How to Model Unitary Oracles

Mark Zhandry (NTT Research & Stanford University)

Q: What does it mean to “efficiently implement” a unitary?

Only recently first pass at formalization by [Bostanci-Efron-Metger-Poremba-Qian-Yuen’23]

Q: How should we model query access to efficient unitaries?

$|\Psi\rangle \rightarrow U\,|\Psi\rangle$ What about inverse, controlling, anything else?

Q: What does a black box unitary (e.g. for separations) look like?

Primitive **P** ➡ Primitive **Q** ⇄ **P**

Adversary **A** for **Q** ➡ Adversary **B** for **P** ⇄ **A**

Our thesis (subject to further scrutiny):

- Efficient implementation = small circuit that implements **U** ***including global phase***, ideally to within ***exponentially-small error***

- Oracles capturing efficient computation should allow **controlling CU, (controlled) inverses $CU^\dagger$,** as well as **conjugates CU^* and transposes CU^T**,

- Black box separations should likewise allow queries to **CU, $CU^\dagger$, CU^*, CU^T**

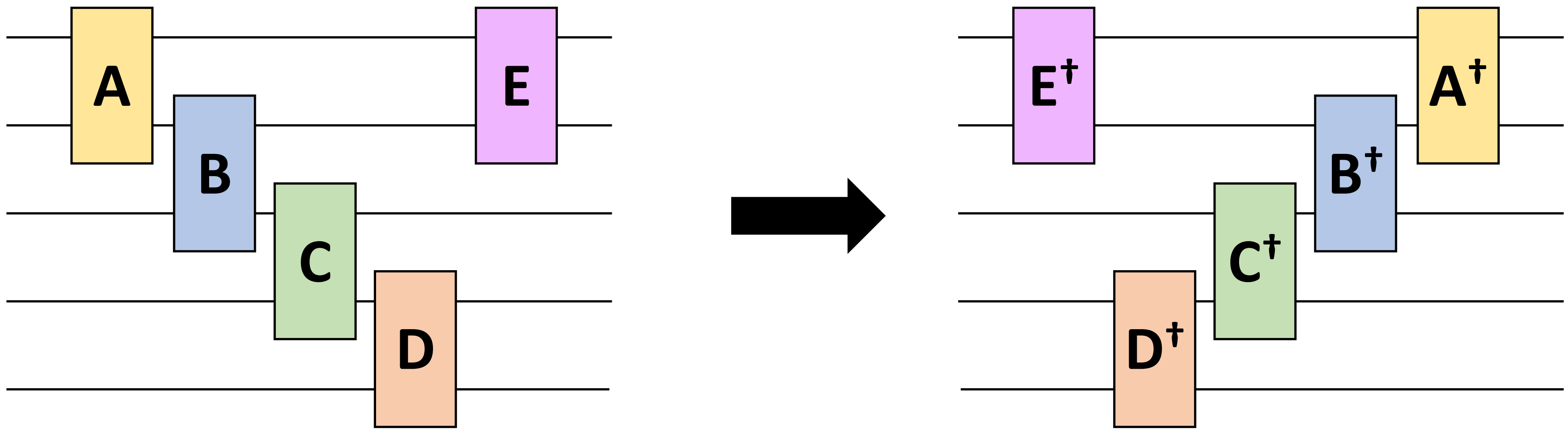
$CU, CU^\dagger$

CU, CU† comes up frequently when using quantum sub-routines

- Gentle Measurements [Winter’99, Aaronson’04]
- Hadamard Test [Aharonov-Jones-Landau’09]
- Phase estimation [Kitaev’95]
- Amplitude amplification where angle unknown [Brassard-Høyer’97, Grover’98]
- Quantum state repair [Chiesa-Ma-Spooner-**Z**’21]
- …

