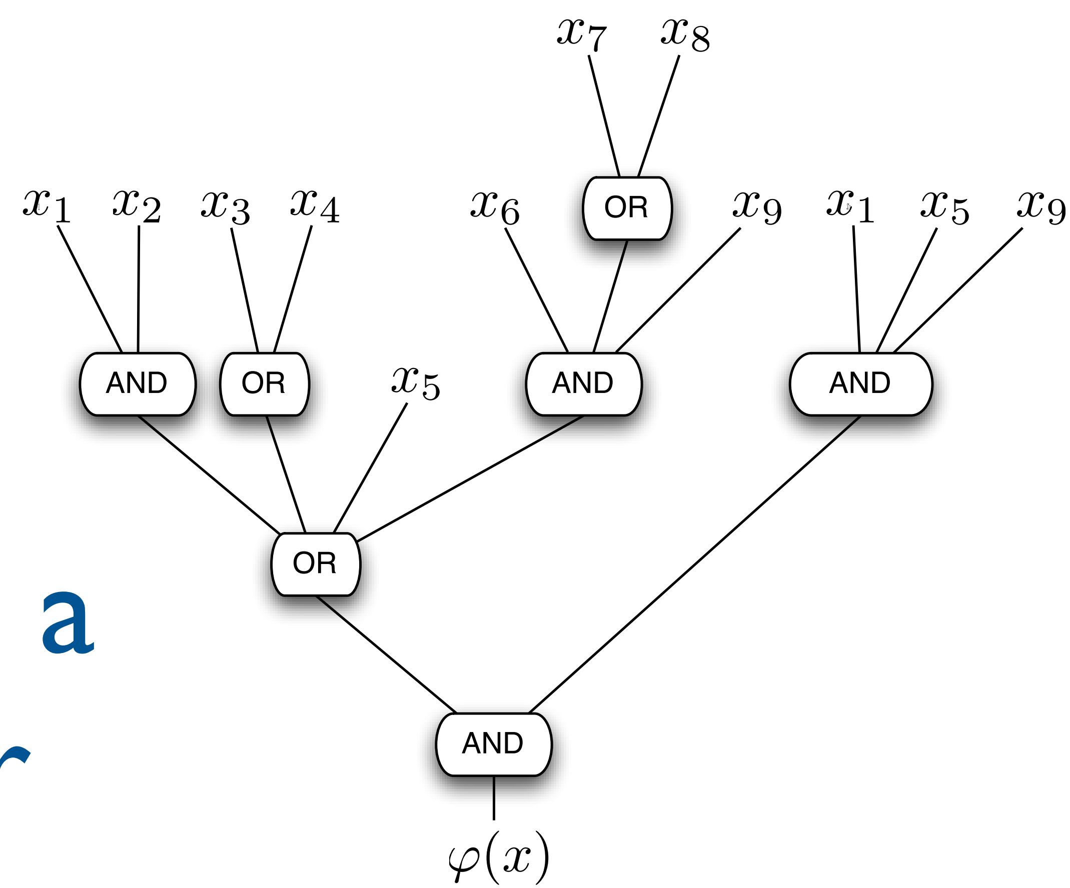
Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer



