

Synthesizing quantum compilers

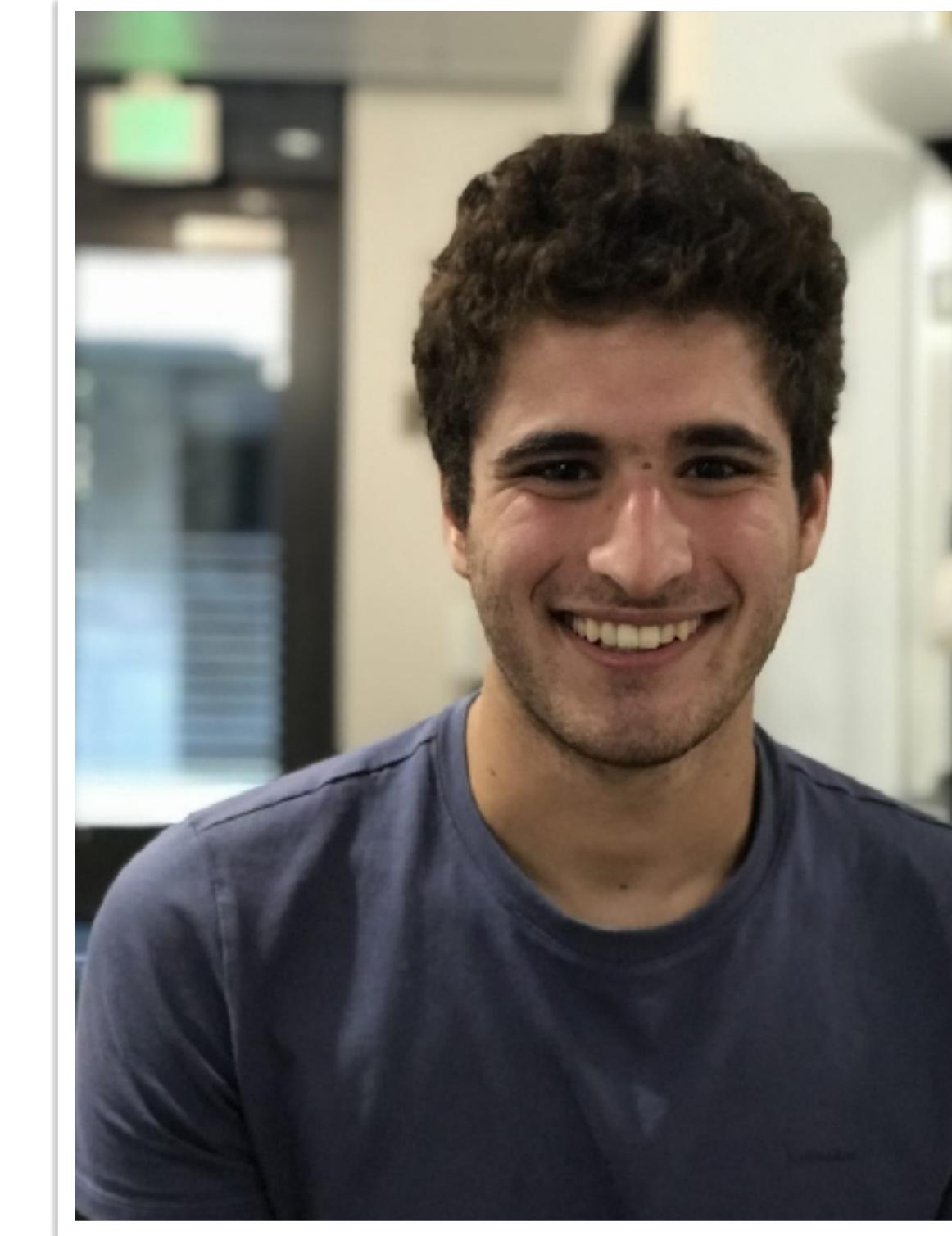
Aws Albarghouthi | University of Wisconsin–Madison



The protagonists



Amanda Xu



Abtin Molavi

Why quantum computing?

Why quantum computing?

factoring integers efficiently using Shor's algorithm

Why quantum computing?

factoring integers efficiently using Shor's algorithm

simulating quantum mechanics, e.g., for material discovery

Why quantum computing?

factoring integers efficiently using [Shor's algorithm](#)

simulating quantum mechanics, e.g., for [material discovery](#)

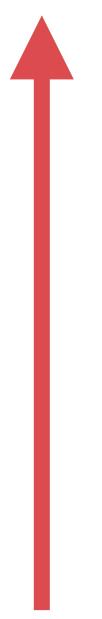
“discover that quantum mechanics was wrong” – Michael Nielsen*

* <https://conversationswithtyler.com/episodes/michael-nielsen/>

Bits vs qubits

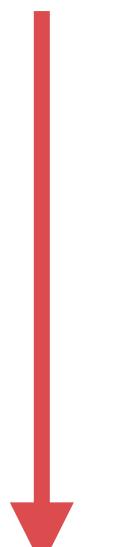
0

1



Bits vs qubits

0



1

Bits vs qubits

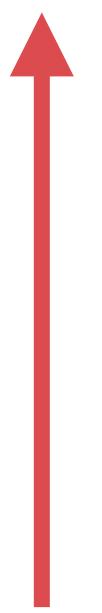
$| 0 \rangle$



$| 1 \rangle$

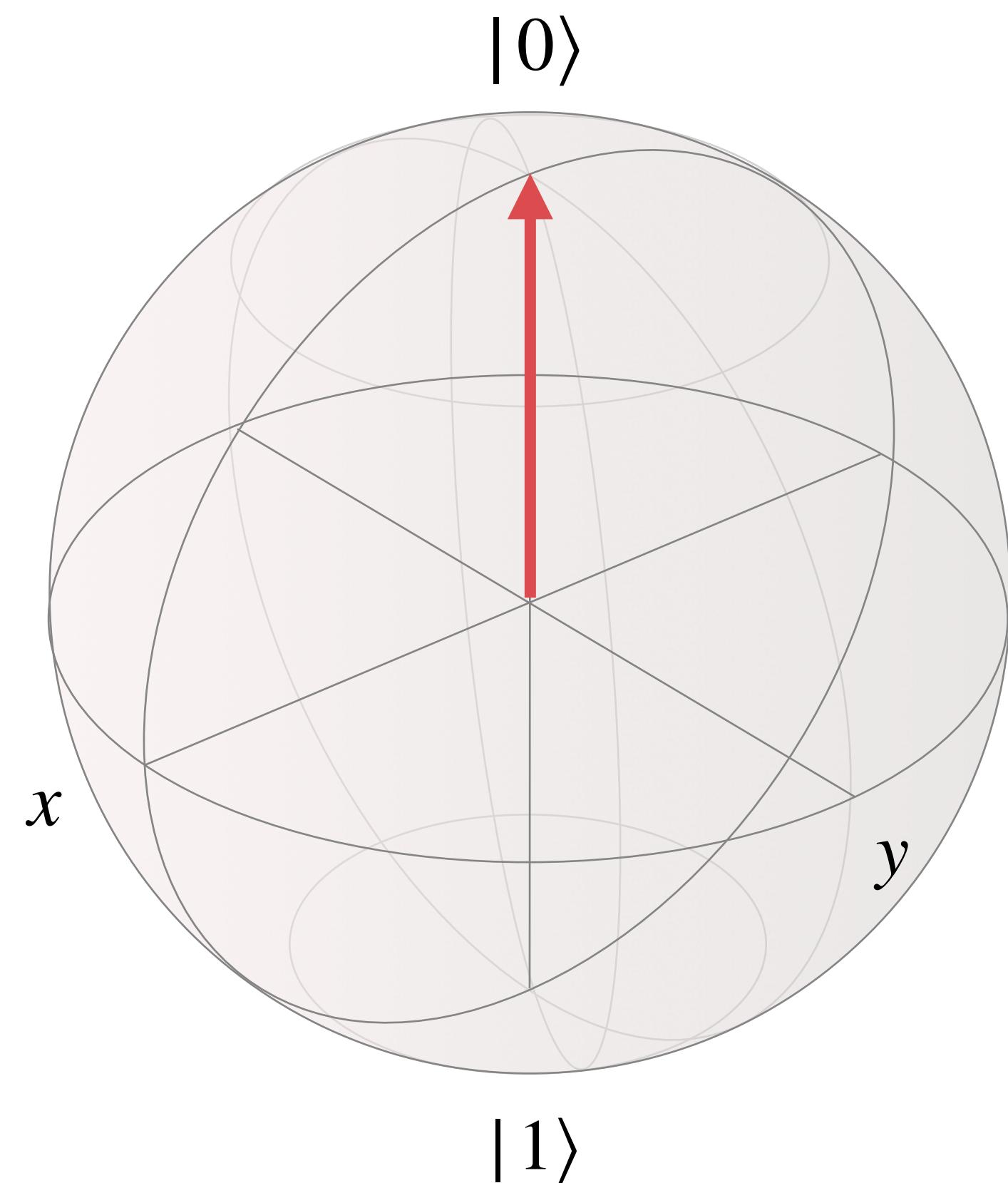
Bits vs qubits

$| 0 \rangle$

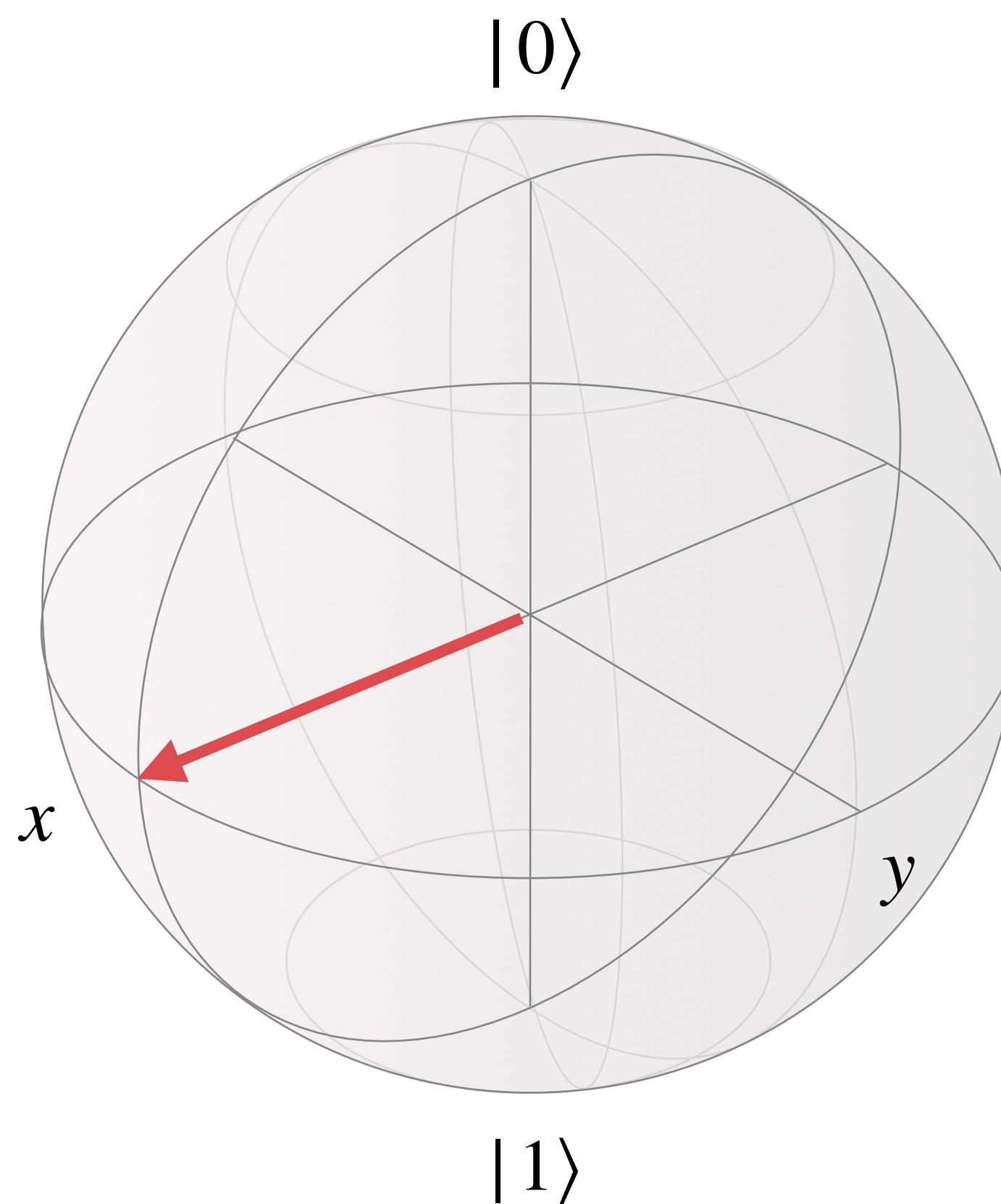


$| 1 \rangle$

Bits vs qubits



Bits vs qubits



Binary vs quantum operations

Classical X gate

$$0 \rightarrow 1$$

$$1 \rightarrow 0$$

Binary vs quantum operations

Classical X gate

$$0 \rightarrow 1$$

$$1 \rightarrow 0$$

Quantum X gate

$$|0\rangle \rightarrow |1\rangle$$

$$|1\rangle \rightarrow |0\rangle$$

Binary vs quantum operations

Classical X gate

$$0 \rightarrow 1$$

$$1 \rightarrow 0$$

Quantum X gate

$$|0\rangle \rightarrow |1\rangle$$

$$|1\rangle \rightarrow |0\rangle$$

Pathsum notation*

$$X : |x\rangle \rightarrow |\neg x\rangle$$

Binary vs quantum operations

Binary vs quantum operations

Pathsum notation

$$H : |x\rangle \rightarrow \sum_y \frac{1}{\sqrt{2}} e^{ixy} |y\rangle$$

Binary vs quantum operations

Pathsum notation

$$H : |x\rangle \rightarrow \sum_y \frac{1}{\sqrt{2}} e^{ixy} |y\rangle$$

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

Binary vs quantum operations

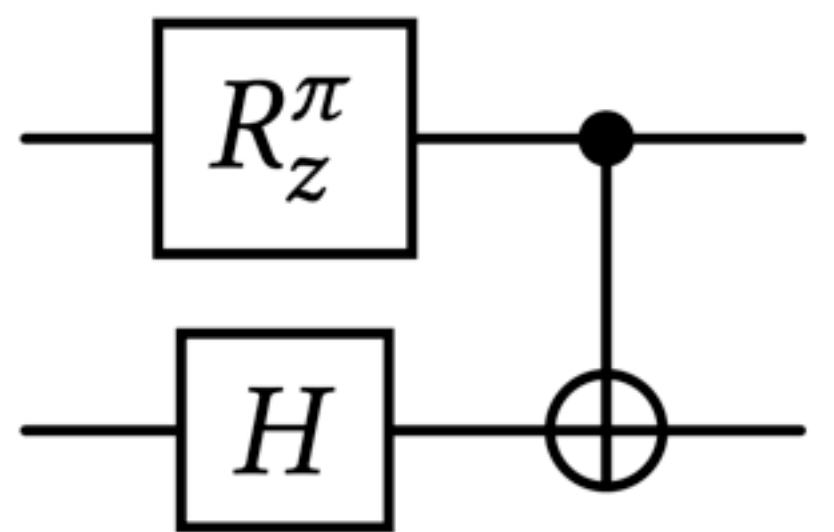
Pathsum notation

$$H : |x\rangle \rightarrow \sum_y \frac{1}{\sqrt{2}} e^{ixy} |y\rangle$$

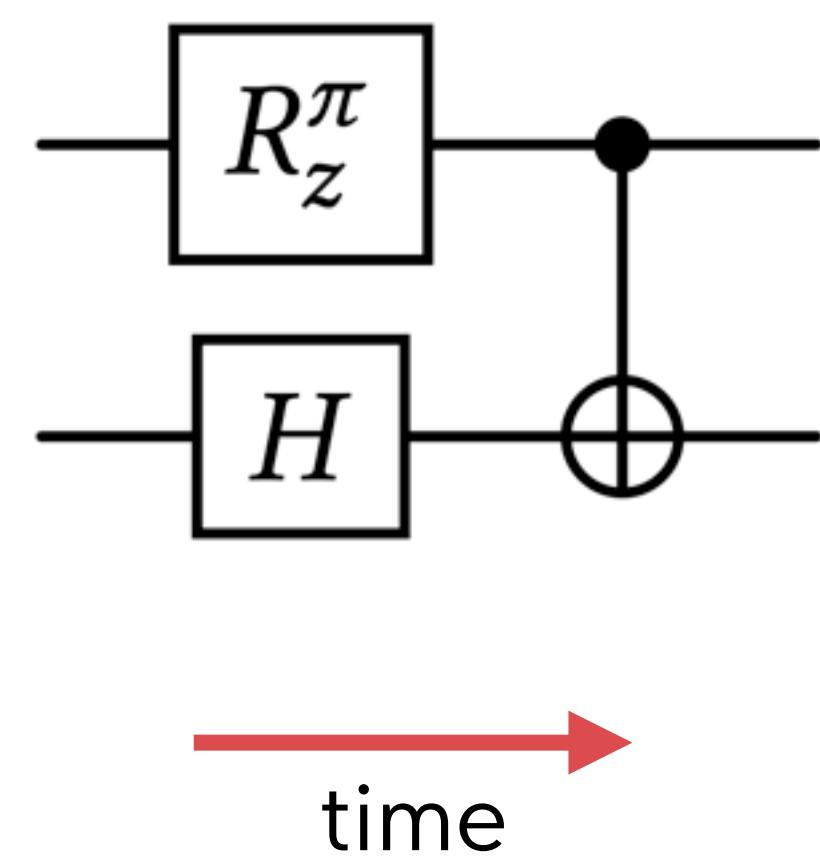
$$R_z(\theta) : |x\rangle \rightarrow e^{i(2x-1)\theta} |x\rangle$$

$$CX : |xy\rangle \rightarrow |x(x \oplus y)\rangle$$

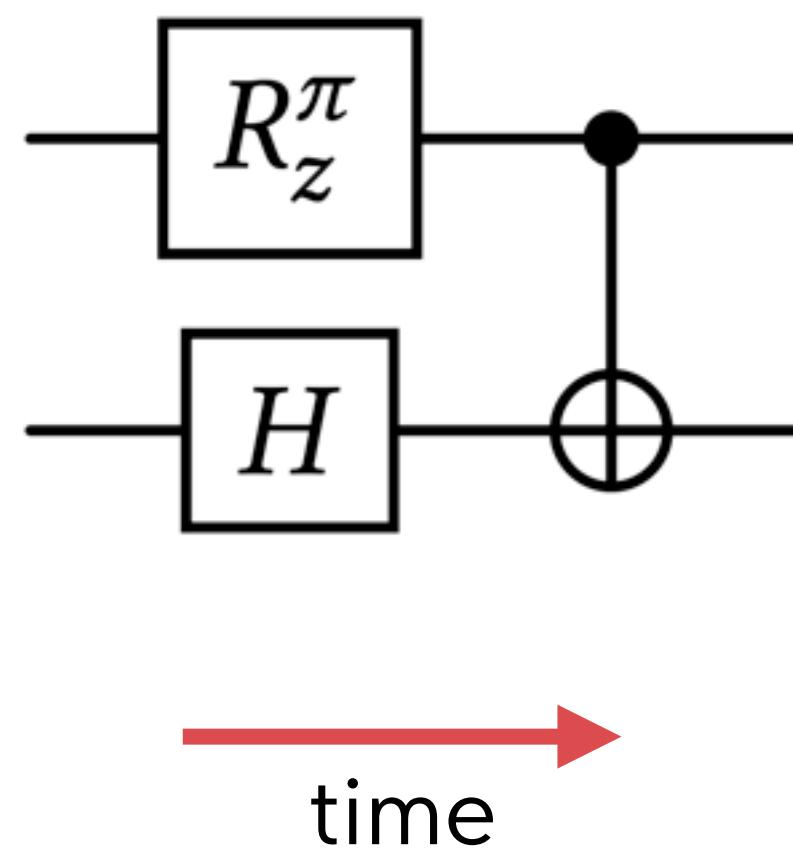
Quantum circuits/programs



Quantum circuits/programs



Quantum circuits/programs



Rz(π) q1;
H q2;
CX q1, q2;

The quantum landscape

The quantum landscape

qubits are unreliable, noisy

The quantum landscape

qubits are unreliable, noisy

NISQ what can we do with noisy qubits?

