc) · 4.8 KB · | Raw

```
26
27  def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
28      """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
29      gate cancellation using commutativity rules and unitary synthesis.
30
31      This pass manager applies the user-given initial layout. If none is given, a search
32      for a perfect layout (i.e. one that satisfies all 2-qubit interactions) is conducted.
33      If no such layout is found, and device calibration information is available, the
34      circuit is mapped to the qubits with best readouts and to CX gates with highest fidelit
35
36      The pass manager then transforms the circuit to match the coupling constraints.
37      It is then unrolled to the basis, and any flipped cx directions are fixed.
38      Finally, optimizations in the form of commutative gate cancellation, resynthesis
39      of two-qubit unitary blocks, and redundant reset removal are performed.
```

Qiskit

https://docs.quantum.ibm.com/api/qiskit/transpiler#optimization-stage

https://github.com/Qiskit/qiskit/blob/stable/1.4/qiskit/transpiler/preset_passmanagers/level3.py

qiskit / qiskit / transpiler / preset_passmanagers / **level3.py** ↑ Top

Code | Blame | 119 lines (104 loc) · 4.8 KB · | Raw

```
26
27  def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
28      """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
29      gate cancellation using commutativity rules and unitary synthesis.
30
31      This pass manager applies the user-given initial layout. If none is given, a search
32      for a perfect layout (i.e. one that satisfies all 2-qubit interactions) is conducted.
33      If no such layout is found, and device calibration information is available, the
34      circuit is mapped to the qubits with best readouts and to CX gates with highest fidelit
35
36      The pass manager then transforms the circuit to match the coupling constraints.
37      It is then unrolled to the basis, and any flipped cx directions are fixed.
38      Finally, optimizations in the form of commutative gate cancellation, resynthesis
39      of two-qubit unitary blocks, and redundant reset removal are performed.
```

Qiskit

https://docs.quantum.ibm.com/api/qiskit/transpiler#optimization-stage

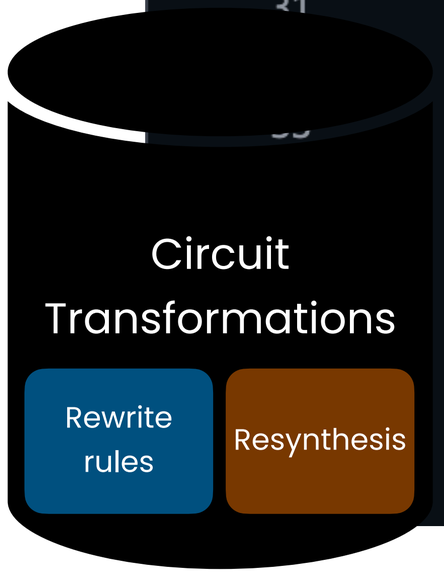
https://github.com/Qiskit/qiskit/blob/stable/1.4/qiskit/transpiler/preset_passmanagers/level3.py

qiskit / qiskit / transpiler / preset_passmanagers / **level3.py** ↑ Top

Code | Blame | 119 lines (104 loc) · 4.8 KB · | Raw

```
26
27  def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
28      """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
29      gate cancellation using commutativity rules and unitary synthesis.
30
31      This pass manager applies the user-given initial layout. If none is given, a search
        for a perfect layout (i.e. one that satisfies all 2-qubit interactions) is conducted.
        If no such layout is found, and device calibration information is available, the
        circuit is mapped to the qubits with best readouts and to CX gates with highest fidelit

        The pass manager then transforms the circuit to match the coupling constraints.
        It is then unrolled to the basis, and any flipped cx directions are fixed.
        Finally, optimizations in the form of commutative gate cancellation, resynthesis
        of two-qubit unitary blocks, and redundant reset removal are performed.
```



Qiskit

https://docs.quantum.ibm.com/api/qiskit/transpiler#optimization-stage

https://github.com/Qiskit/qiskit/blob/stable/1.4/qiskit/transpiler/preset_passmanagers/level3.py

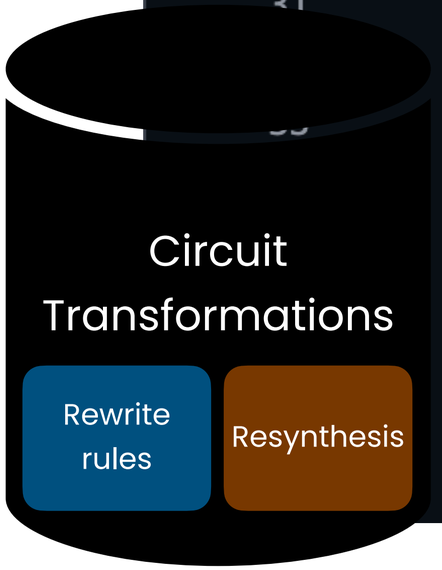
qiskit / qiskit / transpiler / preset_passmanagers / **level3.py** ↑ Top

Code | Blame | 119 lines (104 loc) · 4.8 KB · | Raw

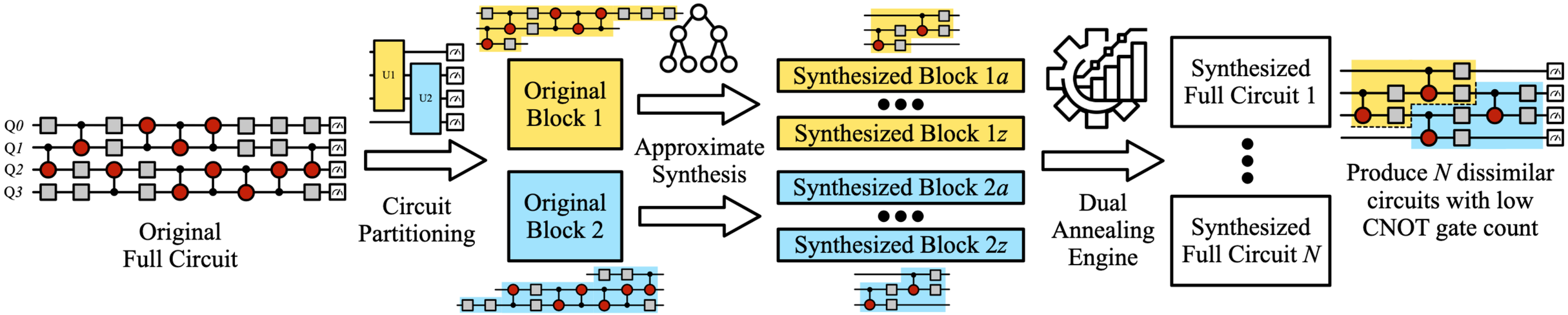
```
26
27  def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
28      """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
29      gate cancellation using commutativity rules and unitary synthesis.
30
31      This pass manager applies the user-given initial layout. If none is given, a search
        for a perfect layout (i.e. one that satisfies all 2-qubit interactions) is conducted.
        If no such layout is found, and device calibration information is available, the
        circuit is mapped to the qubits with best readouts and to CX gates with highest fidelit

        The pass manager then tra                                    he coupling constraints.
        It is then unrolled to th                                    rections are fixed.
        Finally, optimizations in the form of commutative gate cancellation, resynthesis
        of two-qubit unitary blocks, and redundant reset removal are performed.
```

coarse fixed passes



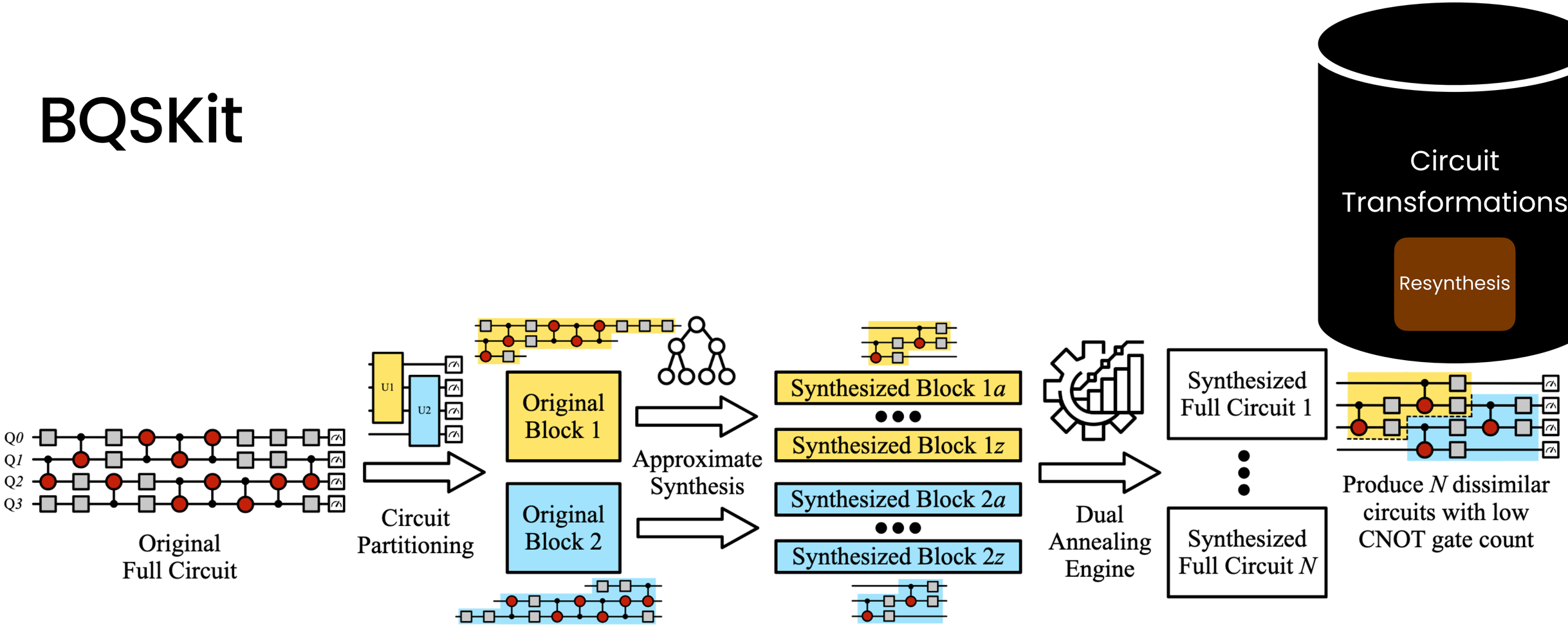
BQSKit



QUEST: systematically approximating quantum circuits for higher output fidelity [Patel et al. 2022]

https://github.com/BQSKit/bqskit

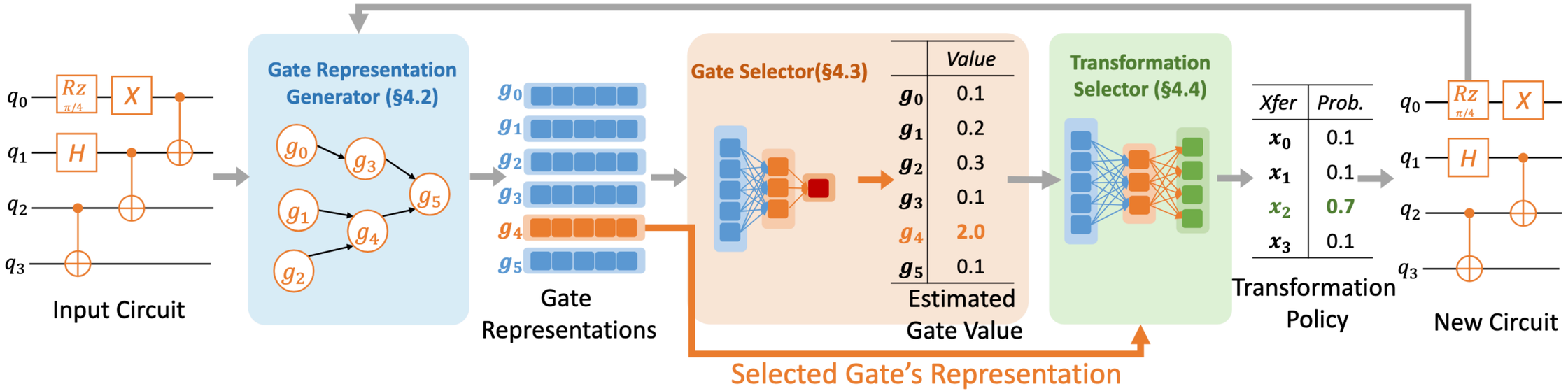
BQSKit



QUEST: systematically approximating quantum circuits for higher output fidelity [Patel et al. 2022]

