Quantum Complexity

A Brief Overview

Fernando Granha Jeronimo

(Updated: 04/21/25)

How powerful are quantum computers?

Are there advantages of quantum computers over classical ones?

How powerful are quantum computers?

Are there advantages of quantum computers over classical ones?

How powerful are quantum computers?

Are there limitations to quantum computers?

Complexity theory sheds light on these questions!

Are there advantages of quantum computers over classical ones?

How powerful are quantum computers?

Are there limitations to quantum computers?

Can quantum computers solve NP-complete problems efficiently?

Can quantum computers solve NP-complete problems efficiently?

So far the answer seems to be no

Can quantum computers solve NP-complete problems efficiently?

So far the answer seems to be no, but there is no formal proof of this…

Can quantum computers solve NP-complete problems efficiently?

So far the answer seems to be no, but there is no formal proof of this…

Such proof seems beyond our current techniques!

However, more concretely, we can ask…

However, more concretely, we can ask…

How many queries are needed for unstructured search in the black-box model?

However, more concretely, we can ask…

How many queries are needed for unstructured search in the black-box model?

In the black-box model, we can prove lower bounds!

However, more concretely, we can ask…

How many queries are needed for unstructured search in the black-box model?

In the black-box model, we can prove lower bounds!

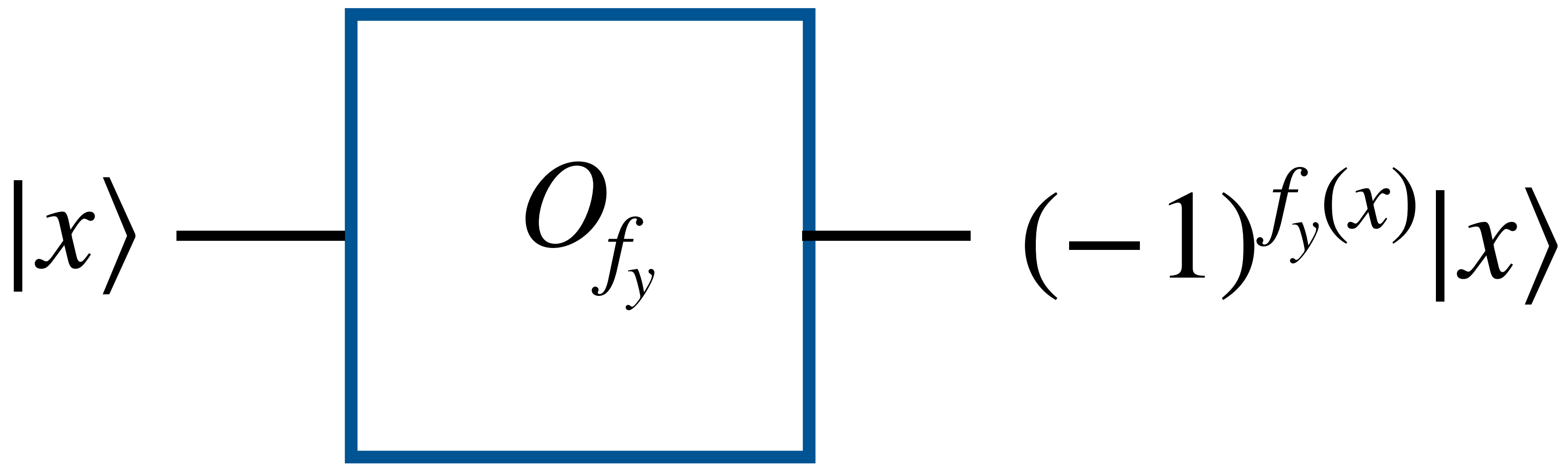
We will see our first lower bound (or impossibility) result

Consider the Phase Oracles

$\forall y \in \{0,1\}^n, \quad f_y \colon \{0,1\}^n \to \{0,1\}$ $\qquad$ $f_y(x) = 1 \iff x = y$

Consider the Phase Oracles

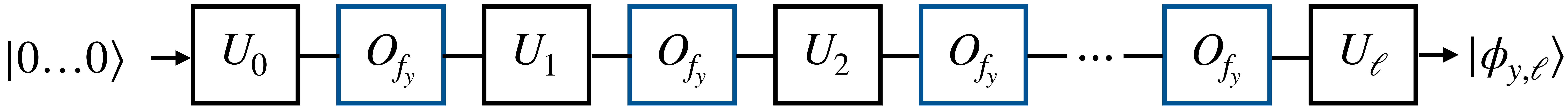
$\forall y \in \{0,1\}^n, \quad f_y \colon \{0,1\}^n \to \{0,1\}$ $\qquad\qquad$ $f_y(x) = 1 \iff x = y$



$$O_{f_y}|x\rangle = (-1)^{f_y(x)}|x\rangle$$

Quantum Query Algorithm

An arbitrary ℓ-query quantum algorithm can be expressed as



for some fixed choice of unitaries $U_0, U_1, \ldots, U_\ell$

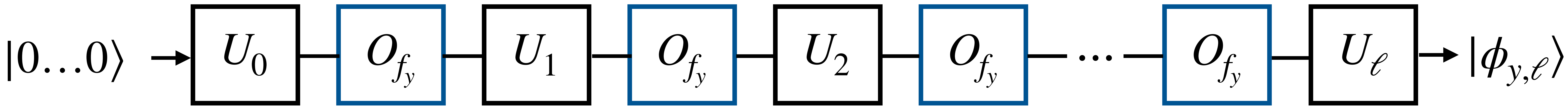
Hybrid Method

Hybrid Method

Break the analysis into a sequence of slightly different computations

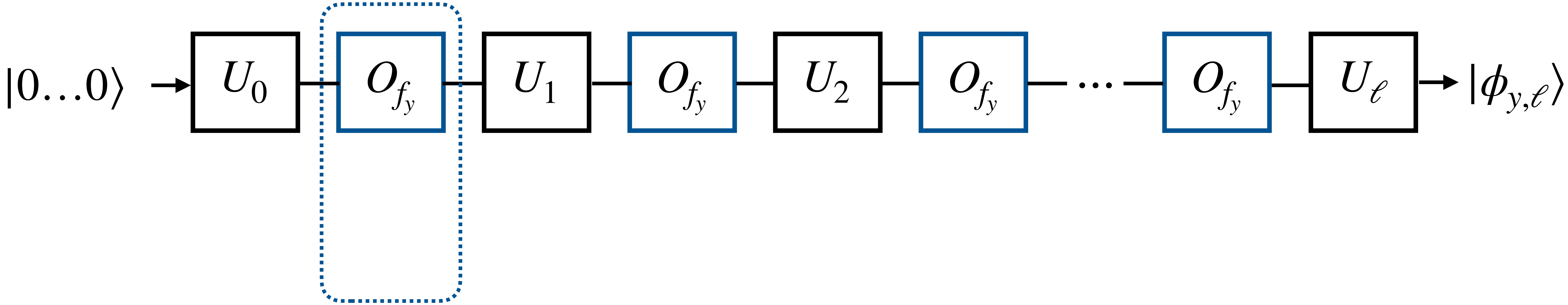
Hybrid Method

Break the analysis into a sequence of slightly different computations



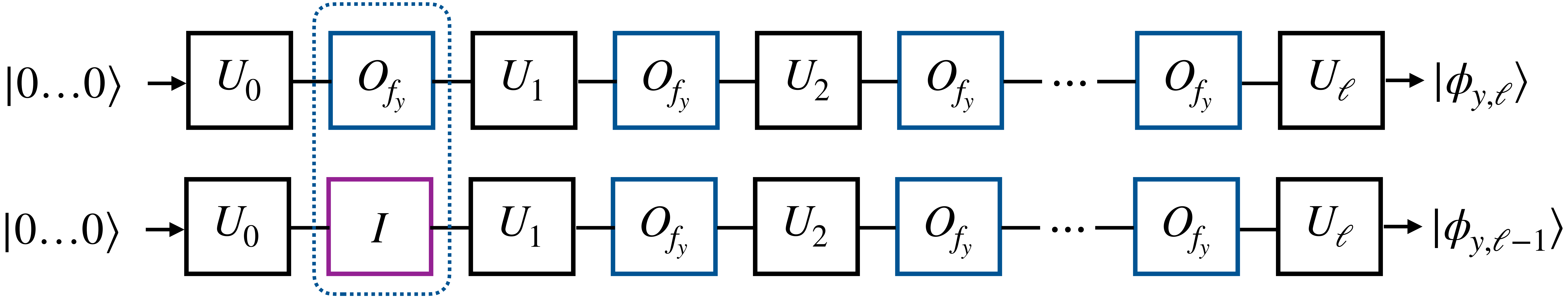
Hybrid Method

Break the analysis into a sequence of slightly different computations

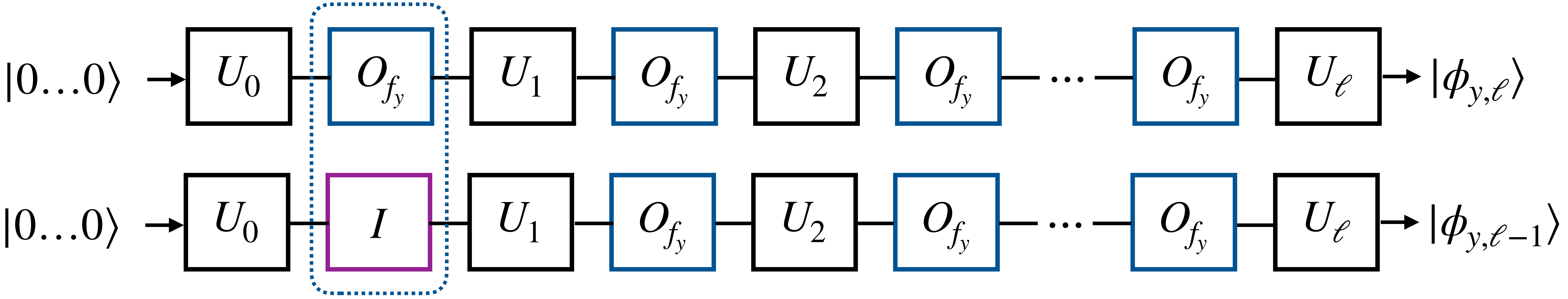


Hybrid Method

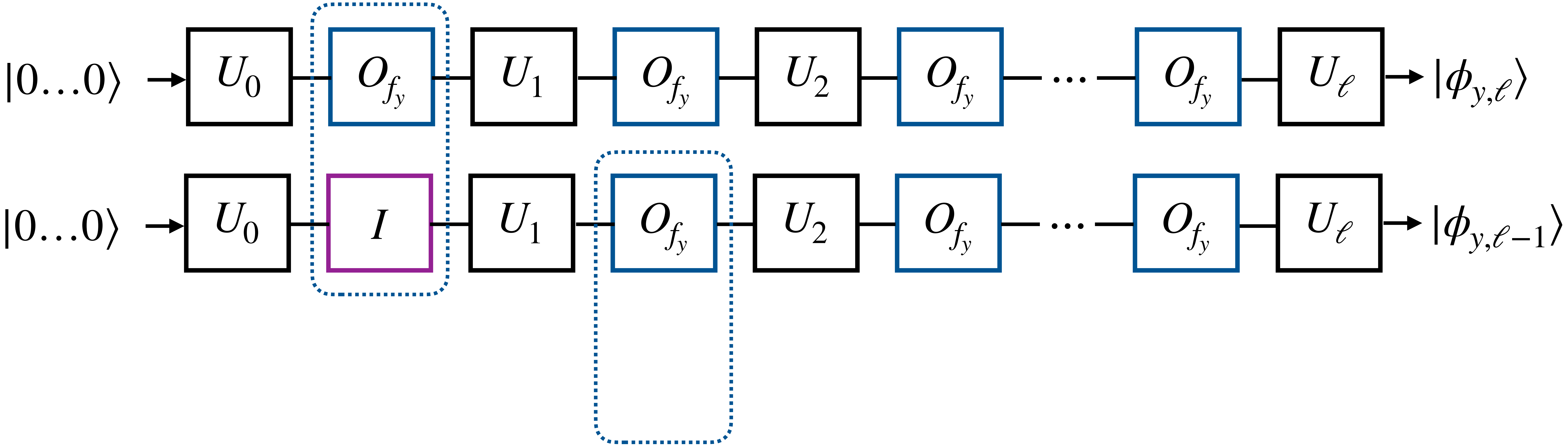
Break the analysis into a sequence of slightly different computations



