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?

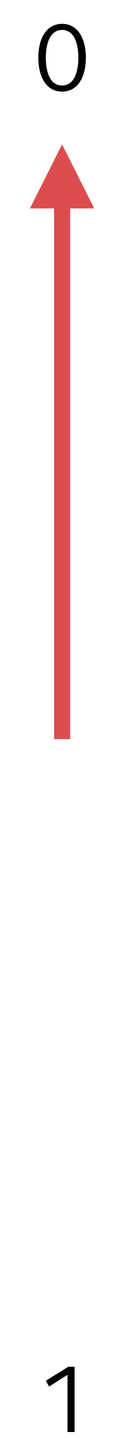
factoring integers efficiently using Shor’s algorithm

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

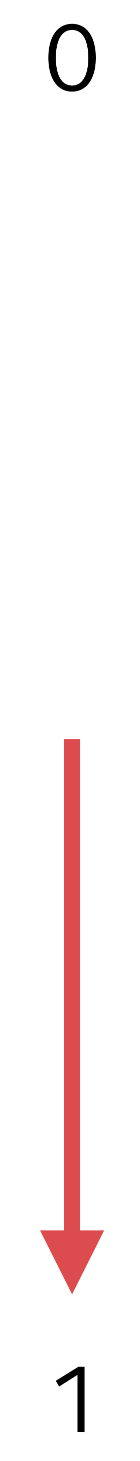
“discover that quantum mechanics was wrong” — Michael Nielsen*

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

Bits vs qubits



Bits vs qubits



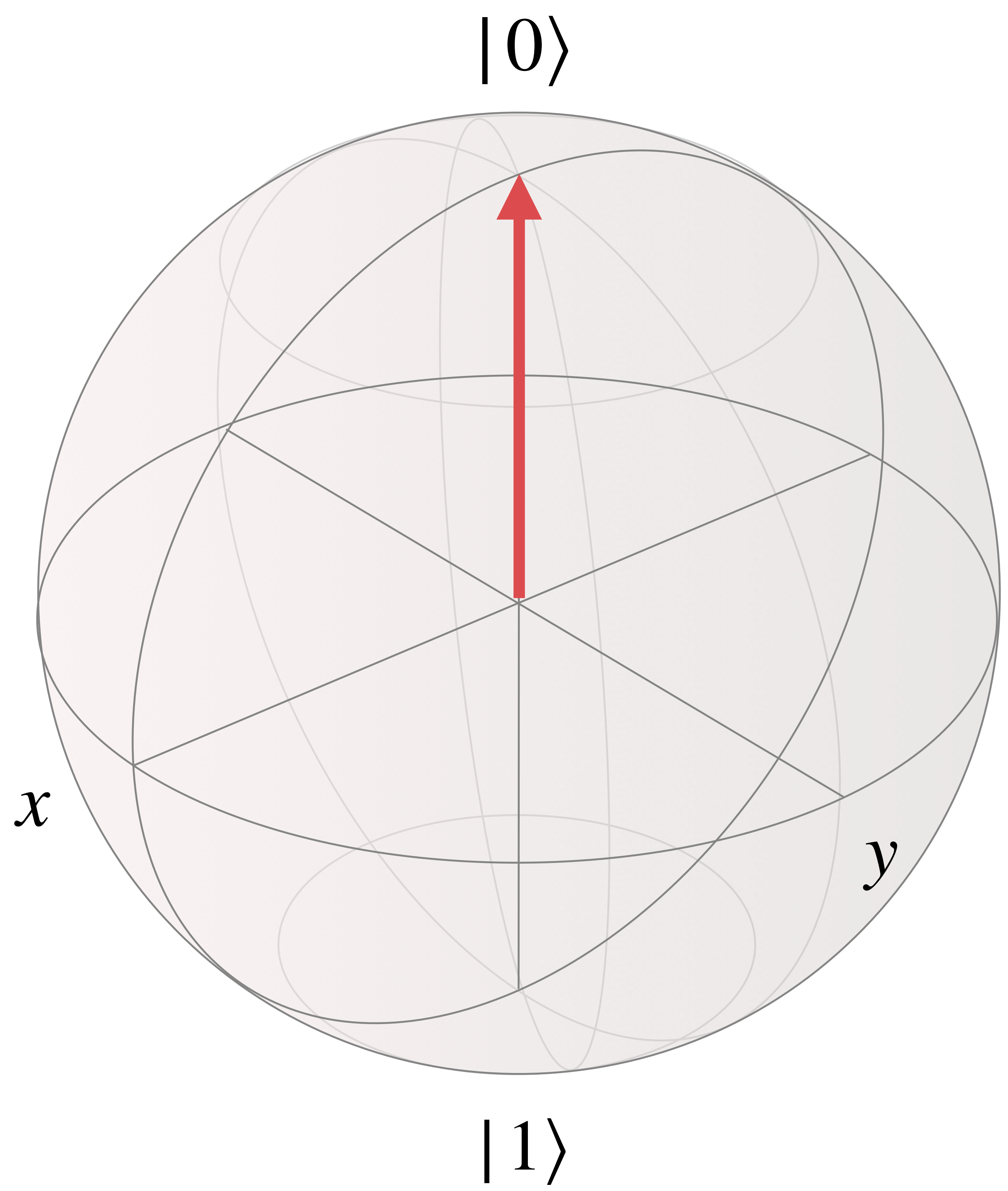
Bits vs qubits

$$|0\rangle$$

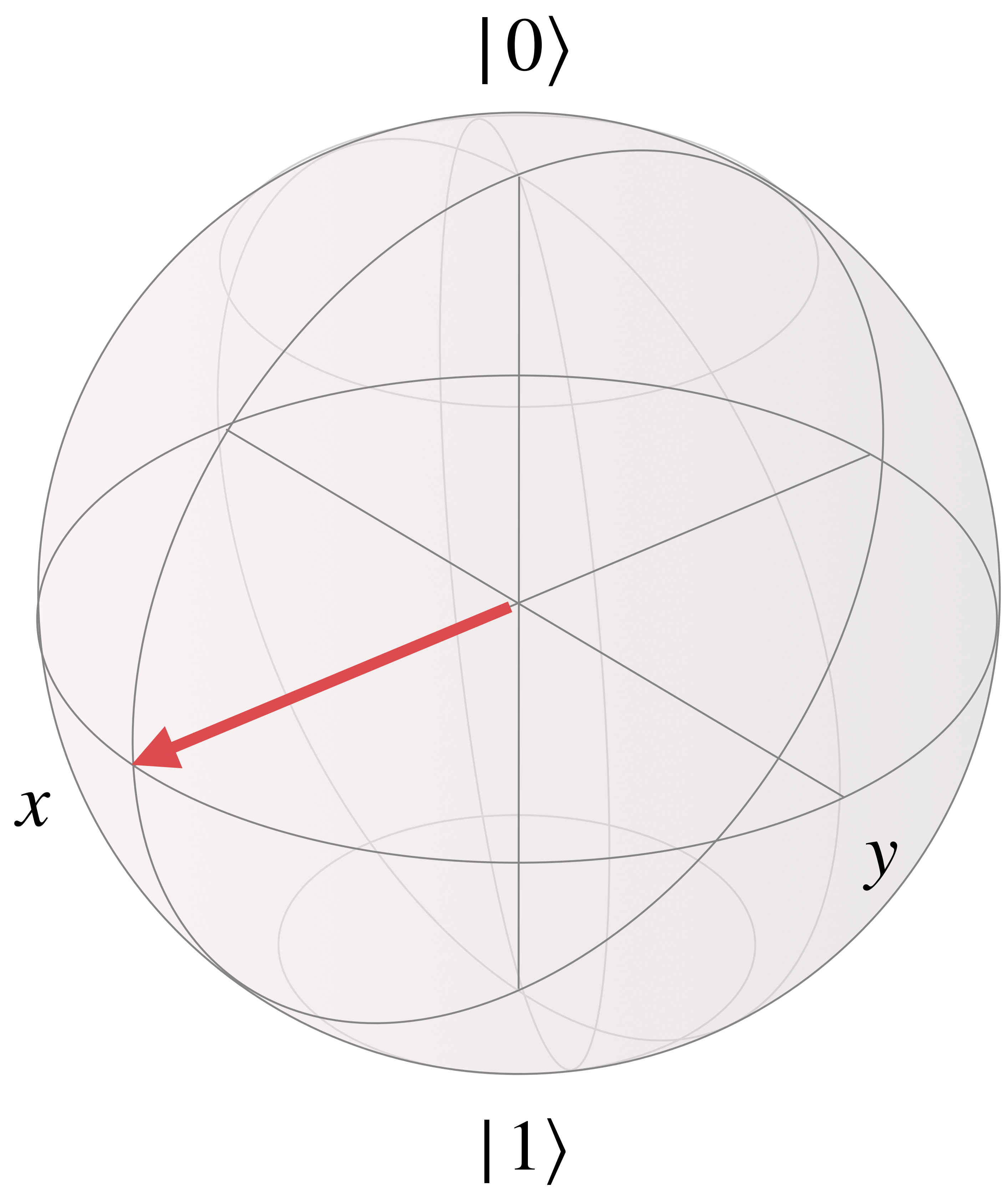
$$|1\rangle$$

Bits vs qubits

Bits vs qubits



Bits vs qubits



Binary vs quantum operations

Classical *X* gate

$$0 \to 1$$

$$1 \to 0$$

* Amy, QPL 2018

Binary vs quantum operations

Classical X gate

$0 \to 1$

$1 \to 0$

Quantum X gate

$|0\rangle \to |1\rangle$

$|1\rangle \to |0\rangle$

* Amy, QPL 2018

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*

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

* Amy, QPL 2018

Binary vs quantum operations

Binary vs quantum operations

Pathsum notation

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

Binary vs quantum operations

Pathsum notation

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

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

Binary vs quantum operations

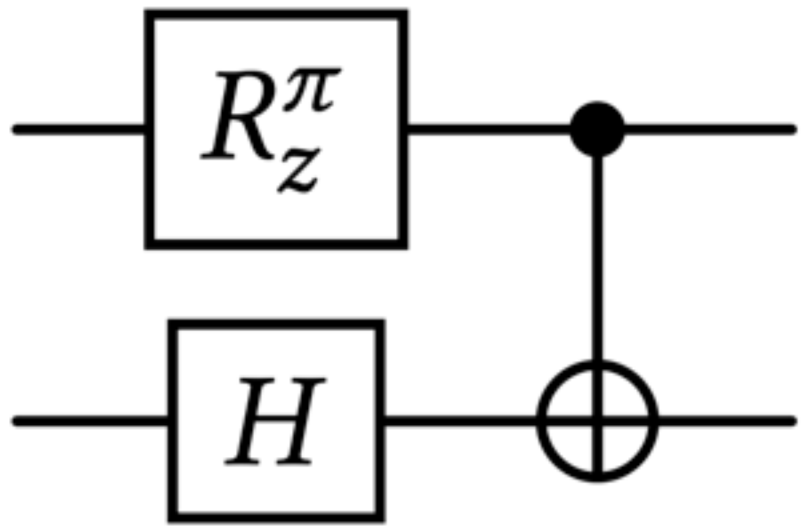
Pathsum notation

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

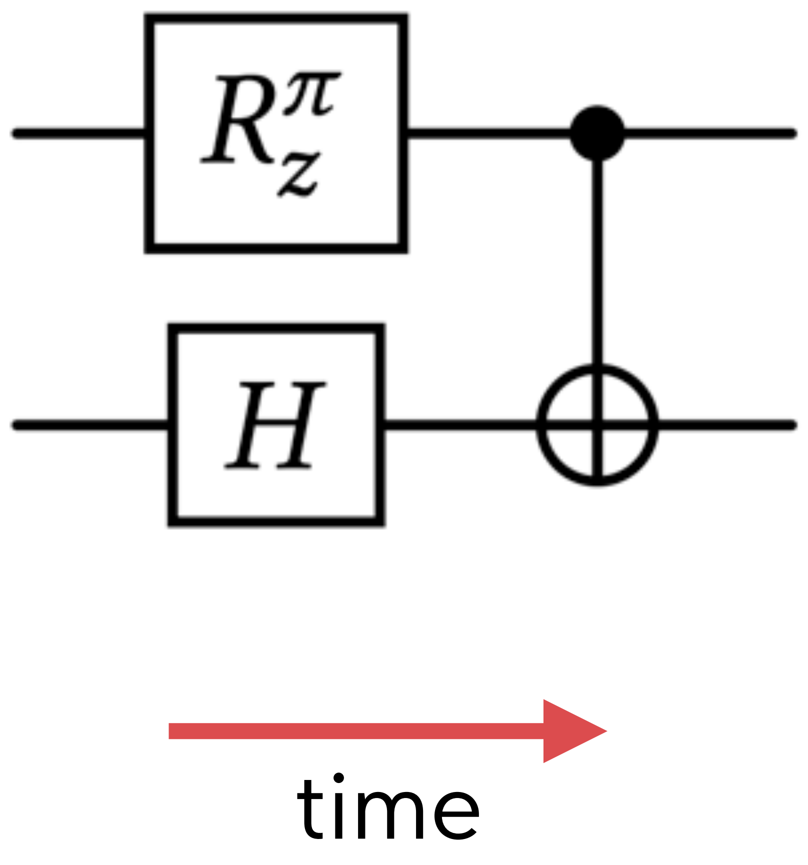
$$R_z(\theta) : |x\rangle \to e^{i(2x-1)\theta} |x\rangle$$

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

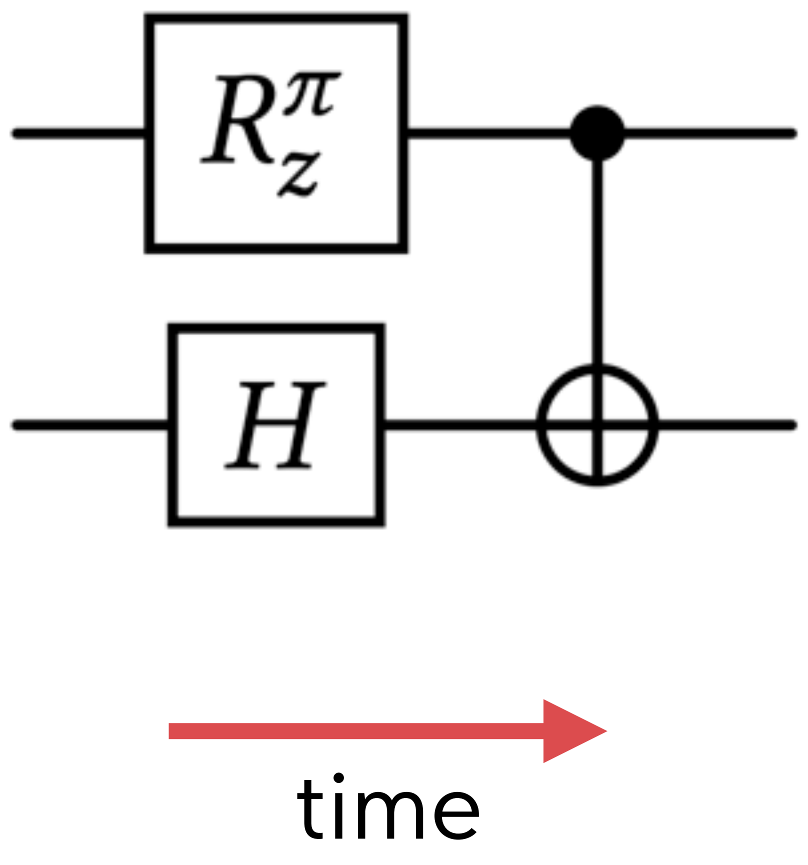
Quantum circuits/programs



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?

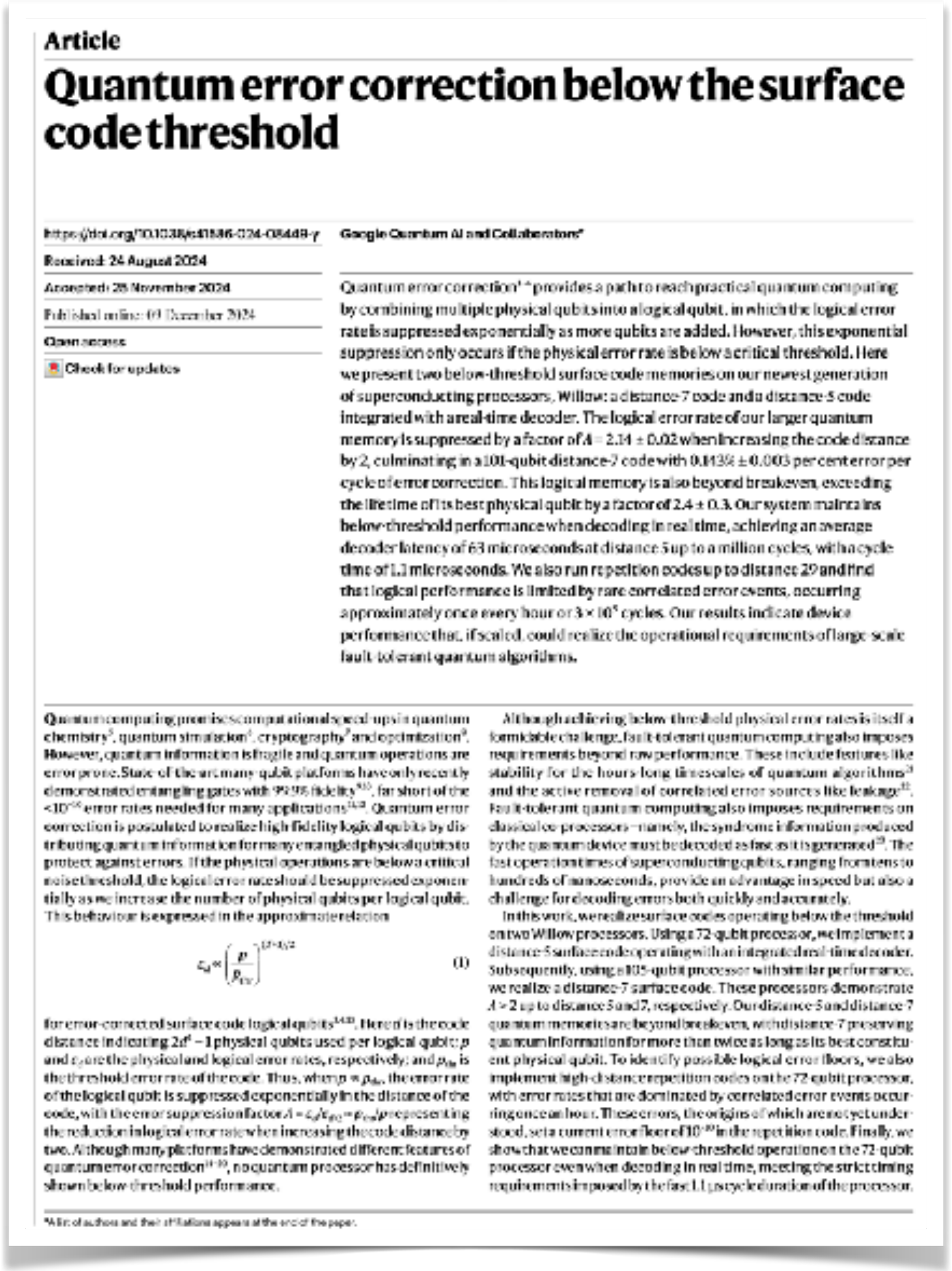
FTQC can we do error correction?

