Structure in Quantum Algorithms:
Quantum Adversary (Upper) Bound

Shelby Kimmel

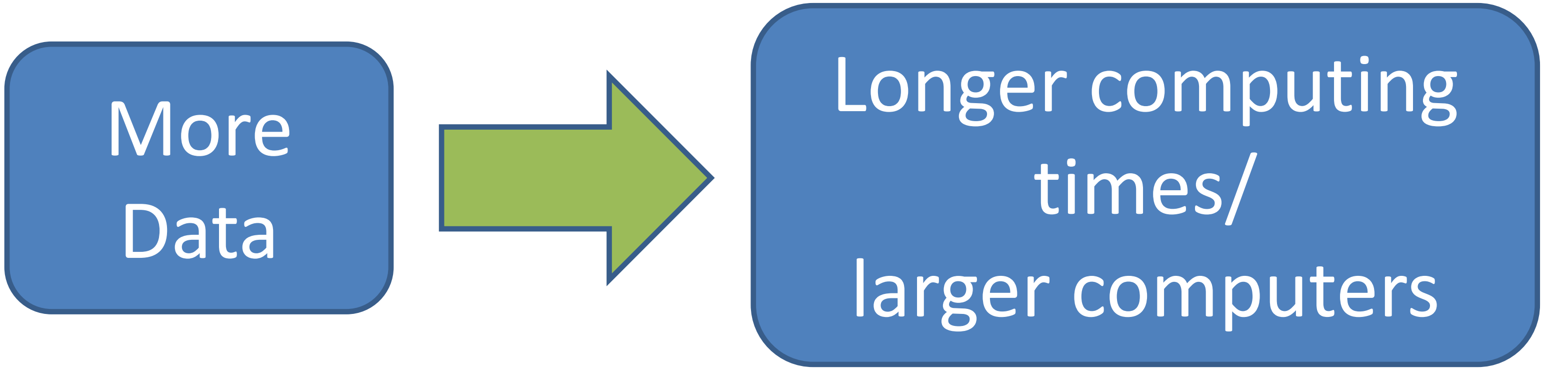
Center for Theoretical Physics,

Massachusetts Institute of Technology

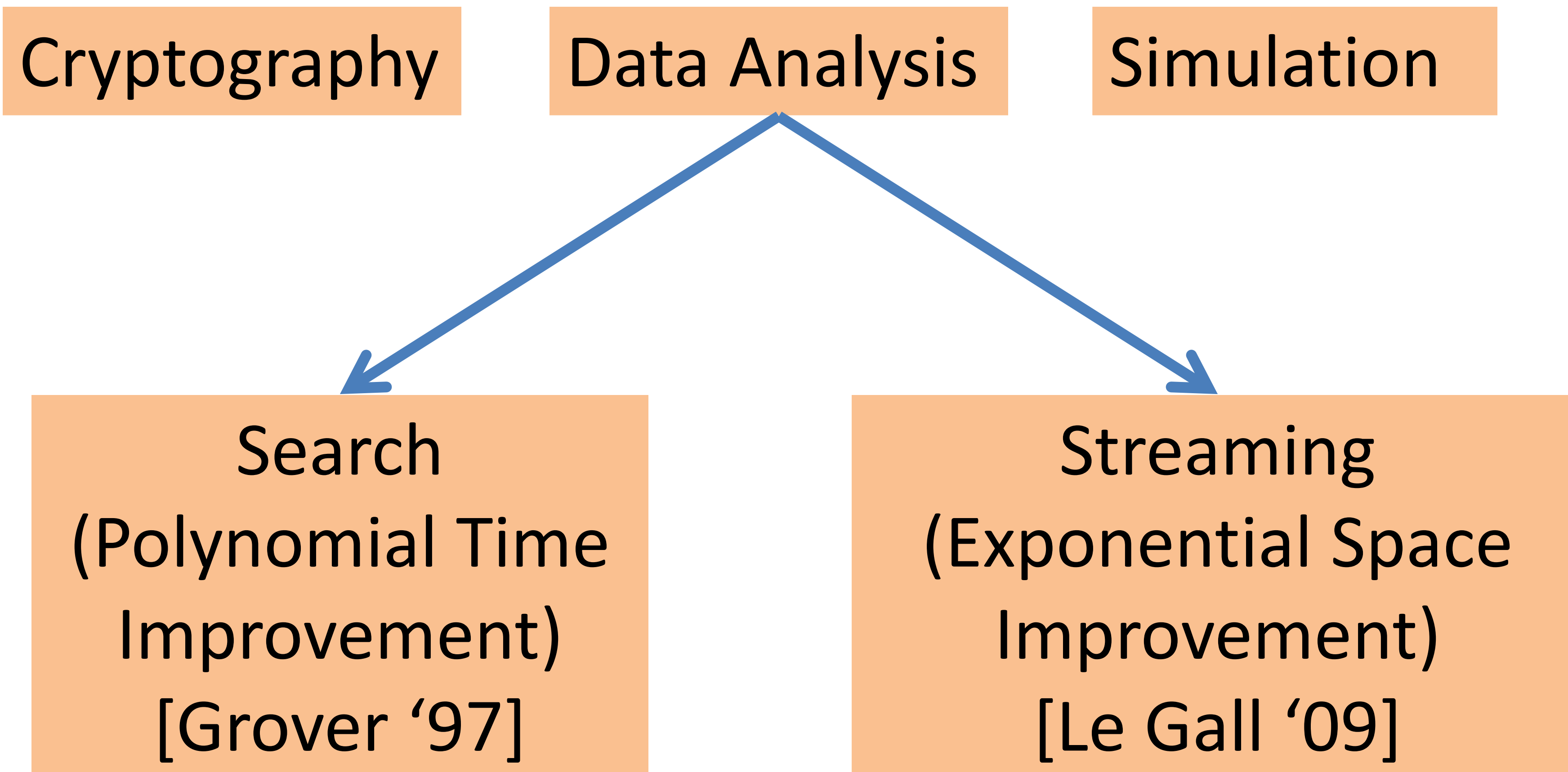
Santa Fe Institute, Jan. 27, 2014

Importance of Computation

- Across disciplines –
  - BIG DATA.
  - MACHINE LEARNING.
  - COMPLEX NETWORKS.



Quantum Computers Can Help!



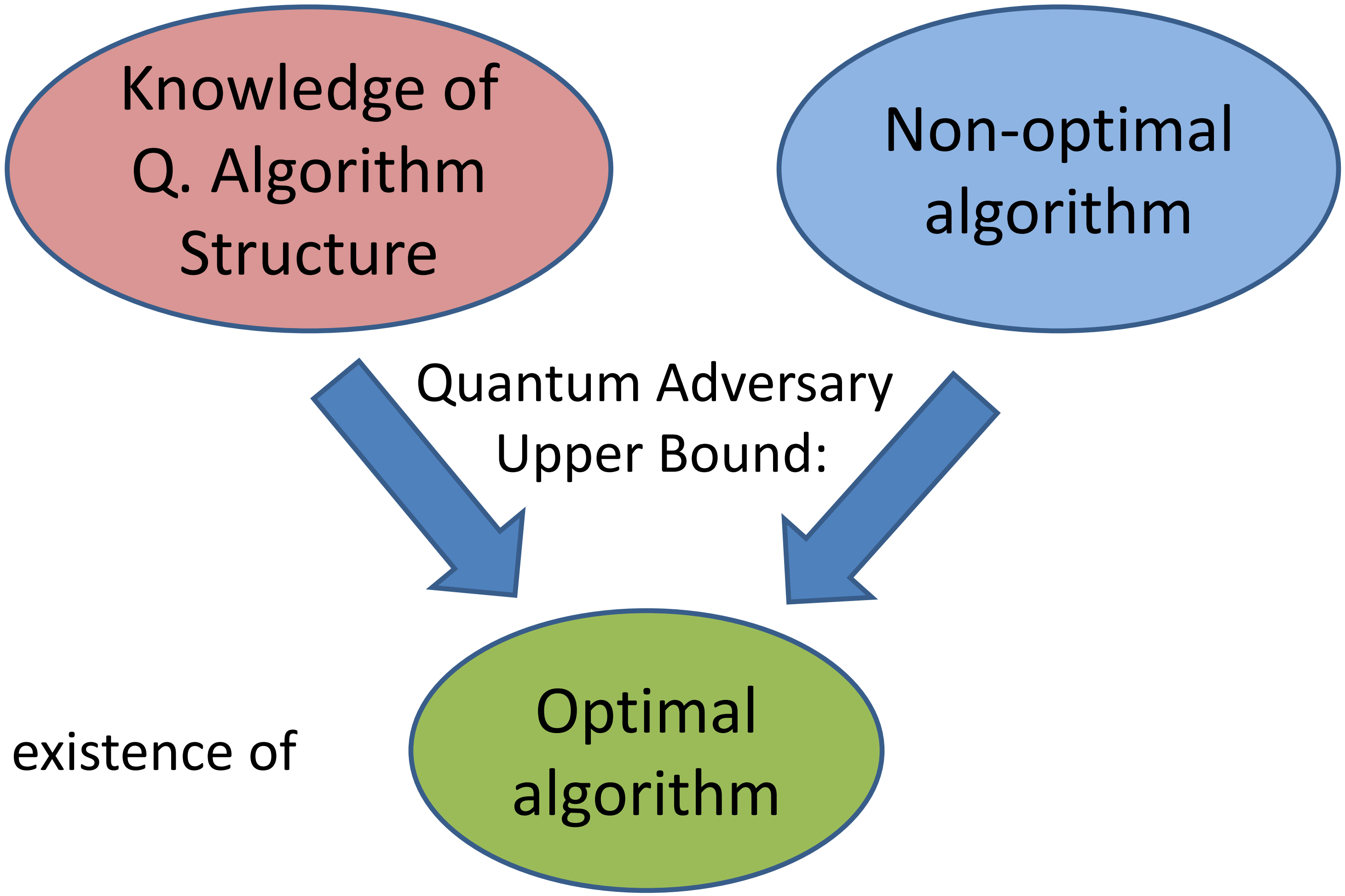
Quantum Computers Can Help!

