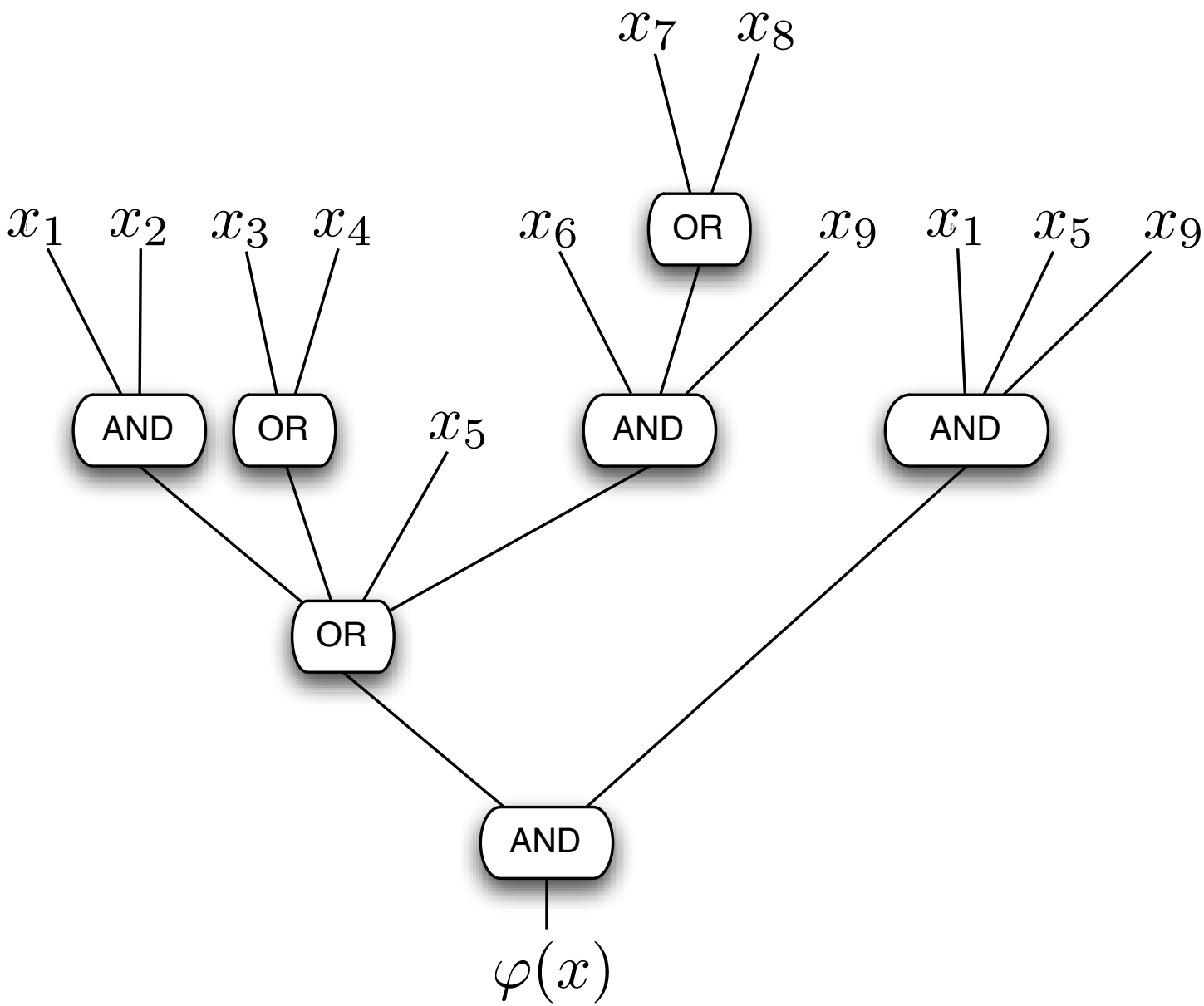
Quantum algorithms for formula evaluation

Ben Reichardt
Caltech

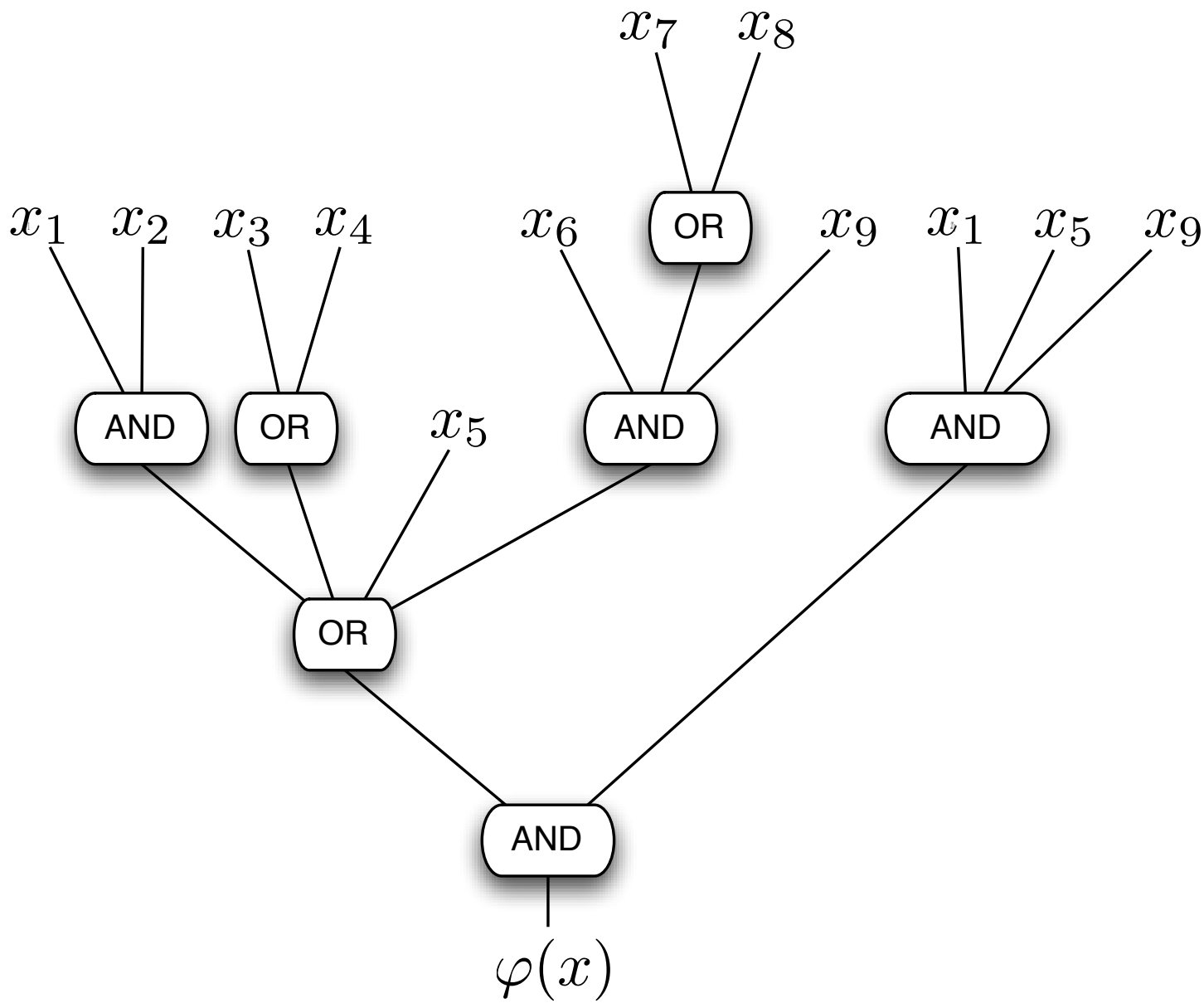
joint work with
Andrew Childs, Robert Špalek,
Shengyu Zhang

Kiyomizudera

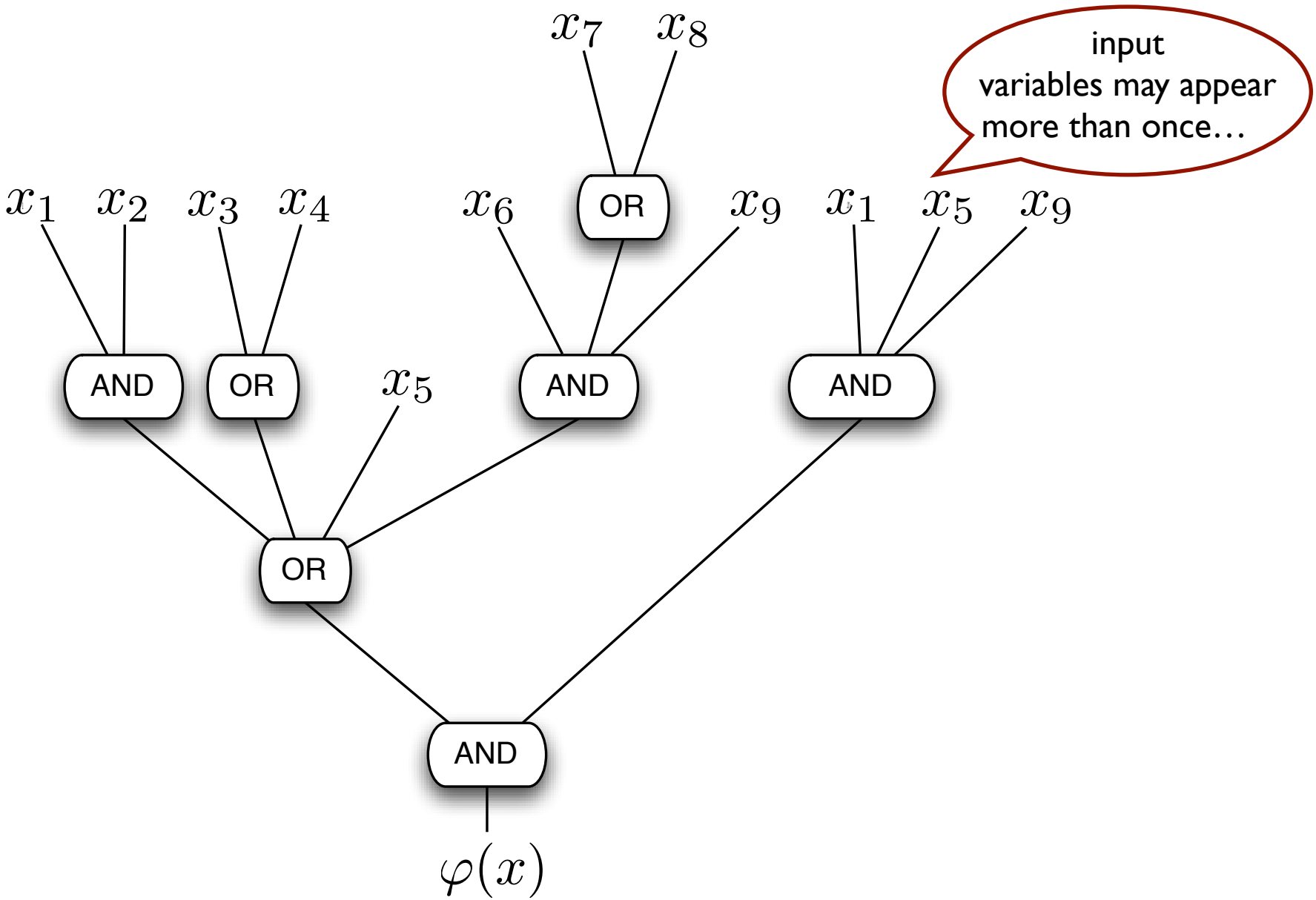
Problem: Evaluate the formula with minimal queries to the input bits x_i.



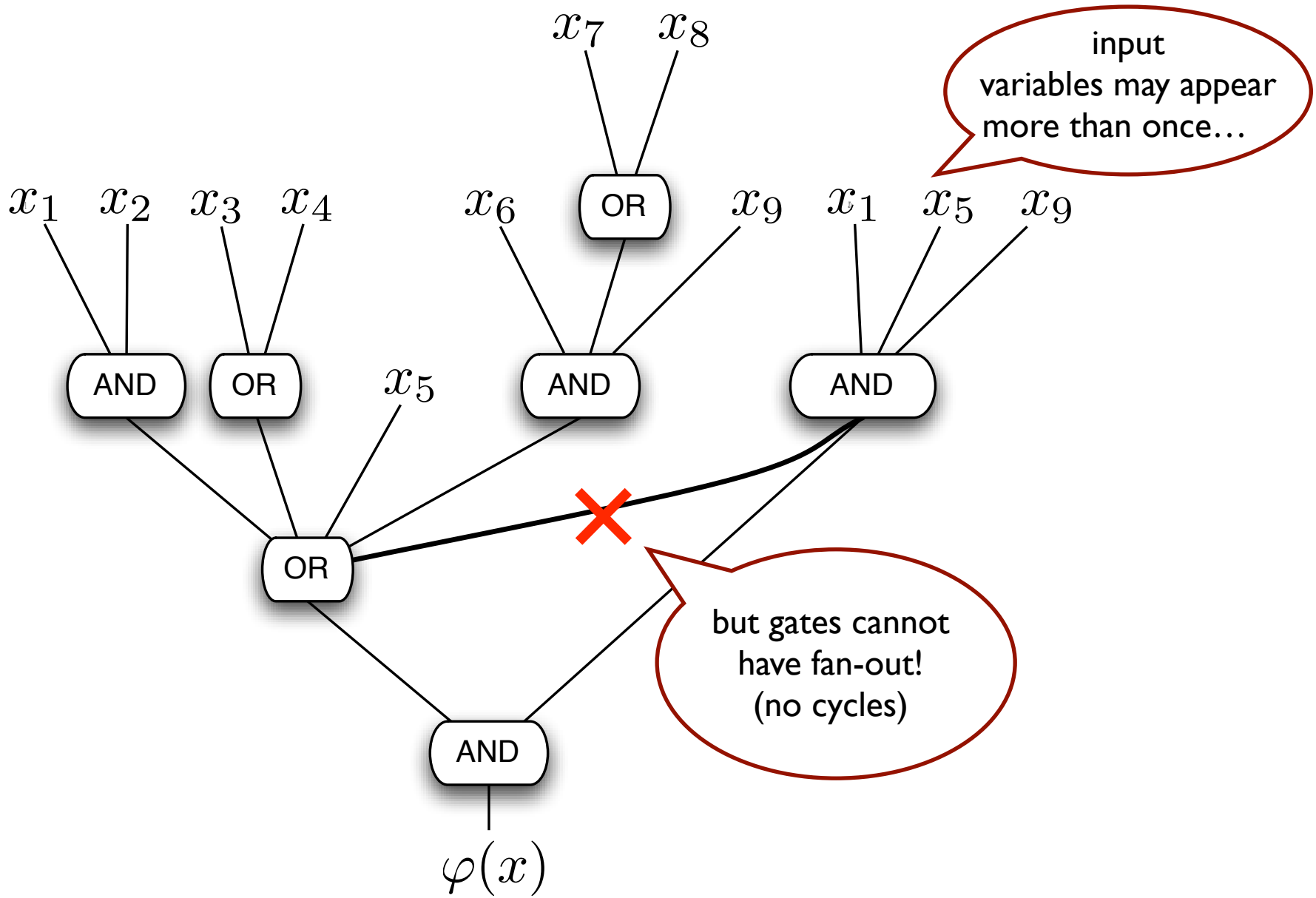
Def: {AND, OR, NOT} Formula = Tree of nested gates



Def: {AND, OR, NOT} Formula = Tree of nested gates

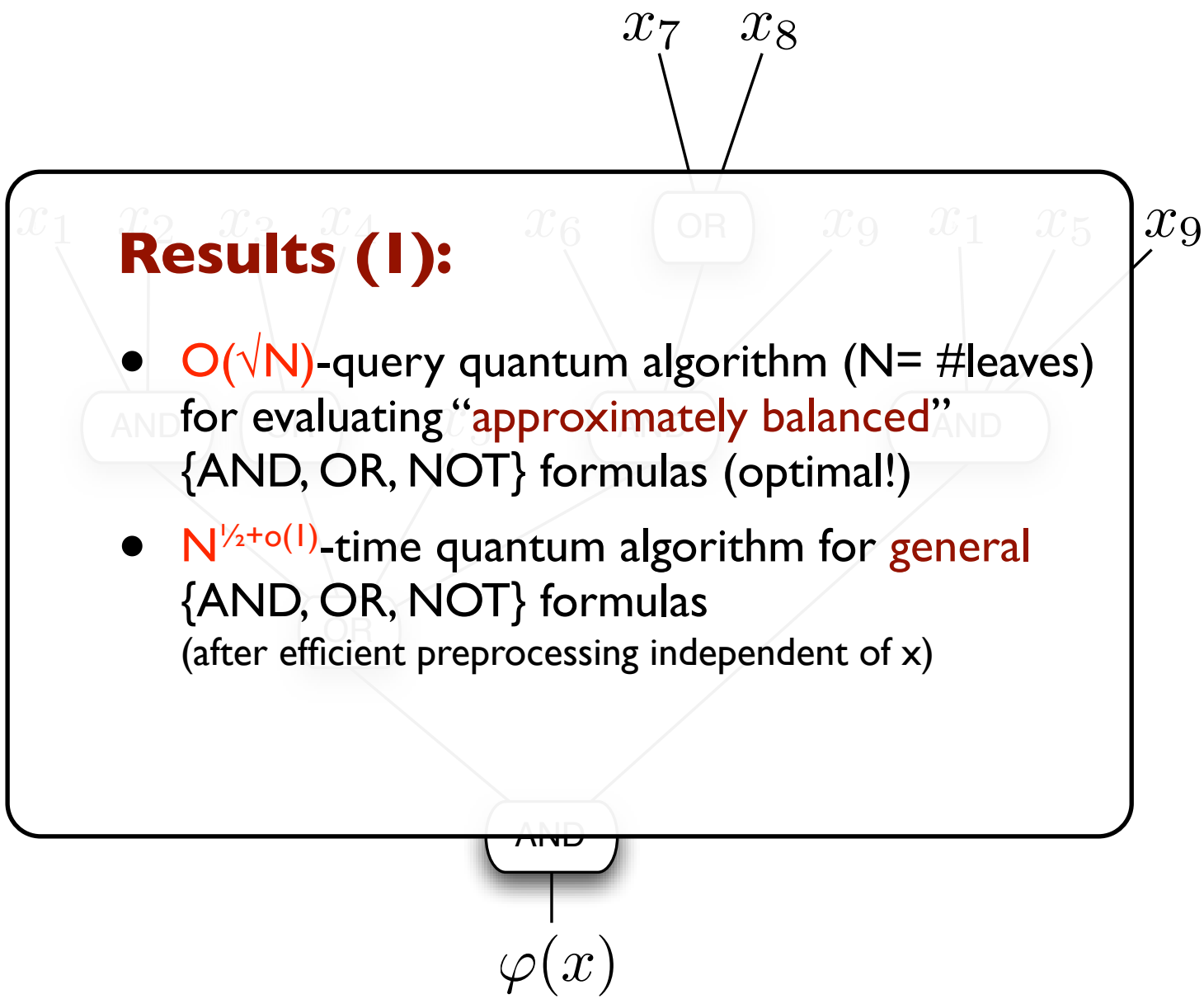


Def: {AND, OR, NOT} Formula = Tree of nested gates



(in a *circuit*, cycles are allowed; but in a *formula*, subexpressions cannot be reused)

Problem: Evaluate the formula with minimal queries to the input bits x_i.



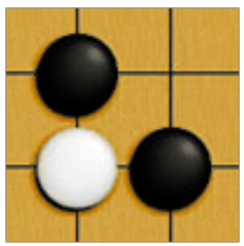
Results (1):

- $O(\sqrt{N})$-query quantum algorithm (N= #leaves) for evaluating "approximately balanced" {AND, OR, NOT} formulas (optimal!)
- $N^{1/2+o(1)}$-time quantum algorithm for general {AND, OR, NOT} formulas
  (after efficient preprocessing independent of x)

- Problem: Evaluate the formula, with minimal queries to the inputs bits x_i.
- Results:
  - $O(\sqrt{N})$-query quantum algorithm for "approximately balanced" AND-OR formulas
  - $N^{1/2+o(1)}$-time quantum algorithm for general AND-OR formulas (after preprocessing)

Problem Motivations:

- Playing "Go"



  - Nodes ↔ game histories
  - White wins if ∃ move s.t. ∀ black moves, ∃ move s.t. …

➡ Two-player game trees are formula trees with alternating levels of AND, OR gates

- Decision version of min-max tree evaluation
  - inputs are real numbers
  - want to decide if minimax is ≥10 or not
- Well-studied classical problem…

Problem history (1/2)

- Problem: Evaluate the formula, with minimal queries to the inputs bits x_i.
- Results:
  - $O(\sqrt{N})$-query quantum algorithm for "approximately balanced" AND-OR formulas
  - $N^{1/2+o(1)}$-time quantum algorithm for general AND-OR formulas (after preprocessing)
- Classical history
  - Deterministic algorithm requires time N
  - Randomized (Las Vegas) algorithm in E-time $O(N^{0.754})$ for balanced binary AND-OR formulas [Snir '85, Saks & Wigderson '86]
    - Flip coins to decide which subtree to evaluate next, short-circuit
    - Optimal [Santha '95]
  - For arbitrary AND-OR formulas $\Omega(N)$ time may be required

$$\log_d \lambda_{max}\left(\begin{pmatrix} 0 & d \\ 1 & \frac{d-1}{2} \end{pmatrix}\right)$$

Problem history (2/2)

- Classical history
  - Randomized algorithm in E-time $\Theta(N^{0.754})$ for balanced binary AND-OR formulas
  - Evaluating an arbitrary AND-OR formula may require $\Omega(N)$ time
- Quantum history
  - Adversary lower bound $\Omega(\sqrt{N})$ queries [Barnum, Saks '04]
  - Grover search: Evaluates $OR(x_1, x_2, \ldots, x_N) = \begin{cases} 1 & \text{if } \exists \text{ an } i : x_i = 1 \\ 0 & \text{otherwise} \end{cases}$

    using $O(\sqrt{N})$ queries ($O(\sqrt{N} \log \log N)$-time)
  - Can be applied recursively to evaluate shallow trees:
    - Evaluates regular depth-d AND-OR formula in $\sqrt{N}\, O(\log N)^{d-1}$ queries [BCW '98]
    - Search on faulty oracles [Høyer, Mosca, de Wolf '03] $\Rightarrow O(\sqrt{N}\, c^d)$ queries

Breakthrough!