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/s41586-024-08449-y

Received: 24 August 2024

Accepted: 28 November 2024

Published online: 09 December 2024

Open access

Check for updates

Google Quantum AI and Collaborators*

Quantum error correction[1-4] 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 Λ = 2.14 ± 0.02 when increasing the code distance by 2, culminating in a 101-qubit distance-7 code with 0.143% ± 0.003 per cent 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 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 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.

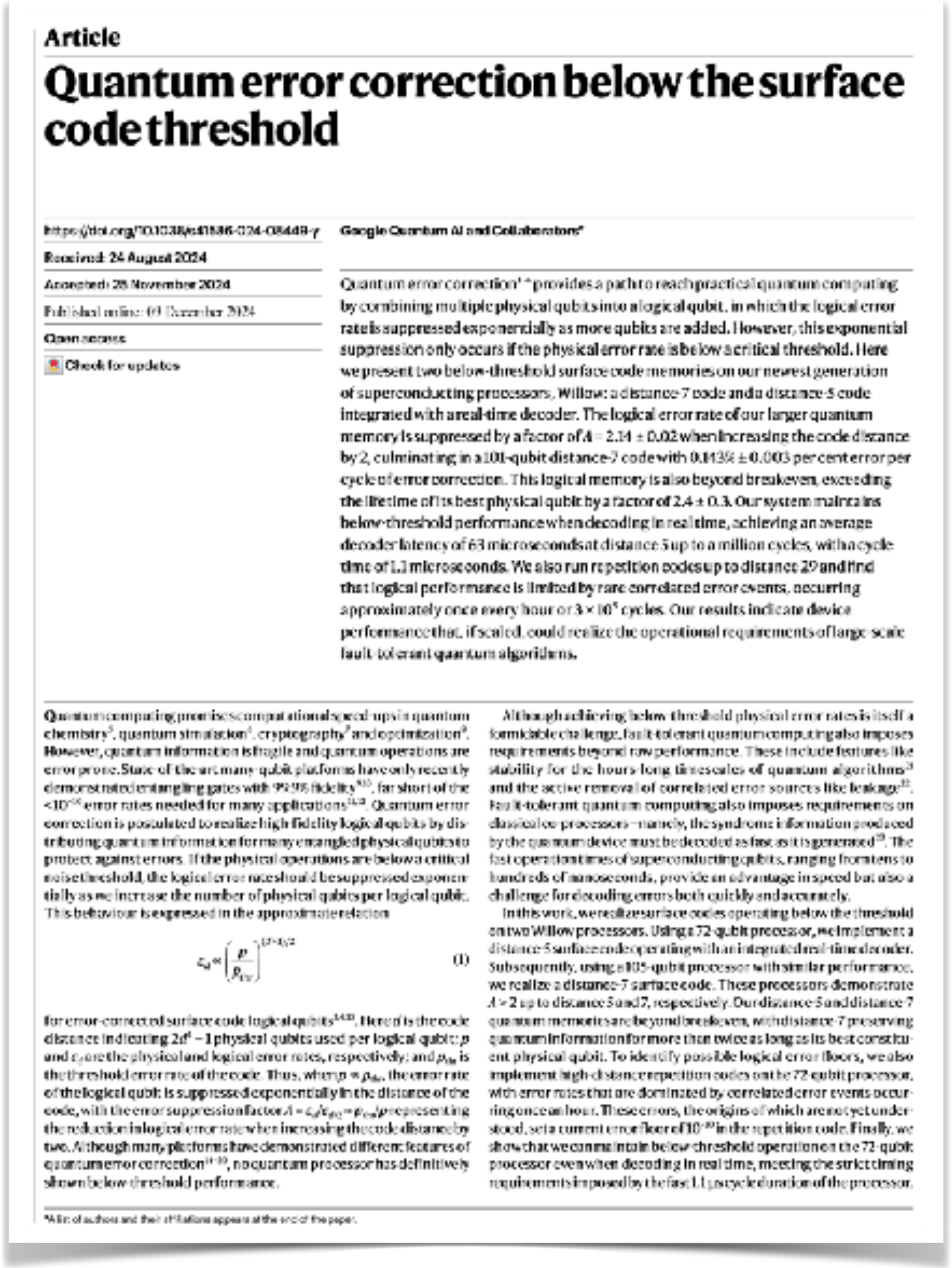
Google Quantum AI et al., Nature 2024

The quantum landscape

qubits are unreliable, noisy

NISQ what can we do with noisy qubits?

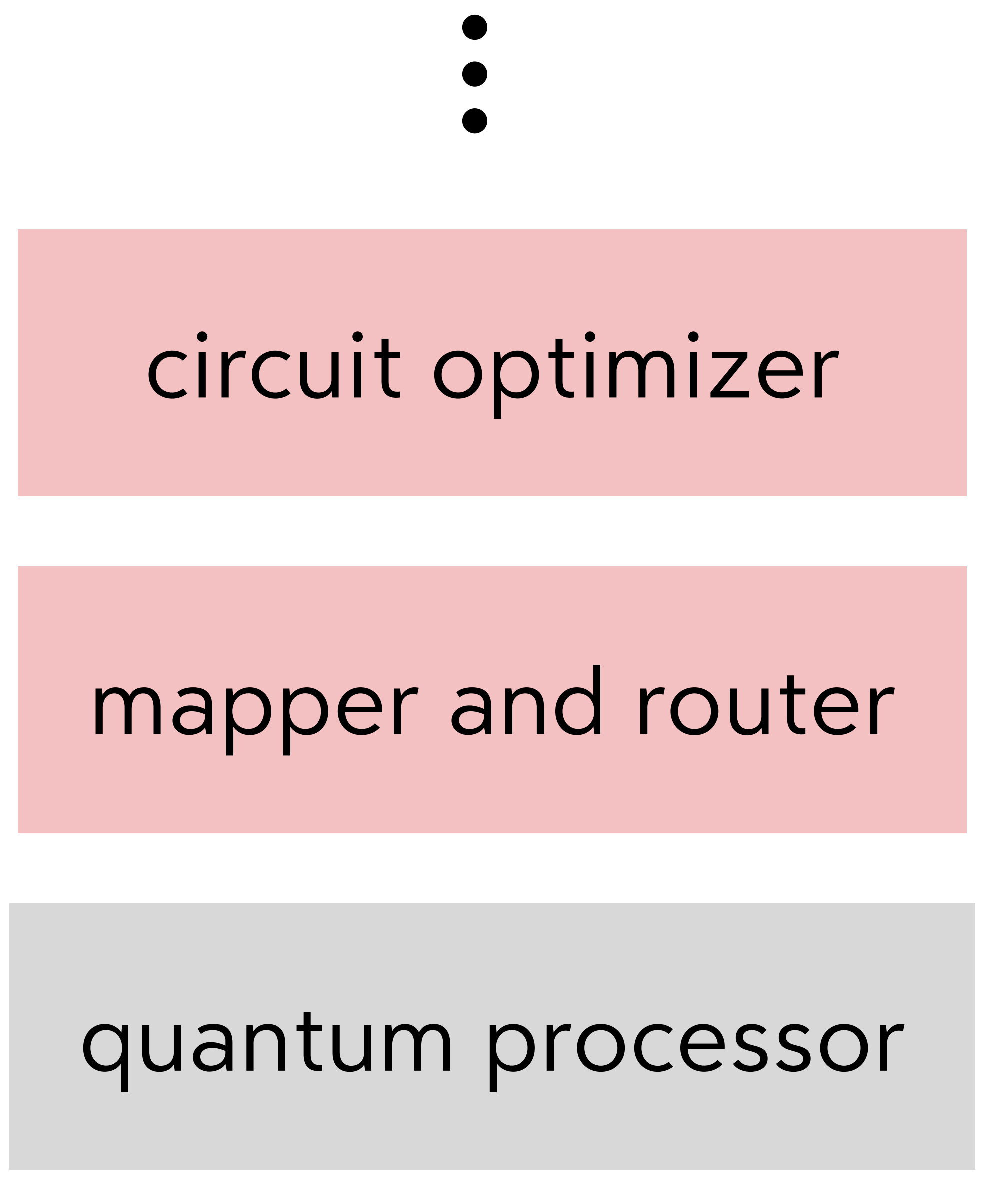
FTQC can we do error correction?



Google Quantum AI et al., Nature 2024

AWS et al., Nature 2025

Anatomy of a quantum compiler



Anatomy of a quantum compiler

⋮

circuit optimizer

mapper and router

quantum processor

Ion Trap

IONQ

QUANTINUUM

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

rigetti · OQC · Google Quantum AI · IBM

Ion Trap

IONQ · QUANTINUUM

Photonic

XANADU

Anatomy of a quantum compiler

⋮

circuit optimizer

mapper and router

quantum processor

Superconducting

rigetti · OQC · Google Quantum AI · IBM

Ion Trap

IONQ · QUANTINUUM

Neutral Atom

|QuEra> Computing Inc.

Photonic

XANADU

Anatomy of a quantum compiler

⋮

circuit optimizer

mapper and router

quantum processor

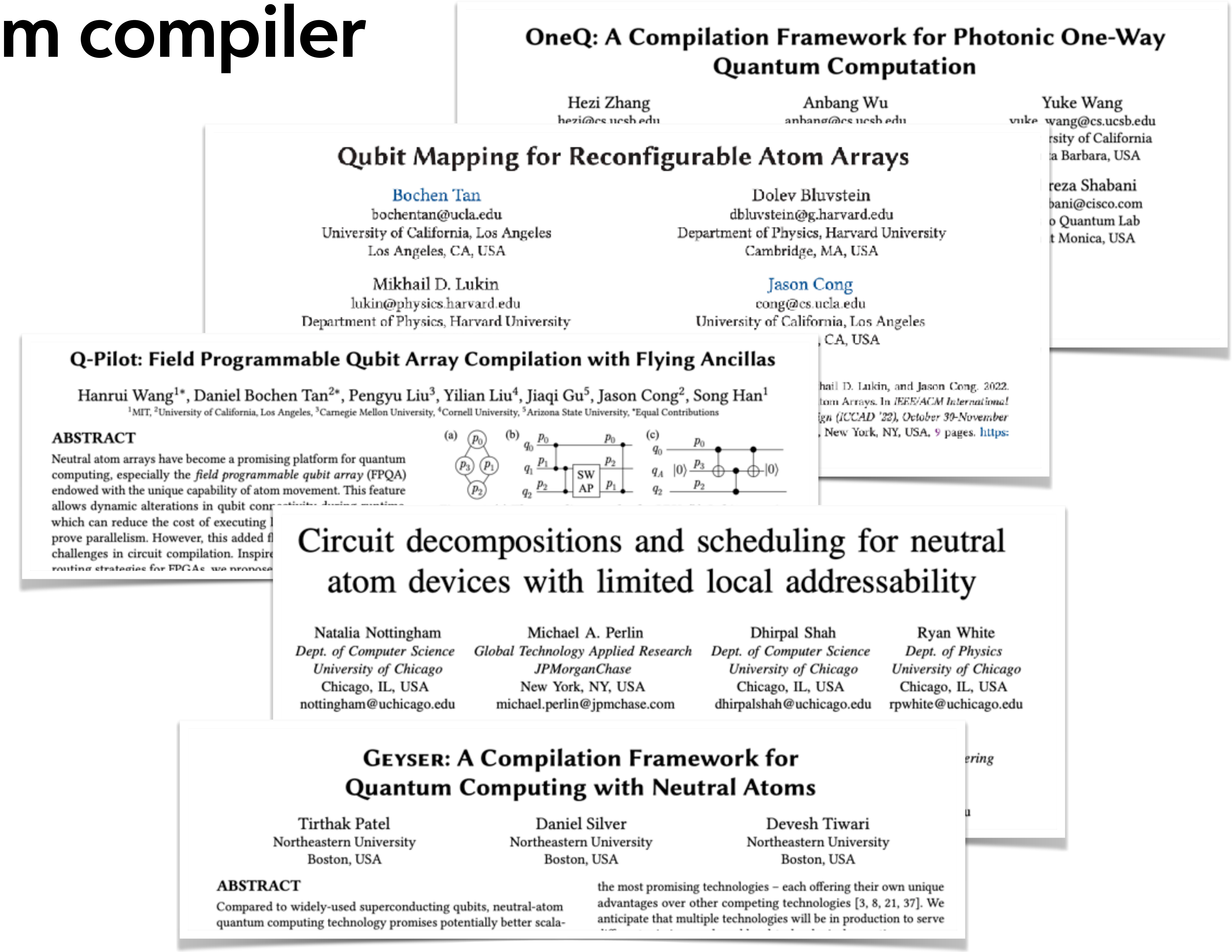
Anatomy of a quantum compiler

⋮

circuit optimizer

mapper and router

quantum processor


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

Hezi Zhang, Anbang Wu, Yuke Wang

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

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

Hanrui Wang[1*], Daniel Bochen Tan[2*], Pengyu Liu[3], Yilian Liu[4], Jiaqi Gu[5], Jason Cong[2], Song Han[1]

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 conn…

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, rpwhite@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