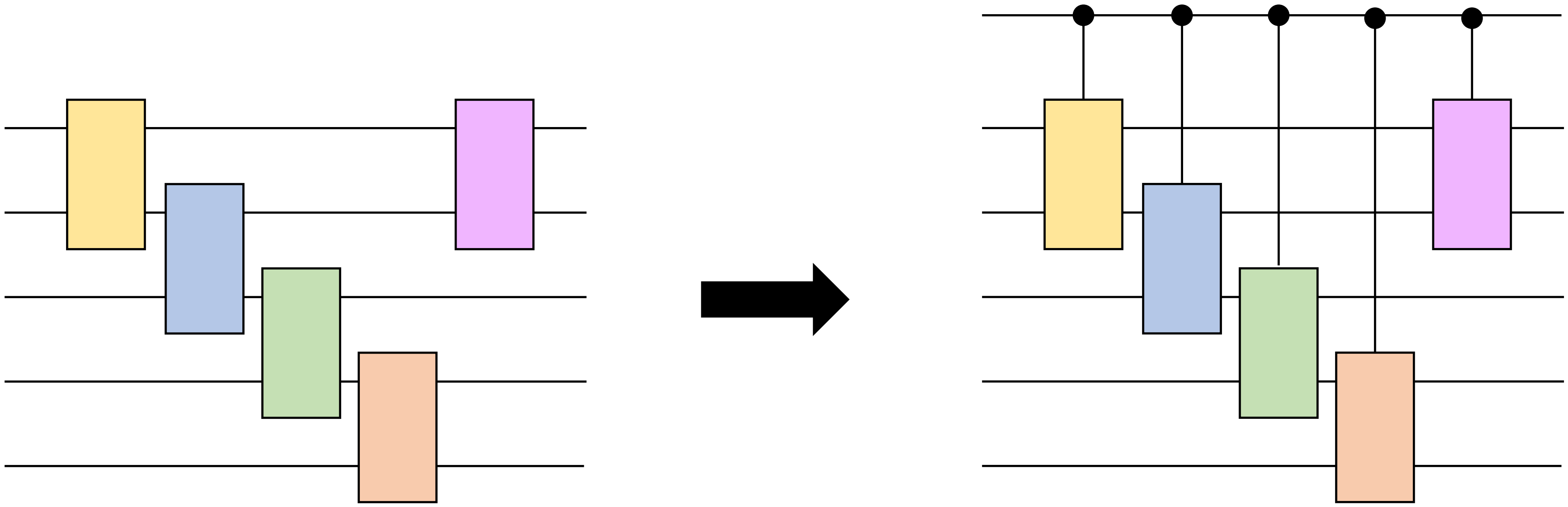
How to implement $\mathbf{U}^\dagger$?

One of the basic rules of linear algebra



How to implement **CU**?

Folklore, but also formalized in [Kim-Tang-Preskill’20]



Since each gate is finite-sized, can implement controlled versions via brute-force

Caveat: Global Phase

If **Q** is a quantum circuit, the unitary implemented by controlling each gate is indeed **CQ**

BUT

We usually ignore overall phase when implementing unitaries

$$\mathbf{Q = e^{i\theta} U} \quad \rightarrow \quad \mathbf{CQ = C(e^{i\theta} U) \neq CU}$$

Inherent with existing notion of universality (defined ignoring global phase)

Example: Controlled QFT with Clifford+T Circuits

Fact: Det(QFT_q)=1 iff **q = 1 mod 8** or **q = 6 mod 8**

Fact: Clifford+T circuits on **n≥3** qubits have determinant **1**

Corollary: Clifford+T circuits cannot implement **QFT_q** including global phase, unless **q = 1 mod 8** or **q = 6 mod 8**.

In particular, cannot implement Shor's algorithm with global phase

Caveat: Global Phase

Thm: There exist families of unitaries that can be computed efficiently when ignoring global phase, but cannot be computed at all when paying attention to global phase, and also cannot be controlled

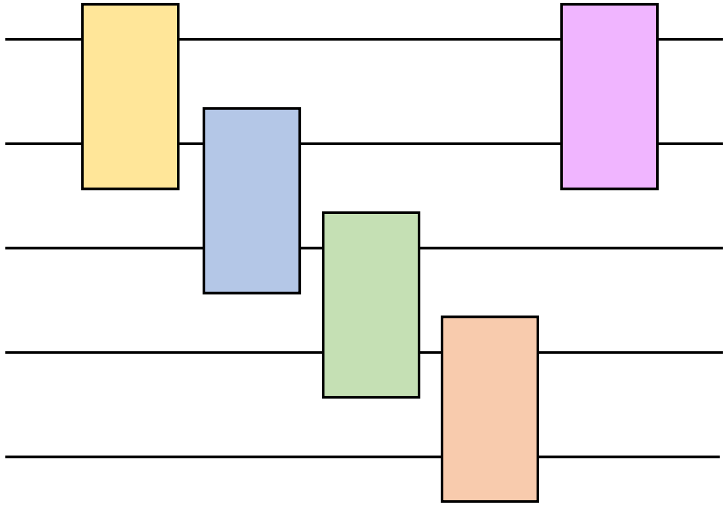
Proof: $\mathbf{U_x = e^{i\,f(x)}\,I}$, where $\mathbf{x}$ encodes instances of the Halting problem