The quantum landscape

qubits are unreliable, noisy

NISQ what can we do with noisy qubits?

FTQC can we do error correction?

The quantum landscape

qubits are unreliable, noisy

NISQ what can we do with noisy qubits?

FTQC can we do error correction?

Article

Quantum error correction below the surface code threshold

<https://doi.org/10.1038/s41586-024-08489-y> Google Quantum AI and Collaborators*

Received: 24 August 2024
Accepted: 26 November 2024
Published online: 01 December 2024
Open access
Check for updates

Quantum error correction¹ provides a path to reach practical quantum computing by combining multiple physical qubits into a logical qubit, in which the logical error rate is suppressed exponentially as more qubits are added. However, this exponential suppression only occurs if the physical error rate is below a critical threshold. Here we present two below-threshold surface code memories on our newest generation of superconducting processors, Willow: a distance-7 code and a distance-5 code integrated with a real-time decoder. The logical error rate of our larger quantum memory is suppressed by a factor of $A = 2.4 \pm 0.02$ when increasing the code distance by 2, culminating in a 101-qubit distance-7 code with $0.013\% \pm 0.003\%$ per error per cycle of error correction. This logical memory is also beyond break-even, exceeding the lifetime of its best physical qubit by a factor of 2.4 ± 0.3 . Our system maintains below-threshold performance when decoding in real time, achieving an average decoder latency of 63 microseconds at distance 5 up to a million cycles, with a cycle time of 1.1 microseconds. We also run repetition codes up to distance 29 and find that the logical performance is limited by rare correlated error events, occurring approximately once every hour or 3×10^6 cycles. Our results indicate device performance that, if scaled, could realize the operational requirements of large-scale fault-tolerant quantum algorithms.

Quantum computing promises computation based upon quantum chemistry², quantum simulation³, cryptography⁴ and optimization⁵. However, quantum information is fragile and quantum operations are error prone. State-of-the-art many-qubit platforms have only recently demonstrated entangling gates with $\sim 99.9\%$ fidelity^{6,7}, far short of the $\sim 10^{-6}$ error rates needed for many applications^{1,2,3}. Quantum error correction is postulated to realize high-fidelity logical qubits by distributing quantum information from many entangled physical qubits to protect against errors. If the physical operations are below a critical noise threshold, the logical error rate is suppressed exponentially as we increase the number of physical qubits per logical qubit. This behavior is expressed in the approximate relation

$$\epsilon_L \approx \left(\frac{p}{p_{th}} \right)^{D/4} \quad (1)$$

for error-corrected surface code logical qubits^{1,8,9}. Here D is the code distance indicating $2^D - 1$ physical qubits used per logical qubit; p and ϵ_L are the physical and logical error rates, respectively; and p_{th} is the threshold error rate of the code. Thus, when $p < p_{th}$, the error rate of the logical qubit is suppressed exponentially in the distance of the code, with the error suppression factor $= \epsilon_L/\epsilon_{L0} = p_{th}/p$ representing the reduction in logical error rate when increasing the code distance by two. Although many platforms have demonstrated different features of quantum error correction^{1,2,3}, no quantum processor has definitively shown below-threshold performance.

*A list of authors and their affiliations appears at the end of the paper.

Google Quantum AI et al., Nature 2024

The quantum landscape

qubits are unreliable, noisy

NISQ what can we do with noisy qubits?

FTQC can we do error correction?

Article Quantum error correction below the surface code threshold

<https://doi.org/10.1101/41846-024-08448-y> Google Quantum AI and Collaborators*

Received: 24 August 2024

Accepted: 26 November 2024

Published online: 01 December 2024

Download

Check for updates

Quantum error correction¹ provides a path to reach practical quantum computing by combining multiple physical qubits into a logical qubit, in which the logical error rate is suppressed exponentially as more qubits are added. However, this exponential suppression only occurs if the physical error rate is below a critical threshold. Here we present two below-threshold surface code memories on our newest generation of superconducting processors, Willow: a distance-7 code and a distance-5 code integrated with area-time decoders. The logical error rate of our larger quantum memory is suppressed by a factor of 4 ± 0.02 when increasing the logical code distance by 2, culminating in a 101-qubit distance-7 code with $0.013\% \pm 0.003\%$ per-center error per cycle of error correction. This logical memory is also beyond breakeven, exceeding the lifetime of its best physical qubit by a factor of 2.4 ± 0.3 . Our system maintains below-threshold performance when decoding in real time, achieving an average decode latency of 63 microseconds at distance 5 up to a million cycles, with a cycle time of 1.1 microseconds. We also run repetition codes up to distance 29 and find that the logical performance is limited by rare correlated error events, occurring approximately once every hour or 3×10^6 cycles. Our results indicate device performance that, if scaled, could realize the operational requirements of large-scale fault-tolerant quantum algorithms.

Quantum computing promises computational advantages in quantum chemistry², quantum simulation³, cryptography⁴ and optimization⁵. However, quantum information is fragile and quantum operations are error prone. State-of-the-art many-qubit platforms have only recently demonstrated entangling gates with 99.9% fidelity^{6,7}, far short of the $<10^{-6}$ error rates needed for many applications^{1,8}. Quantum error correction is necessary to realize high-fidelity logical qubits by distributing quantum information from many entangled physical qubits produced by the quantum device, thus maintaining the generated⁹. The first open challenges of superconducting qubits, ranging from tens to hundreds of nanoseconds, provide an advantage in speed but also a challenge for decoding errors both quickly and accurately.

In this work, we demonstrate error correction codes operating below the threshold physical error rate is imposed by the physical hardware. Using a 72-qubit processor, we implement a distance-5 surface code operating with an integrated time decoder. Subsequently, using a 105-qubit processor, we test the performance of a distance-7 surface code. These processes demonstrate $\lambda/2$ to distance 5 and 7, respectively. Our distance-5 and distance-7 quantum memories are beyond breakeven, with distance 7 preserving quantum information for more than twice as long as its best constituent physical qubit. To identify possible logical error floors, we also implement a high-distance repetition code on the 72-qubit processor, with error rates that are dominated by correlated error events occurring once a hour. These errors, the origins of which are not yet understood, set a current error floor of 10^{-10} in the repetition code. Finally, we show that we can maintain below-threshold operation on the 72-qubit processor even when decoding in real time, meeting the strict timing requirements imposed by the fast 1.1-ns cycle duration of the processor.

*A list of authors and their affiliations appears at the end of the paper.

Google Quantum AI et al., Nature 2024

Article Hardware-efficient quantum error correction via concatenated bosonic qubits

<https://doi.org/10.1101/41846-024-084427>

Received: 1 October 2024

Accepted: 13 January 2025

Published online: 26 February 2025

Open access

Check for updates

To solve problems of practical importance^{1,2}, quantum computers probably need to incorporate quantum error correction, in which a logical qubit is redundantly encoded in many noisy physical qubits^{3,4}. The large physical-qubit overhead associated with error correction motivates the search for more hardware-efficient approaches^{5–8}. Here, using a superconducting quantum circuit⁹, we realize a logical qubit memory formed from the concatenation of encoded bosonic cat qubits with an outer repetition code of distance $d = 5$ (ref. 10). A stabilizing circuit passively protects cat qubits against bit-flips^{10–14}. The repetition code, using ancilla transmons for syndrome measurement, corrects cat qubit phase flips. We study the performance and scaling of the logical qubit memory, finding that the phase-flip correcting repetition code operates below the threshold. The logical bit-flip error is suppressed with increasing cat qubit mean photon number, enabled by our realization of a cat-transmon-based CNOT gate. The minimum measured logical error per cycle is on average 1.5% for the distance-5 code sections, and 1.6% for the distance-5 code. Despite the increased number of off-axis locations of the distance-5 code, the high degree of noise bias preserved during error correction enables comparable performance. These results, where the intrinsic error suppression of the bosonic encodings enables us to achieve hardware-efficient outer error-correcting code, indicate that concatenated bosonic codes can be a compelling model for reaching fault-tolerant quantum computation.

For quantum computers to solve problems in material design, quantum chemistry and cryptography, in which known speed-ups relative to classical computations are attainable, currently proposed algorithms require trillions of qubit gate operations to be applied in an error-free manner¹⁵. Despite impressive progress over the past few decades in reducing qubit error rates at the physical hardware level, the state-of-the-art remains about three orders of magnitude away from these requirements. A path towards closing the error-rate gap is through concatenated fault-tolerant quantum error correction (QEC)¹⁶, which can exponentially suppress errors through the redundant encoding of information across many noisy physical qubits.

Recently, QEC experiments have been performed in various hardware platforms, including superconducting quantum circuits^{17–19}, trapped ions²⁰ and neutral atoms²¹. Some of these experiments are approaching²², or have surpassed²³, the threshold at which scaling of the error-correcting codes leads to exponential improvements in the logical qubit error rate. In these experiments, the qubits are rarely suppressed at the physical level, and the error rates are often suppressed in the error-correcting code. The use of a repetition code enables low overhead because of its large error rate threshold and linear scaling of code distance with physical qubit number^{24,25}. In what follows, we describe a microfabricated superconducting quantum circuit that realizes a distance-5 repetition code, called a logical qubit memory, presents a noise-biased CNOT gate for implementing error syndrome measurements with ancilla transmons and study the logical qubit error correction performance.

Quantum device realizing a distance-5 repetition code
A schematic of our repetition code device and the corresponding superconducting circuit layout shown in Fig. 1. The distance-5 repetition code consists of five bosonic modes that host the data qubits (blue), along with four ancilla qubits (orange). The bosonic modes

A list of authors and their affiliations appears at the end of the paper.

AWS et al., Nature 2025

Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Ion Trap



Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Superconducting



Ion Trap



Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Superconducting



Ion Trap



Photonic



Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Superconducting



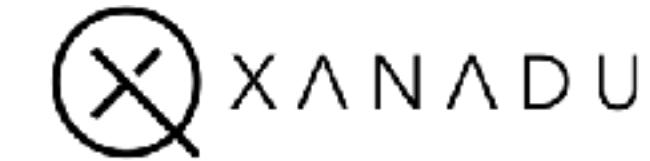
Ion Trap



Neutral Atom



Photonic



Anatomy of a quantum compiler

:

circuit optimizer

mapper and router

quantum processor

Anatomy of a quantum compiler

•
•
•

circuit optimizer

mapper and router

quantum processor

OneQ: A Compilation Framework for Photonic One-Way Quantum Computation

Hezi Zhang
hezi@cs.uchicago.edu

Anbang Wu
anbang@cs.uchicago.edu

Yuke Wang
wang@cs.ucsb.edu
University of California
Santa Barbara, USA

Reza Shabani
bani@cisco.com
Cisco Quantum Lab
Santa Monica, USA

Qubit Mapping for Reconfigurable Atom Arrays

Bochen Tan
bochentan@ucla.edu
University of California, Los Angeles
Los Angeles, CA, USA

Dolev Bluvstein
dbluvstein@g.harvard.edu
Department of Physics, Harvard University
Cambridge, MA, USA

Mikhail D. Lukin
lukin@physics.harvard.edu
Department of Physics, Harvard University

Jason Cong
cong@cs.ucla.edu
University of California, Los Angeles
Los Angeles, CA, USA

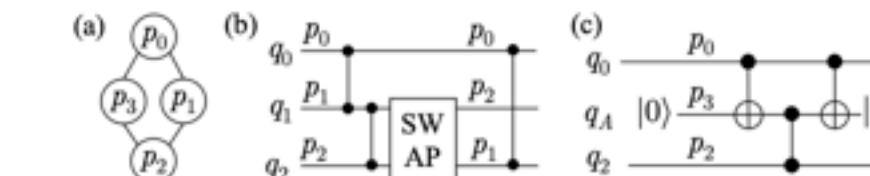
Mikhail D. Lukin, and Jason Cong. 2022.
Q-Pilot: Field Programmable Qubit Array Compilation with Flying Ancillas. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)*, October 30–November 3, 2022, New York, NY, USA, 9 pages. <https://doi.org/10.1109/ICCAD55005.2022.9810001>

Q-Pilot: Field Programmable Qubit Array Compilation with Flying Ancillas

Hanrui Wang^{1*}, Daniel Bochen Tan^{2*}, Pengyu Liu³, Yilian Liu⁴, Jiaqi Gu⁵, Jason Cong², Song Han¹
¹MIT, ²University of California, Los Angeles, ³Carnegie Mellon University, ⁴Cornell University, ⁵Arizona State University, *Equal Contributions

ABSTRACT

Neutral atom arrays have become a promising platform for quantum computing, especially the *field programmable qubit array* (FPQA) endowed with the unique capability of atom movement. This feature allows dynamic alterations in qubit connections, which can reduce the cost of executing logical operations and improve parallelism. However, this added flexibility poses challenges in circuit compilation. Inspired by routing strategies for FPGAs, we propose



Circuit decompositions and scheduling for neutral atom devices with limited local addressability

Natalia Nottingham
Dept. of Computer Science
University of Chicago
Chicago, IL, USA
nottingham@uchicago.edu

Michael A. Perlin
Global Technology Applied Research
JPMorganChase
New York, NY, USA
michael.perlin@jpmchase.com

Dhirpal Shah
Dept. of Computer Science
University of Chicago
Chicago, IL, USA
dhirpalshah@uchicago.edu

Ryan White
Dept. of Physics
University of Chicago
Chicago, IL, USA
rwhite@uchicago.edu

GEYSER: A Compilation Framework for Quantum Computing with Neutral Atoms

Tirthak Patel
Northeastern University
Boston, USA

Daniel Silver
Northeastern University
Boston, USA

Devesh Tiwari
Northeastern University
Boston, USA

ABSTRACT

Compared to widely-used superconducting qubits, neutral-atom quantum computing technology promises potentially better scala-

the most promising technologies – each offering their own unique advantages over other competing technologies [3, 8, 21, 37]. We anticipate that multiple technologies will be in production to serve