Design new quantum algorithms

Practical

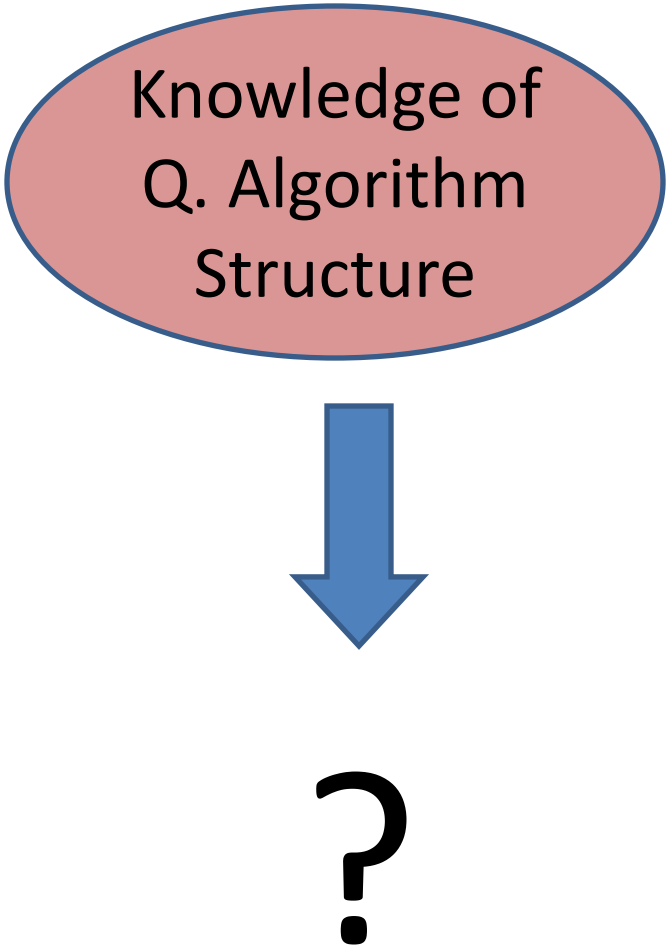
Fundamental

Result



Prove existence of

Larger Goals



Outline

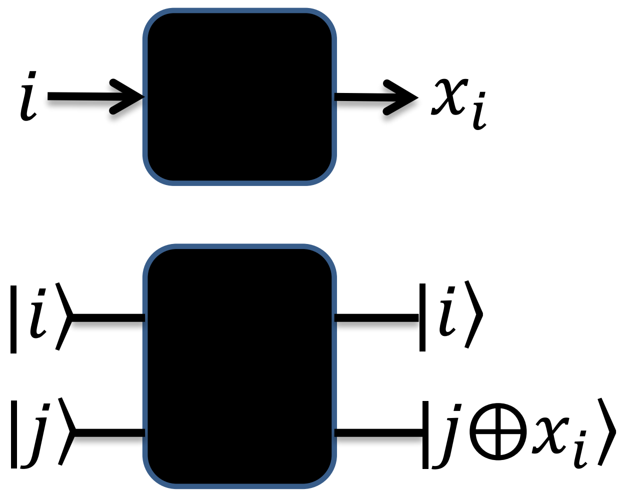
- Oracle Model and Query Complexity
- Quantum Adversary (Upper) Bound
- Application
  - Prove existence of optimal algorithm using Quantum Adversary (Upper) Bound
- Future Work: Adversary Bound and New Models of Computation

Oracle Model

Goal: Determine the value of $f(x_1, \ldots, x_n)$ with inputs $x_i = \{0,1\}$ for a known function f, given an oracle for x

Classical Oracle

Quantum Oracle



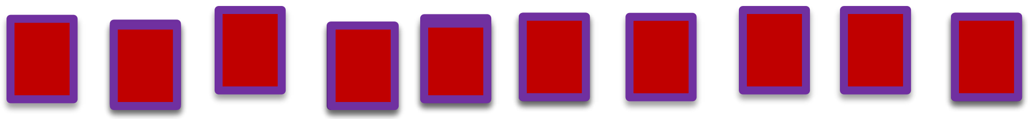
Care about $Q(f) =$ "quantum query complexity"

$=$ # of quantum oracle uses (queries)

Example of Query Complexity

x	$\longrightarrow$	f
$\{0,0,\cdots,0,0\}$	$\longrightarrow$	0
$\{1,1,\cdots,1,1\}$	$\longrightarrow$	0
50% 0, 50% 1	$\longrightarrow$	1

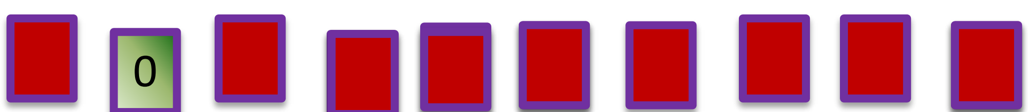
x



Example of Query Complexity

x	$\longrightarrow$	f
$\{0,0,\cdots,0,0\}$	$\longrightarrow$	0
$\{1,1,\cdots,1,1\}$	$\longrightarrow$	0
50% 0, 50% 1	$\longrightarrow$	1

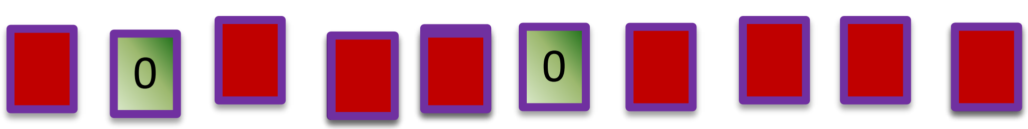
x



Example of Query Complexity

x		f
$\{0,0,\cdots,0,0\}$	$\longrightarrow$	0
$\{1,1,\cdots,1,1\}$	$\longrightarrow$	0
50% 0, 50% 1	$\longrightarrow$	1

x



Example of Query Complexity

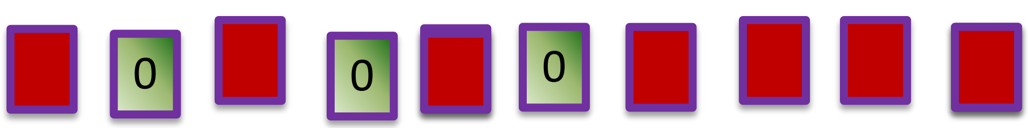
$$x \longrightarrow f$$

$$\{0,0,\cdots,0,0\} \longrightarrow 0$$

$$\{1,1,\cdots,1,1\} \longrightarrow 0$$

$$50\%\ 0, \qquad 50\%\ 1 \longrightarrow 1$$

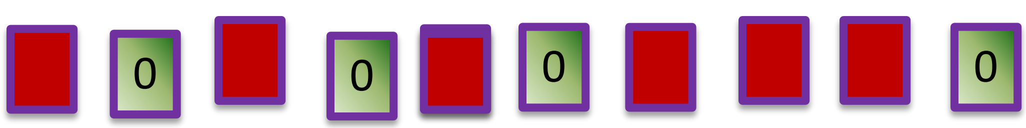
x



Example of Query Complexity

x		f
$\{0,0,\cdots,0,0\}$	$\longrightarrow$	0
$\{1,1,\cdots,1,1\}$	$\longrightarrow$	0
50% 0, 50% 1	$\longrightarrow$	1

x



Example of Query Complexity

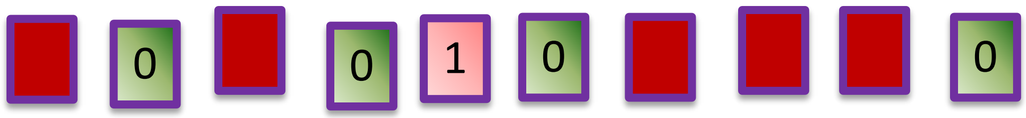
$$x \longrightarrow f$$

$$\{0,0,\cdots,0,0\} \longrightarrow 0$$

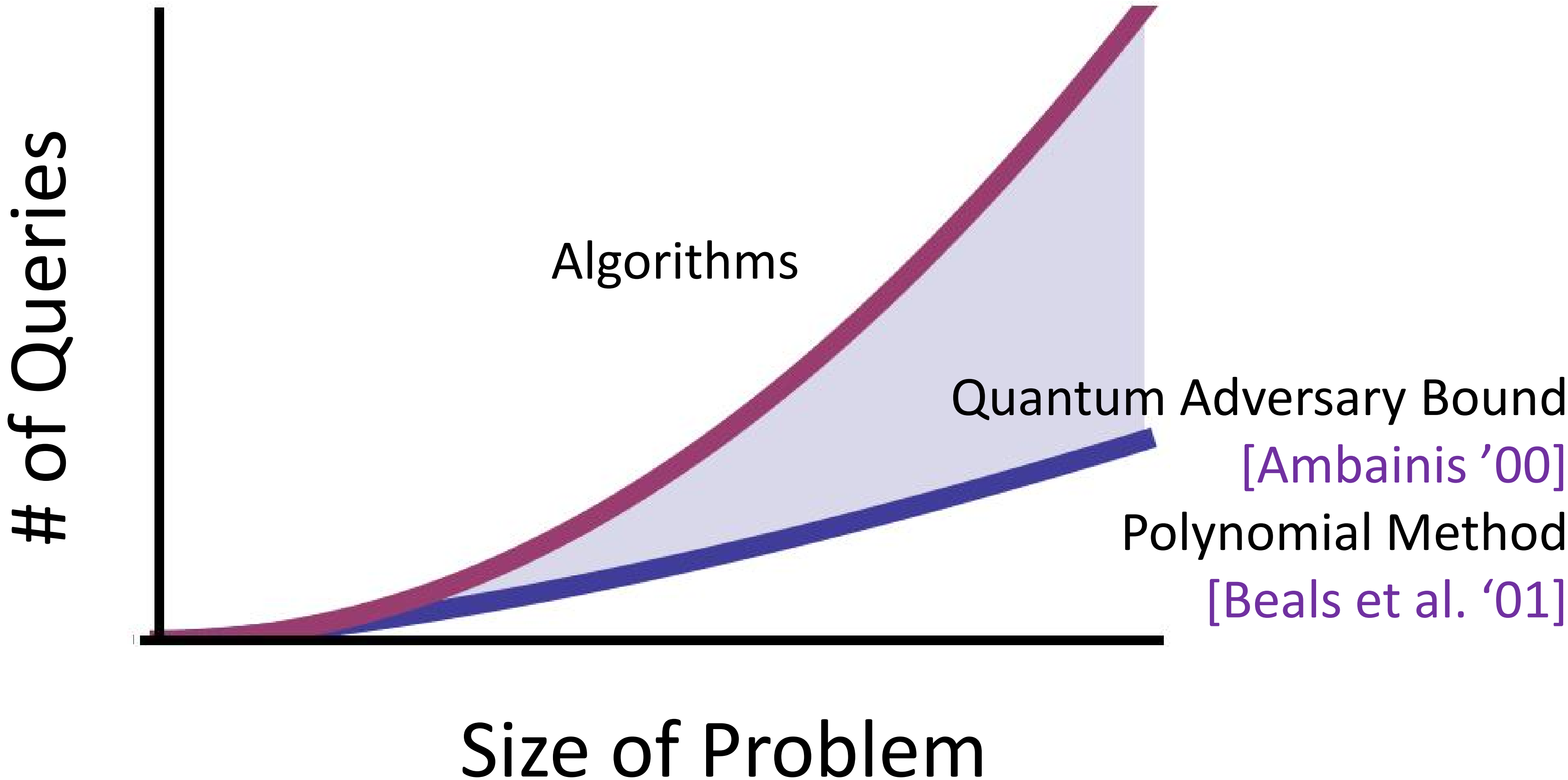
$$\{1,1,\cdots,1,1\} \longrightarrow 0$$