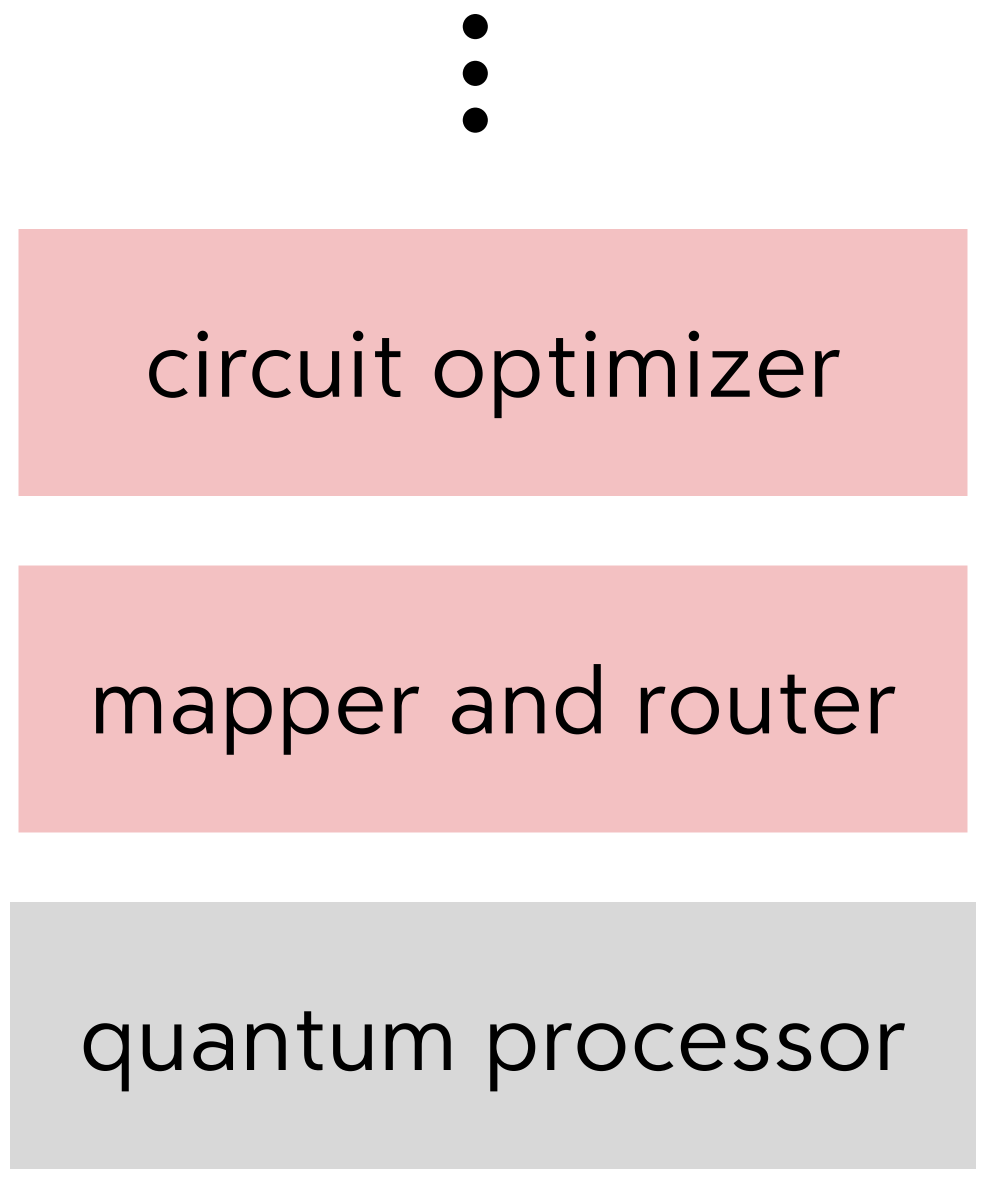
We need to synthesize quantum compilers

huge diversity in qubits, architectures, fault-tolerance schemes

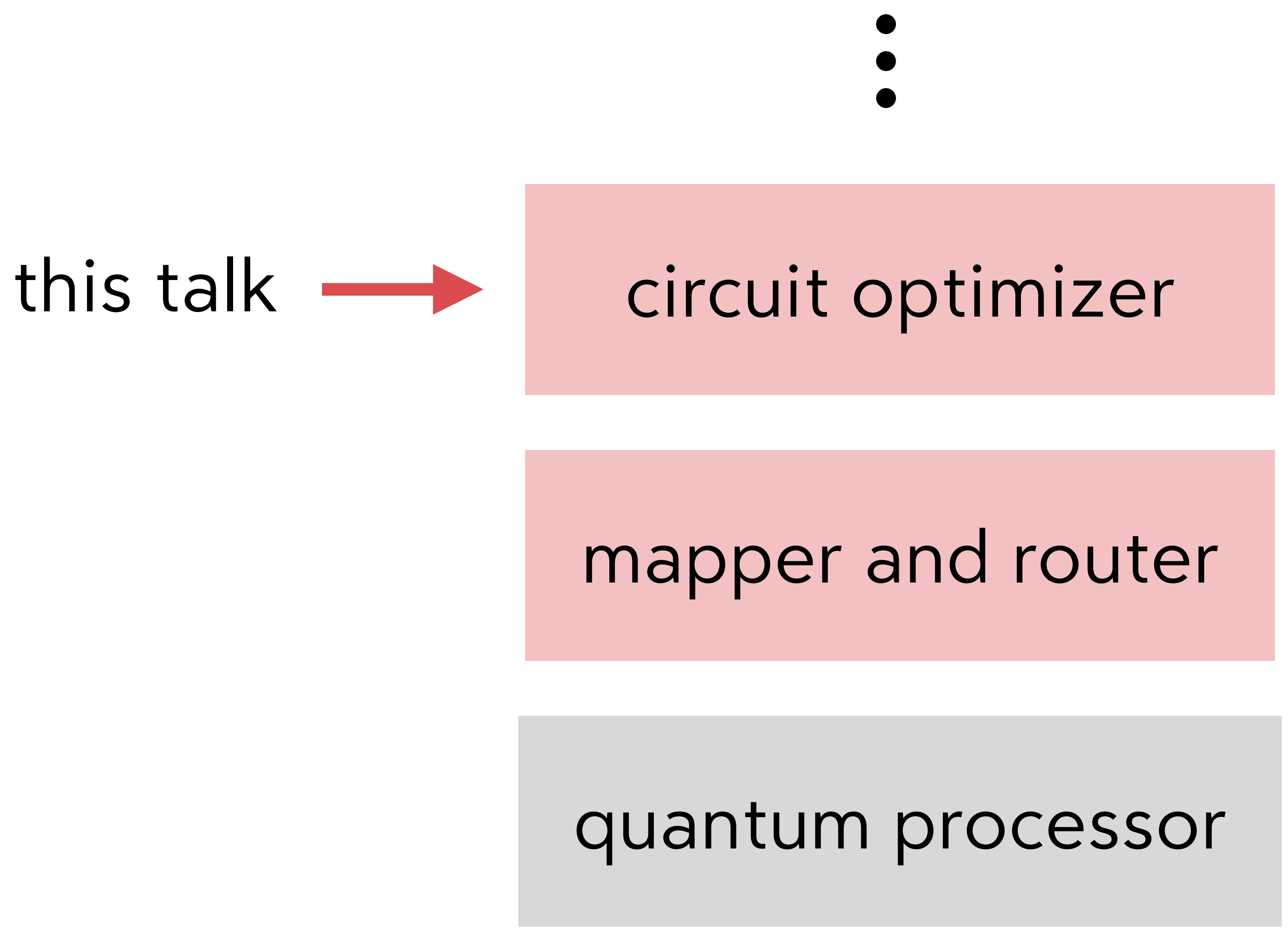
(quantum) compilers are hard to get right*

* Paltenghi & Pradel, OOPSLA 2022

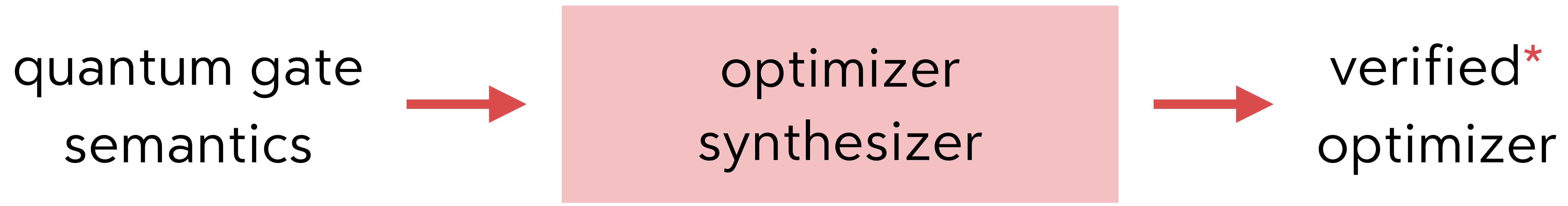
Synthesizing quantum compilers



Synthesizing quantum compilers

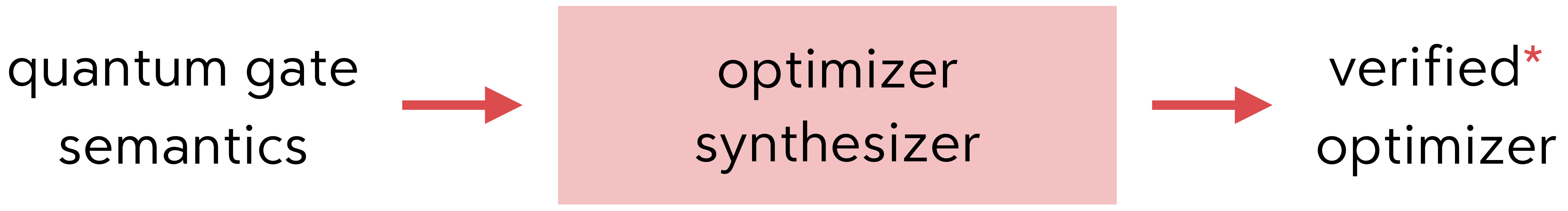


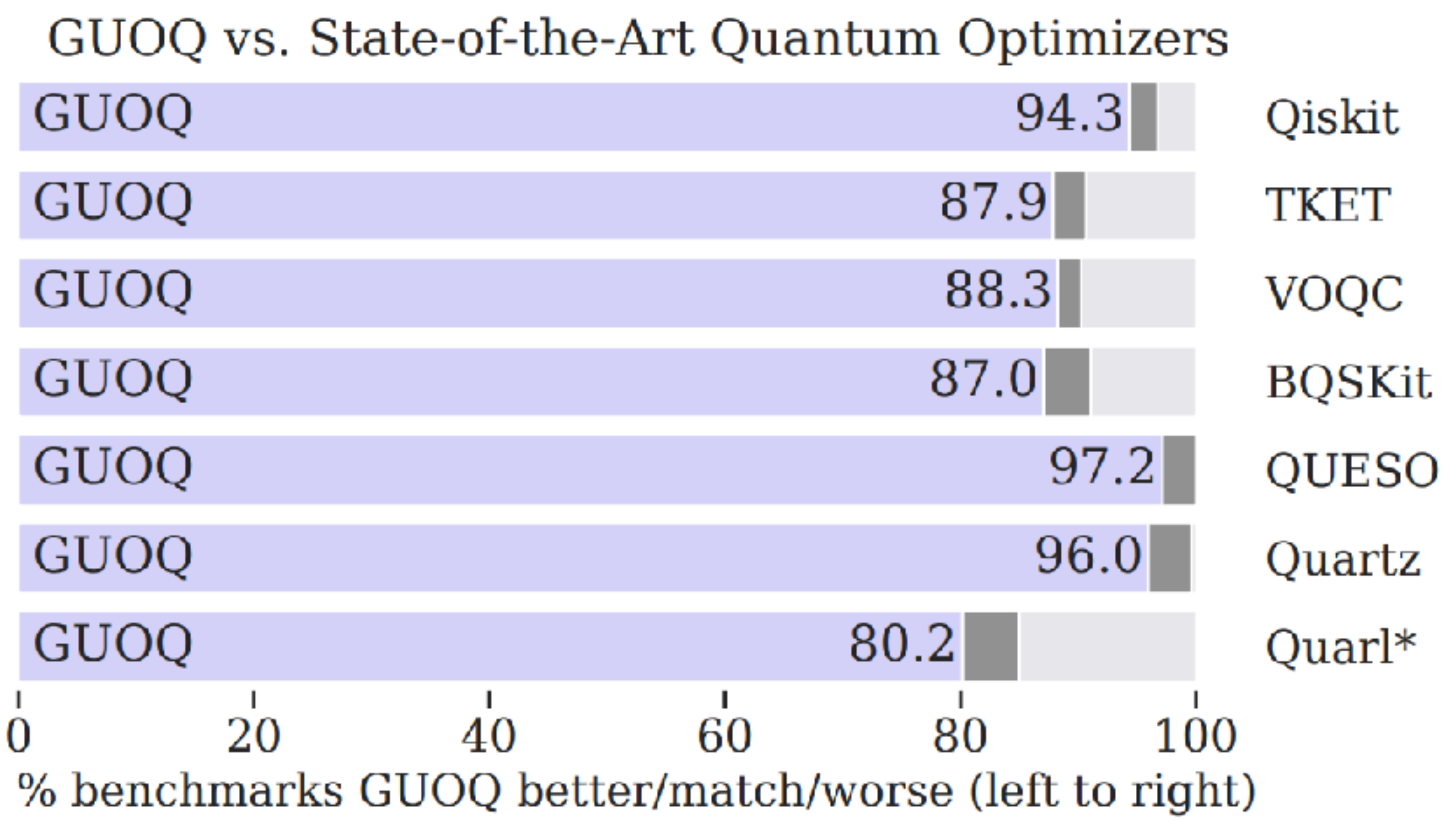
Synthesizing circuit optimizers



* not everything will be verified

Synthesizing circuit optimizers





Synthesizing circuit optimizers

A learning rewrite rules

B optimizing, fast and slow

Synthesizing circuit optimizers

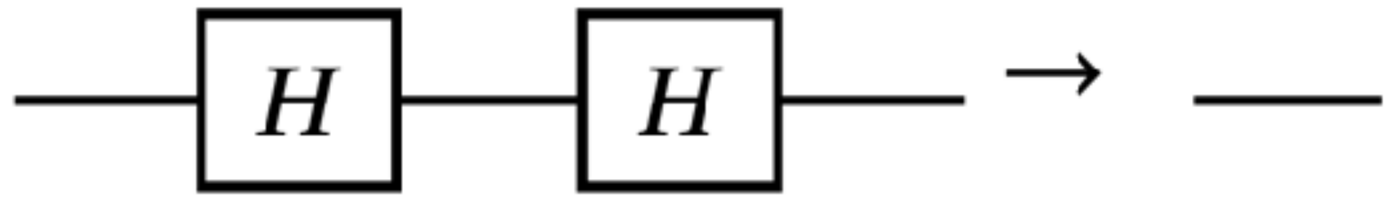
A learning rewrite rules

B optimizing, fast and slow

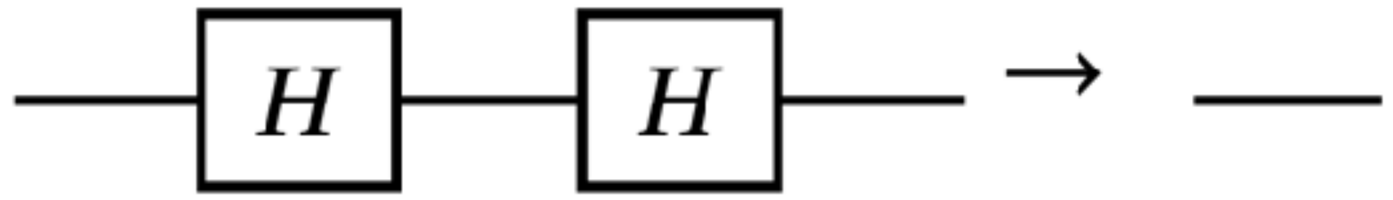
theme simple, classic algorithms go a long way

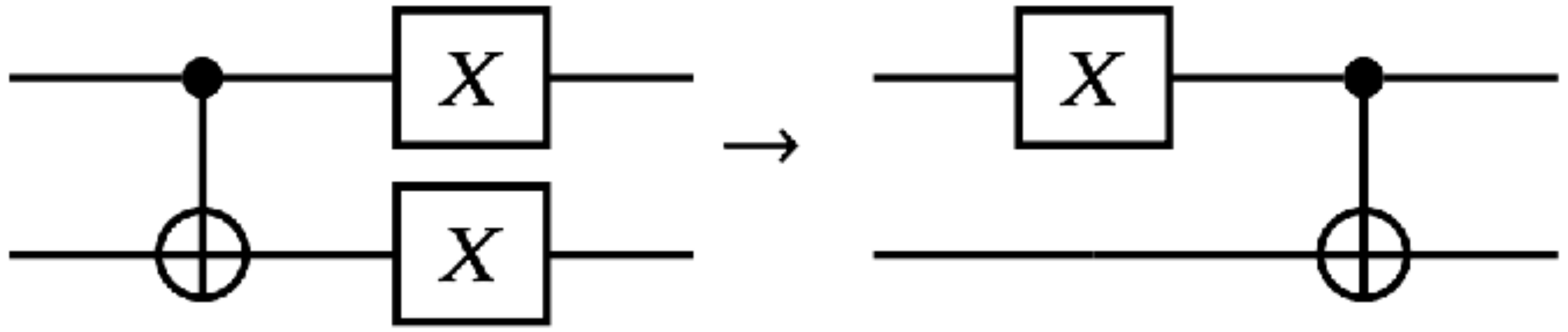
What is a rewrite rule?

What is a rewrite rule?

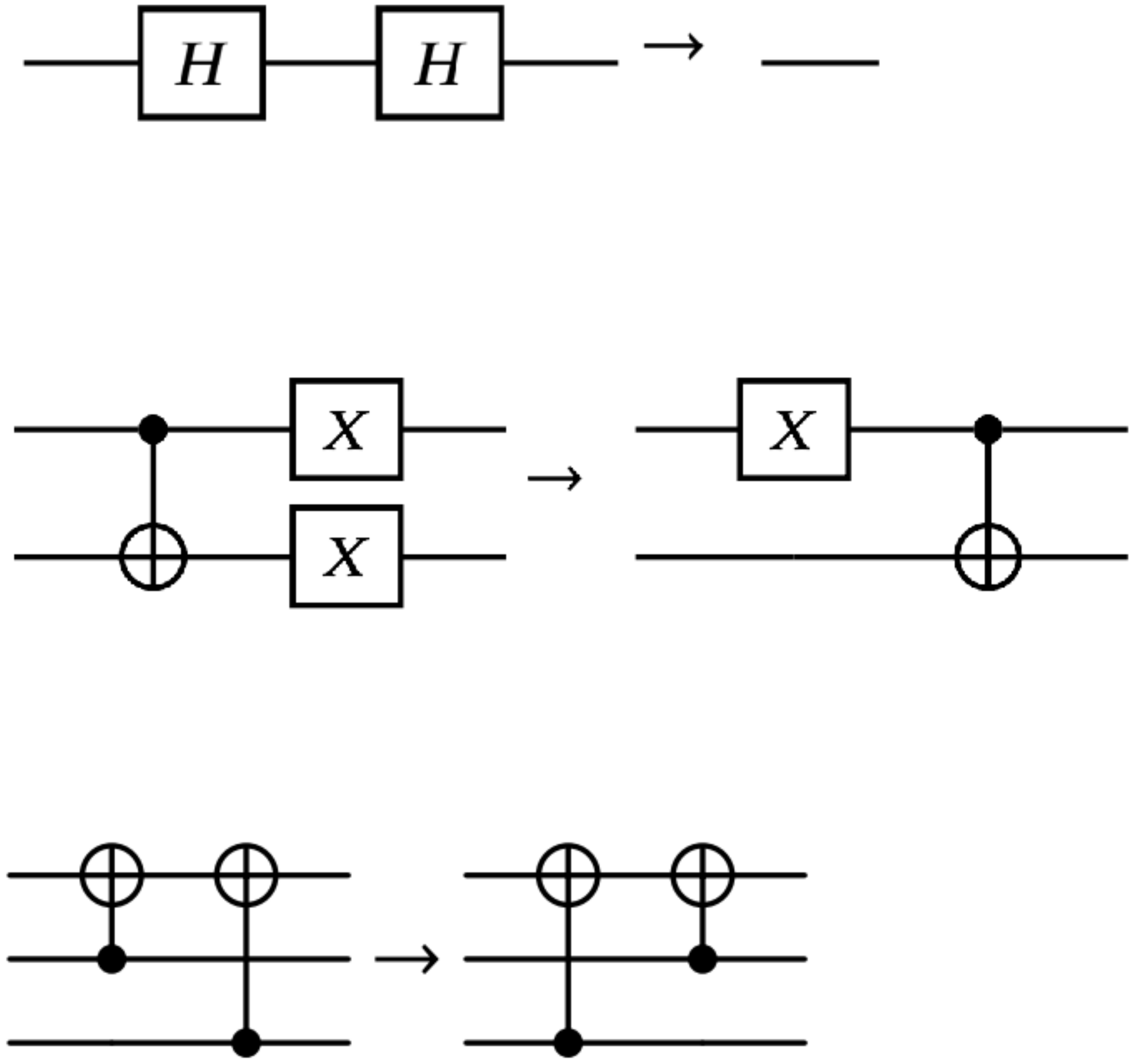


What is a rewrite rule?