We need to synthesize quantum compilers

huge diversity in qubits, architectures, fault-tolerance schemes

(quantum) compilers are hard to get right*

Synthesizing quantum compilers

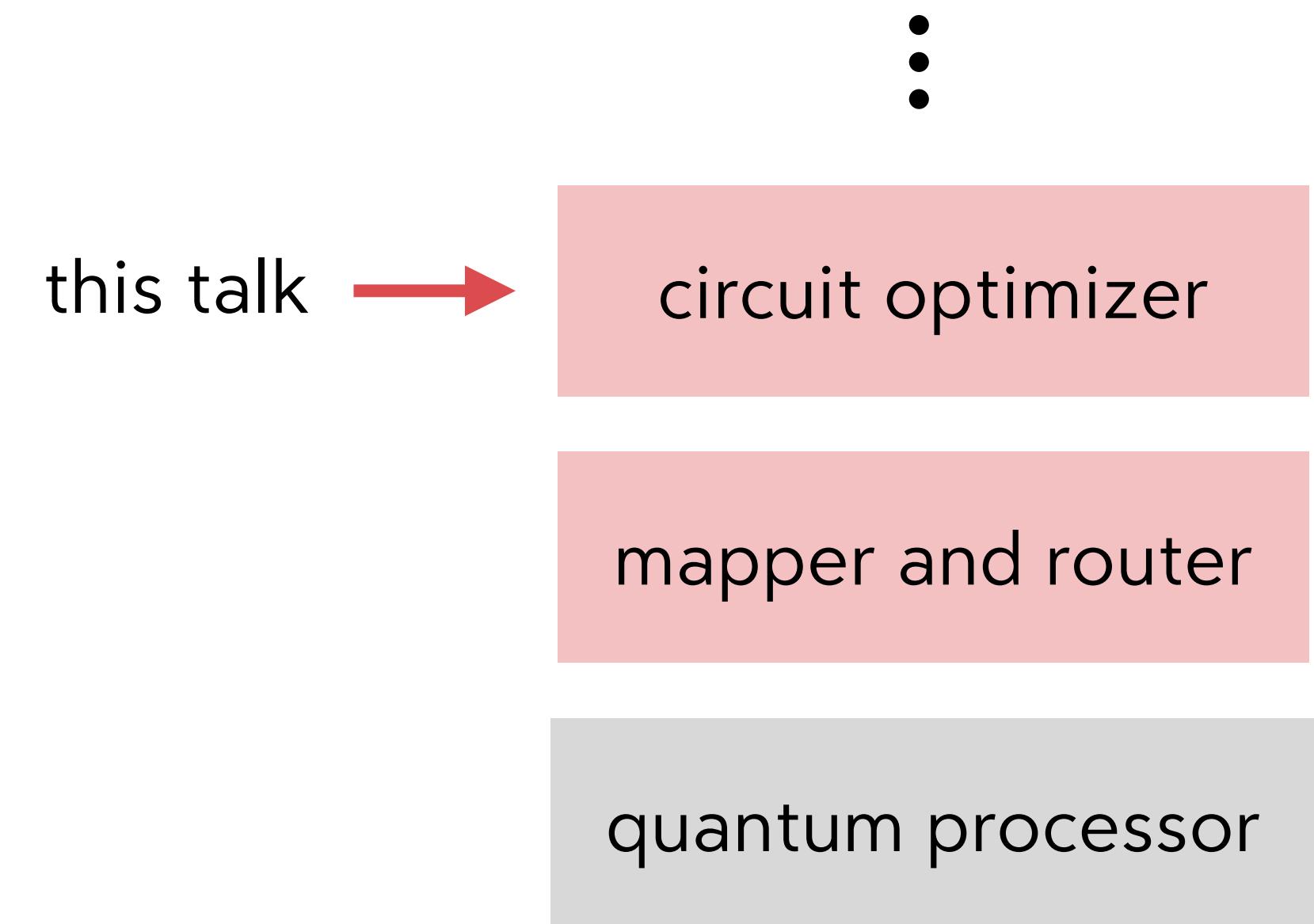
⋮

circuit optimizer

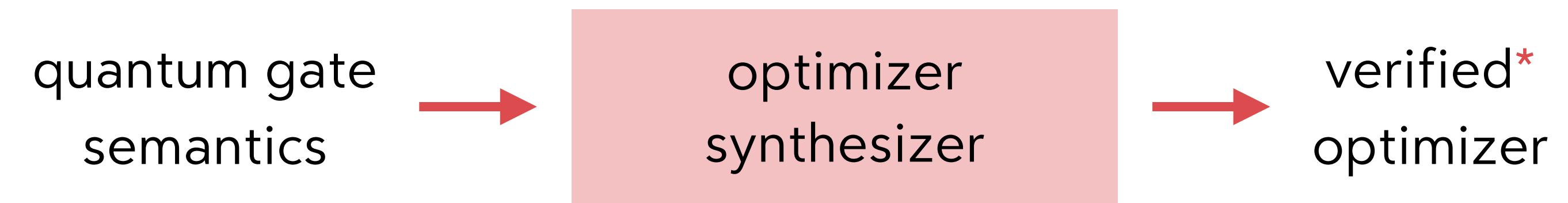
mapper and router

quantum processor

Synthesizing quantum compilers

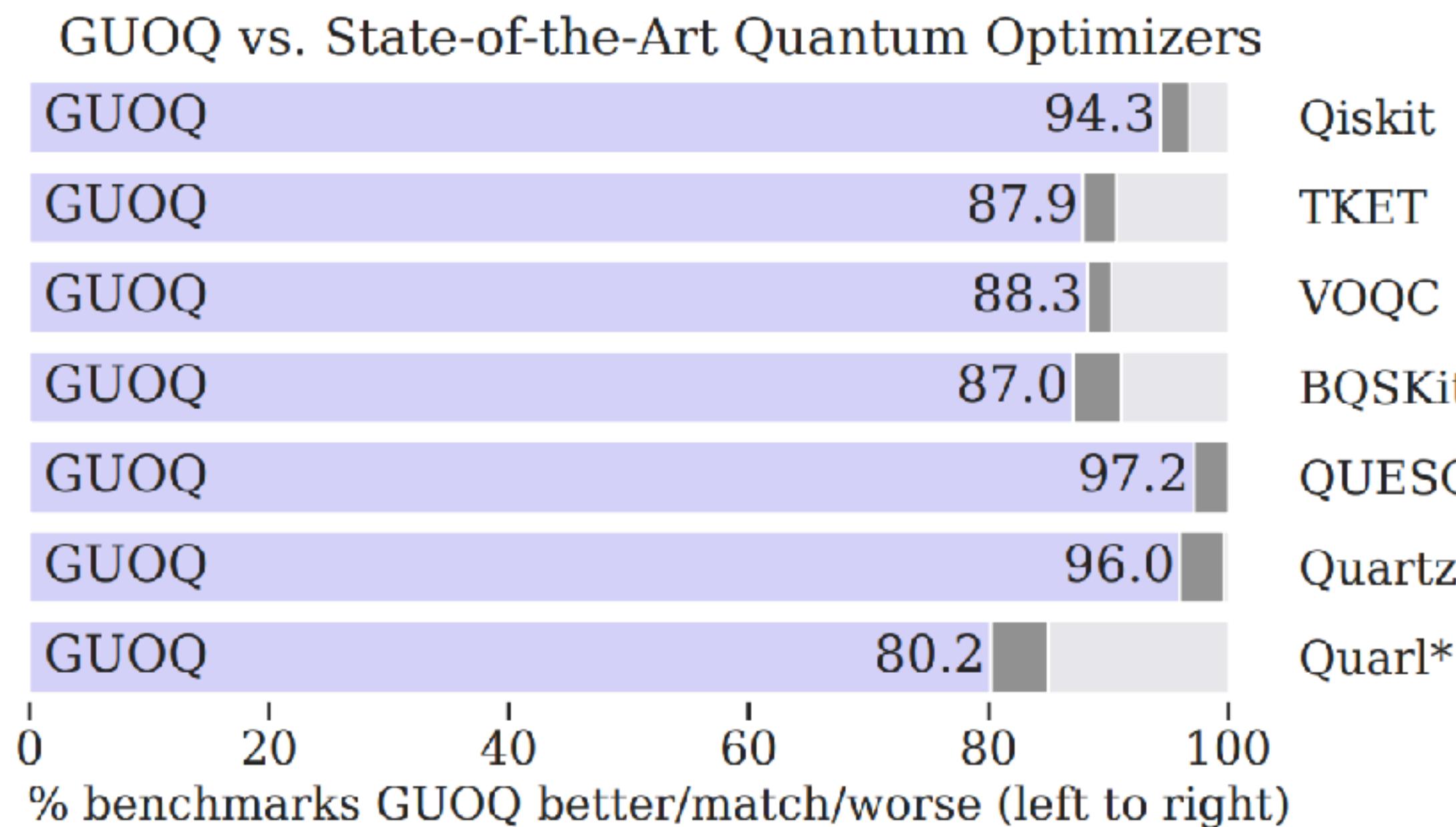
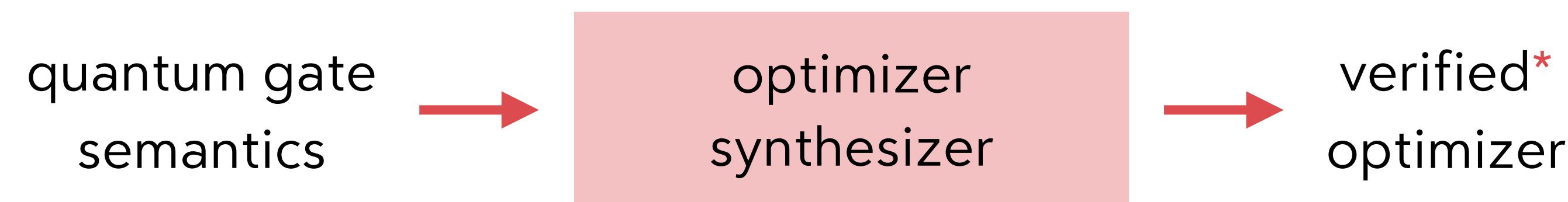


Synthesizing circuit optimizers



* not everything will be verified

Synthesizing circuit optimizers



Synthesizing circuit optimizers

A learning rewrite rules

B optimizing, fast and slow

Synthesizing circuit optimizers

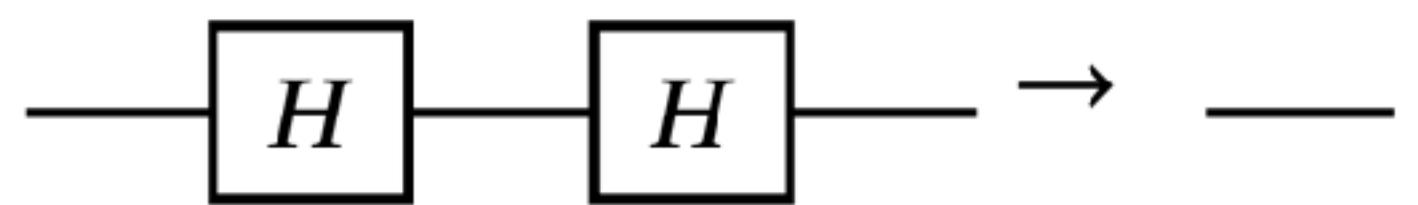
A learning rewrite rules

B optimizing, fast and slow

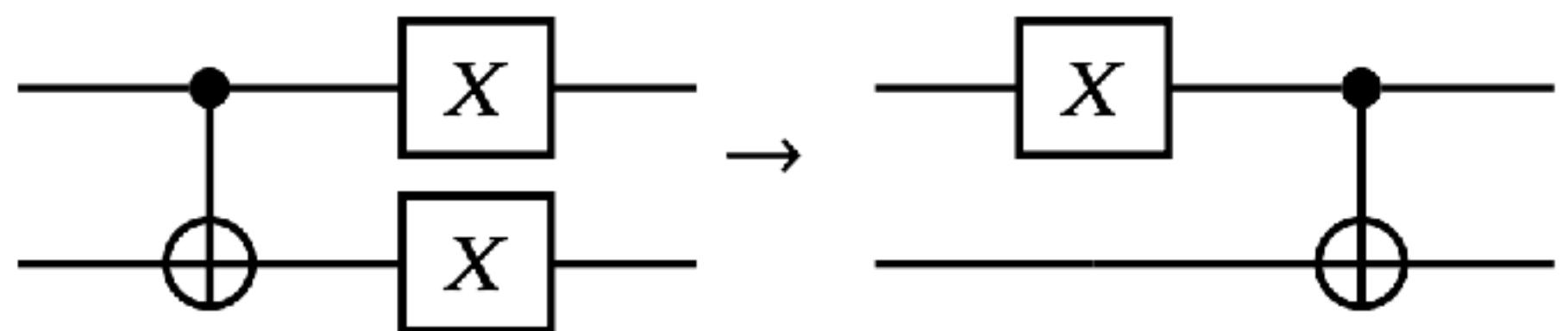
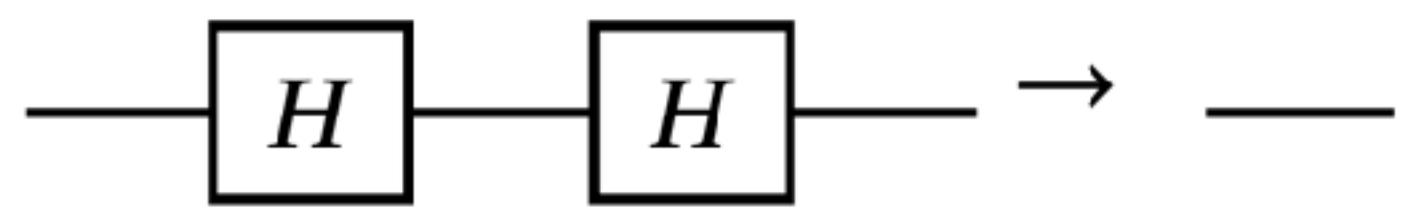
theme simple, classic algorithms go a long way

What is a rewrite rule?

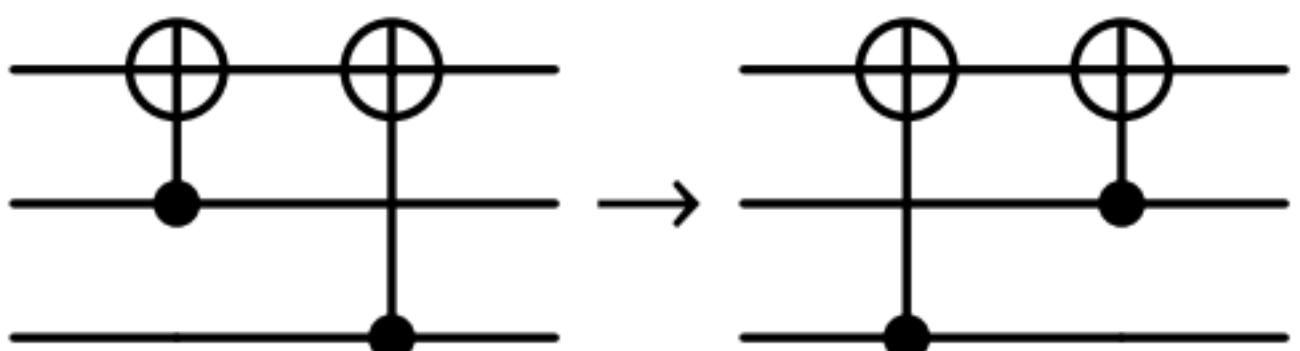
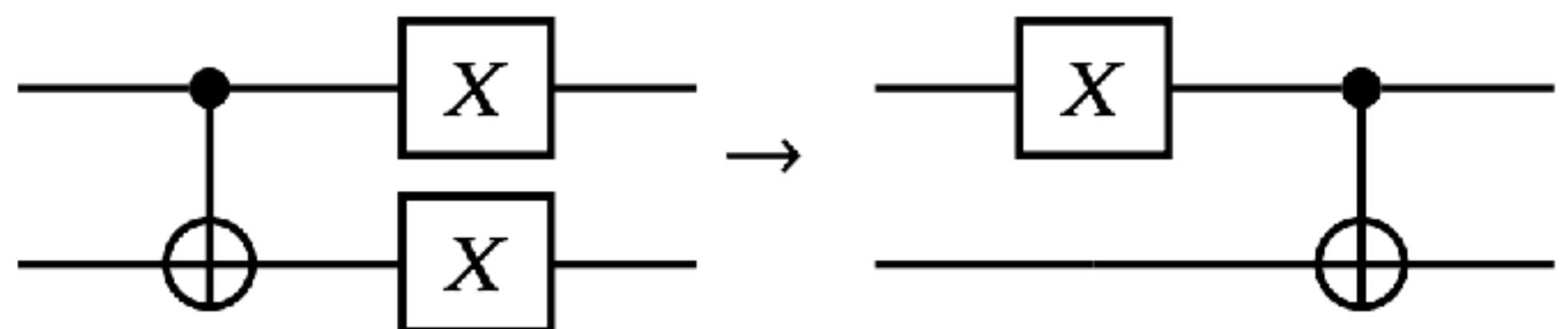
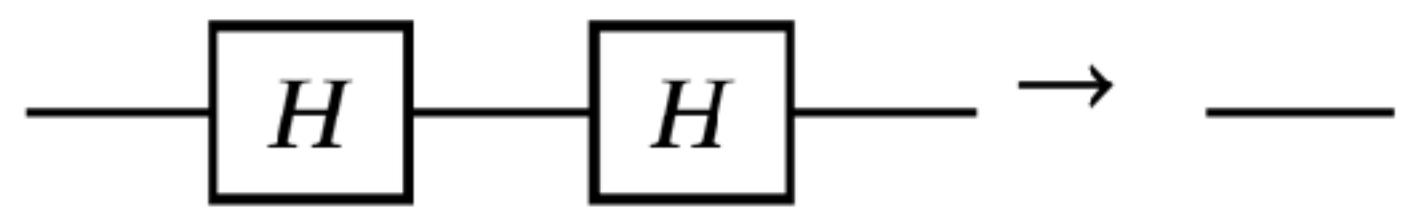
What is a rewrite rule?



What is a rewrite rule?

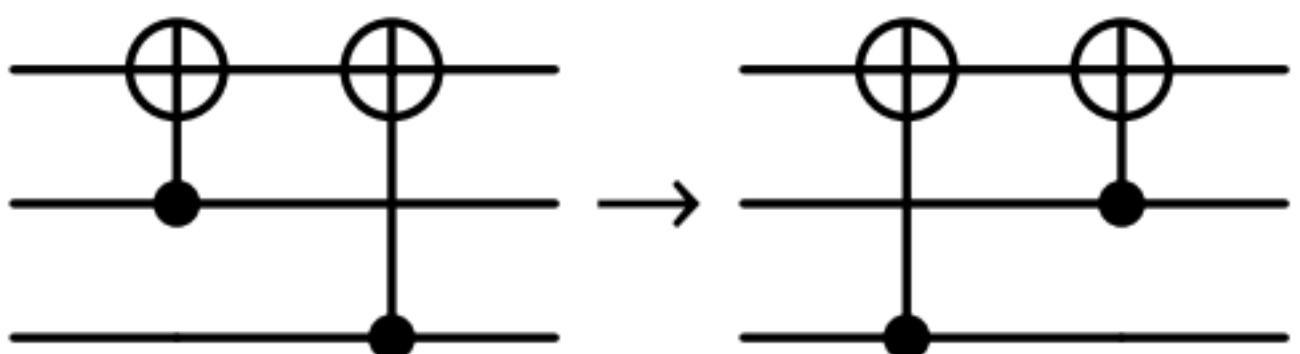
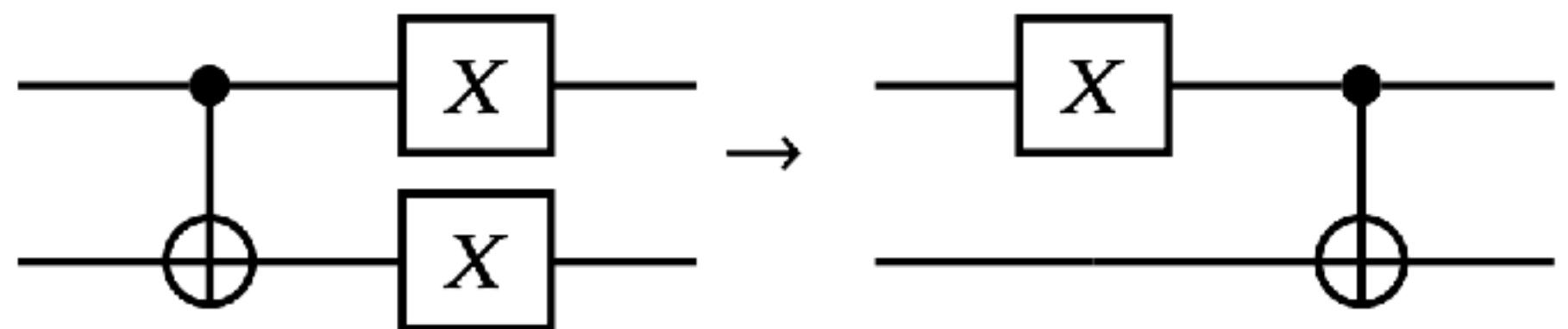
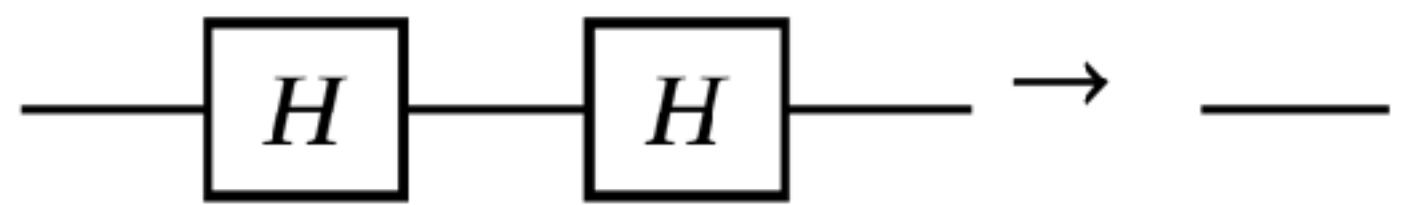


What is a rewrite rule?