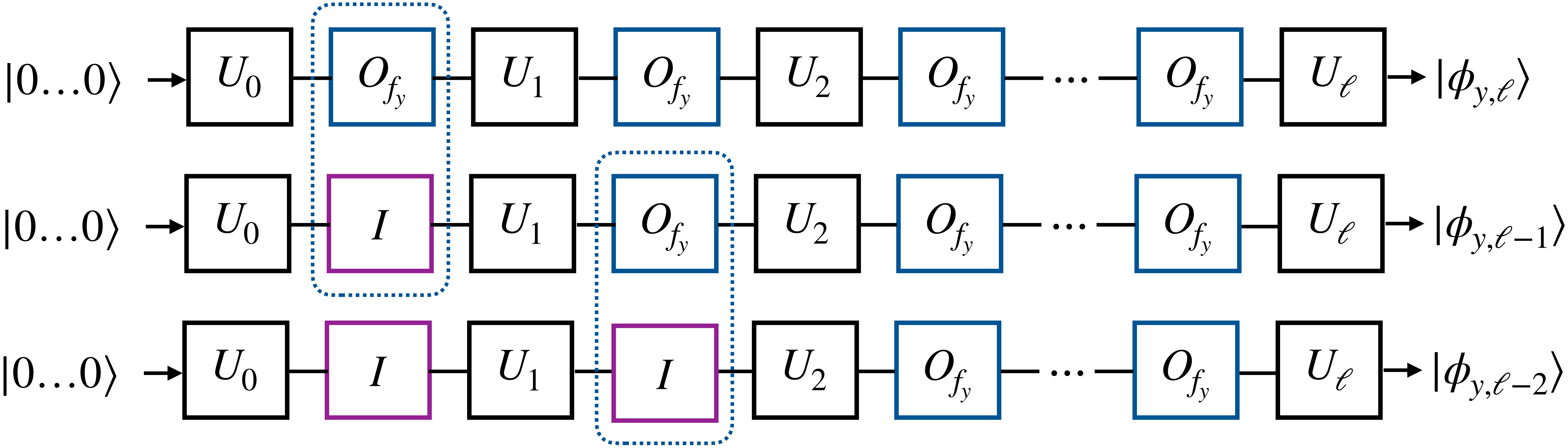
Hybrid Method

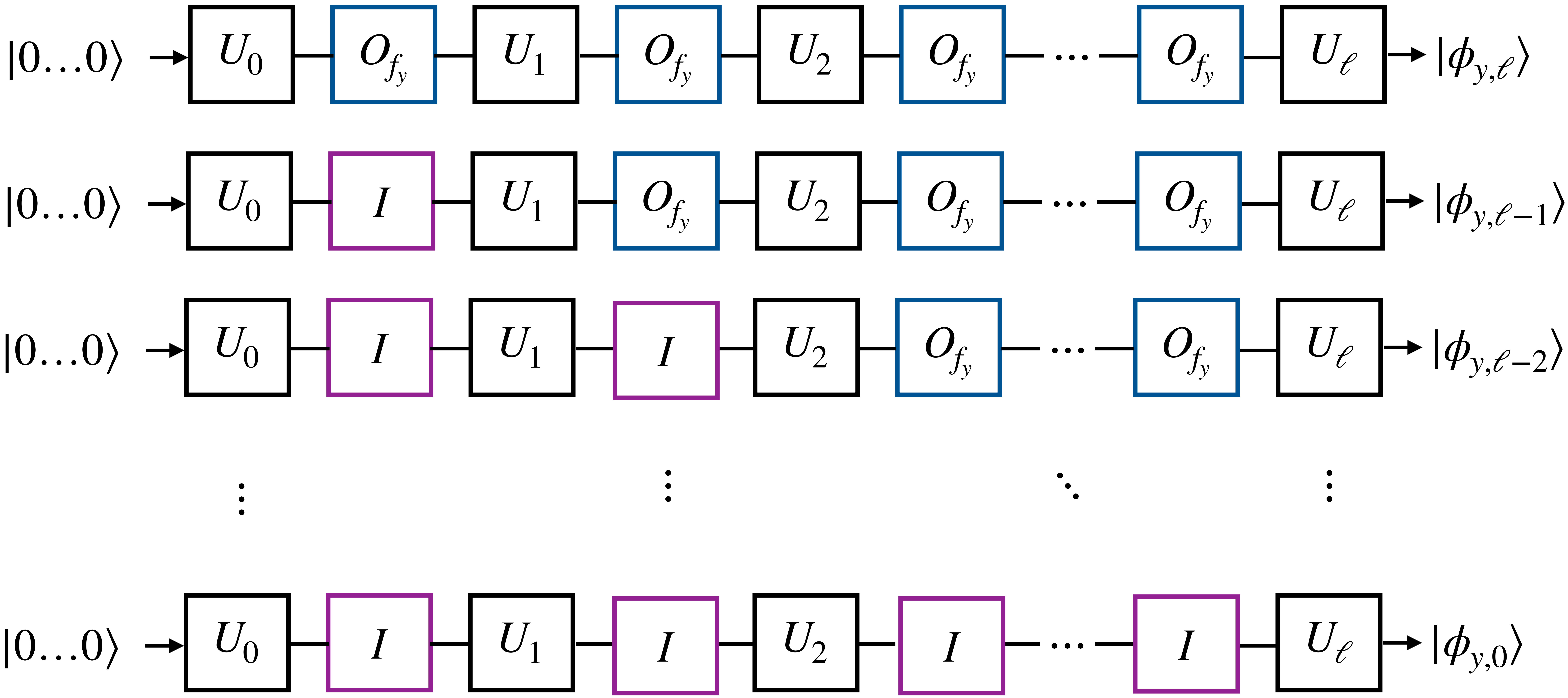


Hybrid Method

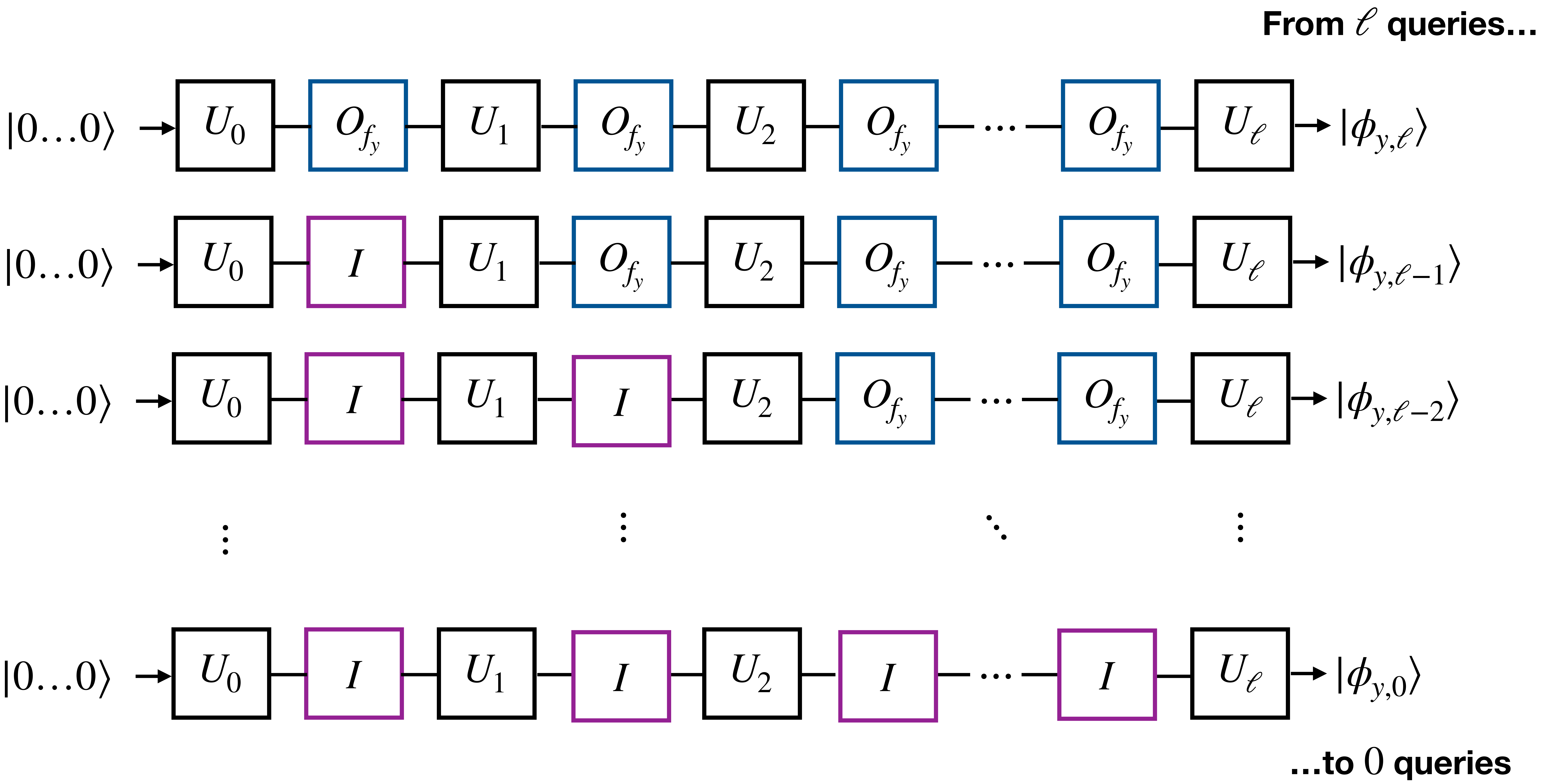


Hybrid Method

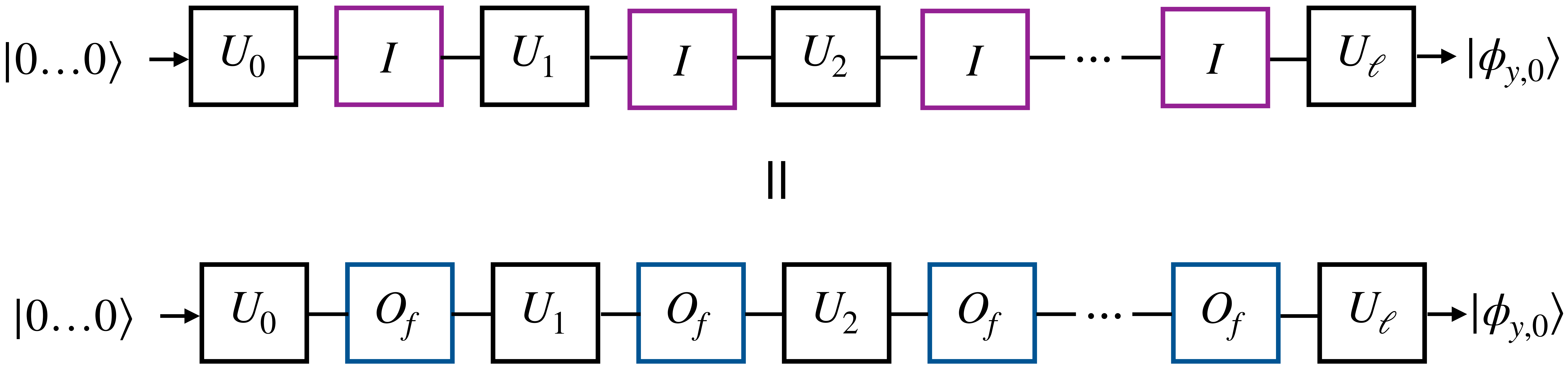




$|0\ldots0\rangle$
U_0
O_{f_y}
U_1
U_2
U_ℓ
I
$|\phi_{y,\ell}\rangle$
$|\phi_{y,\ell-1}\rangle$
$|\phi_{y,\ell-2}\rangle$
$|\phi_{y,0}\rangle$


From ℓ queries...
$|0\ldots0\rangle$
U_0
O_{f_y}
U_1
U_2
U_ℓ
$|\phi_{y,\ell}\rangle$
I
$|\phi_{y,\ell-1}\rangle$
$|\phi_{y,\ell-2}\rangle$
$|\phi_{y,0}\rangle$
...to 0 queries