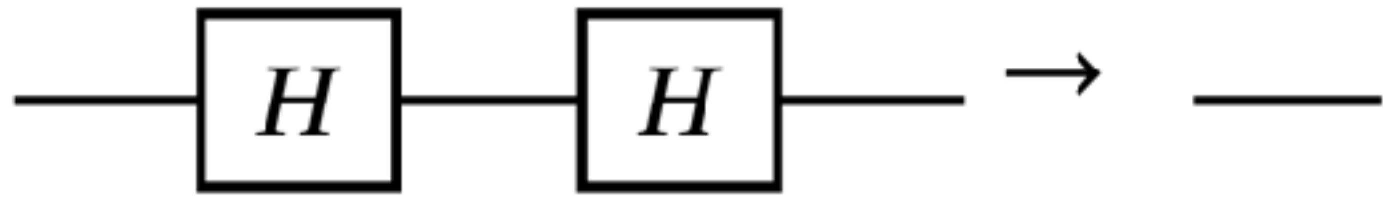


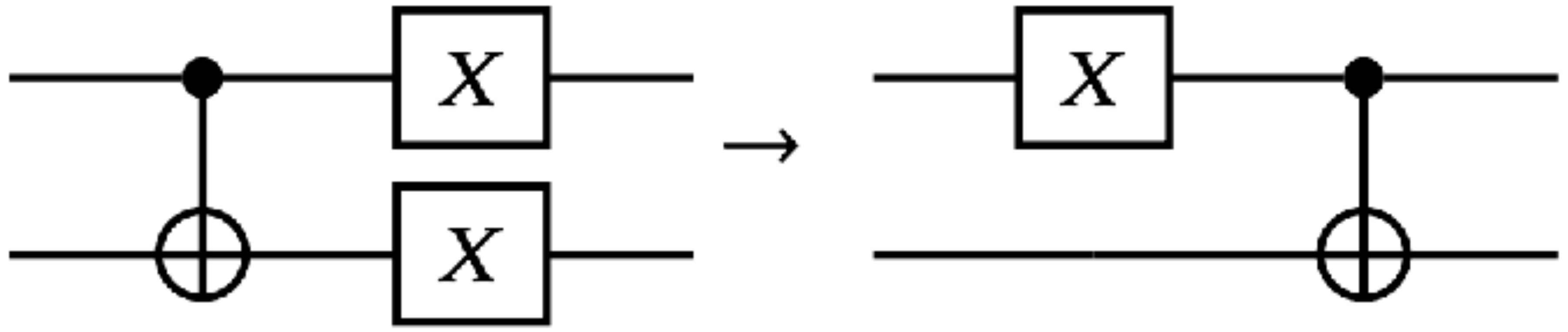
What is a rewrite rule?

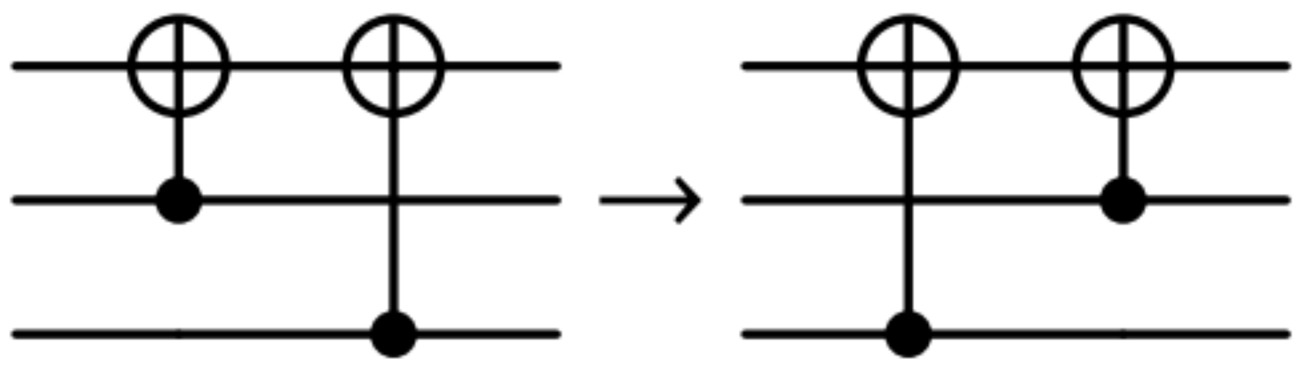


What is a rewrite rule?

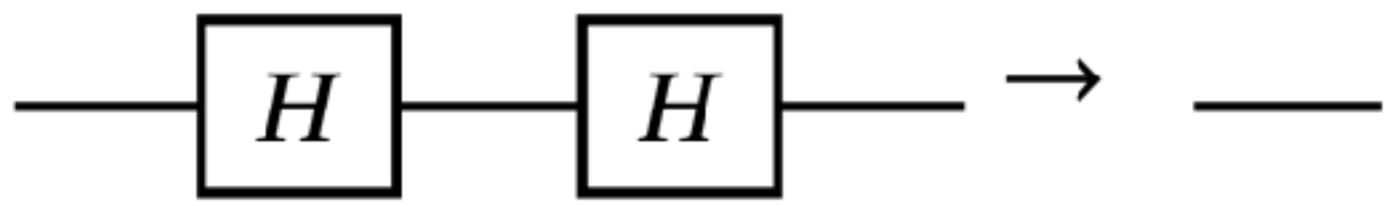
Symbolic rewrite rules

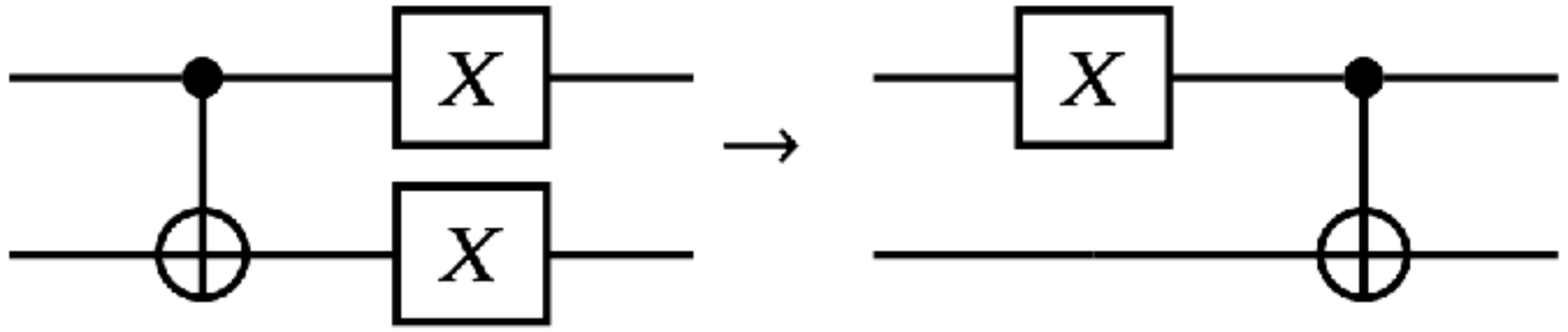


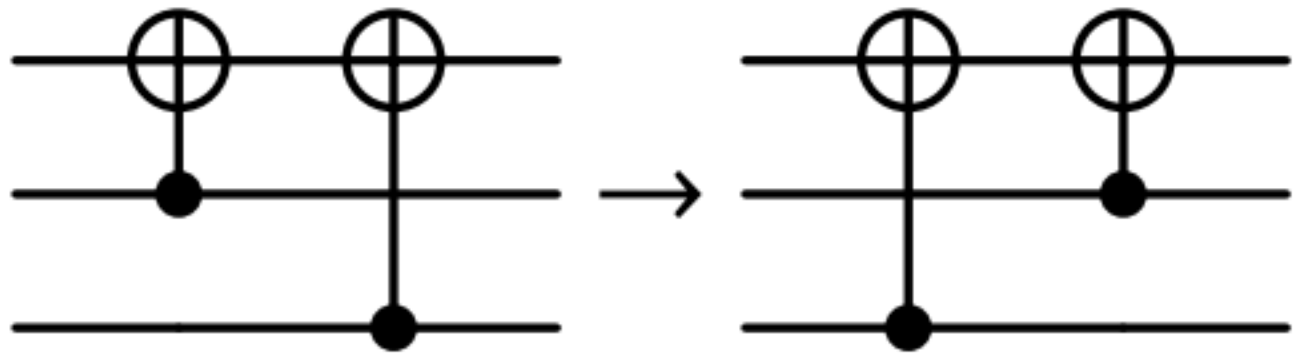




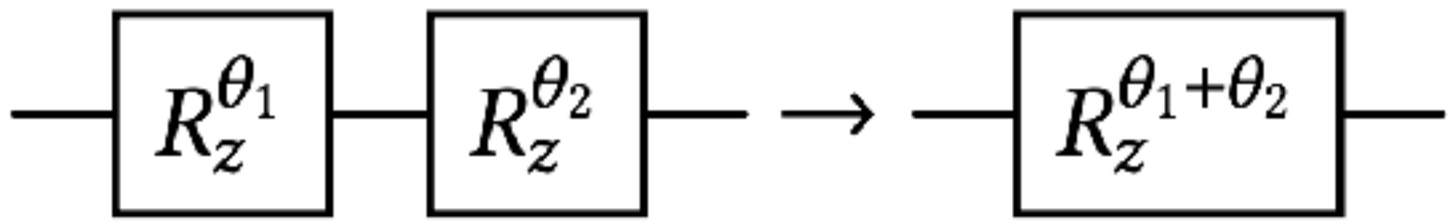
What is a rewrite rule?