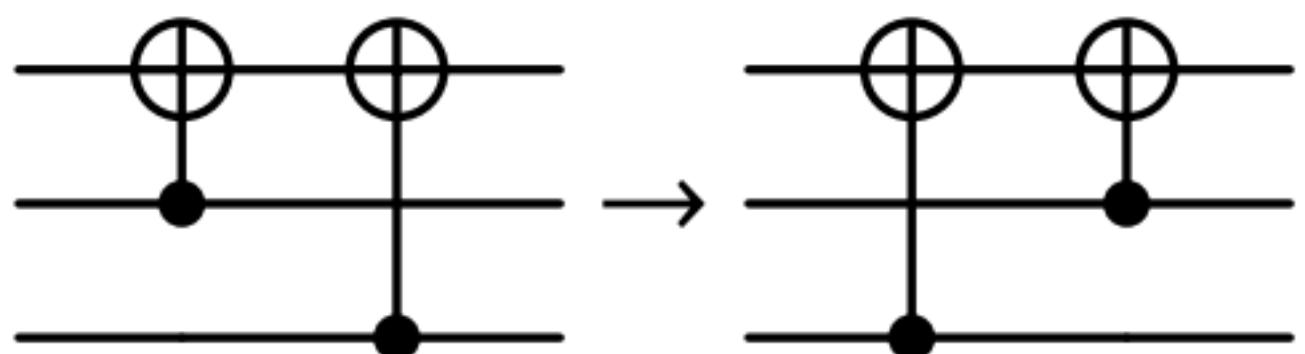
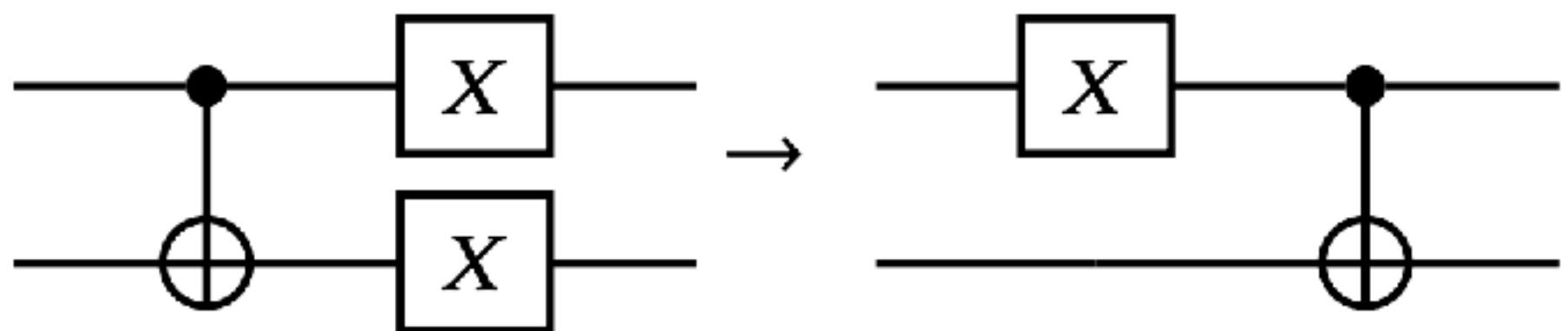
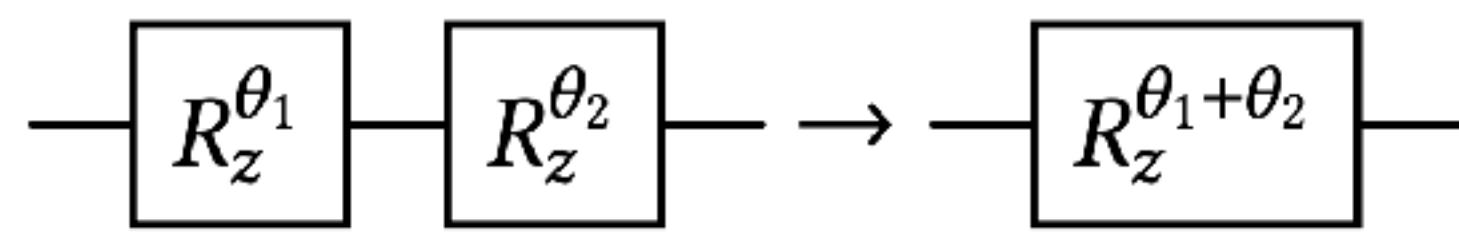
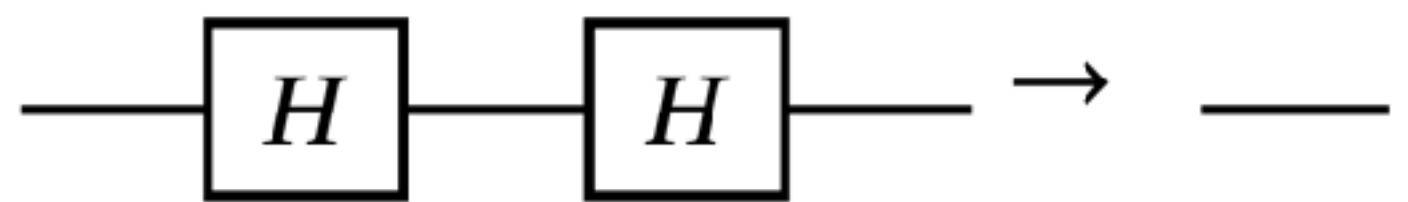
What is a rewrite rule?

Symbolic rewrite rules



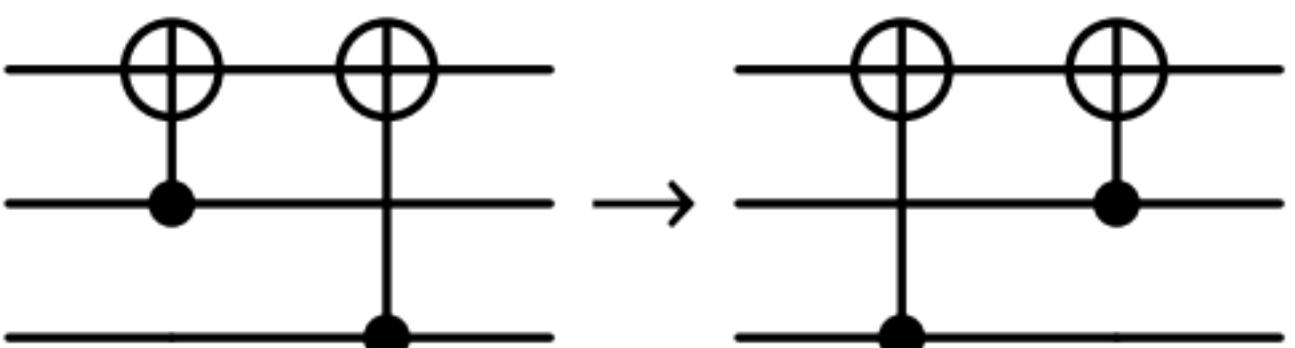
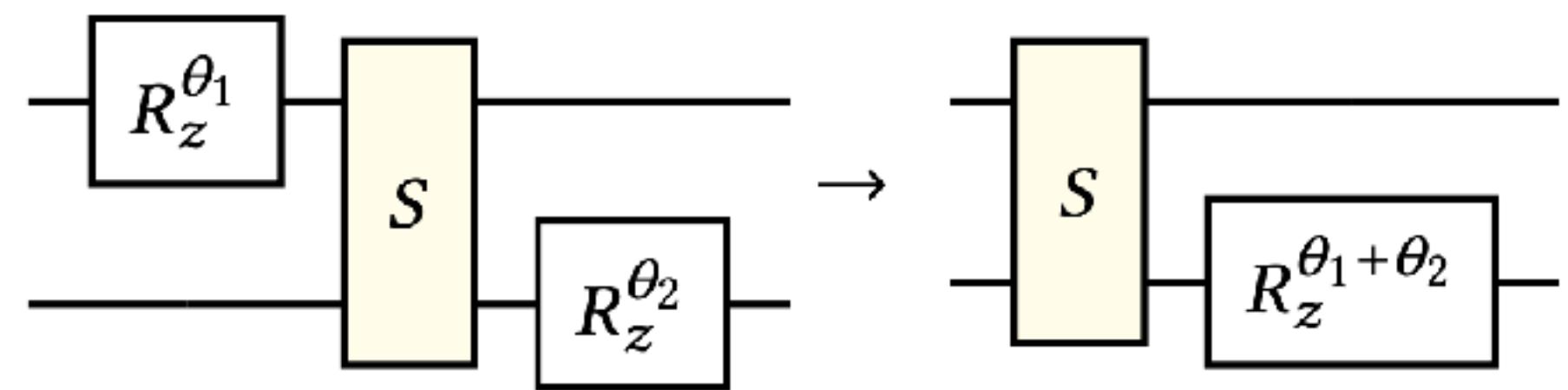
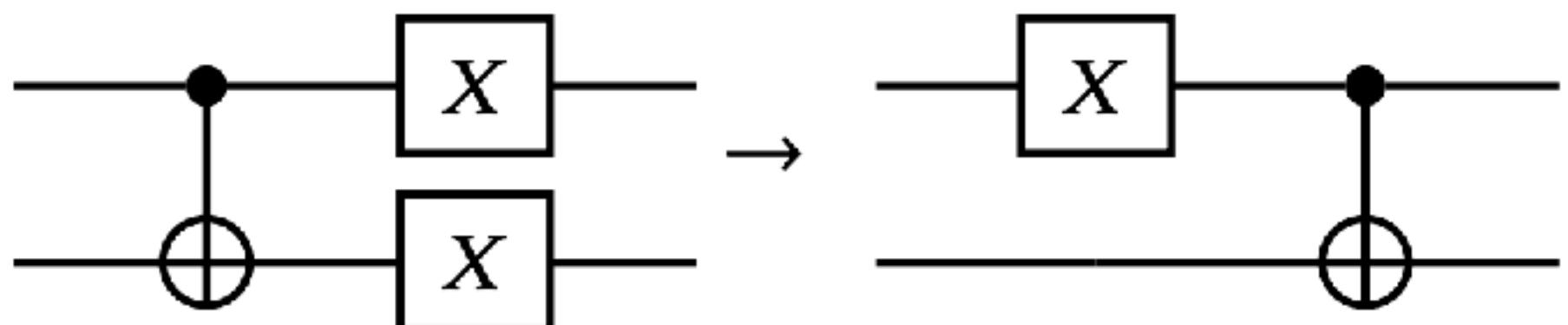
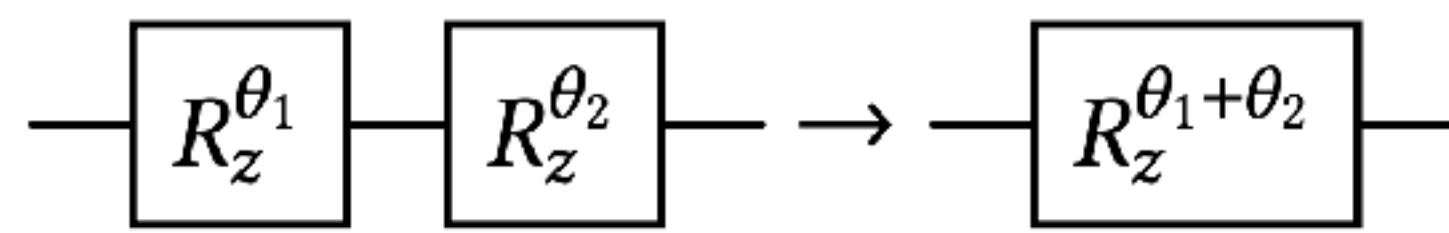
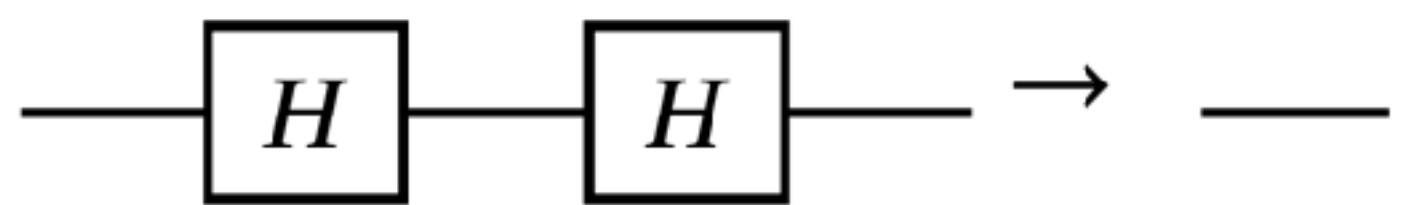
What is a rewrite rule?

Symbolic rewrite rules



What is a rewrite rule?

Symbolic rewrite rules



Challenges in learning rewrite rules

Challenges in learning rewrite rules

how can we search the vast space of (symbolic) rewrite rules?

Challenges in learning rewrite rules

how can we search the vast space of (symbolic) rewrite rules?

how do we schedule rewrite rules?

Naive synthesis of rewrite rules

Naive synthesis of rewrite rules

```
rules = []
```

Naive synthesis of rewrite rules

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

Naive synthesis of rewrite rules

```
rules = []

circuits = enumerate(max_qubits, max_size)

for (c1,c2) in circuits x circuits:
```

Naive synthesis of rewrite rules

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

even for small circuit sizes, we're talking about 10^{11} to 10^{18} rules

Naive synthesis of rewrite rules

```
rules = []

circuits = enumerate(max_qubits, max_size)

for (c1,c2) in circuits x circuits:
    if verify_equivalence(c1,c2):
```

even for small circuit sizes, we're talking about 10^{11} to 10^{18} rules

Naive synthesis of rewrite rules

```
rules = []

circuits = enumerate(max_qubits, max_size)

for (c1,c2) in circuits x circuits:
    if verify_equivalence(c1,c2):
        rules.append(c1 → c2)
```

even for small circuit sizes, we're talking about 10^{11} to 10^{18} rules

Key insights for rewrite synthesis

Key insights for rewrite synthesis

symbolic circuits are polynomials over the complex field

Key insights for rewrite synthesis

symbolic circuits are polynomials over the complex field

polynomial identity testing is easy — Schwartz–Zippel lemma

Key insights for rewrite synthesis

symbolic circuits are polynomials over the complex field

polynomial identity testing is easy — Schwartz–Zippel lemma

a simple, new data structure called a polynomial identity filter (PIF)