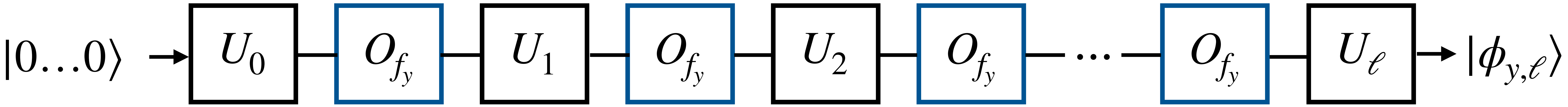
Equal Computations



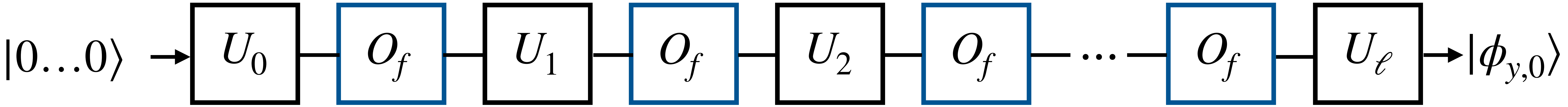
where $f\colon \{0,1\}^n \to \{0,1\}$ **is the identically zero function**

Our Goal

Show that there is some choice of $y \in \{0,1\}^n$ such that the states



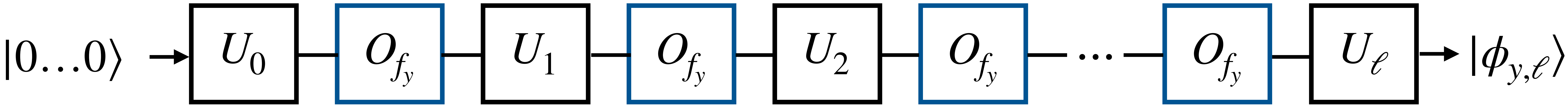
and



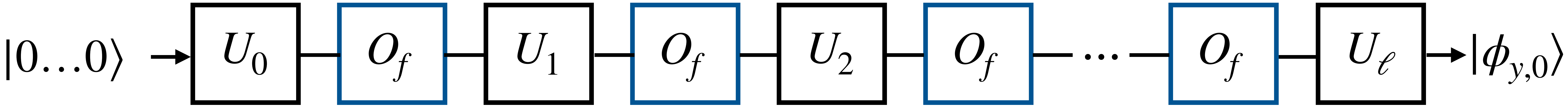
(where $f\colon \{0,1\}^n \to \{0,1\}$ is the identically zero function)

Our Goal

Show that there is some choice of $y \in \{0,1\}^n$ such that the states



and



(where $f\colon \{0,1\}^n \to \{0,1\}$ is the identically zero function)

are very similar unless the number of queries ℓ is sufficiently large

Our Goal

Show that there is some choice of $y \in \{0,1\}^n$ such that the states

$$|0\ldots0\rangle \to \boxed{U_0} - \boxed{O_{f_y}} - \boxed{U_1} - \boxed{O_{f_y}} - \boxed{U_2} - \boxed{O_{f_y}} - \cdots - \boxed{O_{f_y}} - \boxed{U_\ell} \to |\phi_{y,\ell}\rangle$$

and

$$\approx$$

$$|0\ldots0\rangle \to \boxed{U_0} - \boxed{O_{f}} - \boxed{U_1} - \boxed{O_{f}} - \boxed{U_2} - \boxed{O_{f}} - \cdots - \boxed{O_{f}} - \boxed{U_\ell} \to |\phi_{y,0}\rangle$$

(where $f\colon \{0,1\}^n \to \{0,1\}$ is the identically zero function)

are very similar unless the number of queries ℓ is sufficiently large

Our Goal

Show that there is some choice of $y \in \{0,1\}^n$ such that the states

$$|0\ldots0\rangle \rightarrow \boxed{U_0} - \boxed{O_{f_y}} - \boxed{U_1} - \boxed{O_{f_y}} - \boxed{U_2} - \boxed{O_{f_y}} - \cdots - \boxed{O_{f_y}} - \boxed{U_\ell} \rightarrow |\phi_{y,\ell}\rangle$$

and $\quad \approx$

$$|0\ldots0\rangle \rightarrow \boxed{U_0} - \boxed{O_f} - \boxed{U_1} - \boxed{O_f} - \boxed{U_2} - \boxed{O_f} - \cdots - \boxed{O_f} - \boxed{U_\ell} \rightarrow |\phi_{y,0}\rangle$$

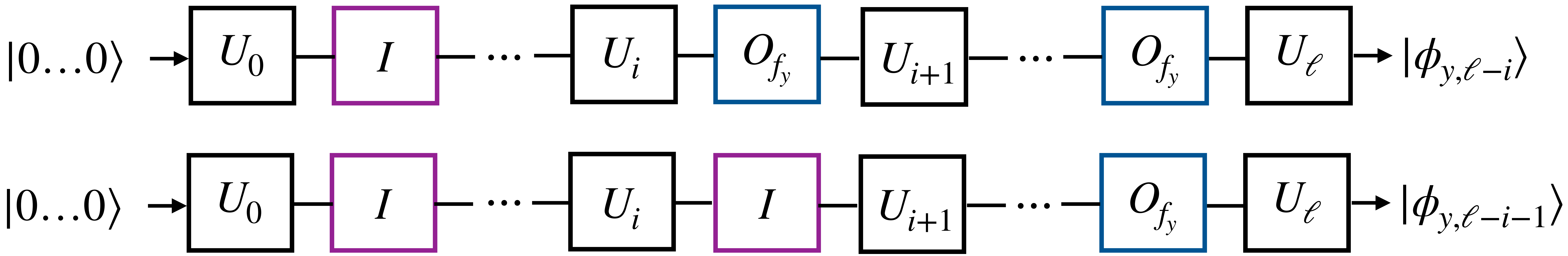
(where $f\colon \{0,1\}^n \rightarrow \{0,1\}$ is the identically zero function)

are very similar unless the number of queries ℓ is sufficiently large

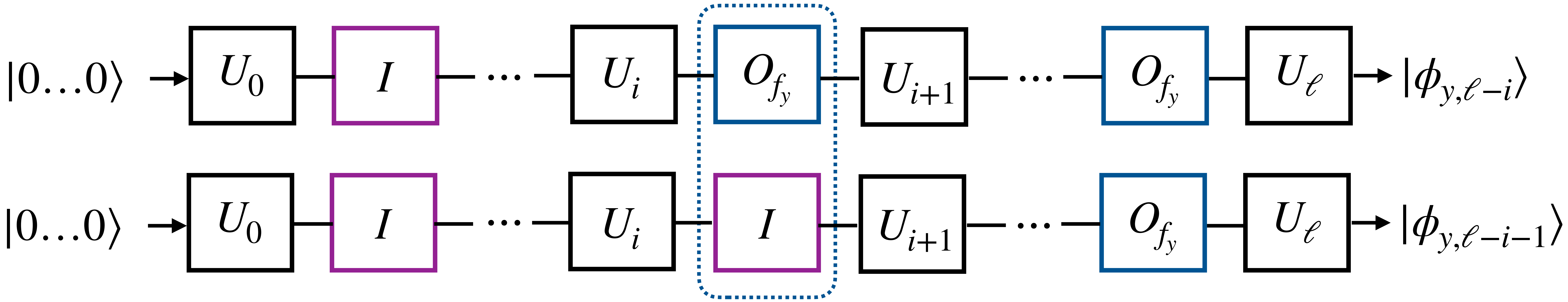
$$\Downarrow$$

The algorithm cannot distinguish well between the oracles $\boxed{O_{f_y}}$ and $\boxed{O_f}$

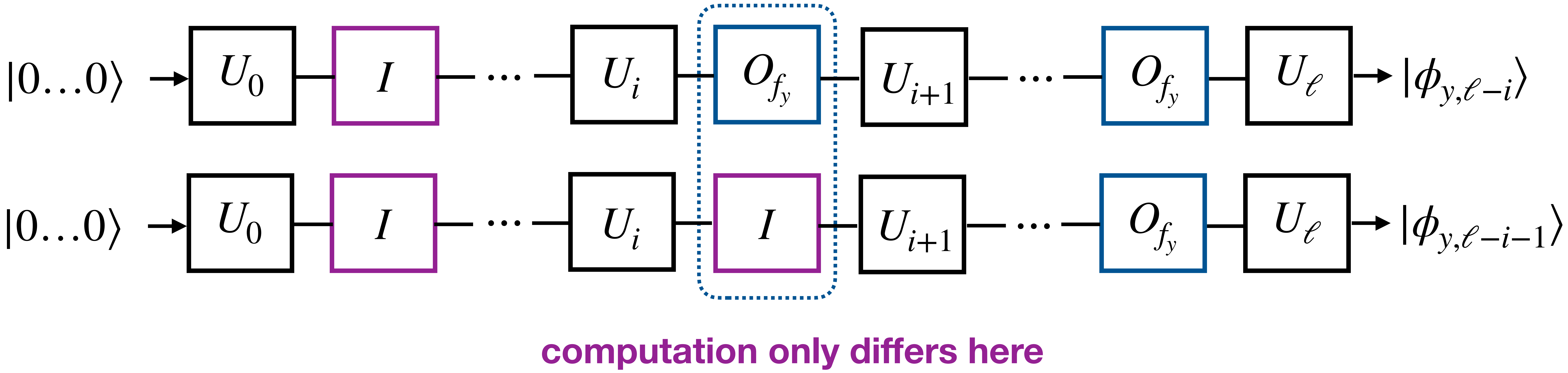
Consider Two Consecutive States



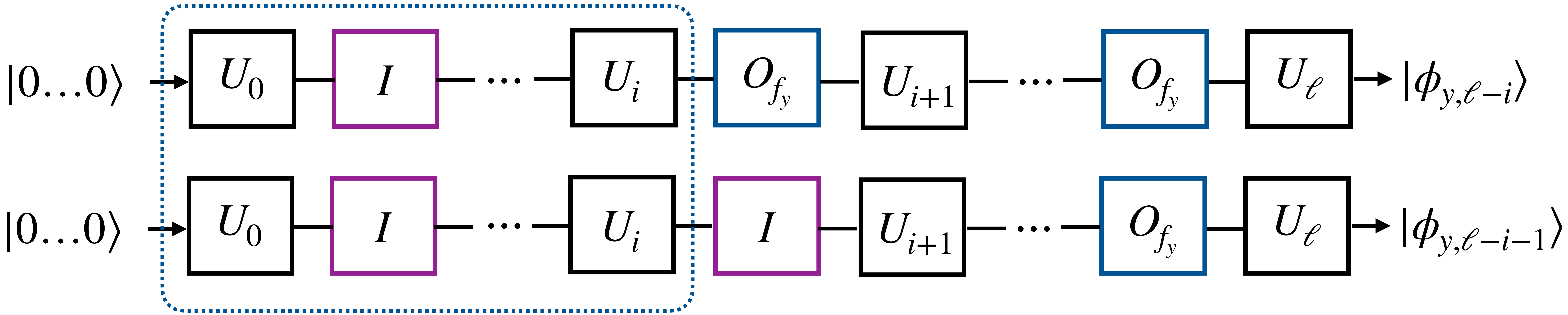
Consider Two Consecutive States



Consider Two Consecutive States



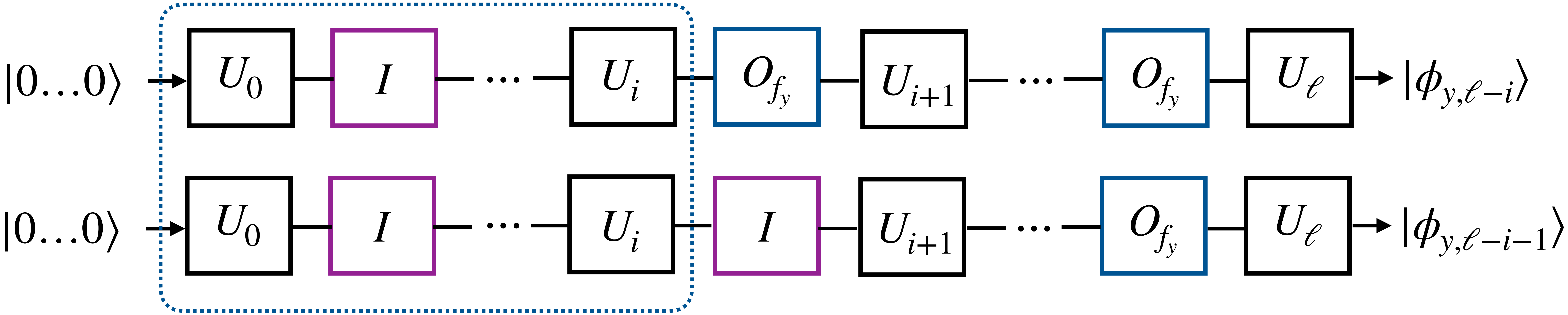
Same state up to this point



$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x\in\{0,1\}^n} \alpha_{i,x} |x\rangle$$