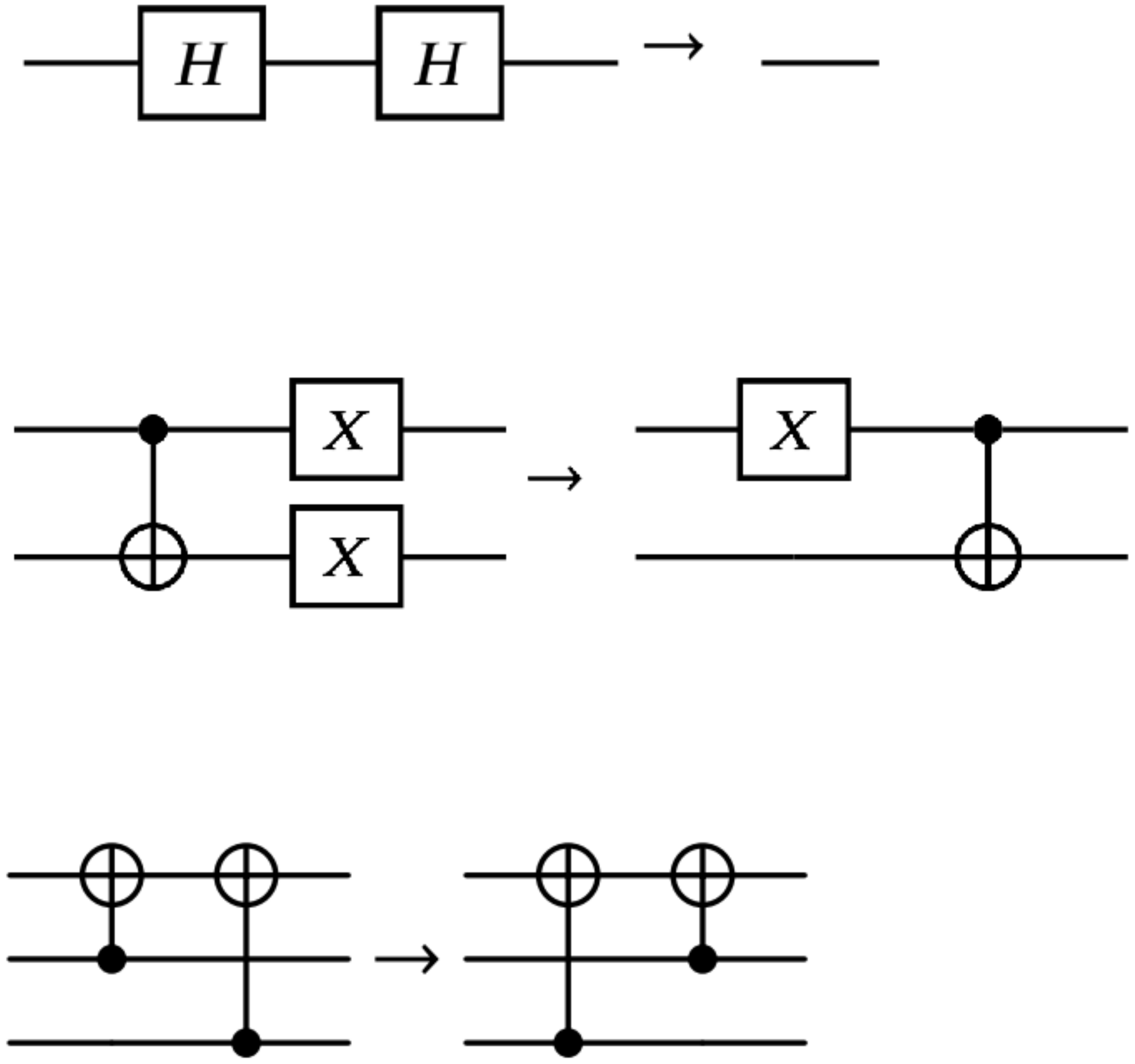




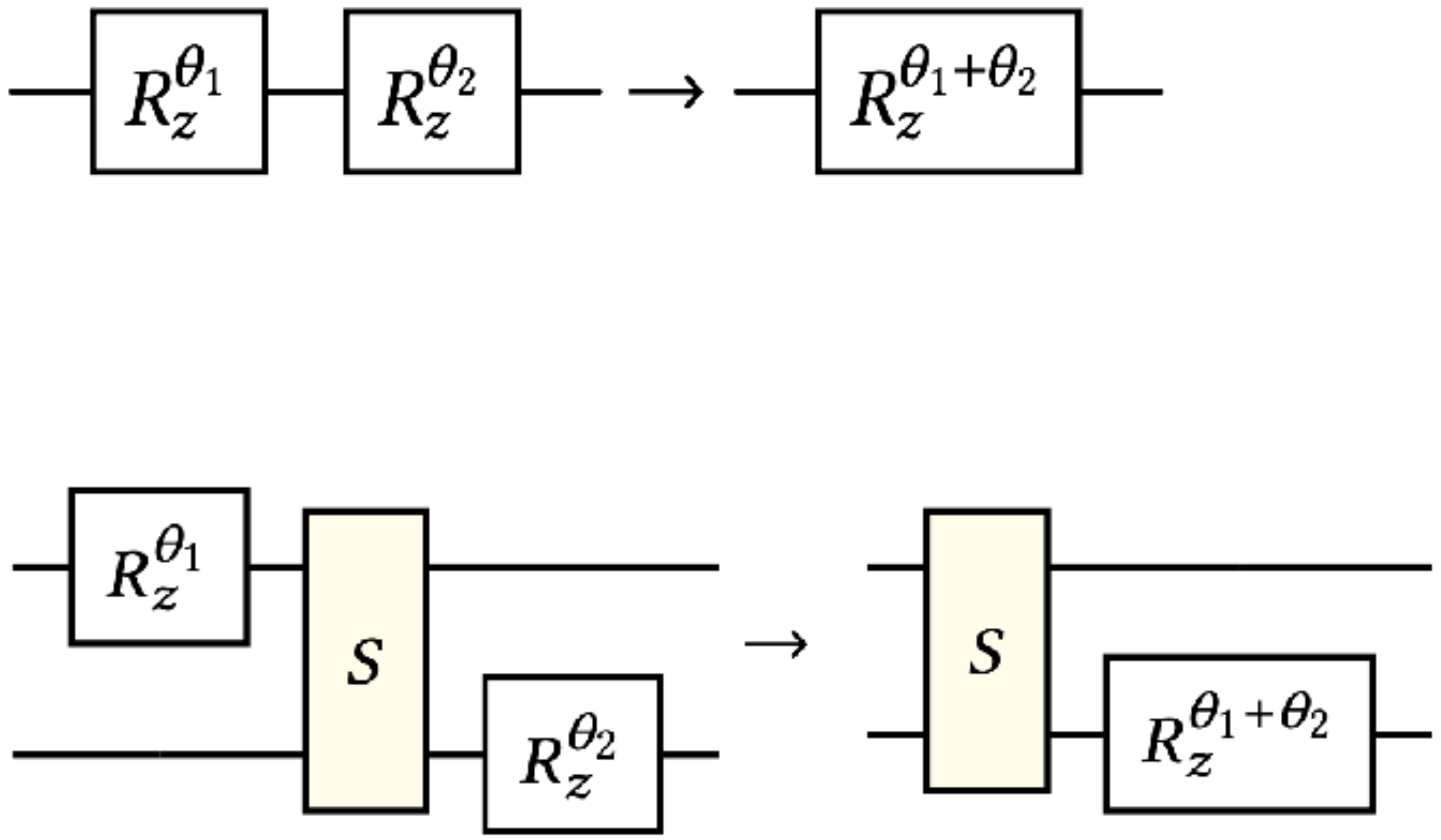
Symbolic rewrite rules



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

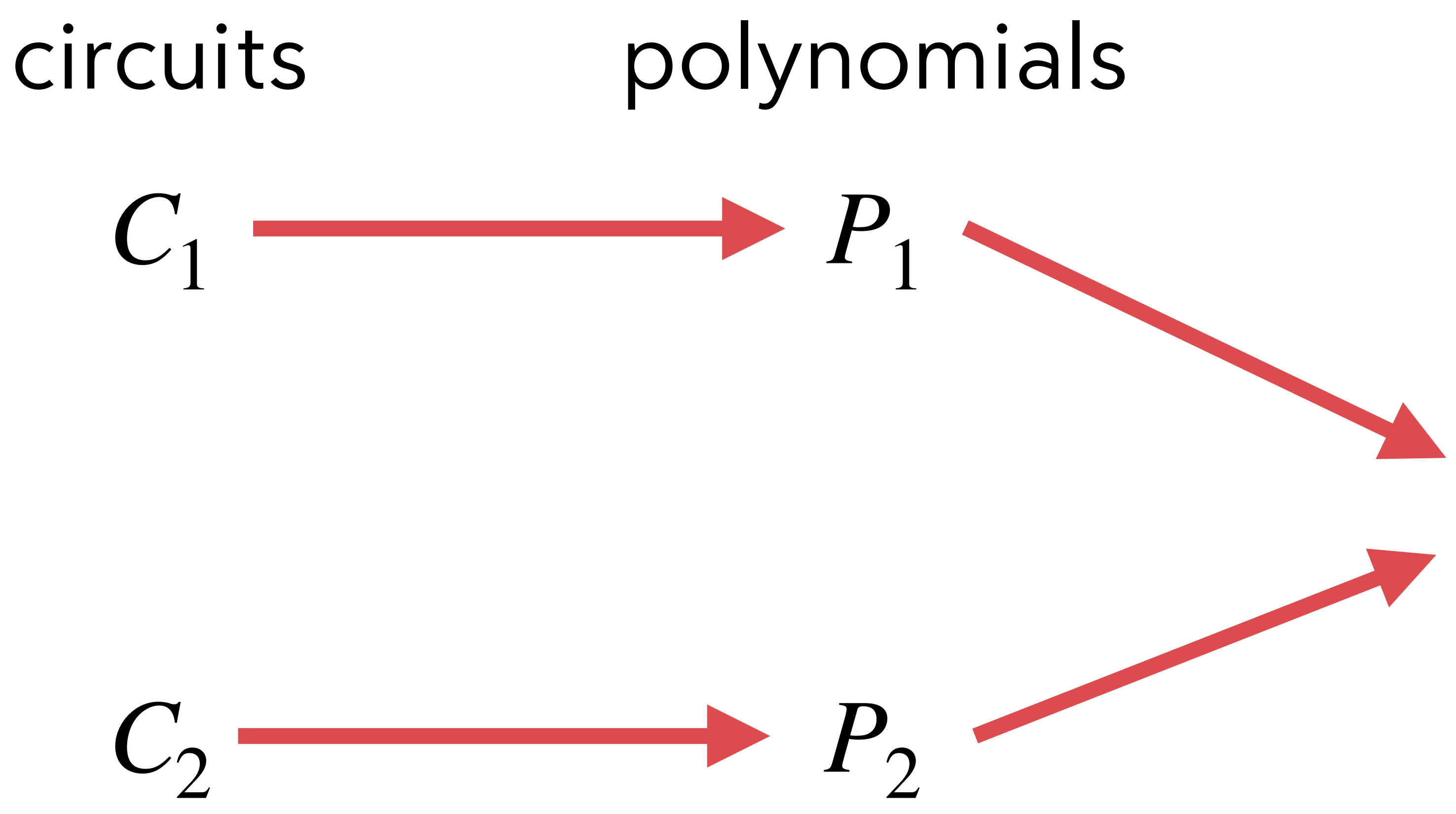
Circuit equivalence

circuits polynomials

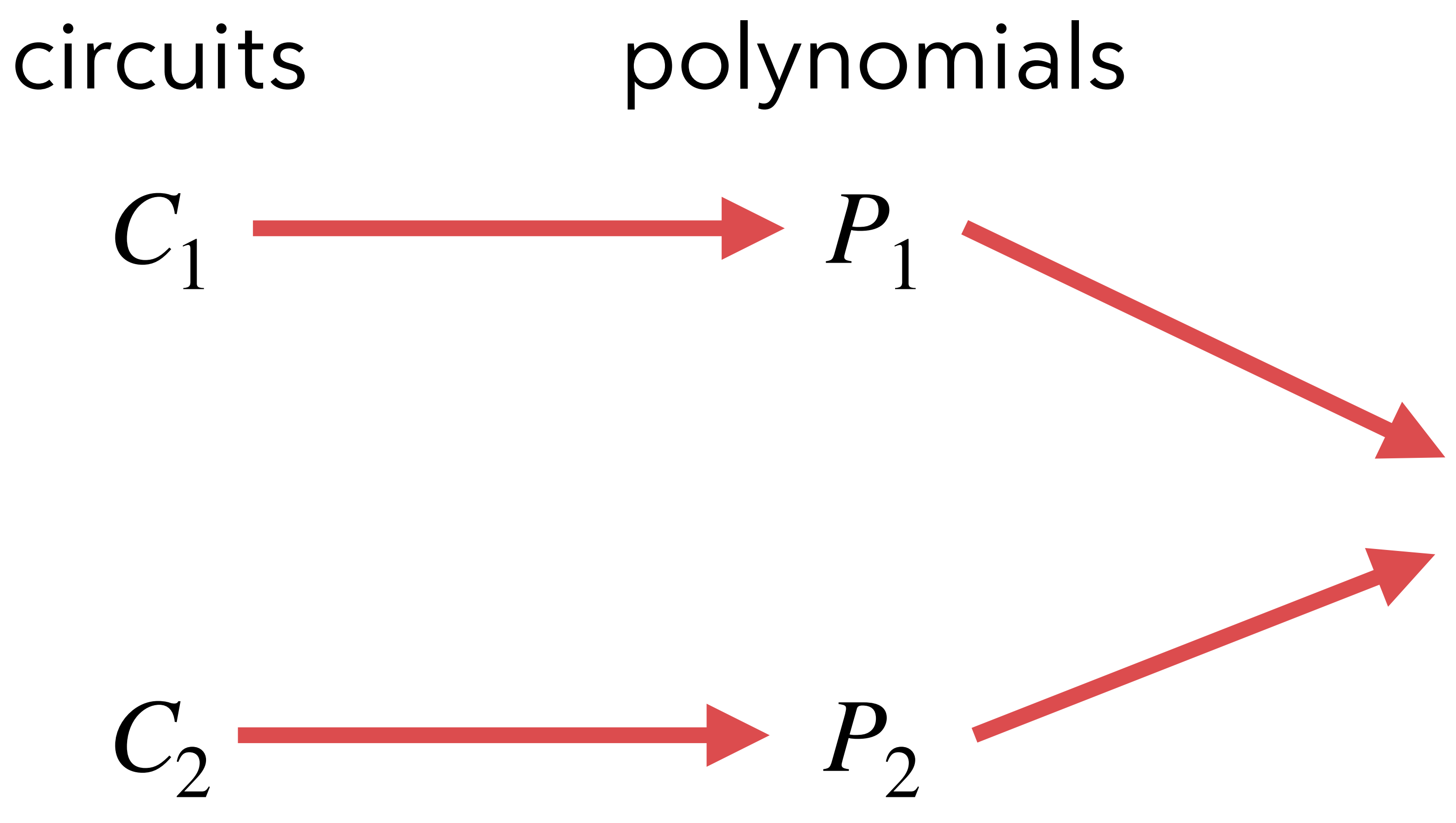
$C_1 \longrightarrow P_1$

$C_2 \longrightarrow P_2$

Circuit equivalence



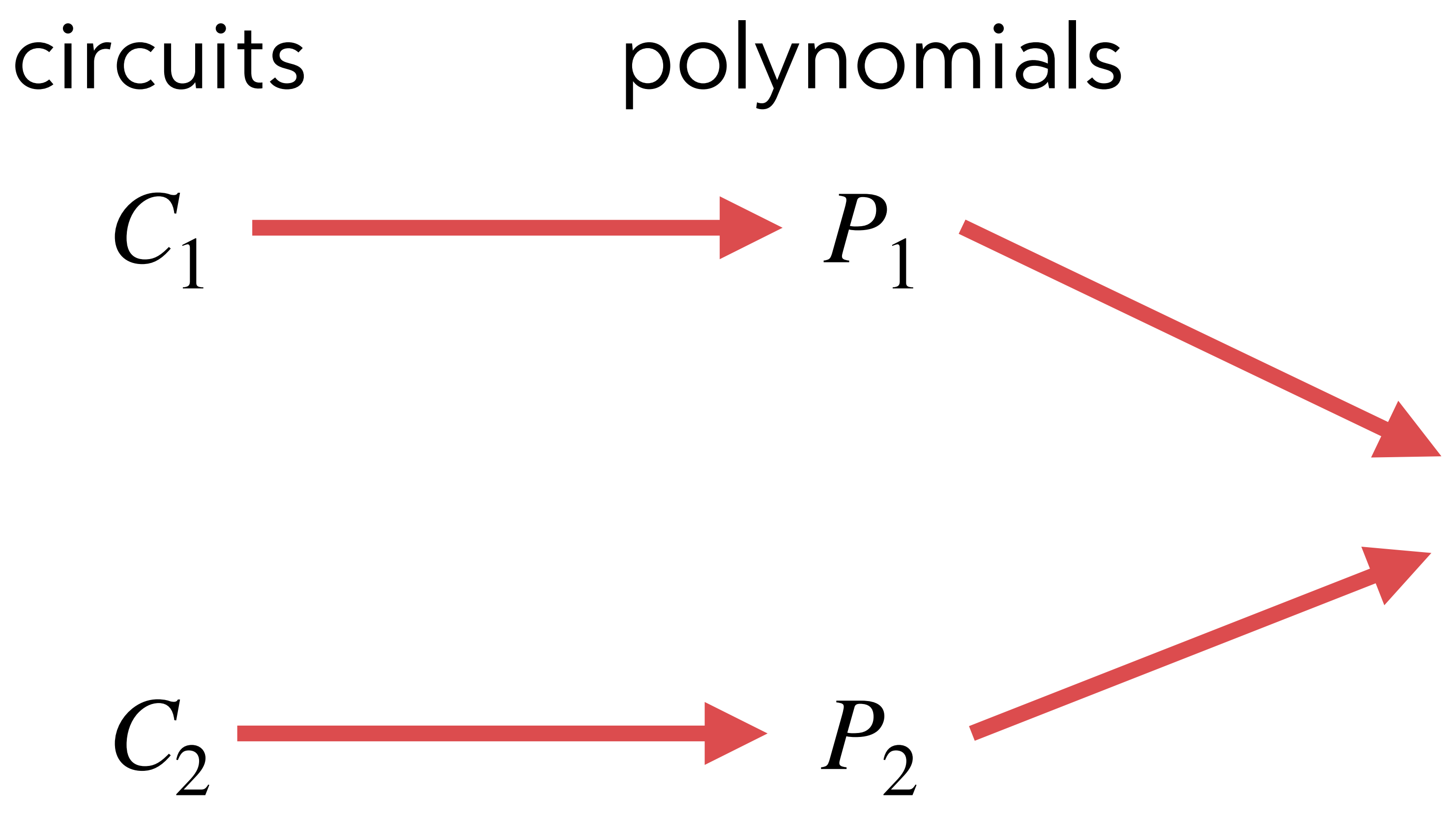
Circuit equivalence



1 randomly sample some values $\mathbf{a}$

2 return $P_1(\mathbf{a}) = P_2(\mathbf{a})$

Circuit equivalence



1 randomly sample some values $\mathbf{a}$

2 return $P_1(\mathbf{a}) = P_2(\mathbf{a})$

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

circuits → polynomials

$C_1 \rightarrow P_1$

$C_2 \rightarrow P_2$

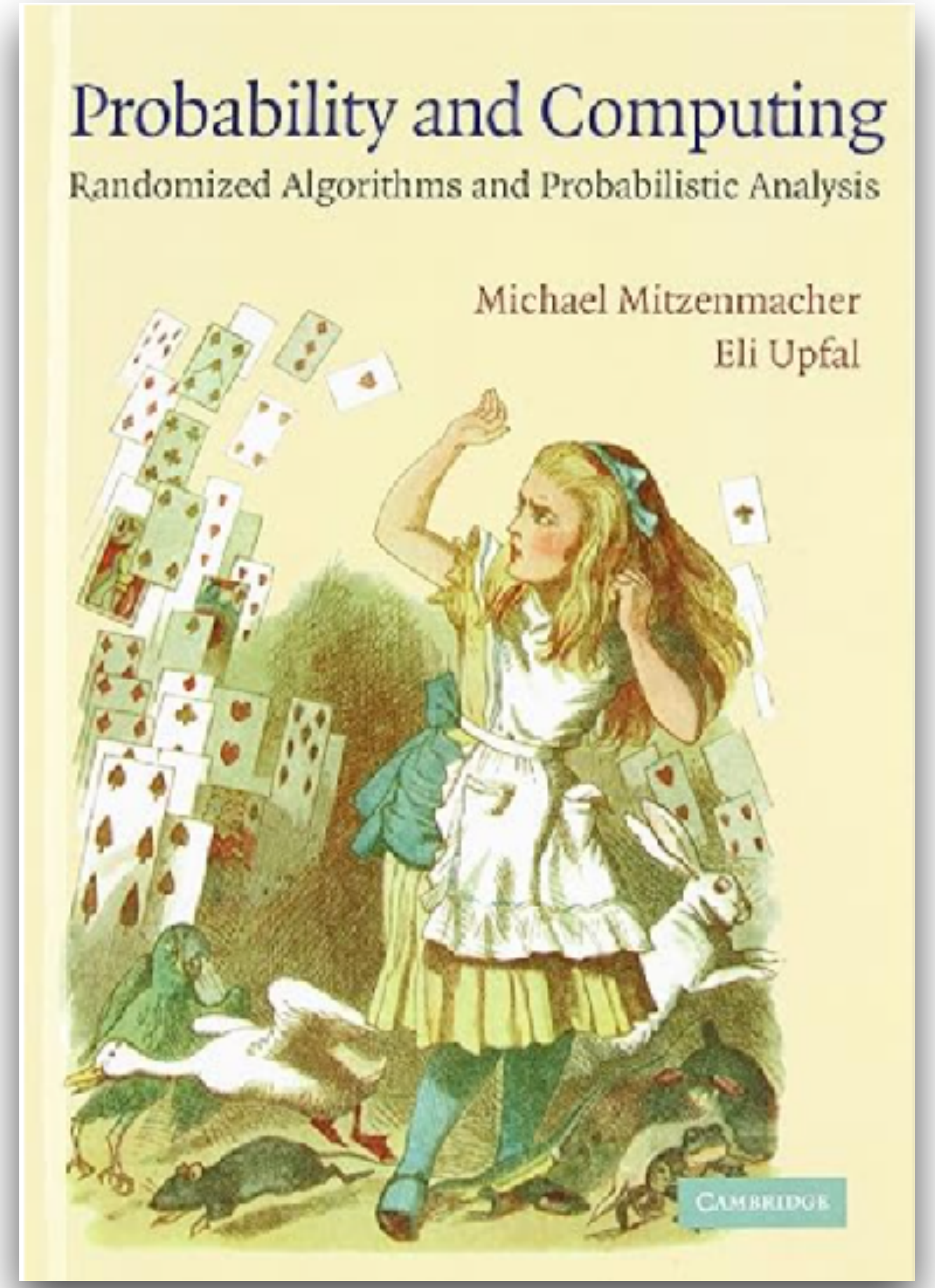
1 randomly sample some values $\mathbf{a}$

2 return $P_1(\mathbf{a}) = P_2(\mathbf{a})$

lemma

if $C_1 = C_2$ then algorithm returns True

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



Polynomial identity filter (PIF)

circuits

C_1

C_2

C_3

$\vdots$

C_n

Polynomial identity filter (PIF)