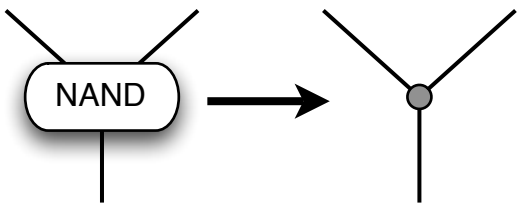
- Classical history
  - Randomized algorithm in E-time $\Theta(N^{0.754})$ for balanced binary AND-OR formulas
  - Evaluating an arbitrary AND-OR formula may require $\Omega(N)$ time
- Quantum history
  - Adversary lower bound $\Omega(\sqrt{N})$ queries [Barnum, Saks '04]
  - Grover search: Evaluates $OR(x_1, x_2, \ldots, x_N) = \begin{cases} 1 & \text{if } \exists \text{ an } i : x_i = 1 \\ 0 & \text{otherwise} \end{cases}$

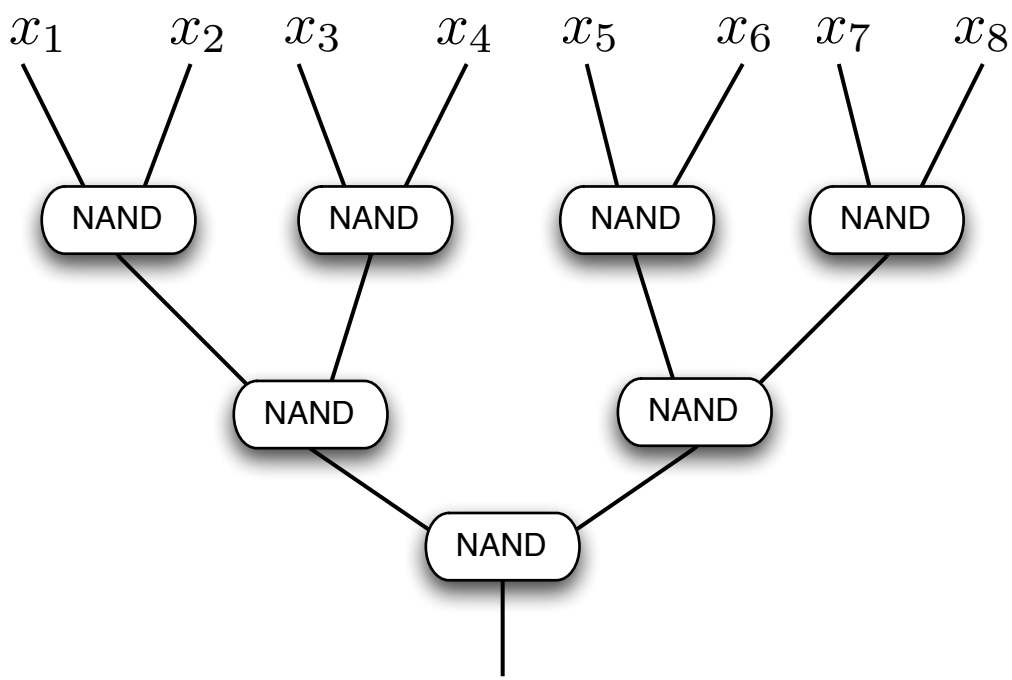
    using $O(\sqrt{N})$ queries ($O(\sqrt{N} \log \log N)$-time)
  - Can be applied recursively to evaluate shallow trees

- Farhi, Goldstone, Gutmann 2007: Breakthrough continuous-time quantum algorithm for evaluating balanced binary NAND formula in $N^{1/2+o(1)}$ queries & time

Farhi, Goldstone, Gutmann '07 algorithm

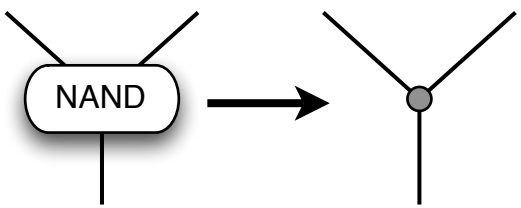
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:

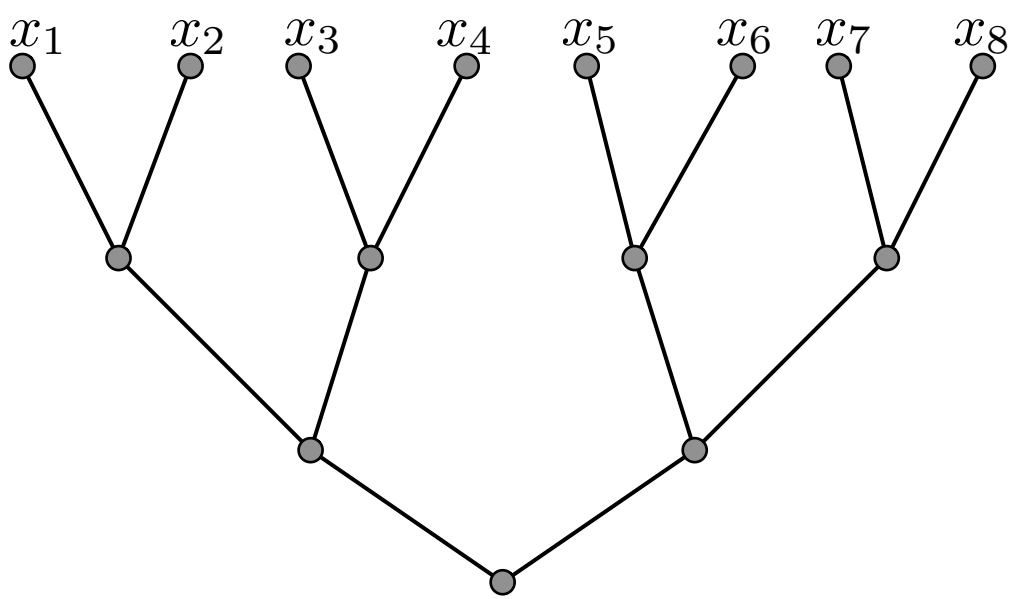




Farhi, Goldstone, Gutmann '07 algorithm

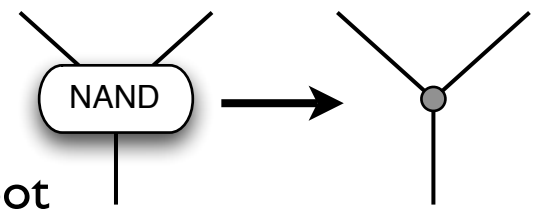
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:

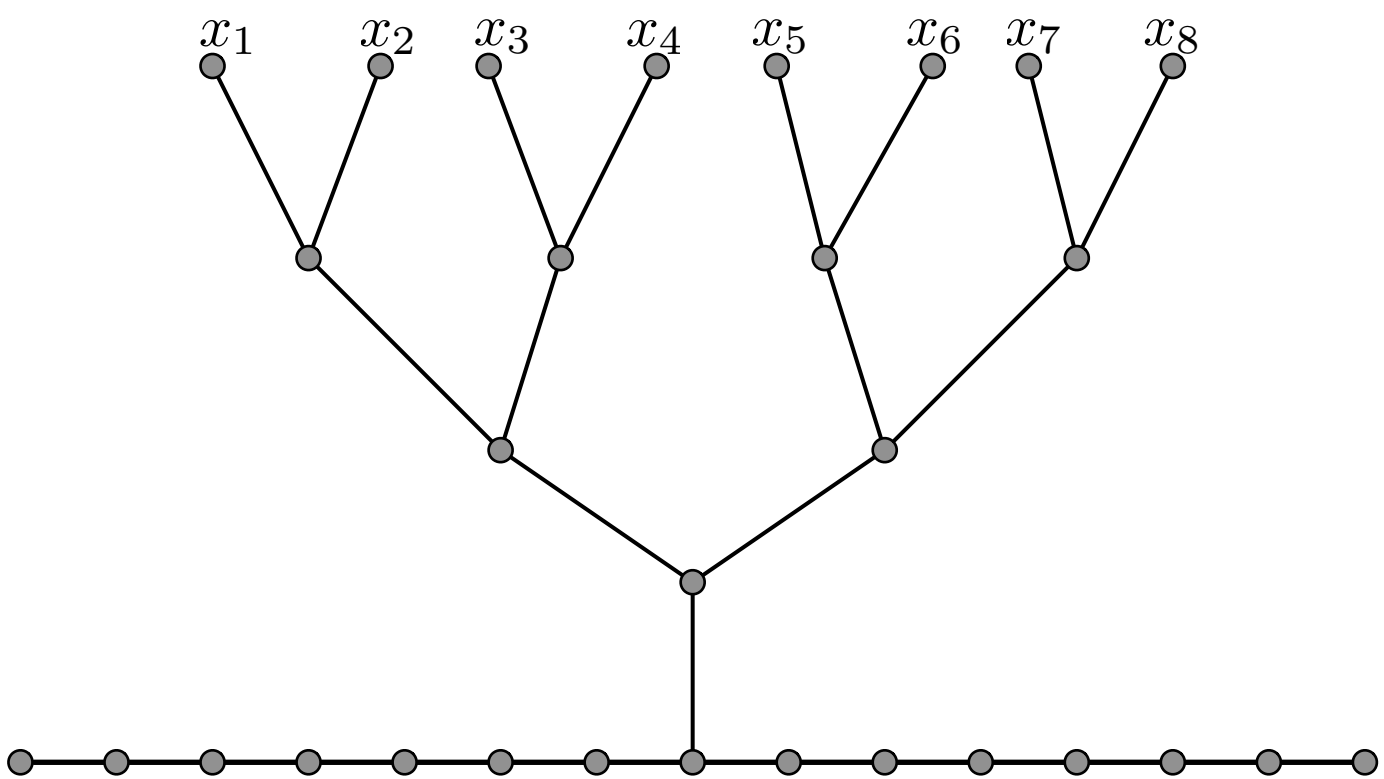




Farhi, Goldstone, Gutmann '07 algorithm

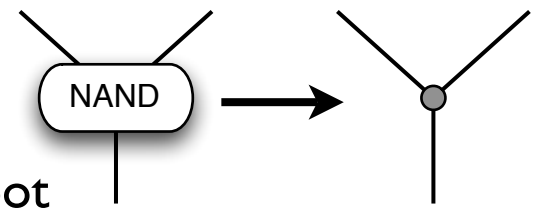
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:
- Attach an infinite line to the root

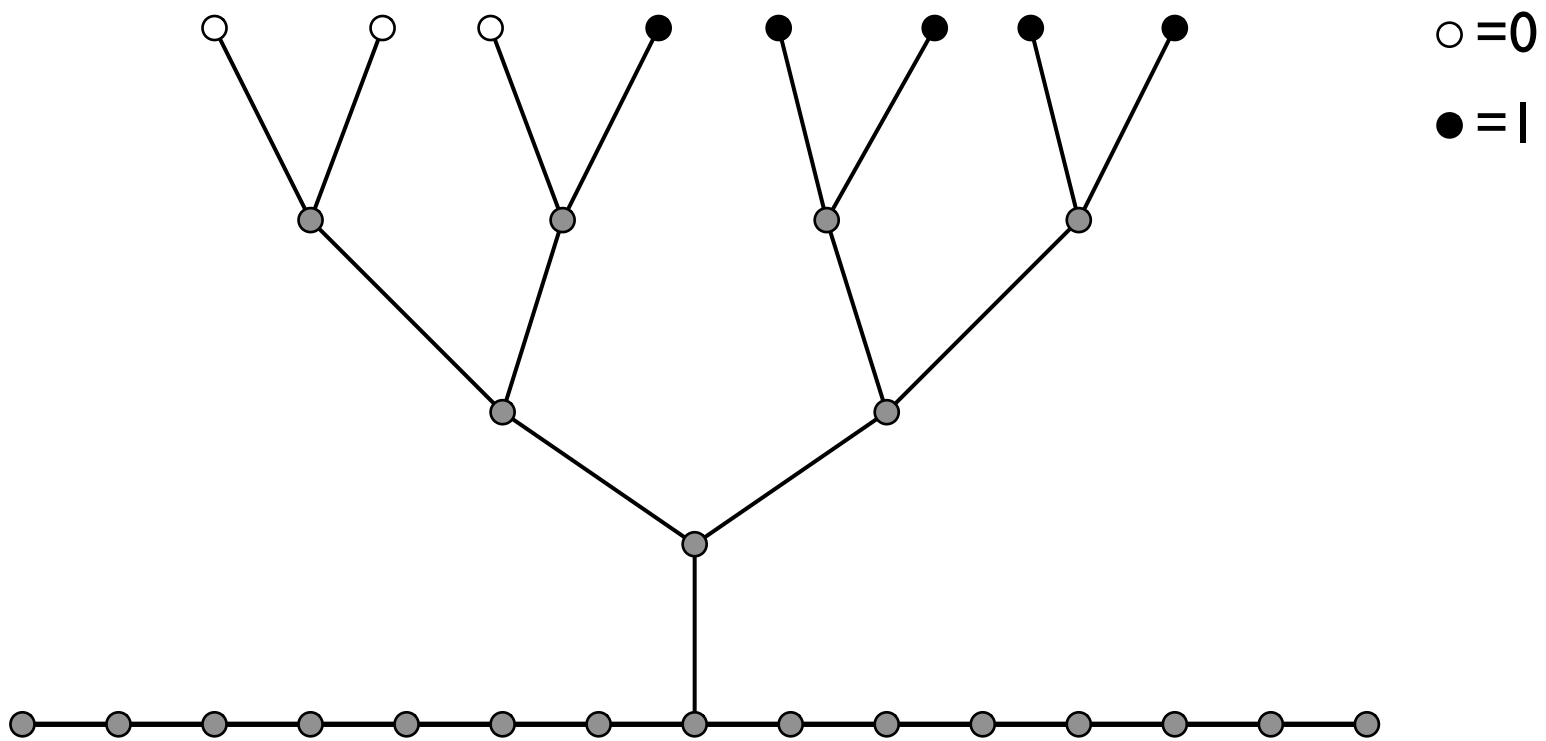




Farhi, Goldstone, Gutmann '07 algorithm

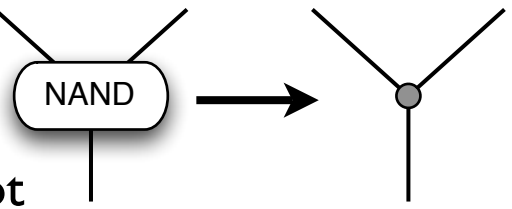
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one…

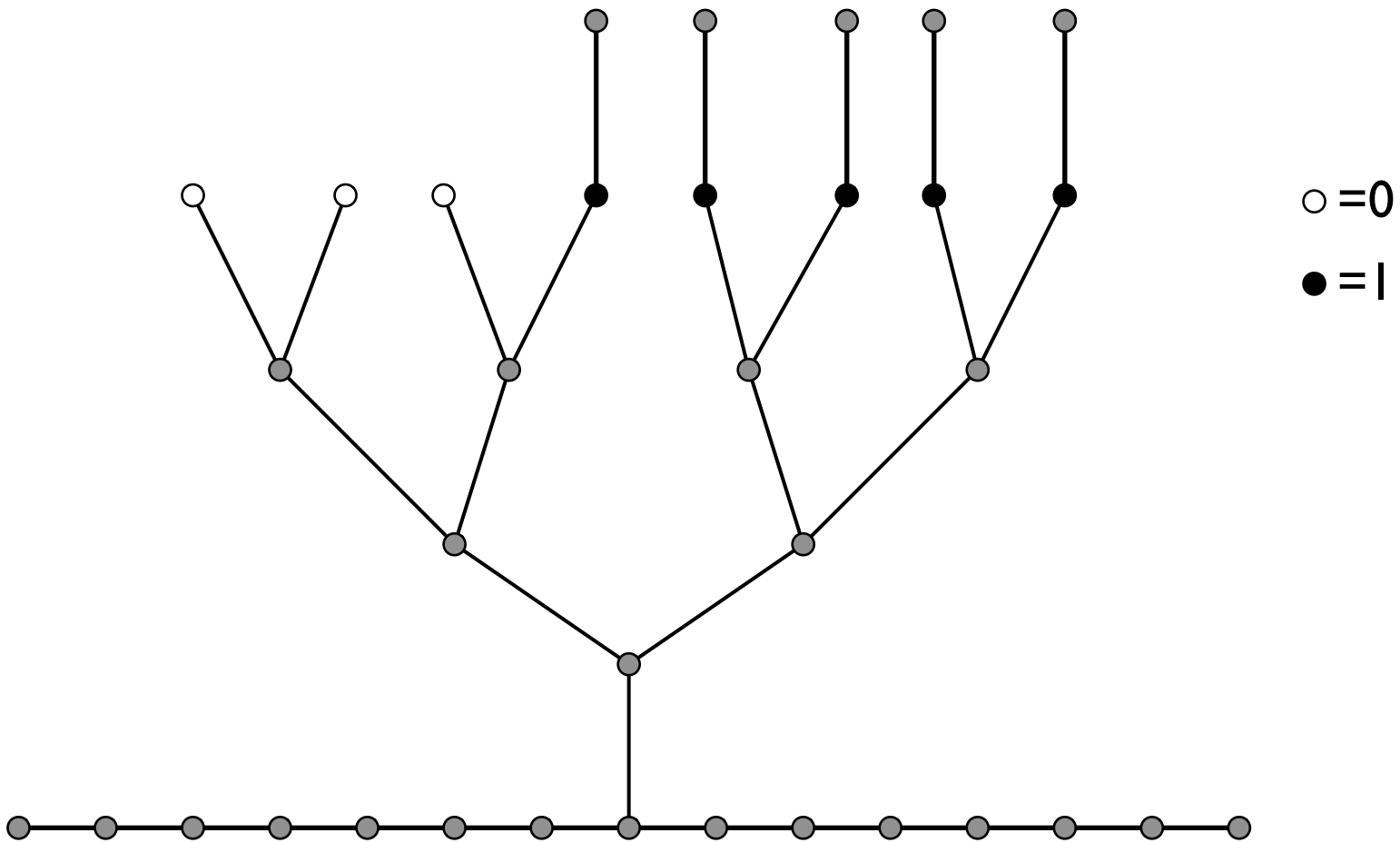




Farhi, Goldstone, Gutmann '07 algorithm

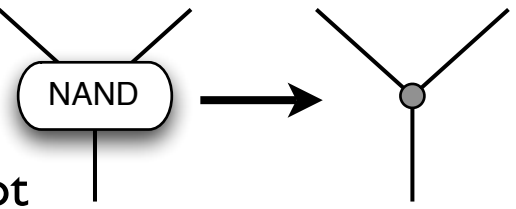
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one…

