How to Model Unitary Oracles

Mark Zhandry (NTT Research & Stanford University)

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

First pass at formalization only recently, 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):

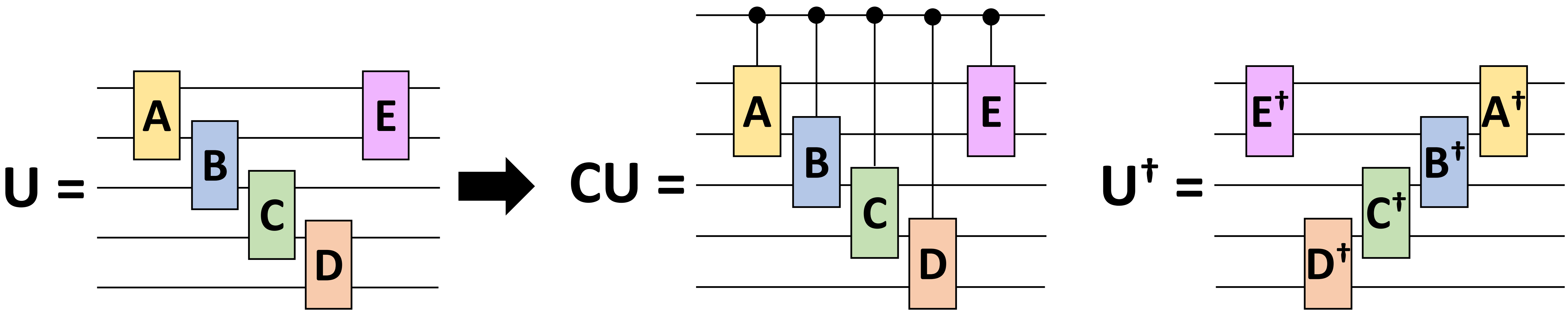
- Efficient implementation = small circuit that implements **U *including global phase***, ideally to within ***exponentially-small error***

Our thesis (subject to further scrutiny):

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

- Black box constructions and reductions should allow **controlling CU, (controlled) inverses $CU^{\dagger}$,** as well as **conjugates CU^{*} and transposes CU^{T}**,