$$50\%\ 0, \quad 50\%\ 1 \longrightarrow 1$$

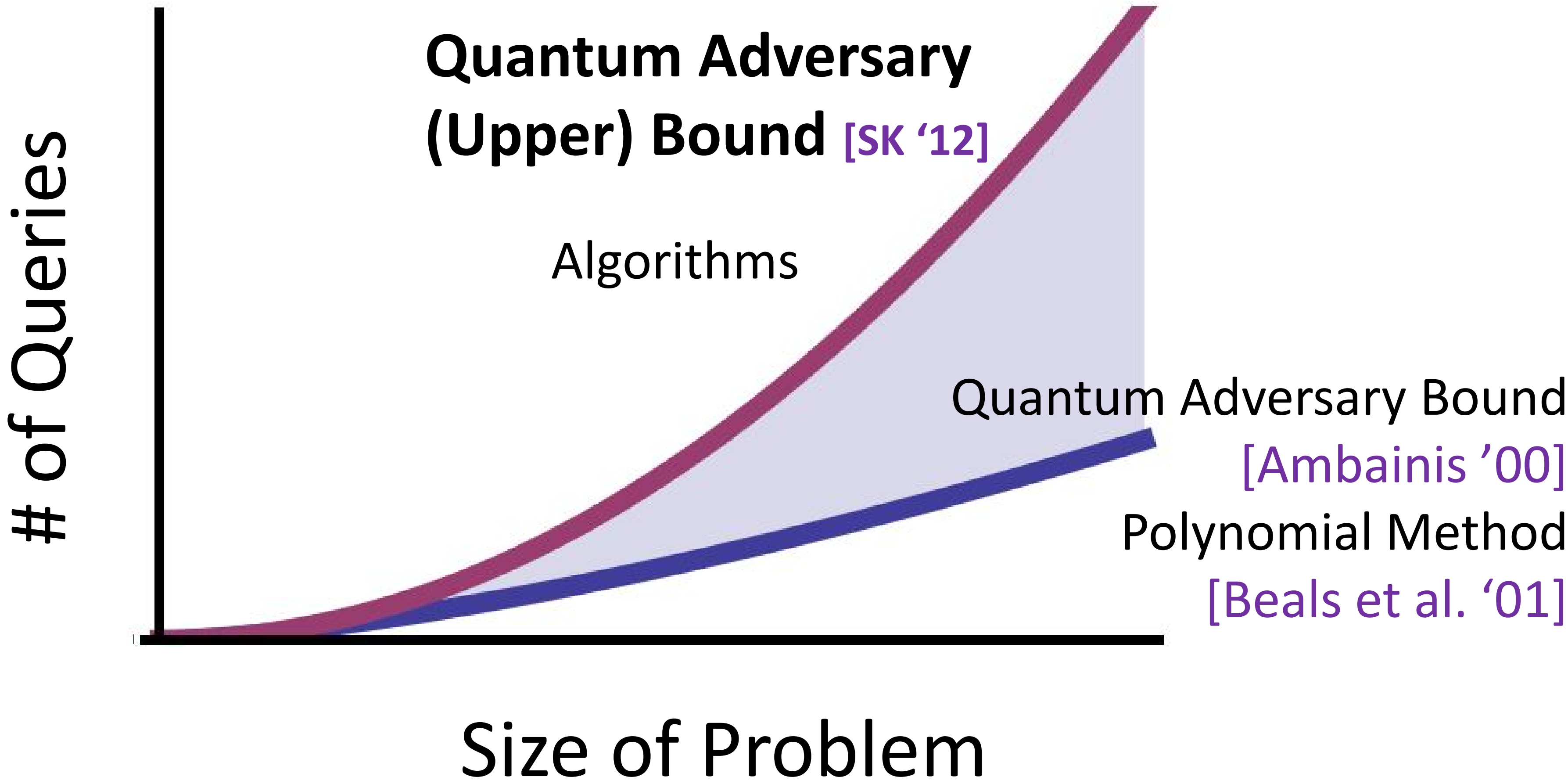
x



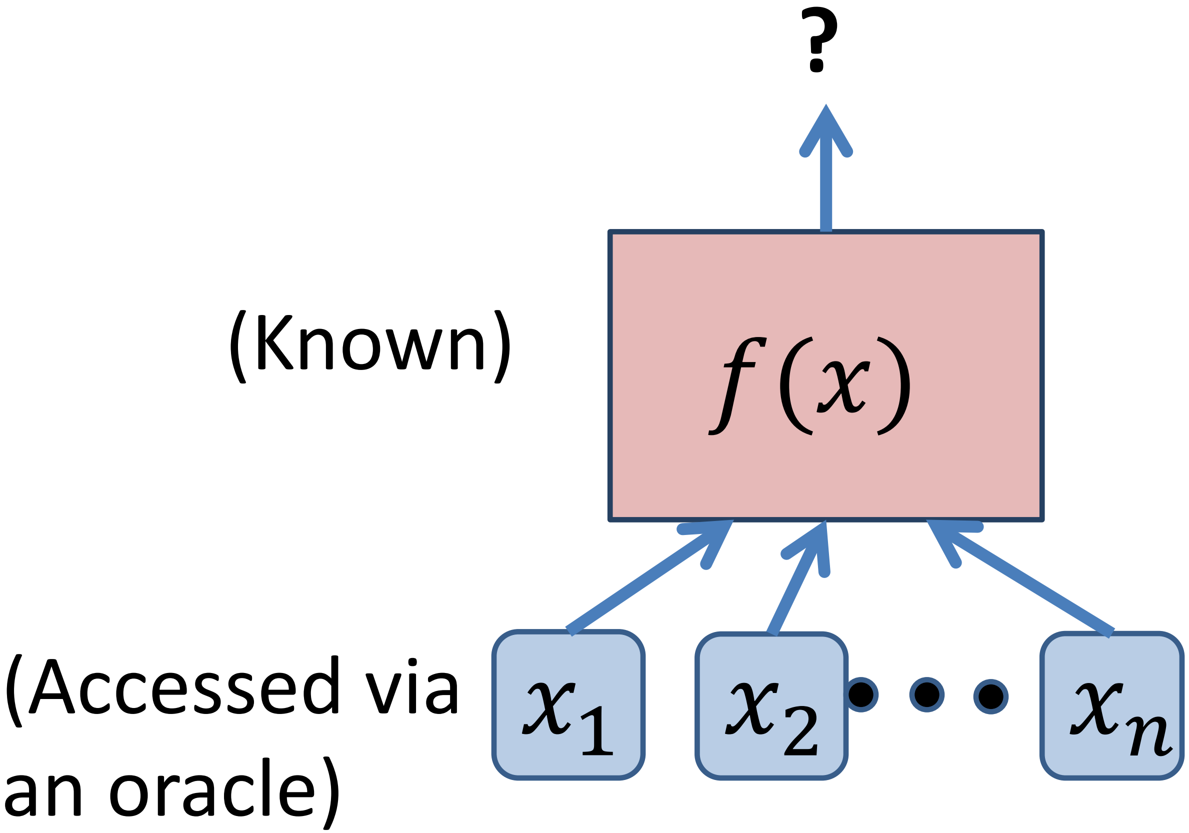
Quantum Query Complexity



Quantum Query Complexity

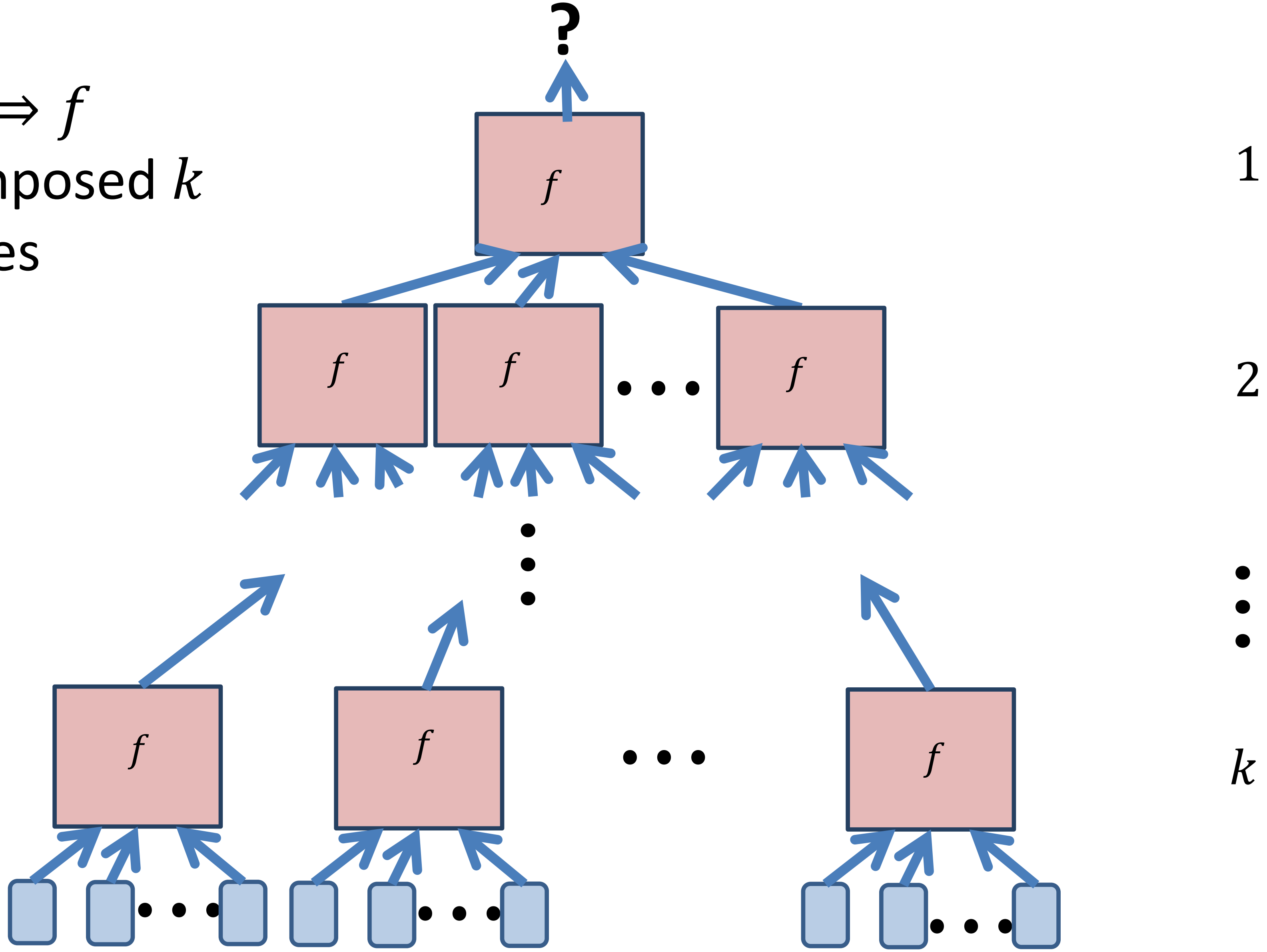


Composed Functions



Composed Functions

$f^k \Rightarrow f$
composed k
times



Quantum Adversary Upper Bound

[SK '12]

Let f be a Boolean function.

Create an algorithm for f^k, with T queries, so learn $Q(f^k)$ is upper bounded by T.

Then $Q(f)$ is upper bounded by $T^{1/k}$.

($\mathrm{Q}(f)$ = quantum query complexity of f)

Quantum Adversary Upper Bound

[SK ’12]

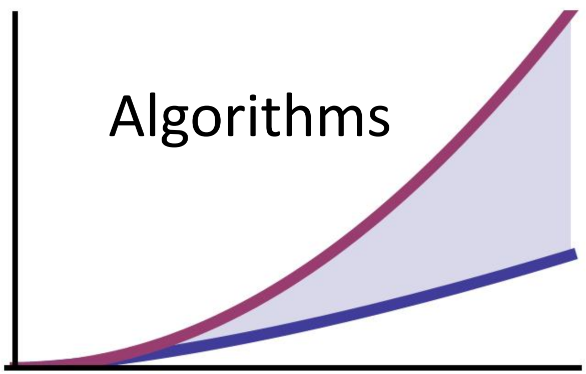
Let f be a Boolean function.