Andris Ambainis
U. Latvia

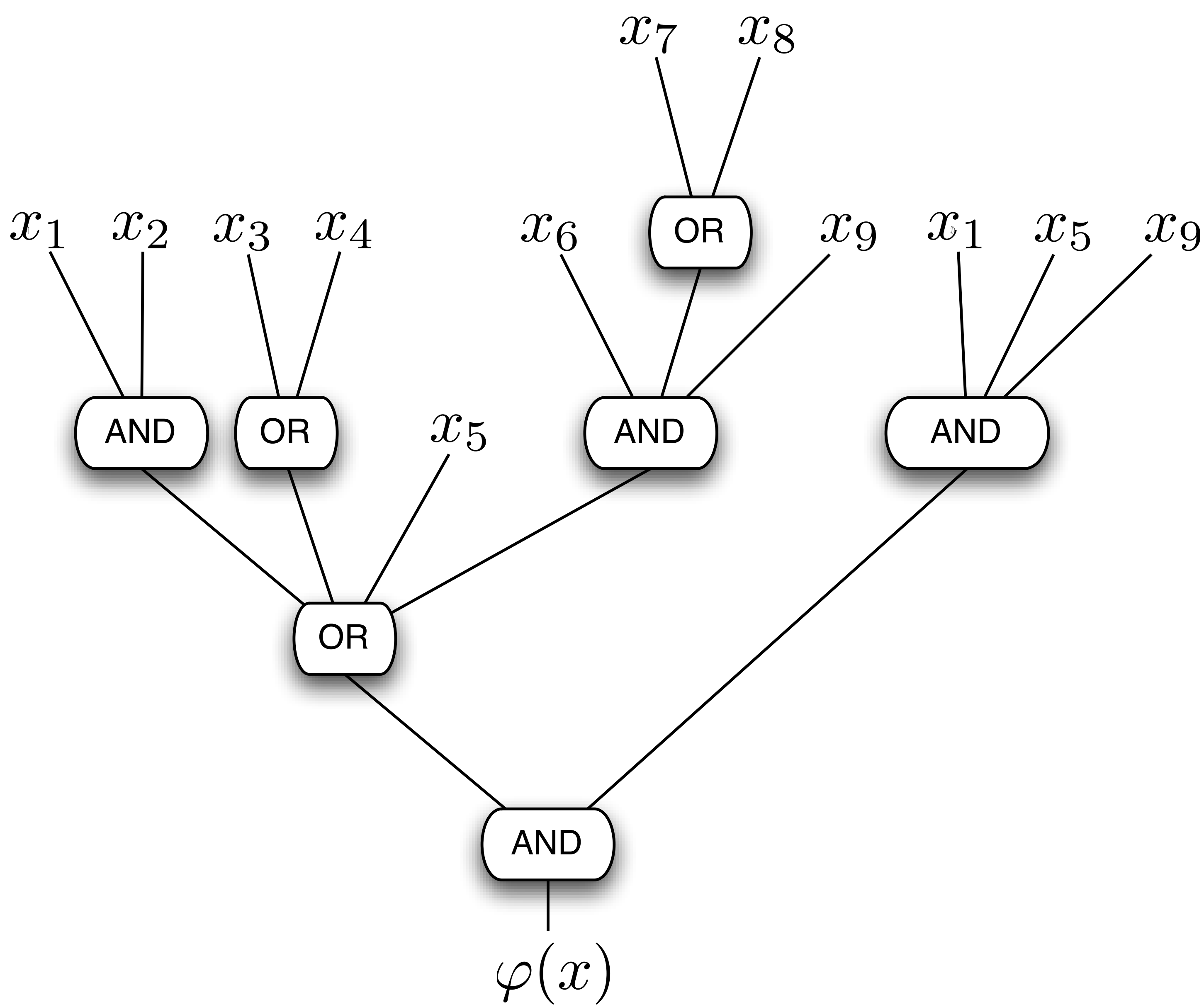
Andrew Childs
U. Waterloo

Robert Špalek
Google

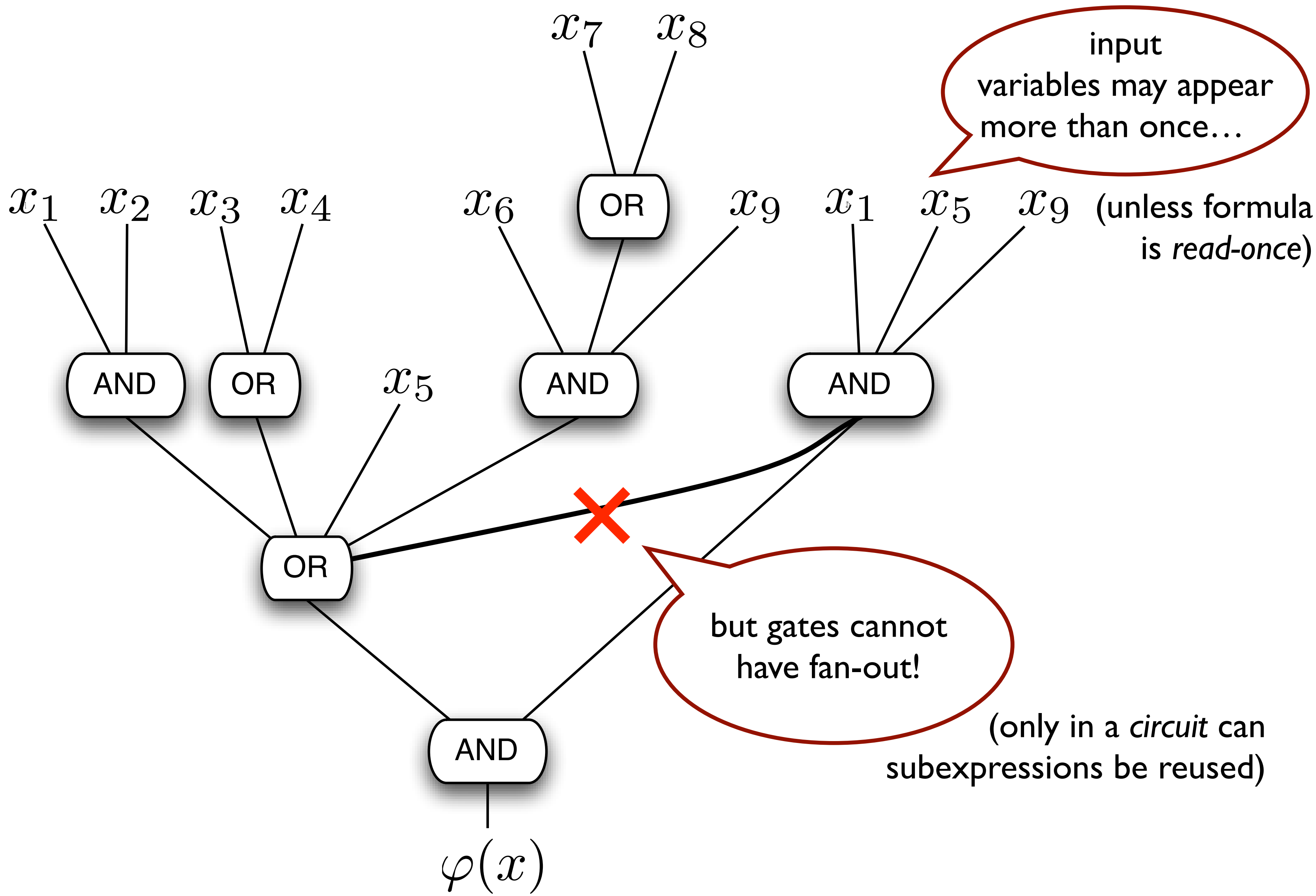
Shengyu Zhang
Caltech

Ben Reichardt
Caltech

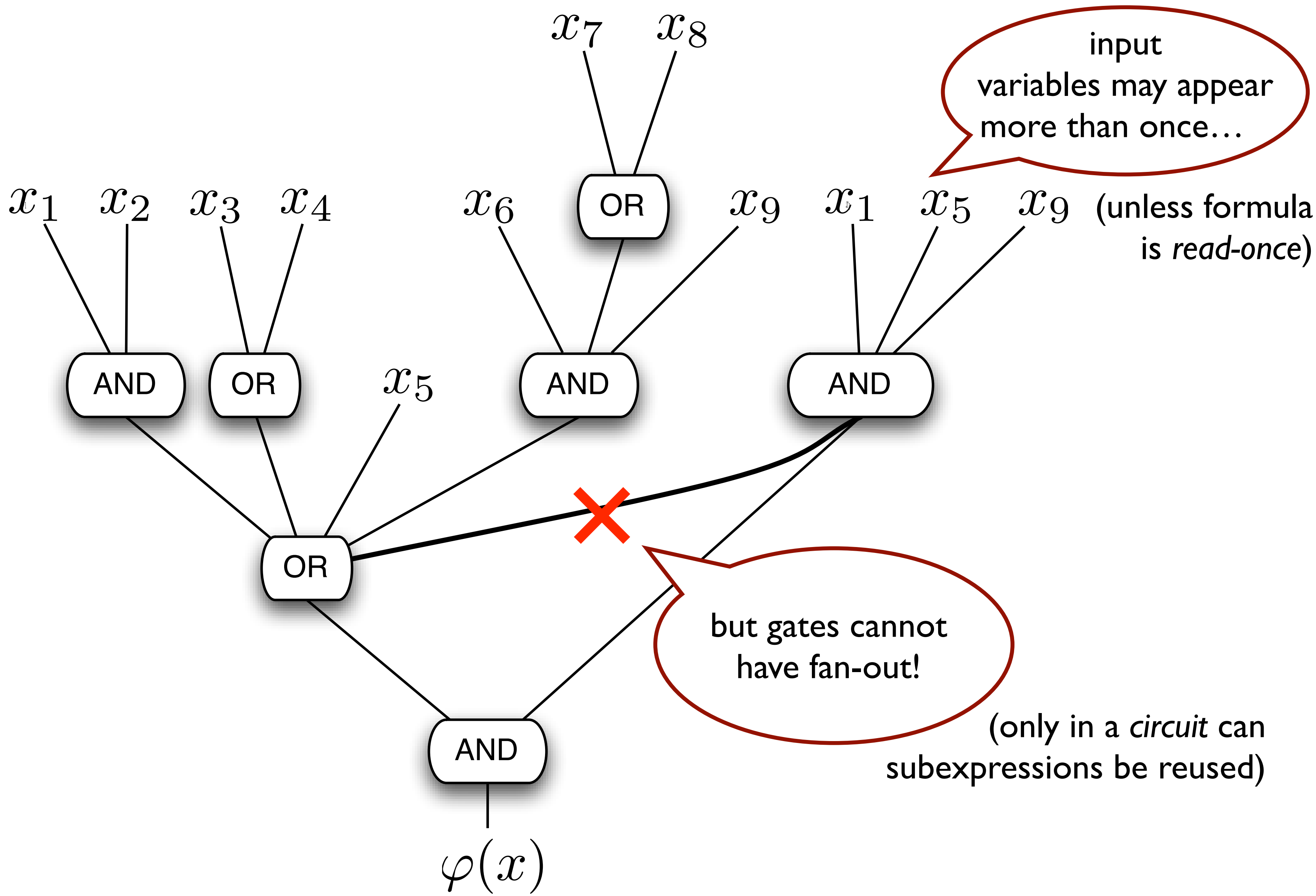
Def: {AND, OR, NOT} Formula = Tree of nested gates



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



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



Problem: Evaluate φ(x). (Formula/Game tree evaluation problem)

Problem history: Classical computation

- Problem: Evaluate the formula, with minimal queries to the inputs bits x_i.
- Classical history
  - Some formulas, e.g., OR(x_1, x_2, …, x_N), require $\Omega(N)$ time
  - Randomized algorithm in E-time $O(N^{0.754})$ ← $\log_d \lambda_{max}(\begin{pmatrix} 0 & d \\ 1 & \frac{d-1}{2} \end{pmatrix})$
    for balanced binary AND-OR formulas [Snir '85, Saks & Wigderson '86]
    - Flip coins to decide which subtree to evaluate next, short-circuit
    - Optimal [SW '86, Santha '95]
  - General formulas, ??

Problem history: Quantum computation

- Classical history
  - Randomized algorithm in E-time $\Theta(N^{0.754})$ for balanced binary formulas
  - Other formulas may require $\Omega(N)$ time
- Quantum history
  - $\Omega(\sqrt{N})$ queries required for read-once [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 [Buhrman, Cleve, Wigderson '98]
    - Search on faulty oracles [Høyer, Mosca, de Wolf '03] $\Rightarrow O(\sqrt{N}\, c^d)$ queries

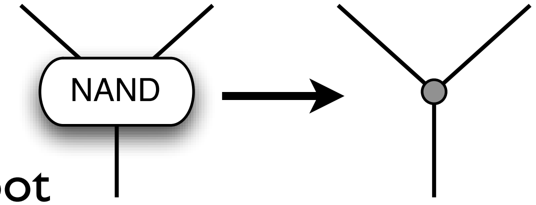
- Classical history
  - Randomized algorithm in E-time $\Theta(N^{0.754})$ for balanced binary formulas
  - Other formulas may require $\Omega(N)$ time
- Quantum history
  - $\Omega(\sqrt{N})$ queries required for read-once [Barnum, Saks '04]
  - Grover search: Evaluates $OR(x_1, x_2, \ldots, x_N) = \begin{cases} 1 & \text{if } \exists \text{ an } i : x_i = 1 \\ 0 & \text{otherwise} \end{cases}$ using $O(\sqrt{N})$ queries ($O(\sqrt{N} \log \log N)$-time)
  - Can be applied recursively to evaluate shallow trees

Quantum Leap!