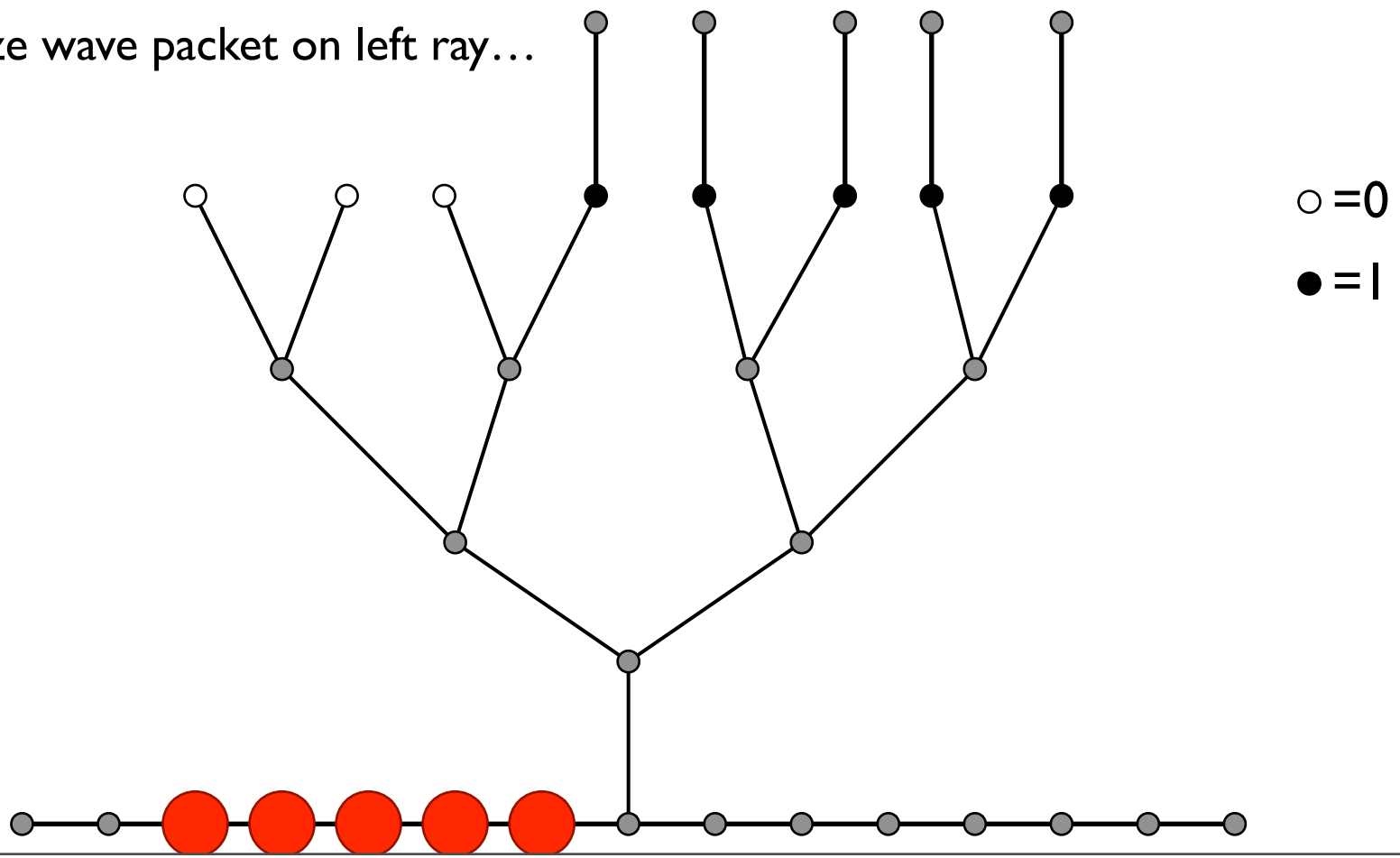




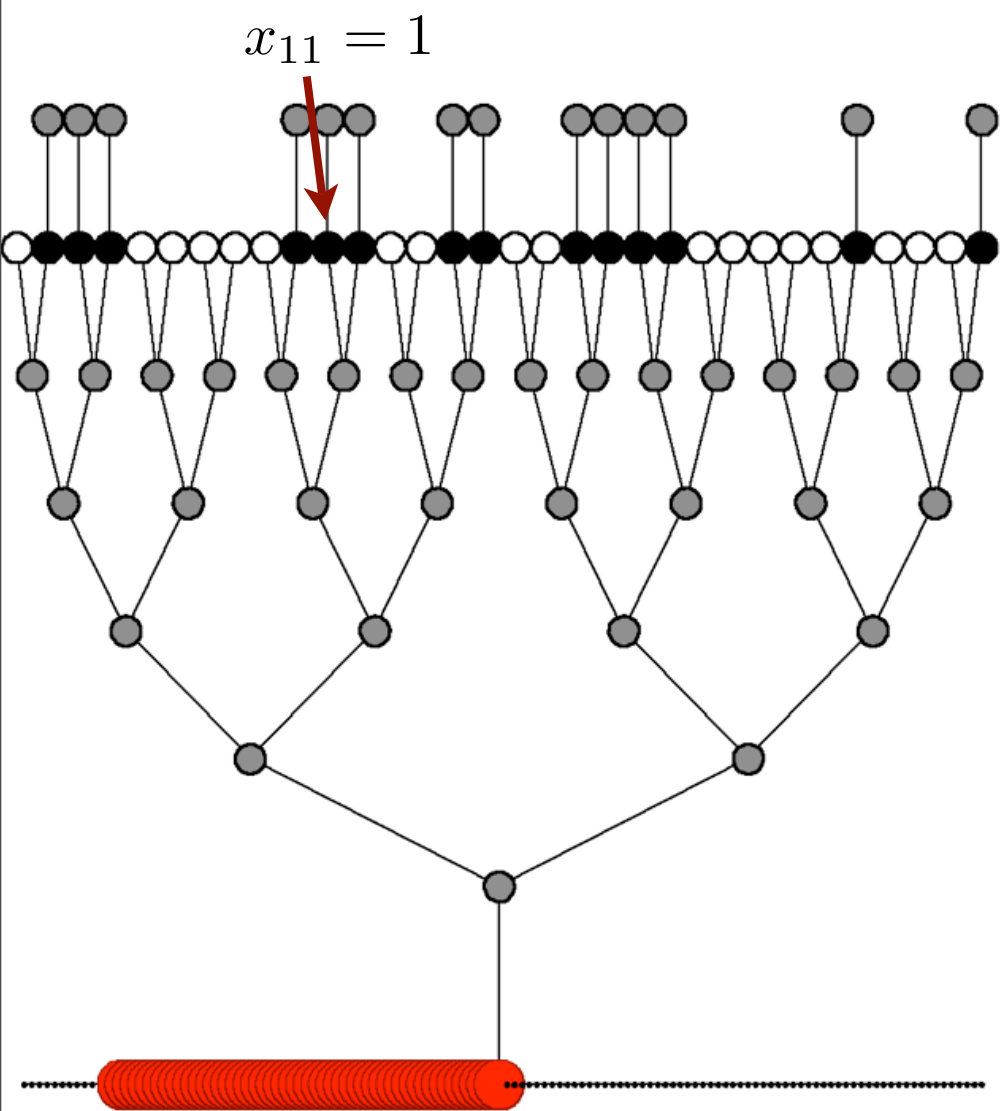
Farhi, Goldstone, Gutmann '07 algorithm

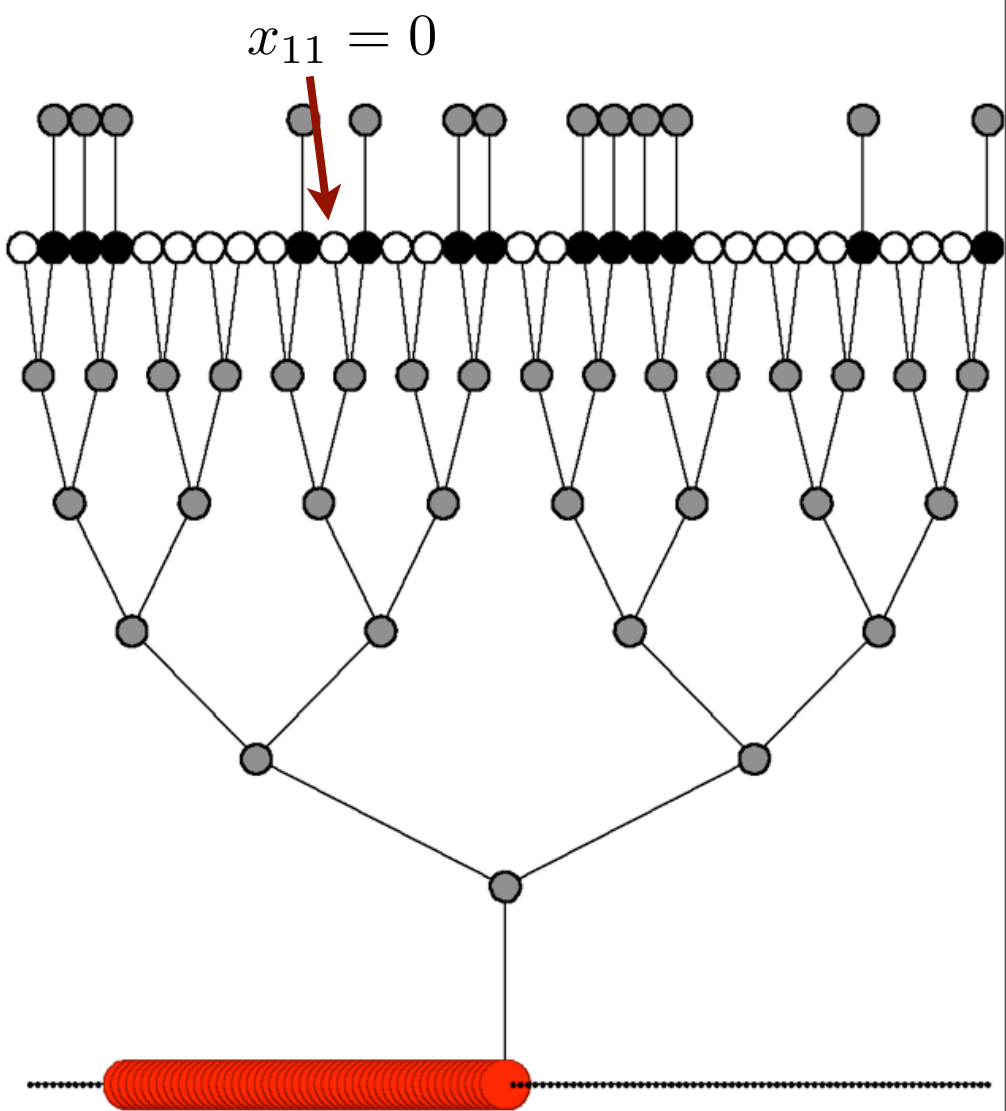
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.
- Convert formula to a tree:
- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one
- Initialize wave packet on left ray…



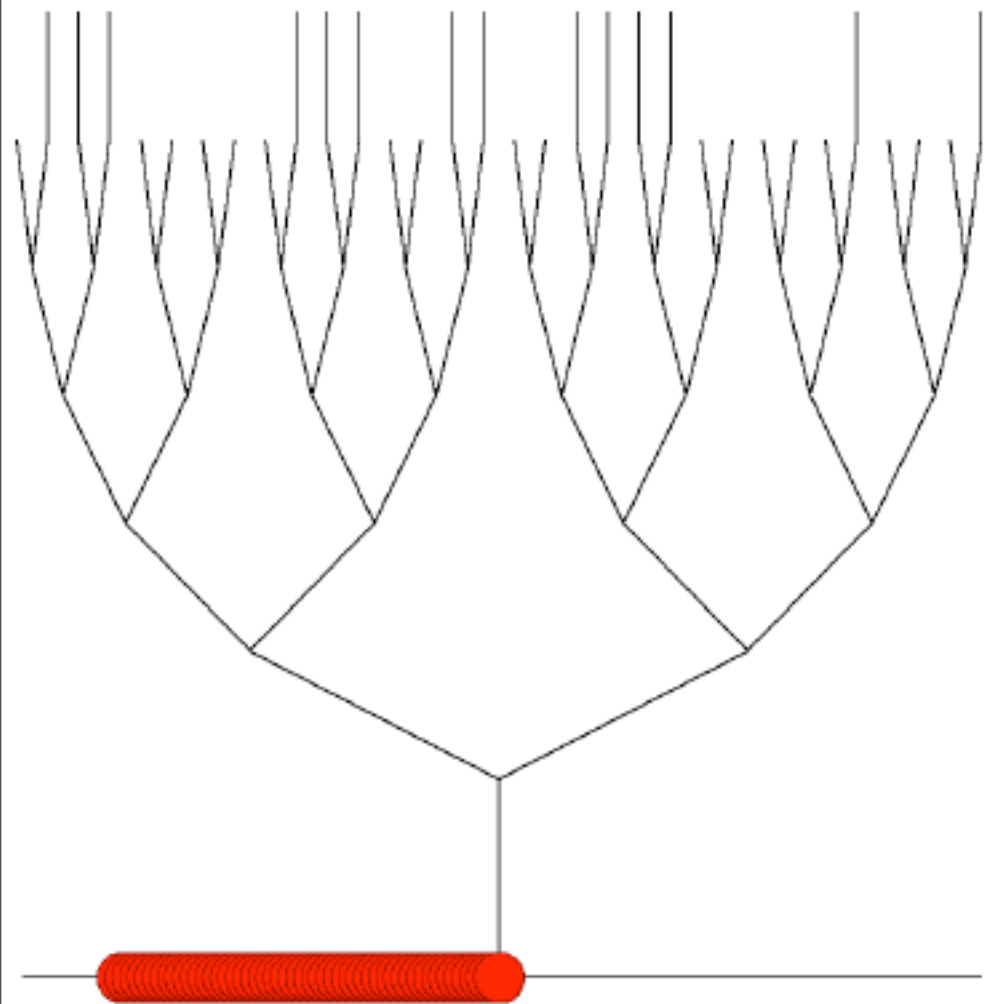


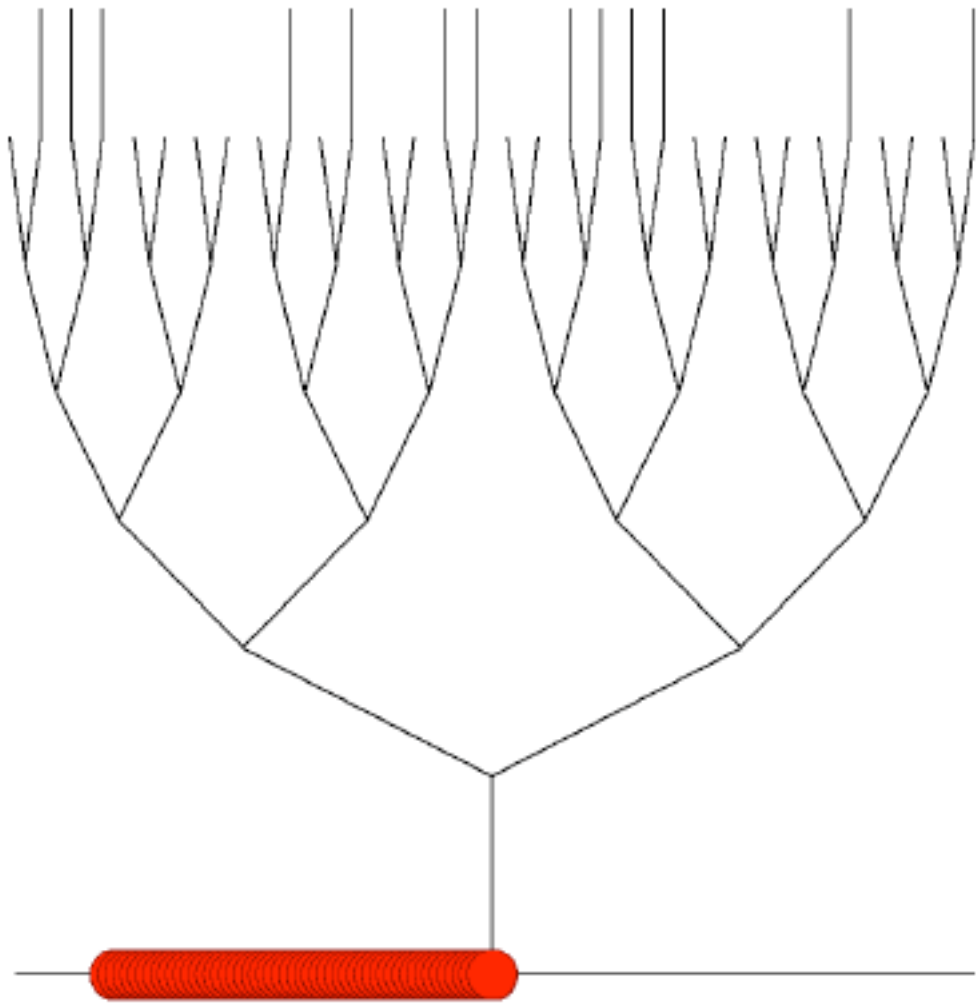
Continuous-time quantum walk [FGG '07]



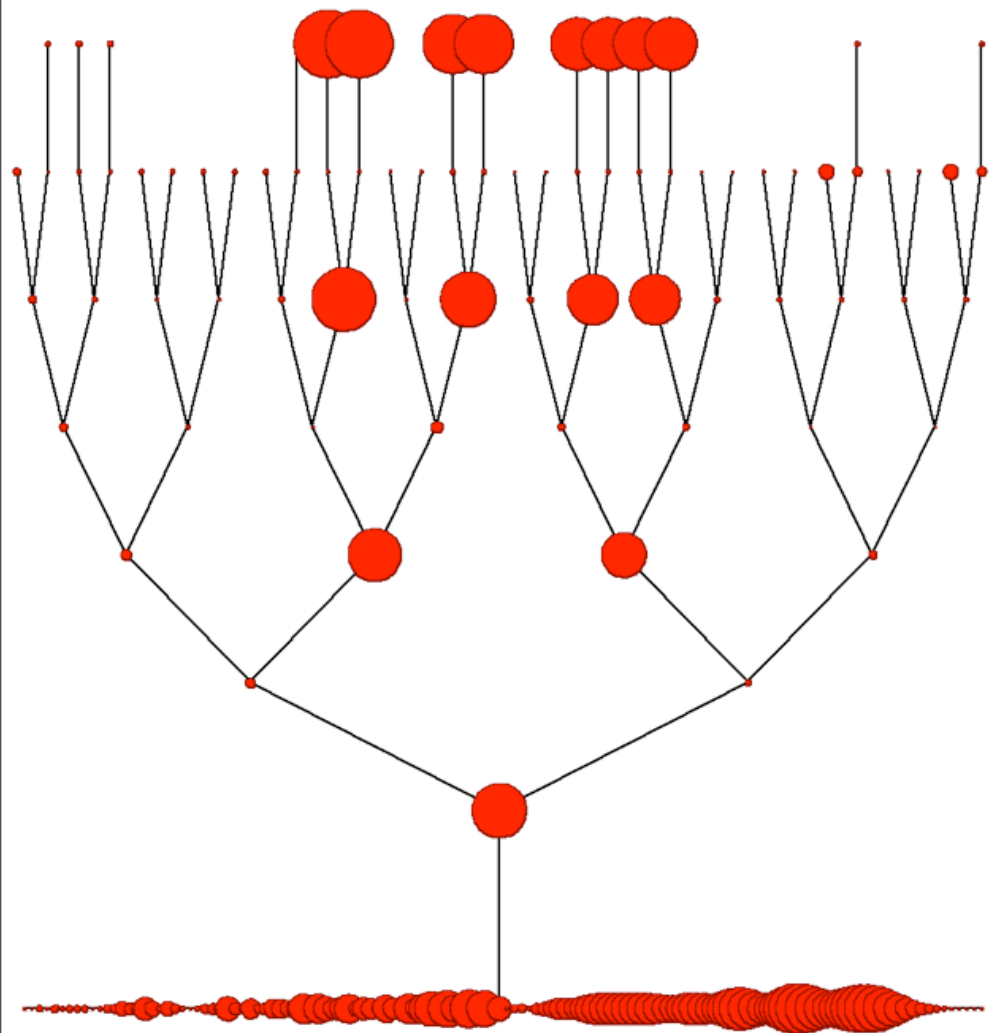


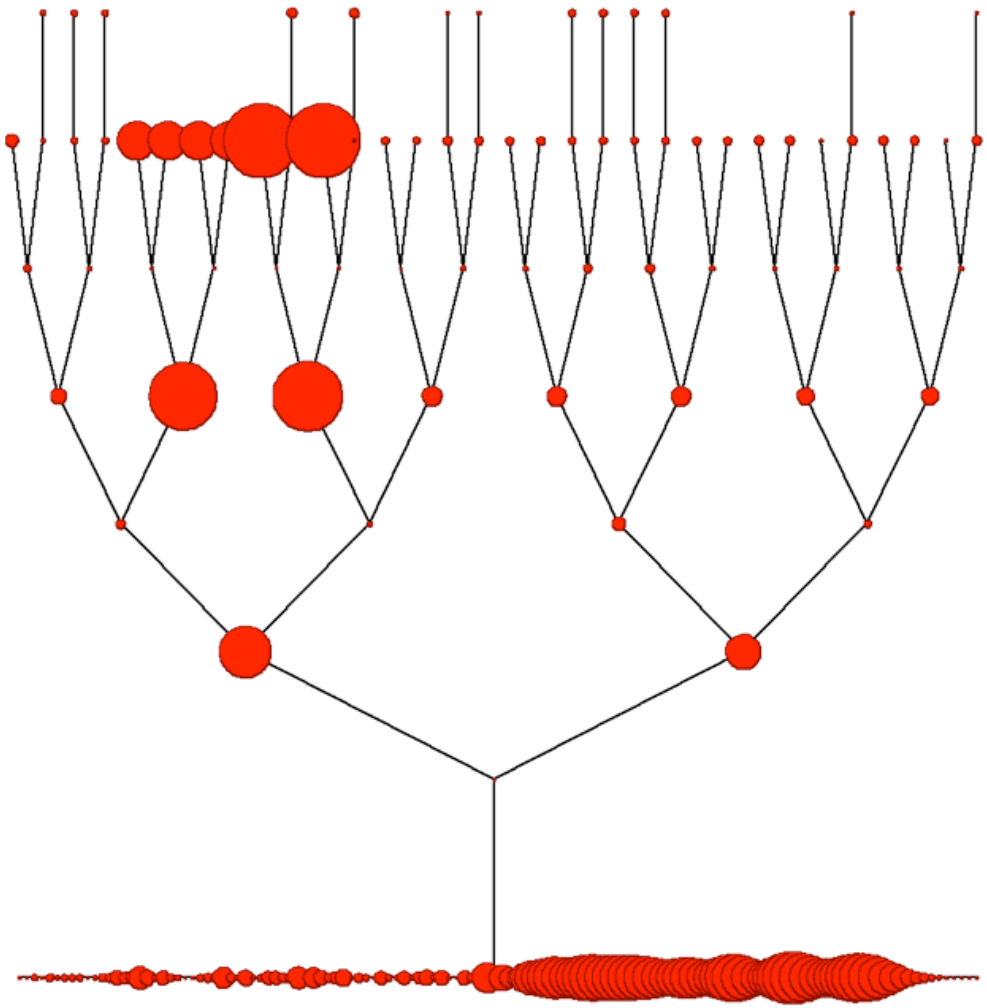
FGG quantum walk $|\psi_t\rangle = e^{iA_G t}|\psi_0\rangle$