Because of this, we posit that "efficient implementation" should include global phase

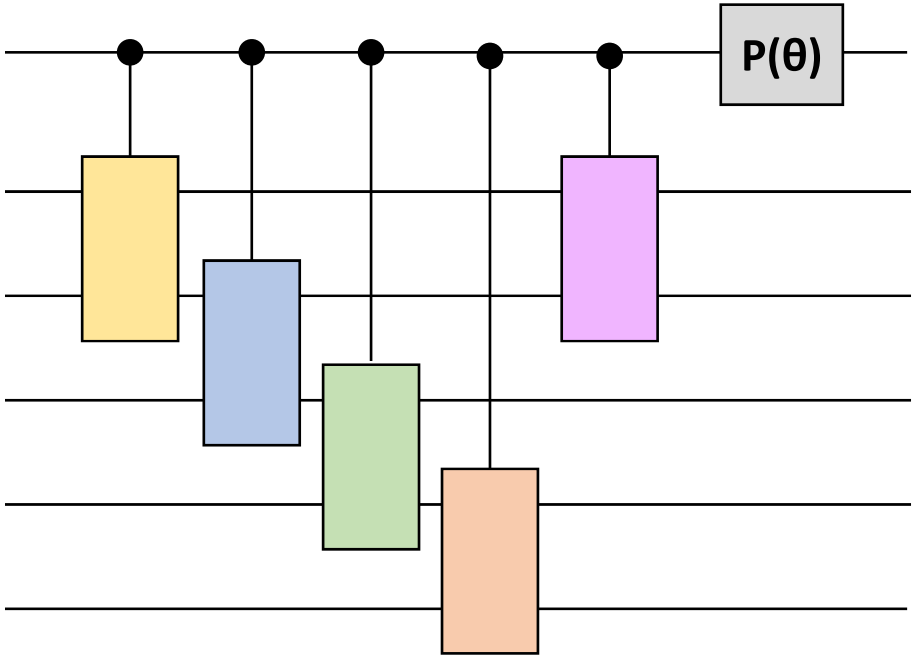
$(\mathbf{Q}, \boldsymbol{\theta})$ implements $\mathbf{U}$ means $\mathbf{U = e^{i\theta}\,Q}$

Fortunately, we generally know the phase $\boldsymbol{\theta}$

How to actually implement **CU**



$\boldsymbol{\theta}$



$\boldsymbol{\theta'}$

(comes from implementing **P(θ)**)

Another Example: Estimating the Jones Polynomial

[Aharonov-Jones-Landau'05]

Blueprint:

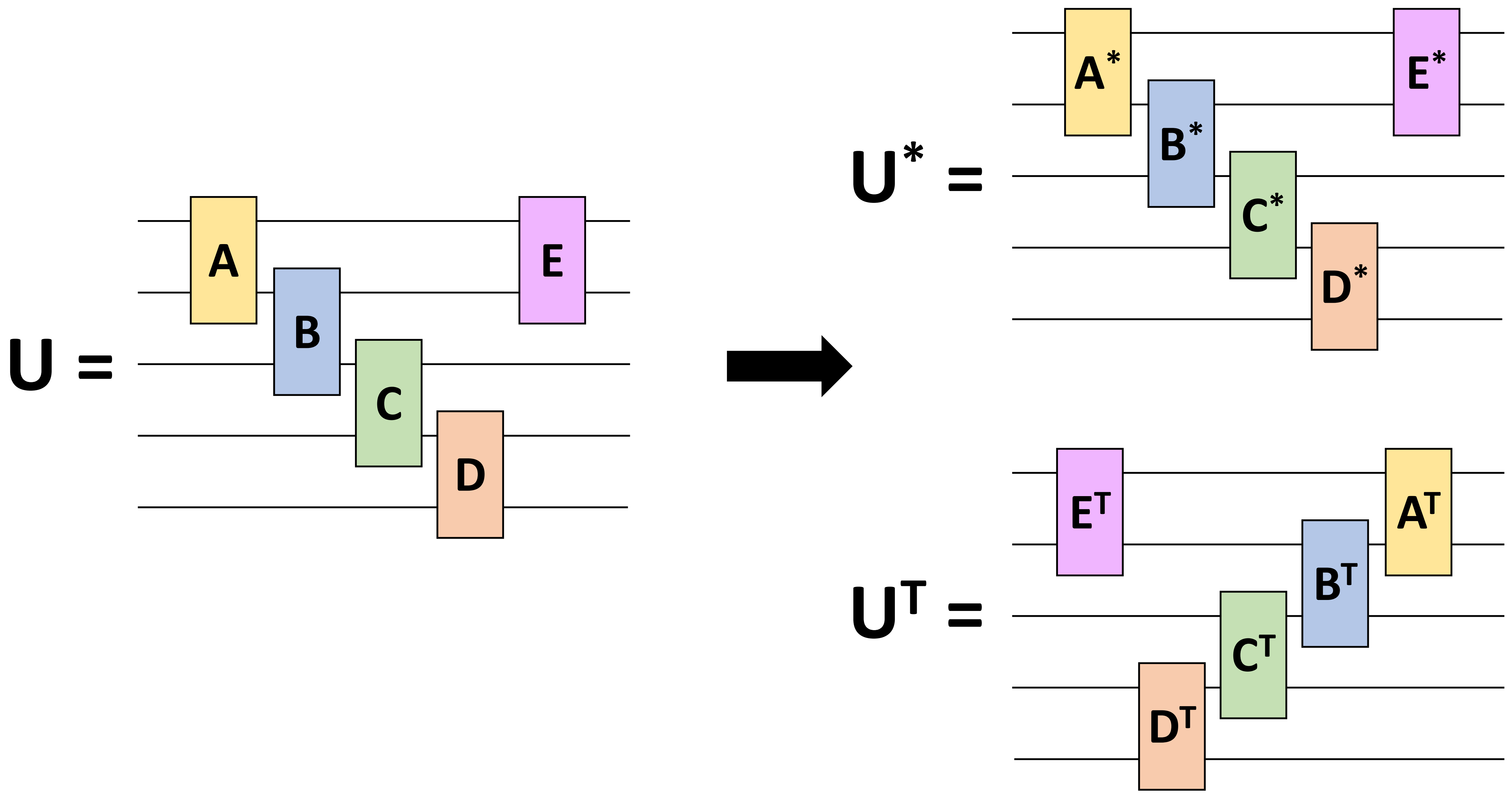
- Knot → circuit **Q** made of unitaries $\mathbf{U_i}$ of polynomial dimension
- Brute-force construct each $\mathbf{U_i}$ → circuit for **Q** over universal gate set
- Hadamard test → estimate $\mathbf{Re[\langle\Psi|Q|\Psi\rangle]}$ for some state $\mathbf{|\Psi\rangle}$
- Estimate gives approximation of Jones polynomial

"Problem": Hadamard test requires controlled operation. If implementing $\mathbf{U_i}$ introduces global phase, will result in incorrect output

Easy Solution: Directly brute-force $\mathbf{CU_i}$

What about $\mathbf{U}^*$, $\mathbf{U}^T$?

How to implement $\mathbf{U}^*$, $\mathbf{U}^T$?



Black-box separations

Often, cannot prove something is hard, but want to nevertheless justify why it's hard

Typical solution: oracle (black-box) separations

E.g. **∃U** s.t. $\mathbf{QMA^U \neq QCMA^U}$ [Aaronson-Kuperberg'07]

Justification: often, the best we can do with a (quantum) circuit is just evaluate it on certain inputs (i.e. treat it as a black box)

How reasonable are black-box separations?

In general, known to fail sometimes (e.g. Chang-Chor-Goldreich-Hartmanis-Håstad-Ranjan-Rohatgi'94)

Nevertheless, seems to be a reasonable heuristic and is widely used throughout classical and quantum complexity theory/cryptography

However, in order for a black-box separation to be most convincing, the oracle should be modeled in a way that best reflects the "real world"

Our thesis: In real world, can implement $\mathbf{U}^*$, $\mathbf{U}^T$, so ideally should include these in oracle model

Unitary vs “Standard” Oracle Separations

Unitary Oracle Separations

Thm [Aaronson-Kuperberg'07]: There is a unitary **U** s.t. $\mathbf{QMA}^{\mathbf{U}} \neq \mathbf{QCMA}^{\mathbf{U}}$

+ a number of follow-up unitary separations in both complexity theory and cryptography

However, unitary oracle separations are considered "non-standard", or at least less desirable that separations relative to classical oracles

Notable research goal: translating unitary separations to classical separations

Question: Can you implement the AK07 oracle using a classical oracle?