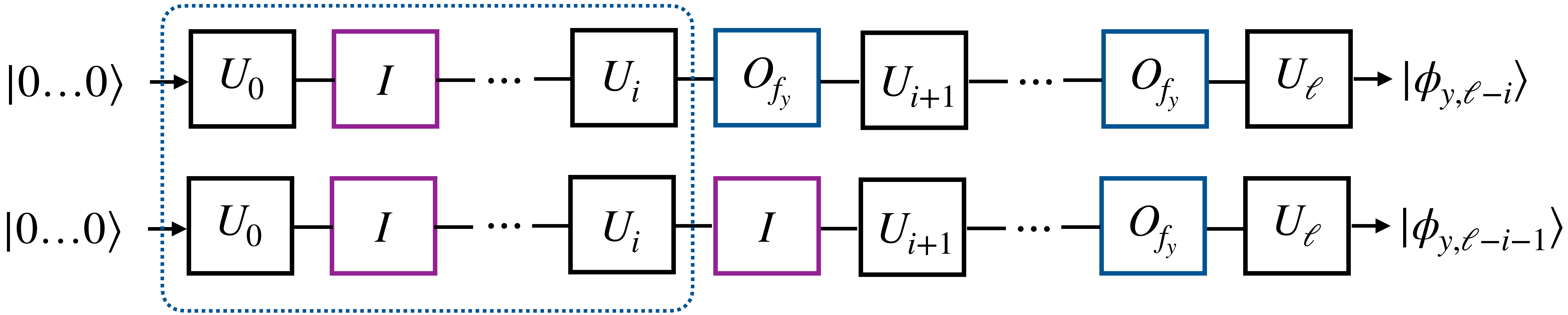
$$\sum_{x \in \{0,1\}^n} |\alpha_{i,x}|^2 = 1$$

Same state up to this point



$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x \in \{0,1\}^n} \alpha_{i,x} |x\rangle$$

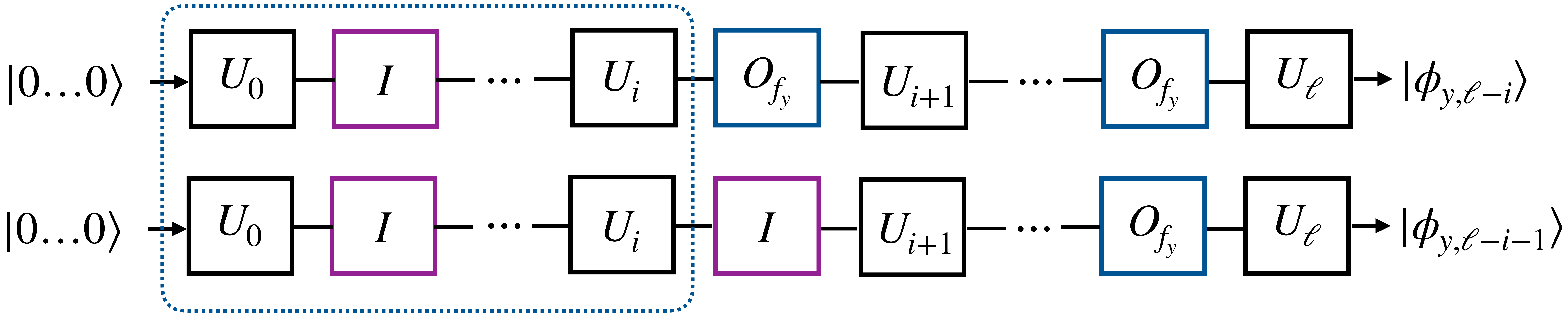
Same state up to this point



$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x \in \{0,1\}^n} \alpha_{i,x} |x\rangle$$

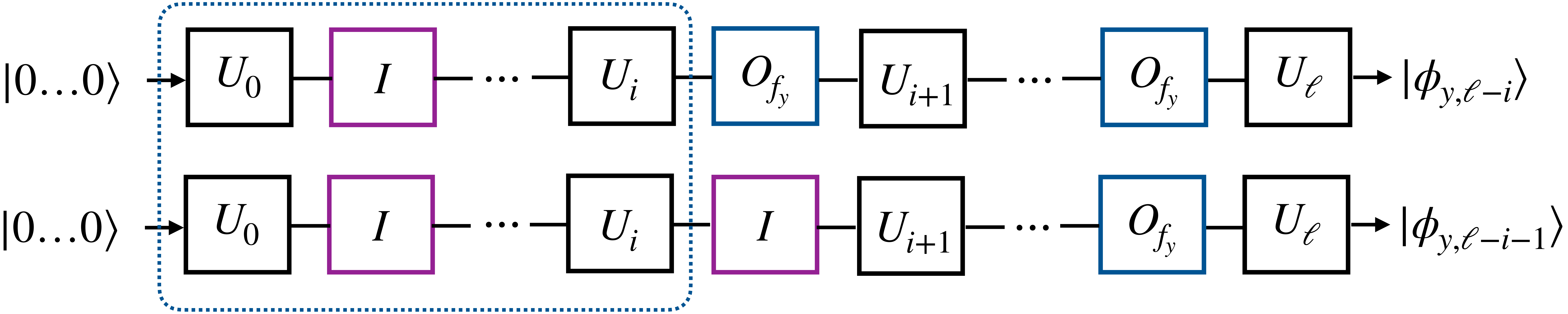
$$\left| I|\psi_i\rangle - O_{f_y}|\psi_i\rangle \right|_2$$

Same state up to this point



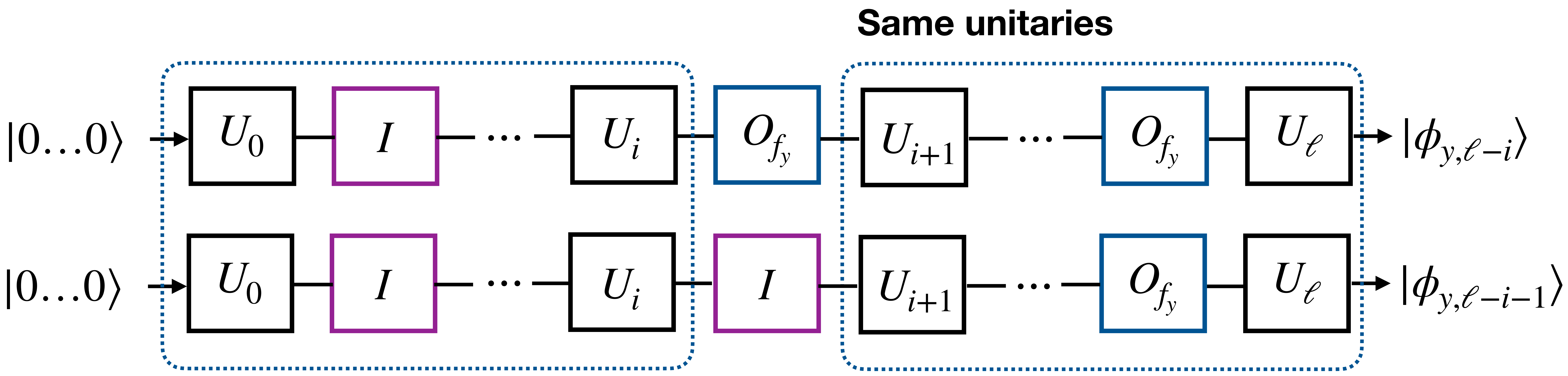
$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x \in \{0,1\}^n} \alpha_{i,x} |x\rangle$$

$$\left| I|\psi_i\rangle - O_{f_y}|\psi_i\rangle \right|_2 = 2|\alpha_{i,y}|$$



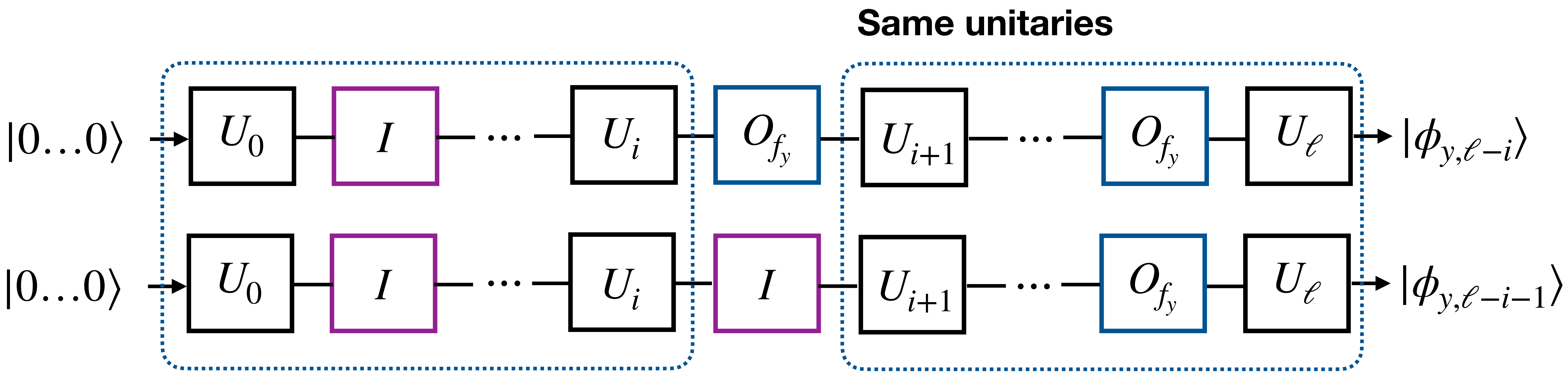
$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x\in\{0,1\}^n} \alpha_{i,x}|x\rangle$$

$$\Big| I|\psi_i\rangle - O_{f_y}|\psi_i\rangle \Big|_2 = 2|\alpha_{i,y}|$$



$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x \in \{0,1\}^n} \alpha_{i,x} |x\rangle$$

$$\Big| I|\psi_i\rangle - O_{f_y}|\psi_i\rangle \Big|_2 = 2|\alpha_{i,y}|$$



$$|\psi_i\rangle = U_i \cdots U_0 |0\ldots0\rangle = \sum_{x\in\{0,1\}^n} \alpha_{i,x}|x\rangle$$

$$\Big|I|\psi_i\rangle - O_{f_y}|\psi_i\rangle\Big|_2 = 2|\alpha_{i,y}| \; = \Big|\,|\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle\Big|_2$$

$$\boxed{2|\alpha_{i,y}| = \left\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \right\|_2}$$

By Triangle inequality,

$$\left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \left\| |\phi_{y,\ell}\rangle - |\phi_{y,\ell-1}\rangle \right\|_2 + \left\| |\phi_{y,\ell-1}\rangle - |\phi_{y,\ell-2}\rangle \right\|_2 + \cdots + \left\| |\phi_{y,1}\rangle - |\phi_{y,0}\rangle \right\|_2$$

$$\boxed{2|\alpha_{i,y}| = \Big| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big|_2}$$

By Triangle inequality,

$$\begin{aligned} \Big| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big|_2 &\leq \Big| |\phi_{y,\ell}\rangle - |\phi_{y,\ell-1}\rangle \Big|_2 + \Big| |\phi_{y,\ell-1}\rangle - |\phi_{y,\ell-2}\rangle \Big|_2 + \cdots + \Big| |\phi_{y,1}\rangle - |\phi_{y,0}\rangle \Big|_2 \\ &= \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}| \end{aligned}$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\boxed{2|\alpha_{i,y}| = \Big|\, |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big|_2}$$

$$\boxed{\Big|\, |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \Big|\, |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big|_2 \leq \frac{1}{2^n} \sum_{y \in \{0,1\}^n} \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\frac{1}{2^n} \sum_{y\in\{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|$$

$$= \frac{1}{2^n} 2 \sum_{i=0}^{\ell-1} \sum_{y\in\{0,1\}^n} |\alpha_{i,y}|$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\frac{1}{2^n} \sum_{y\in\{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{1}{2^n} \sum_{y\in\{0,1\}^n} \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|$$

$$= \frac{1}{2^n} 2 \sum_{i=0}^{\ell-1} \sum_{y\in\{0,1\}^n} |\alpha_{i,y}|$$

(Cauchy-Schwarz)

$$\leq \frac{1}{2^n} 2 \sum_{i=0}^{\ell-1} \sqrt{2^n} \sqrt{\sum_{y\in\{0,1\}^n} |\alpha_{i,y}|^2}$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\sum_{y\in\{0,1\}^n} |\alpha_{i,y}|^2 = 1$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\frac{1}{2^n}\sum_{y\in\{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{1}{2^n}\sum_{y\in\{0,1\}^n}\sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|$$

$$= \frac{1}{2^n}2\sum_{i=0}^{\ell-1}\sum_{y\in\{0,1\}^n} |\alpha_{i,y}|$$

(Cauchy-Schwarz) $$\leq \frac{1}{2^n}2\sum_{i=0}^{\ell-1}\sqrt{2^n}\sqrt{\sum_{y\in\{0,1\}^n} |\alpha_{i,y}|^2}$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\sum_{y \in \{0,1\}^n} |\alpha_{i,y}|^2 = 1$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we get