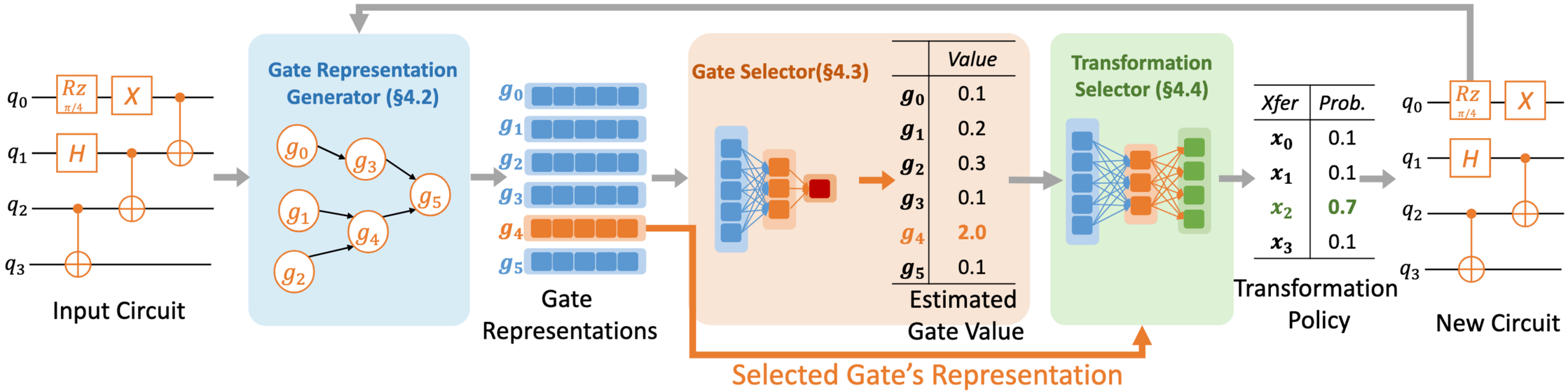
https://github.com/BQSKit/bqskit

Quarl



Quarl: A Learning-Based Quantum Circuit Optimizer [Li et al. 2024]

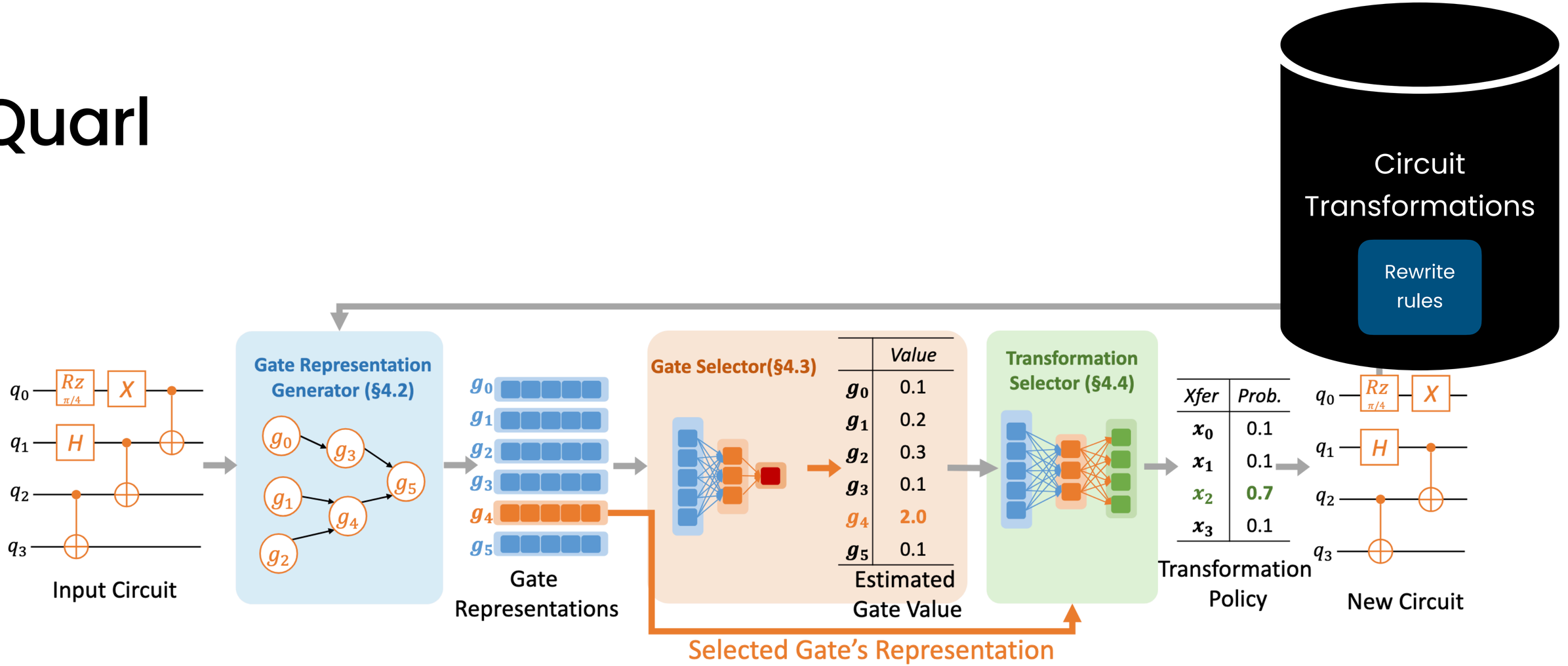
Quarl



* requires an NVIDIA A100 GPU ($$$)

Quarl: A Learning-Based Quantum Circuit Optimizer [Li et al. 2024]

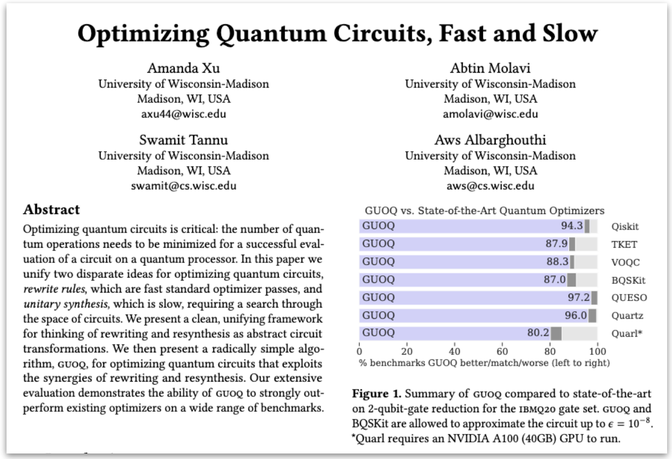
Quarl



* requires an NVIDIA A100 GPU ($$$)

Quarl: A Learning-Based Quantum Circuit Optimizer [Li et al. 2024]

GUOQ



Optimizing Quantum Circuits, Fast and Slow

Amanda Xu
University of Wisconsin-Madison
Madison, WI, USA
axu44@wisc.edu

Abtin Molavi
University of Wisconsin-Madison
Madison, WI, USA
amolavi@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

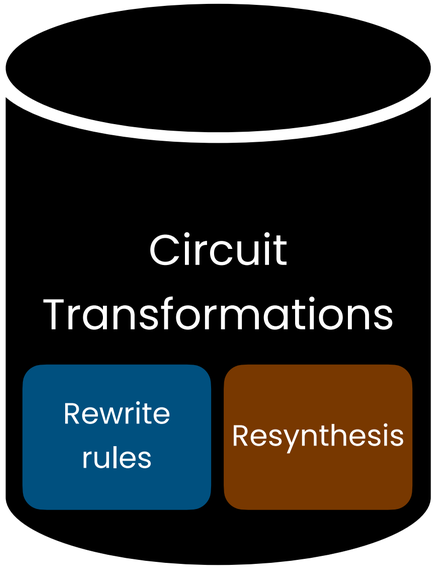
Aws Albarghouthi
University of Wisconsin-Madison
Madison, WI, USA
aws@cs.wisc.edu

Abstract

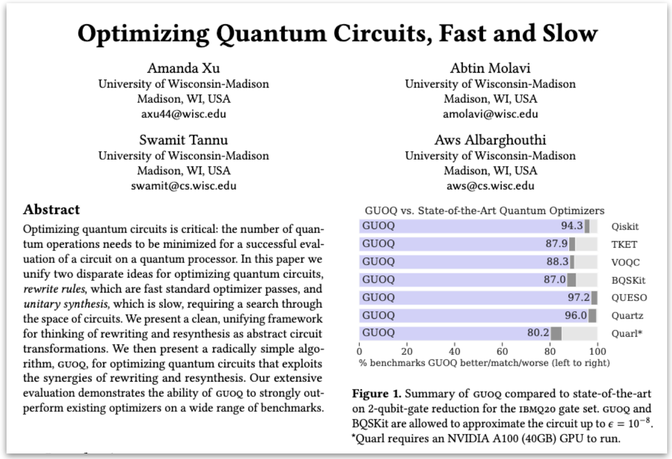
Optimizing quantum circuits is critical: the number of quantum operations needs to be minimized for a successful evaluation of a circuit on a quantum processor. In this paper we unify two disparate ideas for optimizing quantum circuits, *rewrite rules*, which are fast standard optimizer passes, and *unitary synthesis*, which is slow, requiring a search through the space of circuits. We present a clean, unifying framework for thinking of rewriting and resynthesis as abstract circuit transformations. We then present a radically simple algorithm, GUOQ, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOQ to strongly outperform existing optimizers on a wide range of benchmarks.

Figure 1. Summary of GUOQ compared to state-of-the-art on 2-qubit-gate reduction for the IBMQ20 gate set. GUOQ and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quarl requires an NVIDIA A100 (40GB) GPU to run.

ASPLOS talk on April 1st!



GUOQ



Optimizing Quantum Circuits, Fast and Slow

Amanda Xu
University of Wisconsin-Madison
Madison, WI, USA
axu44@wisc.edu

Abtin Molavi
University of Wisconsin-Madison
Madison, WI, USA
amolavi@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

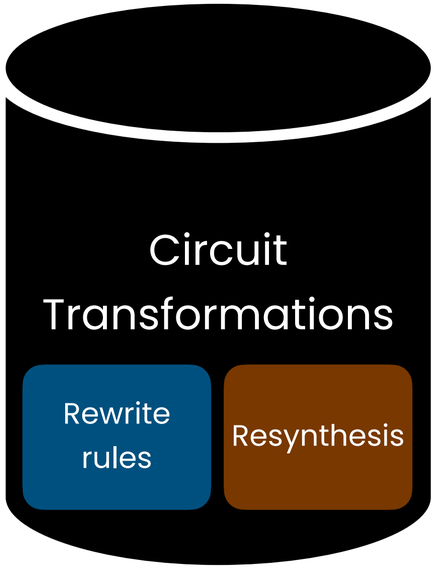
Aws Albarghouthi
University of Wisconsin-Madison
Madison, WI, USA
aws@cs.wisc.edu

Abstract

Optimizing quantum circuits is critical: the number of quantum operations needs to be minimized for a successful evaluation of a circuit on a quantum processor. In this paper we unify two disparate ideas for optimizing quantum circuits, *rewrite rules*, which are fast standard optimizer passes, and *unitary synthesis*, which is slow, requiring a search through the space of circuits. We present a clean, unifying framework for thinking of rewriting and resynthesis as abstract circuit transformations. We then present a radically simple algorithm, GUOQ, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOQ to strongly outperform existing optimizers on a wide range of benchmarks.