Create an algorithm for f^k, with T queries, so learn $Q\left(f^k\right)$ is upper bounded by T.

Then $Q(f)$ is upper bounded by $T^{1/k}$.

Surprising:

- Does not give algorithm for f



Quantum Adversary Upper Bound

[SK '12]

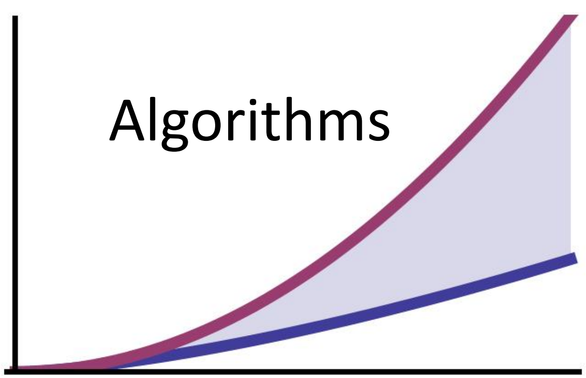
Let f be a Boolean function.

Create an algorithm for f^k, with T queries, so learn $Q(f^k)$ is upper bounded by T.

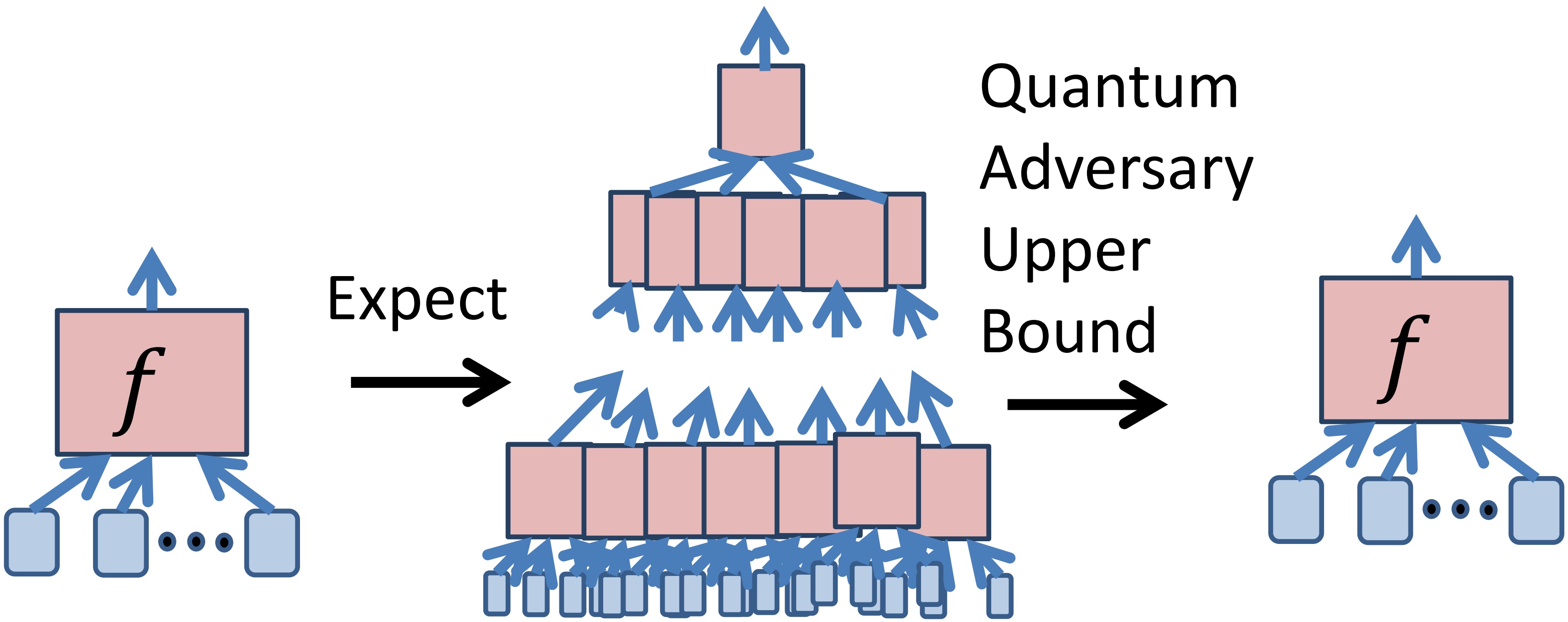
Then $Q(f)$ is upper bounded by $T^{1/k}$.

Surprising:

- Does not give algorithm for f
- This is a useful theorem!