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{1}{2^n} \sum_{y \in \{0,1\}^n} \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|$$

$$= \frac{1}{2^n} 2 \sum_{i=0}^{\ell-1} \sum_{y \in \{0,1\}^n} |\alpha_{i,y}|$$

(Cauchy-Schwarz)

$$\leq \frac{1}{2^n} 2 \sum_{i=0}^{\ell-1} \sqrt{2^n} \sqrt{\sum_{y \in \{0,1\}^n} |\alpha_{i,y}|^2} = \frac{2\ell}{\sqrt{2^n}}$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \le \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we got

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \le \frac{2\ell}{\sqrt{2^n}}$$

$$\boxed{2|\alpha_{i,y}| = \Big\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \Big\|_2}$$

$$\boxed{\Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we got

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{2\ell}{\sqrt{2^n}} \quad \Longrightarrow \exists y \in \{0,1\}^n, \Big\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \Big\|_2 \leq \frac{2\ell}{\sqrt{2^n}}$$

$$\boxed{2|\alpha_{i,y}| = \left\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \right\|_2}$$

$$\boxed{\left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we got

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \frac{2\ell}{\sqrt{2^n}} \implies \exists y \in \{0,1\}^n, \left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \frac{2\ell}{\sqrt{2^n}}$$

$\ell \geq \Omega(\sqrt{2^n})$ **needed to distinguish these states with constant probability!**

$$\boxed{2|\alpha_{i,y}| = \left\| |\phi_{y,\ell-i}\rangle - |\phi_{y,\ell-i-1}\rangle \right\|_2}$$

$$\boxed{\left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \sum_{i=0}^{\ell-1} 2|\alpha_{i,y}|}$$

Averaging over the choice of y, we got

$$\frac{1}{2^n} \sum_{y \in \{0,1\}^n} \left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \frac{2\ell}{\sqrt{2^n}} \qquad \Longrightarrow \exists y \in \{0,1\}^n, \left\| |\phi_{y,\ell}\rangle - |\phi_{y,0}\rangle \right\|_2 \leq \frac{2\ell}{\sqrt{2^n}}$$

$\ell \geq \Omega(\sqrt{2^n})$ **needed to distinguish these states with constant probability!**

$\blacksquare$

Putting it all together, we showed

Putting it all together, we showed

[Bennett, Bernstein, Brassard, Vazirani'94]

We need $\Omega(\sqrt{2^n})$ queries to distinguish whether f is identically zero or not with probability $\Omega(1)$

Putting it all together, we showed

[Bennett, Bernstein, Brassard, Vazirani'94]

We need $\Omega(\sqrt{2^n})$ queries to distinguish whether f is identically zero or not with probability $\Omega(1)$

Grover's algorithm is best possible in the number of queries!

Are there more limitations to quantum computers?

How powerful are efficient quantum computers (the class BQP)?

Complexity Classes 101

Complexity Classes 101

We will first give an overview of some important complexity classes

Complexity Classes 101

We will first give an overview of some important complexity classes

https://complexityzoo.net/Complexity_Zoo

(Great resource to navigate the Zoo of complexity classes)

Classical Complexity Classes 101

P

Classical Complexity Classes 101

P	**The class of decision problems solvable by a deterministic polynomial time Turing machine**

Classical Complexity Classes 101

P **The class of decision problems solvable by a deterministic polynomial time Turing machine**

Examples:

Classical Complexity Classes 101

P **The class of decision problems solvable by a deterministic polynomial time Turing machine**

Examples: **Given a graph and two vertices, are these vertices connected?**

Classical Complexity Classes 101

P **The class of decision problems solvable by a deterministic polynomial time Turing machine**

Examples: **Given a graph and two vertices, are these vertices connected?**

DFS/BFS algorithm

Classical Complexity Classes 101

P The class of decision problems solvable by a deterministic polynomial time Turing machine

Examples: Given a graph and two vertices, are these vertices connected?

DFS/BFS algorithm

Given a number, is it prime or not?

Classical Complexity Classes 101

P **The class of decision problems solvable by a deterministic polynomial time Turing machine**

Examples: **Given a graph and two vertices, are these vertices connected?**

DFS/BFS algorithm

Given a number, is it prime or not?

Non-trivial deterministic primality testing algorithm (AKS algorithm)

Classical Complexity Classes 101

"Efficiently solvable (classically)"

P The class of decision problems solvable by a deterministic polynomial time Turing machine

Examples: Given a graph and two vertices, are these vertices connected?

DFS/BFS algorithm

Given a number, is it prime or not?

Non-trivial deterministic primality testing algorithm (AKS algorithm)

Quantum Complexity Classes 101

BQP

Quantum Complexity Classes 101

BQP — **The class of decision problems solvable by a polynomial time quantum machine**

Quantum Complexity Classes 101

BQP

“Efficiently solvable (quantumly), the quantum version of P”

The class of decision problems solvable by a polynomial time quantum machine

Quantum Complexity Classes 101

BQP

“Efficiently solvable (quantumly), the quantum version of P”

The class of decision problems solvable by a polynomial time quantum machine

Examples:

Quantum Complexity Classes 101

“Efficiently solvable (quantumly), the quantum version of P”

BQP

The class of decision problems solvable by a polynomial time quantum machine

Examples:

All problems in P

Quantum Complexity Classes 101

“Efficiently solvable (quantumly),
the quantum version of P”

BQP

The class of decision problems solvable by a polynomial time quantum machine

Examples:

All problems in P

Classical computation can be made reversible with at most a polynomial overhead

Quantum Complexity Classes 101

"Efficiently solvable (quantumly),
the quantum version of P"

BQP

The class of decision problems solvable by a polynomial time quantum machine

Examples:

All problems in P

Classical computation can be made reversible with at most a polynomial overhead

Given a number and a bound, does it have a prime factor smaller than the bound?

Quantum Complexity Classes 101

“Efficiently solvable (quantumly),
the quantum version of P”

BQP

The class of decision problems solvable by a polynomial time quantum machine

Examples:

All problems in P

Classical computation can be made reversible with at most a polynomial overhead

Given a number and a bound, does it have a prime factor smaller than the bound?

A decision version of Factoring (Shor’s algorithm)

Classical Complexity Classes 101

NP — **The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine**

Classical Complexity Classes 101

NP **The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine**

Examples:

Classical Complexity Classes 101

NP The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine

Examples:

All problems in P

Classical Complexity Classes 101

NP The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine

Examples:

All problems in P

We can simply ignore the proof and run the P machine

Classical Complexity Classes 101

NP The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine

Examples:

All problems in P

We can simply ignore the proof and run the P machine

Given the description of a boolean circuit, does it have an input that evaluates to 1?

Classical Complexity Classes 101

NP The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine

Examples:

All problems in P

We can simply ignore the proof and run the P machine

Given the description of a boolean circuit, does it have an input that evaluates to 1?

Circuit-SAT (well-known NP-complete problem)

Classical Complexity Classes 101

"Solution can be efficiently verified (classically)"

NP The class of decision problems having polynomial size proofs that can be verified by a polynomial-time deterministic Turing machine

Examples:

All problems in P

We can simply ignore the proof and run the P machine

Given the description of a boolean circuit, does it have an input that evaluates to 1?

Circuit-SAT (well-known NP-complete problem)

Quantum Complexity Classes 101

QMA

Quantum Complexity Classes 101

QMA **The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine**

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly),
the quantum version of NP"

QMA The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Quantum Complexity Classes 101

“Solution can be efficiently verified (quantumly),
the quantum version of NP”

QMA

The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

Quantum Complexity Classes 101

“Solution can be efficiently verified (quantumly),
the quantum version of NP”

QMA

The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

All problems in BQP

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly),
the quantum version of NP"

QMA

The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

All problems in BQP

We can simply ignore the quantum proof and run the BQP machine

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly),
the quantum version of NP"

QMA The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

All problems in BQP

We can simply ignore the quantum proof and run the BQP machine

Given a local Hamiltonian and values a and b (with $b - a \geq 1/\text{poly}(n)$),
is the ground state energy below a or above b?

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly), the quantum version of NP"

QMA The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

All problems in BQP

We can simply ignore the quantum proof and run the BQP machine

Given a local Hamiltonian and values a and b (with $b - a \geq 1/\text{poly}(n)$), is the ground state energy below a or above b?

The local Hamiltonian (well-known QMA-complete problem)

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly), the quantum version of NP"

QMA

The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Examples:

All problems in BQP

We can simply ignore the quantum proof and run the BQP machine

Given a local Hamiltonian and values a and b (with $b - a \geq 1/\text{poly}(n)$), is the ground state energy below a or above b?

The local Hamiltonian (well-known QMA-complete problem)

(this problem requires some time to be properly introduced)

Quantum Complexity Classes 101

"Solution can be efficiently verified (quantumly), the quantum version of NP"

QMA

The class of (promise) problems having polynomial-size quantum proofs that can be verified by a BQP machine

Quantum Merlin-Arthur

Examples:

All problems in BQP

We can simply ignore the quantum proof and run the BQP machine

Given a local Hamiltonian and values a and b (with $b - a \geq 1/\text{poly}(n)$), is the ground state energy below a or above b?

The local Hamiltonian (well-known QMA-complete problem)

(this problem requires some time to be properly introduced)

Classical Complexity Classes 101

PSPACE

Classical Complexity Classes 101

PSPACE **The class of decision problems solvable by <u>polynomial space</u> Turing machines**

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by polynomial space Turing machines

EXP

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by polynomial space Turing machines

EXP The class of decision problems solvable by exponential time Turing machines

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by <u>polynomial space</u> Turing machines

EXP The class of decision problems solvable by <u>exponential time</u> Turing machines

BPP

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by polynomial space Turing machines

EXP The class of decision problems solvable by exponential time Turing machines

BPP The class of decision problems solvable by probabilistic polynomial time Turing machines

Classical Complexity Classes 101

PSPACE	**The class of decision problems solvable by <u>polynomial space</u> Turing machines**
EXP	**The class of decision problems solvable by <u>exponential time</u> Turing machines**
BPP	**The class of decision problems solvable by <u>probabilistic polynomial time</u> Turing machines**
MA	

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by polynomial space Turing machines

EXP The class of decision problems solvable by exponential time Turing machines

BPP The class of decision problems solvable by probabilistic polynomial time Turing machines

MA The class of decision problems having polynomial size proofs that can be verified by a BPP machine

Classical Complexity Classes 101

PSPACE The class of decision problems solvable by <u>polynomial space</u> Turing machines

EXP The class of decision problems solvable by <u>exponential time</u> Turing machines

BPP The class of decision problems solvable by <u>probabilistic polynomial time</u> Turing machines

MA The class of decision problems having polynomial size proofs that can be verified by a BPP machine

<u>M</u>erlin-<u>A</u>rthur

Complexity Classes 101

Major open problem in Computer Science and Mathematics

P *vs* **NP**

Complexity Classes 101

Major open problem in Computer Science and Mathematics

"Efficiently solvable (classically)"

P vs **NP**

"Solution can be efficiently verified (classically)"