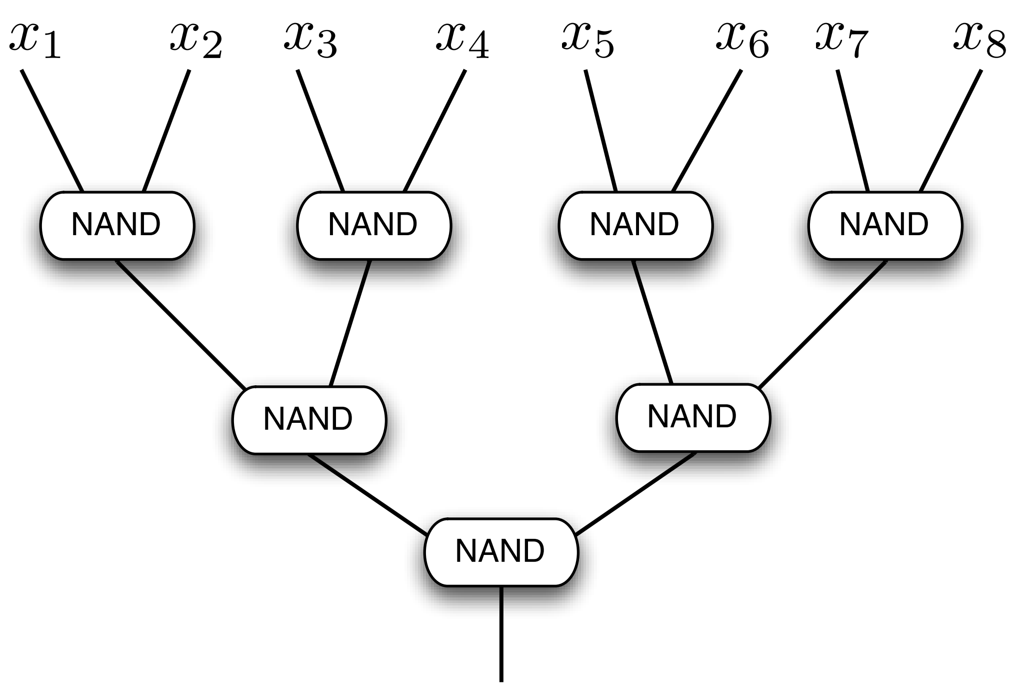
- Farhi, Goldstone, Gutmann 2007: Breakthrough quantum algorithm for evaluating balanced binary AND-OR formula in $N^{\frac{1}{2}+o(1)}$ time

Farhi, Goldstone, Gutmann '07 algorithm

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

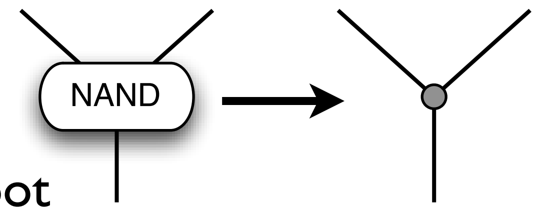


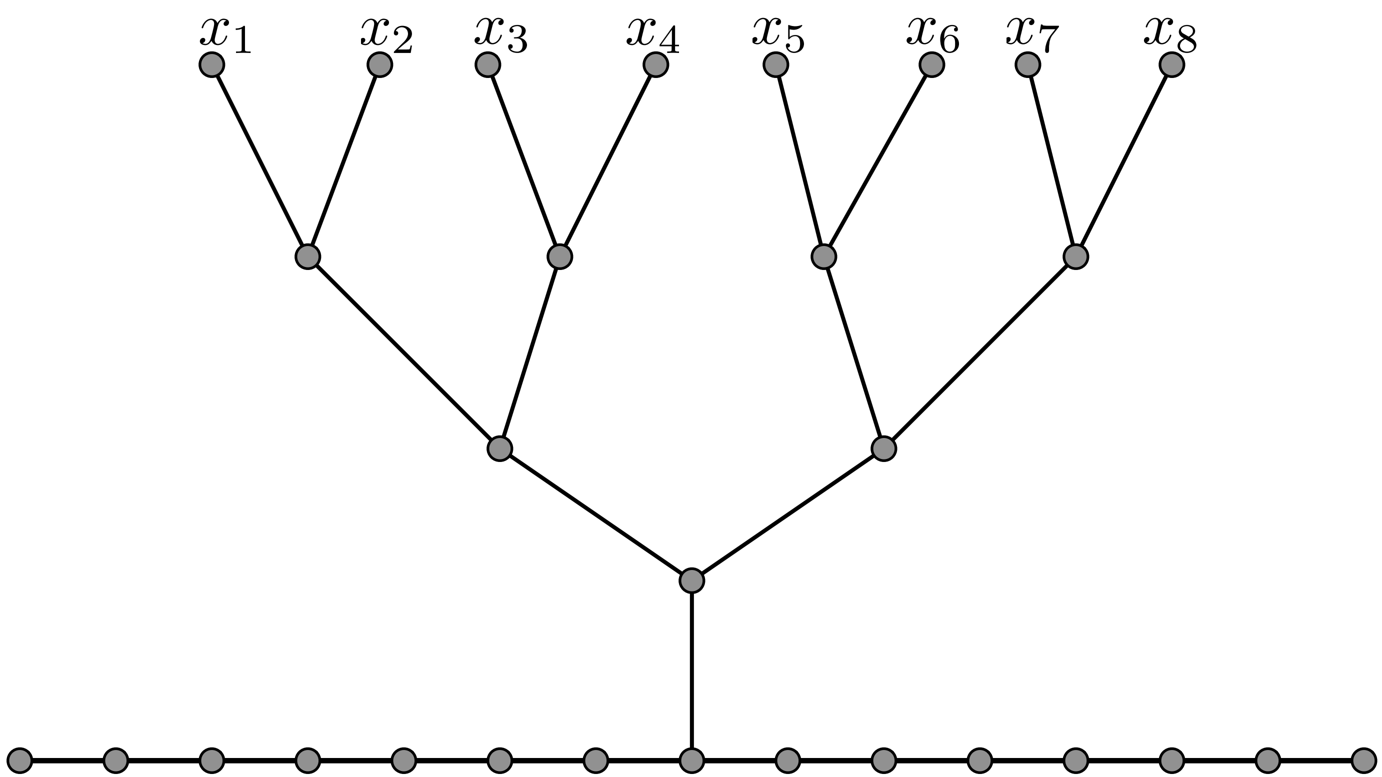
- Attach an infinite line to the root



Farhi, Goldstone, Gutmann ‘07 algorithm

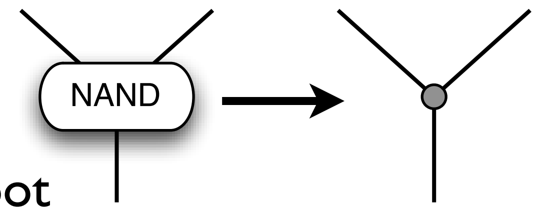
- **Theorem** ([FGG ‘07, CCJY ‘07]): A balanced binary NAND formula can be evaluated in time $N^{½+o(1)}$.
- Convert formula to a tree:
- Attach an infinite line to the root