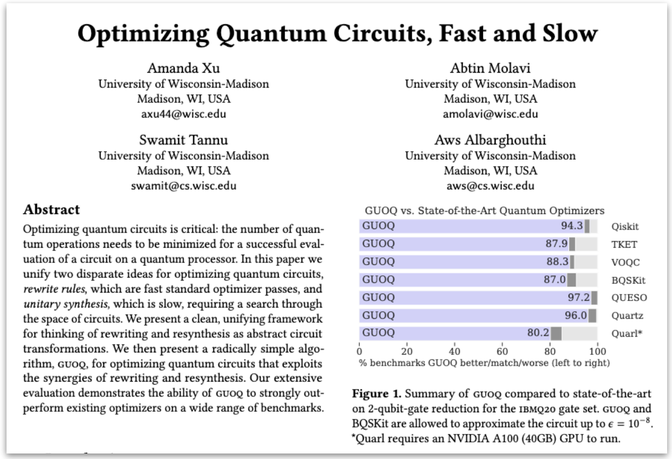
Figure 1. Summary of GUOQ compared to state-of-the-art on 2-qubit-gate reduction for the IBMQ20 gate set. GUOQ and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quarl requires an NVIDIA A100 (40GB) GPU to run.

ASPLOS talk on April 1st!



1. Randomly pick a transformation.
2. Randomly pick a subcircuit to transform.
3. Accept the result if better than current. Else, reject with high probability.

GUOQ



Optimizing Quantum Circuits, Fast and Slow

Amanda Xu
University of Wisconsin-Madison
Madison, WI, USA
axu44@wisc.edu

Abtin Molavi
University of Wisconsin-Madison
Madison, WI, USA
amolavi@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

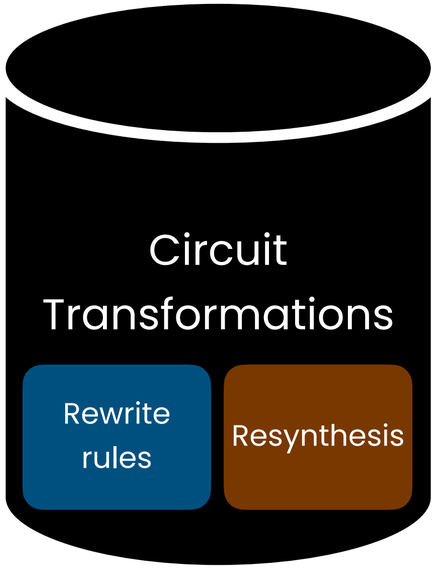
Aws Albarghouthi
University of Wisconsin-Madison
Madison, WI, USA
aws@cs.wisc.edu

Abstract

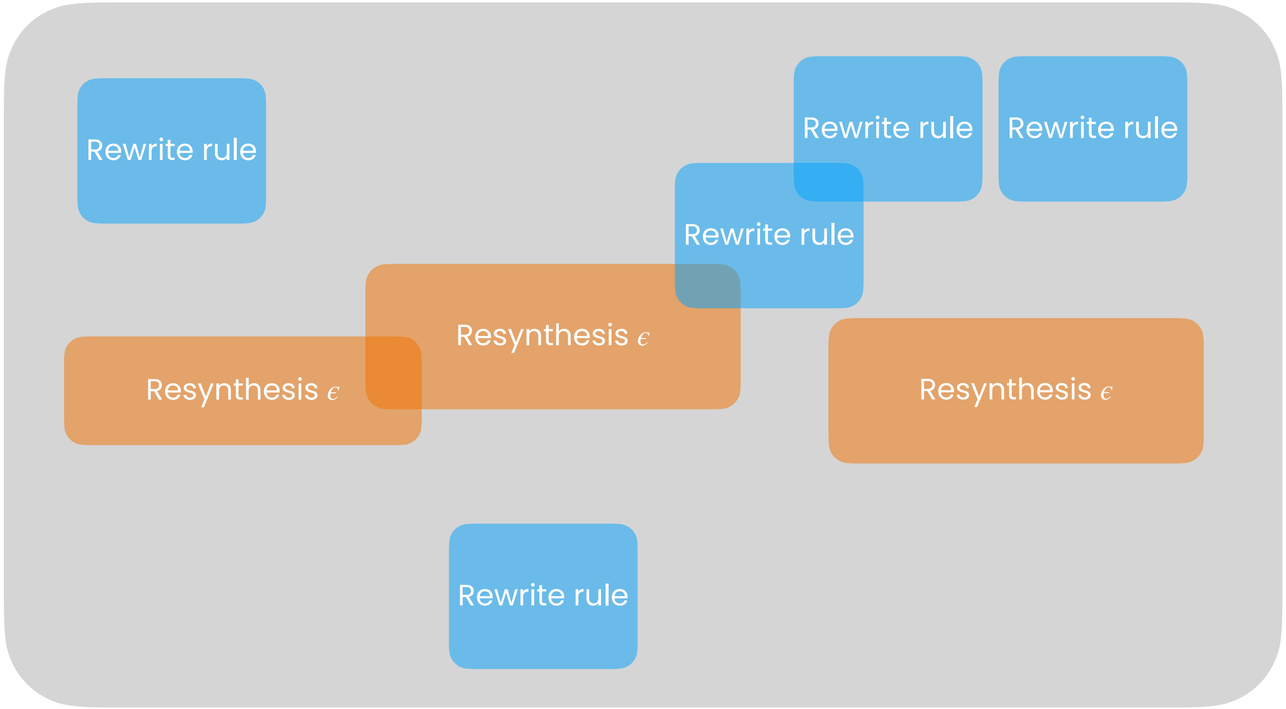
Optimizing quantum circuits is critical: the number of quantum operations needs to be minimized for a successful evaluation of a circuit on a quantum processor. In this paper we unify two disparate ideas for optimizing quantum circuits, *rewrite rules*, which are fast standard optimizer passes, and *unitary synthesis*, which is slow, requiring a search through the space of circuits. We present a clean, unifying framework for thinking of rewriting and resynthesis as abstract circuit transformations. We then present a radically simple algorithm, GUOQ, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOQ to strongly outperform existing optimizers on a wide range of benchmarks.

Figure 1. Summary of GUOQ compared to state-of-the-art on 2-qubit-gate reduction for the IBMQ20 gate set. GUOQ and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quarl requires an NVIDIA A100 (40GB) GPU to run.

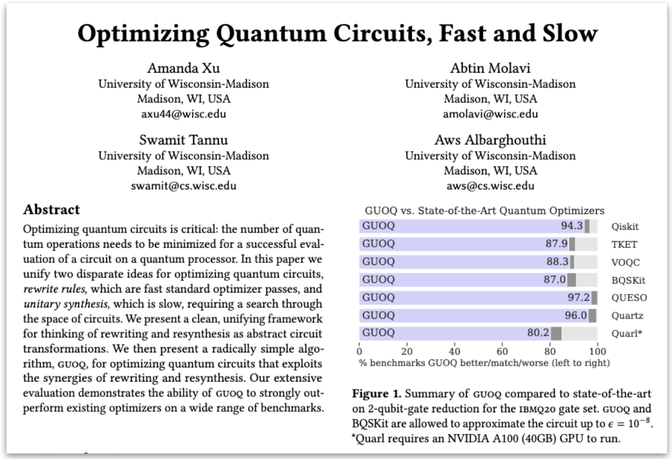
ASPLOS talk on April 1st!



1. Randomly pick a transformation.
2. Randomly pick a subcircuit to transform.
3. Accept the result if better than current. Else, reject with high probability.



GUOQ



Optimizing Quantum Circuits, Fast and Slow

Amanda Xu
University of Wisconsin-Madison
Madison, WI, USA
axu44@wisc.edu

Abtin Molavi
University of Wisconsin-Madison
Madison, WI, USA
amolavi@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

Aws Albarghouthi
University of Wisconsin-Madison
Madison, WI, USA
aws@cs.wisc.edu

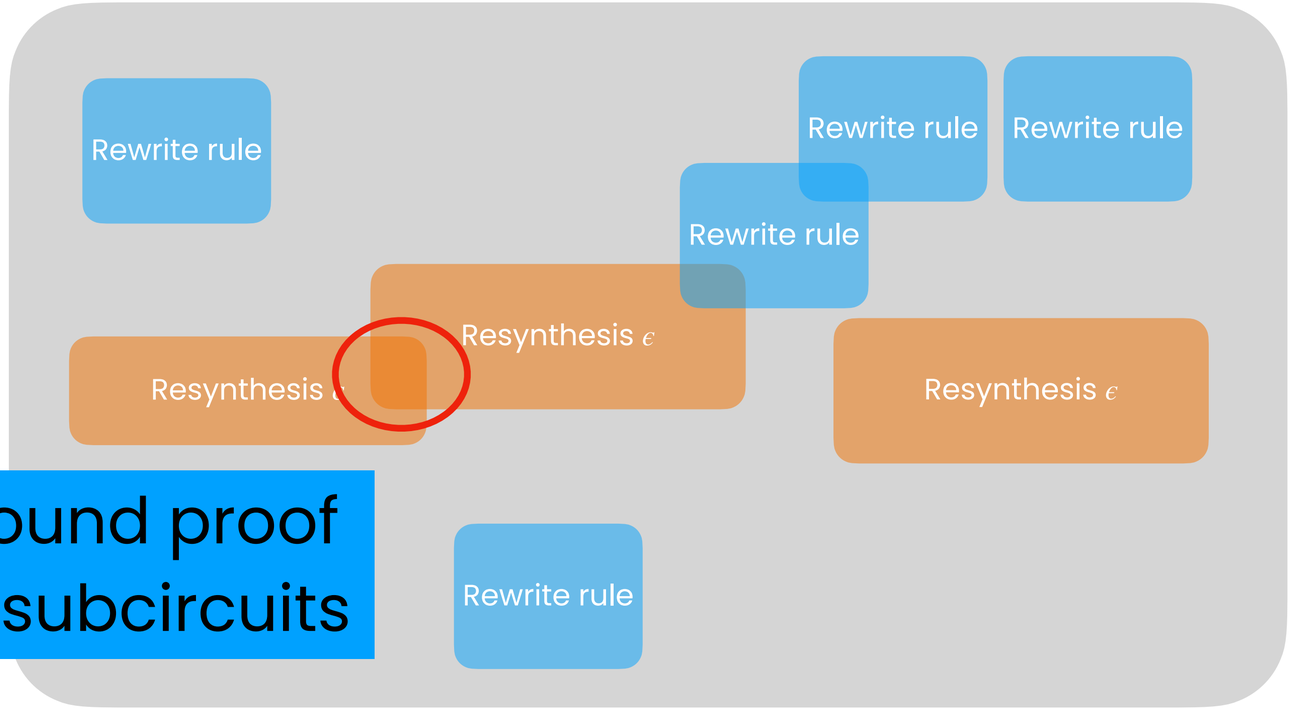
Abstract

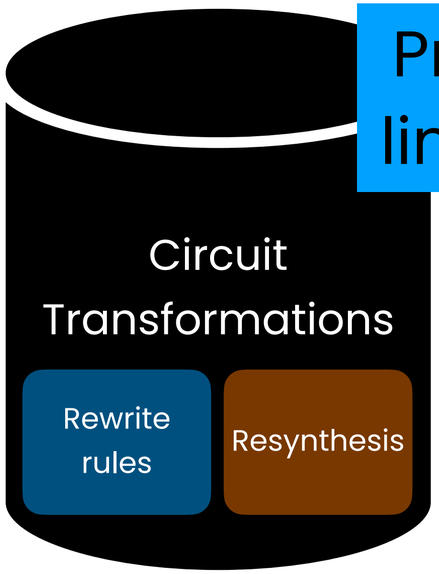
Optimizing quantum circuits is critical: the number of quantum operations needs to be minimized for a successful evaluation of a circuit on a quantum processor. In this paper we unify two disparate ideas for optimizing quantum circuits, *rewrite rules*, which are fast standard optimizer passes, and *unitary synthesis*, which is slow, requiring a search through the space of circuits. We present a clean, unifying framework for thinking of rewriting and resynthesis as abstract circuit transformations. We then present a radically simple algorithm, GUOQ, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOQ to strongly outperform existing optimizers on a wide range of benchmarks.

Figure 1. Summary of GUOQ compared to state-of-the-art on 2-qubit-gate reduction for the IBMQ20 gate set. GUOQ and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quarl requires an NVIDIA A100 (40GB) GPU to run.

ASPLOS talk on April 1st!