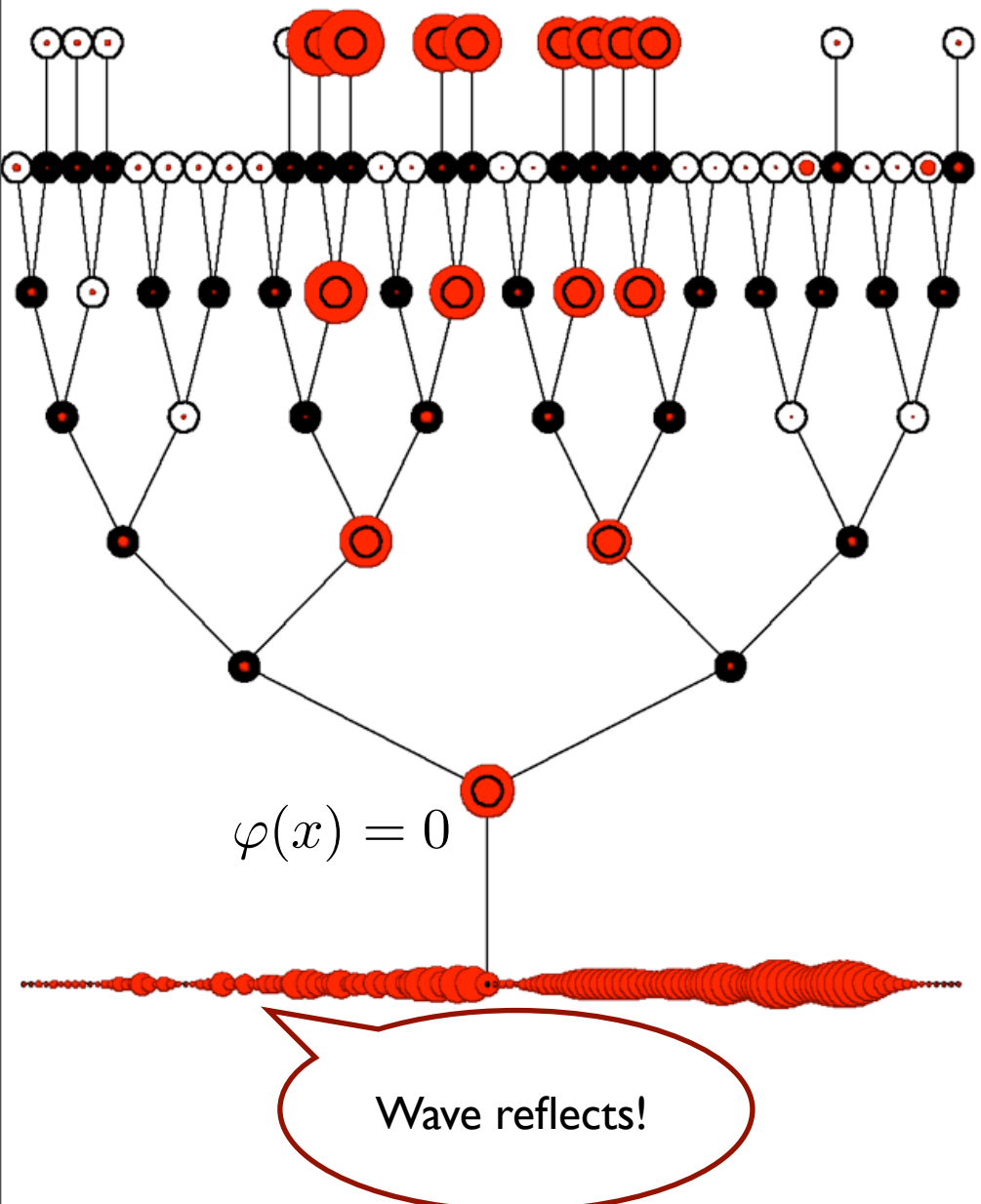


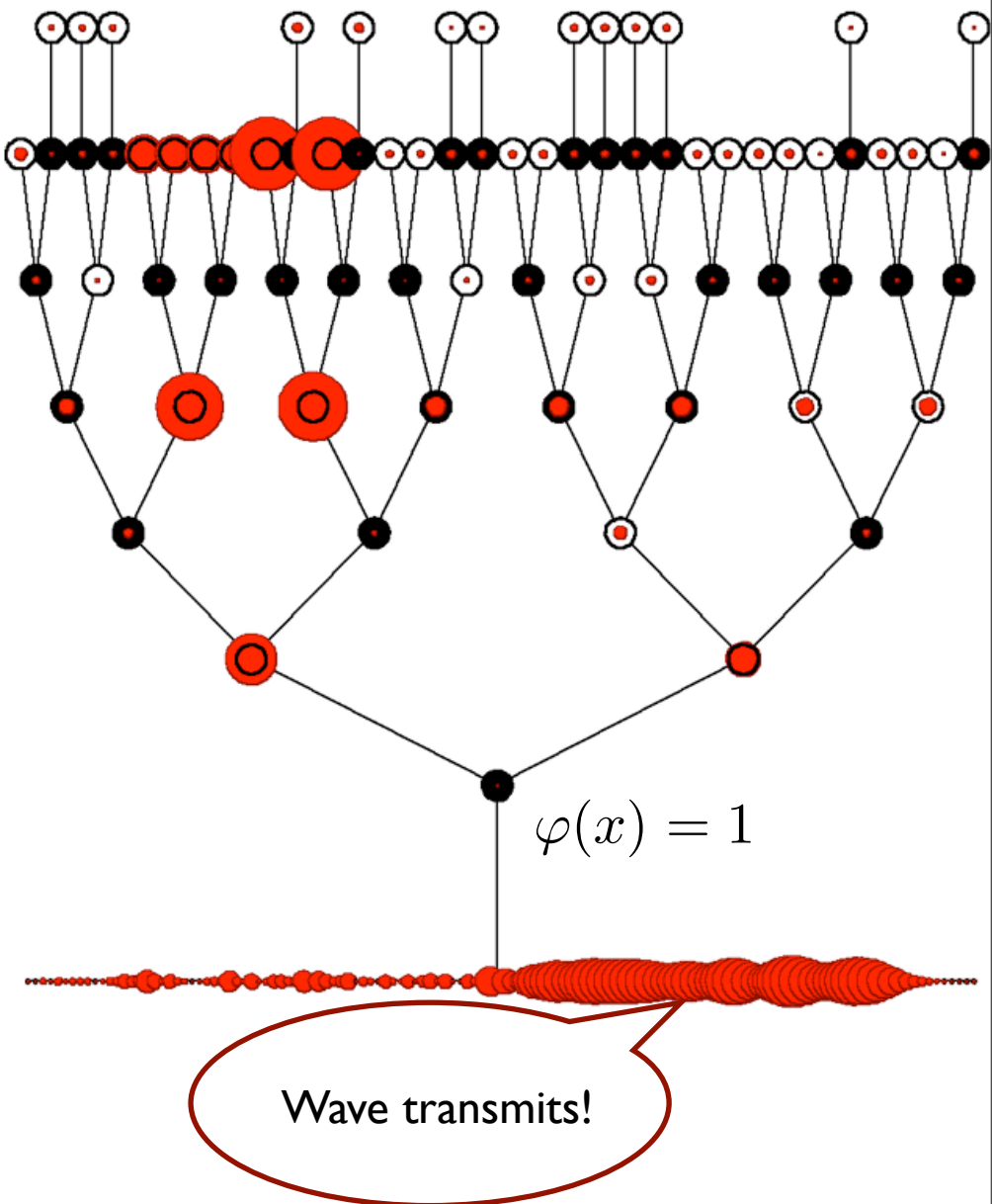
FGG quantum walk $|\psi_t\rangle = e^{iA_G t}|\psi_0\rangle$





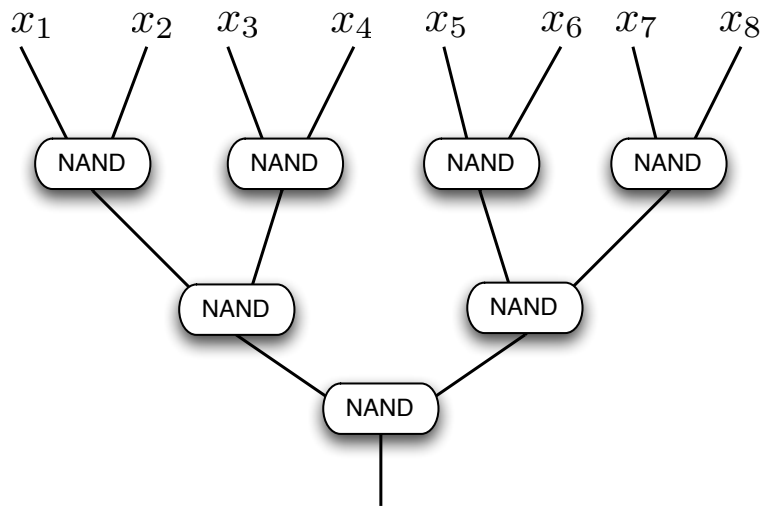
FGG quantum walk $|\psi_t\rangle = e^{iA_G t}|\psi_0\rangle$





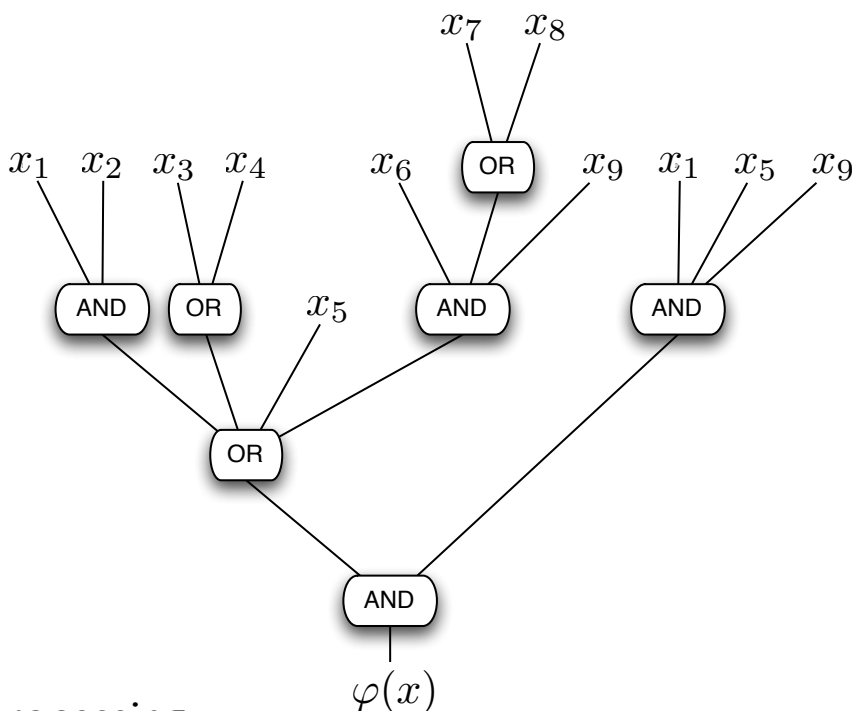
[FGG '07] algorithm

- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{1/2+o(1)}$.



[CRŠZ '07] algorithm

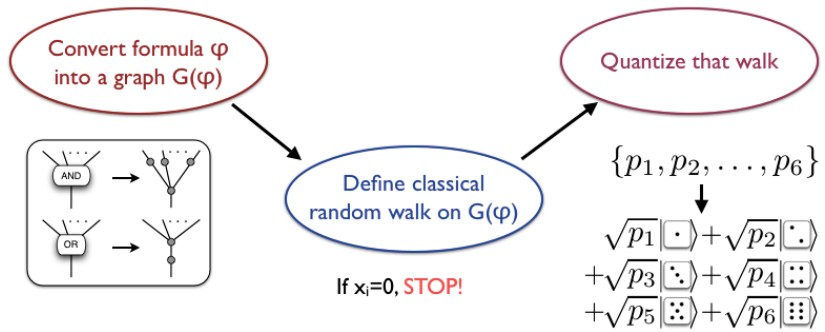
- **Theorem** ([CRŠZ '07]):
  - An "approximately balanced" {AND, OR, NOT} formula can be evaluated with $O(\sqrt{N})$ queries (optimal!).
  - A general {AND, OR, NOT} formula can be evaluated with $N^{1/2+o(1)}$ queries.



Running time is $N^{1/2+o(1)}$ in each case, after efficient preprocessing.

Talk outline

1. The Algorithm



2. Why It Works

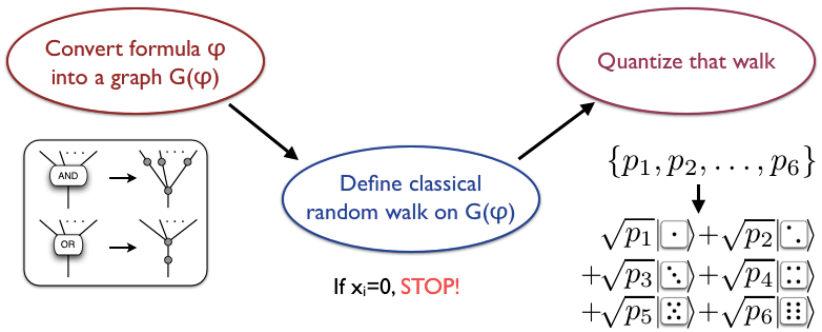
Szegedy correspondence

Zero energy eigenstate analysis

3. Extensions

Talk outline

1. The Algorithm



2. Why It Works

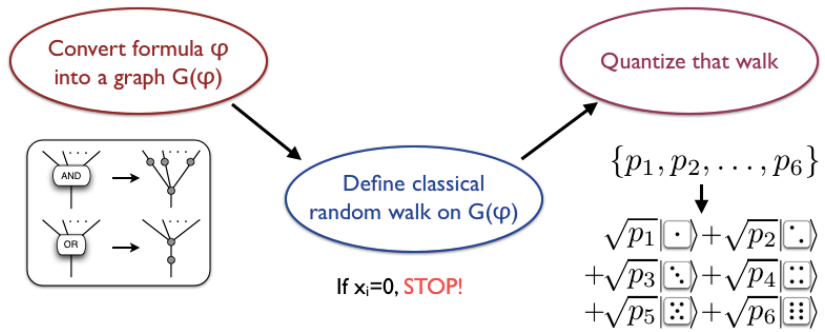
Szegedy correspondence

Zero energy eigenstate analysis

3. Extensions

Talk outline

1. The Algorithm



2. Why It Works

Szegedy correspondence

Zero energy eigenstate analysis

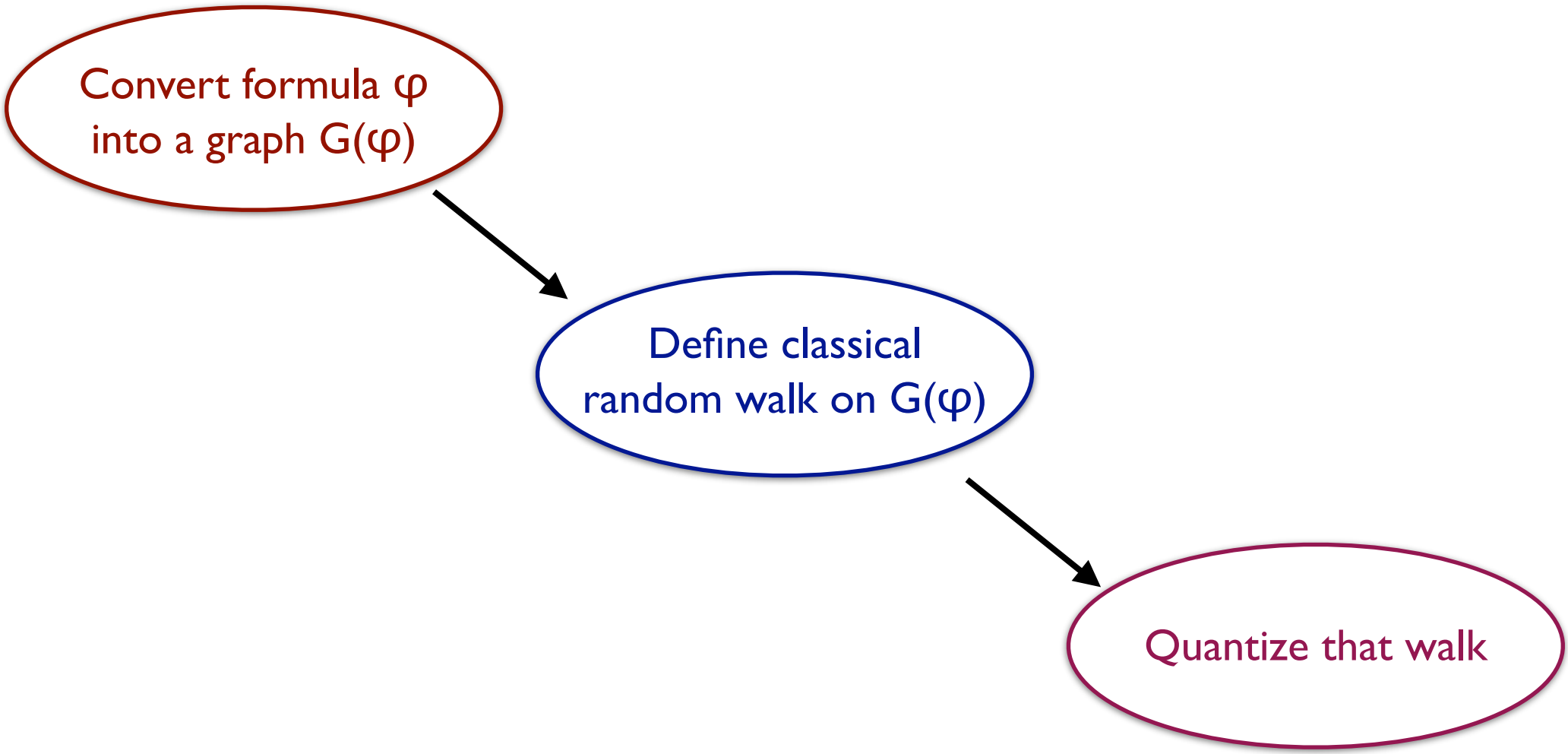
More Gates!