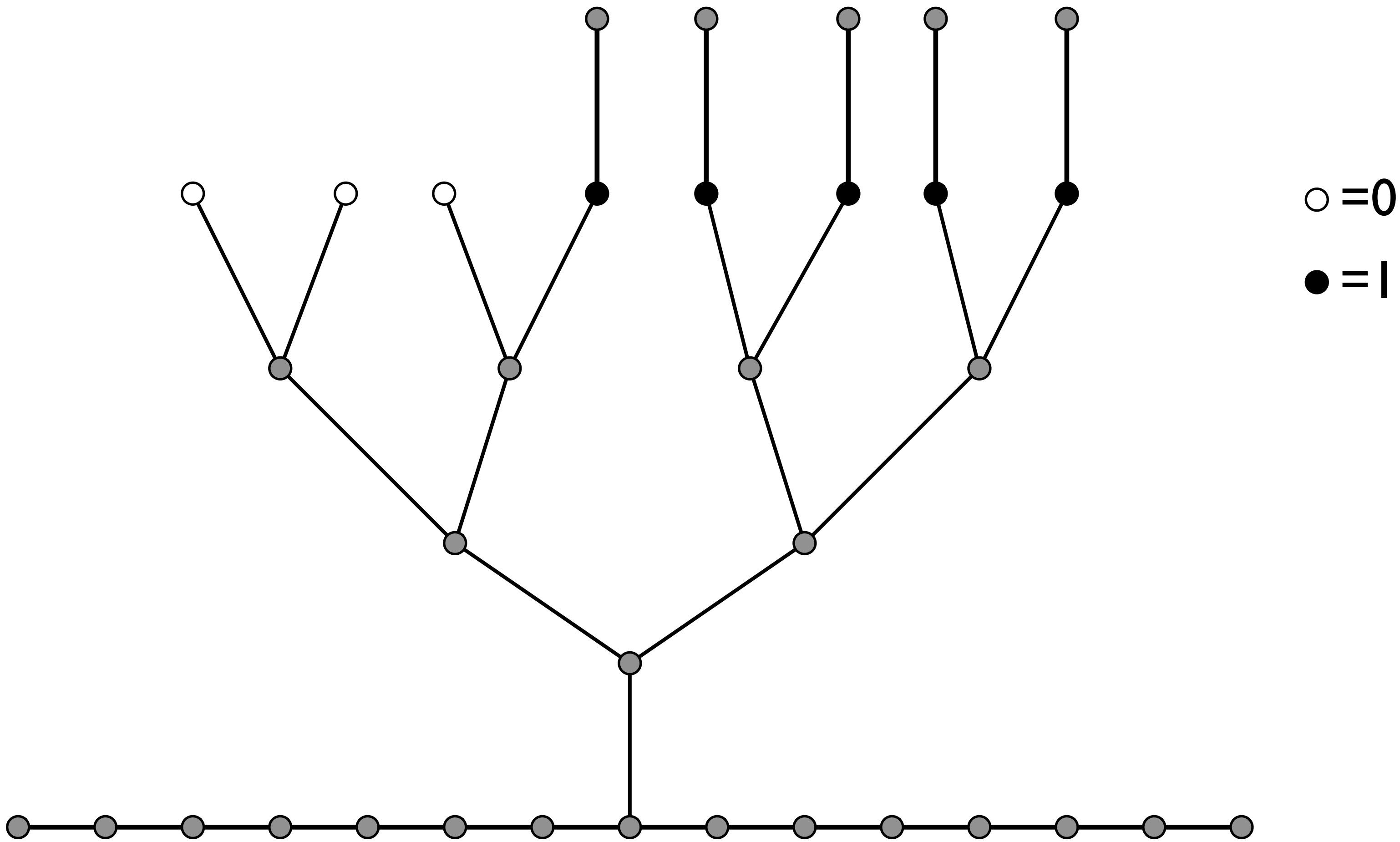


Farhi, Goldstone, Gutmann '07 algorithm

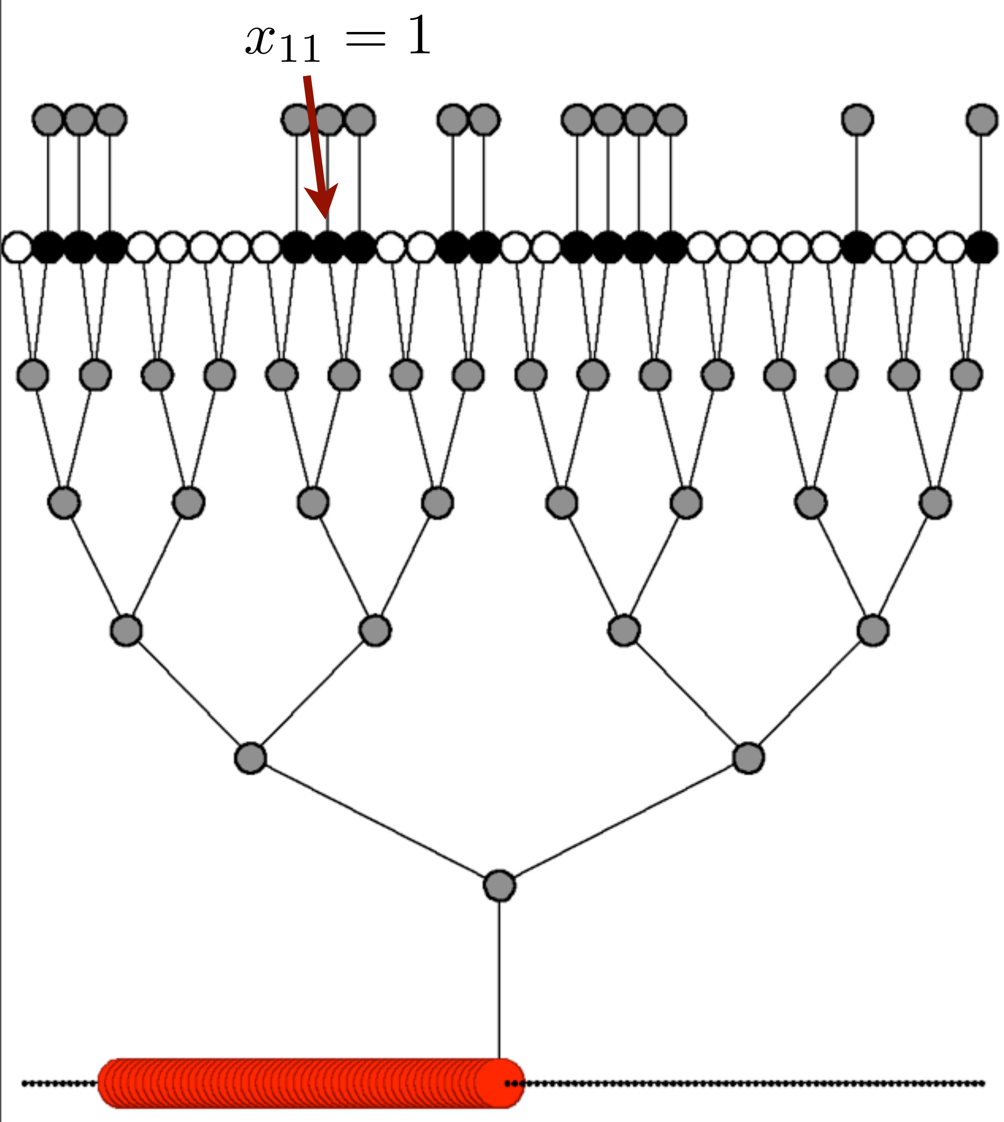
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary NAND formula can be evaluated in time $N^{\frac{1}{2}+o(1)}$.
- Convert formula to a tree:

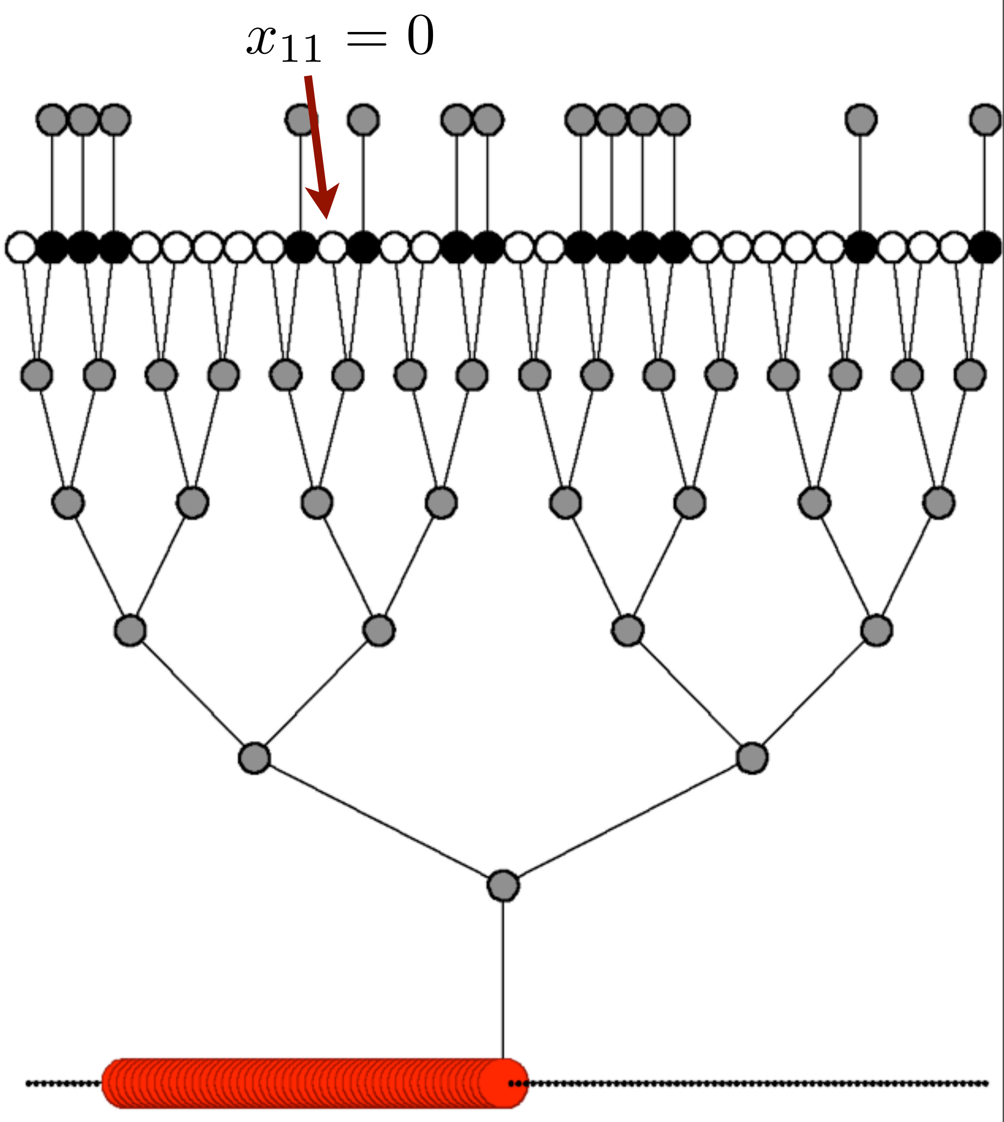


- Attach an infinite line to the root
- Add edges above leaf nodes evaluating to one…