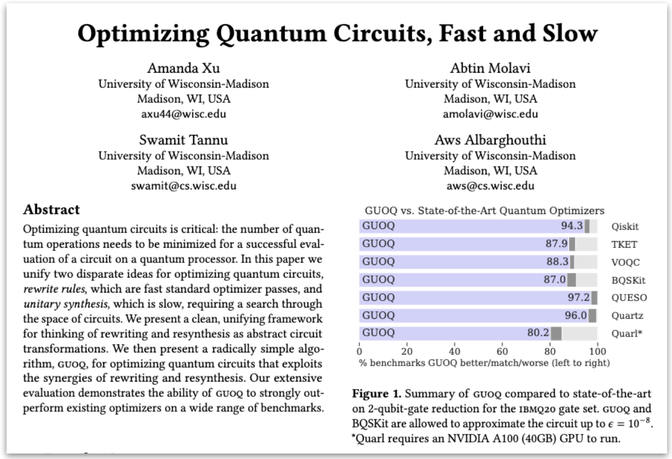
1. Randomly pick a transformation.
2. Randomly pick a subcircuit to transform.
3. Accept the result if better than current. Else, reject with high probability.





Prior work: error bound proof limited to **disjoint** subcircuits

GUOQ



Optimizing Quantum Circuits, Fast and Slow

Amanda Xu
University of Wisconsin-Madison
Madison, WI, USA
axu44@wisc.edu

Abtin Molavi
University of Wisconsin-Madison
Madison, WI, USA
amolavi@wisc.edu

Swamit Tannu
University of Wisconsin-Madison
Madison, WI, USA
swamit@cs.wisc.edu

Aws Albarghouthi
University of Wisconsin-Madison
Madison, WI, USA
aws@cs.wisc.edu

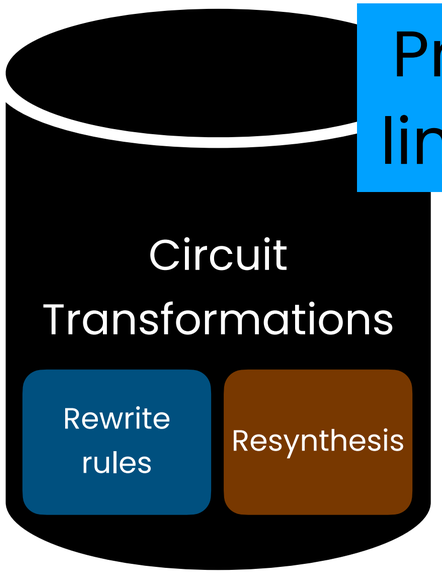
Abstract

Optimizing quantum circuits is critical: the number of quantum operations needs to be minimized for a successful evaluation of a circuit on a quantum processor. In this paper we unify two disparate ideas for optimizing quantum circuits, *rewrite rules*, which are fast standard optimizer passes, and *unitary synthesis*, which is slow, requiring a search through the space of circuits. We present a clean, unifying framework for thinking of rewriting and resynthesis as abstract circuit transformations. We then present a radically simple algorithm, GUOQ, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOQ to strongly outperform existing optimizers on a wide range of benchmarks.

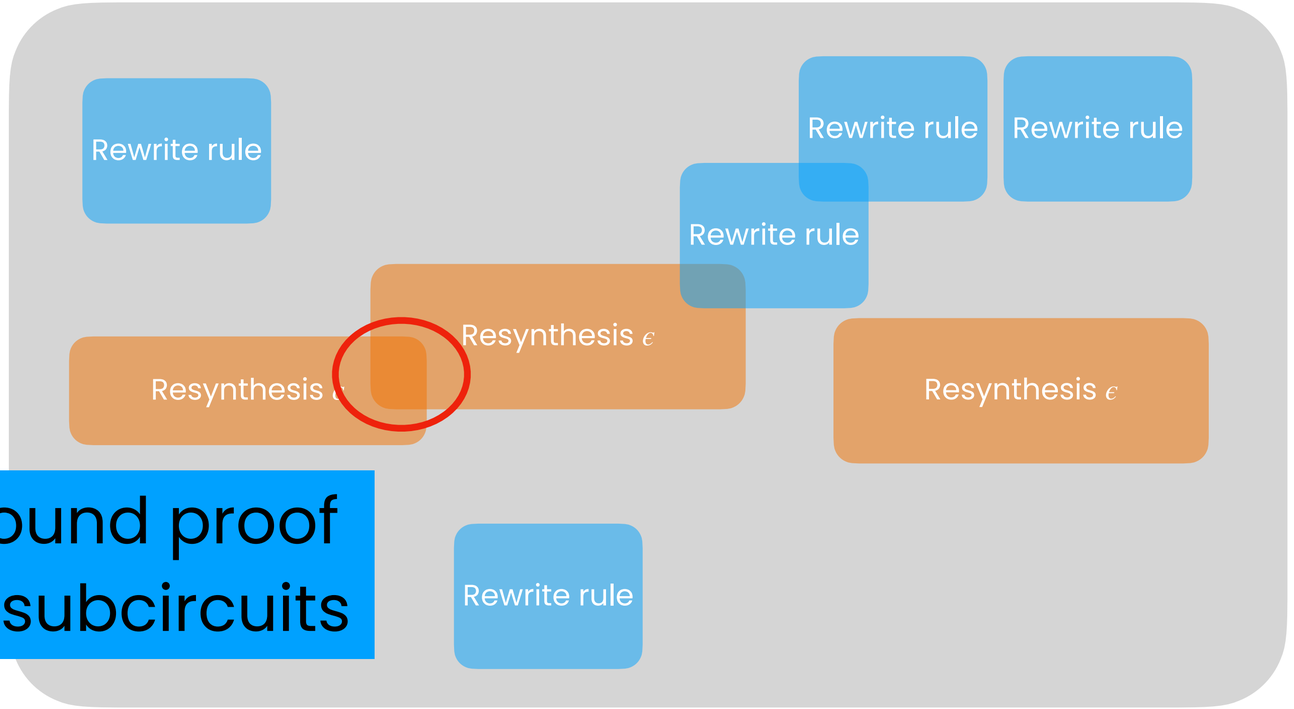
Figure 1. Summary of GUOQ compared to state-of-the-art on 2-qubit-gate reduction for the IBMQ20 gate set. GUOQ and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quarl requires an NVIDIA A100 (40GB) GPU to run.

ASPLOS talk on April 1st!

1. Randomly pick a transformation.
2. Randomly pick a subcircuit to transform.
3. Accept the result if better than current. Else, reject with high probability.



Prior work: error bound proof limited to **disjoint** subcircuits



$$\epsilon_{total} \leq \epsilon + \epsilon + \epsilon$$

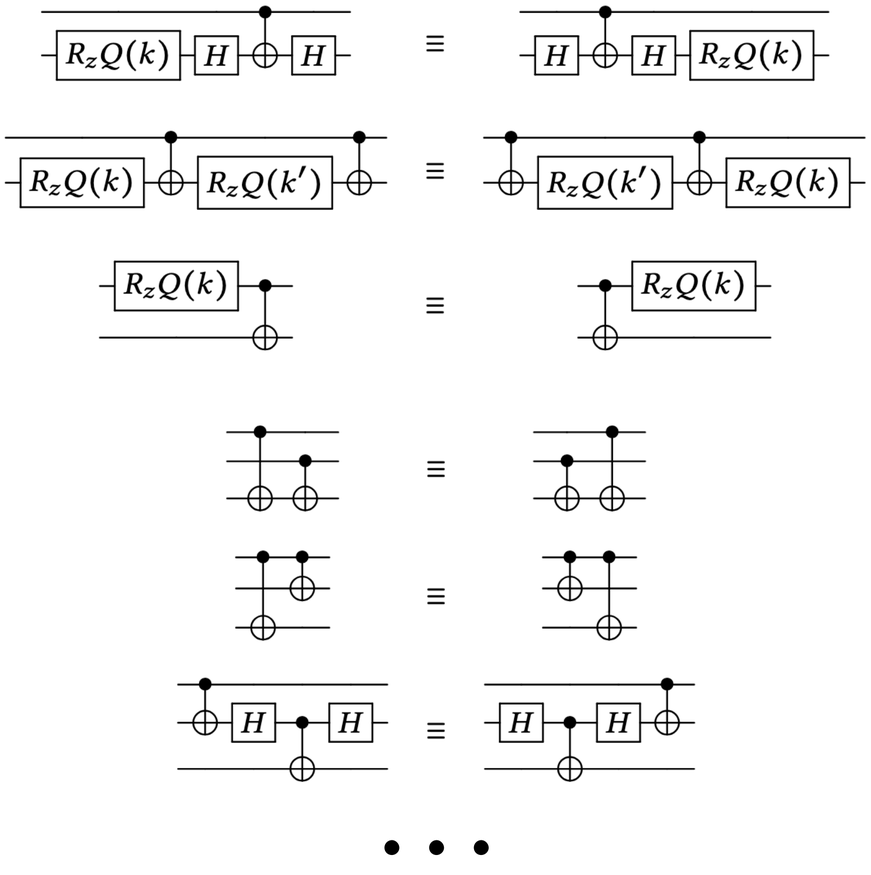
We extended to handle arbitrary subcircuits

GUOQ Motivation: Two Disparate Techniques

Rewrite Rules

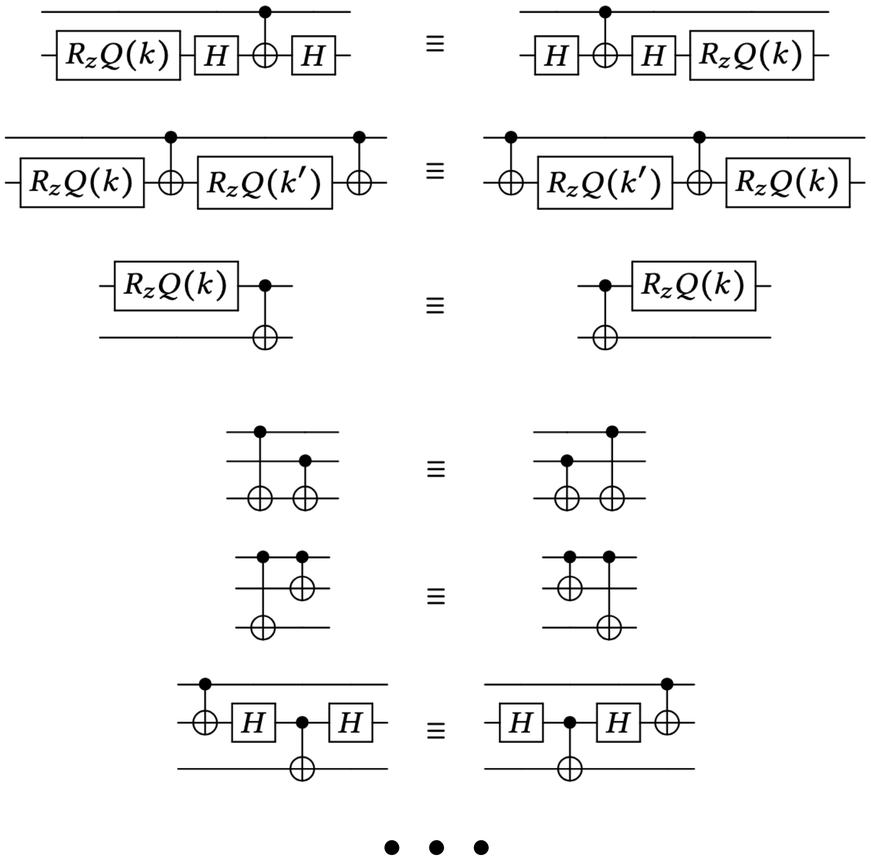
GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