Circuit equivalence

circuits

C_1

C_2

Circuit equivalence

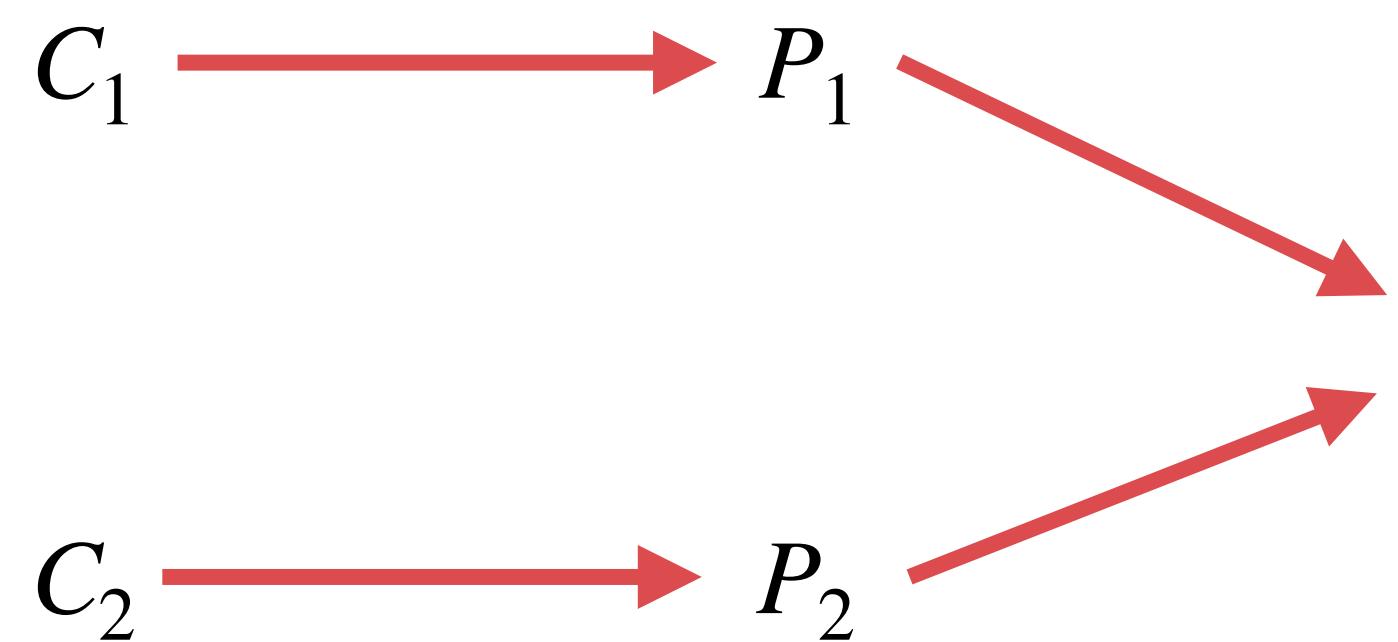
circuits polynomials

$$C_1 \xrightarrow{\hspace{1cm}} P_1$$

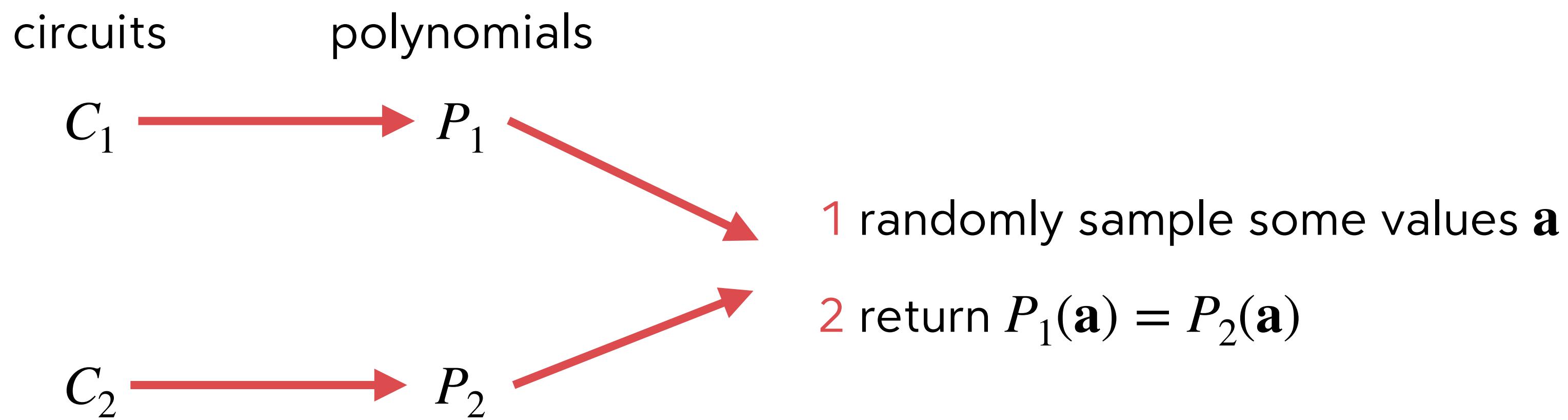
$$C_2 \xrightarrow{\hspace{1cm}} P_2$$

Circuit equivalence

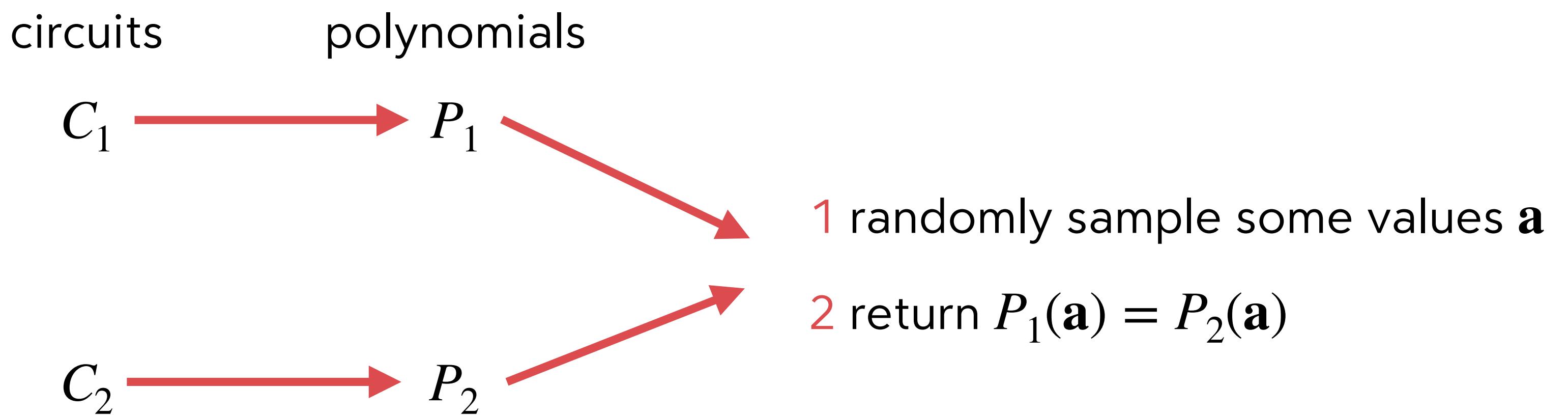
circuits polynomials



Circuit equivalence



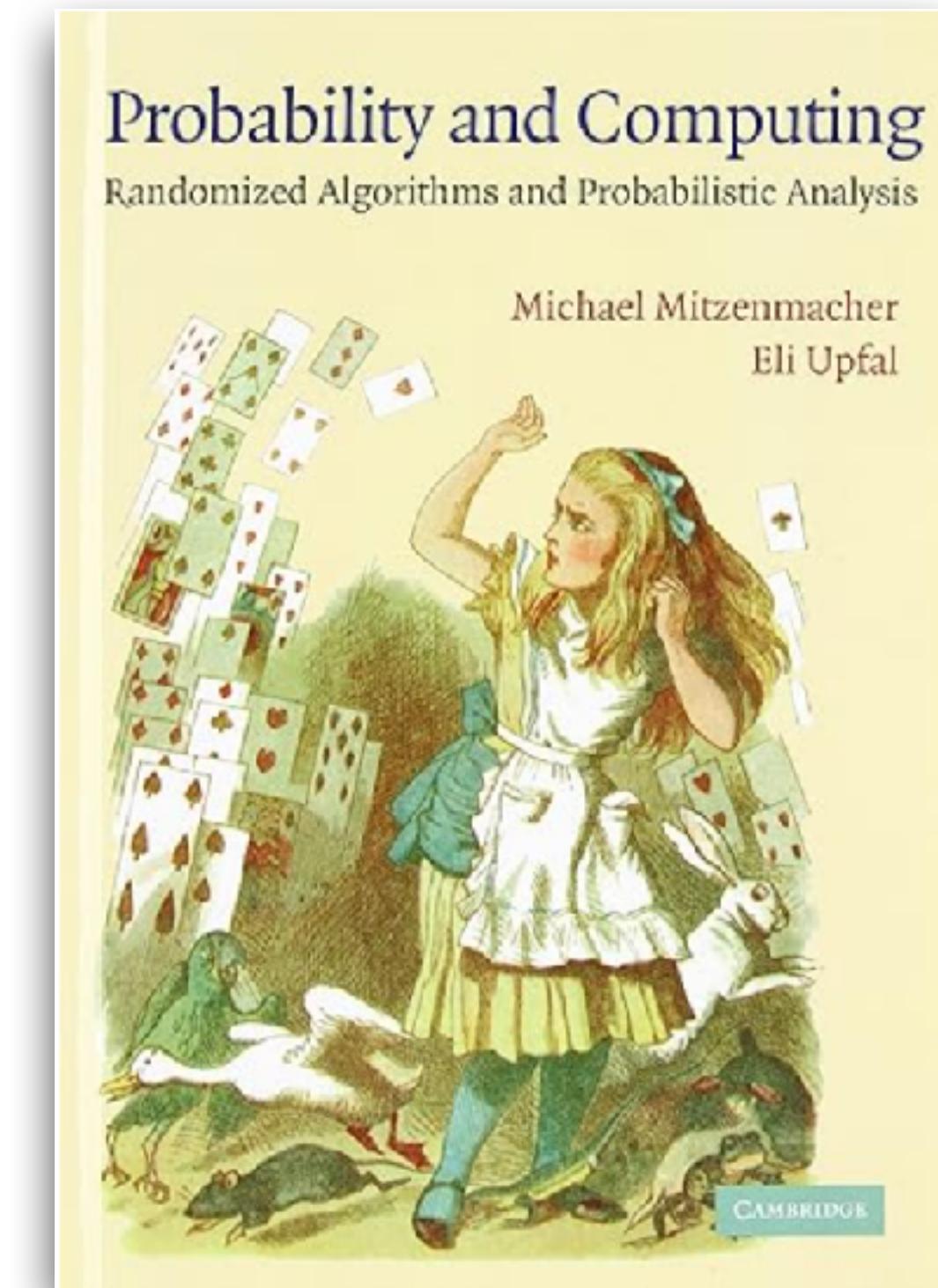
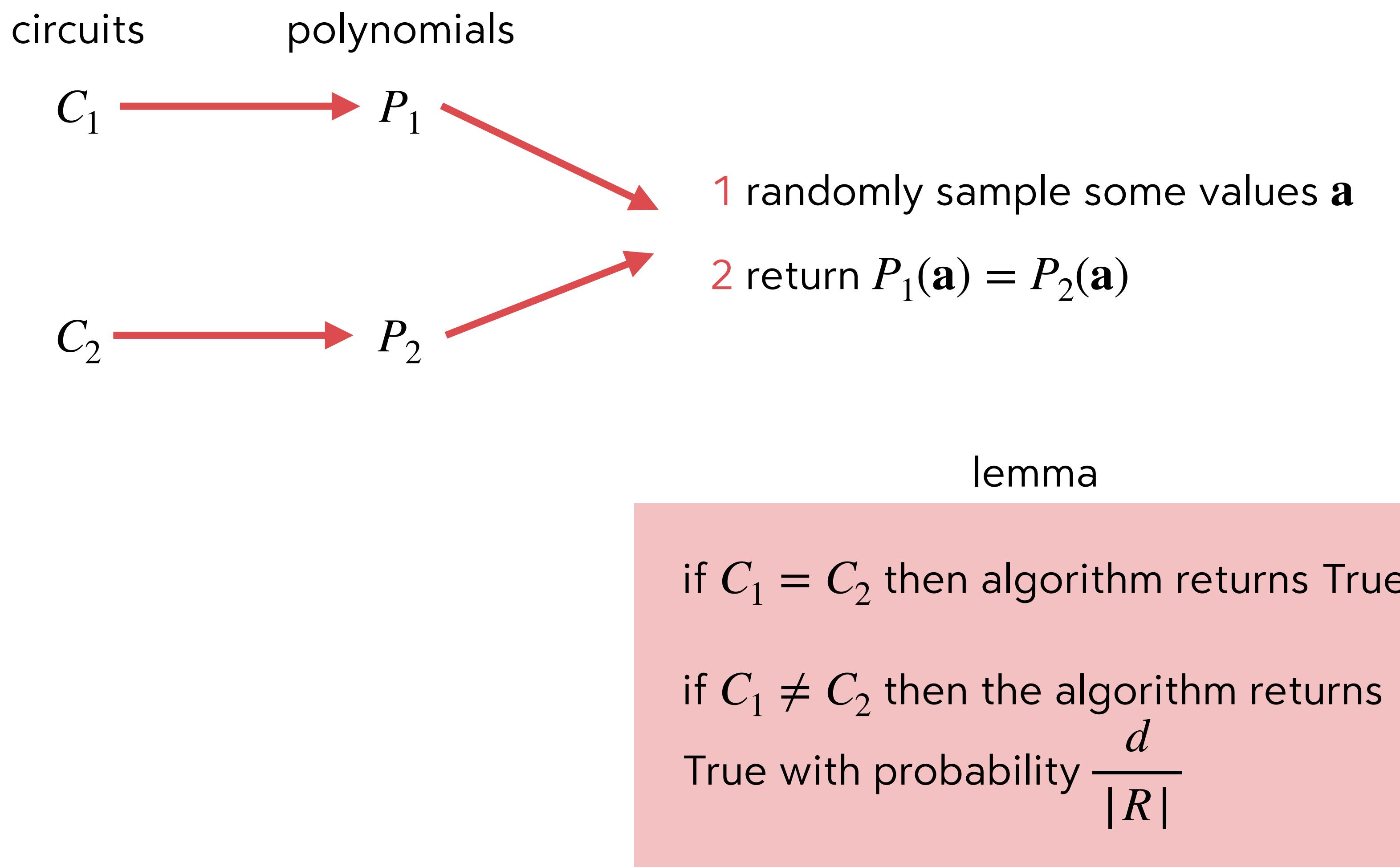
Circuit equivalence



lemma

if $C_1 = C_2$ then algorithm returns True
if $C_1 \neq C_2$ then the algorithm returns
True with probability $\frac{d}{|R|}$

Circuit equivalence



Polynomial identity filter (PIF)

circuits

C_1

C_2

C_3

\vdots

C_n

Polynomial identity filter (PIF)

circuits

C_1

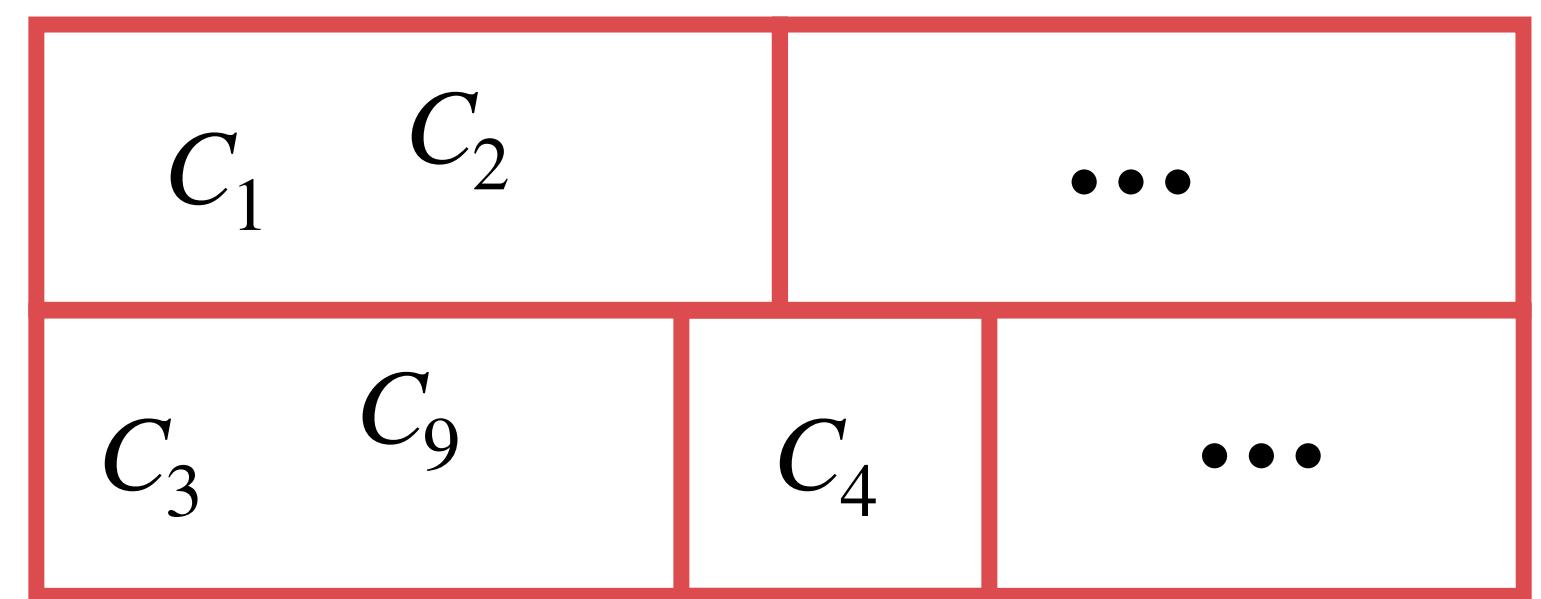
C_2

C_3

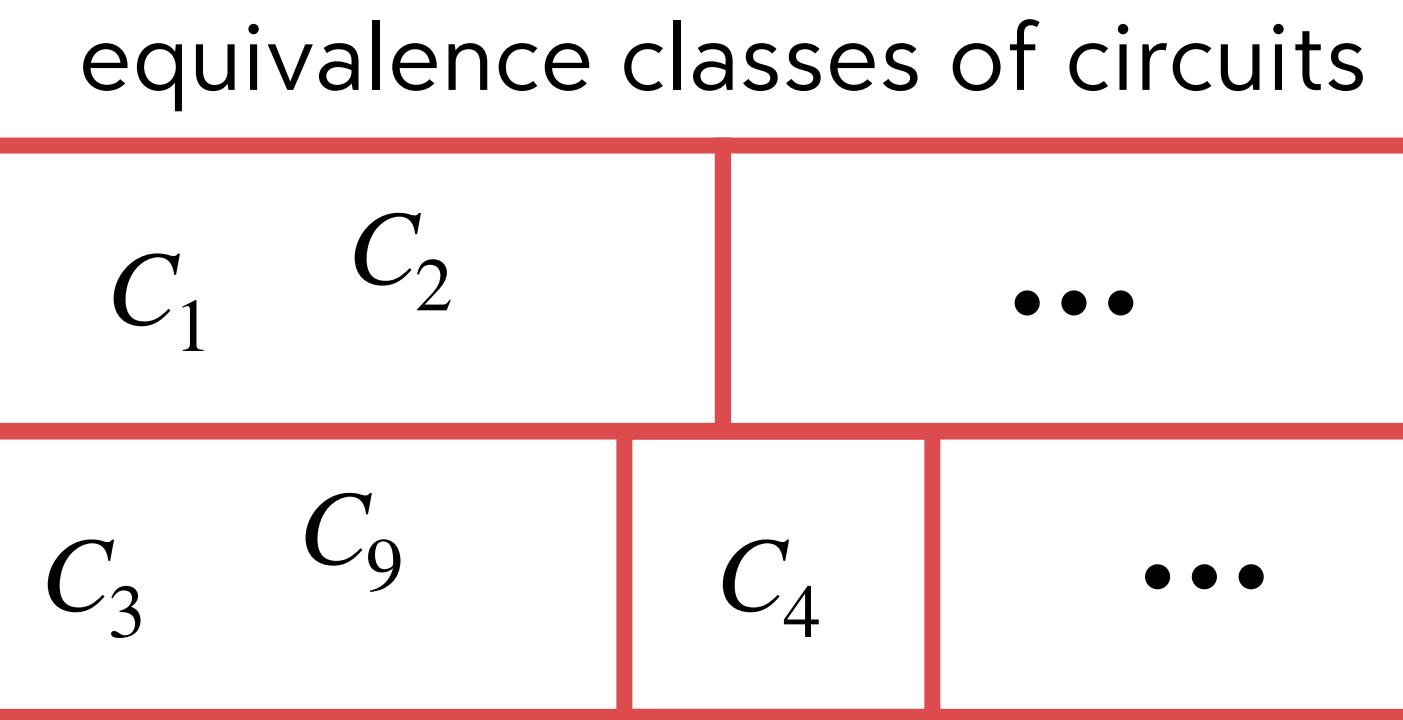
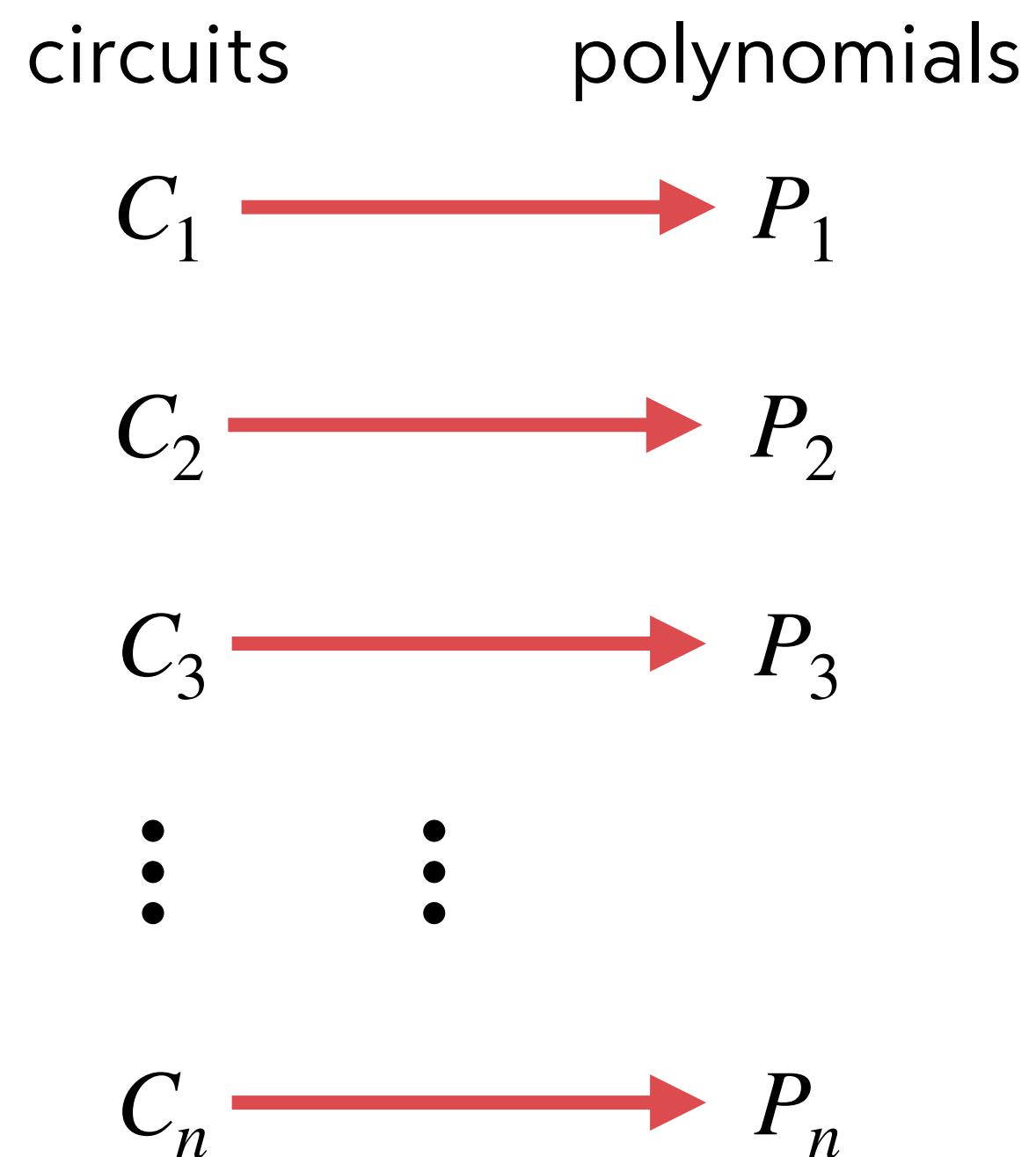
⋮