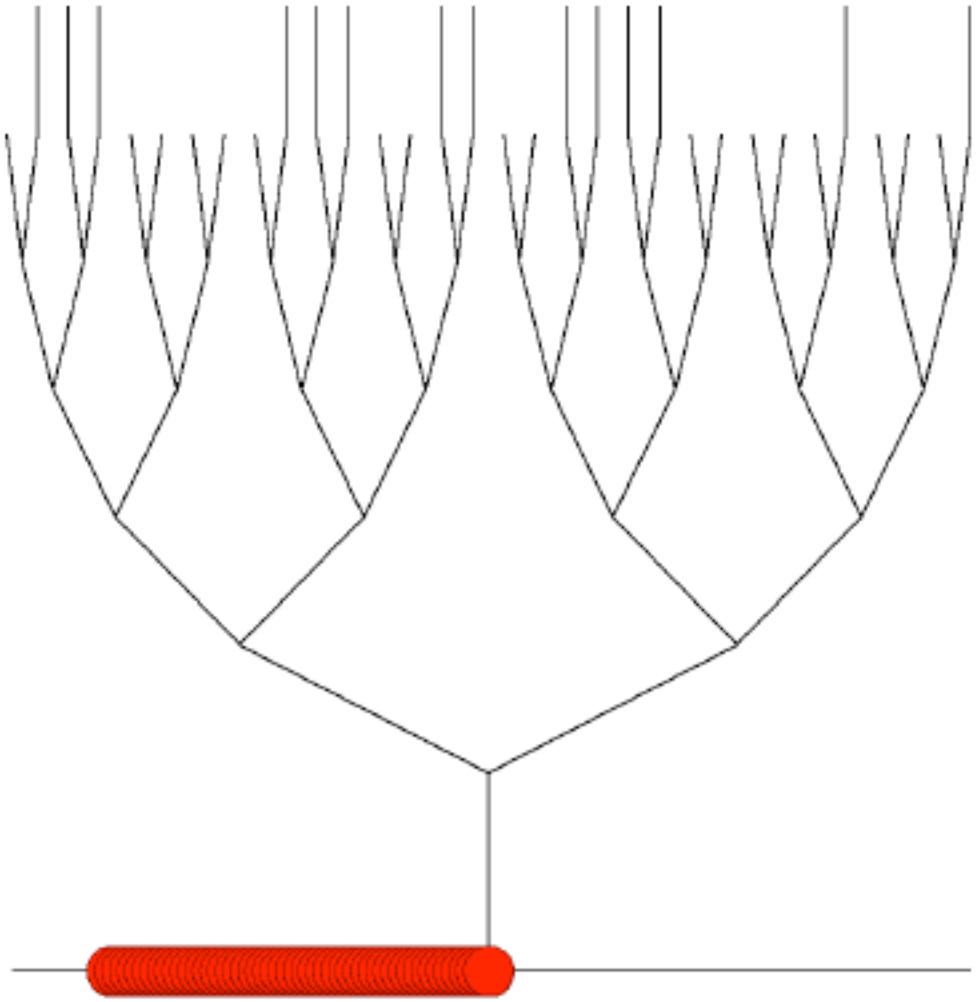


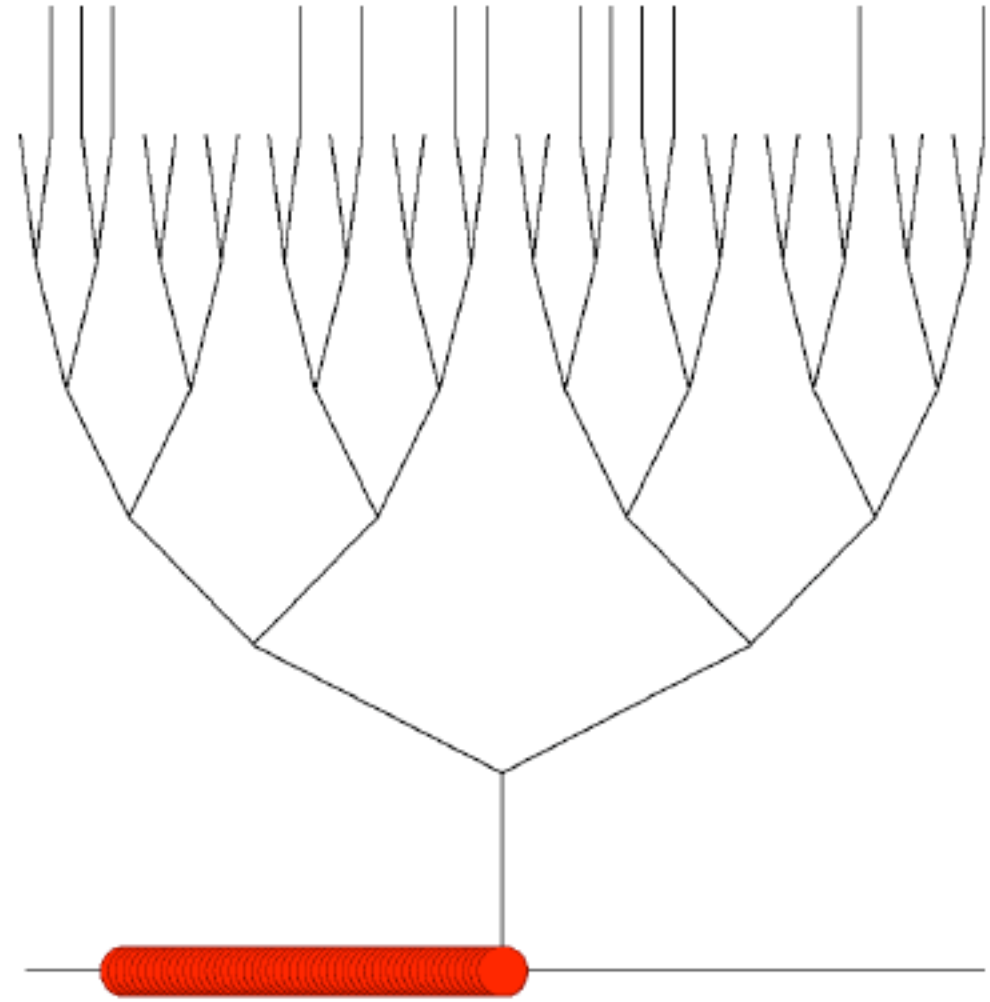
Continuous-time quantum walk [FGG '07]