Quantum Adversary Upper Bound

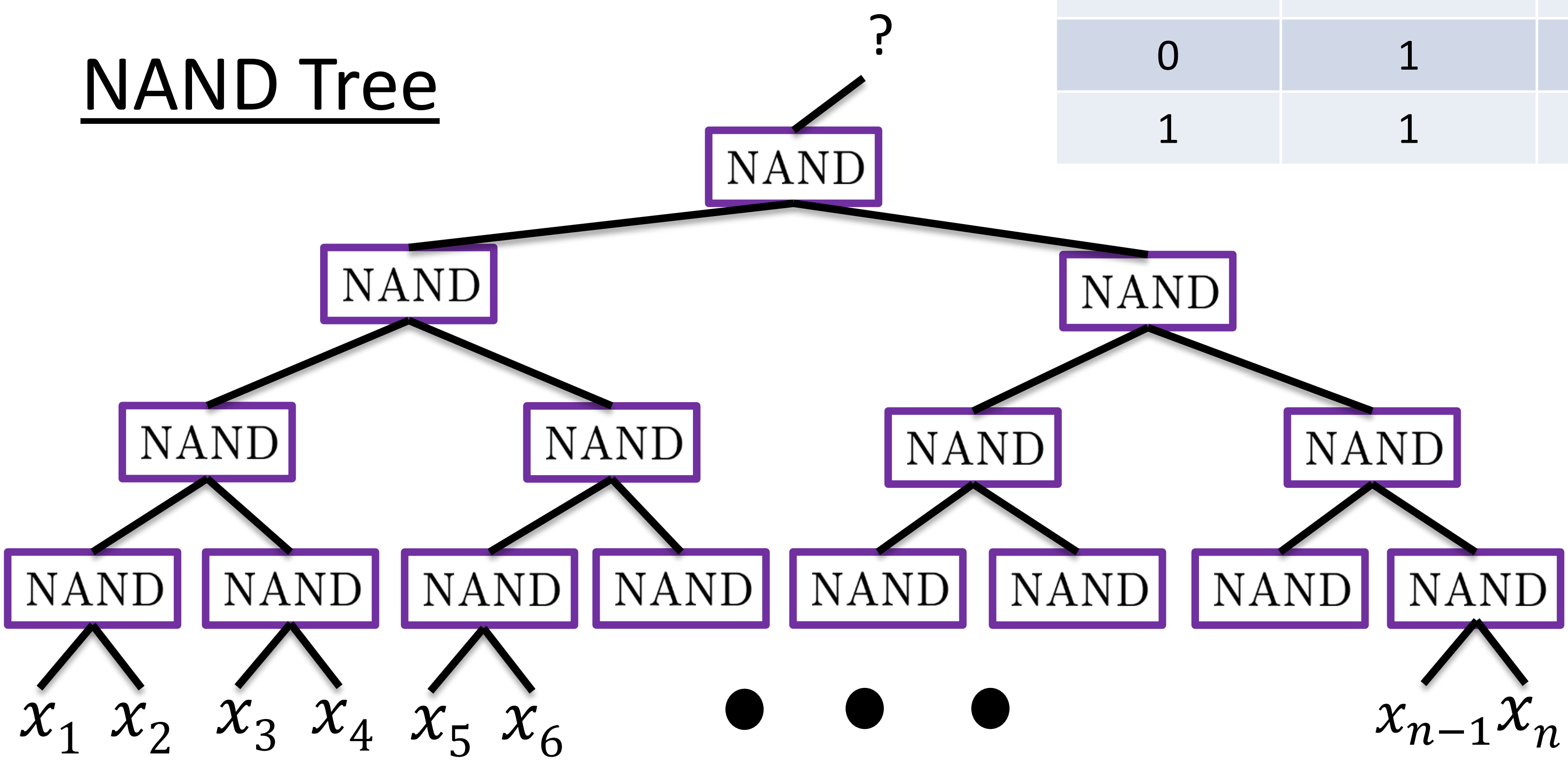


$Q(f) = O(T)$ $Q\left(f^k\right) = O(T^k)$ $Q(f) = O(T)$

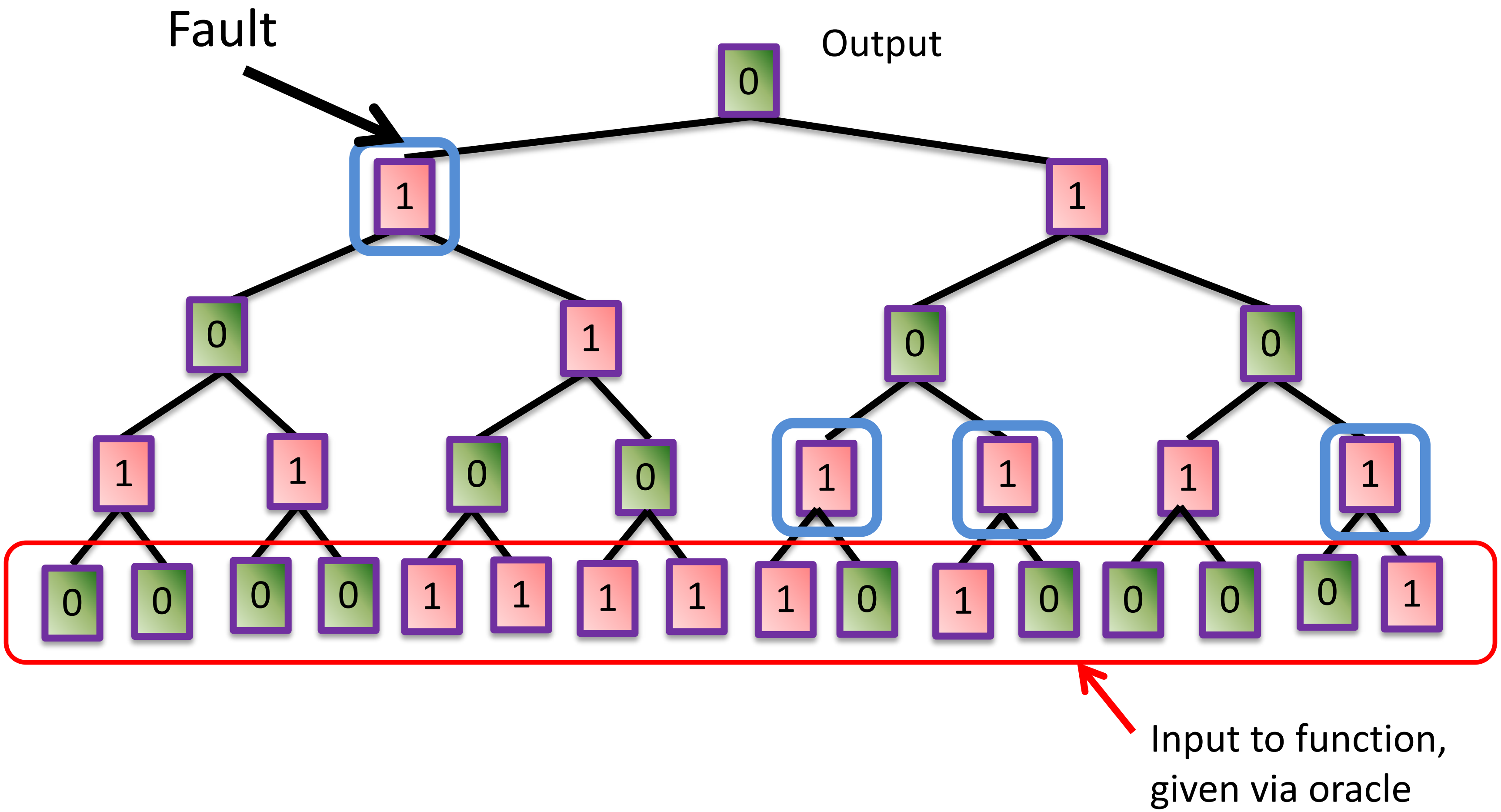
Example: 1-Fault NAND Tree

Input 1	Input 2	NAND
0	0	1
1	0	1
0	1	1
1	1	0

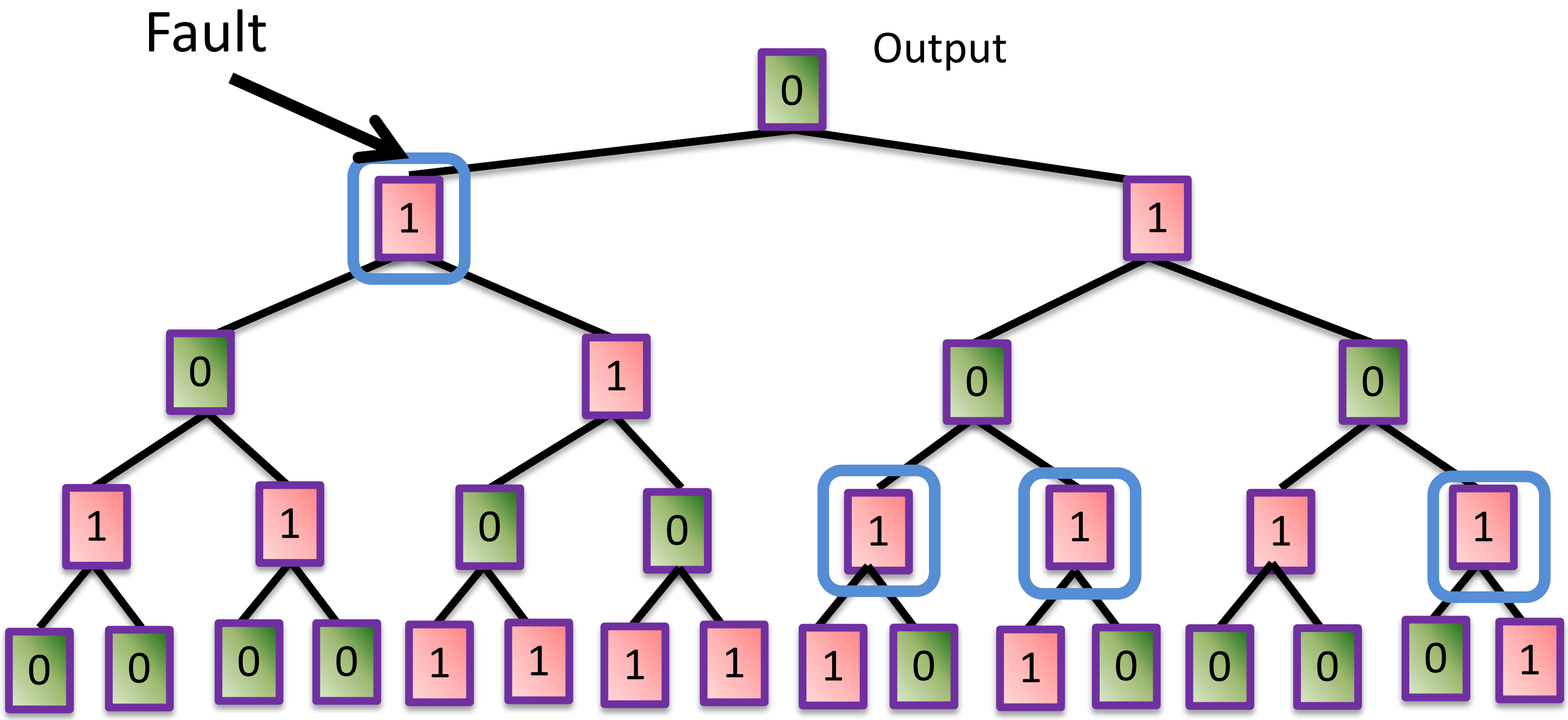
<u>NAND Tree</u>



Example: 1-Fault NAND Tree



Example: 1-Fault NAND Tree



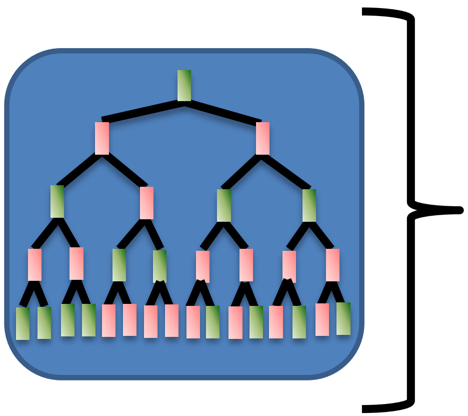
Another view point: 1-Fault NAND Tree is a game tree where the players are promised that they will only have to make one critical decision in the game.

Example: 1-Fault NAND Tree

[Zhan, Hassidim, SK `12]

We found algorithm for k-fault tree using $(2^k \times depth^2)$ queries

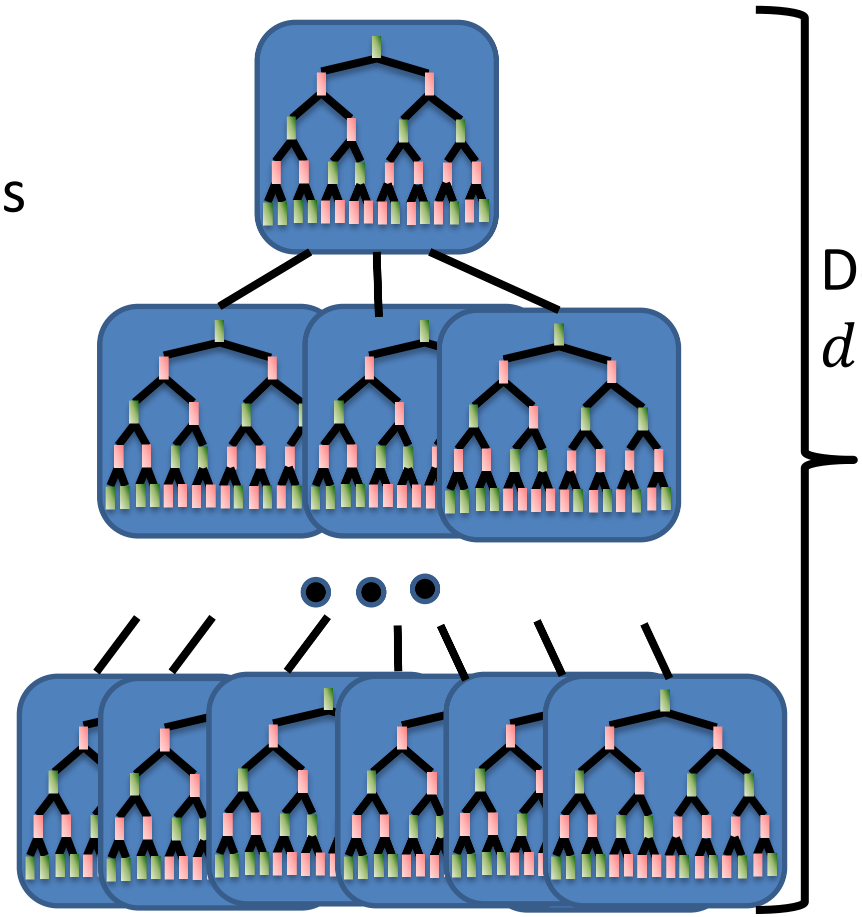
1-Fault NAND Tree



Depth d

$$Q(f) = O(d^2)$$

$[\text{1–Fault NAND Tree}]^{\log d}$



Depth $d \log d$

$$Q\left(f^{\log d}\right) = O(d^3)$$

Quantum Adversary Upper Bound

$1-\text{Fault NAND Tree}$ is a Boolean function

Quantum query complexity of
$[1-\text{Fault NAND Tree}]^{\log d}$ is $O(d^3)$

Then the quantum query complexity of
$[1-\text{Fault NAND Tree}]$ is
$O\left(d^{3/\log d}\right) = \mathrm{O}\left(2^{3\log d/\log d}\right) = O(1)$

Proving the Quantum Adversary Upper Bound: Powerful Tools at work

$ADV^{\pm}(f) = \theta(Q(f))$ [Reichardt, ‘09, ’11]

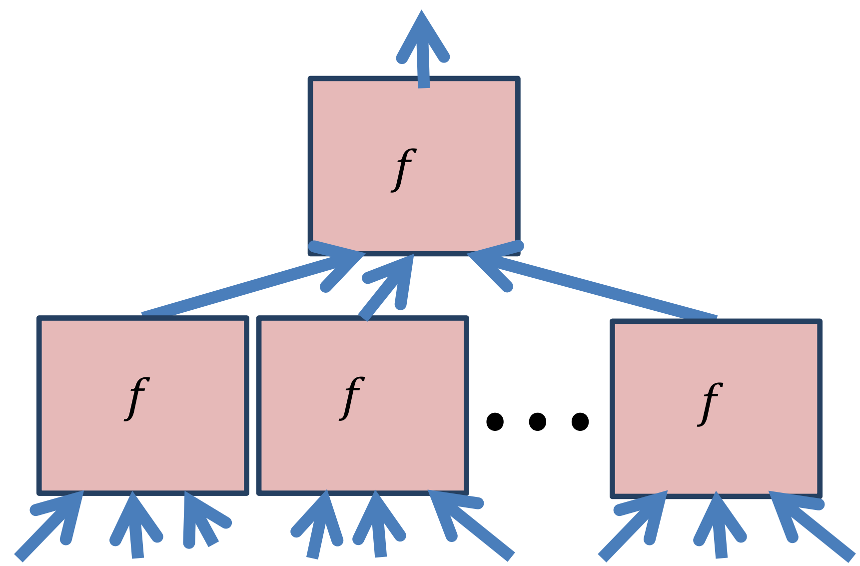
$ADV^{\pm}$ = General Adversary Bound

- Completely characterize query complexity.
- Semi-definite program (size scales exponentially with the # of inputs)
- Strong conditions on its behavior for composed functions.