C_n

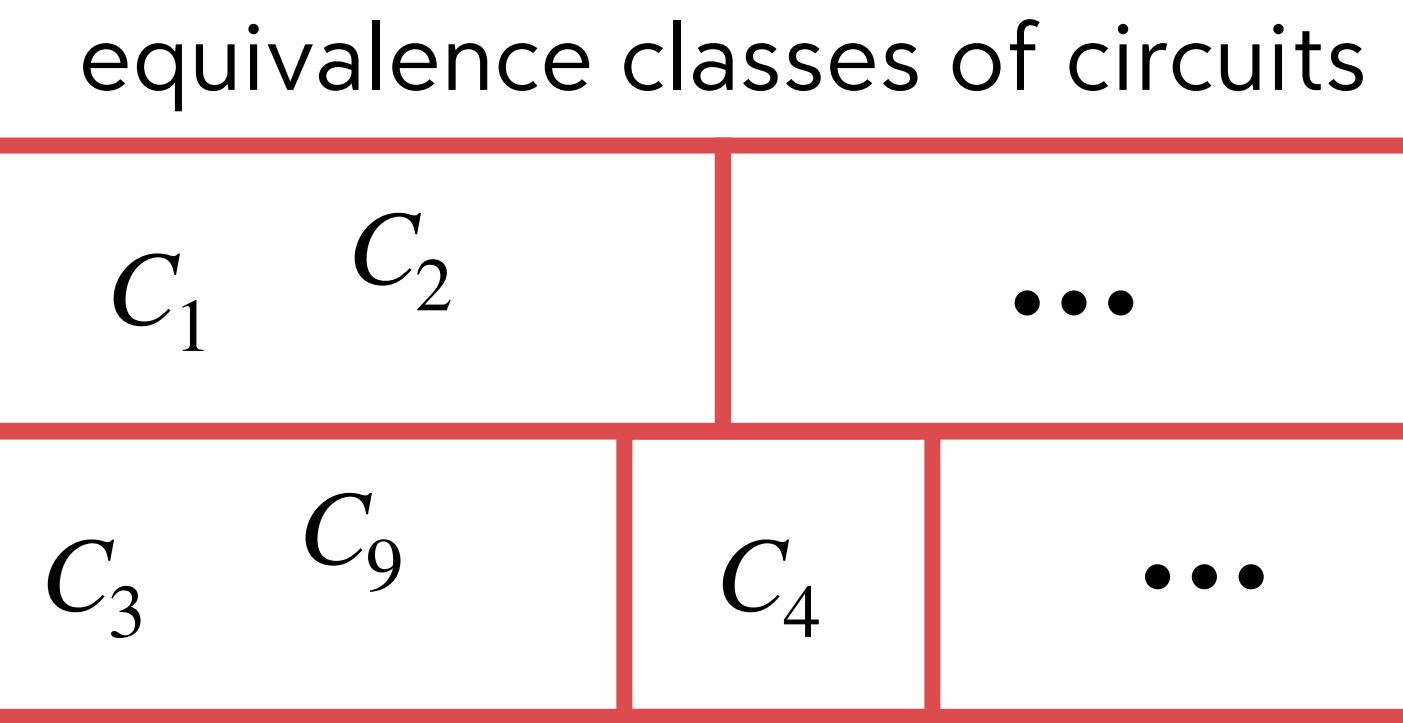
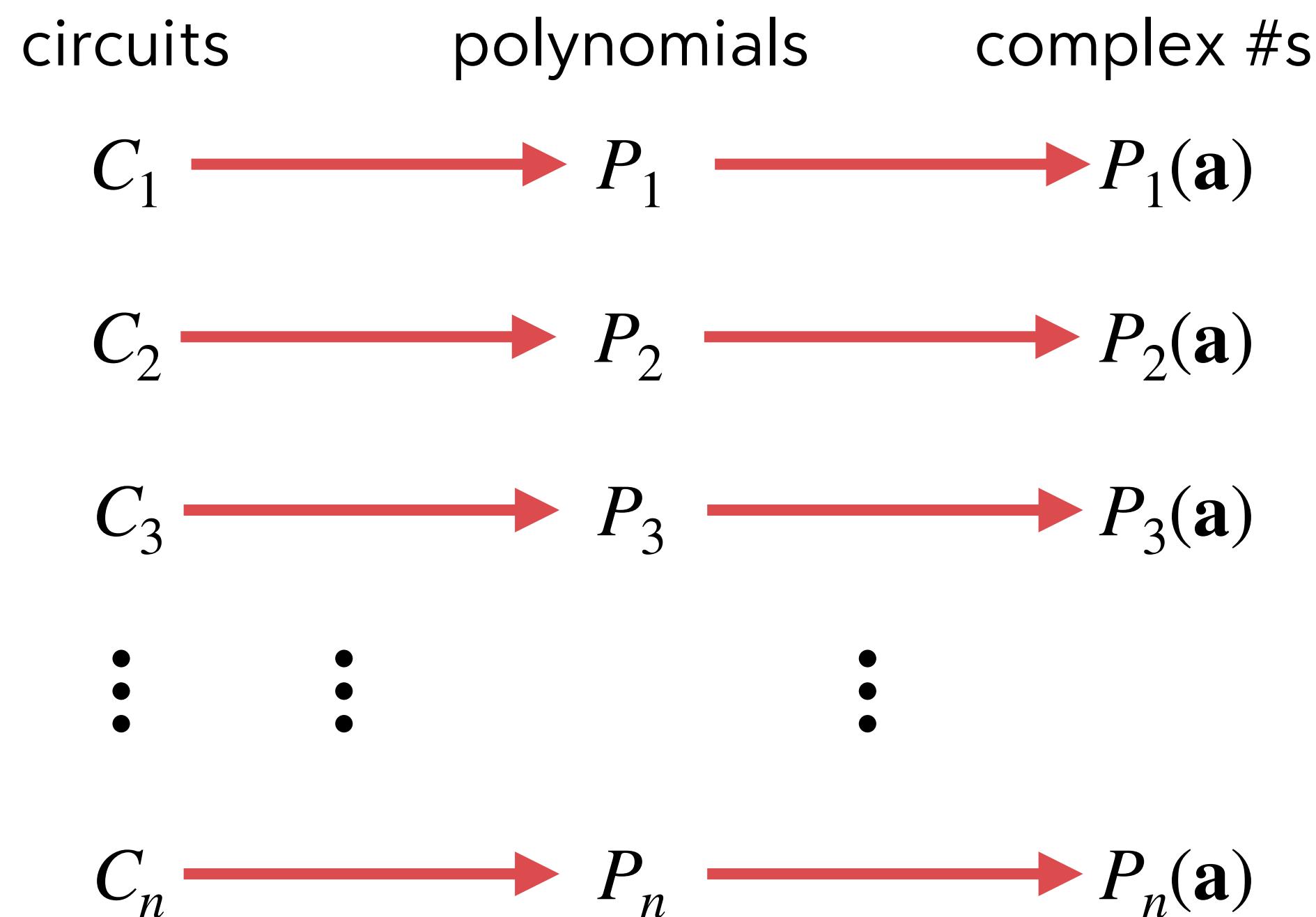
equivalence classes of circuits



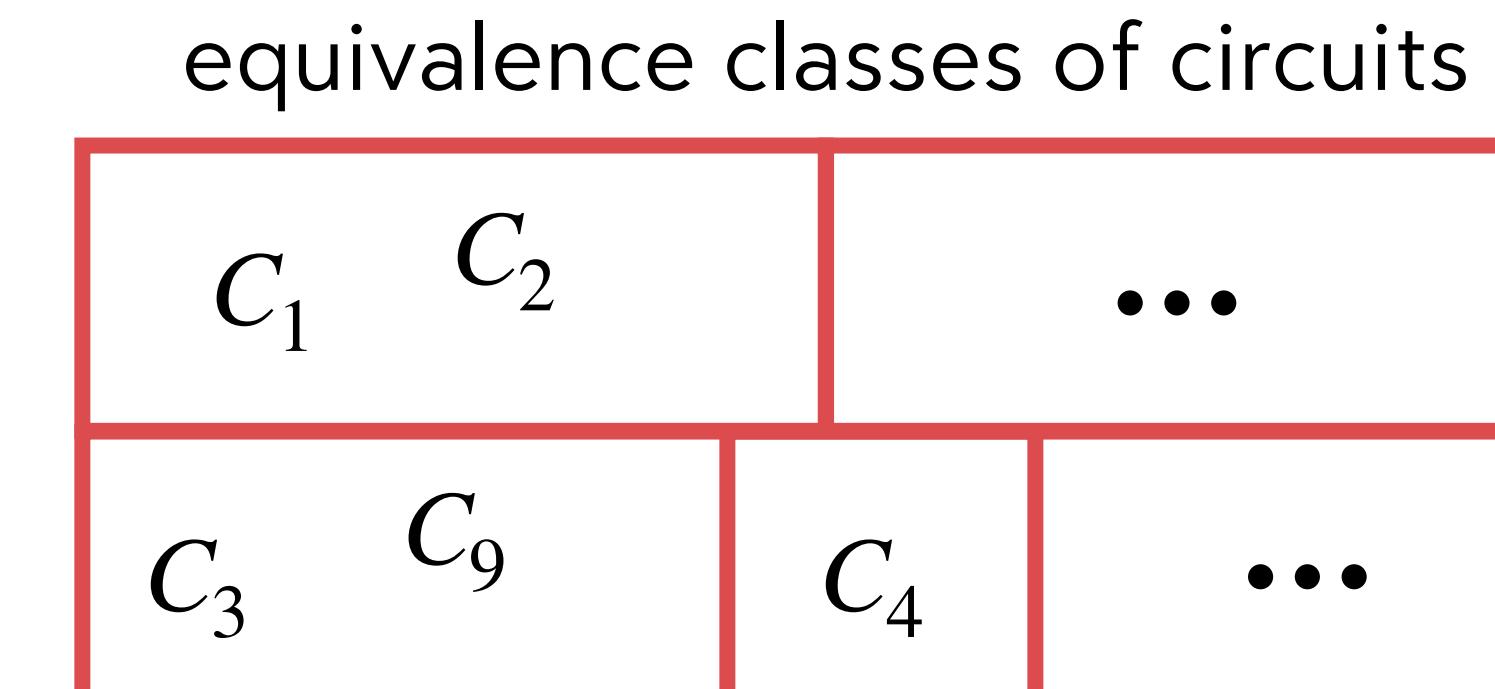
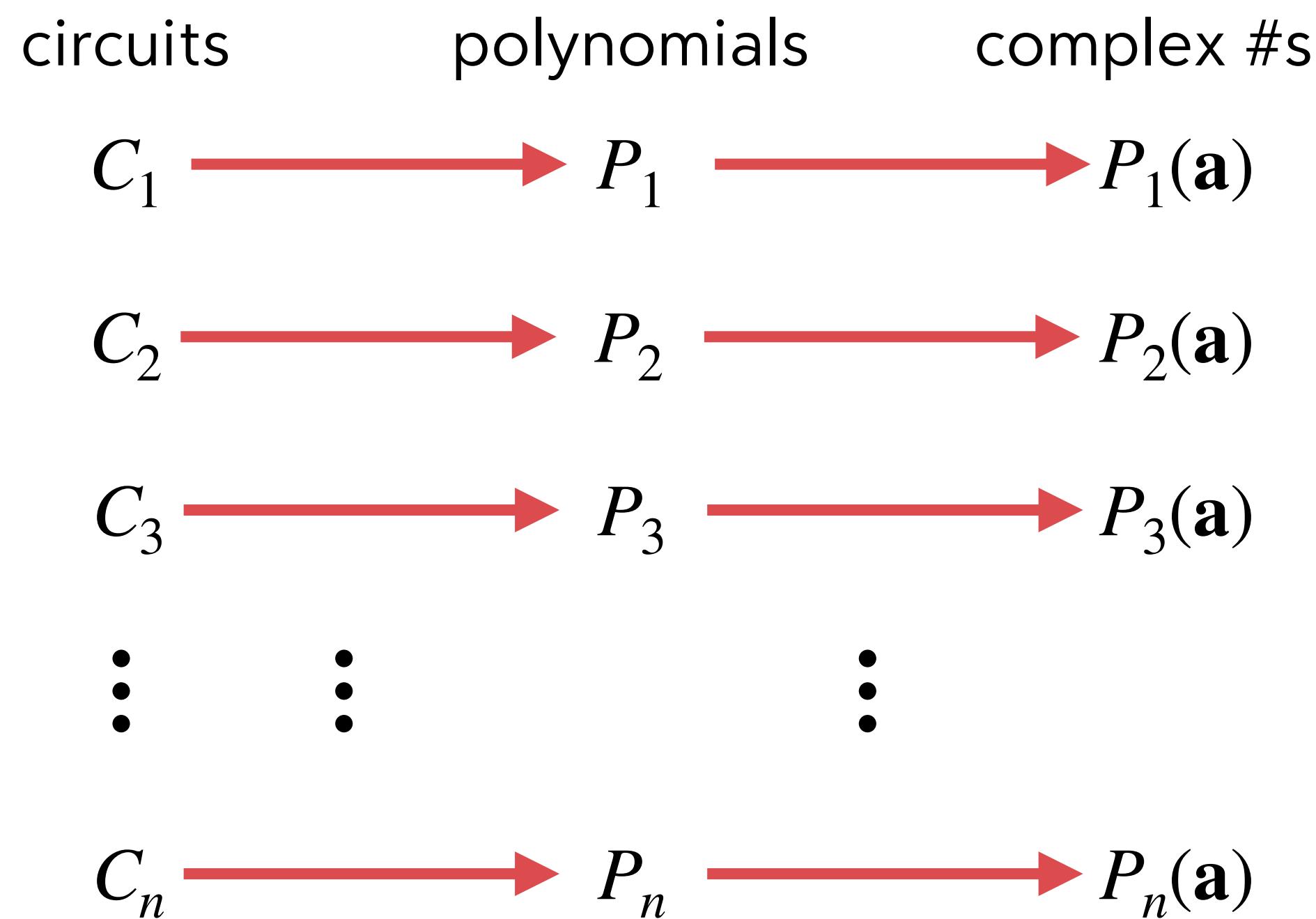
Polynomial identity filter (PIF)



Polynomial identity filter (PIF)



Polynomial identity filter (PIF)



theorem

probability of a wrong rewrite rule at most $\frac{n^2 d}{|R|}$

Symbolic circuits as polynomials

$H \quad q;$
 $Rz(\theta) \quad q;$

Symbolic circuits as polynomials

H $q;$
 $Rz(\theta)$ $q;$



$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right)$$

Symbolic circuits as polynomials

H $q;$
 $Rz(\theta)$ $q;$



amplitude

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right)$$

Symbolic circuits as polynomials

H $q;$
 $Rz(\theta)$ $q;$



\mathbb{C} var amplitude

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right)$$

Symbolic circuits as polynomials

H $q;$
 $Rz(\theta)$ $q;$



$\mathbb{C} \text{ var}$ amplitude

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) - \left(v_{1,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right)$$

Symbolic circuits as polynomials

H $q;$
 $Rz(\theta)$ $q;$



$\mathbb{C} \text{ var}$ amplitude

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) - \left(v_{1,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right)$$

Symbolic circuits as polynomials

H q;
 $Rz(\theta)$ q;



constrained polynomials

rewrite

$$e^{i\theta} \rightarrow z$$

$$e^{i2\theta} \rightarrow z^2$$

...

constrain variables to
unit circle

\mathbb{C} var amplitude

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) - \left(v_{1,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right)$$

Symbolic circuits as polynomials

```
H      q;  
Rz(θ) q;  
symb q;
```

Symbolic circuits as polynomials

```
H      q;  
Rz(θ) q;  
symb q;
```

symb semantics
 $|x\rangle \rightarrow \phi|x\rangle$

Symbolic circuits as polynomials

```
H      q;  
Rz(θ) q;  
symb q;
```



symb semantics
 $|x\rangle \rightarrow \phi|x\rangle$

$$\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta}\right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta}\right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta}\right) + \left(v_{1,1} \cdot \frac{e^{i\pi}}{\sqrt{2}} \cdot e^{i\theta}\right)$$

Symbolic circuits as polynomials

```
H      q;  
Rz(θ) q;  
symb q;
```



symb semantics
 $|x\rangle \rightarrow \phi|x\rangle$

$$\phi \cdot \left[\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{1,1} \cdot \frac{e^{i\pi}}{\sqrt{2}} \cdot e^{i\theta} \right) \right]$$

Symbolic circuits as polynomials

H q;
Rz(θ) q;
symb q;



symb semantics
 $|x\rangle \rightarrow \phi |x\rangle$

general **symb** semantics
 $|x\rangle \rightarrow \phi(x, y) |f(x, y)\rangle$

$$\phi \cdot \left[\left(v_{0,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{0,1} \cdot \frac{1}{\sqrt{2}} \cdot e^{i\theta} \right) + \left(v_{1,0} \cdot \frac{1}{\sqrt{2}} \cdot e^{-i\theta} \right) + \left(v_{1,1} \cdot \frac{e^{i\pi}}{\sqrt{2}} \cdot e^{i\theta} \right) \right]$$