GUOQ Motivation: Two Disparate Techniques

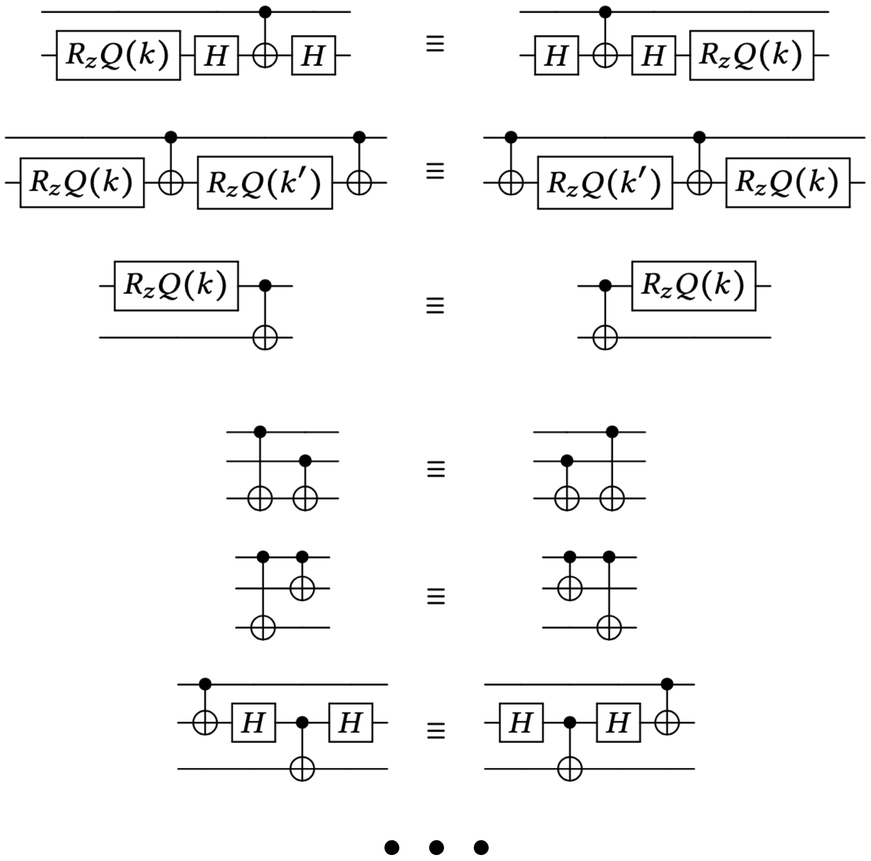
Rewrite Rules



Circuit Resynthesis

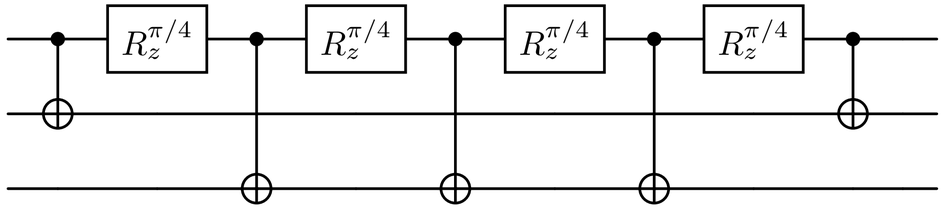
GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