circuits

C_1

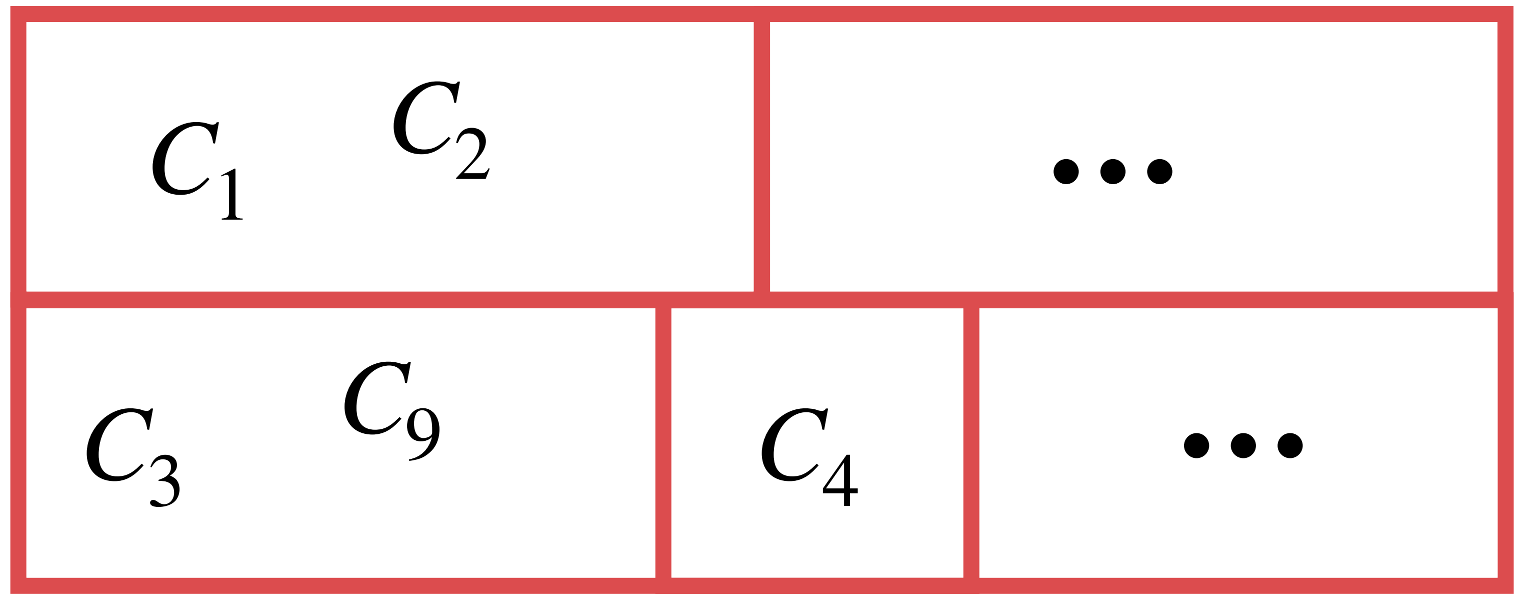
C_2

C_3

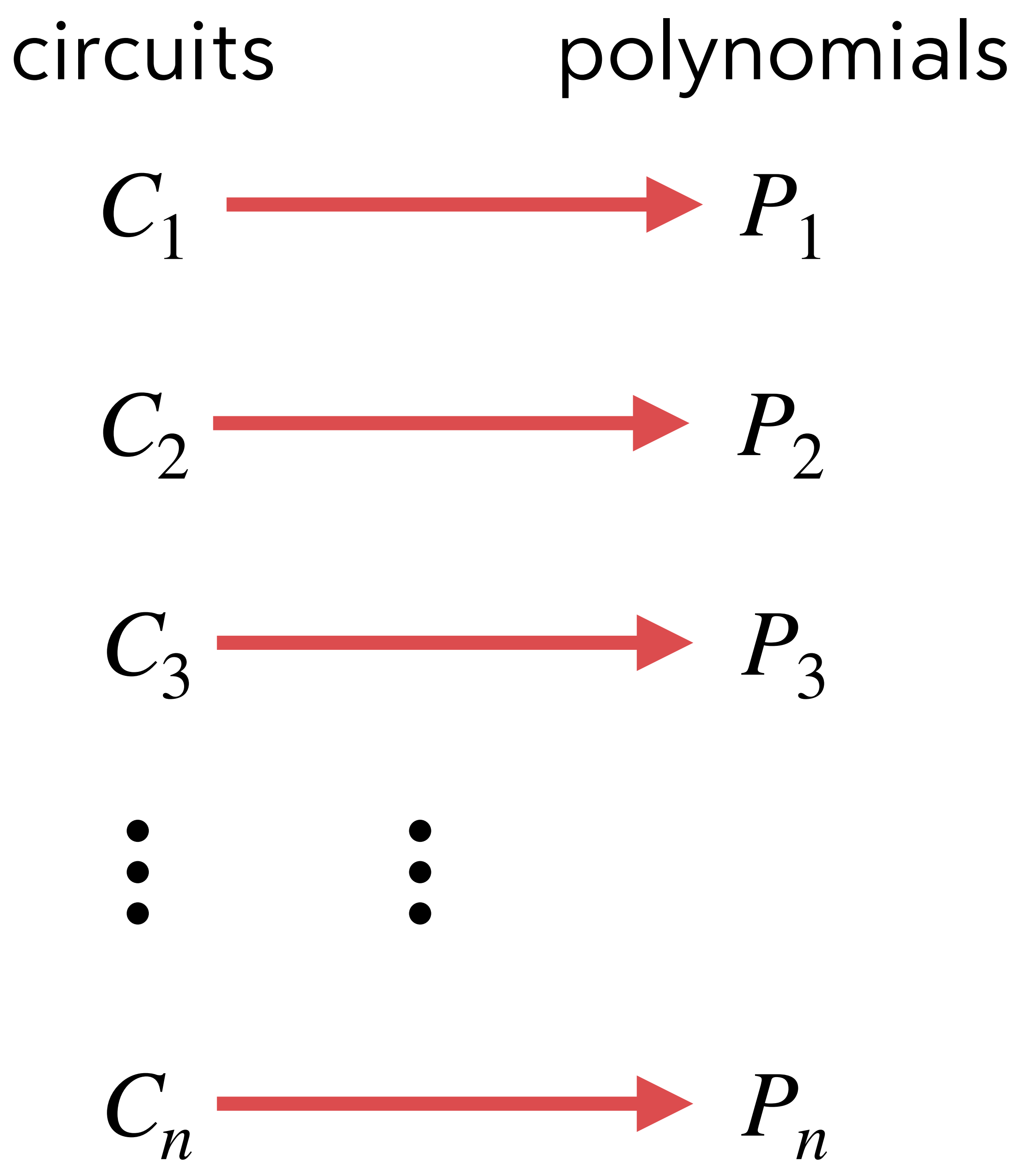
⋮

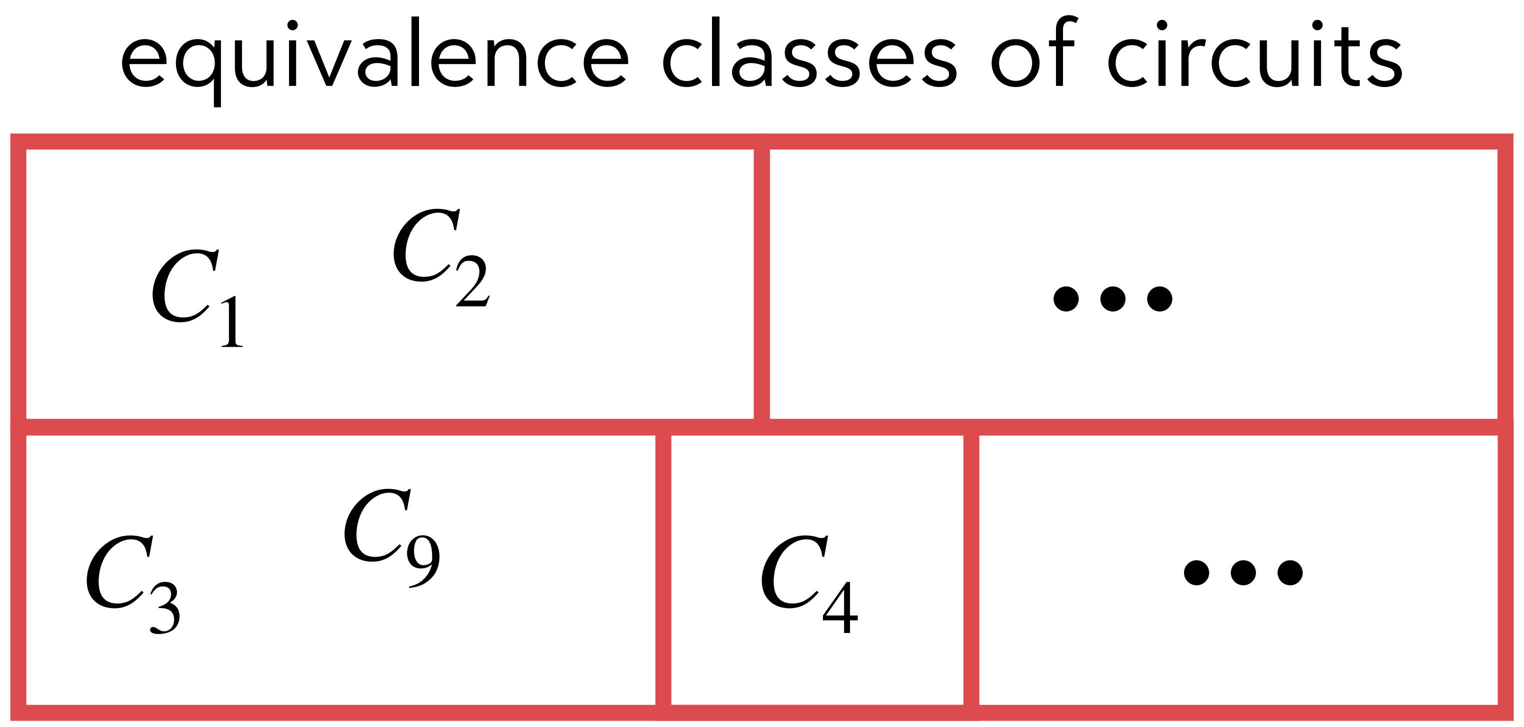
C_n

equivalence classes of circuits



Polynomial identity filter (PIF)





Polynomial identity filter (PIF)

circuits — polynomials — complex #s

$C_1 \longrightarrow P_1 \longrightarrow P_1(\mathbf{a})$

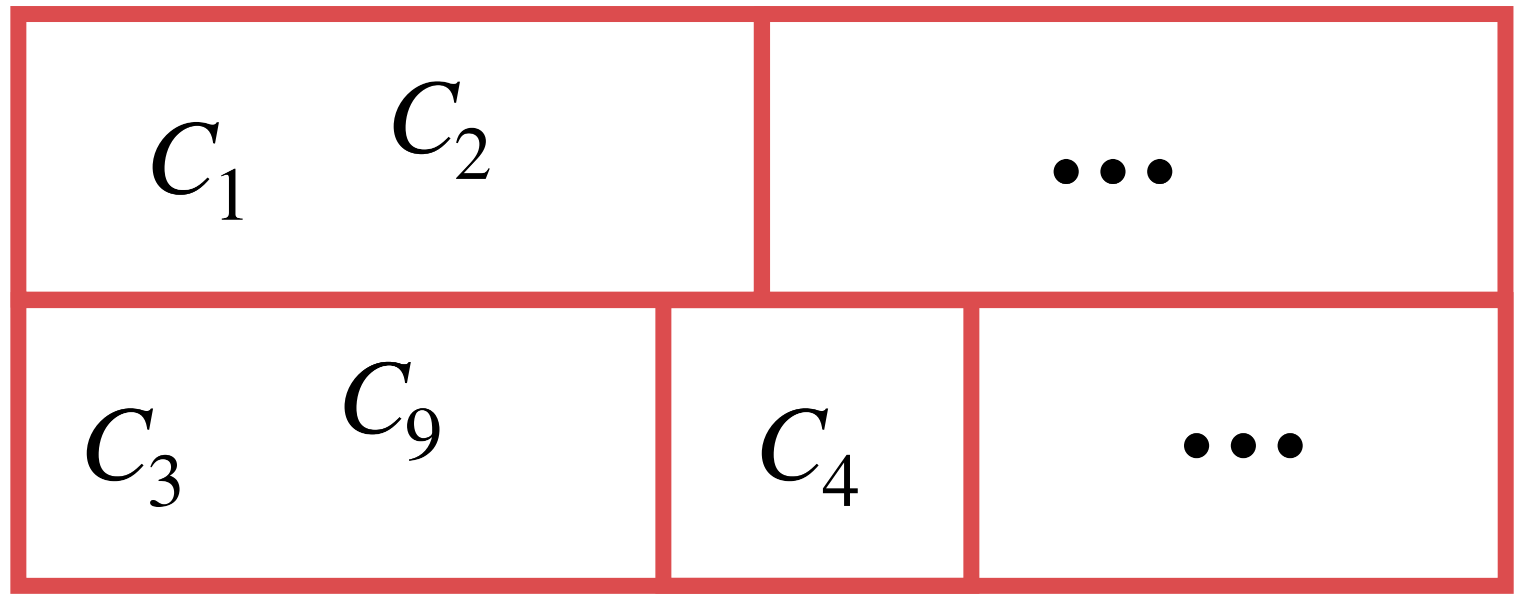
$C_2 \longrightarrow P_2 \longrightarrow P_2(\mathbf{a})$

$C_3 \longrightarrow P_3 \longrightarrow P_3(\mathbf{a})$

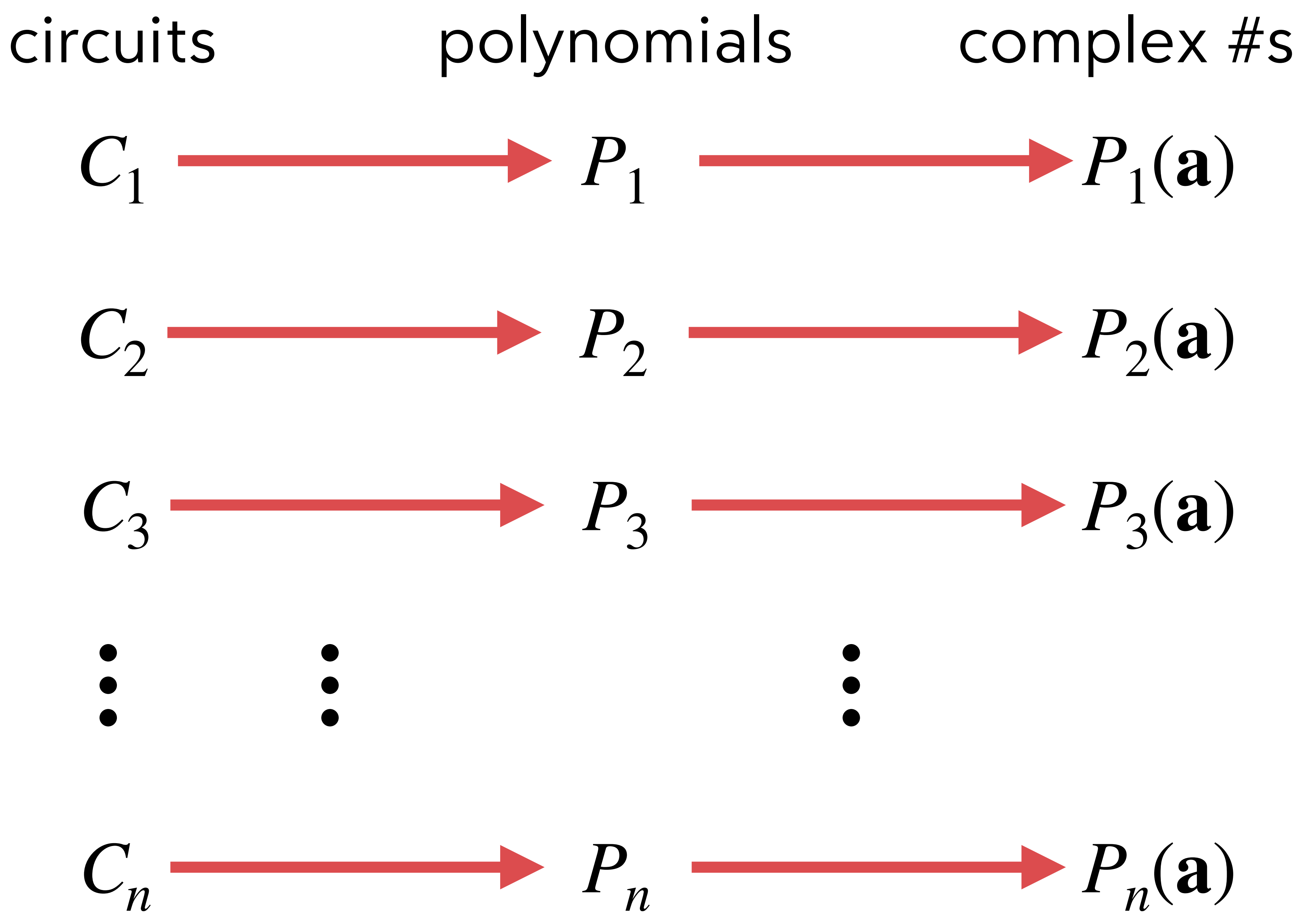
$\vdots \quad \vdots \quad \vdots$

$C_n \longrightarrow P_n \longrightarrow P_n(\mathbf{a})$

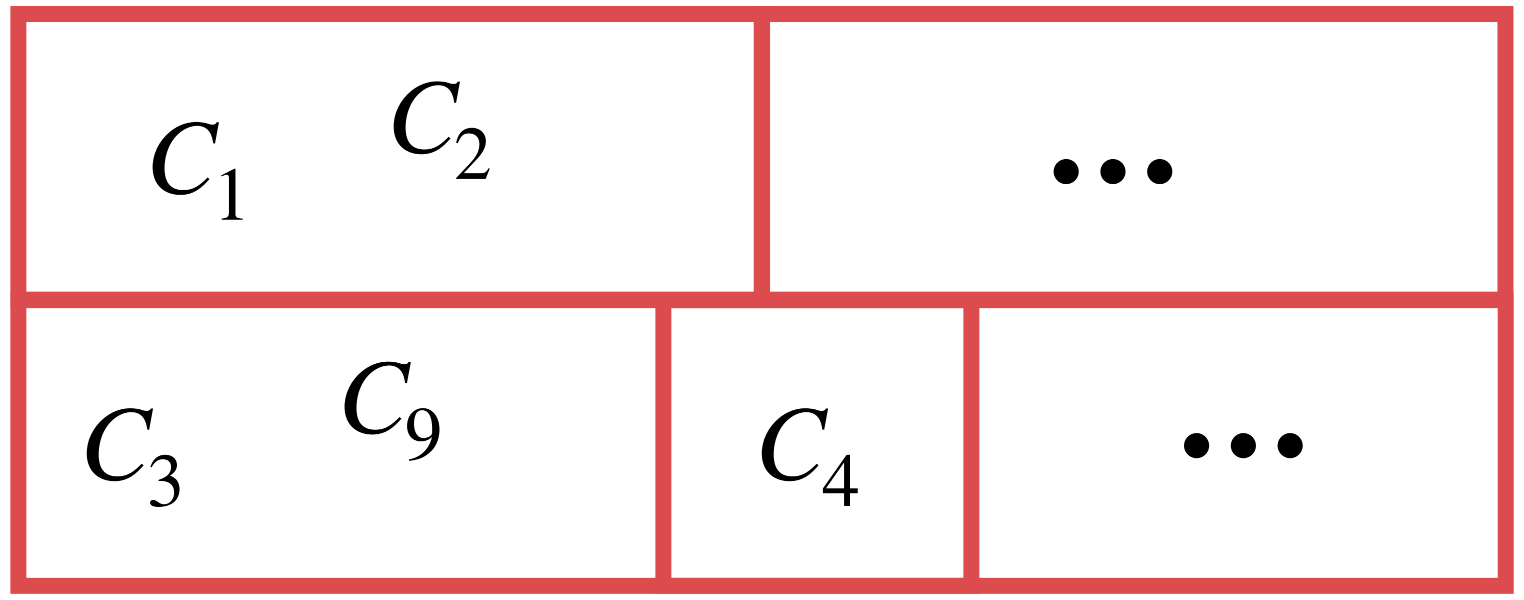
equivalence classes of circuits



Polynomial identity filter (PIF)



equivalence classes of circuits



theorem

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

Symbolic circuits as polynomials

```
H     q;
Rz(θ) q;
```

Symbolic circuits as polynomials

```
H     q;
Rz(θ) q;
```

↓

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

Symbolic circuits as polynomials

```
H      q;
Rz(θ) q;
```

↓

amplitude

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

Symbolic circuits as polynomials

```
H     q;
Rz(θ) q;
```

ℂ var amplitude

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

Symbolic circuits as polynomials

```
H      q;
Rz(θ) q;
```

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;
```

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;
```

ℂ var — amplitude

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

constrained polynomials

rewrite

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

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

…

constrain variables to unit circle

Symbolic circuits as polynomials

```
H     q;
Rz(θ) q;
symb  q;
```

Symbolic circuits as polynomials

```
H      q;
Rz(θ) q;
symb   q;
```

symb semantics

$$|x\rangle \to \phi |x\rangle$$

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

$$|x\rangle \to \phi\,|x\rangle$$

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

Symbolic circuits as polynomials

```
H      q;
Rz(θ)  q;
symb   q;
```

symb semantics

$$|x\rangle \to \phi\,|x\rangle$$

general symb semantics

$$|x\rangle \to \phi(x,y)\,|f(x,y)\rangle$$

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

The power of symbolic circuits

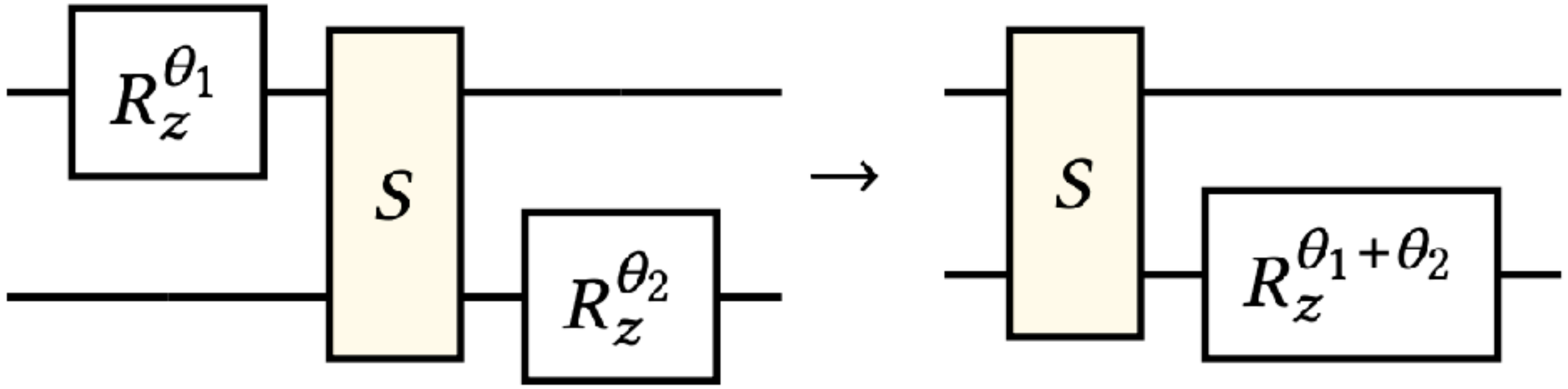
synthesize symbolic rules with long-range interaction