The power of symbolic circuits

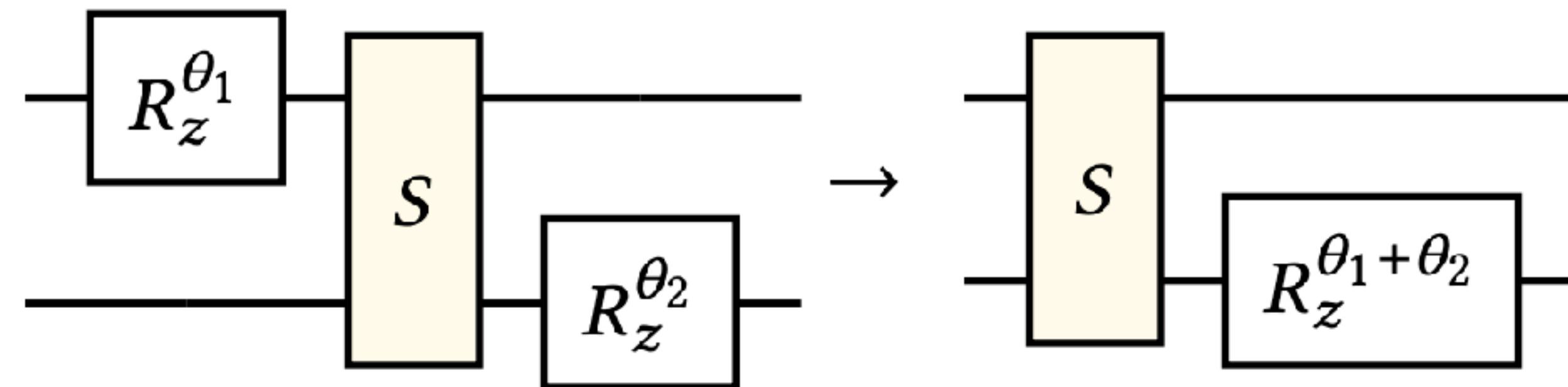
synthesize symbolic rules with long-range interaction

empirically very important set of rules

The power of symbolic circuits

synthesize symbolic rules with long-range interaction

empirically very important set of rules

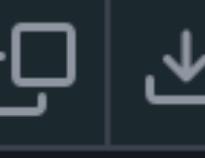


Ordering rewrite rules

Ordering rewrite rules

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

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

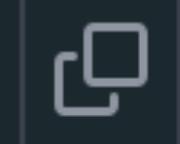
     

```
26
27 def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
28     """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
29     gate cancellation using commutativity rules and unitary synthesis.
30
31     This pass manager applies the user-given 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.
```

Ordering rewrite rules

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

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

```
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.
```

Ordering rewrite rules

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

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

```
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 trans coarse fixed passes coupling constraints.
37     It is then unrolled to the coarse fixed passes sections 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.
```

Optimizing, fast and slow

Optimizing, fast and slow

simulated annealing

Optimizing, fast and slow

simulated annealing

- pick one of the rules

Optimizing, fast and slow

simulated annealing

- pick one of the rules
- apply it to a subcircuit

Optimizing, fast and slow

simulated annealing

- pick one of the rules
- apply it to a subcircuit
- if the circuit is smaller, accept, otherwise reject with high probability

Optimizing, fast and slow

simulated annealing

- pick one of the rules
- apply it to a subcircuit
- if the circuit is smaller, accept, otherwise reject with high probability

dynamically generate new rule **1.5% of the time**

Optimizing, fast and slow

simulated annealing

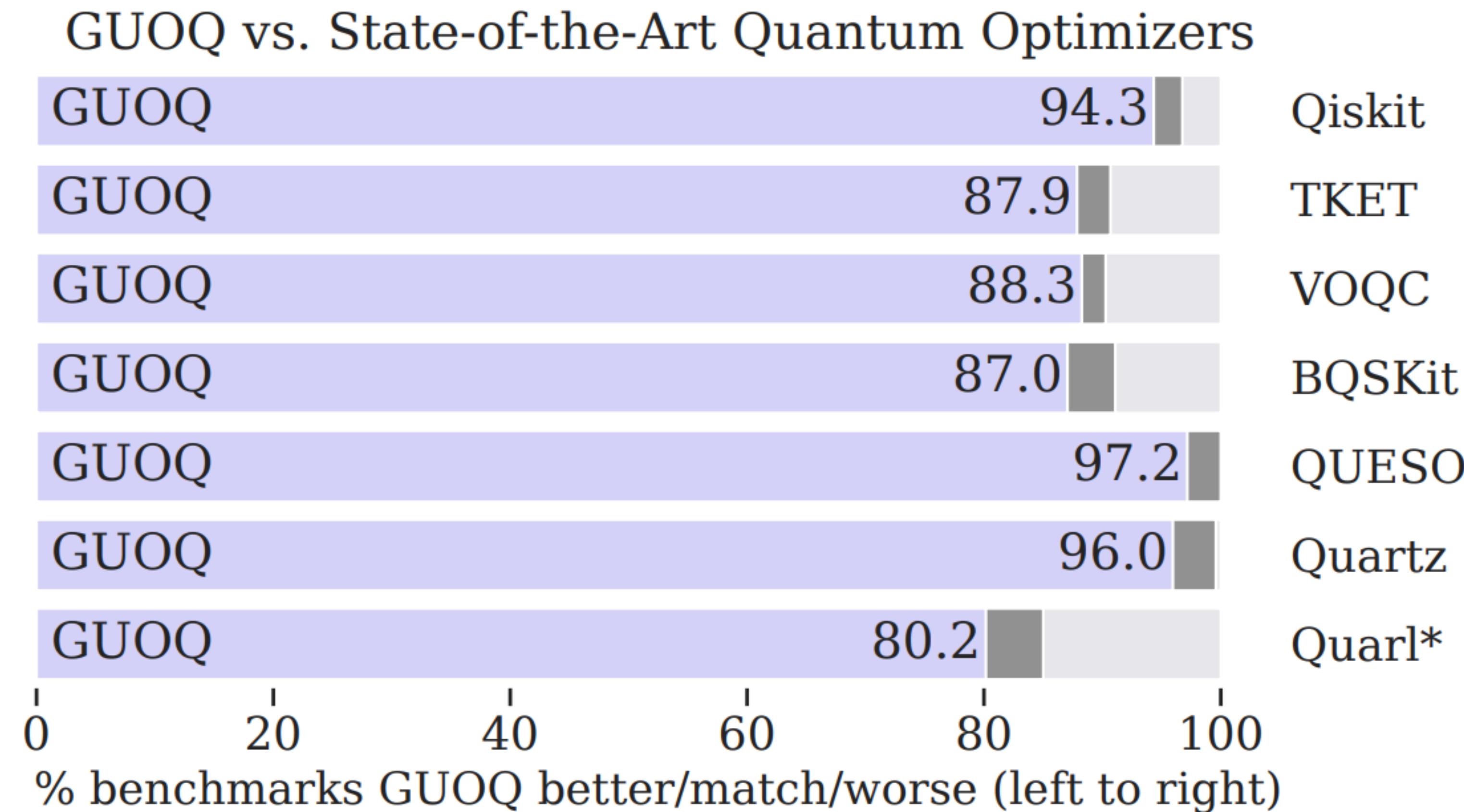
- pick one of the rules
- apply it to a subcircuit
- if the circuit is smaller, accept, otherwise reject with high probability

dynamically generate new rule **1.5% of the time**

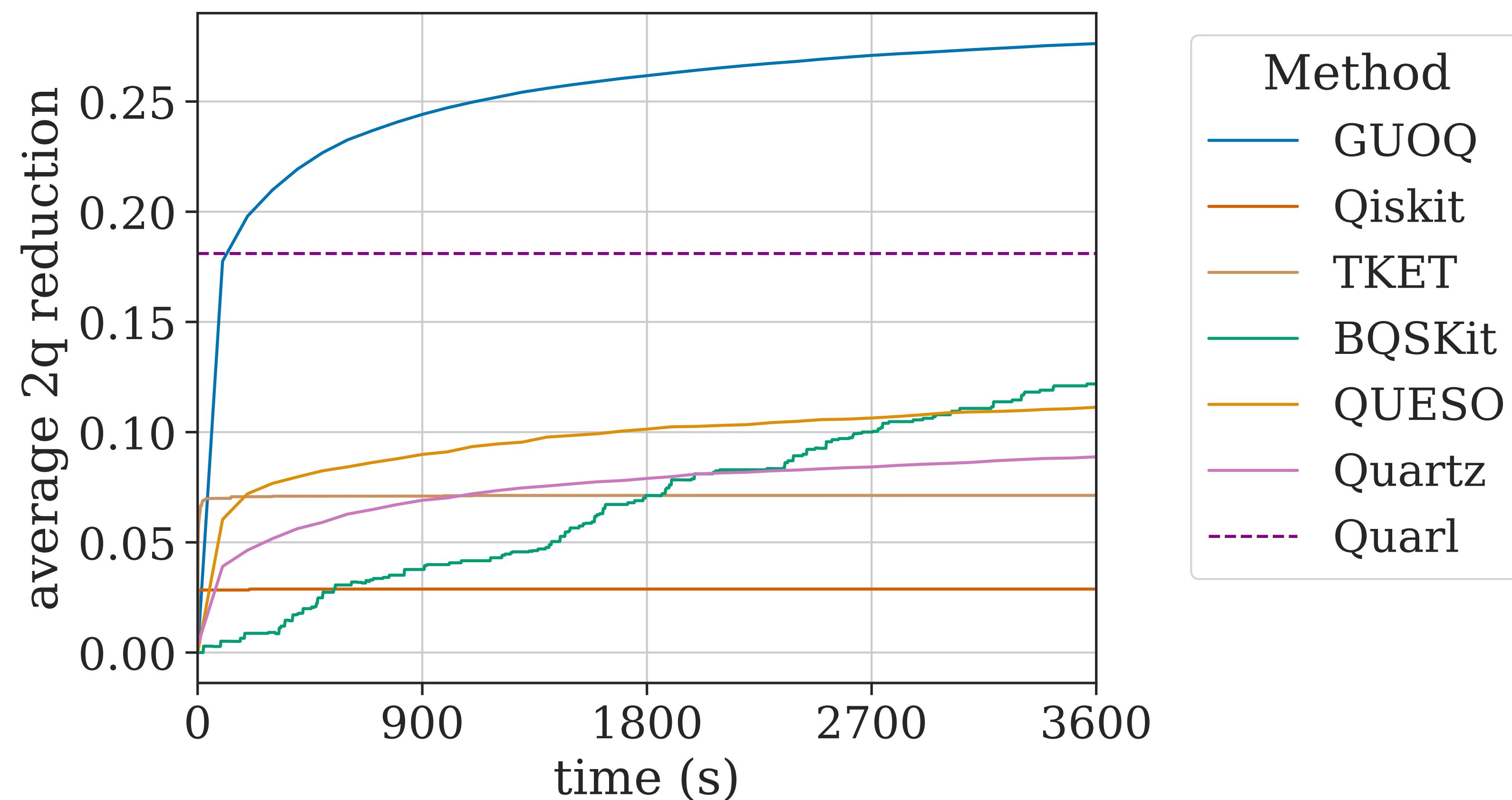
- use “resynthesis” tools—see our ASPLOS 2025 paper

Evaluation: Comparisons

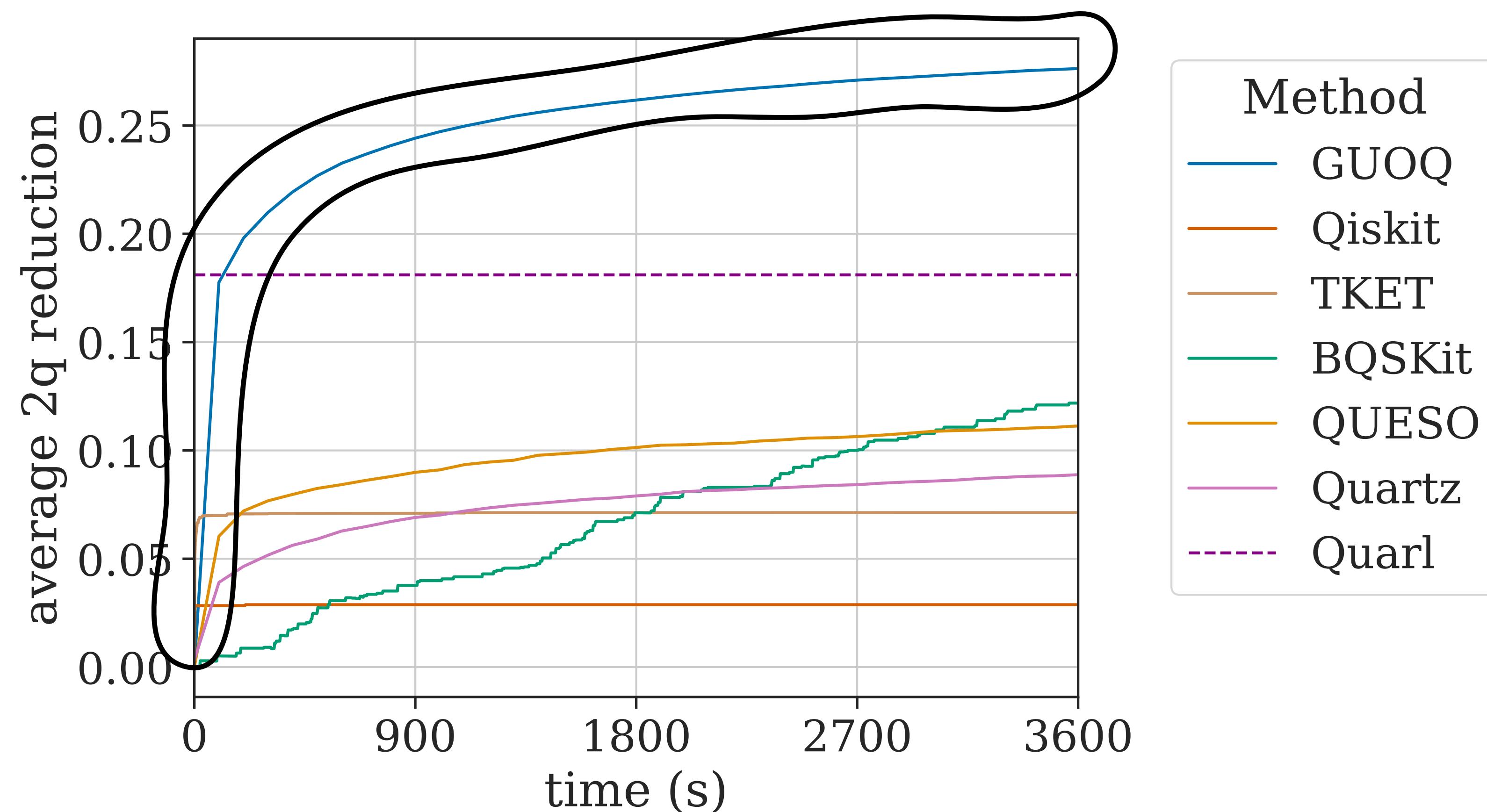
synthesis time: 1.2 min vs 10.4 min (Quartz)