Complexity Classes 101

Major open problem in Computer Science and Mathematics

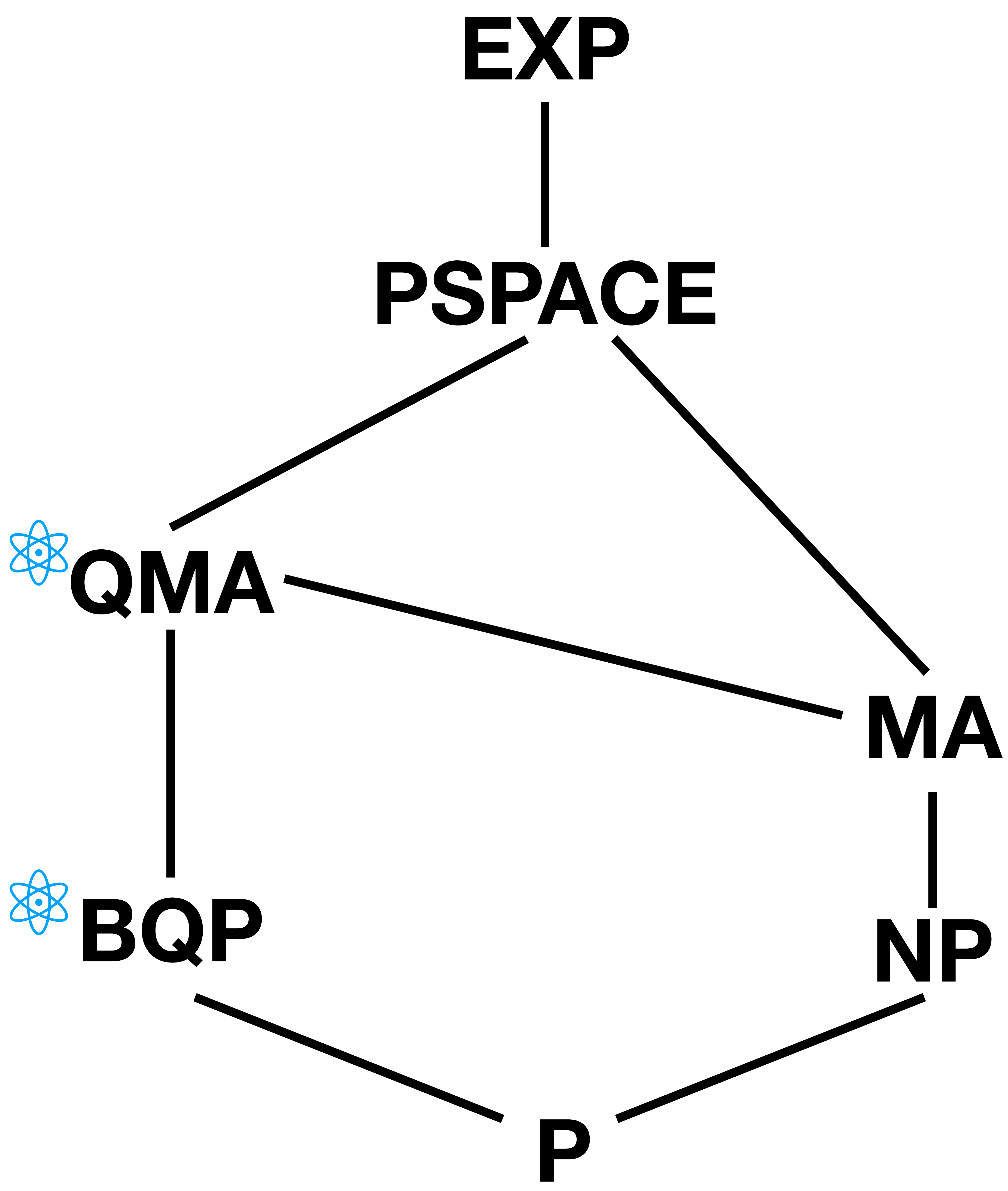
"Efficiently solvable (classically)"

P *vs* **NP**

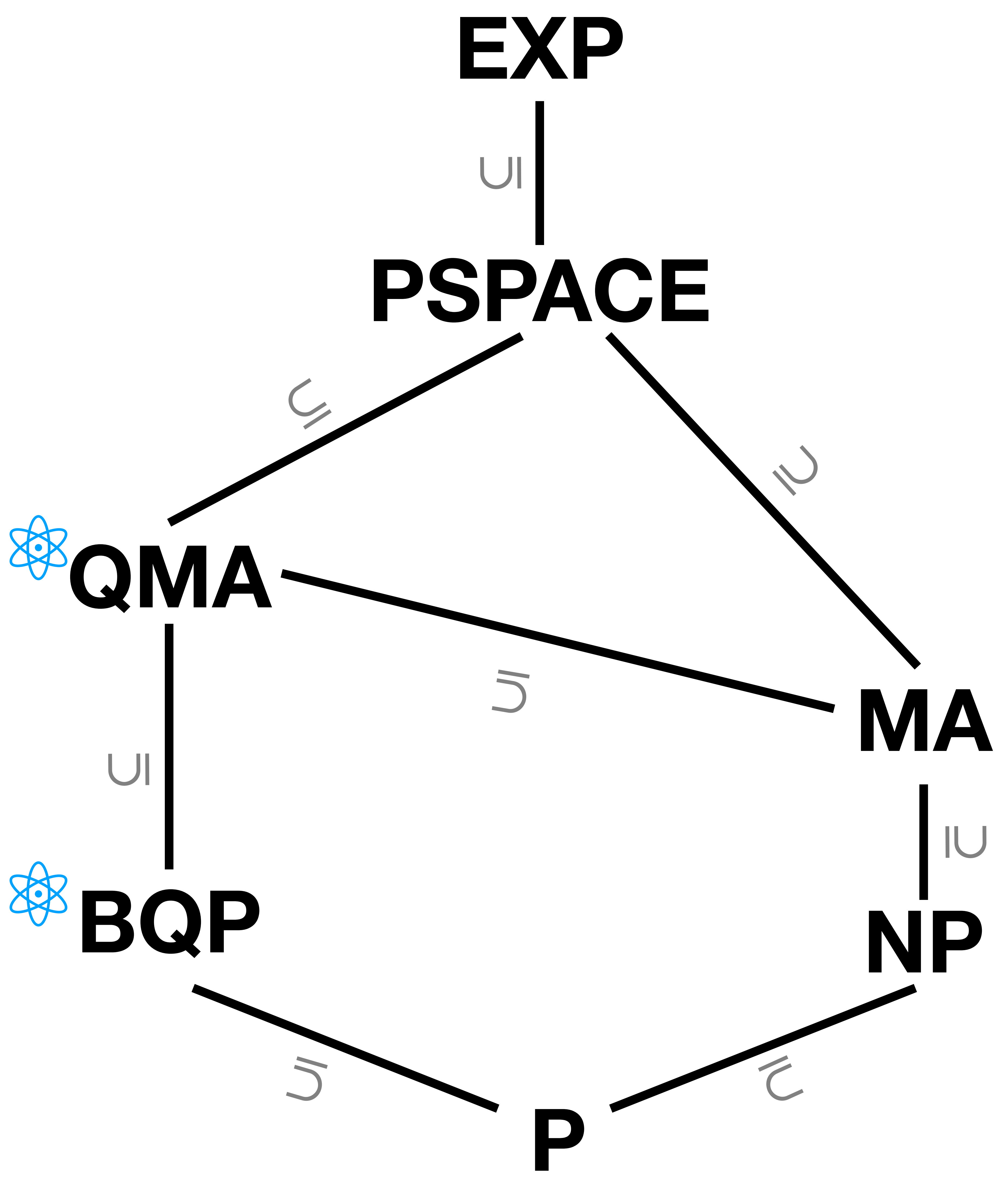
"Solution can be efficiently verified (classically)"

"Is proving a theorem harder than verifying it?"

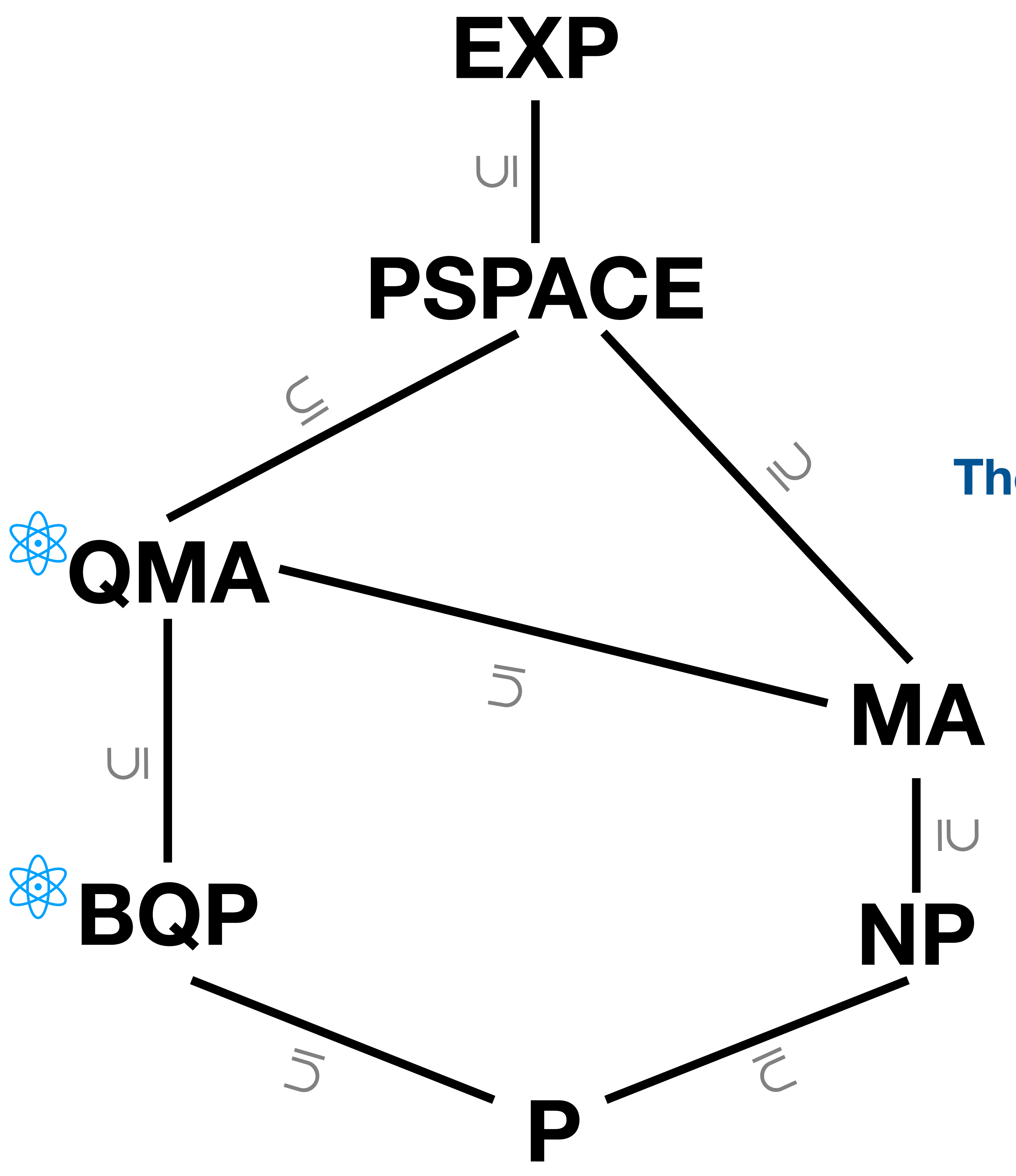
Classical and Quantum Complexity Classes