Fushimi-Inari

Chion-In

Katsura Imperial Villa

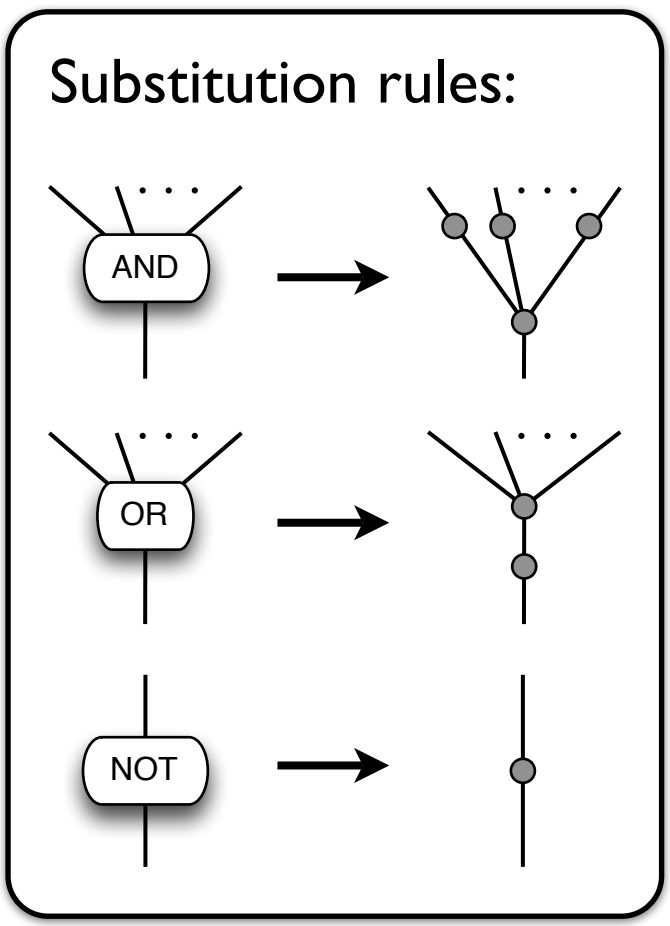
Formula evaluation algorithm

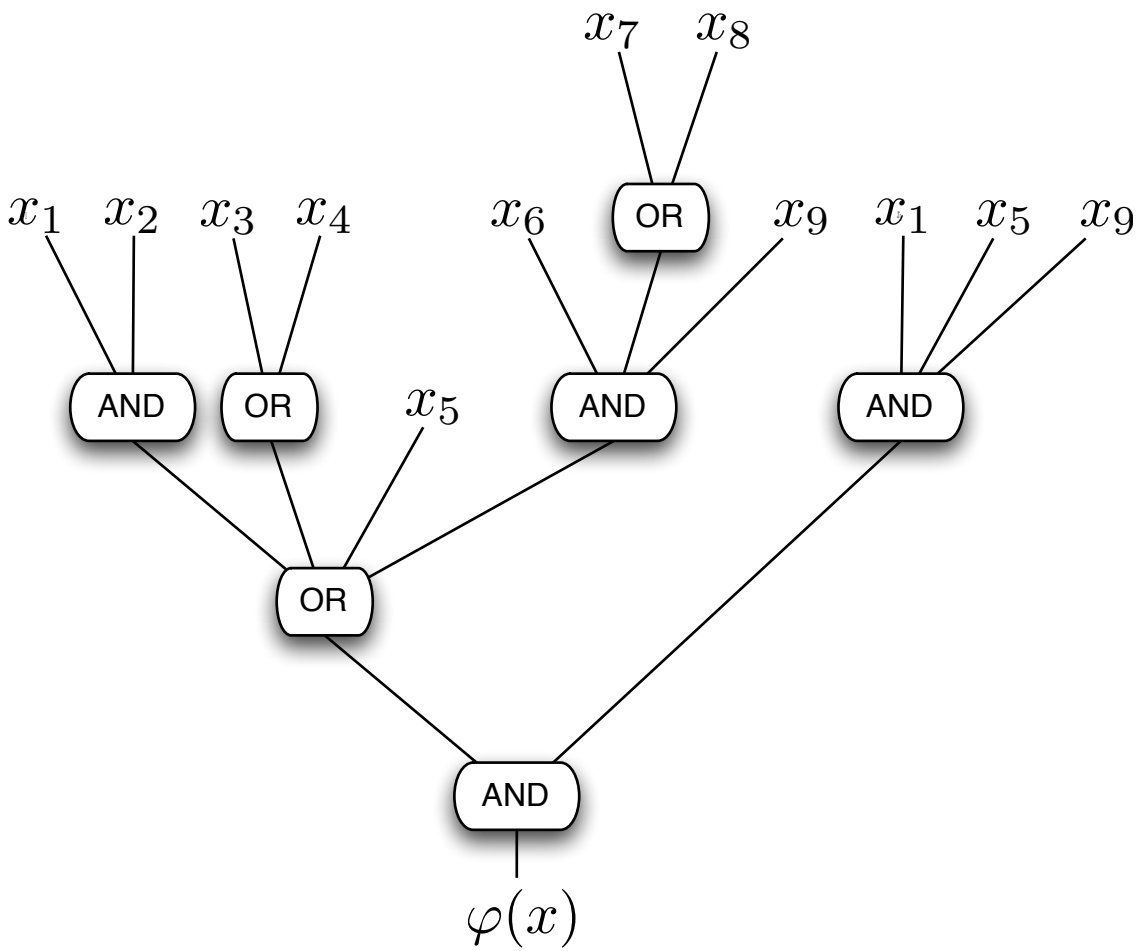


Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk

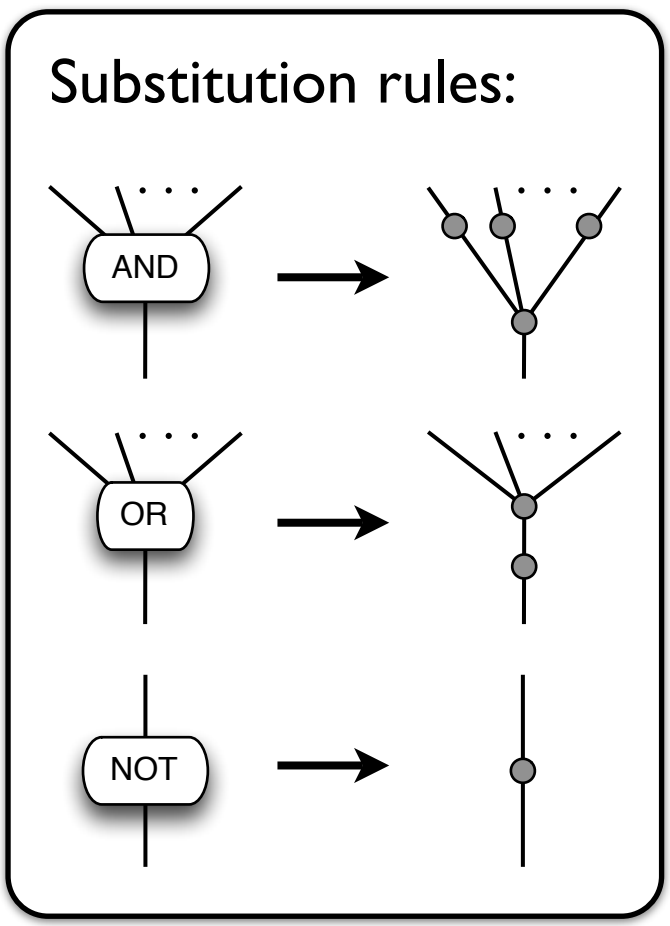


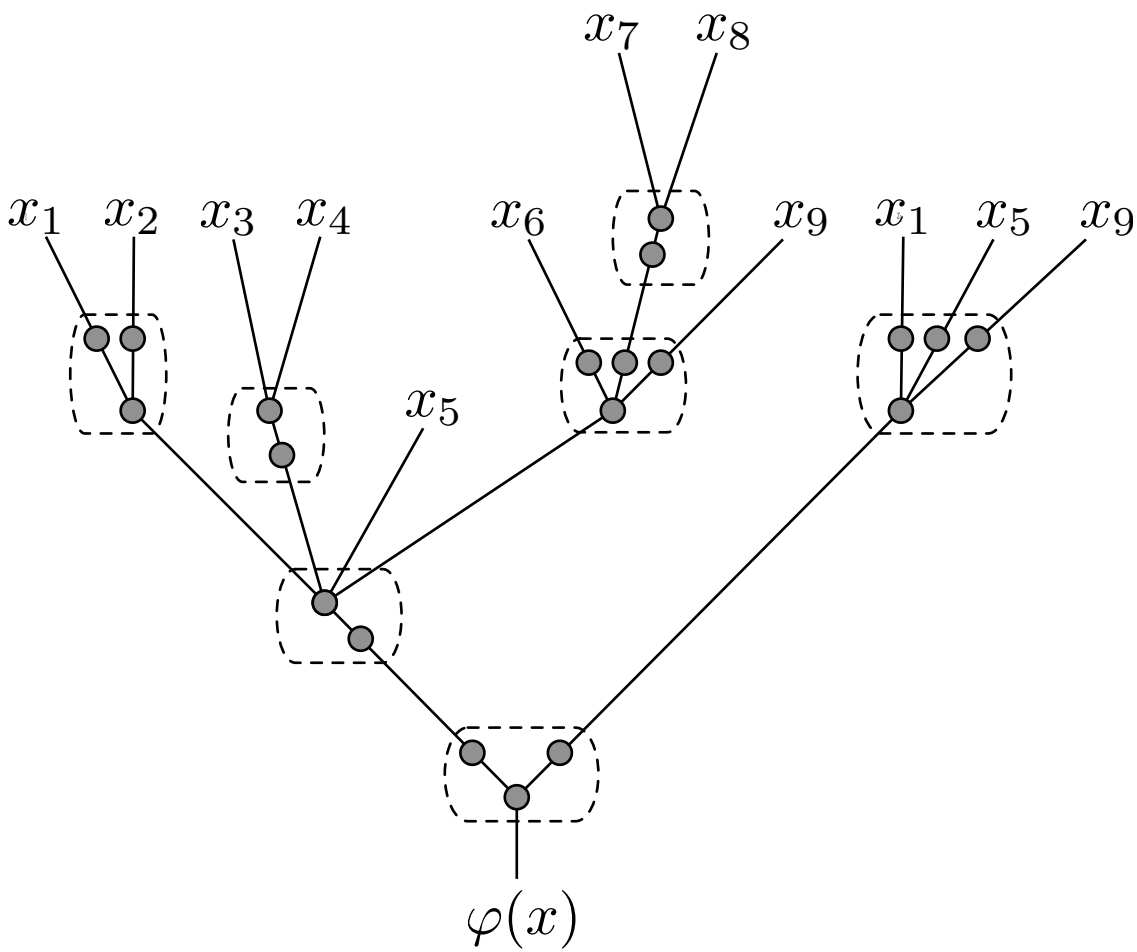


Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk

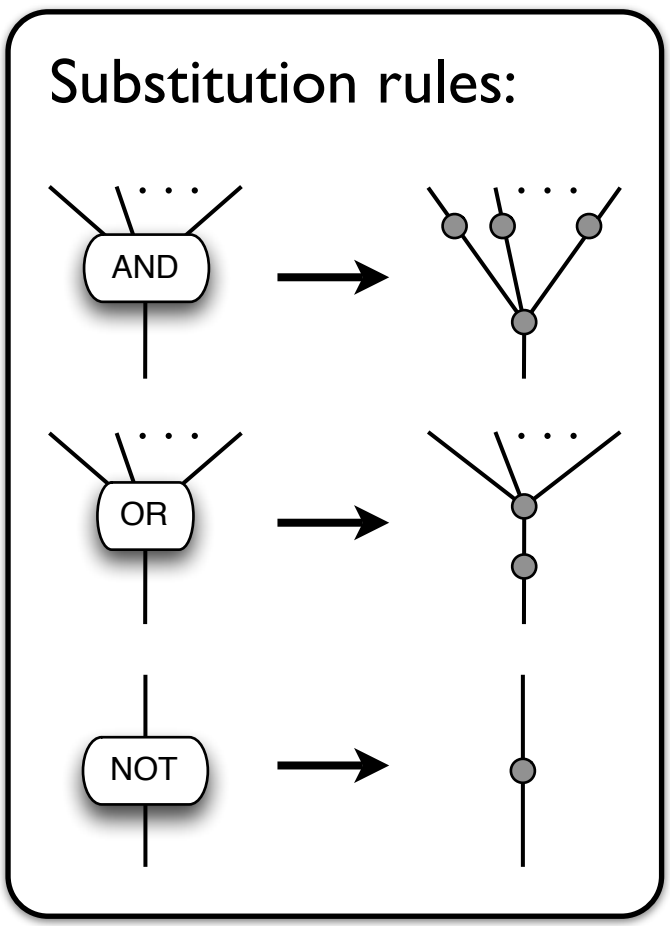


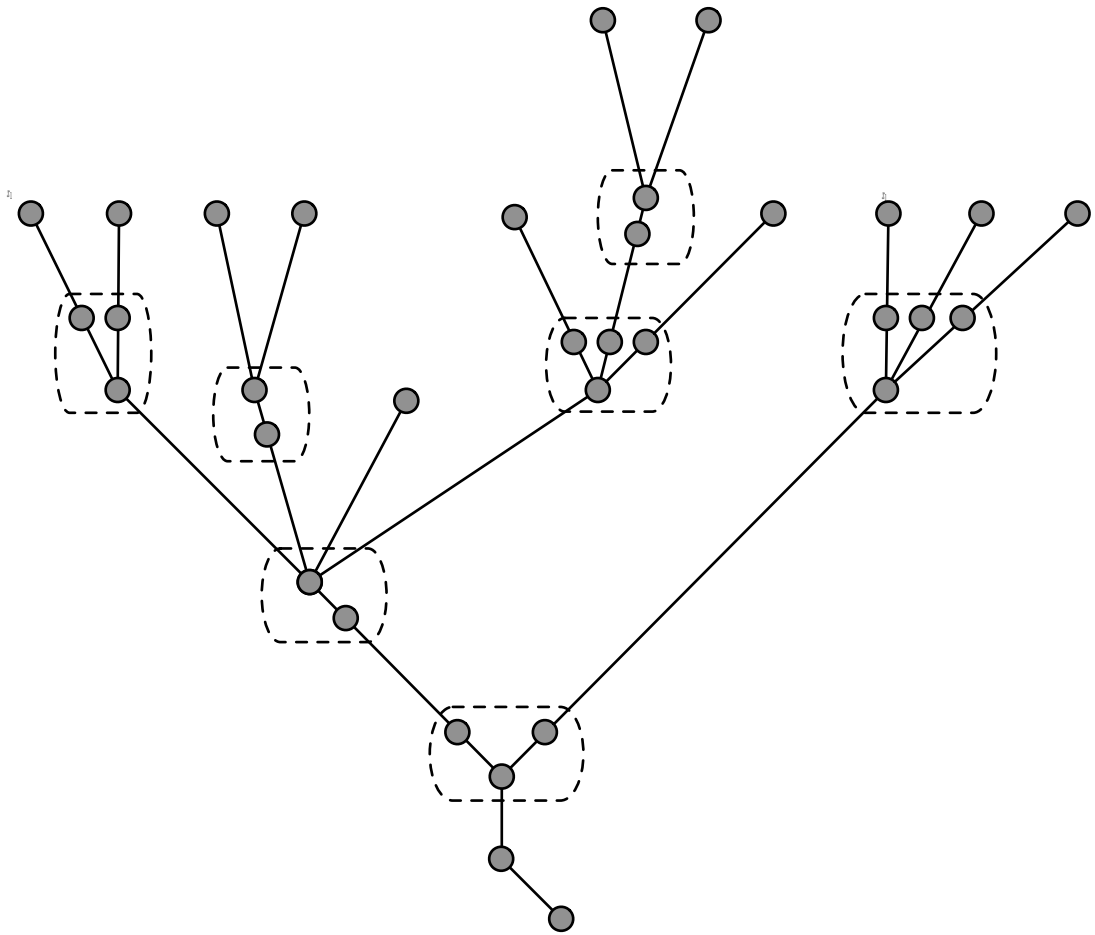


Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk

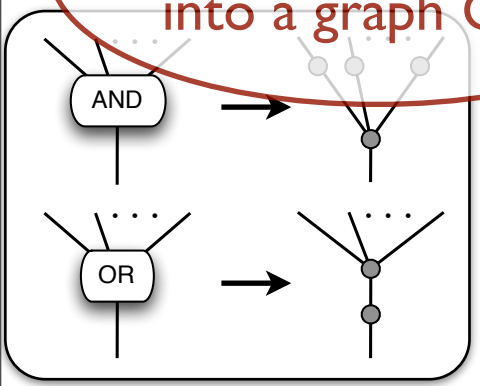




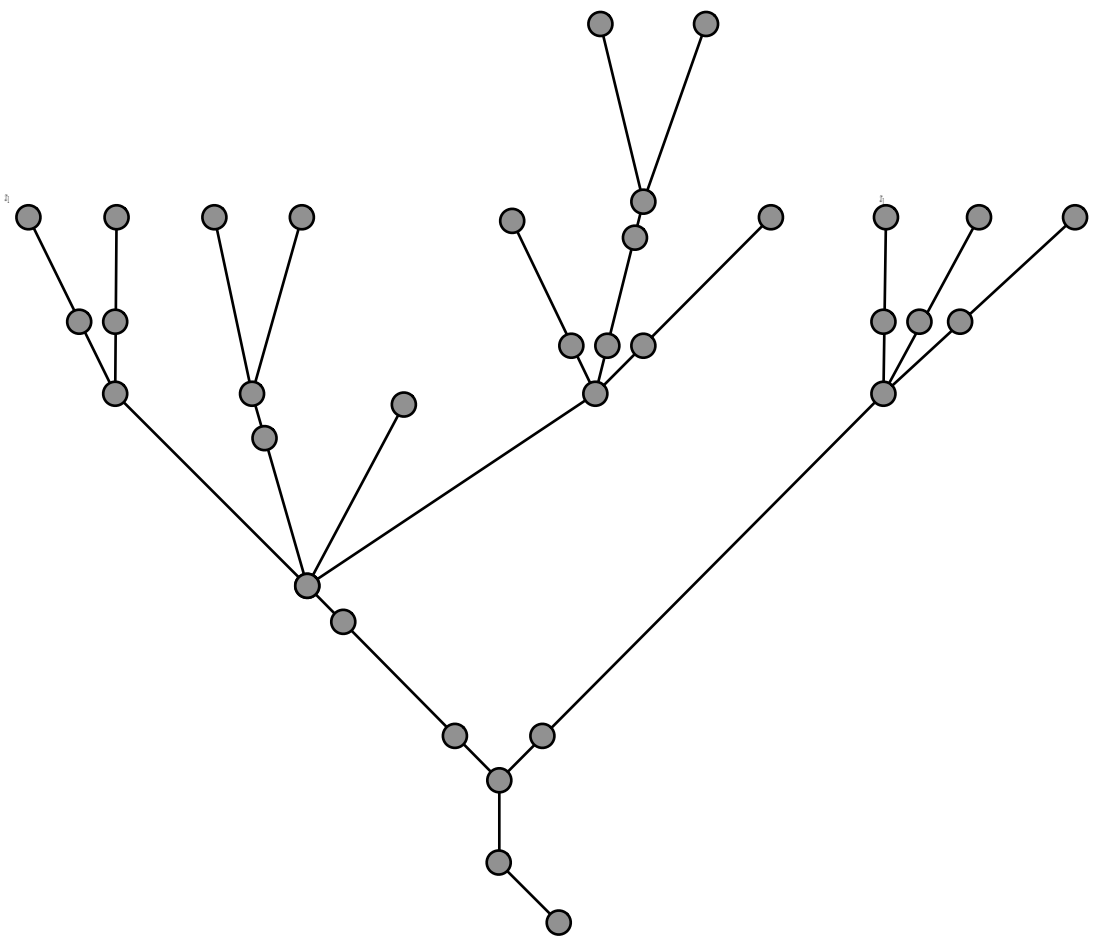
Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk



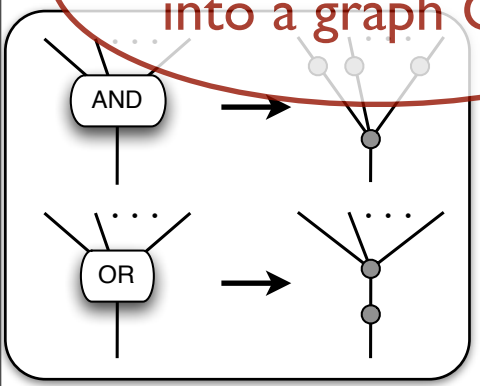
- P(stepping to subtree) ∝ √(size of that subtree)/√s_p
  - (For a balanced tree, walk is uniform)



Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk

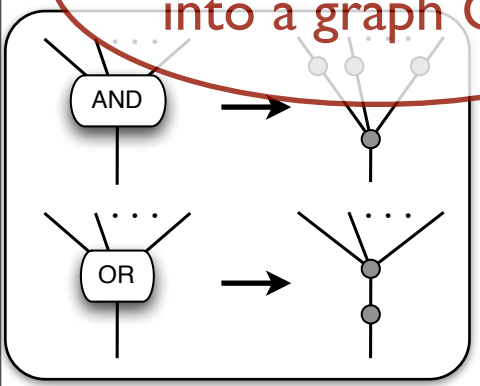


- P(stepping to subtree) ∝ √(size of that subtree)/√s_p
  - (For a balanced tree, walk is uniform)
- Make leaves (inputs) evaluating to 0 probability sinks

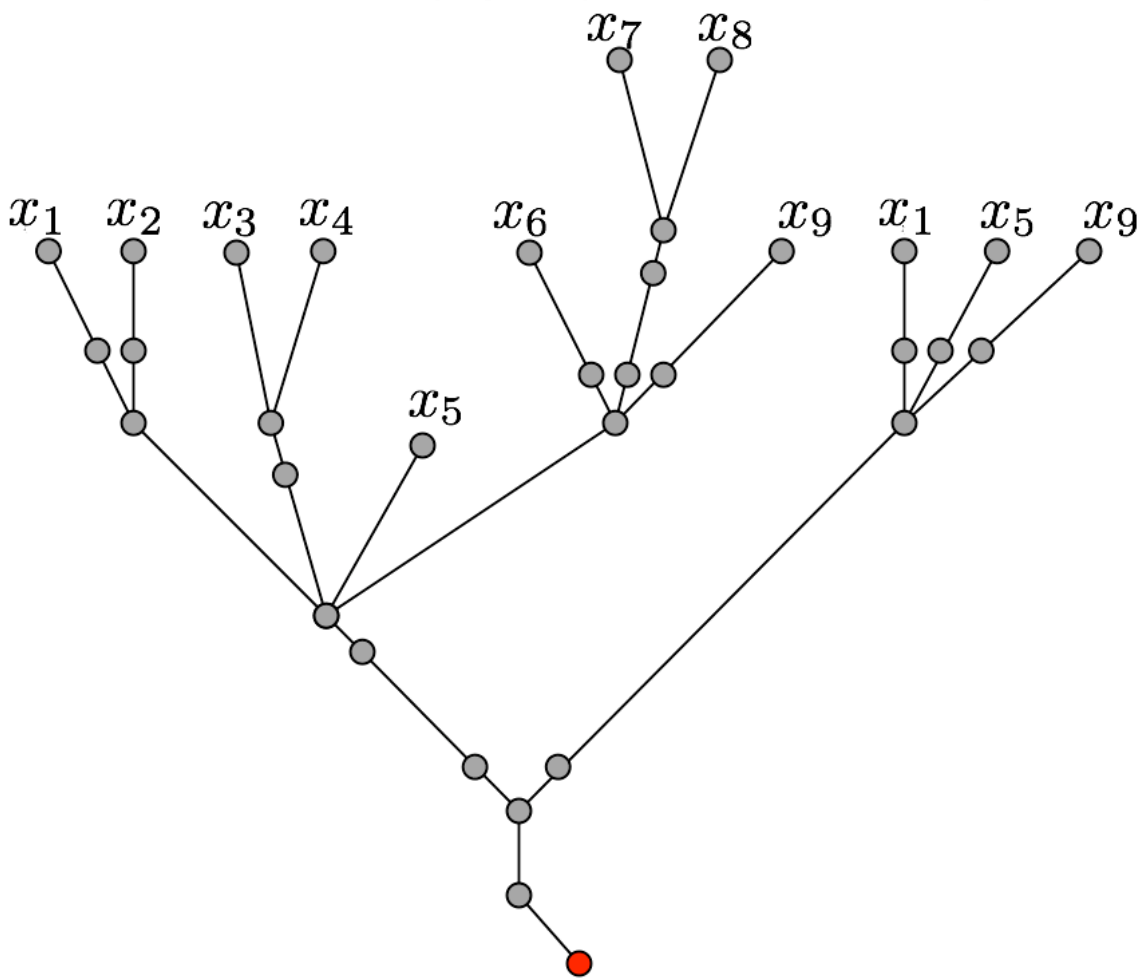
Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk



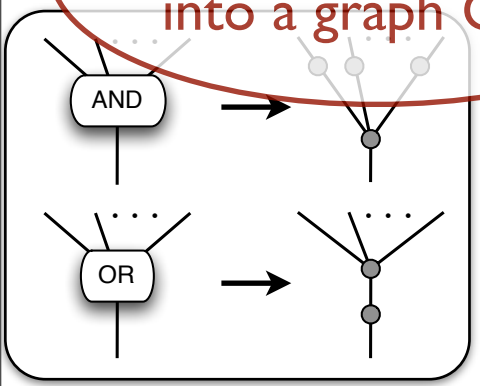
- P(stepping to subtree) ∝ √(size of that subtree)/√s_p
  - (For a balanced tree, walk is uniform)
- Make leaves (inputs) evaluating to 0 probability sinks



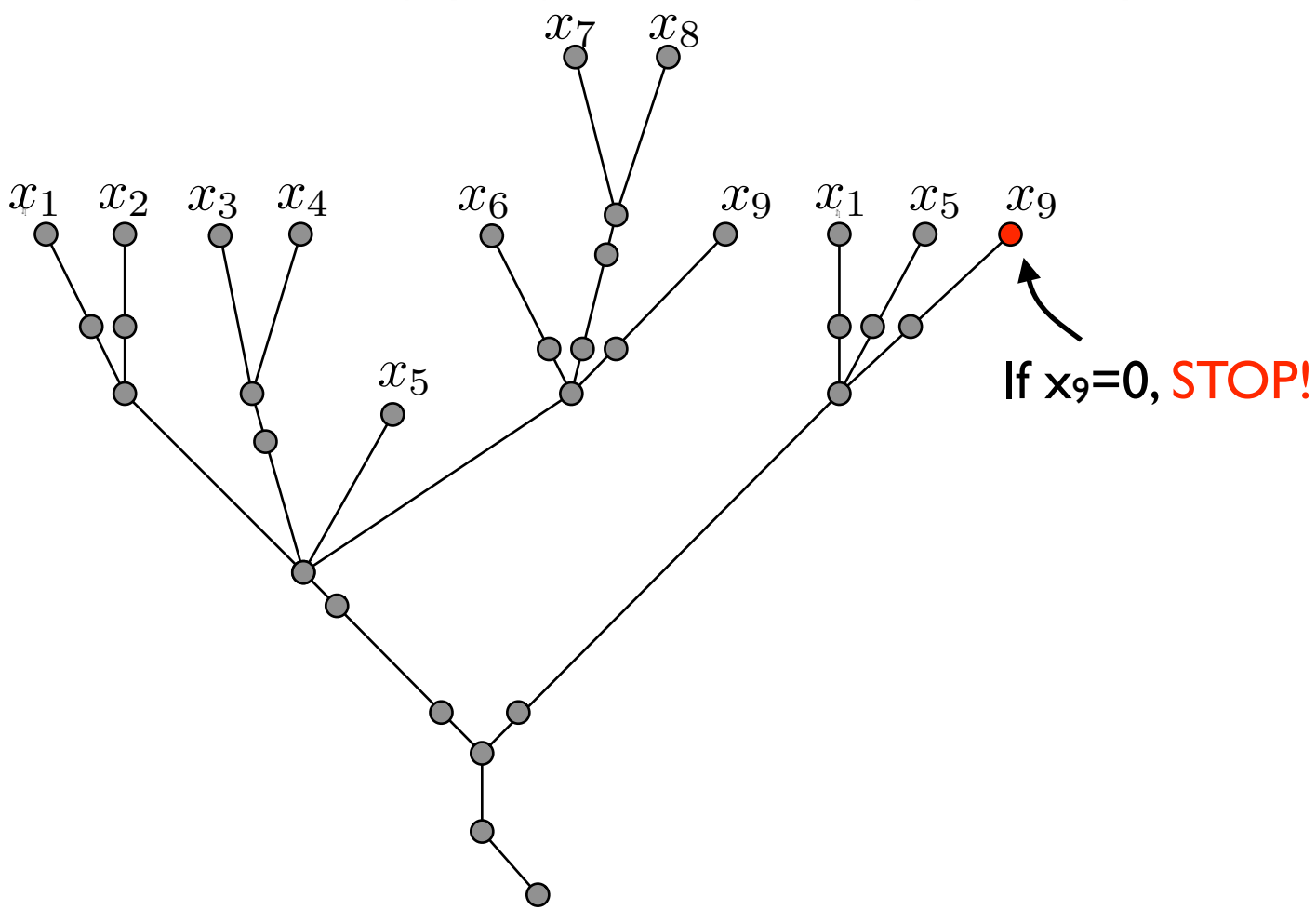
Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk



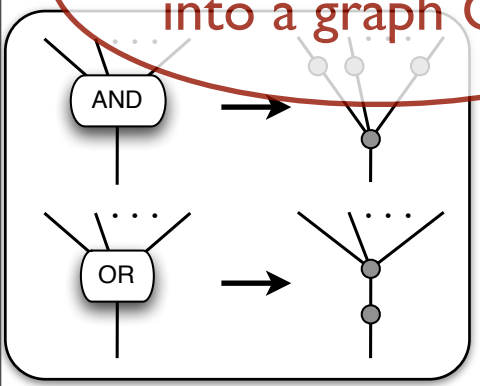
- P(stepping to subtree) ∝ √(size of that subtree)/√s_p
  - (For a balanced tree, walk is uniform)
- Make leaves (inputs) evaluating to 0 probability sinks



Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk



If x_i=0, STOP!