Version of this question appeared in AK07 as the "unitary synthesis problem"

Attempt 1: Indistinguishability

$$Q^{O} \underset{\text{(query bounded)}}{\approx} U$$

Problem: adversary can also query **O** directly → may reveal more information about **U** not revealed by queries

Question: Can you implement the AK07 oracle using a classical oracle?

Version of this question appeared in AK07 as the "unitary synthesis problem"

Attempt 2: Indifferentiability

[Maurer-Renner-Holenstein'04]

$$\mathbf{Q^O, O} \underset{\text{(query bounded)}}{\approx} \mathbf{U, Sim^U}$$

Good enough for most cryptographic separations, possibly for "efficient" complexity separations (excl. witness classes like QMA)

Question: Can you implement the quantum oracles using a classical oracle, under indifferentiability?

Thm (informal): No! Unless your quantum oracle allows access to $\mathbf{U^\dagger, U^*, U^T}$ (with caveats; also not lack of controlling)

Proof Idea:

$$\mathbf{U^* \approx \left(Q^{Sim^U} \right)^* = (Q^*)^{\left(Sim^U\right)^*} \approx (Q^*)^{Sim^U}}$$

By indifferentiability, conjugating both sides

Each non-oracle gate conjugated

Since $\mathbf{Sim^U}$ is supposed to look like a classical function

Application: How (not) to construct *indifferentiable* random unitaries

Pseudorandom unitaries from pseudorandom functions

Thm [Ma-Huang'25]:

$$\mathbf{C\,P\,F\,C'} \quad \approx \quad \mathbf{U}$$

(with inverse queries)

C,C' = random Cliffords

$\mathbf{F = \sum_x |x\rangle\langle x|\, e^{i\,2\,\pi\, f(x)\,/\,q}}$ for random function **f**

$\mathbf{P = \sum_x |p(x)\rangle\langle x|}$ for random permutation **p**

Note: PFC construction due to [Metger-Poremba-Sinha-Yuen'24]

Natural question: can we build *indifferentiable* random unitaries if **F,P** replaced with a random function/permutation?

⬇

Necessary-seeming first step: can we build PRUs from PRFs, such that PRU is secure against queries to $\mathbf{U}$, $\mathbf{U}^{\dagger}$, $\mathbf{U}^{*}$, $\mathbf{U}^{T}$ (*****-security?)

*-attack on **CPFC'** when **q=2**

$$
\begin{aligned}
U^* &= (C\ P\ F\ C')^* \\
&= C^*\ P\ F\ (C')^* \\
&= (X^{\theta} Z^{\phi})\ C\ P\ F\ C'\ (X^{\theta'} Z^{\phi'}) \\
&= (X^{\theta} Z^{\phi})\ U\ (X^{\theta'} Z^{\phi'})
\end{aligned}
$$

*-attack on **CPFC'** when **q=2**

Suppose for the moment that $\mathbf{U^{*} = X^{\theta} \, U \, X^{\theta'}}$

➡ Essentially instance of Simon's oracle

➡ Can find **θ,θ'** in poly-many queries

Can distinguish, since clearly such shifts should not hold for Haar random **U**

*-attack on **CPFC'** when **q=2**

Can likewise break if $\mathbf{U^* = Z^{\phi} U Z^{\phi'}}$

Combining **X**'s and **Z**'s: **X** and **Z** don't commute, but $\mathbf{X \otimes X}$ and $\mathbf{Z \otimes Z}$ do

➡ Two queries give Simon's oracle with shift $(\boldsymbol{\theta}, \boldsymbol{\theta'}, \boldsymbol{\phi}, \boldsymbol{\phi'})$

Technically, need controlled oracle to implement Simon's algorithm. Can remove by querying $\mathbf{U \otimes U^*}$ vs $\mathbf{U^* \otimes U}$

Is there anything beyond **CU, $U^\dagger$, U^*, U^T?**

(Anti-) Homomorphisms on Unitaries

CU, U* are *homomorphisms* on unitaries

$$\mathbf{C(UV) = (CU)(CV)} \qquad \mathbf{(UV)^* = (U^*)(V^*)}$$

U^T, U† are *anti-homomorphisms*

$$\mathbf{(UV)^T = (V^T)(U^T)} \qquad \mathbf{(UV)^\dagger = (V^\dagger)(U^\dagger)}$$

All anti-homomorphisms are the inverse of some homomorphism

Can efficiently compute (anti-)homomorphisms by applying them gate-by-gate

Concrete question: what homomorphisms can be efficiently computed? Is there anything except **CU, U*** ?