Proving the Quantum Adversary Upper Bound: Powerful Tools at work

Lemma 2: $ADV^{\pm}(f^k) \geq ADV^{\pm}(f)^k$
[Hoyer, Lee, Spalek, '07, SK '11 (for partial functions)]

- Given a matrix that maximizes objective function of SDP of $ADV^{\pm}(f)$, construct a matrix satisfying the SDP for f^k
- When f is partial, set entries corresponding to non-valid inputs to 0. Need to check that things go through



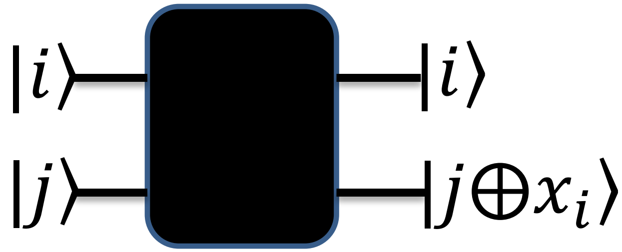
Long Story Short

- Quantum adversary upper bound can prove the existence of optimal quantum algorithms for
  – 1-Fault NAND Tree
  – Other constant fault trees
- I found explicit algorithms that match.

- Can we take advantage of the structure of quantum algorithms to prove other results?

My Goal: Apply $ADV^{\pm}$ to new models

Quantum
Oracle



Pros of Oracle Model

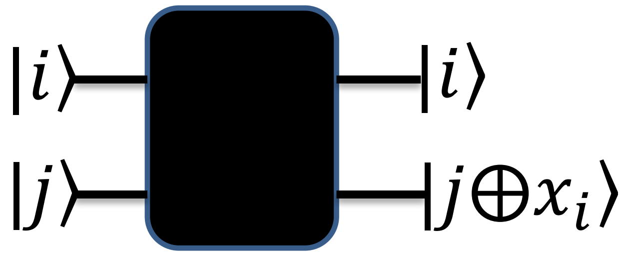
- Have powerful tools to bound $Q(f)$

Cons of Oracle Model

- Assumes you can implement oracle perfectly
- Black boxes usually not black
- Only takes into account oracle uses, not time or space necessary to solve problem

My Goal: Apply $ADV^{\pm}$ to new models

Quantum
Oracle



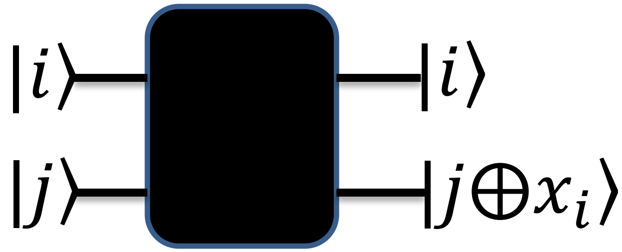
What if oracle has error?

- With probability p does nothing. [Regev, Schiff ‘08]

Conjecture: Require $p < Q(f)^{-1}$

My Goal: Apply $ADV^{\pm}$ to new models

Quantum Oracle



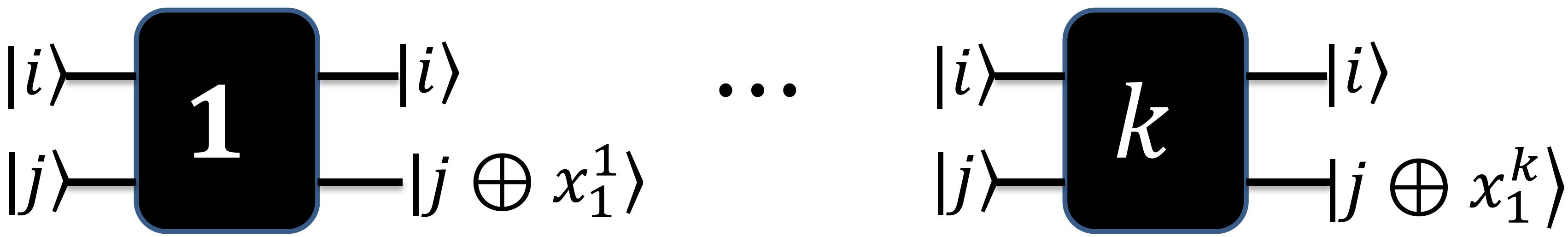
Pros of Oracle Model

- Have powerful tools to bound $Q(f)$

Cons of Oracle Model

- Assumes you can implement oracle perfectly
- Black boxes usually not black
- Only takes into account oracle uses, not time or space necessary to solve problem

My Goal: Apply $ADV^{\pm}$ to new models



Cost c_1 Cost c_k

More realistic model:

- Can use knowledge of x to create multiple oracles with different types of information
- Different operations take different times to implement

Long Story Short

- Quantum adversary upper bound can prove the existence of optimal quantum algorithms for
  – 1-Fault NAND Tree
  – Other constant fault trees
- I found explicit algorithms that match.

- Can we take advantage of the structure of quantum algorithms to prove other results?

Proving Quantum Adversary Upper Bound

Lemma 1: $ADV^{\pm}(f) = \theta(Q(f))$ [Reichardt, '09, '11]

Lemma 2: $ADV^{\pm}(f^k) \geq ADV^{\pm}(f)^k$
[Hoyer, Lee, Spalek, '07, SK '11 (for partial functions)]

Proof [SK '11]:

$$Q(f^k) = O(T)$$

$$ADV^{\pm}(f^k) = O(T)$$

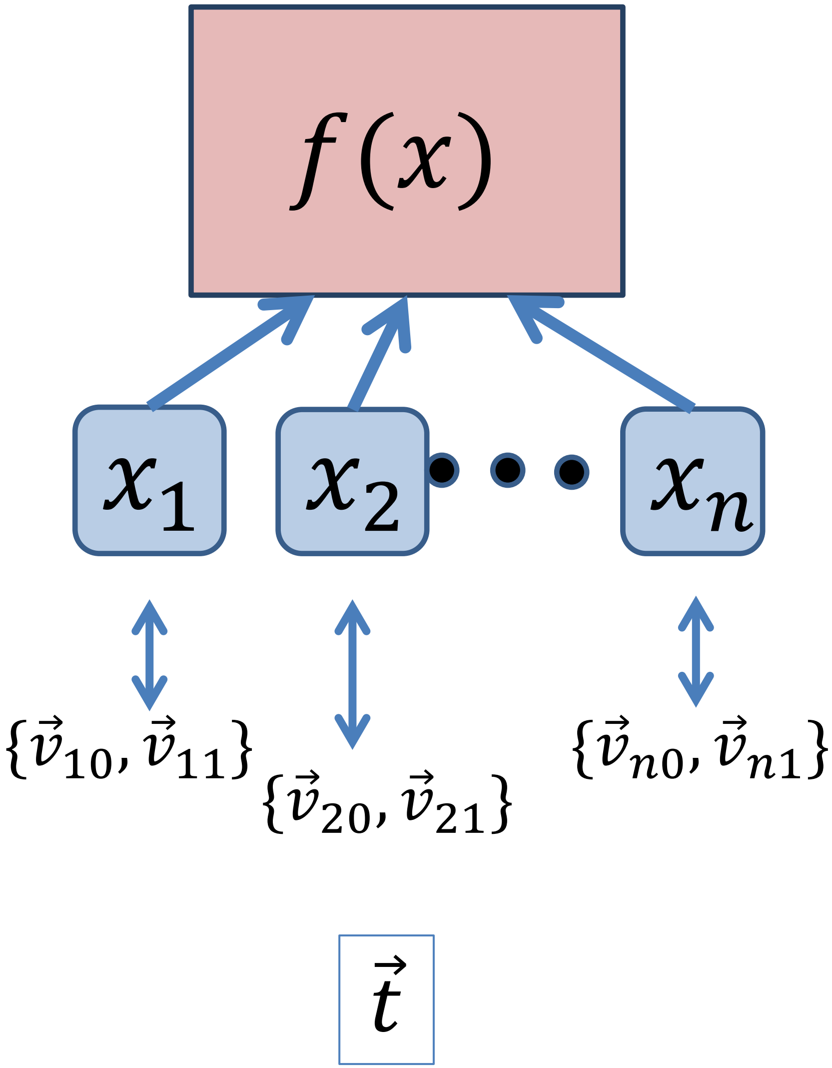
$$ADV^{\pm}(f)^k = O(T)$$

$$ADV^{\pm}(f) = O(T^{1/k})$$

Matching Algorithm?

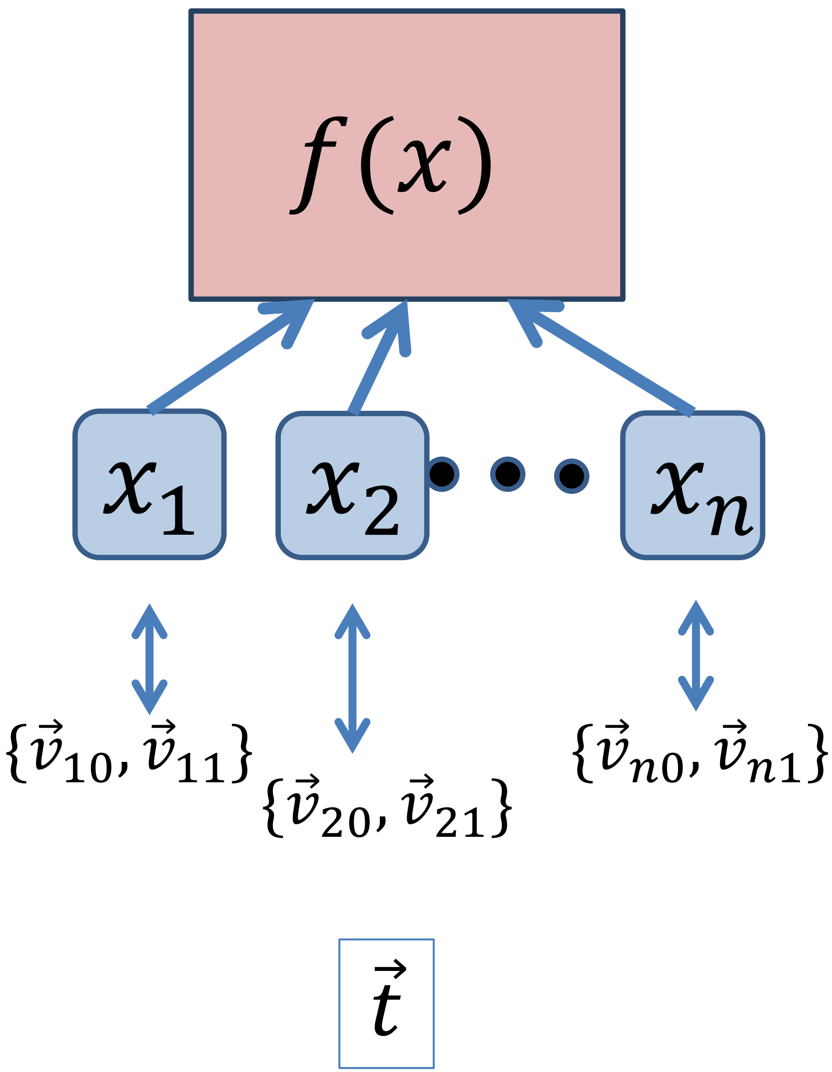
- For all c-Fault NAND Trees, O(1) query algorithms must exist.
- Can we find them?