- Classically, roll a dice to determine next step
- Quantumly, the dice is part of the quantum state. Instead of randomizing the dice between steps, apply a unitary operator to it.

Transition probabilities

$$\{p_1, p_2, \ldots, p_6\}$$

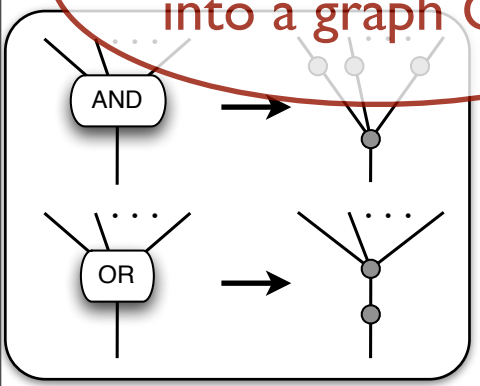
U = reflection about the state

$$\sqrt{p_1}\left|\boxdot\right\rangle + \sqrt{p_2}\left|\boxdot\right\rangle + \sqrt{p_3}\left|\boxdot\right\rangle + \sqrt{p_4}\left|\boxdot\right\rangle + \sqrt{p_5}\left|\boxdot\right\rangle + \sqrt{p_6}\left|\boxdot\right\rangle$$

Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

Quantize that walk



If x_i=0, STOP!

- Classically, roll a dice to determine next step
- Quantumly, the dice is part of the quantum state. Instead of randomizing the dice between steps, apply a unitary operator to it.
  - Probability sinks in the classical r.w. (inputs x_i=0) become phase flips in the qu. walk ⇒ standard phase flip oracle

Transition probabilities

$$\{p_1, p_2, \ldots, p_6\}$$

U = reflection about the state

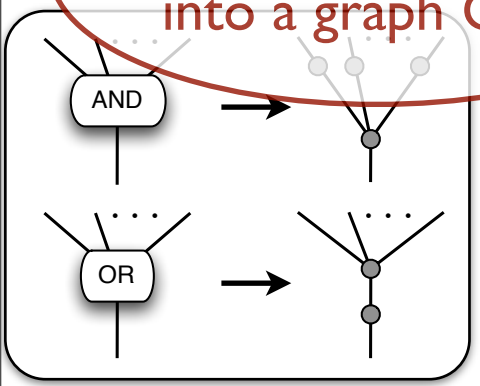
$$\sqrt{p_1}\left|\boxdot\right\rangle+\sqrt{p_2}\left|\boxdot\right\rangle$$
$$+\sqrt{p_3}\left|\boxdot\right\rangle+\sqrt{p_4}\left|\boxdot\right\rangle$$
$$+\sqrt{p_5}\left|\boxdot\right\rangle+\sqrt{p_6}\left|\boxdot\right\rangle$$

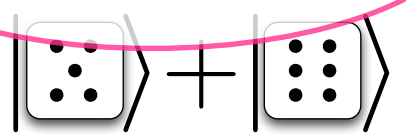
Convert formula φ into a graph G(φ)

Define classical random walk on G(φ)

If x_i=0, STOP!

Quantize that walk



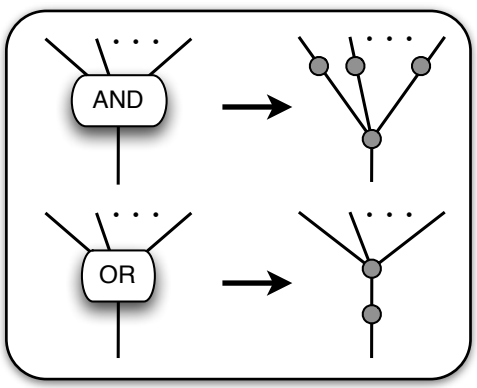


The Algorithm:

- Start at the root
- Apply phase estimation to the quantum walk with precision $1/\sqrt{N}$ (i.e., run the walk for time $\sqrt{N}$)
  - If phase is 0 or π, output "φ(x)=1"
  - Otherwise output "φ(x)=0"

Formula evaluation algorithm

Convert formula φ into a graph G(φ)



Define classical random walk on G(φ)

P(stepping to subtree)

∝ √(size of that subtree)/√s_p

If x_i=0, STOP!

Quantize that walk

$$\{p_1, p_2, \ldots, p_6\}$$

$$\sqrt{p_1}\left|\cdot\right\rangle + \sqrt{p_2}\left|\cdot\cdot\right\rangle + \sqrt{p_3}\left|\cdot\cdot\right\rangle + \sqrt{p_4}\left|\cdot\cdot\right\rangle + \sqrt{p_5}\left|\cdot\cdot\right\rangle + \sqrt{p_6}\left|\cdot\cdot\right\rangle$$

2. Why It Works

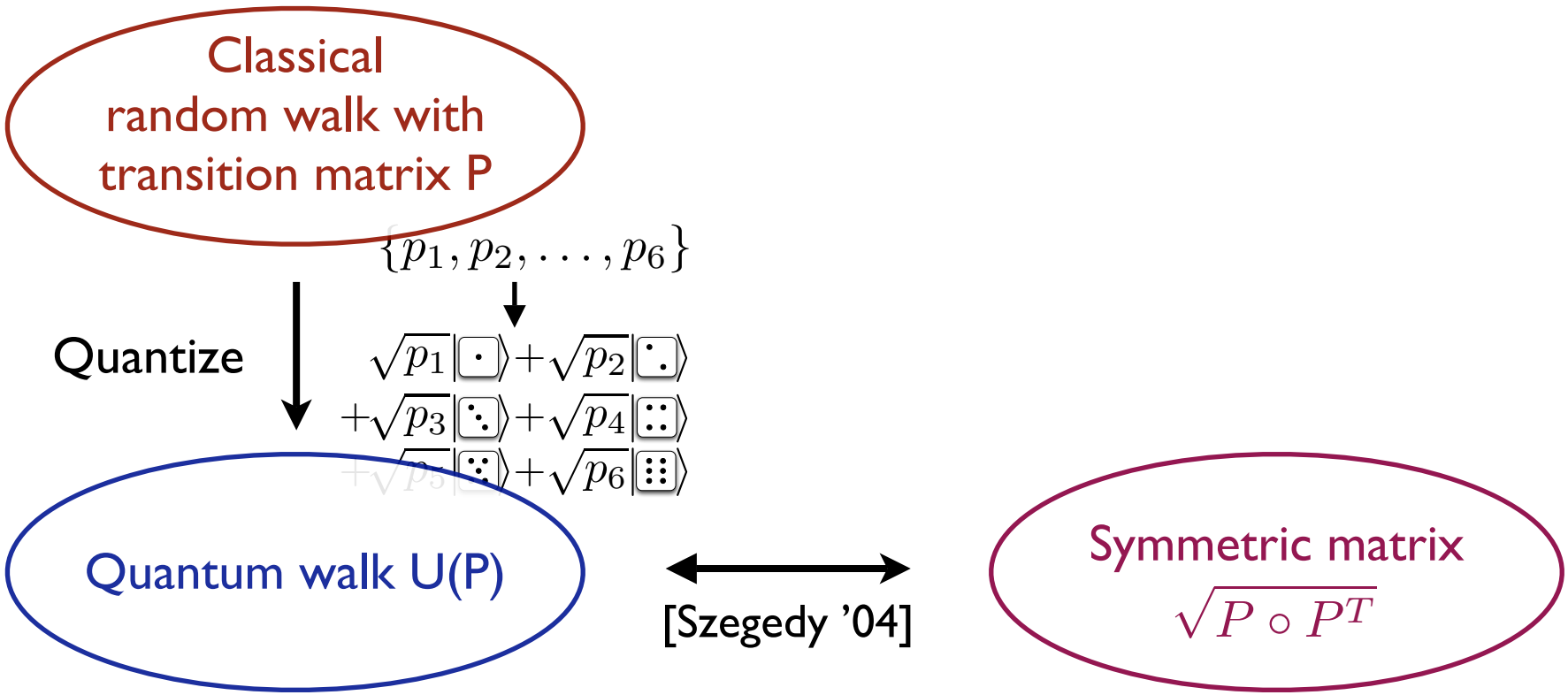
The Algorithm:

- Start at the root
- Apply phase estimation to the quantum walk with precision $1/\sqrt{N}$ (i.e., run the walk for time $\sqrt{N}$)
  - If phase is 0 or π, output "$\varphi(x)=1$"
  - Otherwise output "$\varphi(x)=0$"

- Outputs $\varphi(x)=1$ "iff" there is an eigenstate of the walk operator U that overlaps the root and has corresponding $|\text{eigenvalue}| < 1/\sqrt{N}$

$\therefore$ We need to carry out spectral analysis of the quantum walk U

Spectral analysis 1: Szegedy correspondence

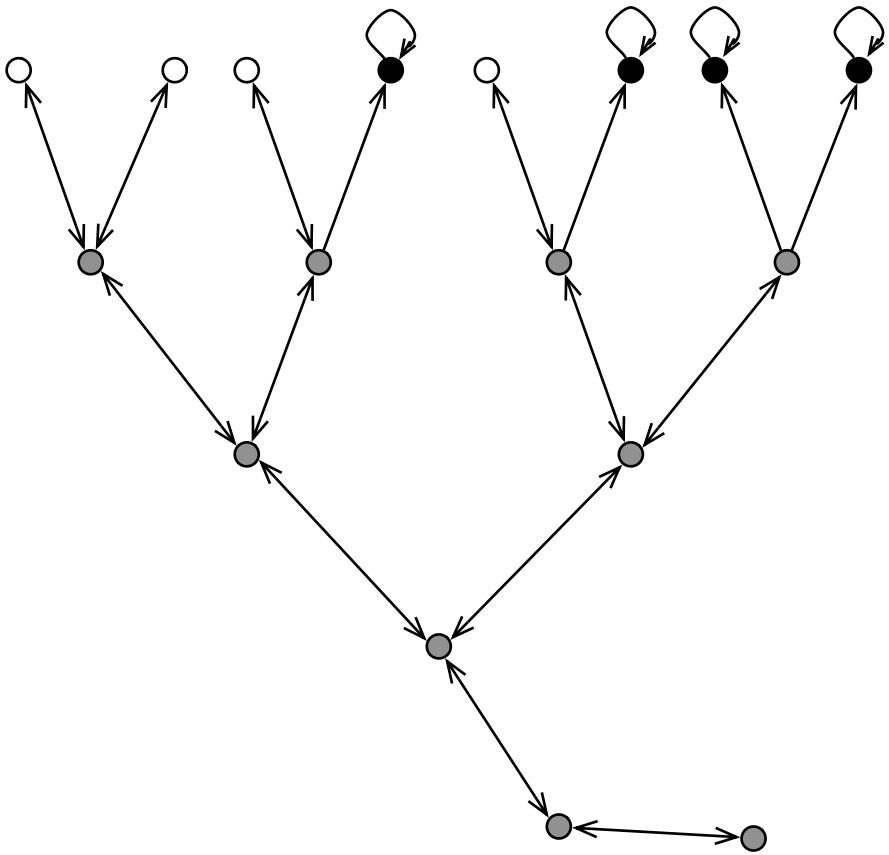


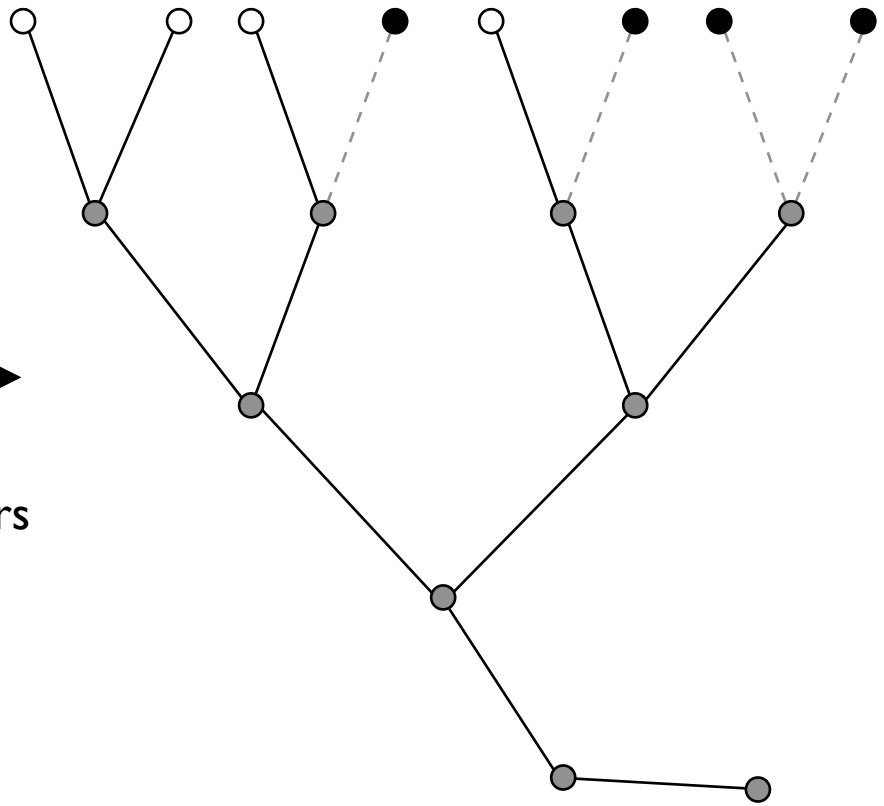
- Correspondence between spectrum and eigenvalues of quantum walk and those of a symmetric matrix
  (If P is symmetric, then P and U(P) have corresponding eigensystems)
  - Halves the dimensions
  - Real instead of complex operators

Spectral analysis 1: Szegedy correspondence

Quantum coined walk U on:

$\sqrt{P \circ P^T}$ = Weighted Adj. matrix of:





eigenvalues
& eigenvectors

○ = 1

● = 0

2|E| dimensions

|V| dimensions

The Algorithm:

- Start at the root
- Apply phase estimation to the quantum walk with precision $1/\sqrt{N}$ (i.e., run the walk for time $\sqrt{N}$)
  - If phase is 0 or π, output "φ(x)=1"
  - Otherwise output "φ(x)=0"

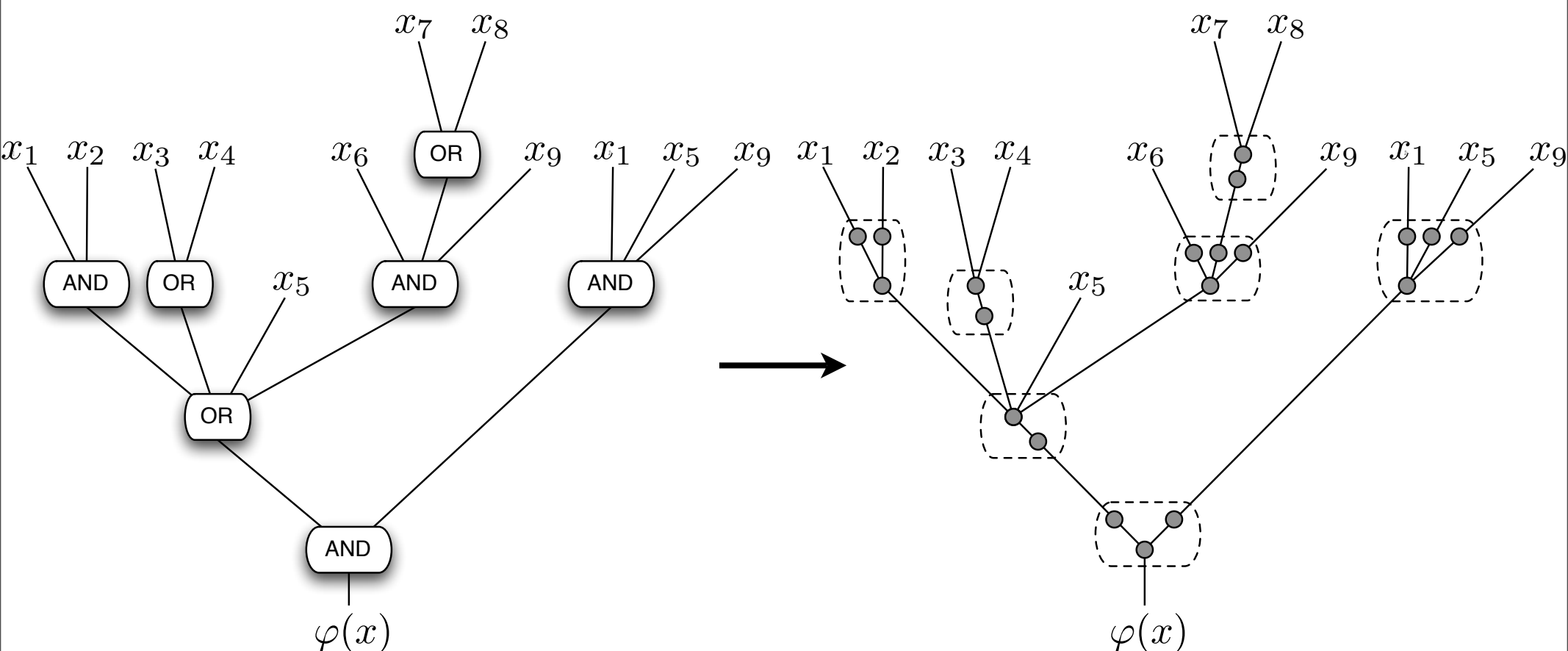
- Outputs φ(x)=1 "iff" there is an eigenstate of $A_G = \sqrt{P \circ P^T}$ that overlaps the root and has corresponding |eigenvalue| < $1/\sqrt{N}$

∴ We need to carry out spectral analysis of A_G

- **Main Theorem:**
  - Adjacency matrix A_G has eigenvalue E=0 eigenvector with Ω(1) support on r" when φ(x)=1.
  - A_G has no eigenvalues $E \in (-1/\sqrt{N}, 1/\sqrt{N})$ with support on r" when φ(x)=0.

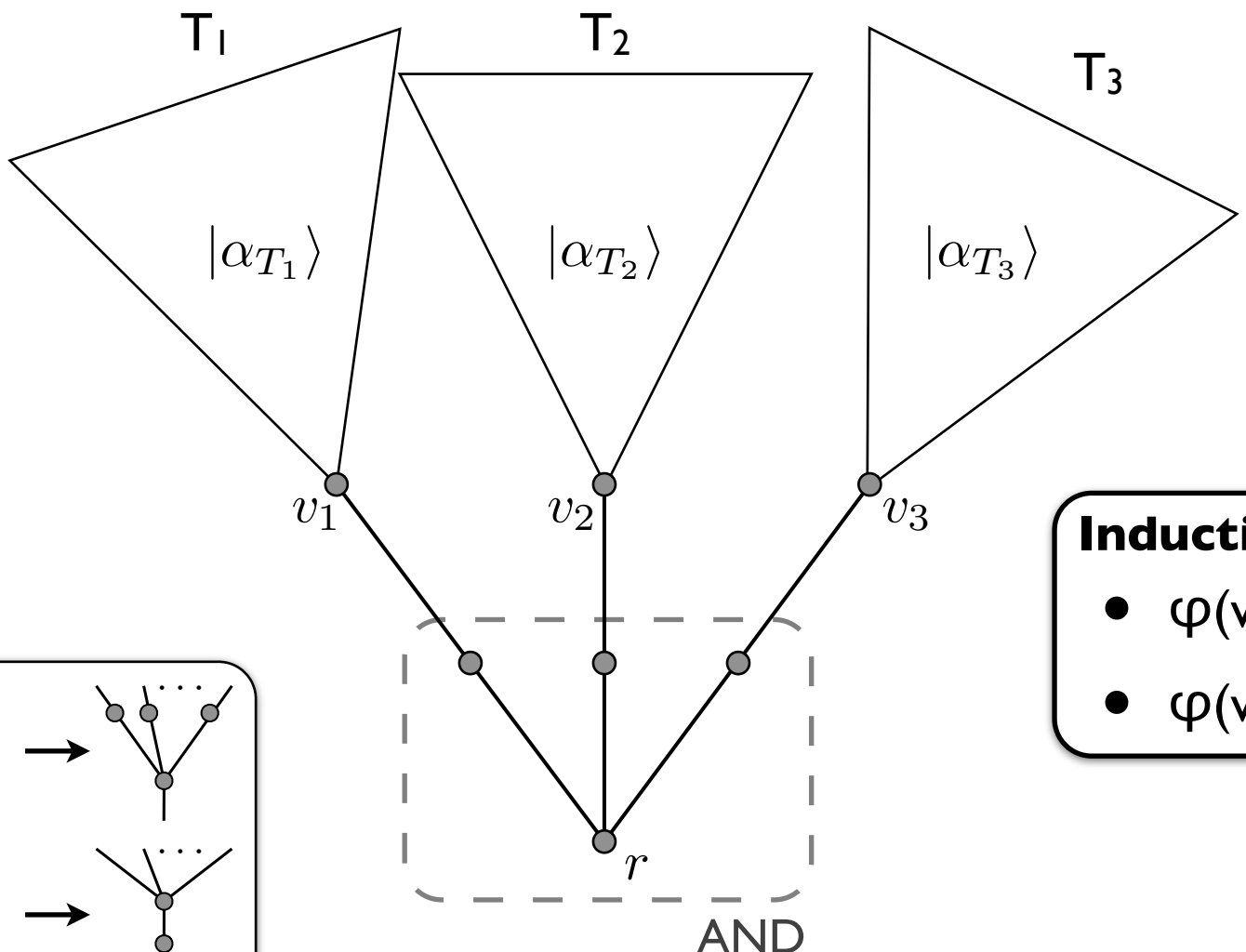
- **Theorem:** φ(x)=1 $\Leftrightarrow$ ∃ an E=0 eigenstate of A_G supported on root r.