How to implement **CU, U†**



Common 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]
- …

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

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

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

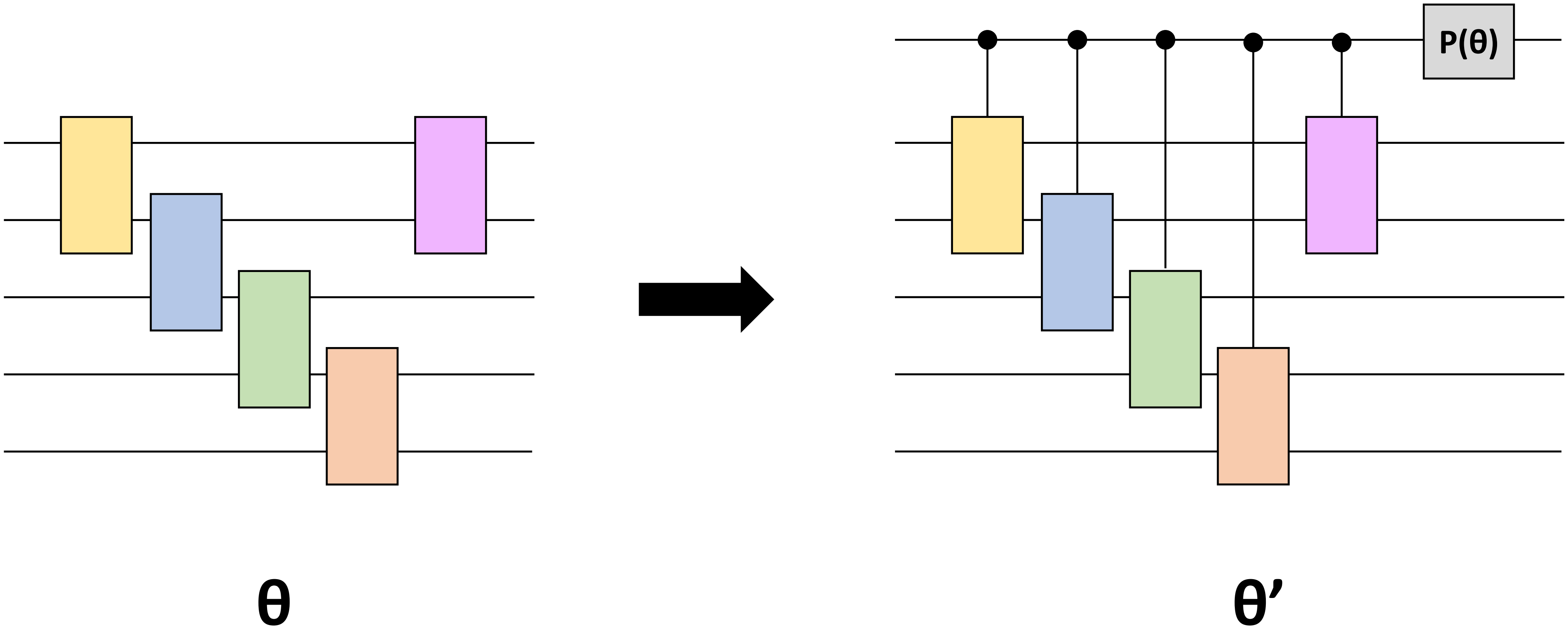
Caveat: Global Phase

If we want “efficient implementation” to facilitate controlling, need to know global phase

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

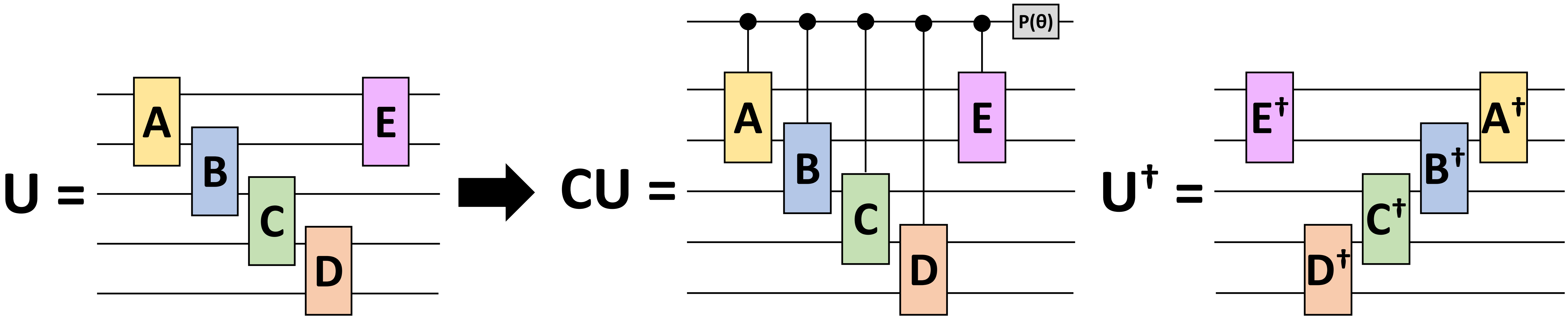
Fortunately, we generally know the phase **θ**