Classical and Quantum Complexity Classes

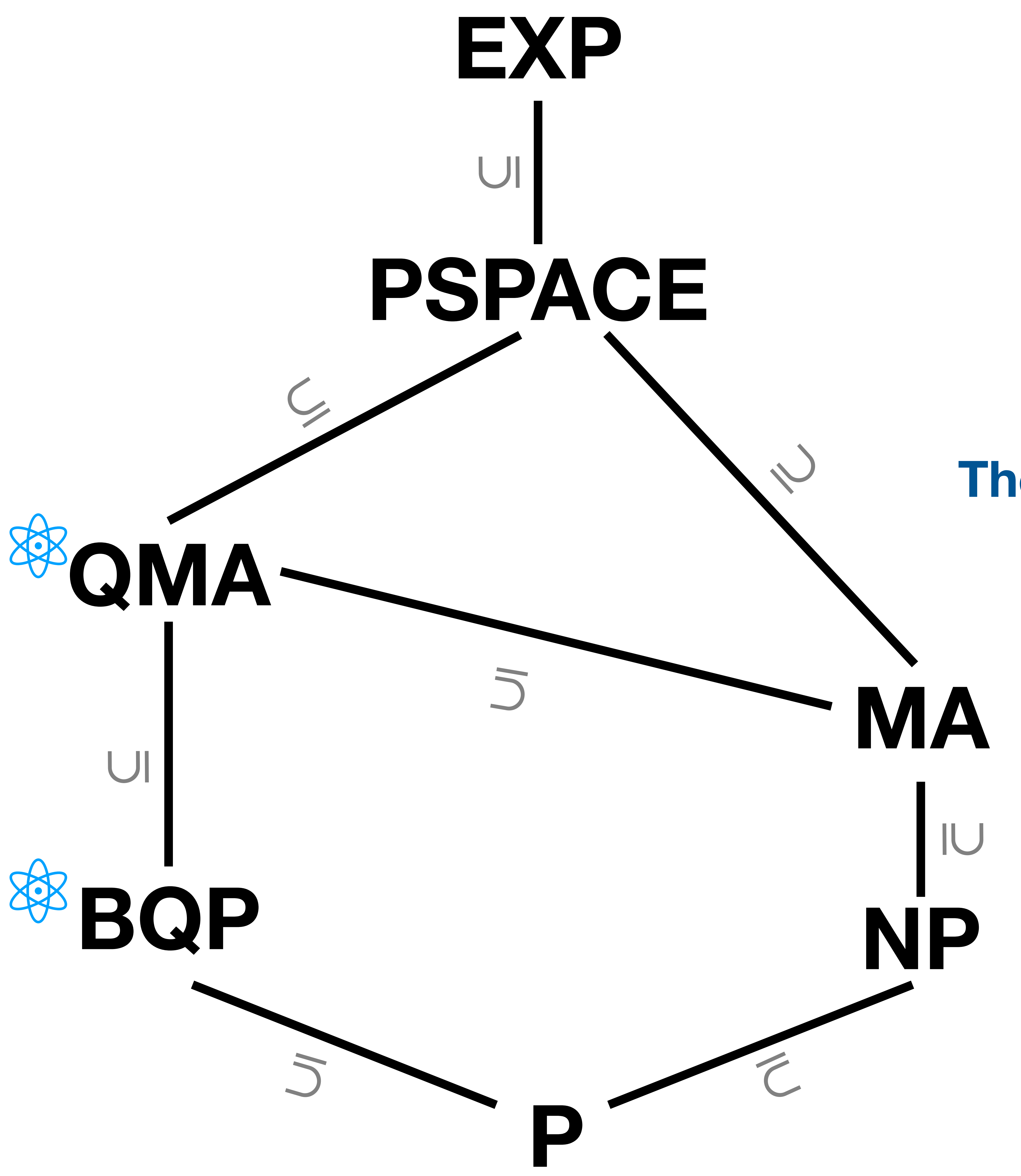


Classical and Quantum Complexity Classes



These inclusions were rigorously proved!

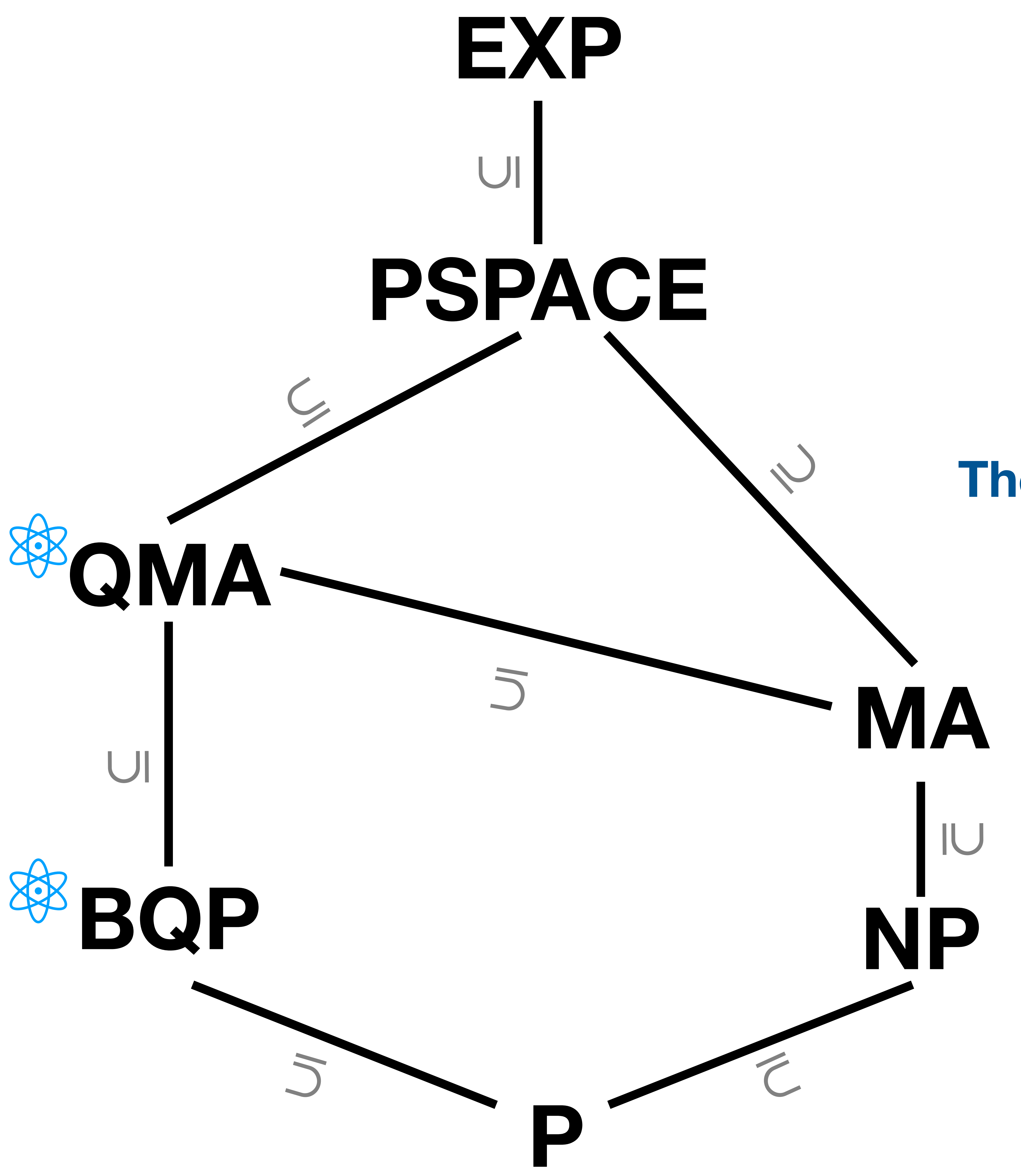
Classical and Quantum Complexity Classes



These inclusions were rigorously proved!

BQP and NP are conjectured to be incomparable!

Classical and Quantum Complexity Classes

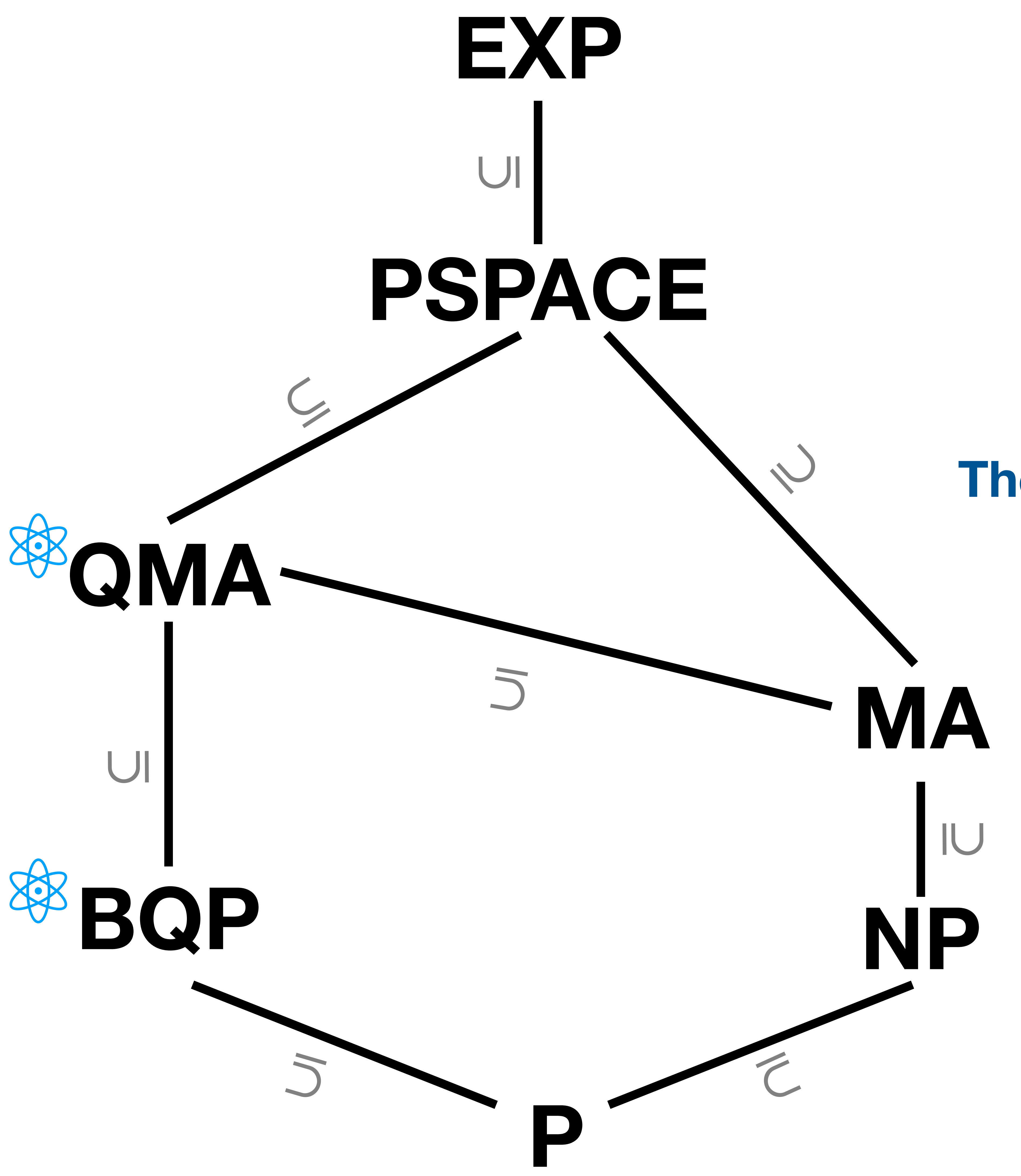


These inclusions were rigorously proved!

BQP and NP are conjectured to be incomparable!