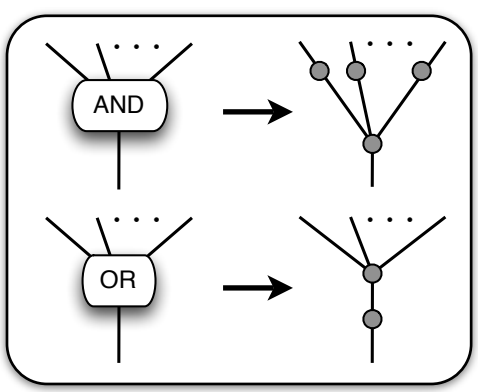
Proof



- **Theorem:** $\varphi(x)=1 \iff \exists$ an E=0 eigenstate of A_G supported on root r.

Construct eigenstate $|\alpha\rangle = \sum_v \alpha_v |v\rangle$ by induction

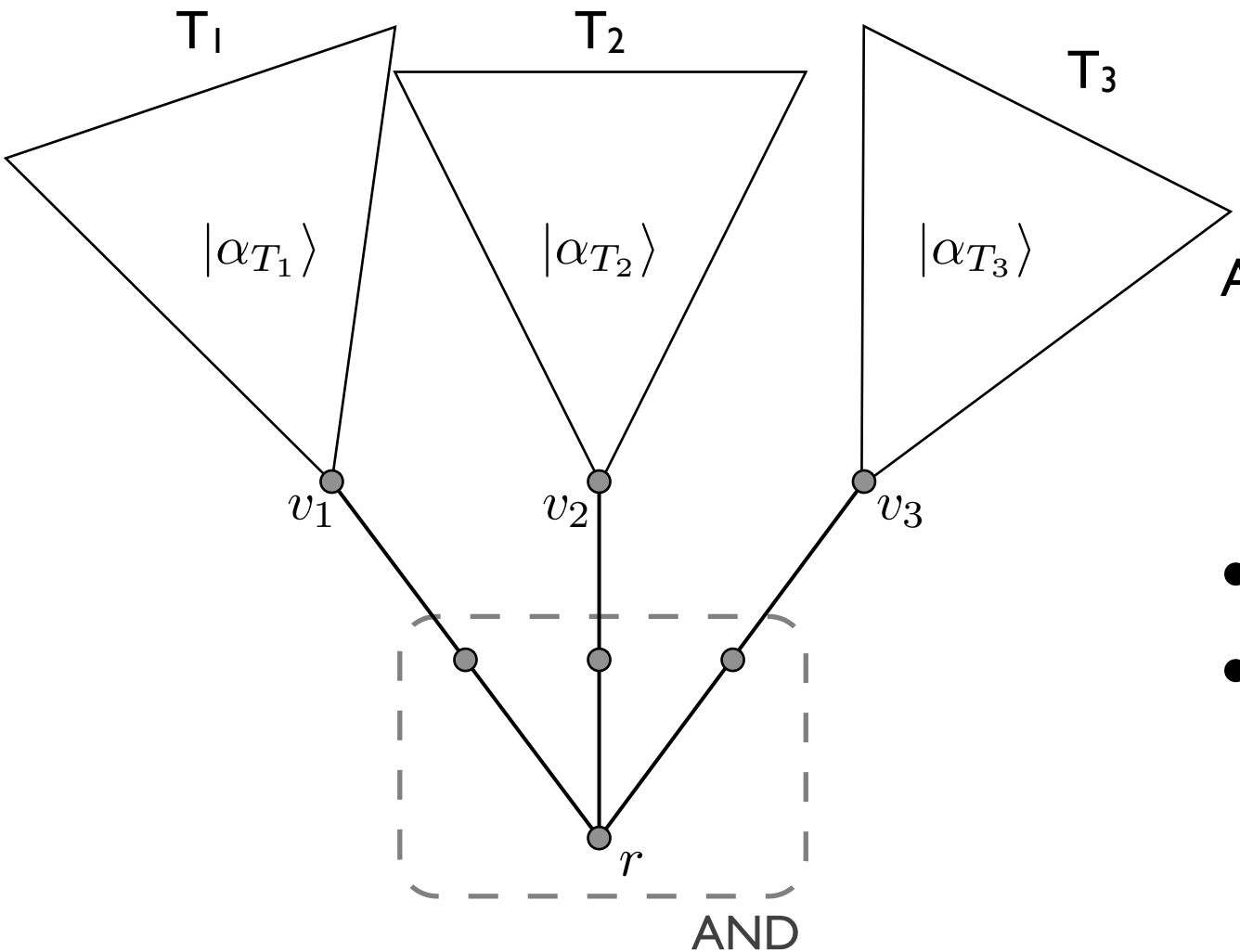




Inductive Hypothesis:

- $\varphi(v)=0 \Rightarrow \alpha_v=0$
- $\varphi(v)=1 \Rightarrow \alpha_v$ can be $\neq 0$

- **Inductive Hypothesis:**
  - $\varphi(v)=0 \Rightarrow \alpha_v=0$
  - $\varphi(v)=1 \Rightarrow \alpha_v$ can be $\neq 0$



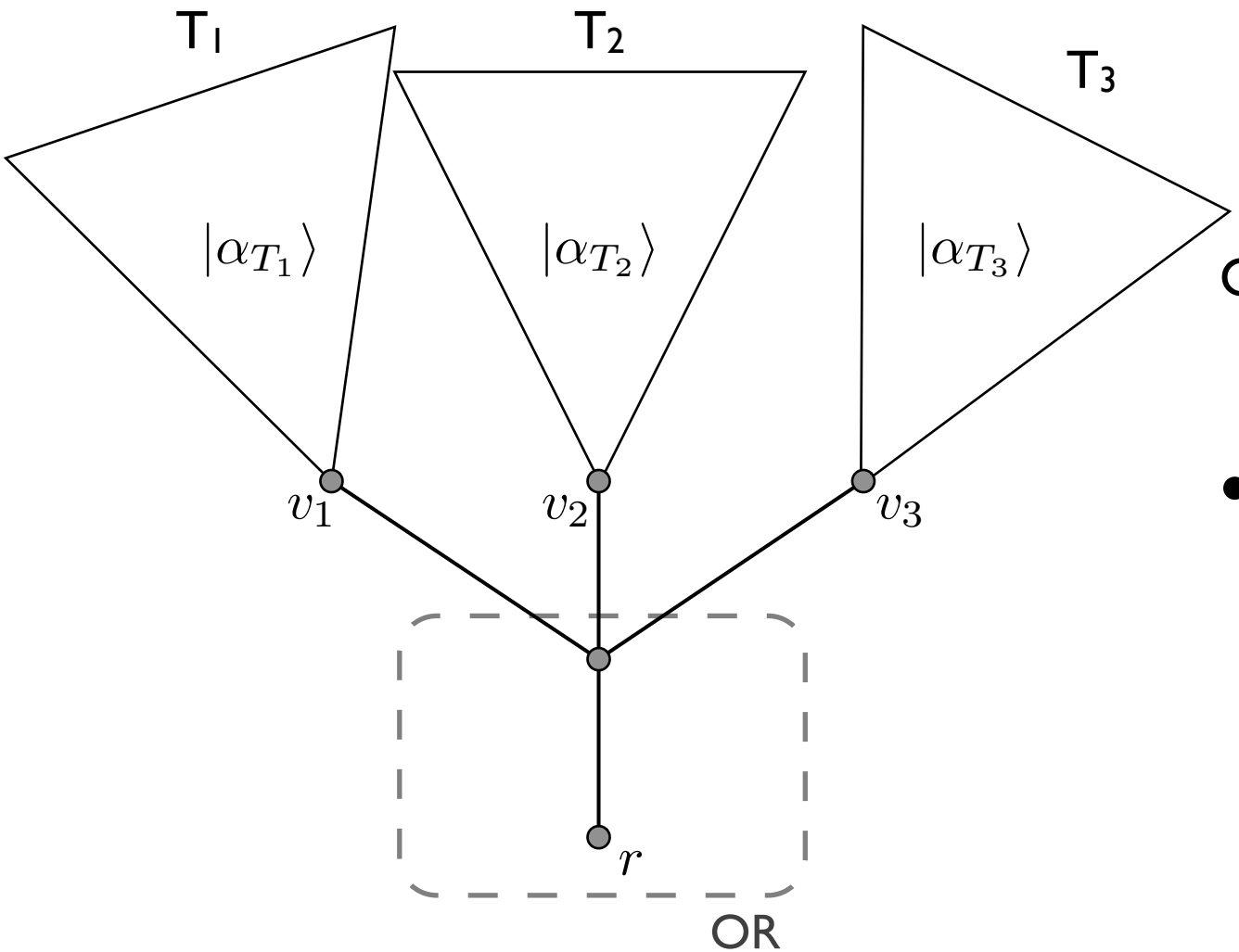
AND gate gadget constraints:

$$\alpha_{v_1} + \alpha_r = 0$$
$$\alpha_{v_2} + \alpha_r = 0$$
$$\alpha_{v_3} + \alpha_r = 0$$

- If any $\varphi(v_i)=0$, $\alpha_{v_i}=0 \Rightarrow \alpha_r=0$
- If all $\varphi(v_i)=1$, can scale each $|\alpha_{T_i}\rangle$ so $\alpha_{v_1}=\alpha_{v_2}=\alpha_{v_3}\neq 0$, then set $\alpha_r=-\alpha_{v_i}\neq 0$

✓ AND

- **Inductive Hypothesis:**
  - $\varphi(v)=0 \Rightarrow \alpha_v=0$
  - $\varphi(v)=1 \Rightarrow \alpha_v$ can be $\neq 0$



OR gate gadget constraint:

$$\alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3} + \alpha_r = 0$$

- α_r can be $\neq 0 \Leftrightarrow$ at least one $\alpha_{v_i} \neq 0 \Leftrightarrow$ at least one $\varphi(v_i)=1$

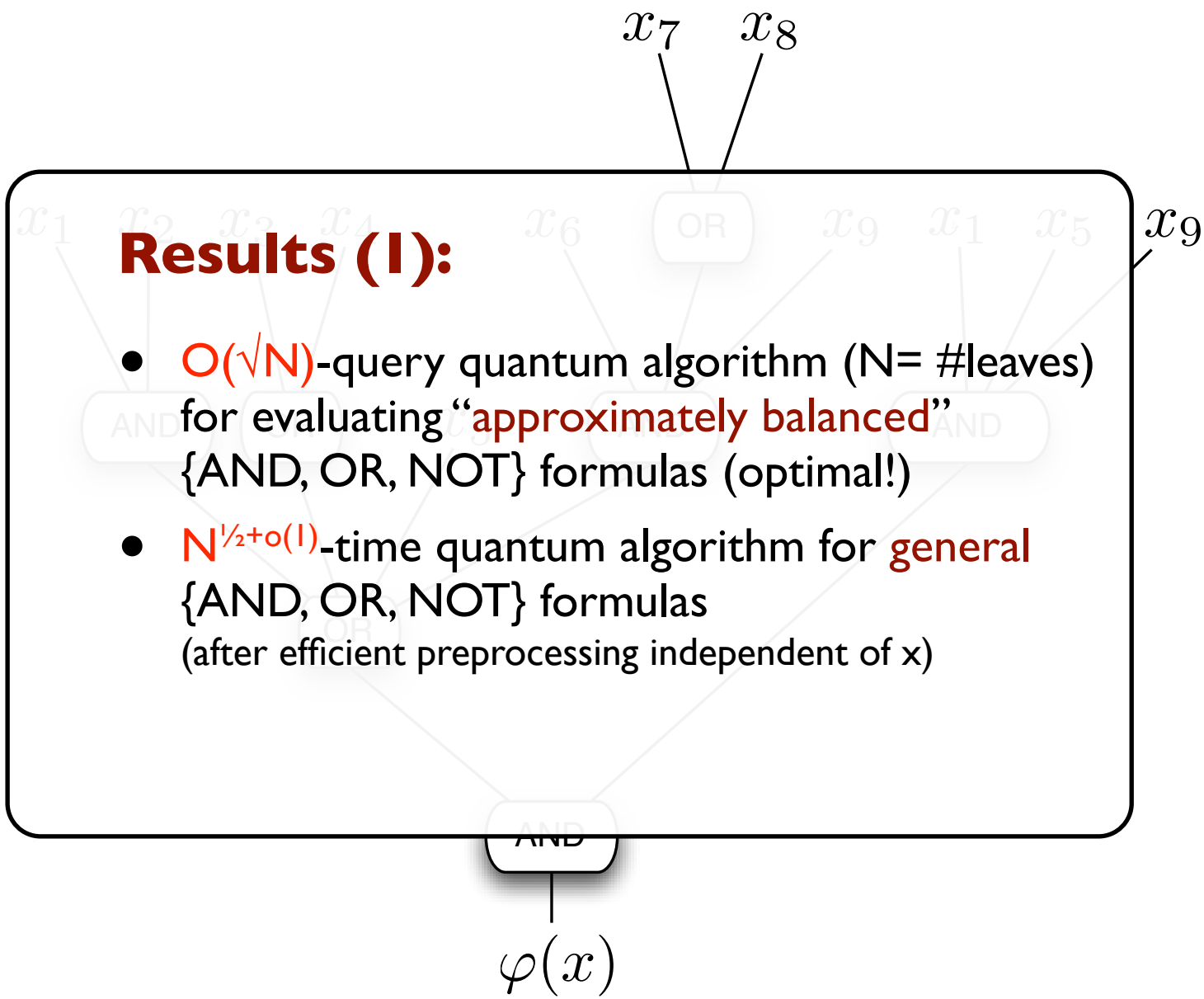
✓ OR

- **Theorem:** $\varphi(x)=1 \iff \exists$ an $E=0$ eigenstate of A_G supported on root r. ✓

- **Main Theorem:**
  - Adjacency matrix A_G has eigenvalue $E=0$ eigenvector with $\Omega(1)$ support on r" when $\varphi(x)=1$.
  - A_G has no eigenvalues $E \in (-1/\sqrt{N}, 1/\sqrt{N})$ with support on r" when $\varphi(x)=0$.

- Remains to show support α_r is large ($\Omega(1)$) when $\varphi(r)=0$, and that there is a large spectral gap ($1/\sqrt{N}$) away from $E=0$ when $\varphi(r)=1$.
- Proofs by same induction but quantitative.

Problem: Evaluate the formula with minimal queries to the input bits x_i.



Results (1):

- $O(\sqrt{N})$-query quantum algorithm (N= #leaves) for evaluating "approximately balanced" {AND, OR, NOT} formulas (optimal!)
- $N^{1/2+o(1)}$-time quantum algorithm for general {AND, OR, NOT} formulas
  (after efficient preprocessing independent of x)

3. Extensions

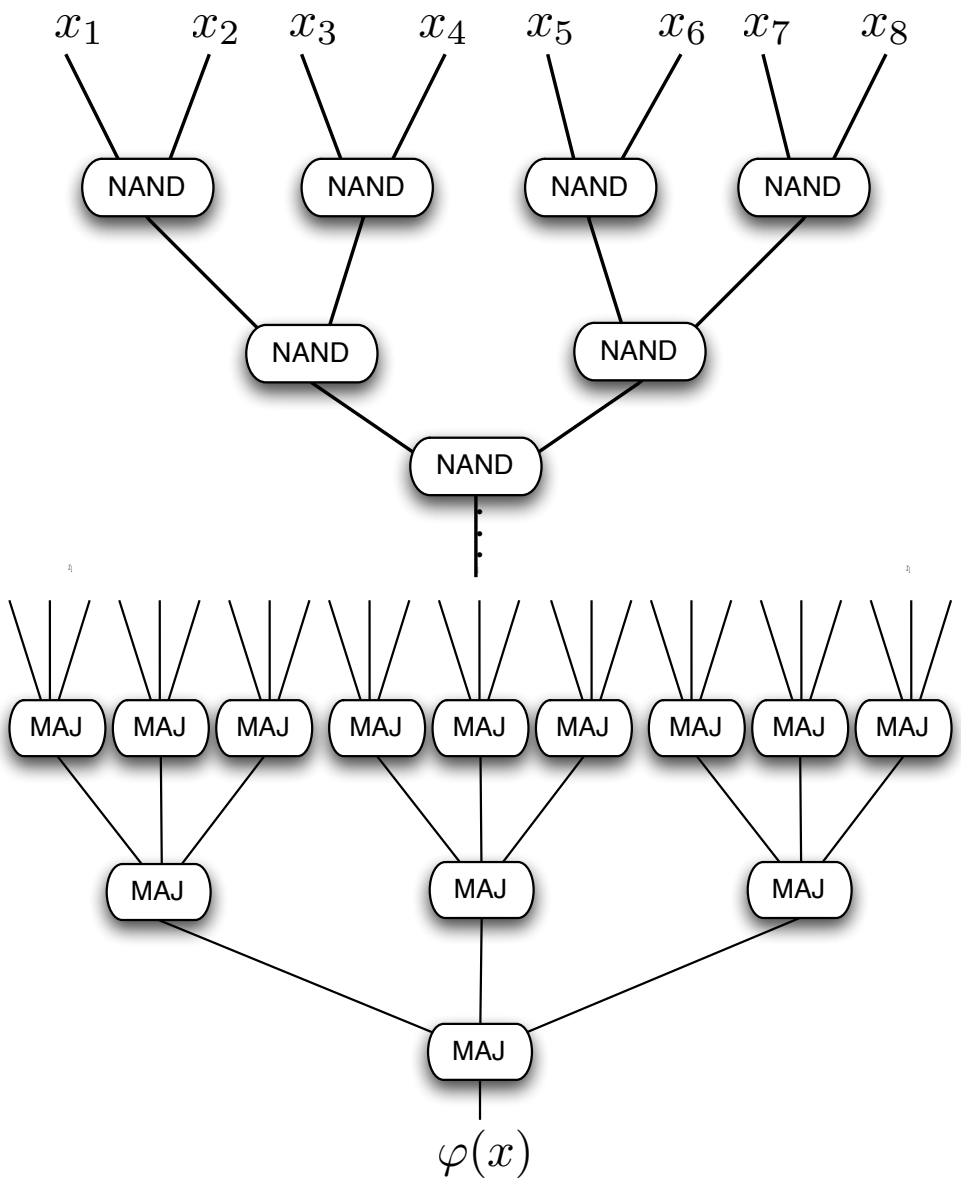

3. Extensions


Your Good Times are
Just Begining With
Lucky TORII

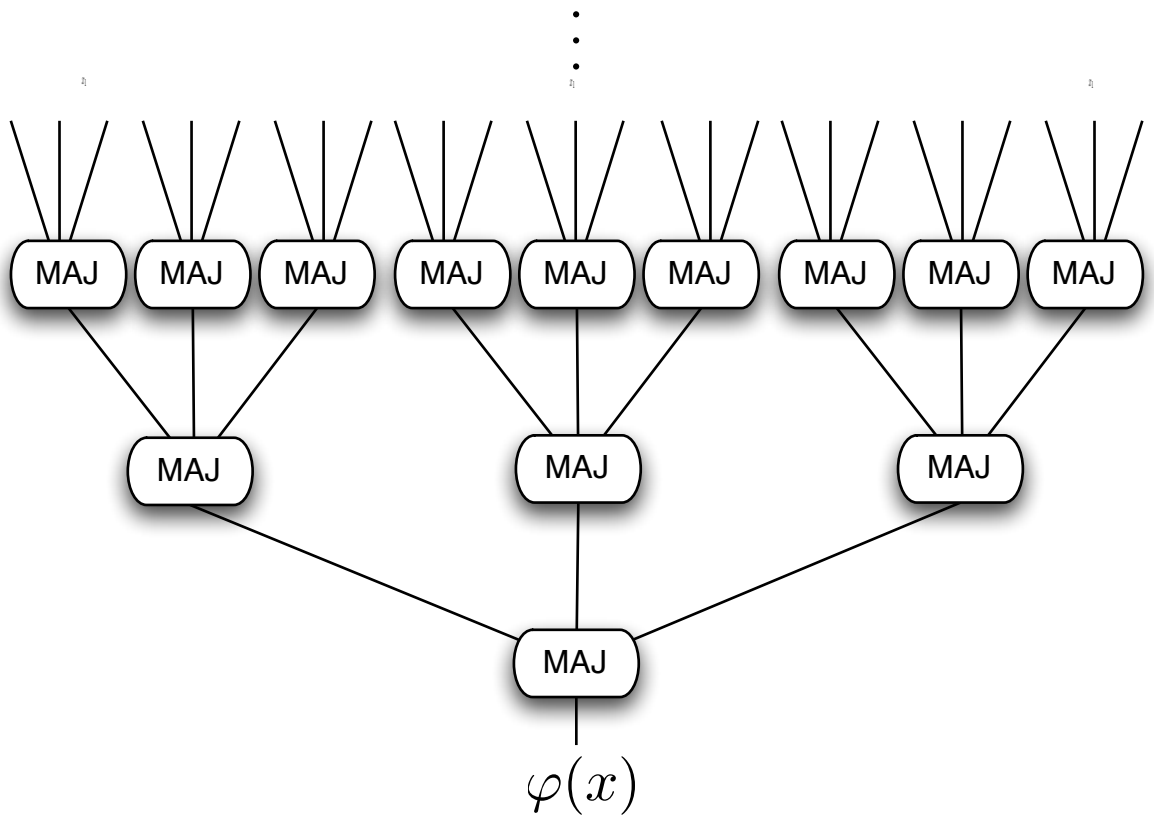
joint work with Robert Špalek

Extension: Formulas on different gate sets

- Cost to evaluate a formula that uses other gates besides {AND, OR, NOT}?
- First step: Balanced iterative functions (the same function composed on itself)
  - [Farhi, Goldstone, Gutmann '07]: Balanced recursive NAND gate formula
  - Other balanced iterative functions?