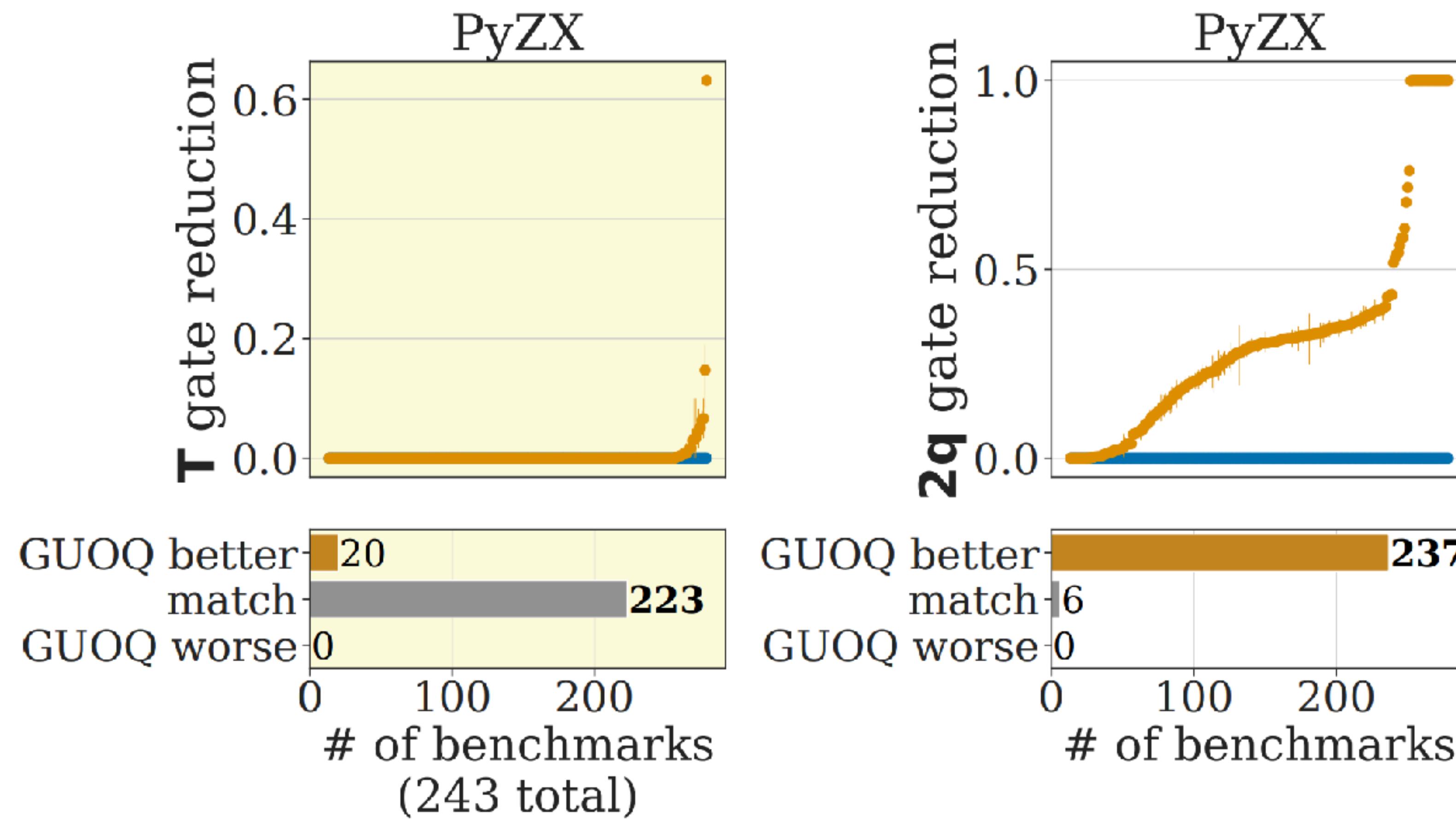
A closer look at reduction



A closer look at reduction



Evaluation: FTQC



Synthesizing quantum compilers

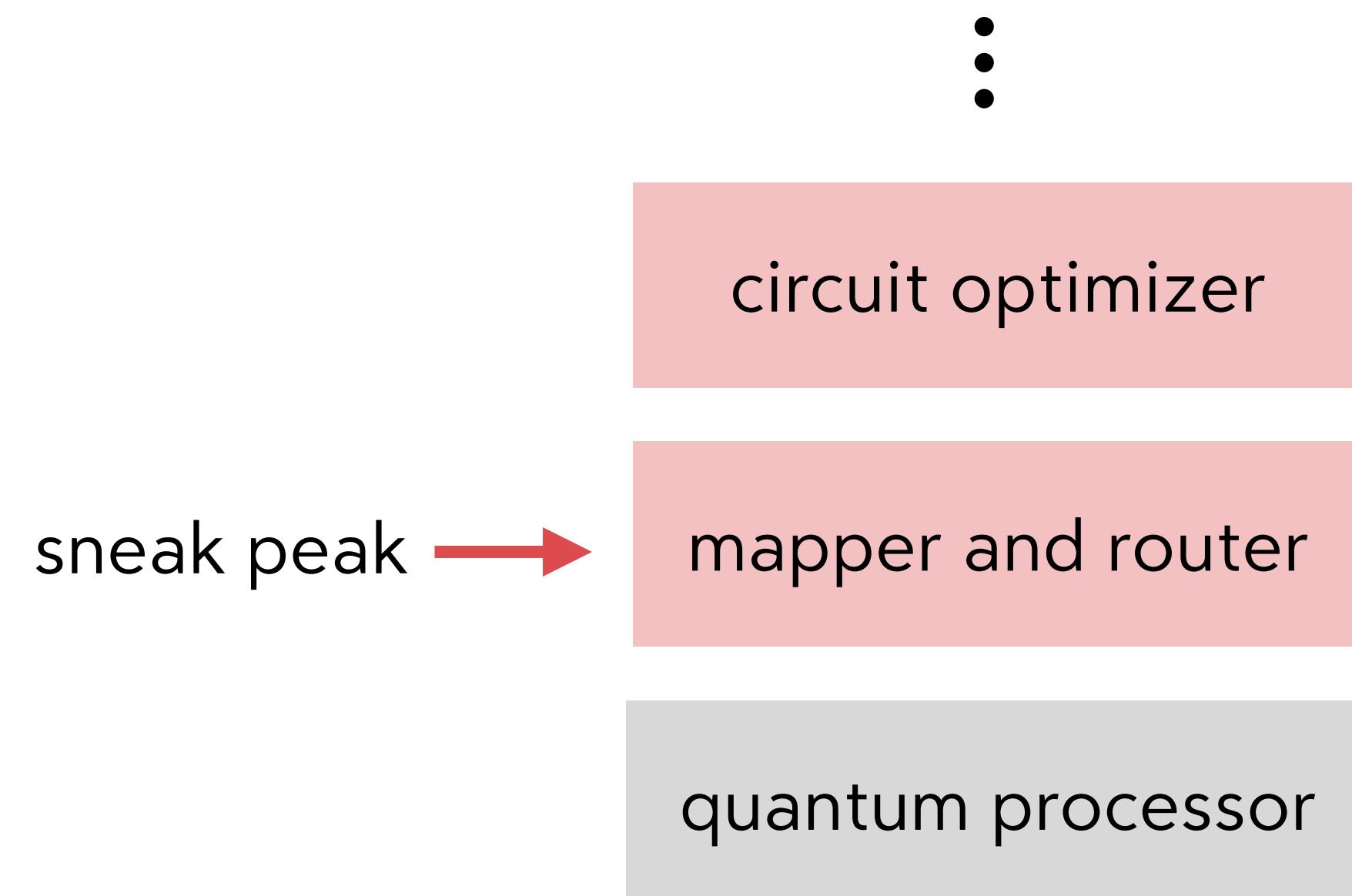
⋮

circuit optimizer

mapper and router

quantum processor

Synthesizing quantum compilers



circuit optimizer

mapper and router

quantum processor





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

Near-term quantum computers are expected to work in an environment where each operation is noisy, with no error correction. Therefore, quantum circuit optimizers are applied to minimize the number of noisy operations. Today, physicists are constantly experimenting with novel devices and architectures. For every new physical substrate and for every modification of a quantum computer, we need to modify or rewrite major pieces of the optimizer to run successful experiments. In this paper, we present GUOO, an efficient approach for automatically synthesizing a quantum-circuit optimizer for a given quantum device. For instance, in 1.2 minutes, GUOO can synthesize an optimizer with high-probability correctness guarantees for IBM computers that significantly outperforms leading compilers, such as IBM's Qiskit and TKET, on the majority (85%) of the circuits in a diverse benchmark suite.

A number of theoretical and algorithmic insights underlie GUOO: (1) An algebraic approach for representing rewrite rules and their semantics. This facilitates reasoning about complex *symbolic* rewrite rules that are beyond the scope of existing techniques. (2) A fast approach for probabilistically verifying equivalence of quantum circuits by reducing the problem to a special form of *polynomial identity testing*. (3) A novel probabilistic data structure, called a *polynomial identity filter* (PIF), for efficiently synthesizing rewrite rules. (4) A beam search-based algorithm that efficiently applies the synthesized symbolic rewrite rules to optimize quantum circuits.

CCS Concepts: • Software and its engineering → Compilers; • Hardware → Quantum computation.

Additional Key Words and Phrases: quantum computing, probabilistic verification

ACM Reference Format:

Amanda Xu, Abtin Molavi, Lauren Pick, Swamit Tannu, and Aws Albargouthi. 2023. Synthesizing Quantum Circuit Optimizers. *Proc. ACM Program. Lang.* 7, PLDI, Article 140 (June 2023), 25 pages. <https://doi.org/10.1145/3591254>

1 INTRODUCTION

The dream of quantum computing has been around for decades, but it is only recently that we have begun to witness promising physical realizations of quantum computers. Quantum computers enable efficient simulation of quantum mechanical phenomena, potentially opening the door to advances in quantum physics, chemistry, material design, and beyond. Near-term quantum computers with several dozens of qubits are expected to operate in a noisy environment without error correction, in a model of computation called *Noisy Intermediate Scale Quantum* (NISQ) computing [Preskill 2018].

Authors' addresses: Amanda Xu, University of Wisconsin-Madison, Madison, WI, USA, axu44@wisc.edu; Abtin Molavi, University of Wisconsin-Madison, Madison, WI, USA, amolavi@wisc.edu; Lauren Pick, University of Wisconsin-Madison, Madison, WI, USA, lpick@wisc.edu; Swamit Tannu, University of Wisconsin-Madison, Madison, WI, USA, stannu@wisc.edu; Aws Albargouthi, University of Wisconsin-Madison, Madison, WI, USA, aws@cs.wisc.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License.
© 2023 Copyright held by the owner/author(s).
2475-1421/2023/5-ART140
<https://doi.org/10.1145/3591254>

140

Optimizing Quantum Circuits, Fast and Slow

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

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

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

Aws Albargouthi
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, GUOO, for optimizing quantum circuits that exploits the synergies of rewriting and resynthesis. Our extensive evaluation demonstrates the ability of GUOO to strongly outperform existing optimizers on a wide range of benchmarks.

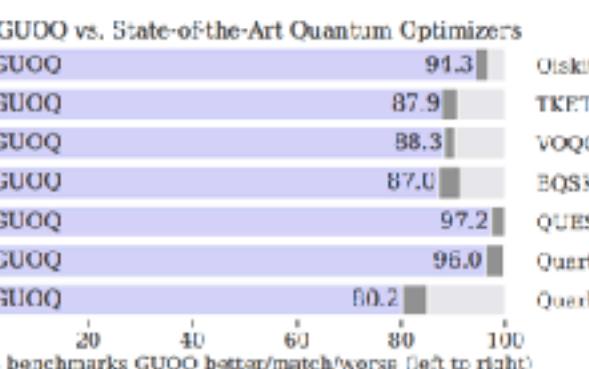


Figure 1. Summary of GUOO compared to state-of-the-art on 2-qubit gate reduction for the TMMQ20 gate set. GUOO and BQSKit are allowed to approximate the circuit up to $\epsilon = 10^{-8}$. *Quartz requires an NVIDIA A100 (40GB) GPU to run.

1 Introduction

Quantum computing enables efficient simulation of quantum mechanical phenomena, promising to catalyze advances in quantum physics, chemistry, materials science, and beyond. Near-term quantum computers with more than a thousand qubits operating in a noisy environment without error correction have already been deployed, marking the current era of *Noisy Intermediate Scale Quantum* (NISQ) computing [48]. Recent groundbreaking experiments have implemented error-corrected logical qubits and demonstrated potential for reducing *logical error* [7, 12]. Although many challenges remain, *fault tolerant quantum computing* (FTQC) is on the horizon.

In both NISQ and FTQC, reducing errors is a critical obstacle to overcome. Every quantum operation has a probability of failure causing a quantum execution to quickly devolve into random noise. The NISQ paradigm aims to mitigate these errors in the absence of error correction primarily by reducing the number of operations. However, error correction in FTQC is not a panacea and introduces its own unique bottlenecks [9, 58], which can render the error correction scheme useless if left untamed. Especially in the near term, FTQC architectures may face challenges in handling large circuit depths due to physical imperfections such as two-level system (TLS) drift, qubit leakage, high-energy particle strikes,

and crosstalk [1, 7, 38]. Therefore, it is of utmost importance to reduce the number of operations for FTQC as well.

Current approaches tackling quantum circuit optimization primarily apply *peephole optimization* using a fixed set of *rewrite rules*. Some tools use a small set of hand-crafted rules [20, 29, 40], while others automatically synthesize rules [66, 67]. The idea is to apply rewrite rules in a schedule, transforming subcircuits to semantically equivalent ones with fewer operations. *Rewrite rules are fast* to apply—match a pattern and rewrite it—but inherently only perform local optimizations.

An orthogonal line of work has been studying the problem of *unitary synthesis*. A unitary matrix precisely represents the semantics of a quantum program. Some quantum algorithms are simple to state in the form of a unitary but nontrivial to decompose into elementary operations that can be executed on hardware [15, 18]. Thus, a large body of work has focused on synthesizing quantum circuits that implement a given unitary matrix [4, 13, 26, 43, 50, 51, 59, 62, 68]. Recent works [44, 65] have applied these algorithms to optimize quantum circuits by partitioning large circuits into manageable-sized *subcircuits* consisting of a few qubits at most and then *resynthesizing* each subcircuit to produce a new subcircuit whose unitary is equivalent, or close enough,

circuit optimizer

mapper and router

quantum processor

-
-
-

Qubit Mapping and Routing via MaxSAT

Abtin Molavi, Amanda Xu, Martin Diges, Lauren Pick, Swamit Tannu, Aws Albarghouthi
 University of Wisconsin-Madison, Madison, WI, USA
 {amolavi, axu44, mdiges, lpick2, stannu, albarghouthi}@wisc.edu

Abstract—Near-term quantum computers will operate in a noisy environment, without error correction. A critical problem for near-term quantum computing is laying out a logical circuit onto physical device with limited connectivity between qubits. This is known as the *qubit mapping and routing* (QMR) problem, an intractable combinatorial problem. It is important to solve QMR as optimally as possible to reduce the amount of added noise, which may render a quantum computation useless. In this paper, we present a novel approach for optimally solving the QMR problem via a reduction to *maximum satisfiability* (MAXSAT). Additionally, we present two novel relaxation ideas that shrink the size of the MAXSAT constraints by exploiting the structure of a quantum circuit. Our thorough empirical evaluation demonstrates: (1) the scalability of our approach compared to state-of-the-art optimal QMR techniques (*solves more than 3x benchmarks with 40x speedup*), (2) the significant cost reduction compared to state-of-the-art heuristic approaches (*an average of ~5x sweep reduction*), and (3) the power of our proposed constraint relaxations.

Index Terms—quantum computing, qubit mapping

I. INTRODUCTION

Quantum computers enable efficient simulation of quantum mechanical phenomena, and therefore open up the door to advances in quantum physics, chemistry, material design, optimization, machine learning, and beyond. Unfortunately, near-term quantum computers face significant reliability challenges as quantum hardware is highly error-prone: quantum bits (qubits) used for computation are sensitive to environmental noise. Furthermore, implementing *quantum error correction* [1] to detect and correct hardware errors requires thousands of physical qubits, and therefore is unlikely to become viable soon. In the meantime, near-term quantum computers with several dozens of qubits are expected to operate in a noisy environment without any error correction using a model of computation called *noisy intermediate-scale quantum* (NISQ) computing [2].

A critical problem in NISQ computing is laying out a logical circuit onto a physical device with limited connectivity between qubits. This is known as the *qubit mapping and routing* (QMR) problem. Specifically, we can only apply two-qubit gates on physically adjacent qubits, so we need to move (route) qubits to physically adjacent locations. Qubit routing is a noisy process that can be detrimental to successful execution. Thus, our goal is to lay out the circuit in such a way that minimizes the required routing.

Solving QMR optimally is known to be NP-hard [3]. Thus, a majority of the proposed techniques have been heuristic in nature, producing suboptimal results [4]. A small number of techniques have been proposed for solving QMR optimally,

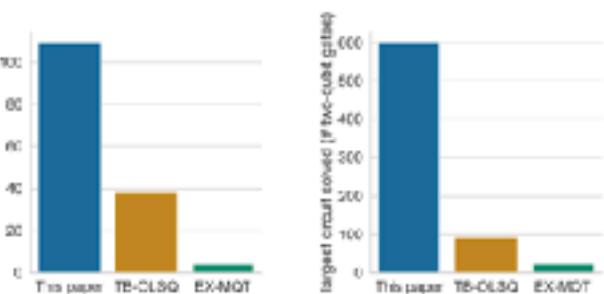


Fig. 1: Comparison against constraint-based tools

mostly by reducing the problem to optimizing an objective function subject to constraints, e.g., *integer linear programming* or *satisfiability modulo theories* [5], [6], [7]. While such *constraint-based* approaches produce optimal results with minimum noise, they have not been scalable to larger circuits.

In this paper, we propose a novel constraint-based approach that significantly advances the state of the art (see Fig. 1). We believe that scaling constraint-based approaches is an important problem for two reasons: (1) With heuristic QMR techniques, one can easily add an unacceptable amount of noise for NISQ computers, producing uninformative outputs. (2) Constraint-based techniques present an optimal baseline with which to evaluate the solution quality of heuristic algorithms, and can therefore help us understand and improve their operation.

QMR as MAXSAT. Our primary insight is that we can reduce the QMR problem to *maximum satisfiability* (MAXSAT) [8, Chapter 19]. MAXSAT is the optimization analogue of the Boolean satisfiability (SAT) problem. While SAT solving is the canonical NP-complete problem, the past two decades have witnessed impressive advances in SAT solving with industrial-grade tools applied at scale (e.g., at Amazon [9], SAT solvers are invoked millions of times daily). MAXSAT solvers are typically simple loops that repeatedly invoke a SAT solver to get better and better solutions. Compared to other approaches that use *satisfiability modulo theories* (SMT) solvers [5], [6], [7], MAXSAT solvers are lighter weight as they do not require complex theory-solver interaction. At a high level, we demonstrate that a MAXSAT approach *can* and *should* be used for solving QMR constraints.

As summarized in Fig. 1, compared to state-of-the-art constraint-based tools [5], [10], our approach can solve significantly more QMR problems (~3x) and scale to larger

Molavi et al., MICRO 2022

Dependency-Aware Compilation for Surface Code Quantum Architectures

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

Practical applications of quantum computing depend on fault-tolerant devices with error correction. Today, the most promising approach is a class of error correcting codes called *surface codes*. We study the problem of compiling quantum circuits for quantum computers implementing surface codes. Optimal or near-optimal compilation is critical for both efficiency and correctness. The compilation problem requires (1) *mapping* circuit qubits to the device qubits and (2) *routing* execution paths between interacting qubits. We solve this problem efficiently and near-optimally with a novel algorithm that exploits the *dependency structure* of circuit operations to formulate discrete optimization problems that can be approximated via *simulated annealing*, a classic and simple algorithm. Our extensive evaluation shows that our approach is powerful and flexible for compiling realistic workloads.

1 Introduction

Quantum computation promises to surpass classical methods in important domains, potentially unlocking breakthroughs in materials science, chemistry, machine learning, and beyond. However, as individual physical qubits and operations are error-prone, these applications require an error-correction scheme for detecting and correcting faults. Quantum error-correction suppresses errors with redundancy: encoding the state of a single logical qubit using several physical qubits. Experimentalists have recently demonstrated error suppression for a single logical qubit [2, 49, 63] and small multi-qubit systems [1, 13, 21, 48].

To harness the full of the fault-tolerant quantum computers on the horizon, we need optimizing compilers that convert circuit-level descriptions of quantum programs to error-corrected elementary operations while preserving as much parallelism as possible. Quantum compute is a scarce resource, so inefficient compilation can be extremely costly. Further, the longer the computation, the higher the probability of logical errors, which affect the result.

Therefore, our goal is to answer the following question:

How can we compile a given circuit for a fault-tolerant device such that execution time is minimized?
 We target a well-studied type of error-correction scheme called a *surface code* [25, 35, 42]. A surface code quantum device embeds logical qubits into a two-dimensional grid of physical qubits. Two-qubit gates impose limitations on the execution of a quantum circuit by introducing contention constraints. Each two-qubit gate occupies a path on the grid and simultaneous paths cannot cross. Gates which can theoretically be executed in parallel may be forced into sequential execution if the path of one “blocks” the other, as shown in Fig. 1. A compiler must carefully *map* qubits to grid locations and *route* two-qubit gates such that such conflicts between gates are minimized and parallelism is maximized. We call this the *surface code mapping and routing* (SCMR) problem.

Existing work on the SCMR problem is limited along two axes: *optimality* and *generality* (see Table 1 for a summary): (1) *optimality*: some techniques do not optimize execution time [59], or optimize routing with respect to a fixed, trivial mapping [10]; (2) *generality*: other techniques

Authors' Contact Information: Abtin Molavi, University of Wisconsin-Madison, Madison, WI, USA, amolavi@wisc.edu; Amanda Xu, University of Wisconsin-Madison, Madison, WI, USA, axu44@wisc.edu; Swamit Tannu, University of Wisconsin-Madison, Madison, WI, USA, stannu@wisc.edu; Aws Albarghouthi, University of Wisconsin-Madison, Madison, WI, USA, aws@cs.wisc.edu.

Molavi et al., OOPSLA 2025

```
pip install wisq
```

```
https://qqq-wisc.github.io/
```