Method 1: Span Programs [Zhan, Hassidim, SK '12, SK, '13]



$f(\vec{x}_i) = 1$ iff
$\vec{t} \in SPAN\{\vec{v}_{1i}, \vec{v}_{2i}, \ldots, \vec{v}_{ni}\}$

Method 1: Span Programs [Zhan, Hassidim, SK '12, SK, '13]



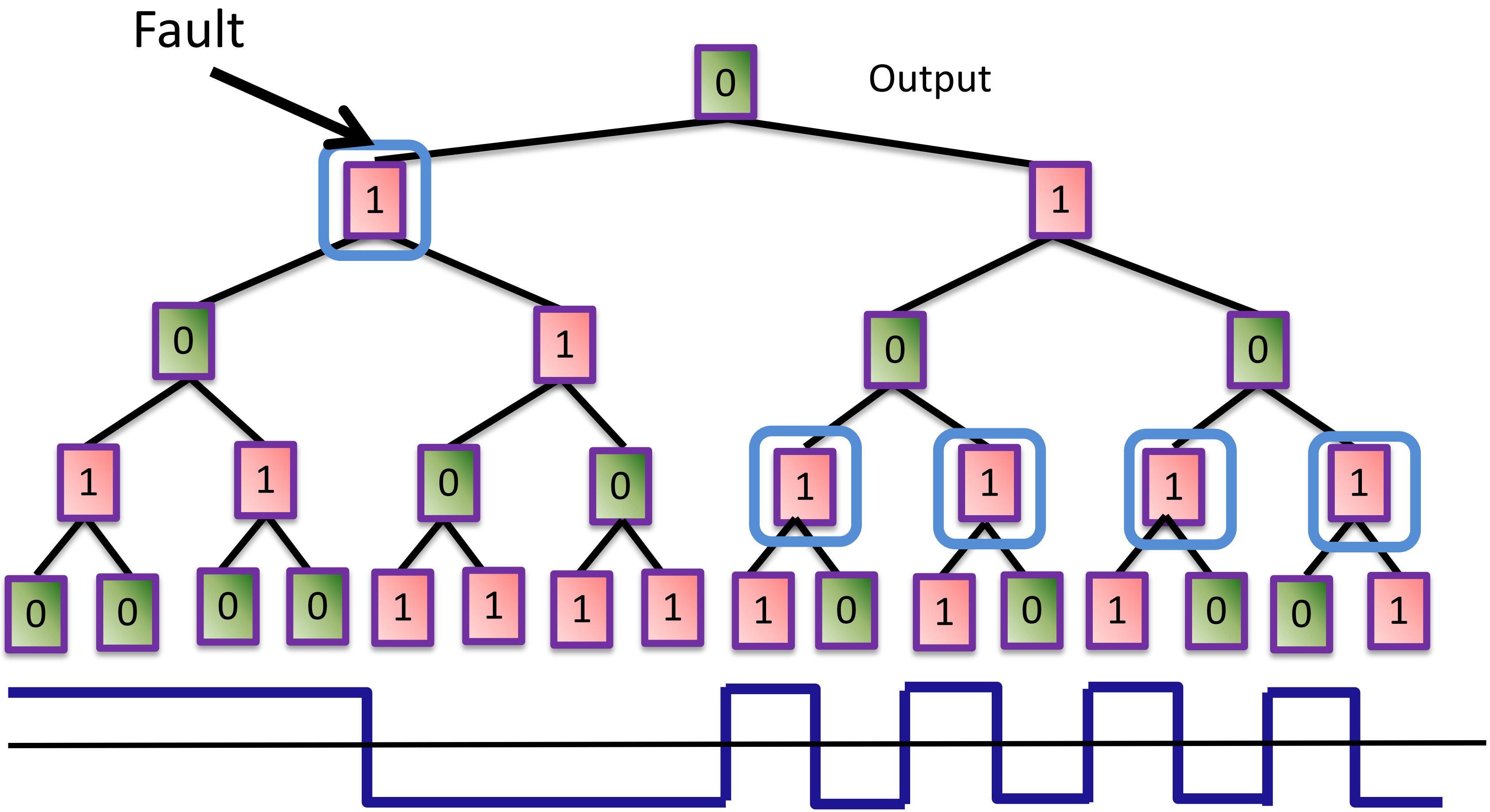
$f(\vec{x}_i) = 1$ iff
$\vec{t} \in SPAN\{\vec{v}_{1i}, \vec{v}_{2i}, \ldots, \vec{v}_{ni}\}$

AND:

$$\vec{v}_{11} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \vec{v}_{21} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \vec{t} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

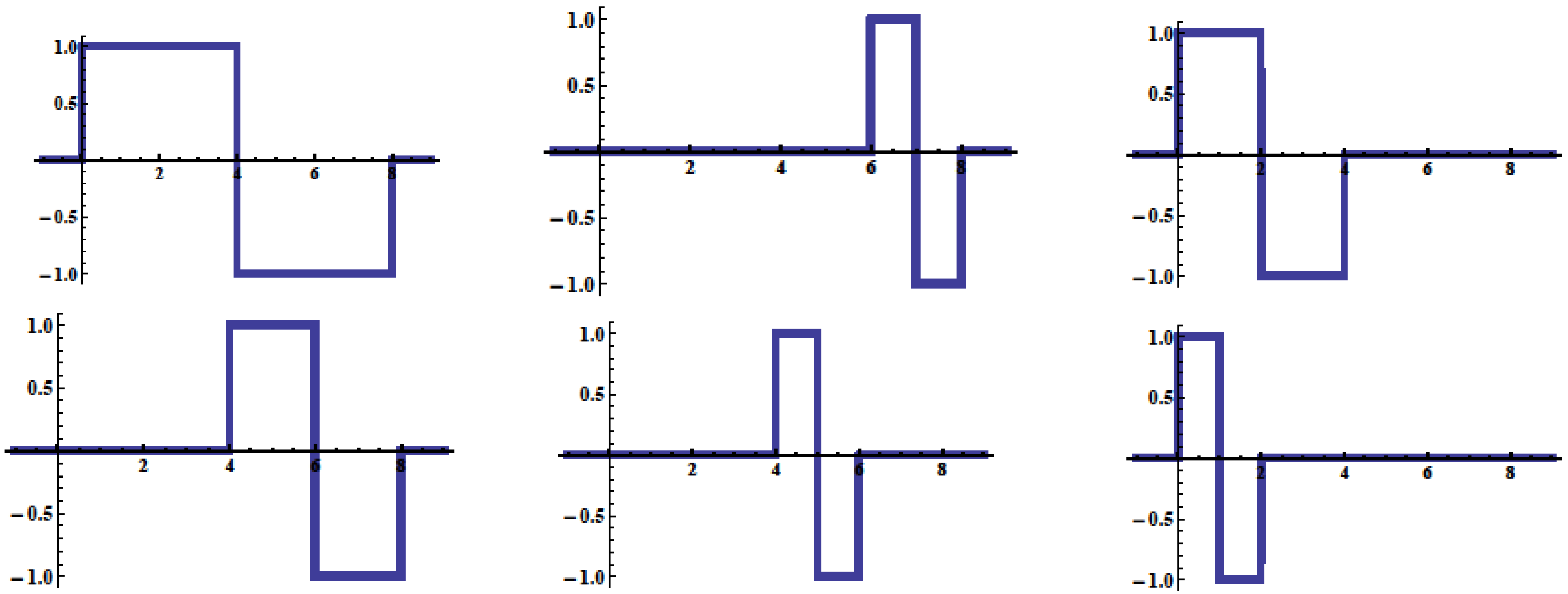
All other: $\binom{0}{0}$

Method 2: Haar Transform

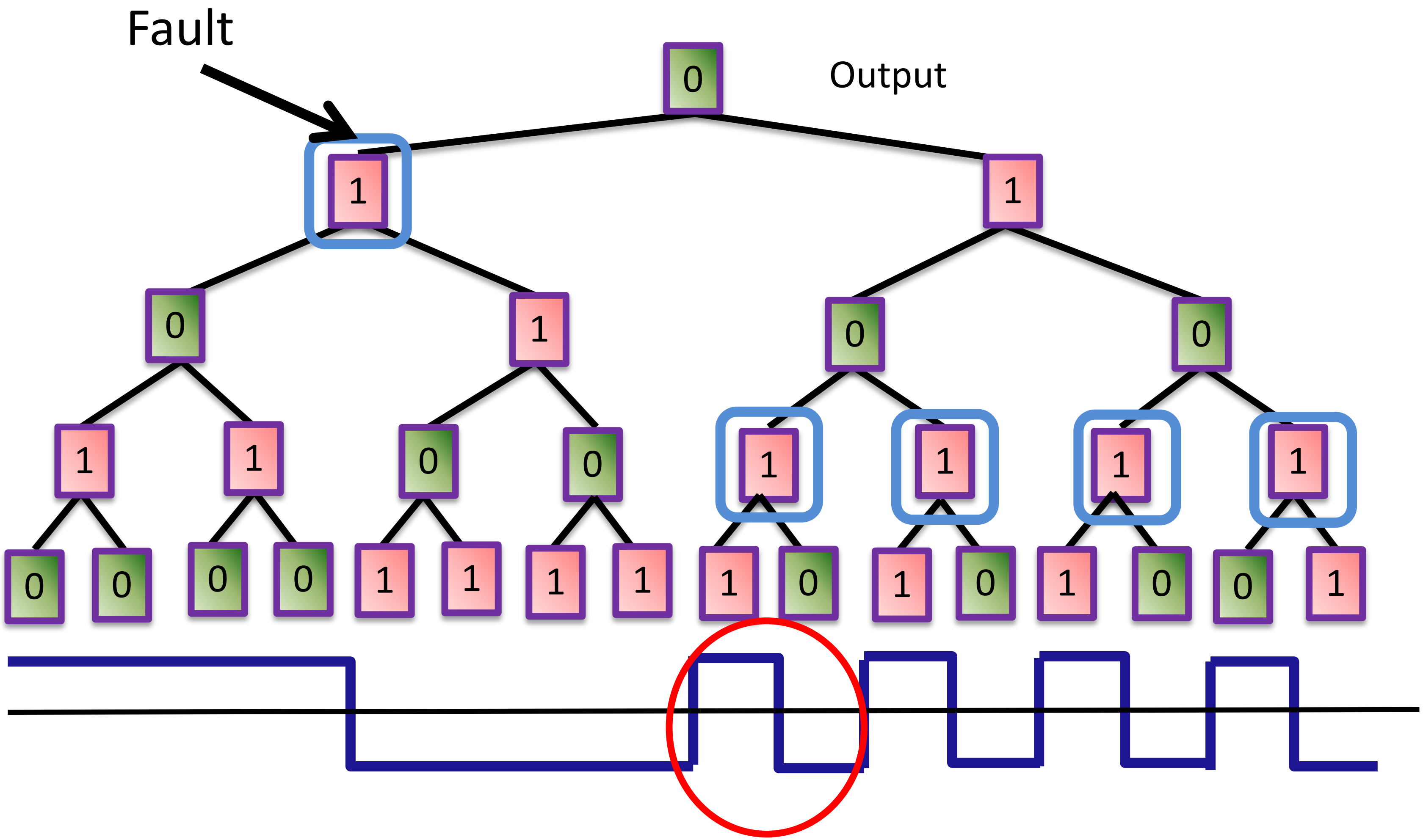


Method 2: Haar Transform

- Start in superposition: $\frac{1}{\sqrt{n}} \sum |i\rangle$.
- Apply Oracle. Phases= 
- Measure in Haar Basis (efficient, Hoyer ’97)