Recursive 3-bit majority tree



- What is the classical complexity of evaluating recursive MAJ_3 tree?

  [Boppana '86]

- Answer: Unknown!
  - Between $\Omega\left((7/3)^d\right)$ and $o\left((8/3)^d\right)$ for depth d

    [Jayram, Kumar & Sivakumar '03]

Recursive 3-bit majority tree

- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - Between $\Omega\left((7/3)^d\right)$ and $o\left((8/3)^d\right)$ [Jayram, Kumar & Sivakumar '03]
- Best quantum lower bound is $\Omega\left(\sqrt{C_0(f)C_1(f)}\right) = \Omega(2^d)$
- Quantum algorithm:
  - Expand majority into {AND, OR} gates:

$$\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$$

∴ {AND, OR} formula size increases is 5^d

∴ $O(\sqrt{5^d}) = O(2.24^d)$-query algorithm!

Recursive 3-bit majority tree

- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - Between $\Omega\left((7/3)^d\right)$ and $o\left((8/3)^d\right)$ [Jayram, Kumar & Sivakumar '03]

- Best quantum lower bound is $\Omega\left(\sqrt{C_0(f)C_1(f)}\right) = \Omega(2^d)$

- Quantum algorithm:
  - Expand majority into {AND, OR} gates:

$$\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$$

∴ {AND, OR} formula size increases is 5^d

∴ $O(\sqrt{5^d}) = O(2.24^d)$-query algorithm!

$< 7/3$ better than classical lower bound

Recursive 3-bit majority tree

- Classical complexity to evaluate recursive MAJ3-gate tree is unknown:
  - Between $\Omega\left((7/3)^d\right)$ and $o\left((8/3)^d\right)$ [Jayram, Kumar & Sivakumar '03]
- Best quantum lower bound is $\Omega\left(\sqrt{C_0(f)C_1(f)}\right) = \Omega(2^d)$
- Quantum algorithm:
  - Expand majority into {AND, OR} gates:

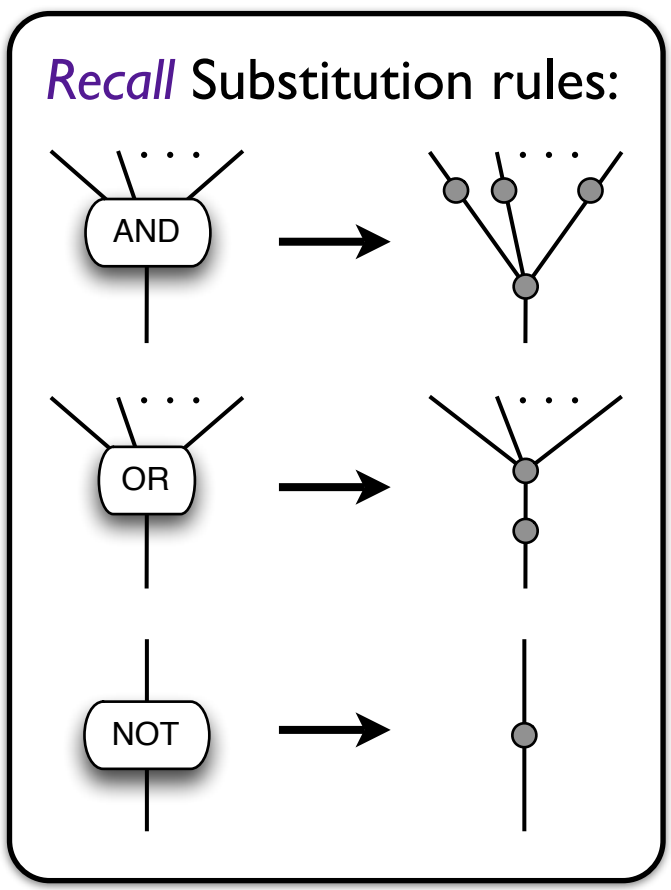
$$\mathrm{MAJ3}(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$$

  - ∴ {AND, OR} formula size increases is 5^d
  - ∴ $O(\sqrt{5^d}) = O(2.24^d)$-query algorithm
  - In fact, inputs are not arbitrary; worst-case inputs are promised not to occur
  - ∴ Improved analysis gives $O\left(\left(\sqrt{3+\sqrt{2}}\right)^d\right) = O\left(2.101\ldots^d\right)$

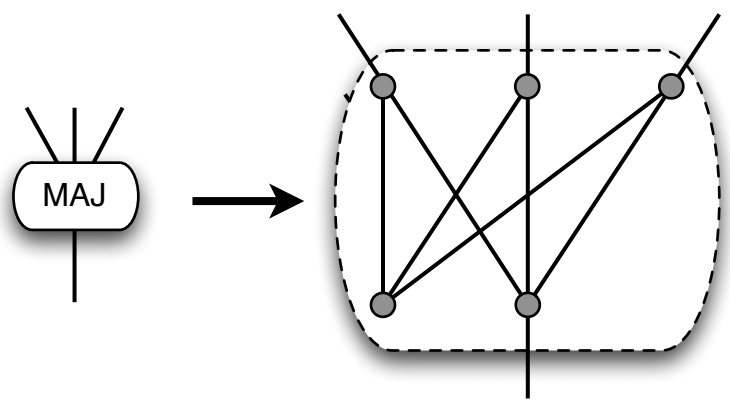
Different gate sets: Gate gadgets

- Classical complexity between $\Omega\left((7/3)^d\right)$ and $O\left((2.655\ldots)^d\right)$
- Quantum lower bound: $\Omega(2^d)$
- Quantum upper bound: $\mathrm{MAJ3} = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$ ⇒ O($\sqrt{5}^d$) = O(2.236…^{d})
  - improved analysis ⇒ O($\sqrt{(3+\sqrt{2})}^d$) = O(2.101…^{d})

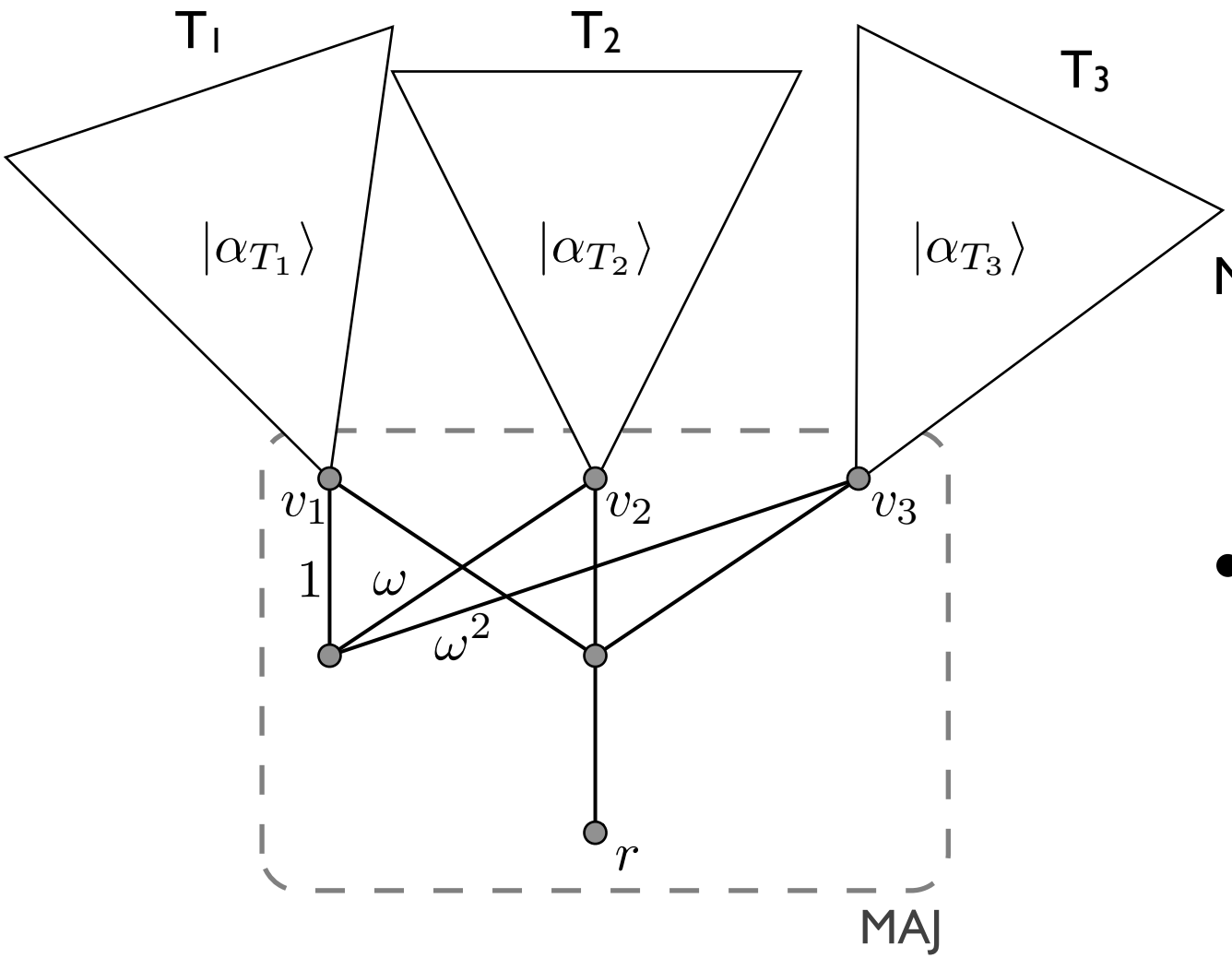
- Gate gadgets:



New MAJ3 substitution rule:



- **Inductive Hypothesis:**
  - $\varphi(v)=0 \Rightarrow \alpha_v=0$
  - $\varphi(v)=1 \Rightarrow \alpha_v$ can be $\neq 0$



MAJ3 gate gadget constraints:

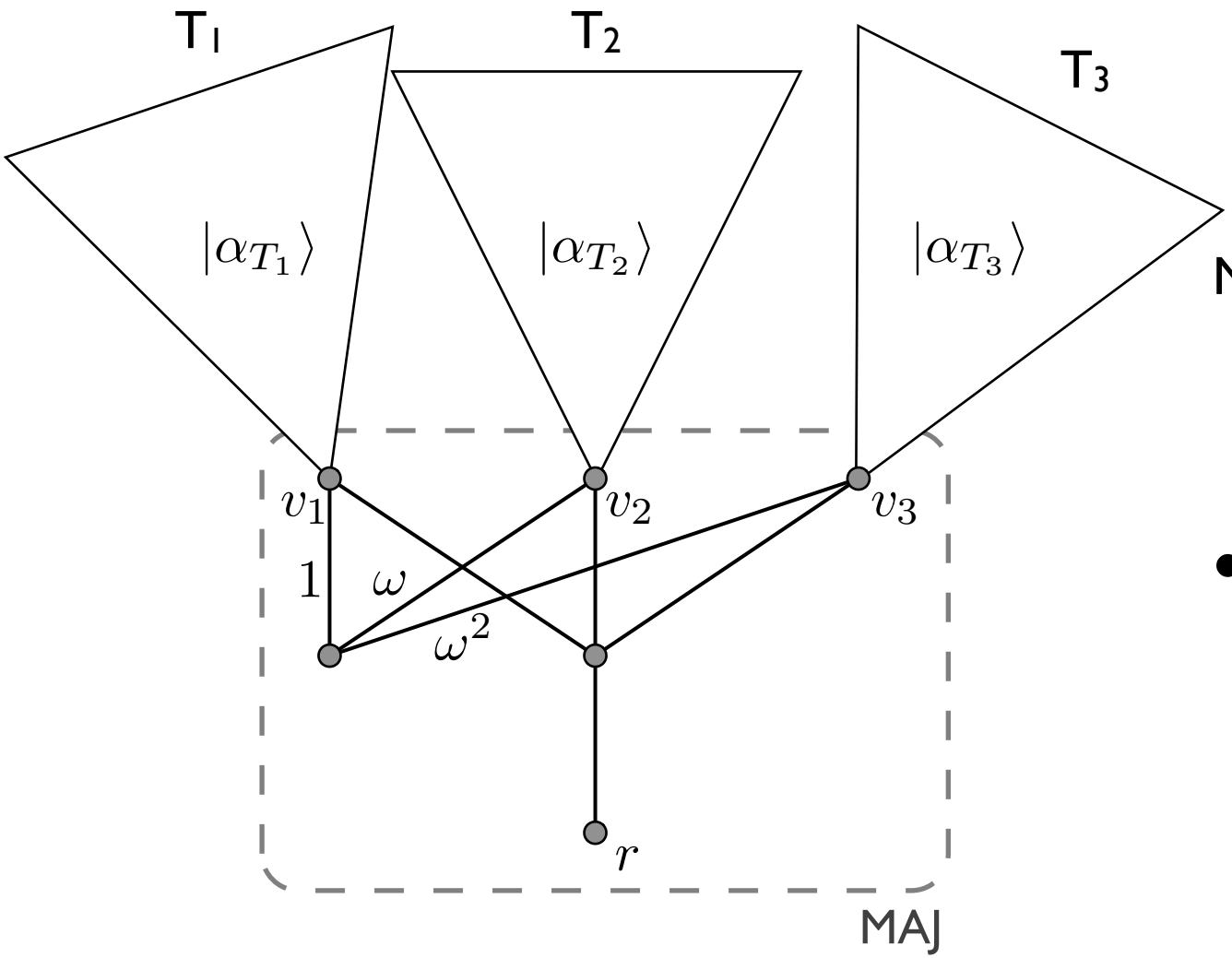
$$-\alpha_r = \alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3}$$

$$\alpha_{v_1} + \omega\alpha_{v_2} + \omega^2\alpha_{v_3} = 0$$

- At least two $\varphi(v_i)$ must be 1 to satisfy second constraint nontrivially.

✓ MAJ

- **Inductive Hypothesis:**
  - $\varphi(v)=0 \Rightarrow \alpha_v=0$
  - $\varphi(v)=1 \Rightarrow \alpha_v$ can be $\neq 0$



MAJ3 gate gadget constraints:

$$-\alpha_r = \alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3}$$

$$\alpha_{v_1} + \omega\alpha_{v_2} + \omega^2\alpha_{v_3} = 0$$

- At least two $\varphi(v_i)$ must be 1 to satisfy second constraint nontrivially.

✓ MAJ

$\Rightarrow$ (Near) Optimal $O(2^{d(1+o(1))}) = O(N^{\log_3 2+o(1)})$-query balanced recursive MAJ_3 formula evaluation algorithm

More results…

⇒ (Near) Optimal $O(2^{d(1+o(1))}) = O(N^{\log_3 2+o(1)})$-query balanced recursive MAJ_3 formula evaluation algorithm, based on new substitution gadget

- Furthermore, either by
  - Analysis of AND-OR formula expansion on promised inputs,
  - Or by constructing new "gadget" substitution rules

AKA "**Span programs**," well-studied in classical complexity theory

⇒ Nearly optimal algorithms for iterative versions of **all 3-bit functions**, some 4-bit functions (38 of 208 inequivalent functions)

Remarks on formula evaluation algorithms:

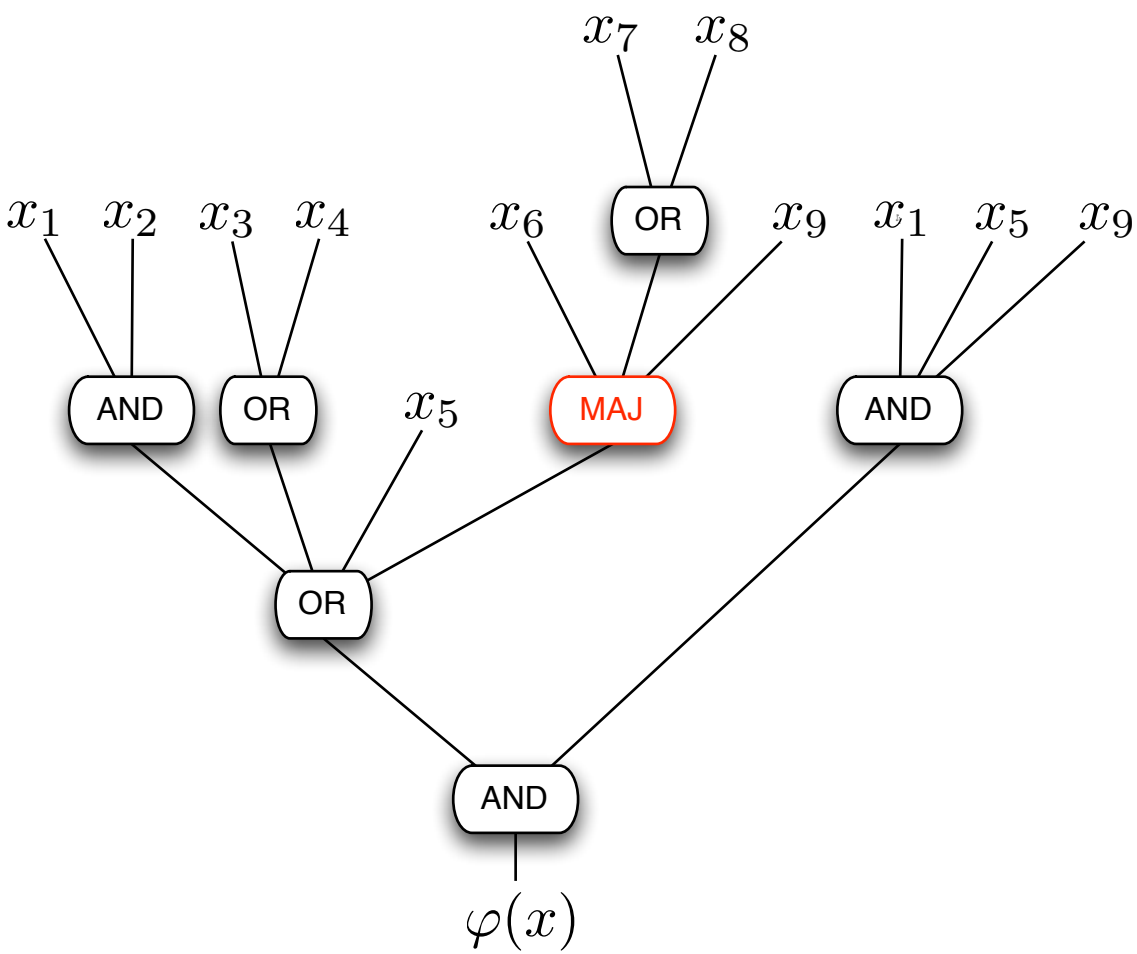
Classical vs. Quantum

- Classical complexity of evaluating balanced k-ary alternating AND-OR tree is $(k/2)^{depth} = N^{\sim(1-1/\log_2 k)}$ — approaches N as k increases
  - Classical complexity of evaluating general AND-OR formulas is not known?
- Classical complexity of evaluating iterative MAJ_3 formula is unknown: between $\Omega\left((7/3)^d\right)$ and $o\left((8/3)^d\right)$
  - (the generalization of the optimal AND-OR algorithm is *not* optimal when applied to MAJ_3 trees)

- Quantumly, complexity is $N^{1/2}$ queries always, all the way up to k=N (i.e., evaluating $OR(x_1,\ldots,x_N)$, Grover search)
  - General AND-OR formulas can be evaluated with $N^{1/2+o(1)}$ queries
- Quantumly, the AND-OR algorithm generalizes to give optimal algorithm for evaluating iterated f, where f is any 3-bit function

Further extensions: Mixing different gates

- Algorithm works for unbalanced formulas with mixtures of different gates — but is it optimal?



Further extensions: Mixing different gates

- Algorithm works for unbalanced formulas with mixtures of different gates — but is it optimal?
- Answer: Work in progress. Unclear, but in general, probably *not*.

Promising:

- "Layered" balanced formulas (at each depth, one gate type used) are okay
- {AND, OR, +} compose well on top; e.g., $\mathrm{AND}(c_1,\ldots,c_k) = \sqrt{(c_1^2+\ldots+c_k^2)}$
- "Adversary-balanced" formulas on gate set S:
  - S' = {arbitrary two- or three-bit gates, $\mathrm{EQUAL}_{O(1)}$ gates}
  - S = { bounded-size {AND, OR, NOT, PARITY} formulas on inputs that are themselves possibly gates from S' }

Discouraging:

- With the exception of {AND, OR, PARITY}, we do not know how to evaluate optimally formulas with inputs that do not have balanced $\mathrm{ADV}^+ = \mathrm{ADV}^\pm$ bounds
  - Simple examples (~~e.g., $\mathrm{MAJ}_3(x_1,x_2,x_3 \oplus x_4)$, $\mathrm{MAJ}_3(x_1,x_2,x_3 \wedge x_4)$~~) suggest that gadget weights cannot be optimized to match lower bounds

(x1 & (x2 v x3)) v (!x1 & (x4 v (!x2 & !x3)))

- Results:
  - $N^{1/2+o(1)}$-time quantum algorithm for general {AND, OR, NOT} formulas (after efficient preprocessing); $O(\sqrt{N})$-query quantum algorithm for "approximately balanced" {AND, OR, NOT} trees (optimal!)
  - Nearly optimal query qu. algs. for iterative versions of all 3-bit functions (e.g., MAJ_3), and for 38 of 208 inequivalent 4-bit functions—extended to "adversary balanced" S-formulas

Open problems

- More optimal formula types:
  - Extension to allow other gates
  - Mix different gates in the same formula. Inductive hypotheses are compatible, so algorithm works — but it may or may not be optimal.
  - Better lower bounds: Want to understand qu. lower bounds ADV^+ versus $ADV^\pm$ [Høyer, Lee, Špalek '07], esp. gate composition
- Can a witness set be extracted from the eigenstate?
- Classical connections:
  - More significant connections to classical "span programs"
  - Significance of classical random walk?
  - Open Classical ?: Is [BCE'91] formula rebalancing optimal?
    - Does there exist formula φ, k such that every equivalent φ' of depth at most k log N has size(φ') $\geq N^{1+1/\log k}$?