NP $\not\subseteq$ BQP

Classical and Quantum Complexity Classes



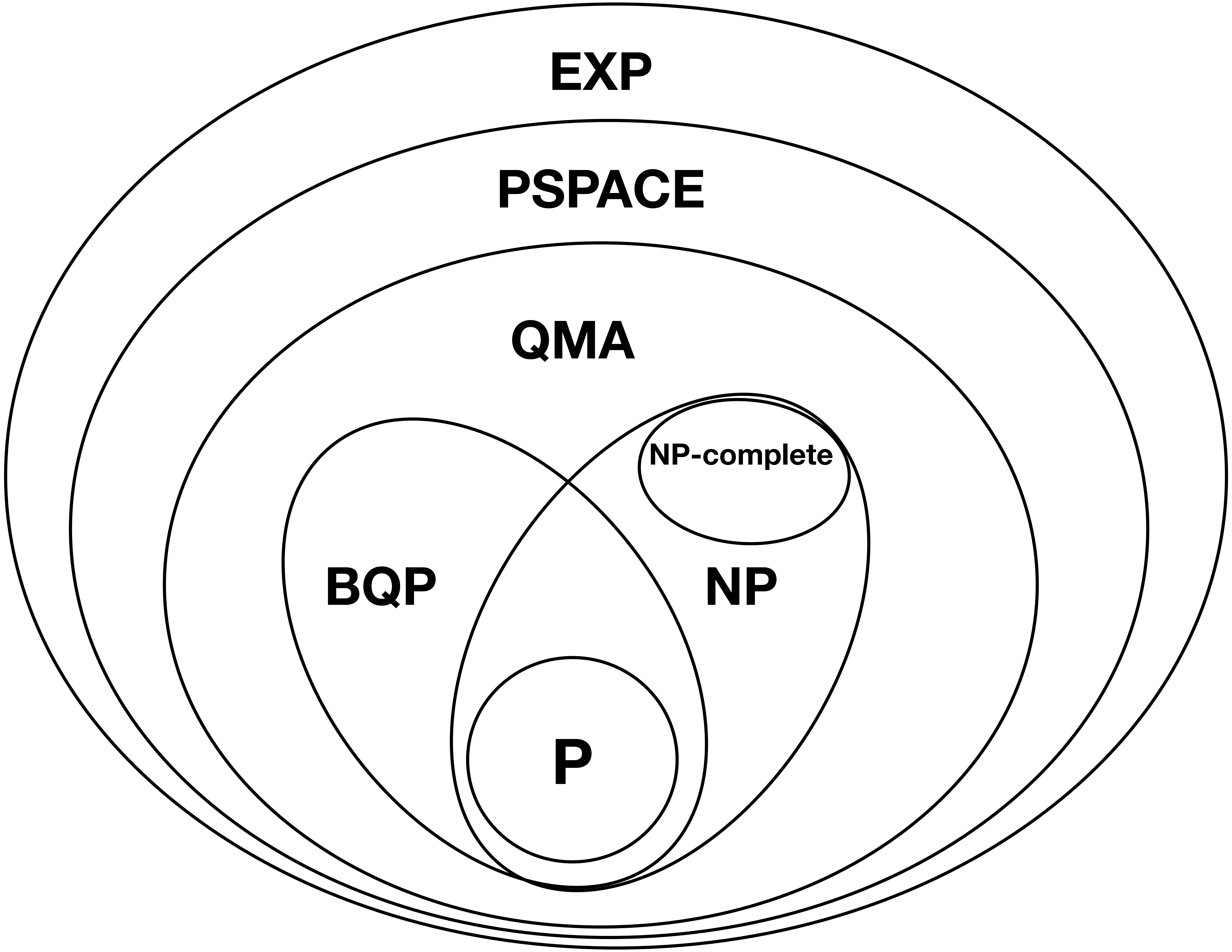
These inclusions were rigorously proved!

BQP and NP are conjectured to be incomparable!

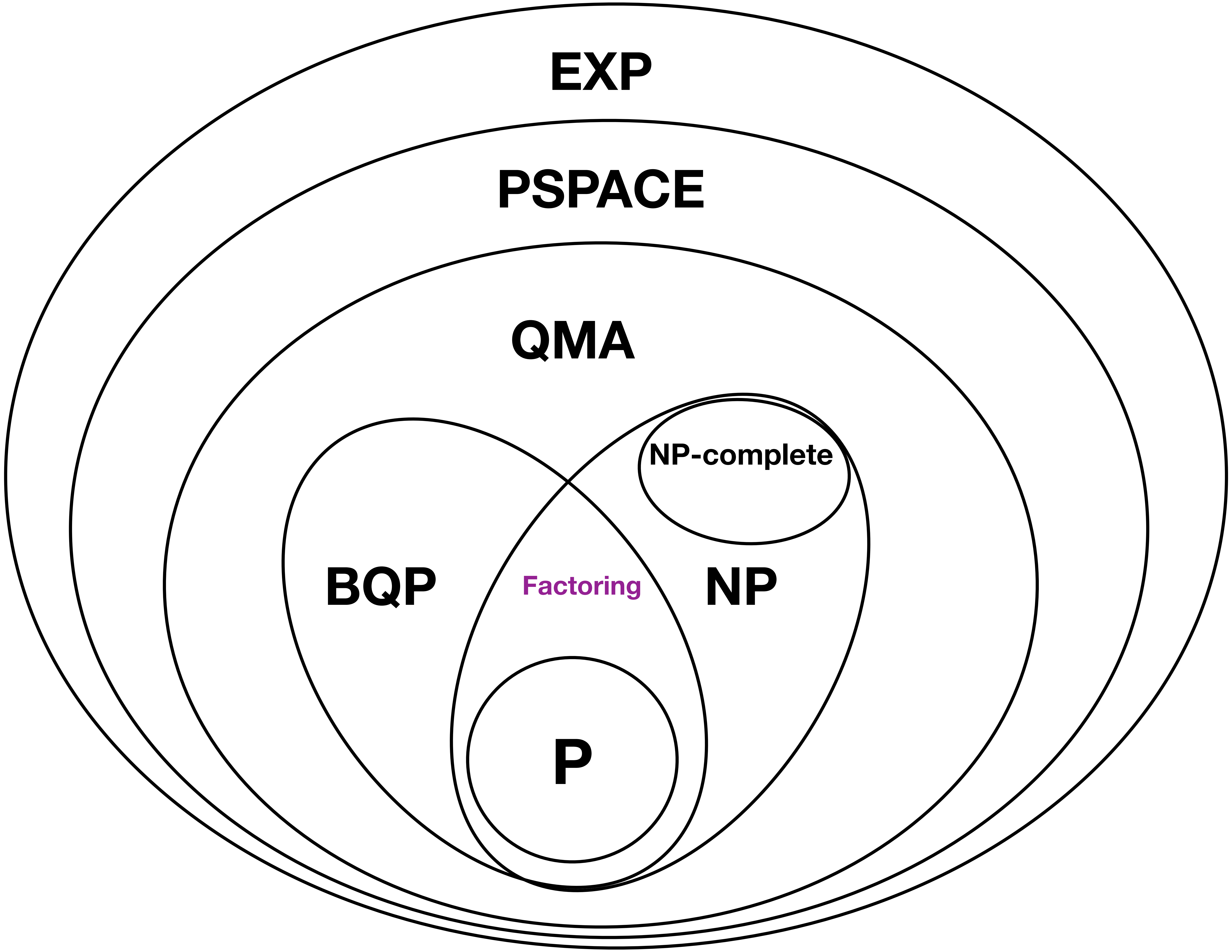
NP $\not\subseteq$ BQP

BQP $\not\subseteq$ NP

Conjectured Complexity Landscape



<u>Conjectured</u> Complexity Landscape



Can we give a complexity upper bound on BQP?

Warmup

EXP

$\subseteq$

BQP

Warmup

EXP

BQP

<u>Implication</u>: efficient quantum computers will never provide more than some exponential speed-ups (if any) over classical ones

A Better Upper Bound on BQP

PSPACE

$\rotatebox{90}{$\subseteq$}$

BQP

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ **(where L is language in BQP)**

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0\ldots0}_{\ell}\rangle$$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$U_m U_{m-1} \cdots U_2 U_1 | \underbrace{0 \ldots 0}_{\ell} \rangle$$

BQP verifier

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0\ldots0}_{\ell}\rangle$$

BQP verifier

$m = \textbf{poly}(n), \ell = \textbf{poly}(n)$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0\ldots0}_{\ell}\rangle$$

BQP verifier

$m =$ **poly**$(n), \ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \stackrel{?}{\in} L$ (where L is language in BQP)

$$\sum_{y \in \{0,1\}^\ell} \alpha_y |y\rangle = U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0 \ldots 0}_{\ell}\rangle$$

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$\sum_{y \in \{0,1\}^\ell} \alpha_y |y\rangle = U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0\ldots0}_{\ell}\rangle$$

BQP verifier

$m =$ **poly**$(n), \ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

BQP verifier measures the first qubit and accept iff outcome is $|1\rangle$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$ (where L is language in BQP)

$$\sum_{y \in \{0,1\}^\ell} \alpha_y |y\rangle = U_m U_{m-1} \cdots U_2 U_1 |\underbrace{0\ldots0}_{\ell}\rangle$$

BQP verifier

$m =$ **poly**$(n), \ell =$ **poly**(n)

U_i **is 2-local,** $\forall i \in [m]$

Can we compute a single amplitude $\alpha_{y'}$ in PSPACE?

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m U_{m-1} \cdots U_2 U_1 | 0 \ldots 0 \rangle$$

BQP verifier

$m =$ **poly**$(n), \ell =$ **poly**(n)

U_i **is 2-local,** $\forall i \in [m]$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m U_{m-1} \cdots U_2 U_1 | 0 \ldots 0 \rangle$$

BQP verifier

$m =$ **poly**$(n), \ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m I U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m I U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

$$= \sum_{y_m \in \{0,1\}^\ell} \langle y' | U_m | y_m \rangle \langle y_m | U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y'|U_m I U_{m-1} I \cdots I U_2 I U_1|0\ldots 0\rangle$$

$$= \sum_{y_m \in \{0,1\}^\ell} \langle y'|U_m|y_m\rangle\langle y_m|U_{m-1} I \cdots I U_2 I U_1|0\ldots 0\rangle$$

$$= \sum_{y_m,\ldots,y_2 \in \{0,1\}^\ell} \langle y'|U_m|y_m\rangle\langle y_m|U_{m-1}|y_{m-1}\rangle\langle y_{m-1}|\cdots|y_3\rangle\langle y_3|U_2|y_2\rangle\langle y_2|U_1|0\ldots 0\rangle$$

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y'|U_m I U_{m-1} I \cdots I U_2 I U_1 |0\ldots 0\rangle$$

$$= \sum_{y_m \in \{0,1\}^\ell} \langle y'|U_m|y_m\rangle\langle y_m|U_{m-1} I \cdots I U_2 I U_1|0\ldots 0\rangle$$

$$= \sum_{y_m,\ldots,y_2 \in \{0,1\}^\ell} \langle y'|U_m|y_m\rangle\langle y_m|U_{m-1}|y_{m-1}\rangle\langle y_{m-1}|\cdots|y_3\rangle\langle y_3|U_2|y_2\rangle\langle y_2|U_1|0\ldots 0\rangle$$

...

Each factor is efficient to compute

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m I U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

$$= \sum_{y_m \in \{0,1\}^\ell} \langle y' | U_m | y_m \rangle \langle y_m | U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

$$= \sum_{y_m, \ldots, y_2 \in \{0,1\}^\ell} \langle y' | U_m | y_m \rangle \langle y_m | U_{m-1} | y_{m-1} \rangle \langle y_{m-1} | \cdots | y_3 \rangle \langle y_3 | U_2 | y_2 \rangle \langle y_2 | U_1 | 0 \ldots 0 \rangle$$

...

Each factor is efficient to compute, and there are only $m =$ poly(n)

BQP verifier

$m =$ **poly**(n), $\ell =$ **poly**(n)

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\alpha_{y'} = \langle y' | U_m I U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

$$= \sum_{y_m \in \{0,1\}^\ell} \langle y' | U_m | y_m \rangle \langle y_m | U_{m-1} I \cdots I U_2 I U_1 | 0 \ldots 0 \rangle$$

$$= \sum_{y_m, \ldots, y_2 \in \{0,1\}^\ell} \langle y' | U_m | y_m \rangle \langle y_m | U_{m-1} | y_{m-1} \rangle \langle y_{m-1} | \cdots | y_3 \rangle \langle y_3 | U_2 | y_2 \rangle \langle y_2 | U_1 | 0 \ldots 0 \rangle$$

...

Each factor is efficient to compute, and there are only $m = \textbf{poly}(n)$

$\Longrightarrow$ summation can be computed in PSPACE

BQP verifier

$m = \textbf{poly}(n), \ell = \textbf{poly}(n)$

U_i **is 2-local**, $\forall i \in [m]$

Fact: $I = \sum_{y \in \{0,1\}^\ell} |y\rangle\langle y|$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\Pr[\textbf{BQP verifier accepts } x] = \sum_{y \in \{0,1\}^\ell :\ y_1 = 1} |\alpha_y|^2$$

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\Pr[\textbf{BQP verifier accepts } x] = \sum_{y \in \{0,1\}^\ell :\ y_1 = 1} |\alpha_y|^2$$

$\Longrightarrow$ **this probability can be computed in PSPACE**

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\Pr[\textbf{BQP verifier accepts } x] = \sum_{y \in \{0,1\}^\ell :\, y_1 = 1} |\alpha_y|^2$$

$\Longrightarrow$ **this probability can be computed in PSPACE**

$\Longrightarrow$ **BQP $\subseteq$ PSPACE**

Proving the Upper Bound

Given $x \in \{0,1\}^n, \quad x \overset{?}{\in} L$

$$\Pr[\textbf{BQP verifier accepts } x] = \sum_{y \in \{0,1\}^{\ell}:\ y_1 = 1} |\alpha_y|^2$$

$\implies$ **this probability can be computed in PSPACE**

$\implies$ **BQP $\subseteq$ PSPACE**

$\blacksquare$

Thank you!

Thank you!

More Questions?