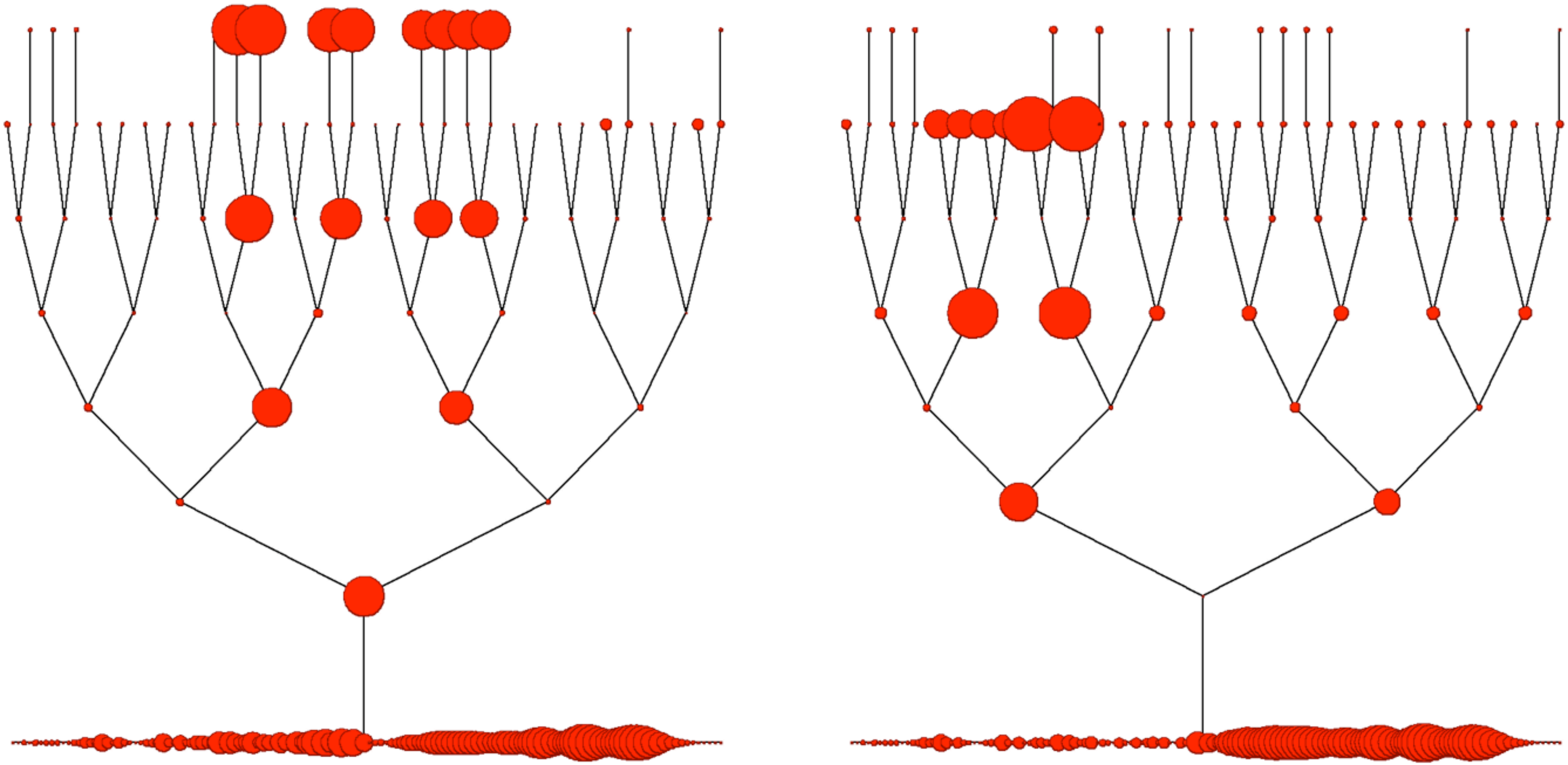


FGG quantum walk $|\psi_t\rangle = e^{iA_G t}|\psi_0\rangle$

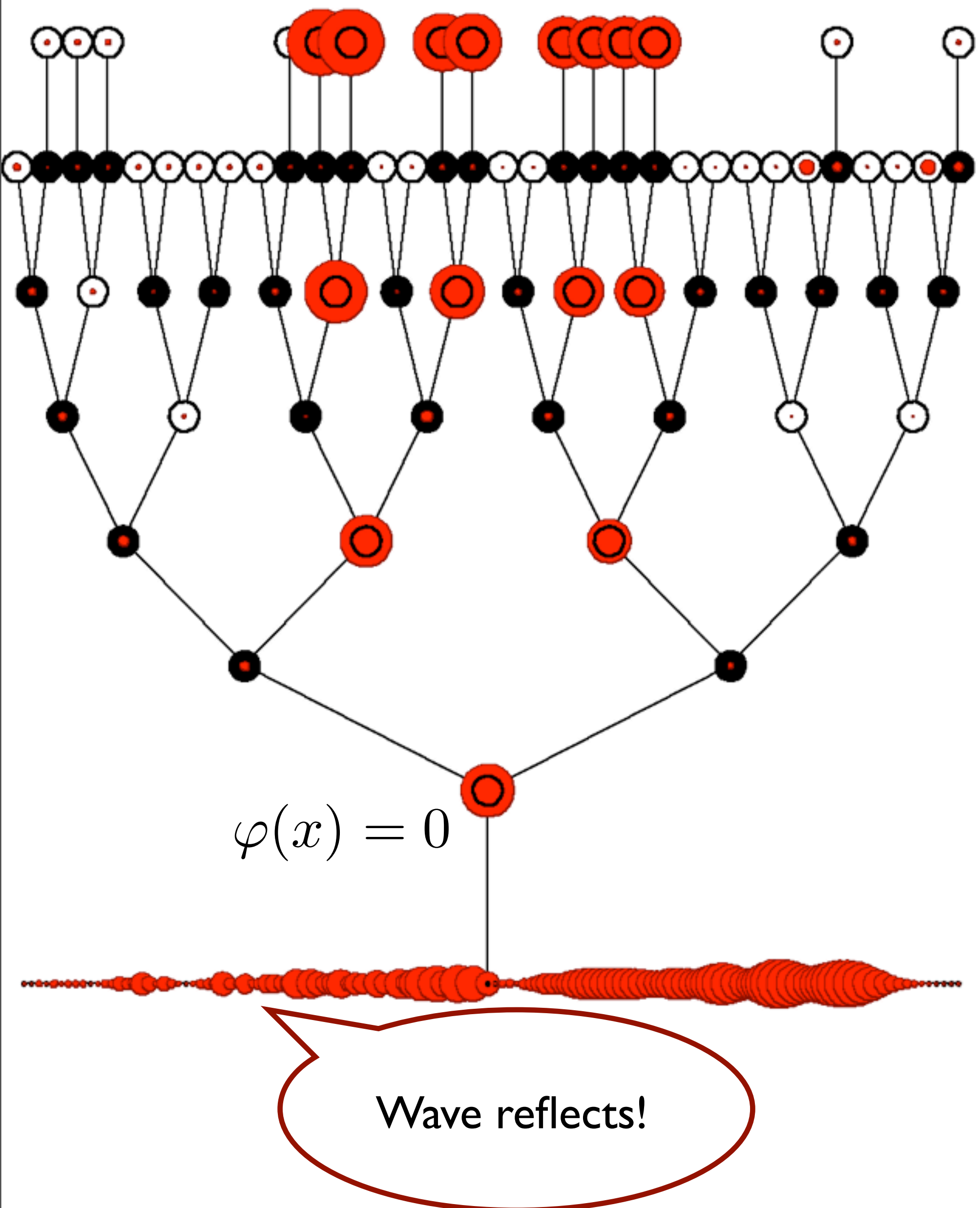


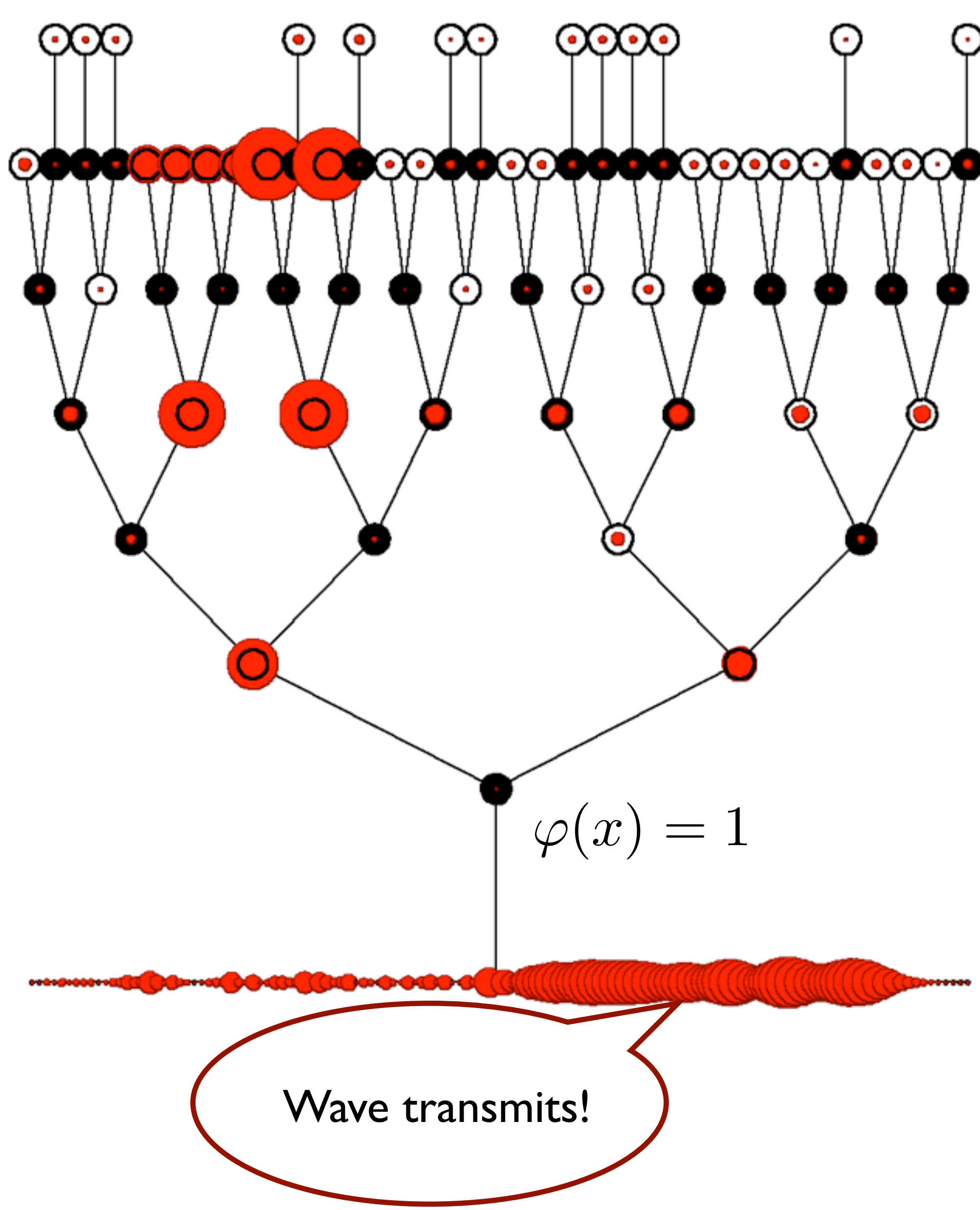


FGG quantum walk $|\psi_t\rangle = e^{iA_G t}|\psi_0\rangle$



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

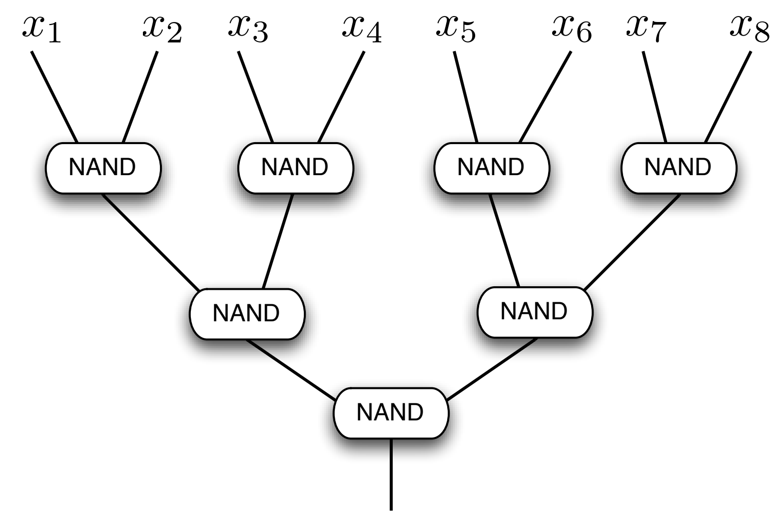




[FGG '07] algorithm

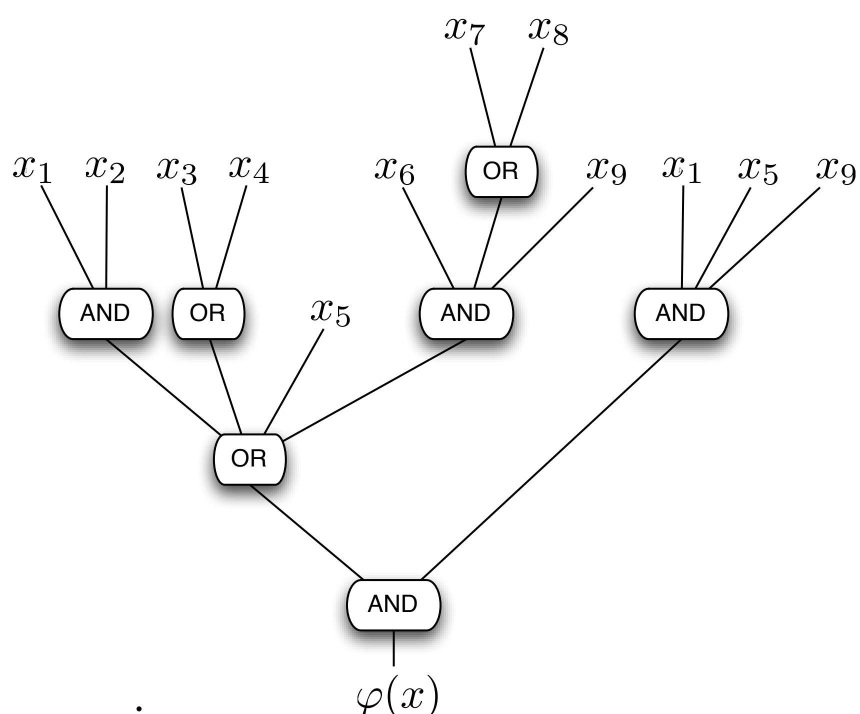
- **Theorem** ([FGG '07, CCJY '07]): A balanced binary AND-OR formula can be evaluated in time $N^{½+o(1)}$.

  Analysis by scattering theory.



[ACRŠZ '07] algorithm