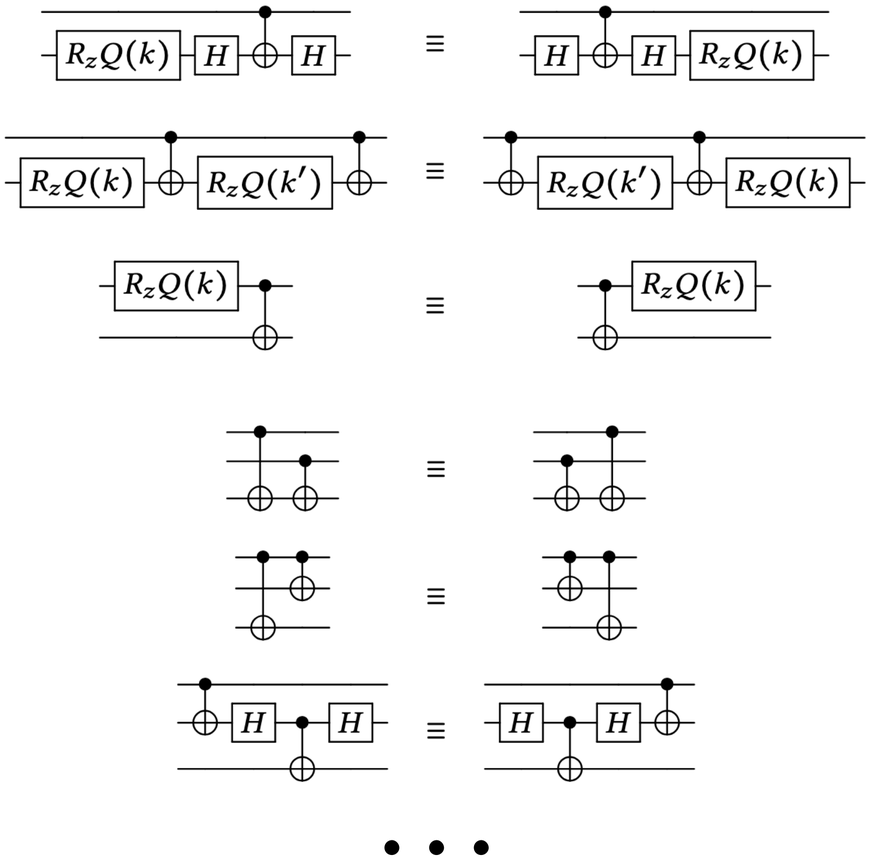
• • •

Circuit Resynthesis

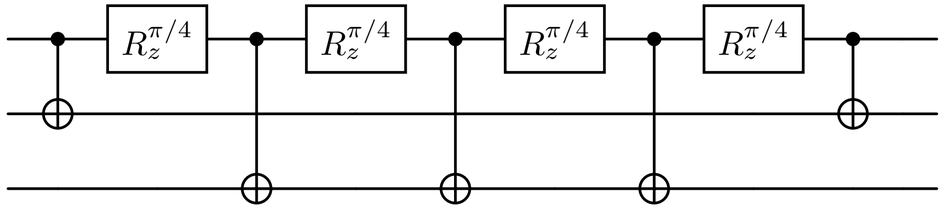


GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



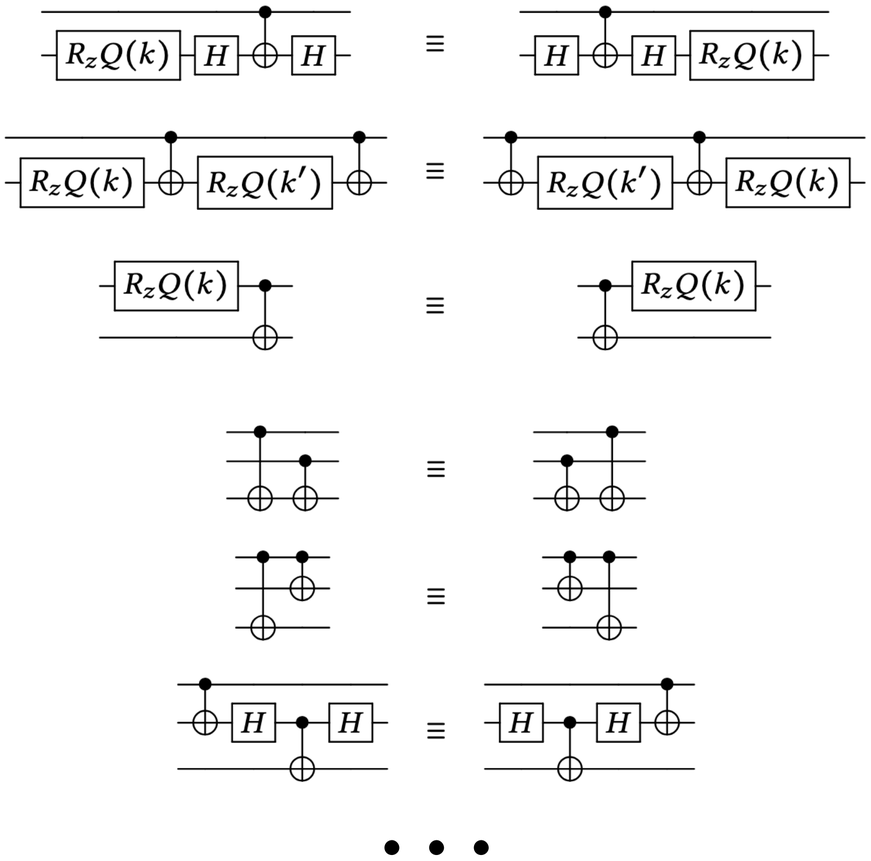
Circuit Resynthesis



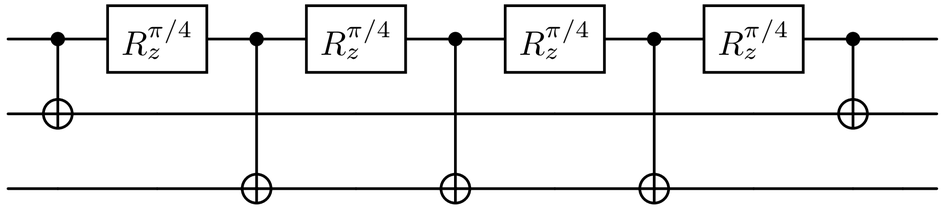
specification

GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



Circuit Resynthesis

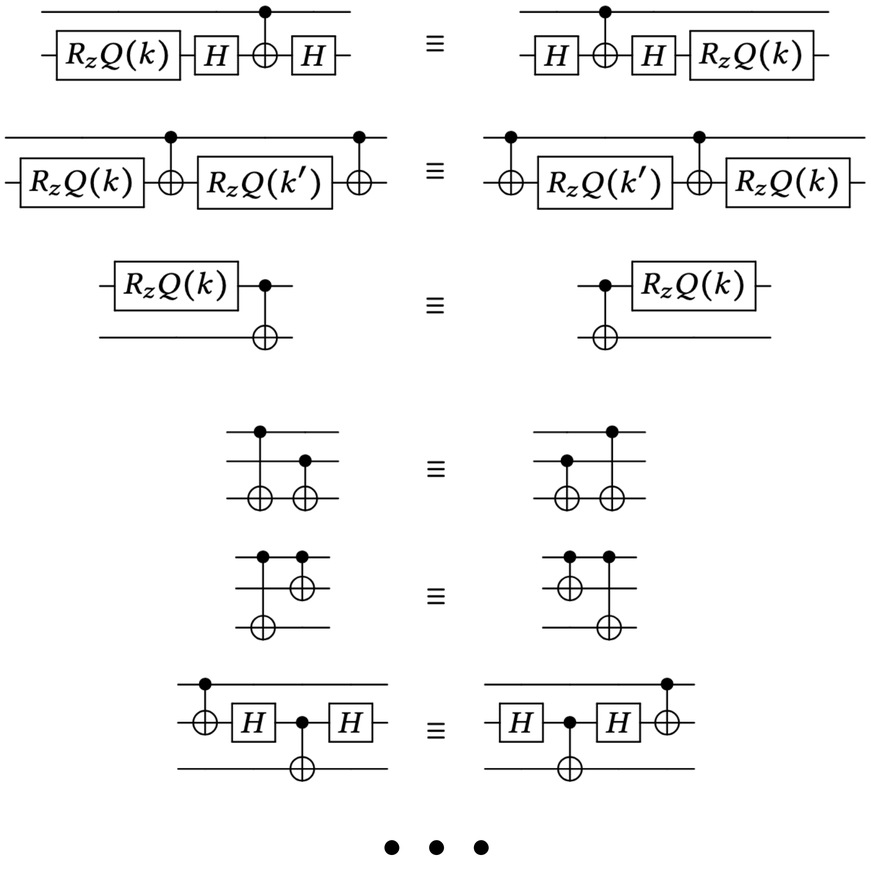


specification

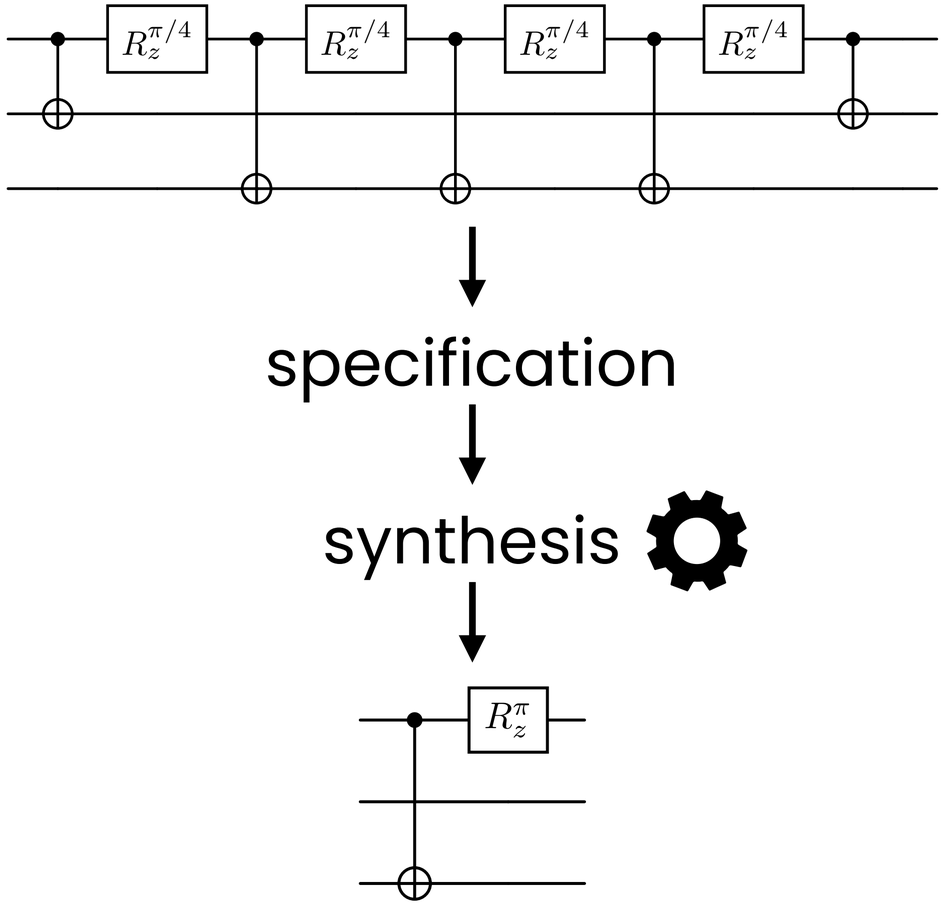
synthesis

GUOQ Motivation: Two Disparate Techniques

Rewrite Rules

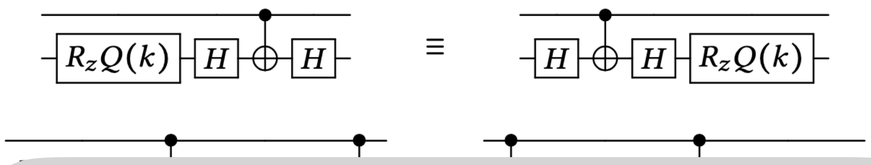


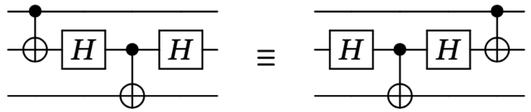
Circuit Resynthesis



GUOQ Motivation: Two Disparate Techniques

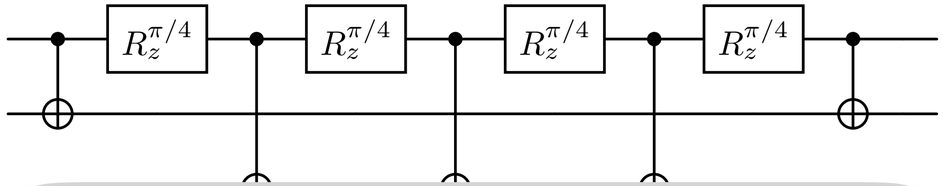
Rewrite Rules

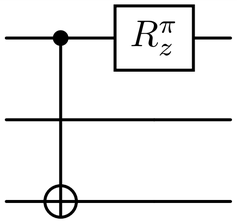




• • •

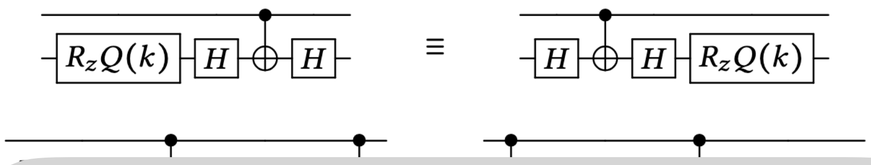
Circuit Resynthesis



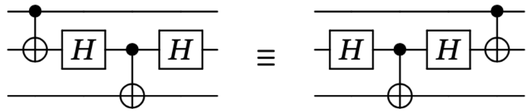


GUOQ Motivation: Two Disparate Techniques

Rewrite Rules

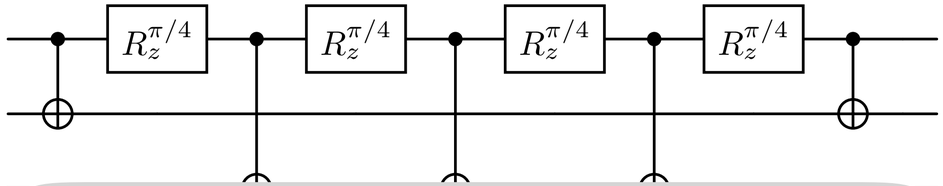


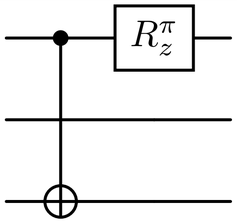
Fast



• • •

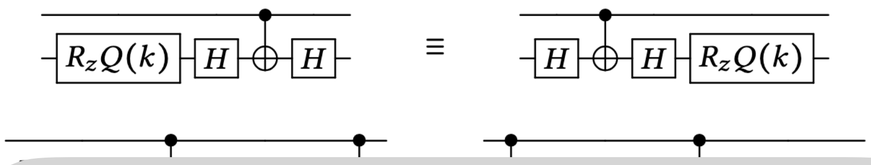
Circuit Resynthesis



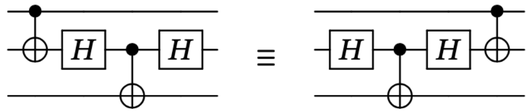


GUOQ Motivation: Two Disparate Techniques

Rewrite Rules

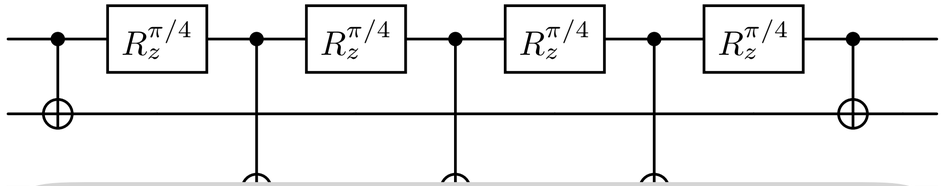


Fast

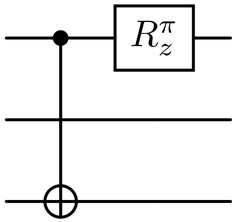


• • •

Circuit Resynthesis

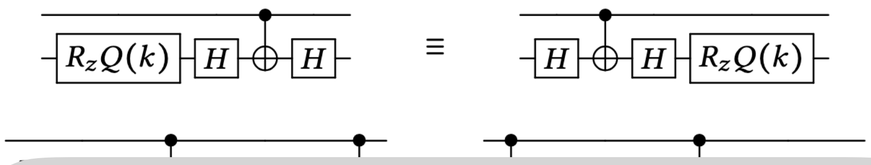


Slow



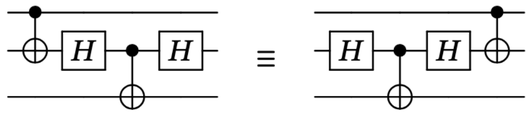
GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



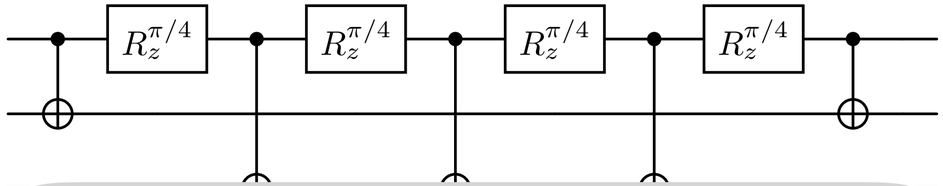
Fast

Limited by # gates

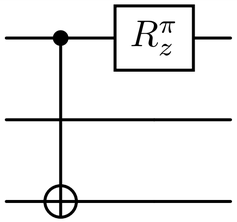


• • •

Circuit Resynthesis

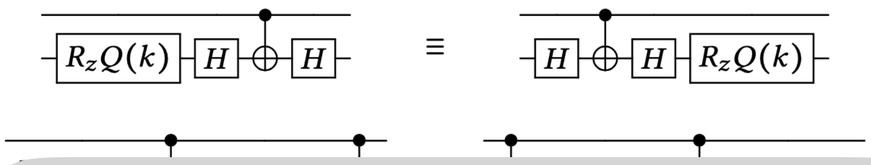


Slow



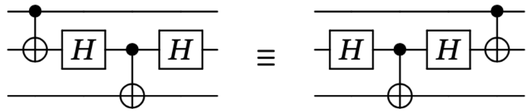
GUOQ Motivation: Two Disparate Techniques