The determinant as a homomorphism

Given unitary circuit **Q**, can compute **det(Q)** by taking the determinant of each gate and multiplying

However, this ignores the role ancillas!

Ancillas

In general, when computing a unitary **U** using a circuit **Q**, **Q** may involve ancillas

Typically, ancillas are initialized to $|\mathbf{0}\rangle$ and returned to $|\mathbf{0}\rangle$ at the end

$$\mathbf{Q}\,(\,|\Psi\rangle|0\rangle\,) = (\,\mathbf{U}|\Psi\rangle\,)\,|0\rangle$$

$$\mathbf{Q} = \begin{pmatrix} \mathbf{U} & \\ & \mathbf{V} \end{pmatrix} \Rightarrow$$

V may be arbitrarily related to **U**

Det(Q) = Det(U)Det(V)

Det(Q) and **Det(U)** arbitrarily related

Ancilla-Respecting Homomorphisms

H’ is an ancilla-respecting implementation of a homomorphism **H** if:

$$\mathbf{H'}\begin{pmatrix} \mathbf{U} & \\ & \mathbf{V} \end{pmatrix} = \begin{pmatrix} \mathbf{H(U)} & \\ & \mathbf{J(U,V)} \end{pmatrix}$$

CU, U* are both implementations of themselves

No efficient ancilla-respecting homomorphisms beyond **CU, U***

Thm (this work): Let **H** be some *continuous* homomorphism. Then either:
- **H(U)** can be implemented by polynomially-many queries to **CU** or **CU***, or
- **H** has no efficient ancilla-respecting implementation

Proof for determinant: Suppose **det** had implementation **det'**

Let **W** be diagonal matrix with entries $\mathbf{d_i}$, $\mathbf{\Pi}$ be some permutation matrix

$\mathbf{\Pi W \Pi^\dagger}$ is diagonal, so can write

$$\mathbf{\Pi W \Pi^\dagger} = \begin{pmatrix} \mathbf{W_\Pi} & \\ & \mathbf{V_\Pi} \end{pmatrix}$$

No efficient ancilla-respecting homomorphisms beyond **CU, U***

Thm (this work): Let **H** be some *continuous* homomorphism. Then either:

- **H(U)** can be implemented by polynomially-many queries to **CU** or **CU***, or
- **H** has no efficient ancilla-respecting implementation

Proof for determinant:

$$\Pi W \Pi^\dagger = \begin{pmatrix} W_\Pi & \\ & V_\Pi \end{pmatrix}$$

Ancilla-respecting:

$$\det'(\Pi W \Pi^\dagger) = \begin{pmatrix} \mathrm{Det}(W_\Pi) & \\ & J(\Pi, W) \end{pmatrix}$$

No efficient ancilla-respecting homomorphisms beyond **CU, U***

Thm (this work): Let **H** be some *continuous* homomorphism. Then either:

- **H(U)** can be implemented by polynomially-many queries to **CU** or **CU***, or
- **H** has no efficient ancilla-respecting implementation

Proof for determinant:

$$\mathbf{det'(\Pi W \Pi^\dagger)} = \begin{pmatrix} \mathbf{det(W_\Pi)} & \\ & \mathbf{J(\Pi, W)} \end{pmatrix}$$

But also $\mathbf{det'(\Pi W \Pi^\dagger) = det'(\Pi)\, det'(W)\, det'(\Pi)^\dagger}$

➡ For any fixed **∏**, $\mathbf{det(W_\Pi)}$ is linear combination of entries of **det'(W)**

No efficient ancilla-respecting homomorphisms beyond **CU, U***

Thm (this work): Let **H** be some *continuous* homomorphism. Then either:

- **H(U)** can be implemented by polynomially-many queries to **CU** or **CU***, or
- **H** has no efficient ancilla-respecting implementation