- **Theorem**:
  - An "approximately balanced" AND-OR formula can be evaluated with O(√N) queries (optimal for read-once!).
  - A general AND-OR formula can be evaluated with $N^{½+o(1)}$ queries.



Running time is $N^{½+o(1)}$ in each case, after efficient preprocessing.

Remarks on formula evaluation algorithms:

Classical vs. Quantum

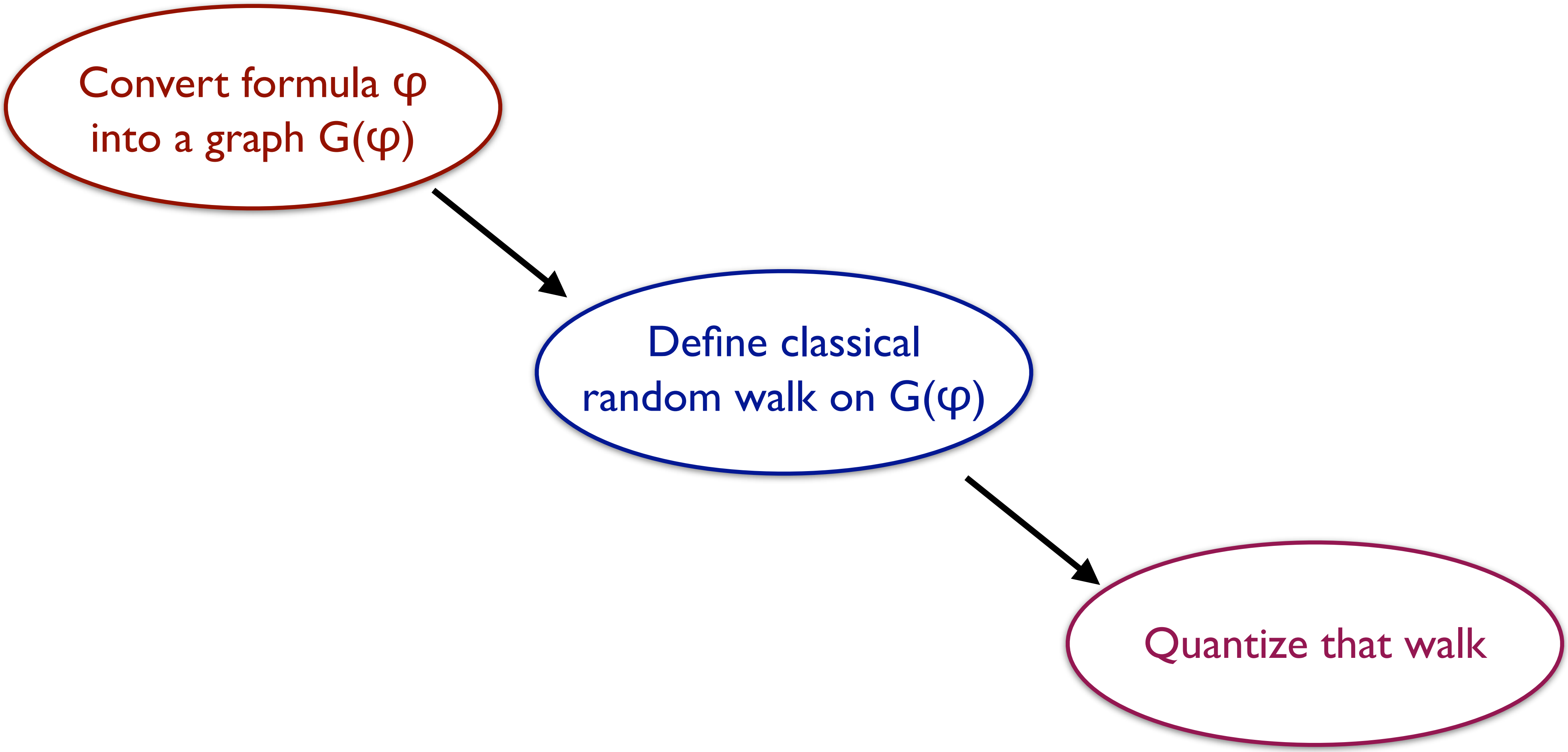
Classical

- Classical complexity of evaluating balanced k-ary alternating AND-OR tree is $(k/2)^{\text{depth}} = N^{\sim(1-1/\log_2 k)}$ — 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)