Rewrite Rules



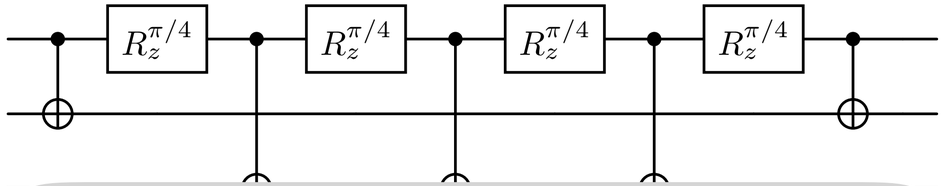
Fast

Limited by # gates



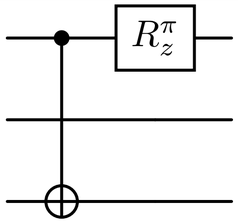
• • •

Circuit Resynthesis



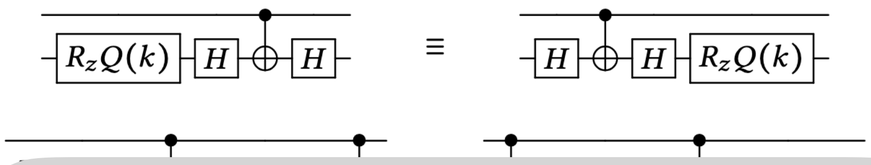
Slow

Limited by # qubits



GUOQ Motivation: Two Disparate Techniques

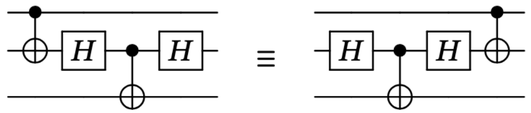
Rewrite Rules



Fast

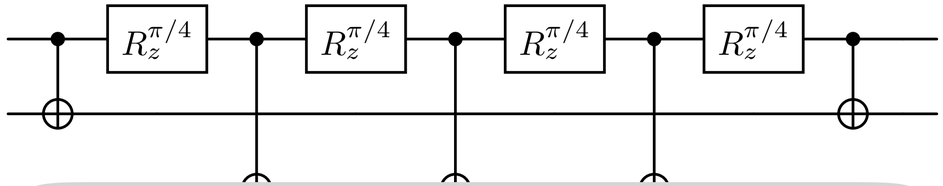
Limited by # gates

Exact



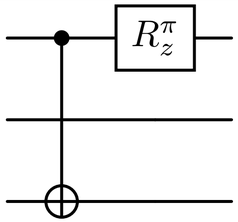
• • •

Circuit Resynthesis



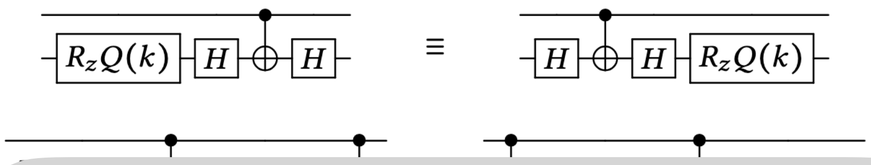
Slow

Limited by # qubits



GUOQ Motivation: Two Disparate Techniques

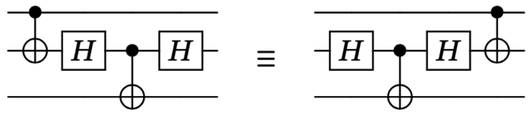
Rewrite Rules



Fast

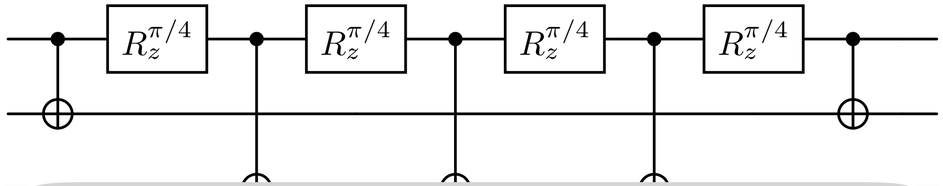
Limited by # gates

Exact



• • •

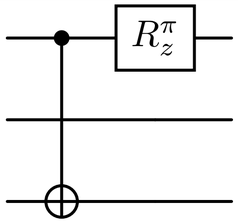
Circuit Resynthesis



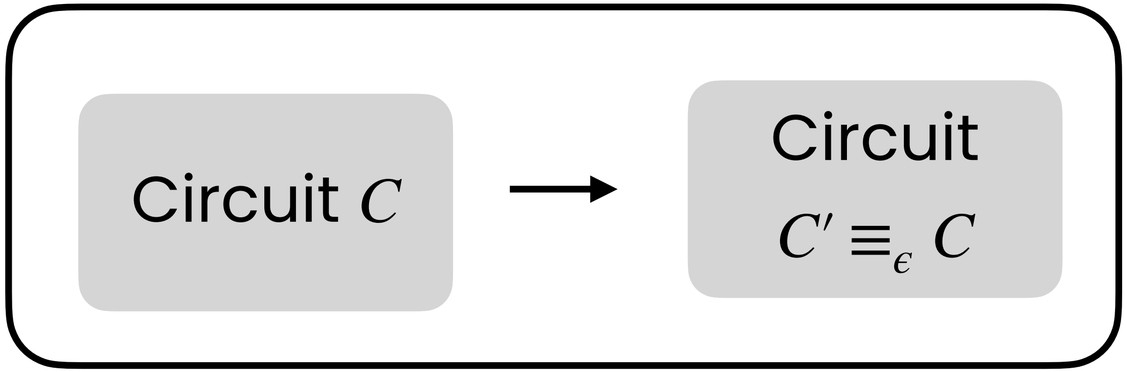
Slow

Limited by # qubits

Approximate

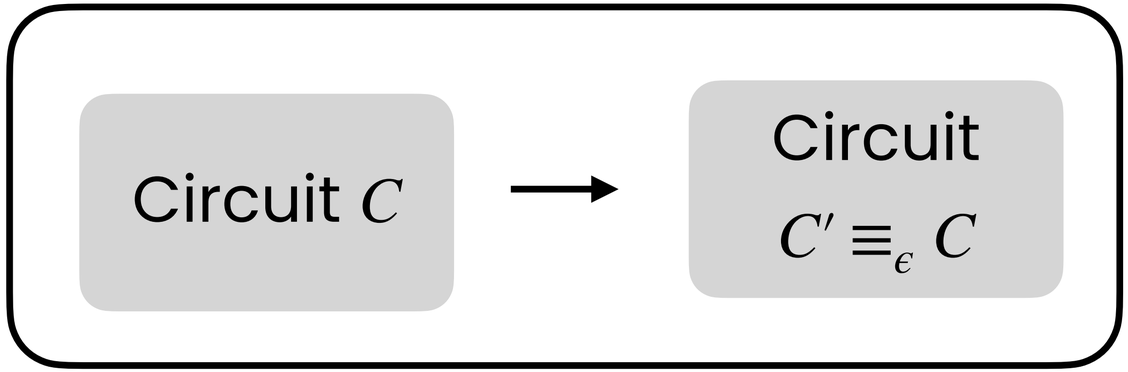


Unifying Abstract Circuit Transformations

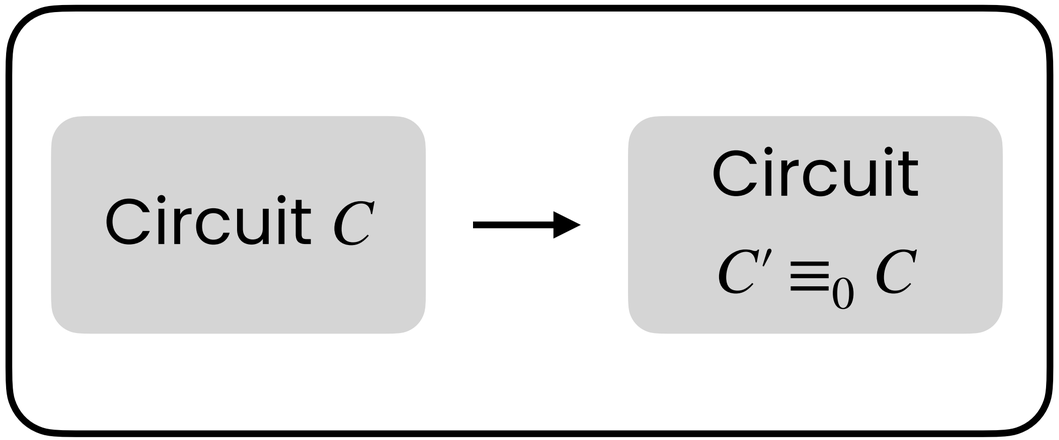


transformation $\tau_\epsilon : \mathscr{C} \to \mathscr{C}$

Unifying Abstract Circuit Transformations

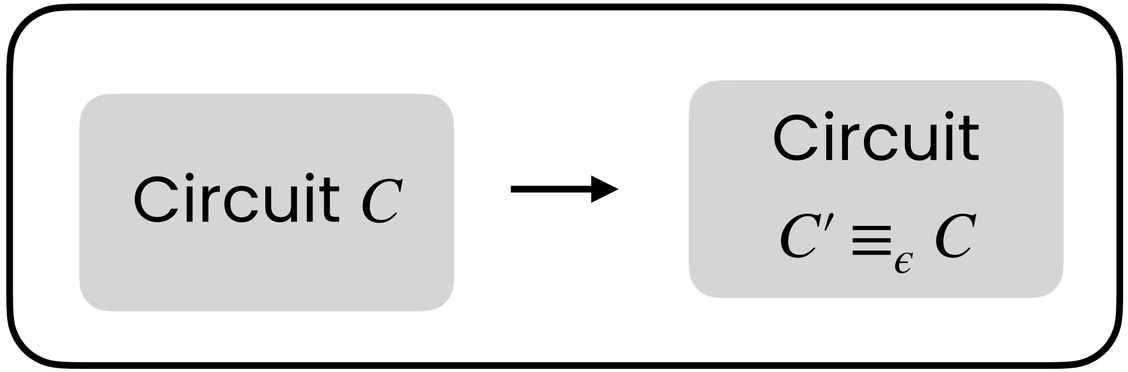


transformation $\tau_\epsilon : \mathscr{C} \to \mathscr{C}$

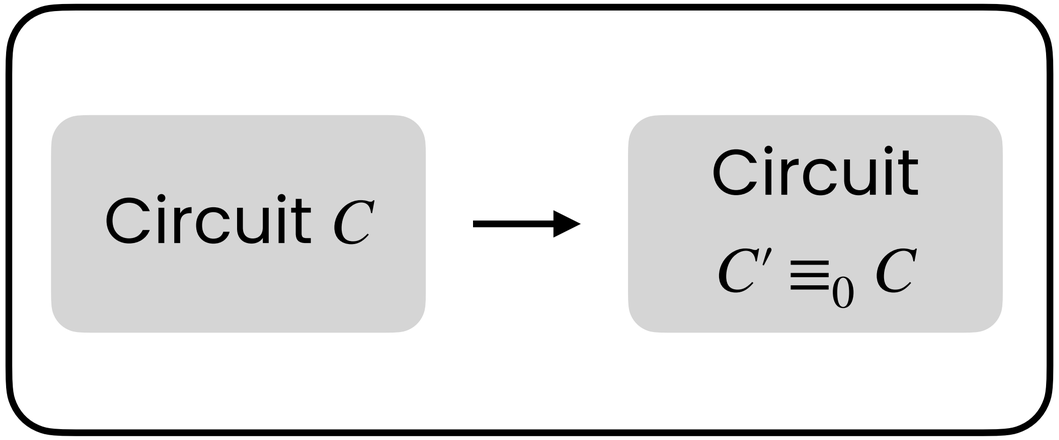


rewrite rule

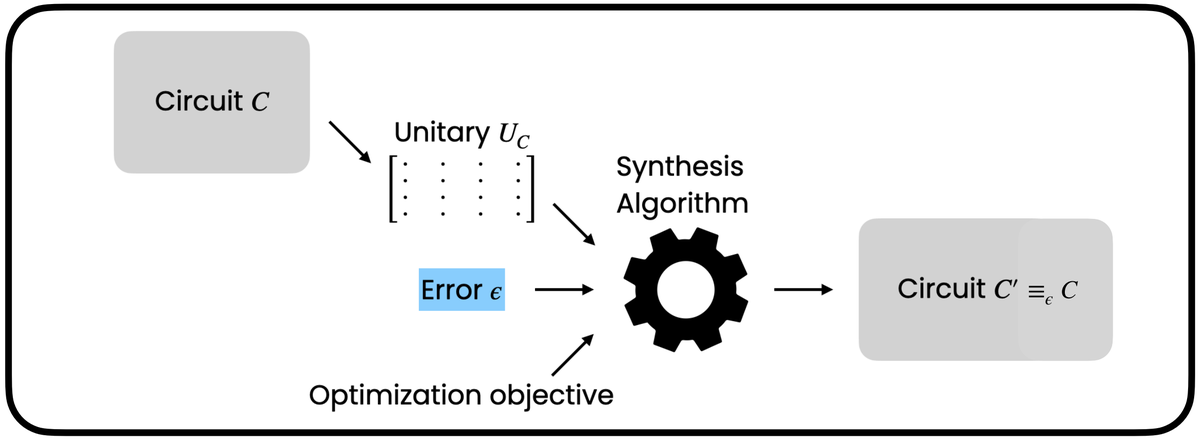
Unifying Abstract Circuit Transformations