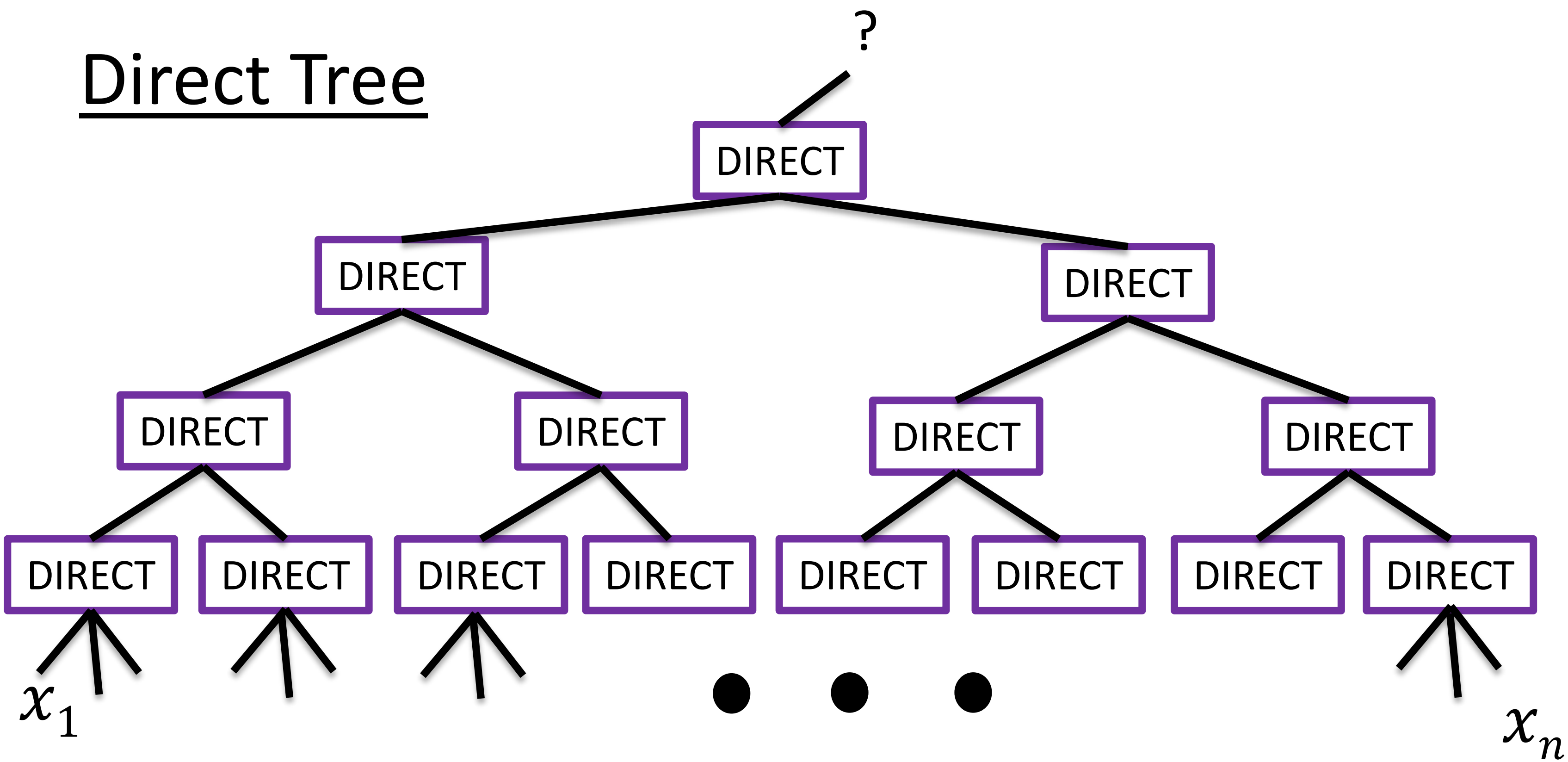


Method 2: Haar Transform



Extension: c-Fault Direct Tree

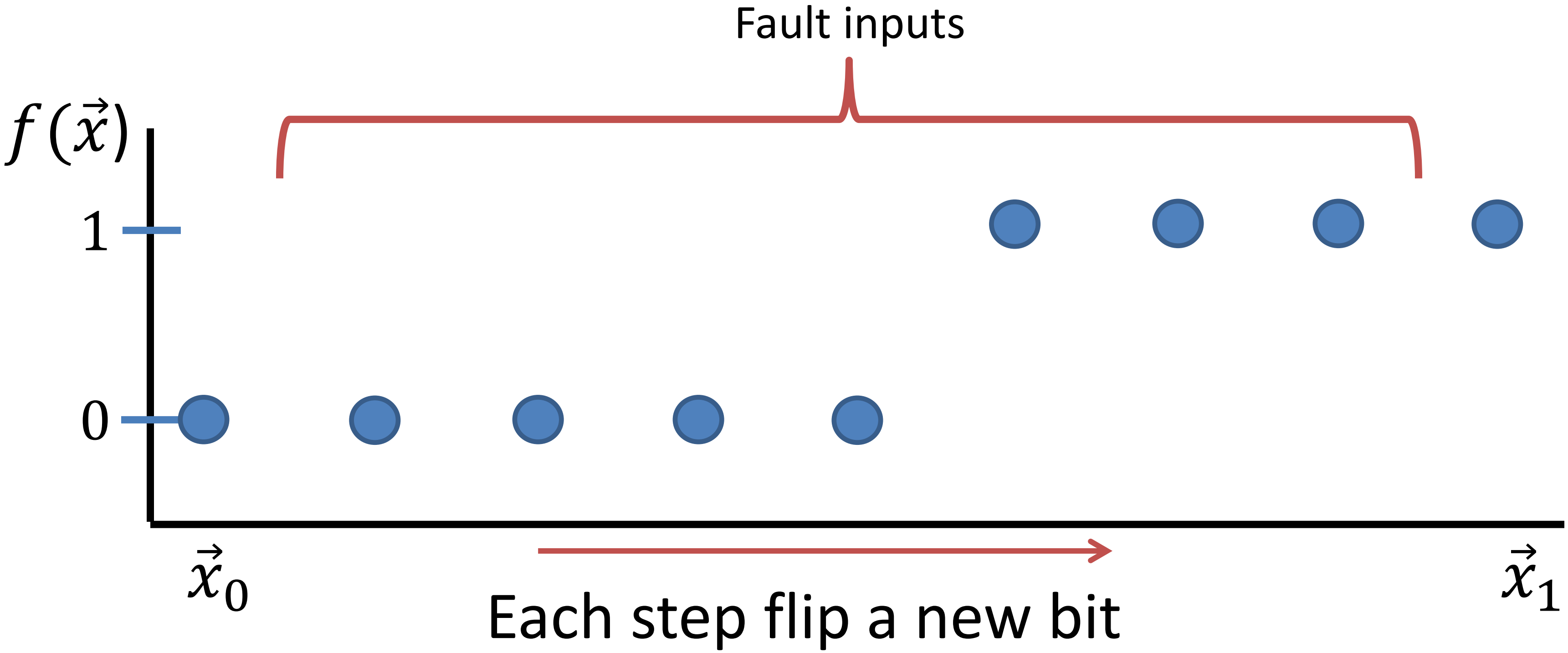
Direct Tree



DIRECT → generalization of monotonic.

Direct Functions

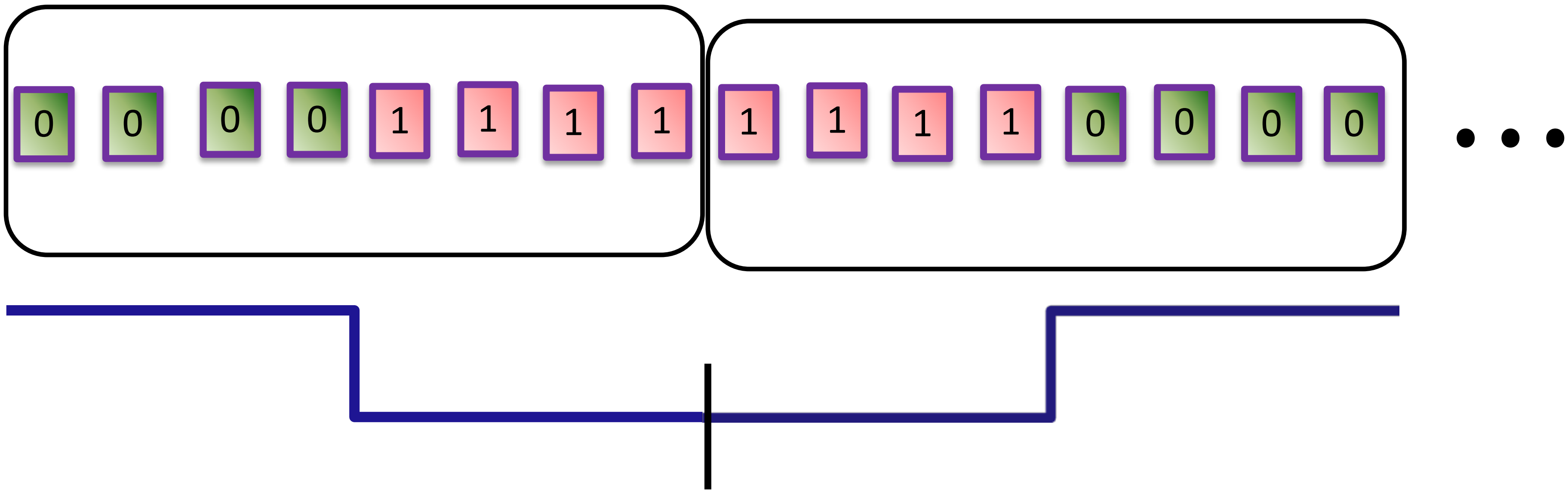
- Examples: Majority, NOT-Majority
- Generalization of monotonic



Open Questions: Unique Result?

- Classically is it possible to prove the existence of an algorithm without creating it?
  - Probabilistic/Combinatorial algorithms can prove that queries exist that will give an optimal algorithm, but would need to do a brute-force search to find them [Grebinski and Kucherov, ‘97]

Application: Period Finding



Summary and Open Questions

- Quantum adversary upper bound can prove the existence of quantum algorithms
  - 1-Fault NAND Tree
  - Other constant fault trees

- Are there other problems where this technique will be useful?
- Do the matching algorithms have other applications?
- Other Adversary SDP applications?

Types of Quantum Algorithms

Structured Algorithms

- Shor's Algorithm
- Hidden Subgroup
- Phase Estimation

Unstructured Algorithms

- Grover's Algorithm
- Element Distinctness

By understanding the structure underlying quantum algorithms, can we find and design new algorithms?

Future Work

- This result uses powerful tools and deep understanding of quantum algorithm
- BUT – model of computation is limited
- Use similar tools to understand new (and more realistic) models of quantum algorithms?