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 · Raw

```python

def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
    """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
    gate cancellation using commutativity rules and unitary synthesis.

    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.
```

Ordering rewrite rules

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

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

```python

def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
    """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
    gate cancellation using commutativity rules and unitary synthesis.

    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.
```

Ordering rewrite rules

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

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

```python

def level_3_pass_manager(pass_manager_config: PassManagerConfig) -> StagedPassManager:
    """Level 3 pass manager: heavy optimization by noise adaptive qubit mapping and
    gate cancellation using commutativity rules and unitary synthesis.

    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 trans                                coupling constraints.
    It is then unrolled to the                                ctions 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

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

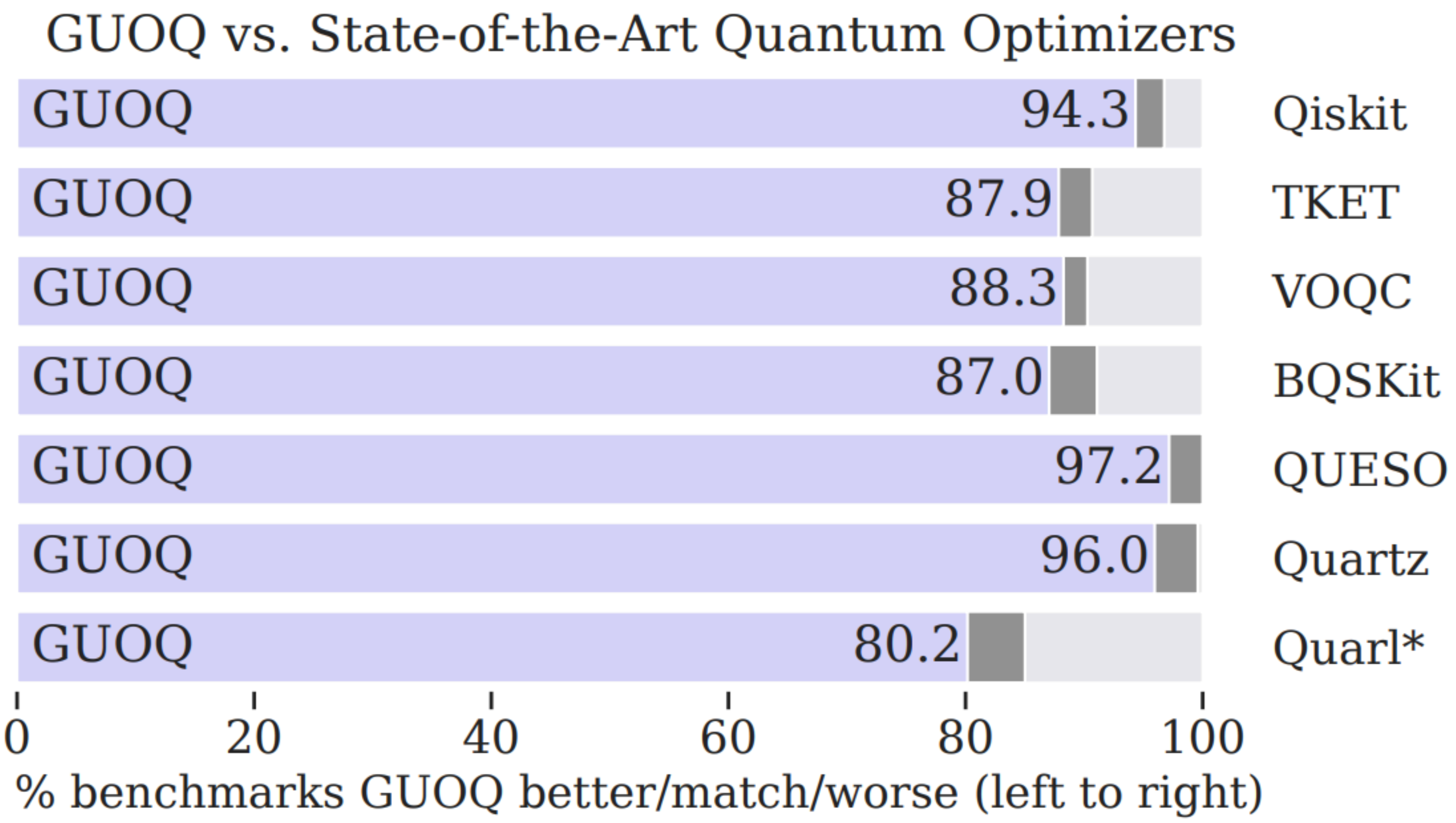
- pick one of the rules