Proof for determinant:

For any fixed $\boldsymbol{\Pi}$, $\mathbf{det(W_{\Pi})}$ is linear combination of entries of **det'(W)**

$\mathbf{det(W_{\Pi}) = d_{i1}\, d_{i2} \ldots}$ for arbitrary subsets **{i1, i2,… }**

$\mathbf{dim(\ \{\ det(W_{\Pi})\ \}_{\Pi}\ ) = (\ dim(Q)\ choose\ dim(U)\ ) \approx 2^{2^n}}$

For even 1-qubit ancilla, can take to be $\mathbf{2 \times 2^n}$ $\mathbf{2^n}$ for n-qubit unitary

No efficient ancilla-respecting homomorphisms beyond **CU, U***

Thm (this work): Let **H** be some *continuous* homomorphism. Then either:

- **H(U)** can be implemented by polynomially-many queries to **CU** or **CU***, or
- **H** has no efficient ancilla-respecting implementation

Proof for determinant:

For any fixed Π, $\mathbf{det}(W_\Pi)$ is linear combination of entries of **det'(W)**

$\mathbf{dim}(\{\mathbf{det}(W_\Pi)\}_\Pi) \approx 2^{2^n}$ ➡ **det'(W)** needs at least $\approx 2^{2^n}$ entries

det'(W) is at least $2^{2^n/2} \times 2^{2^n/2}$

det'(W) is a unitary on at least 2^{n-1} qubits ➡ Inefficient!!!

Ancilla complexity

Thm (this work): Suppose **PH** $\nsubseteq$ **BPP**. Then there is a family of quantum circuits that can be computed efficiently with 2 ancillas, but not 0 ancillas

In particular, obtain a *quantum* complexity separation from a purely classical separation

Thm (this work): Suppose **PH** $\subsetneq$ **BPP**. Then there is a family of quantum circuits that can be computed efficiently with 2 ancillas, but not 0 ancillas

Proof idea: Let **C** be a classical log-depth circuit

[Cleve'91]: $\mathbf{U_C|x,b\rangle = |x,b\oplus C(x)\rangle}$
implemented efficiently using 2 ancillas

Now suppose $\mathbf{U_C}$ can be implemented efficiently using 0 ancillas

➡ $\mathbf{det(U_C) = (-1)^{parity(C)}}$ can be computed efficiently (classically), just given **C**

➡ $\mathbf{\oplus P \subseteq P}$

Thm (this work): Suppose $\mathbf{PH} \not\subseteq \mathbf{BPP}$. Then there is a family of quantum circuits that can be computed efficiently with 2 ancillas, but not 0 ancillas

Proof idea:

[Toda'91]: $\mathbf{PH} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$

$\Downarrow$

So if $\oplus\mathbf{P} \subseteq \mathbf{P}$, then $\mathbf{PH} \subseteq \mathbf{BPP}$, a contradiction

Errors

So far, have assumed perfect implementations of unitaries

But in general, may only have approximate implementations. How should the errors be modeled?

Model of [Bostanci-Efron-Metger-Poremba-Qian-Yuen'23]:
for any desired inverse-poly error **ε**, can construct circuit **Q**
that is **ε**-close to **U** (diamond distance as quantum channel)

Why inverse poly, and not negligible errors or even exponential?

Inverse poly good enough for many applications, but often seems less than what techniques give us and what we may want/need

Consider family of unitaries $\mathbf{\{U_k\}_k}$ that is a PRU (e.g. CPFC')

Suppose we implement $\mathbf{U_k}$ using concrete circuit $\mathbf{Q_k}$, that has inverse-poly error $\boldsymbol{\varepsilon}$

⬇

$\mathbf{Q_k}$ is actually *insecure.* Adversary can make **poly(1/ε)** queries, overall error will be ≈1

Takeaway: For cryptographic primitives, should really insist on at least negligible errors, in practice exponentially-small errors

Remark: Even in classical world, sampling tasks (e.g. discrete Gaussians for lattice crypto) usually expected to have *exponentially-small* errors

$\Downarrow$

Exponentially-small errors may be a better modeling choice in many settings

Clarification:

Thm (this work): Suppose **PH** $\not\subseteq$ **BPP**. Then there is a family of quantum circuits that can be computed efficiently with 2 ancillas, but not 0 ancillas ***(in exponentially-small error model)***

Thanks!