How to actually implement **CU**

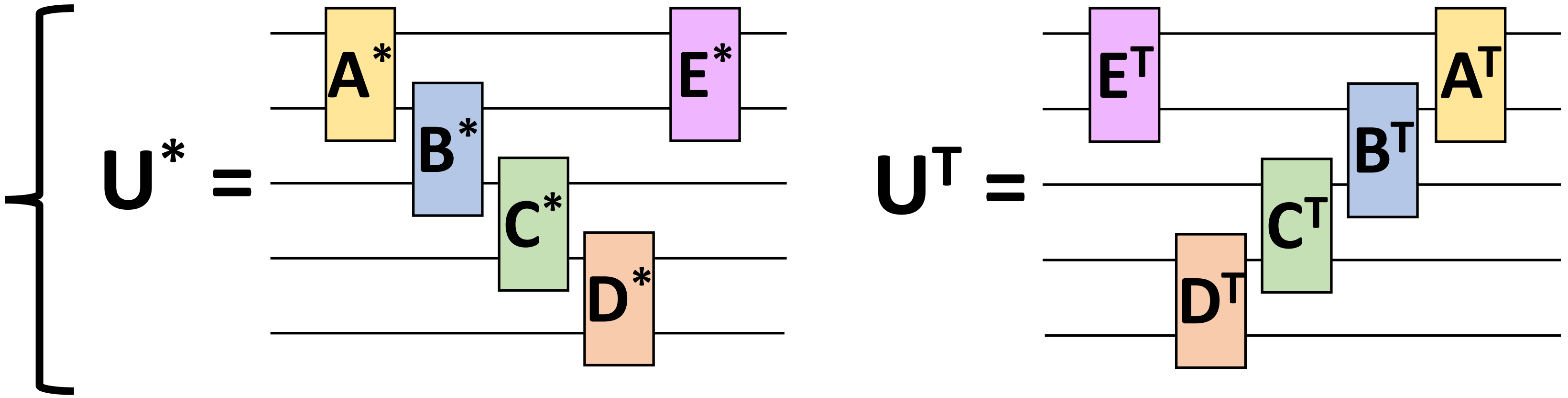


(comes from implementing **P(θ)**)

How to implement **CU, U†, U*, U^T**



Much less common in the literature



Just because we can model $\mathbf{U}^*$**,** $\mathbf{U}^\mathbf{T}$, should we?

This work: several results supporting that yes, we should

Black-box separations

We can implement **CU, $U^\dagger$, U^*, U^T** in real world, so any oracle separation including these is "closer" to reality

Thm (informal): Cannot construct quantum oracle **U** from any classical oracle **C**, unless black-box unitary includes **$U^\dagger$, U^*, U^T** (with caveats; also note lack of controlling)

Note: this theorem gives necessary conditions, but no indication how to actually build **U**

Implication: cannot generically lift unitary oracle separations to classical oracle separations unless this modelling is followed

Black-box reductions

Likewise, reductions utilizing a unitary adversary **U**
may make use of **CU,** $\mathbf{U^\dagger}$**,** $\mathbf{U^*}$**,** $\mathbf{U^T}$

Thm (informal): Under certain (admittedly contrived) conditions, can extend the length of 1-time pseudorandom state generators by 1 qubit. Reduction inherently require $\mathbf{U^*}$

Public Random Unitary Model

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 (secret) function **f**

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

Interesting question: can making **F,P** public allow us to construct public random unitaries from random functions/permutations?

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

Public Random Unitary Model

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

Thm (this work): When **q=2**, **CPFC'** is not *****-secure

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

(Anti-) Homomorphisms on Unitaries

CU, U* are *homomorphisms* on unitaries

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

(Anti-) Homomorphisms on Unitaries

CU, U^* are *homomorphisms* on unitaries

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

U^T, $U^\dagger$ are *anti-homomorphisms*

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

All anti-homomorphisms are the inverse of some homomorphism

(Anti-) Homomorphisms on Unitaries

CU, U^* are *homomorphisms* on unitaries

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

U^T, $U^\dagger$ are *anti-homomorphisms*

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

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*** ?

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 implementation for unitaries *using even 1 ancilla qubit*

Most interesting unitaries use ancillas

Ancilla complexity

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

Idea: determinants are a homomorphism that works on circuits with 0 ancillas, but not on circuits using ancillas

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

Thanks!