- apply it to a subcircuit
- if the circuit is smaller, accept, otherwise reject with high probability

dynamically generate new rule 1.5% of the time

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

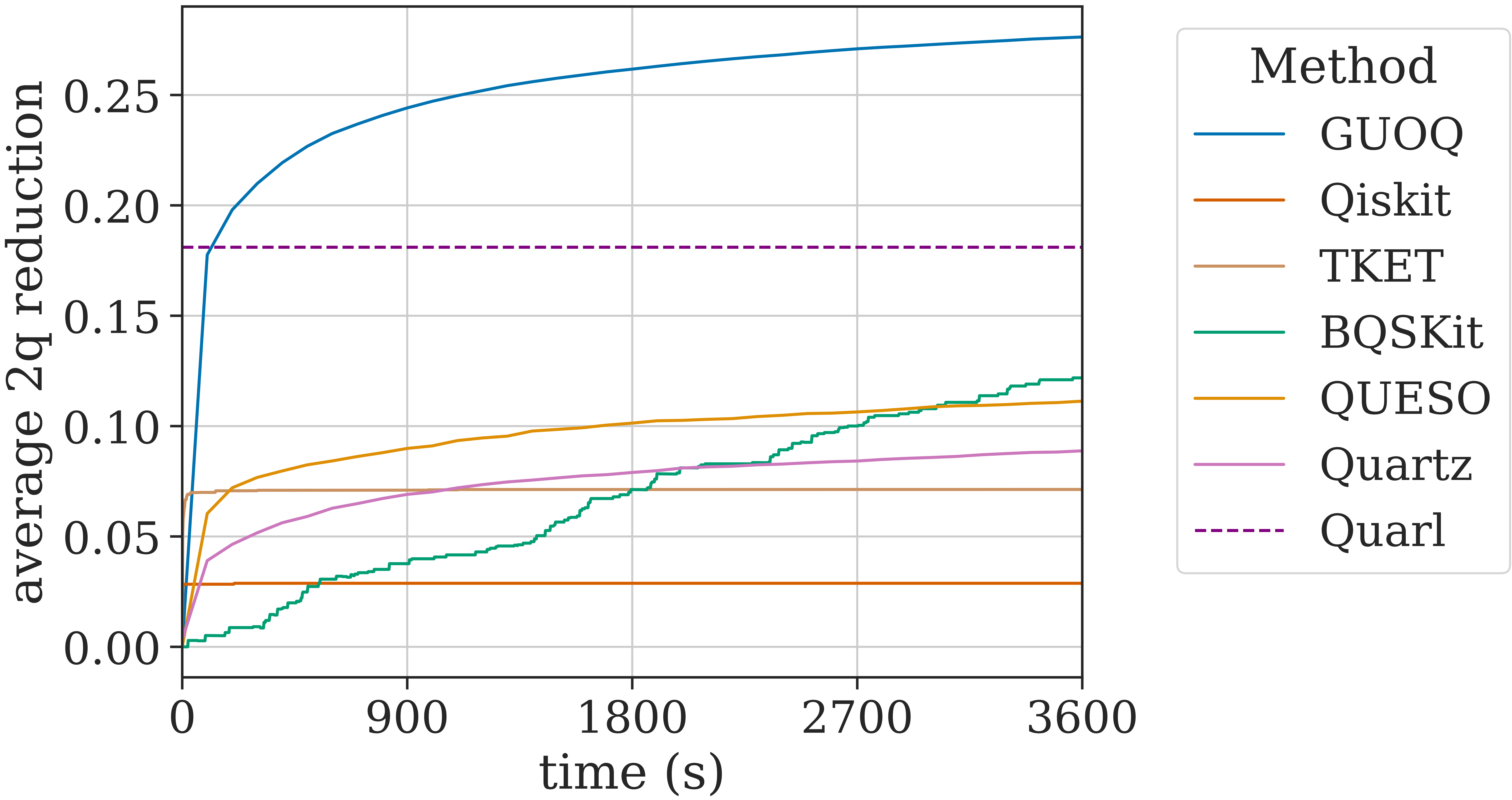
Evaluation: Comparisons

synthesis time: 1.2 min vs 10.4 min (Quartz)

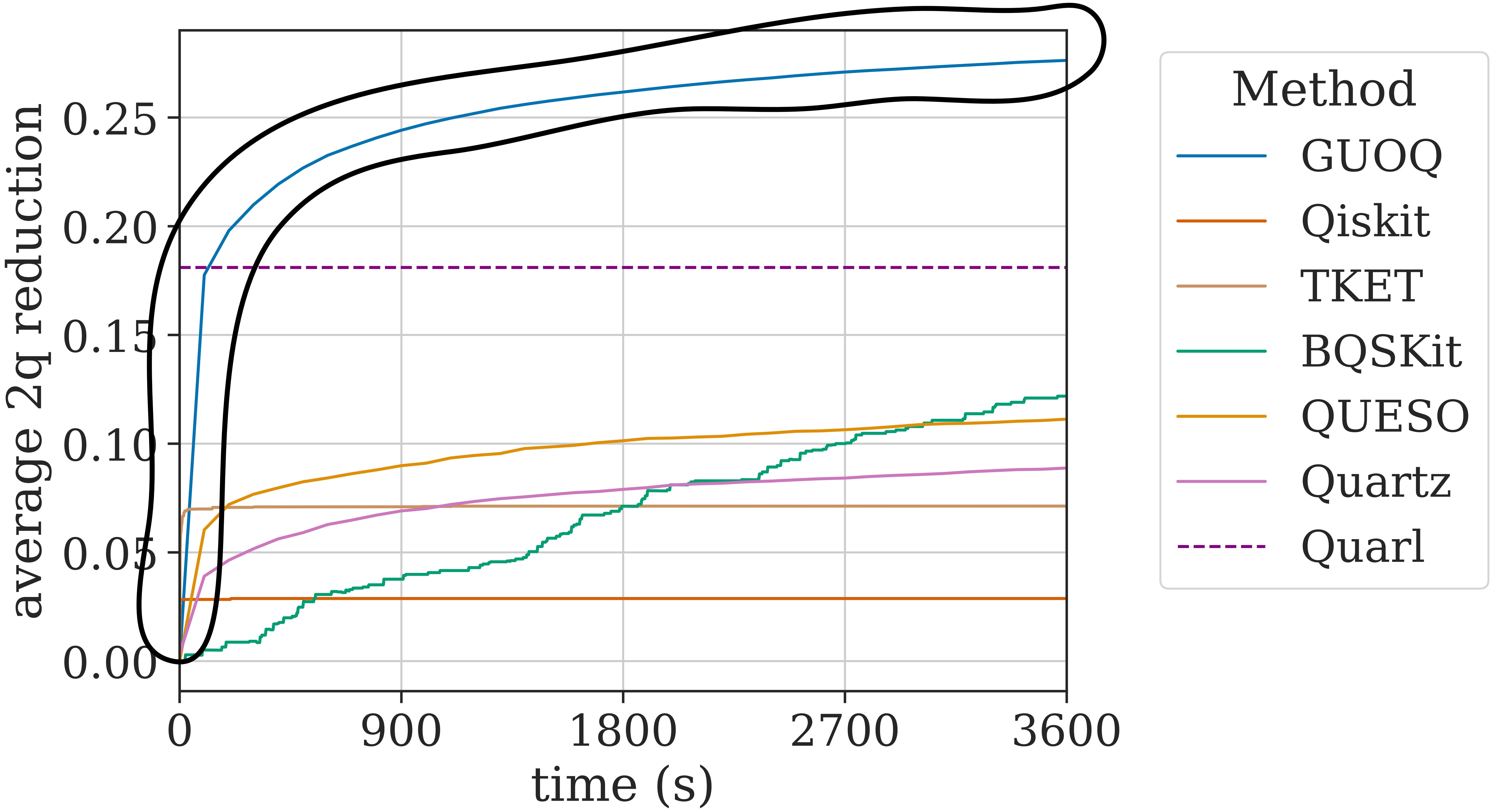


* Li et al., OOPSLA 2024

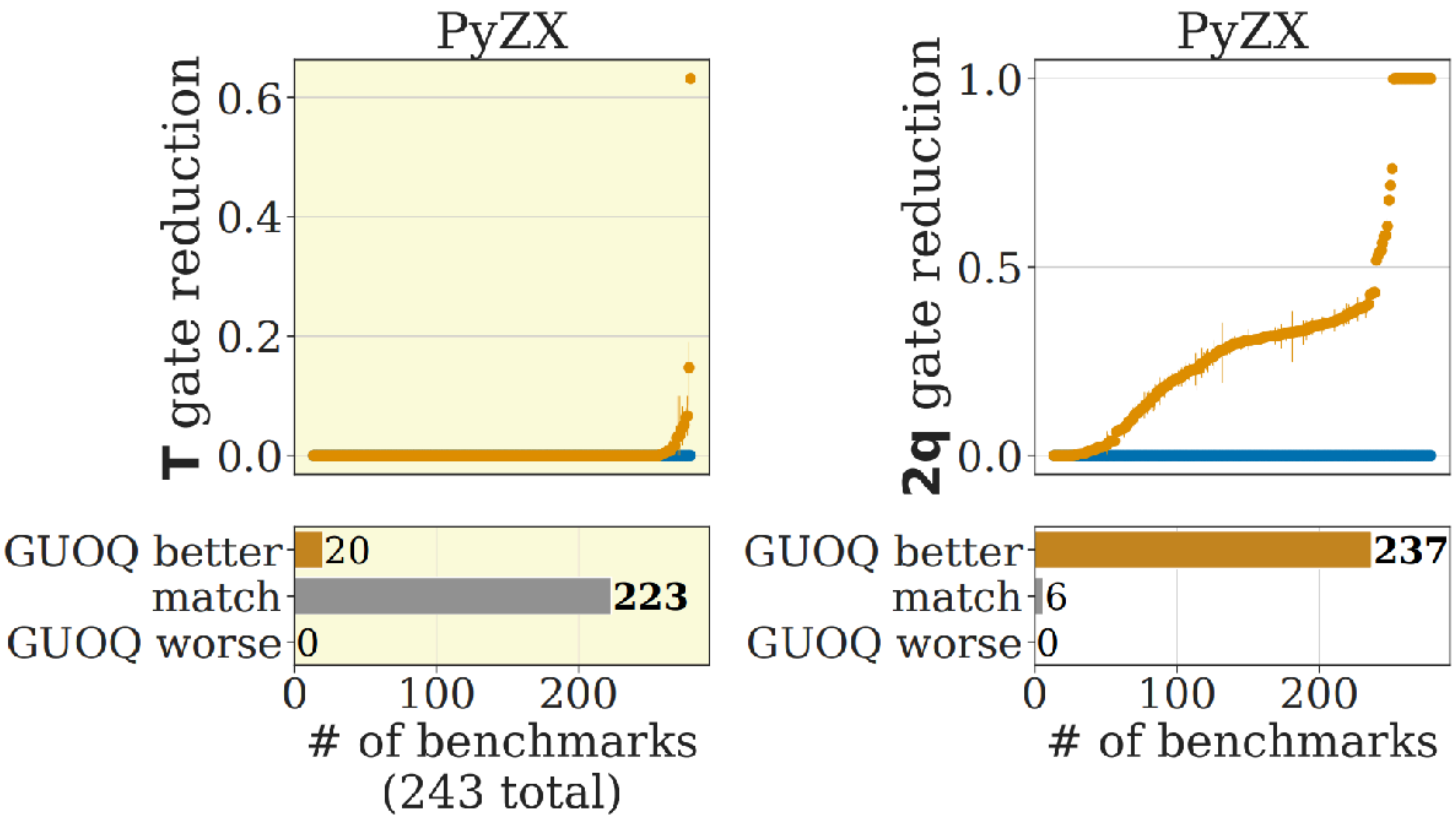
A closer look at reduction



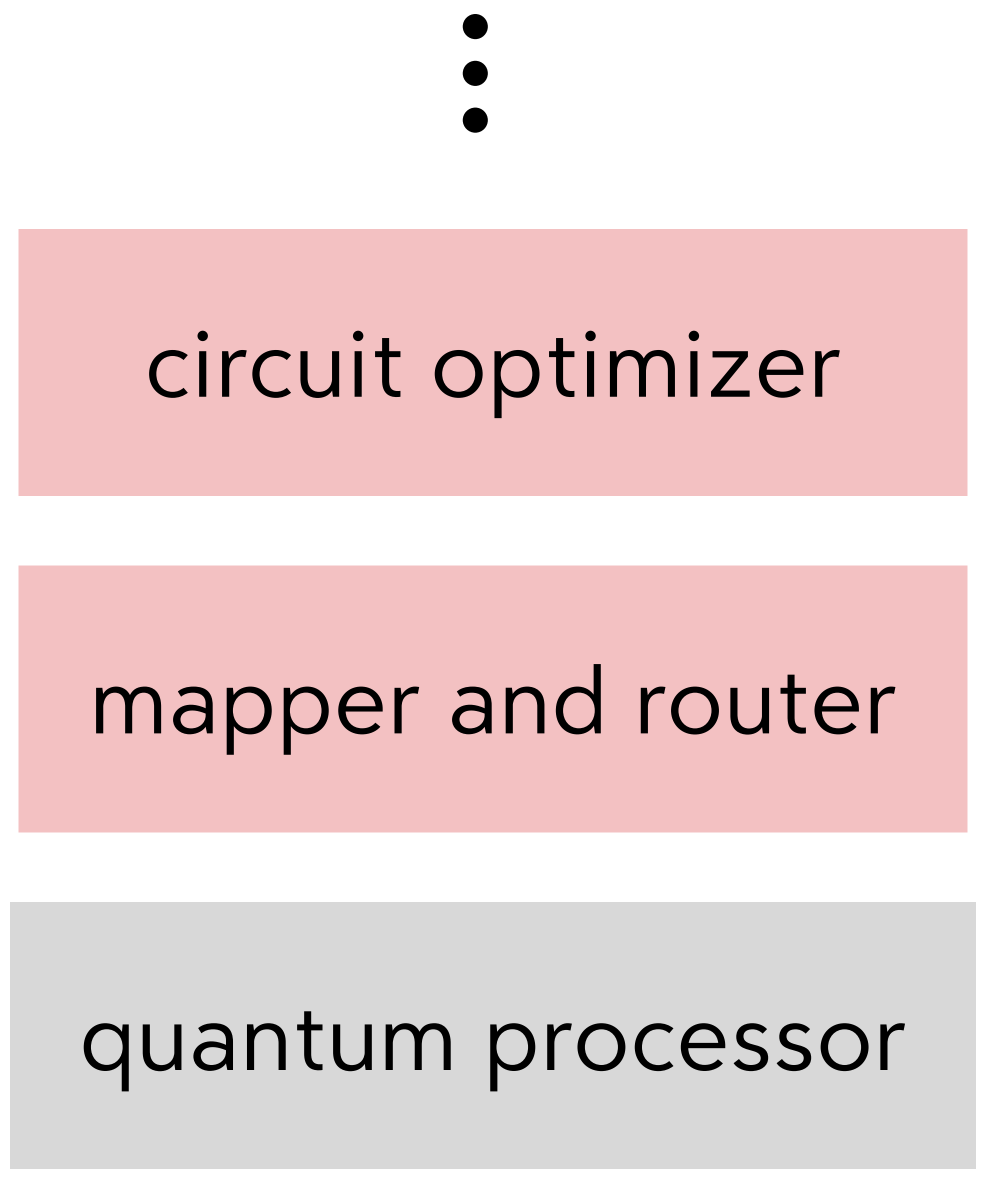
A closer look at reduction



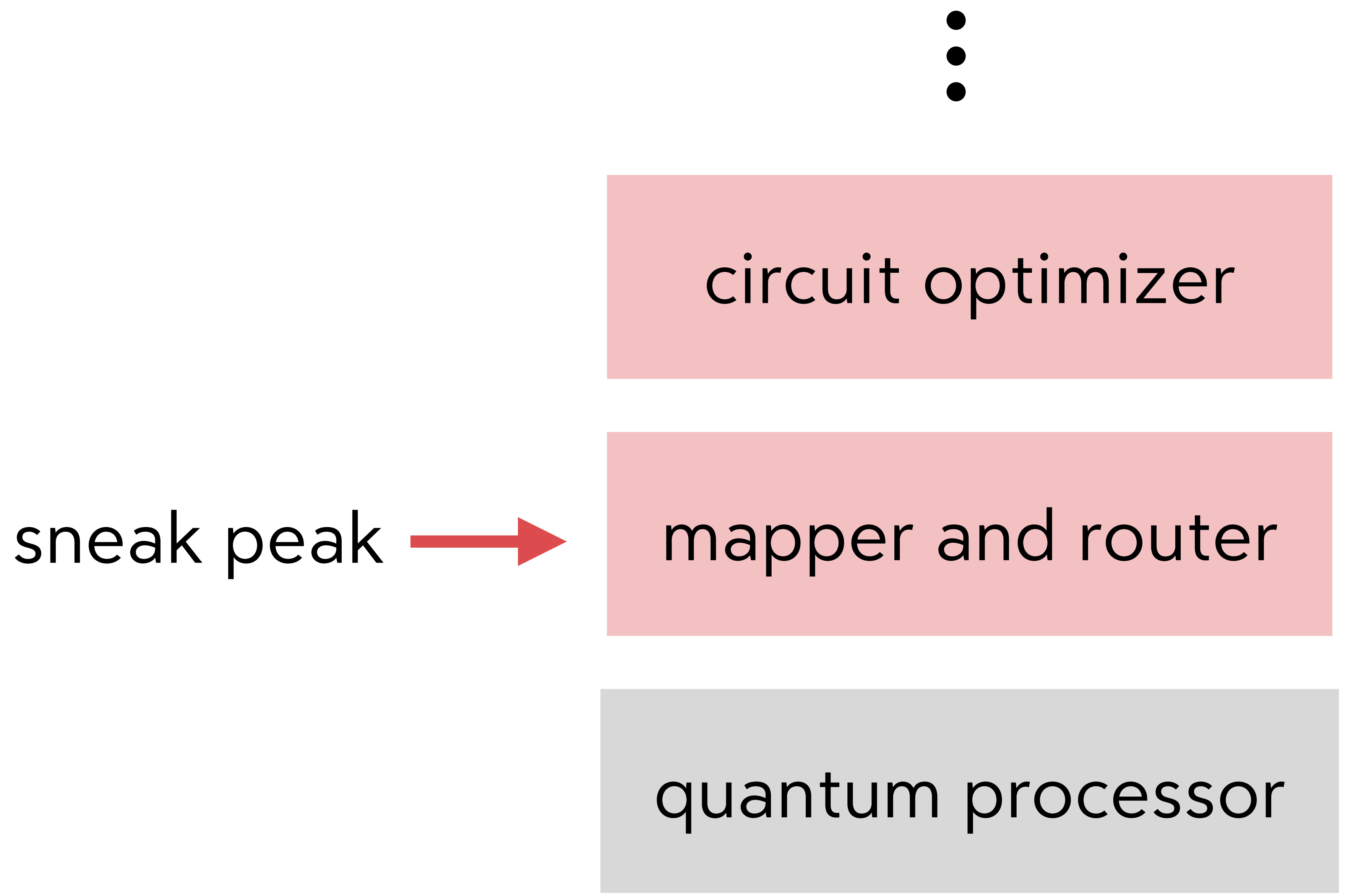
Evaluation: FTQC



Synthesizing quantum compilers



Synthesizing quantum compilers

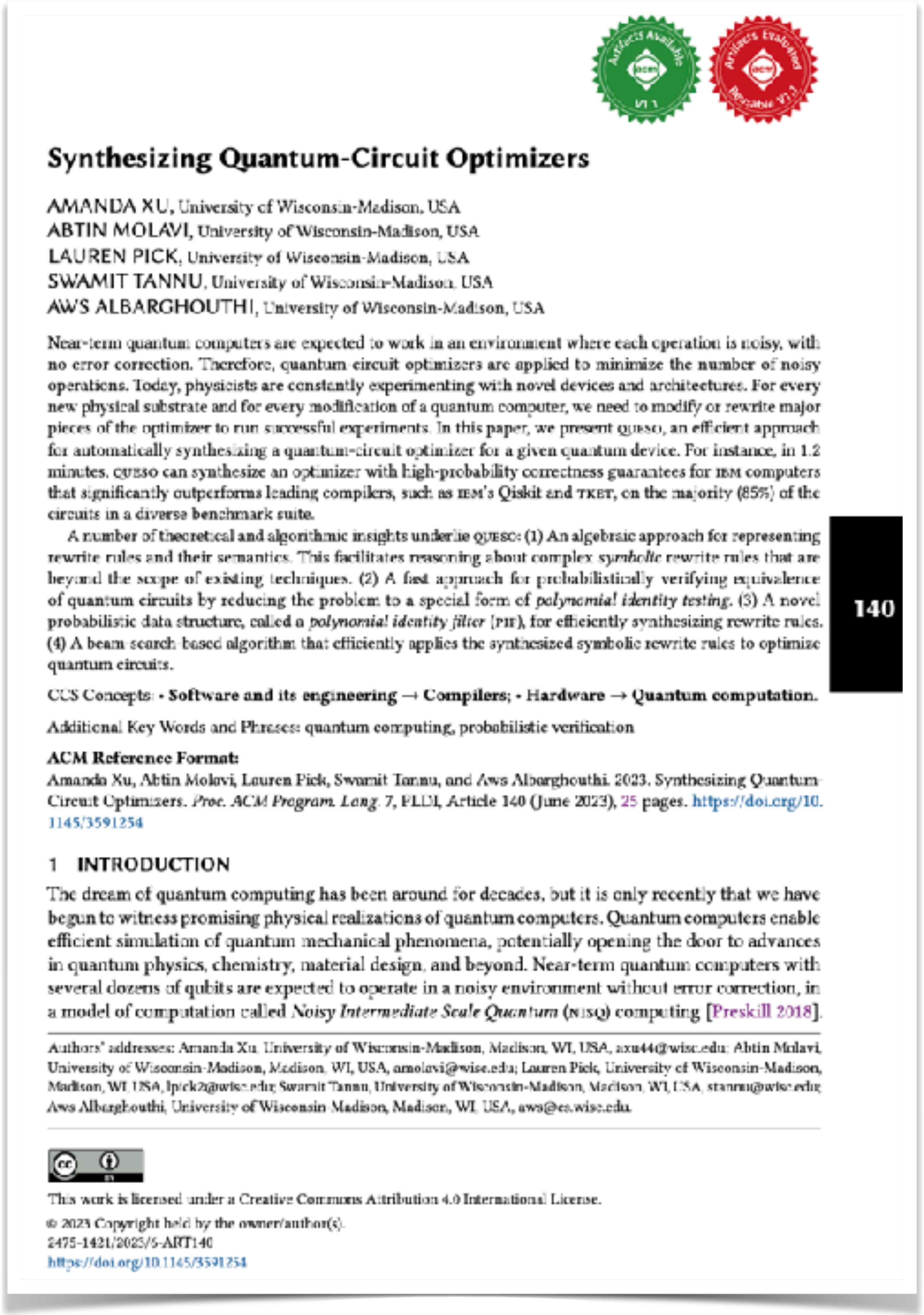


⋮

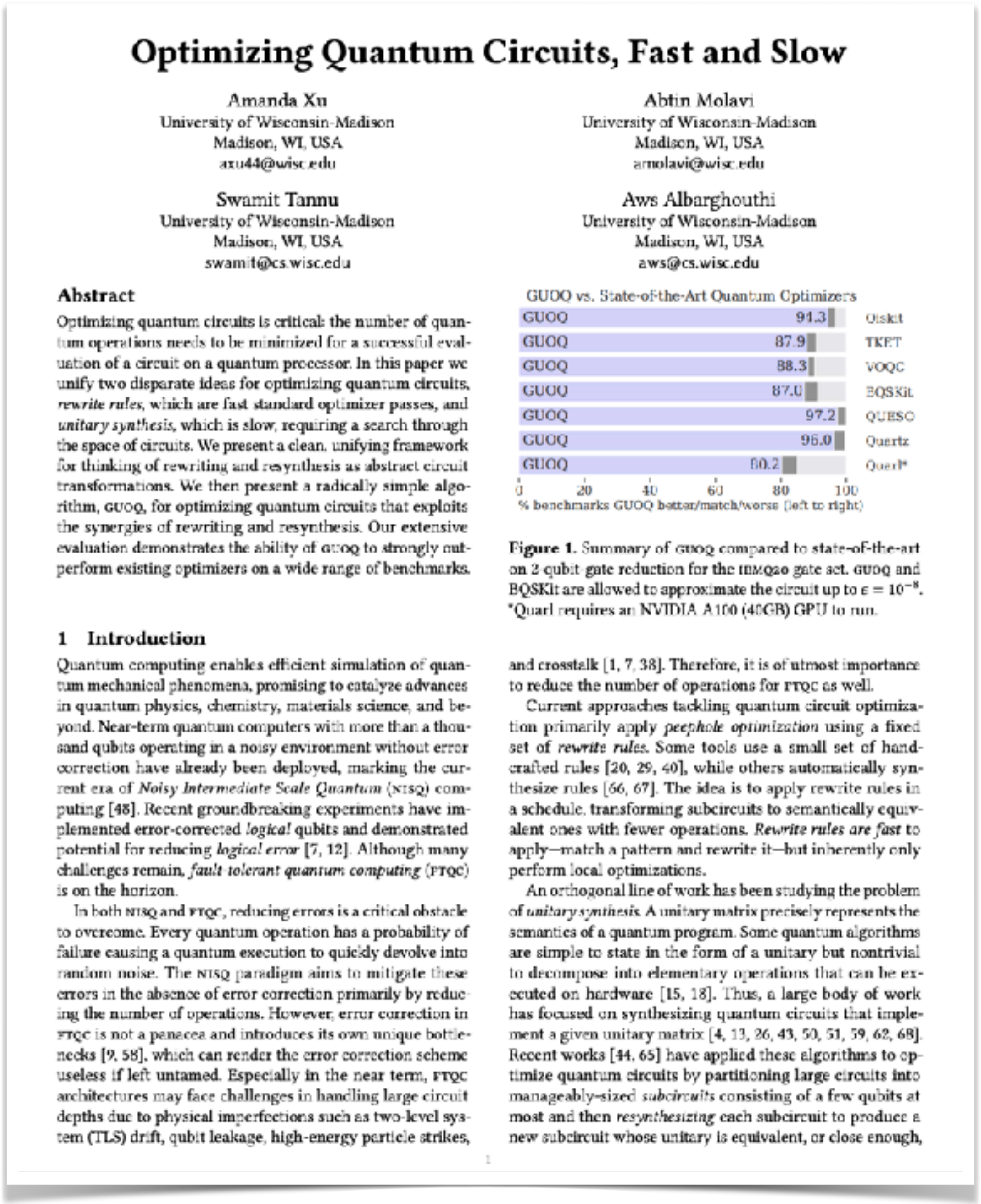
circuit optimizer

mapper and router

quantum processor



Xu et al., PLDI 2023



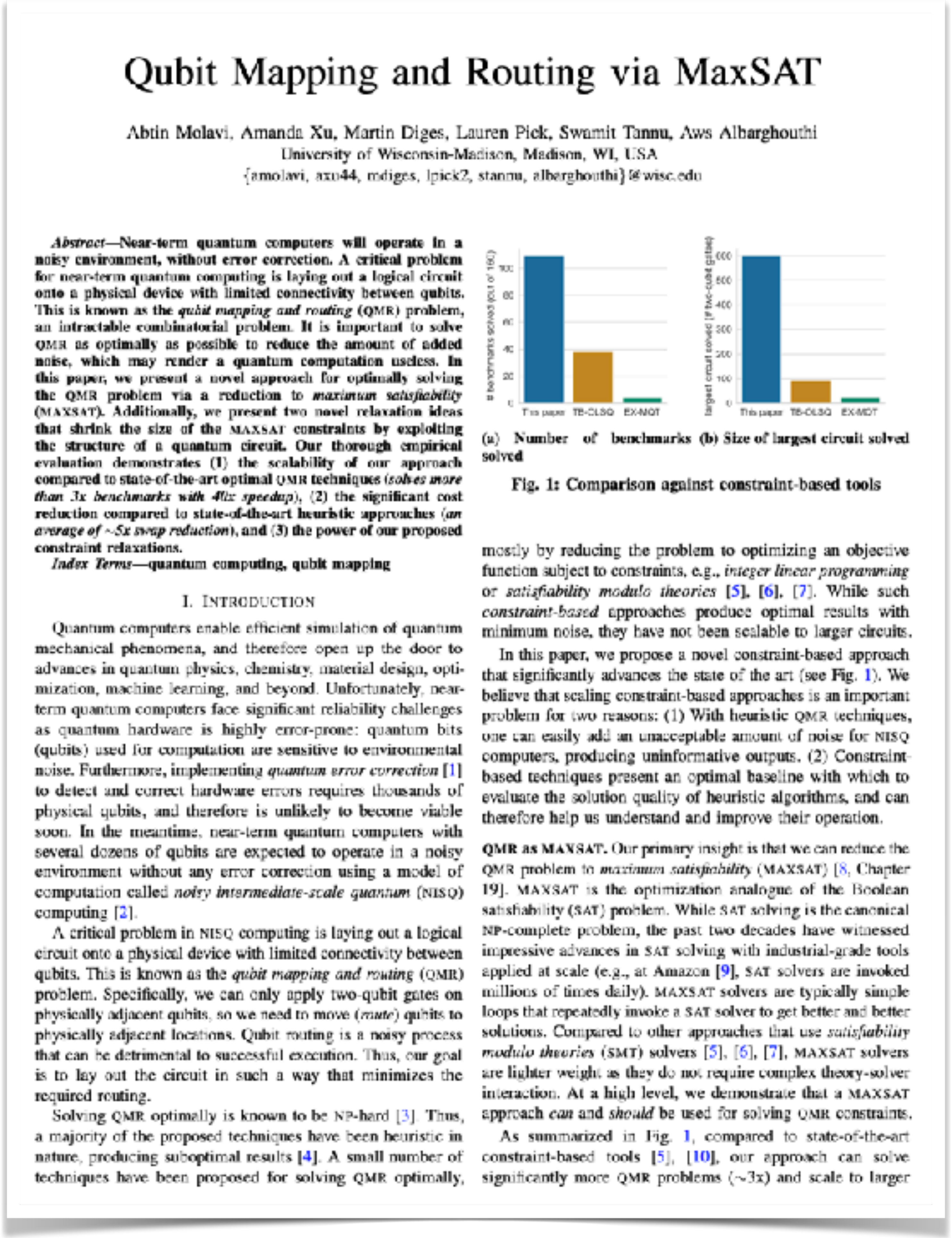
Xu et al., ASPLOS 2025

⋮

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 a 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 swap reduction*), and (3) the power of our proposed constraint relaxations.**

***Index Terms*—quantum computing, qubit mapping**

(a) Number of benchmarks solved (b) Size of largest circuit solved

Fig. 1: Comparison against constraint-based tools

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, 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

arXiv:2311.18042v2 [quant-ph] 23 Oct 2024

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/
```