transformation $\tau_\epsilon : \mathscr{C} \to \mathscr{C}$

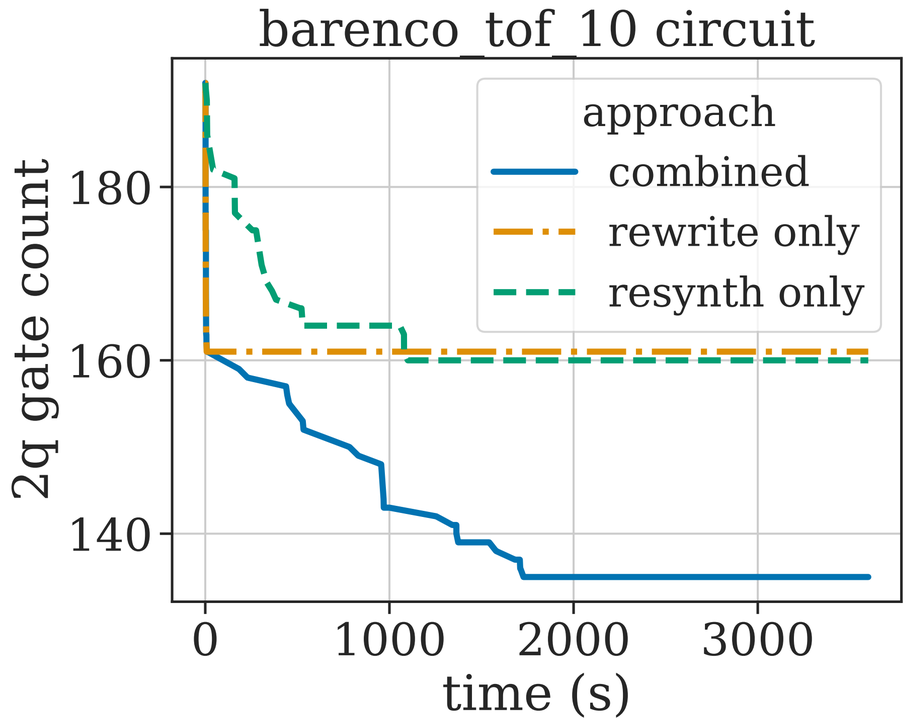


rewrite rule

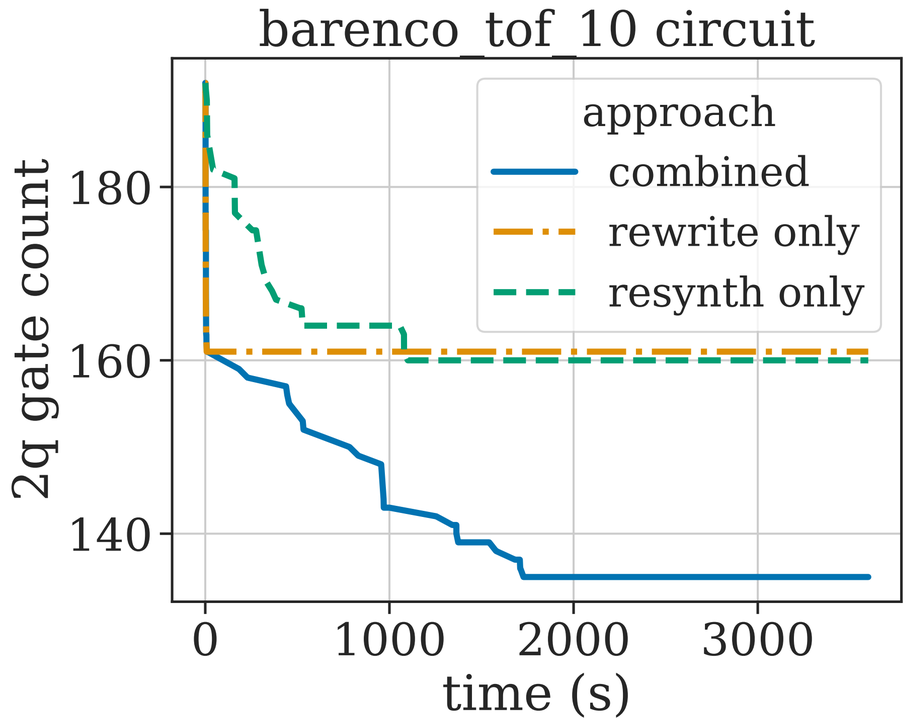


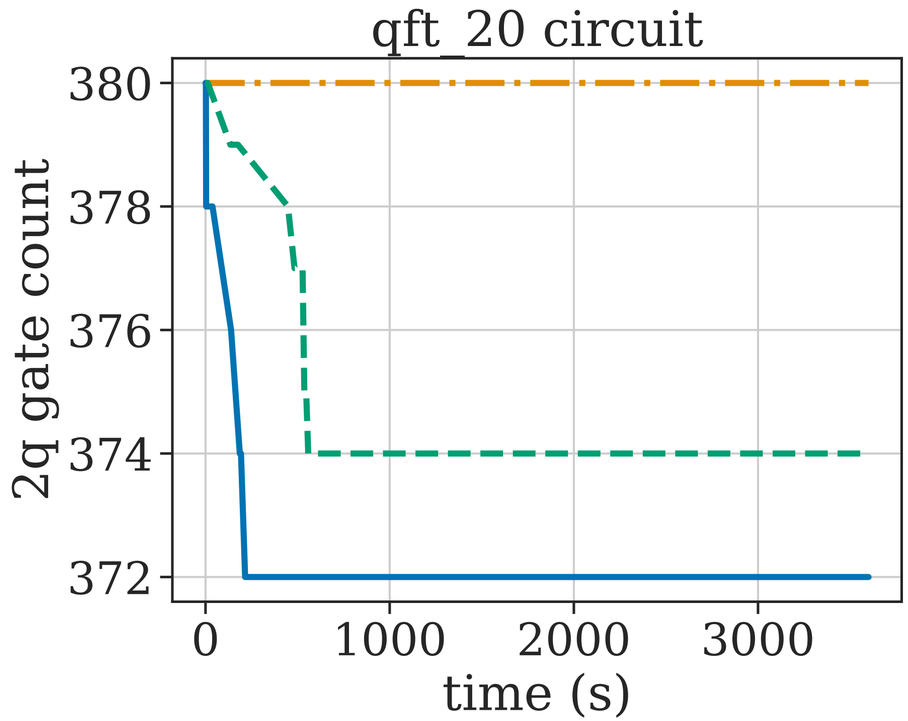
resynthesis

GUOQ Search Exploits Synergy

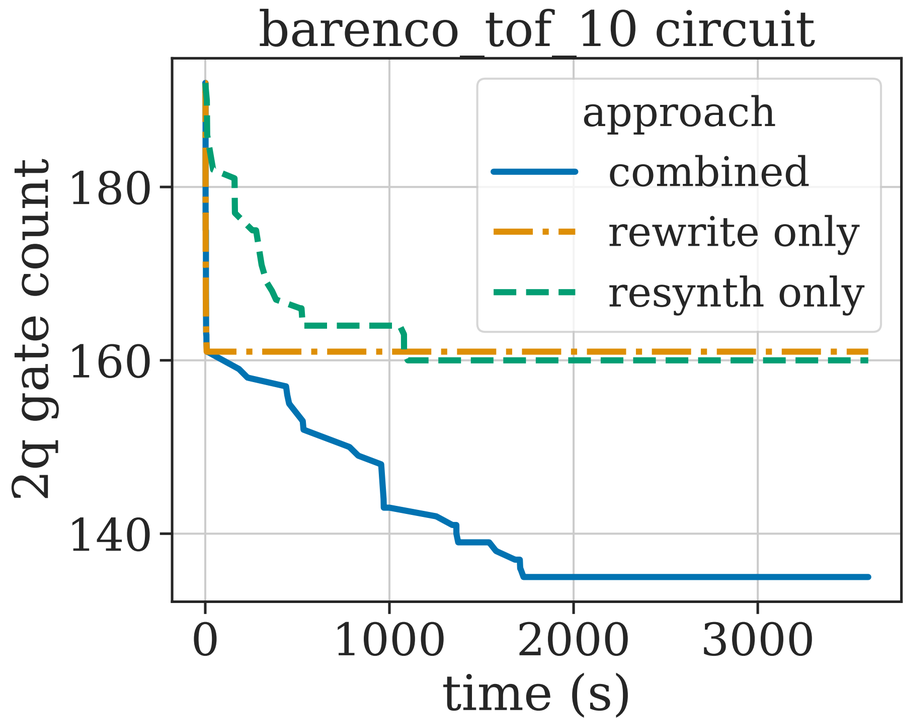


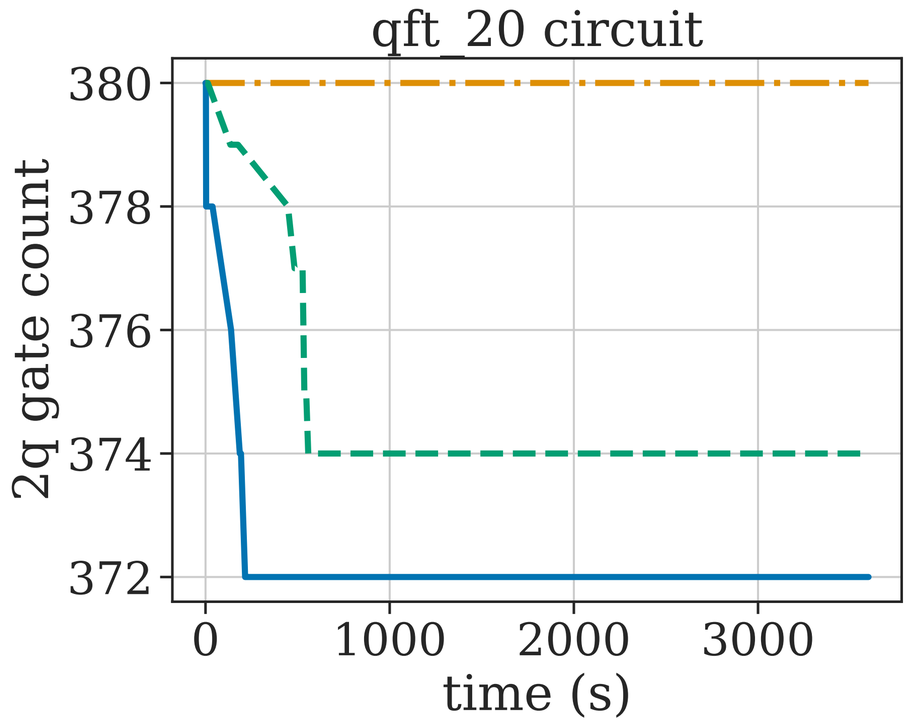
GUOQ Search Exploits Synergy





GUOQ Search Exploits Synergy





Also significantly outperforms state-of-the-art!

Come to the ASPLOS talk on April 1st!

Optimizing for FTQC

Optimizing for FTQC

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

Optimizing for FTQC

discrete

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

Optimizing for FTQC

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

discrete

"magic state distillation"

Optimizing for FTQC

discrete

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

"magic state distillation"

Eastin-Knill theorem: no transversal universal set

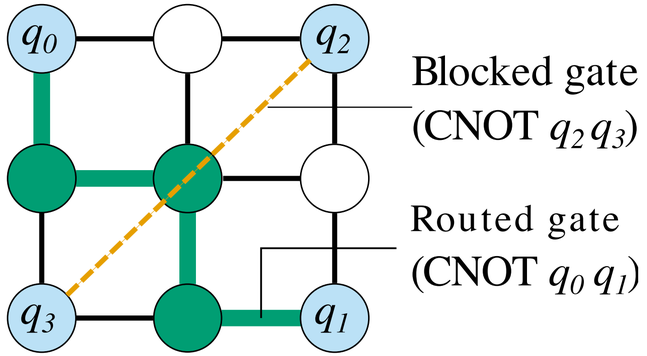
Optimizing for FTQC

discrete

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

"magic state distillation"

Eastin-Knill theorem: no transversal universal set



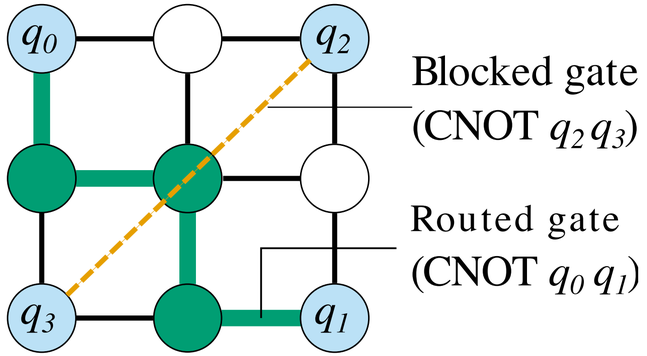
Optimizing for FTQC

discrete

Clifford + T gate set: $\{T, T^\dagger, S, S^\dagger, H, CX\}$

"magic state distillation"

Eastin-Knill theorem: no transversal universal set



😟

BQSKit
GUOQ
VOQC
TKET
Qiskit
Quarl
Quartz
QUESO

Optimizing for FTQC

Open Research Question

Optimizing for FTQC

Open Research Question

Reducing T-count with the ZX-calculus

Aleks Kissinger and John van de Wetering

Radboud University Nijmegen
January 20, 2020

Linear and Non-linear Relational Analyses for Quantum Program Optimization

MATTHEW AMY, Simon Fraser University, Canada
JOSEPH LUNDERVILLE, Simon Fraser University, Canada