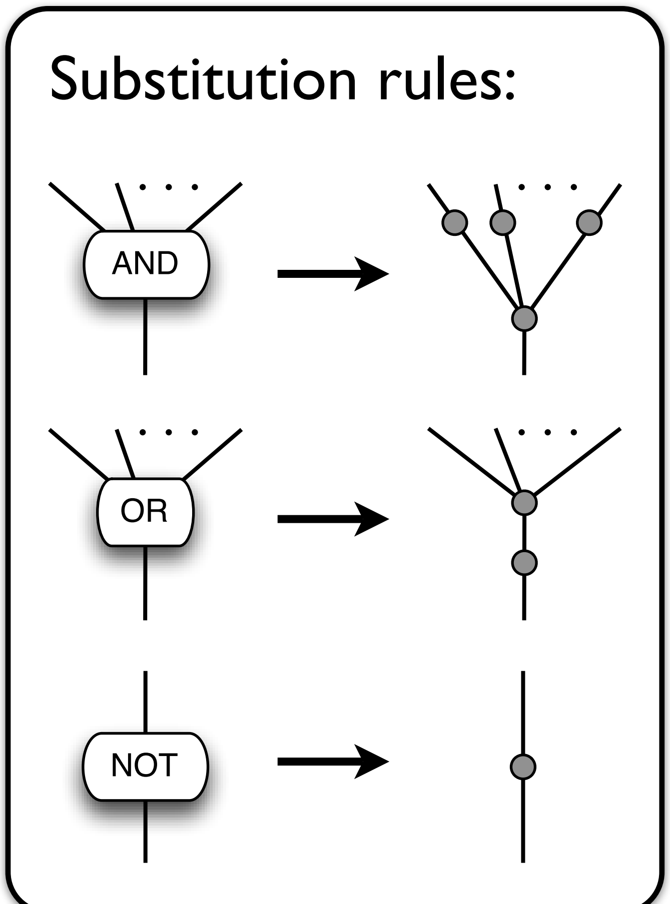
Quantum

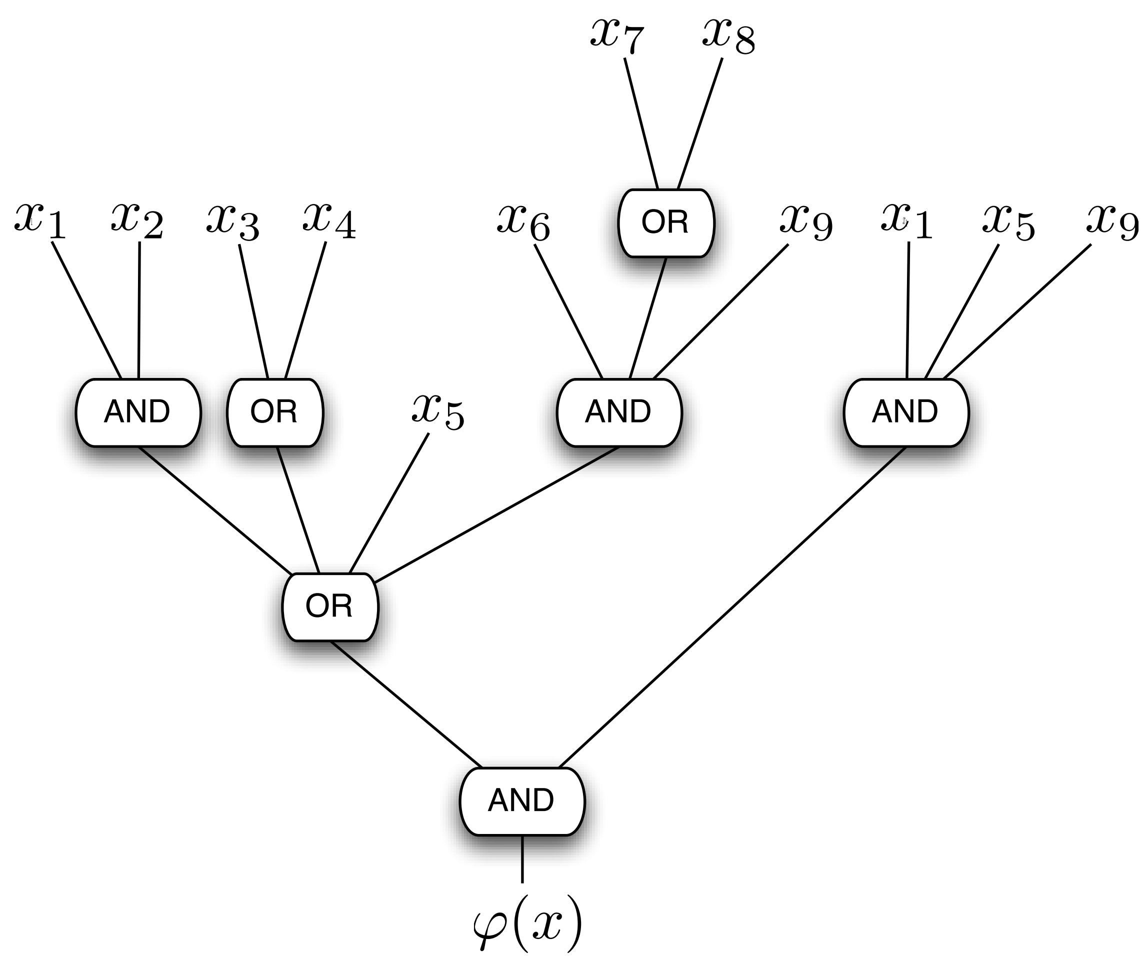
- Quantumly, complexity is $\sqrt{N}$ queries always, all the way up to k=N (i.e., evaluating $OR(x_1,\ldots,x_N)$, Grover search)
  - General AND-OR formulas can be evaluated with $N^{1/2+o(1)}$ queries
- Expanding MAJ_3 into AND-OR gates gives $O(\sqrt{5}^d)$ quantumly.
- Also, the algorithm *generalizes* to give optimal algorithm for evaluating iterated f, where f is any 3-bit function

Formula evaluation algorithm



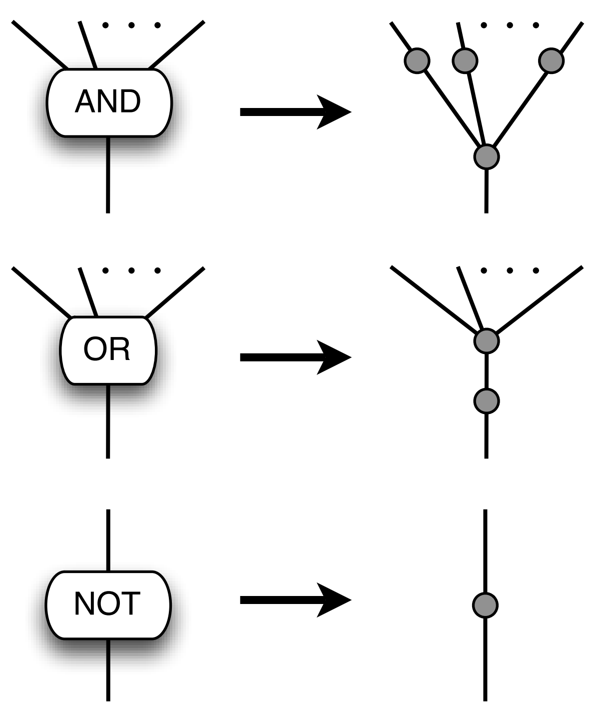
Convert formula φ into a graph G(φ) → Define classical random walk on G(φ) → Quantize that walk

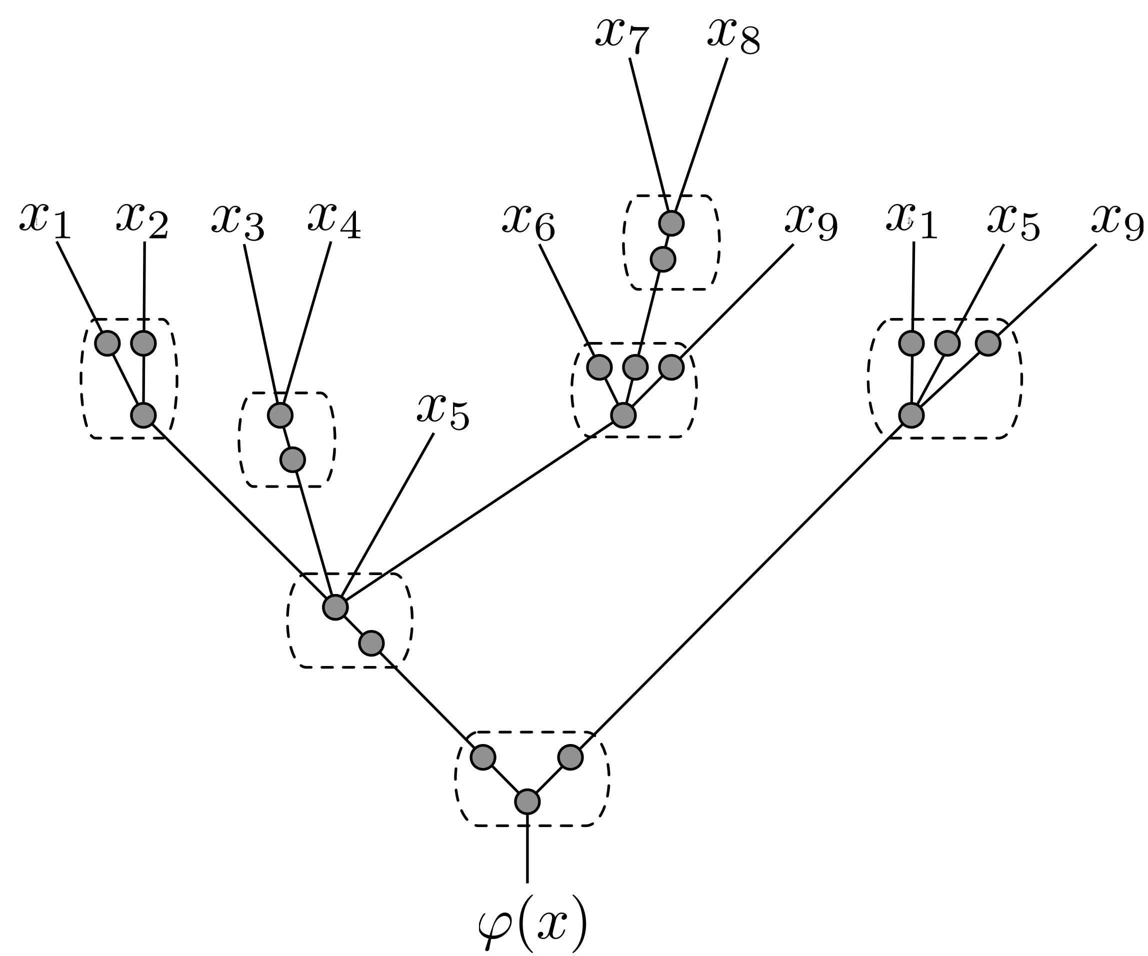




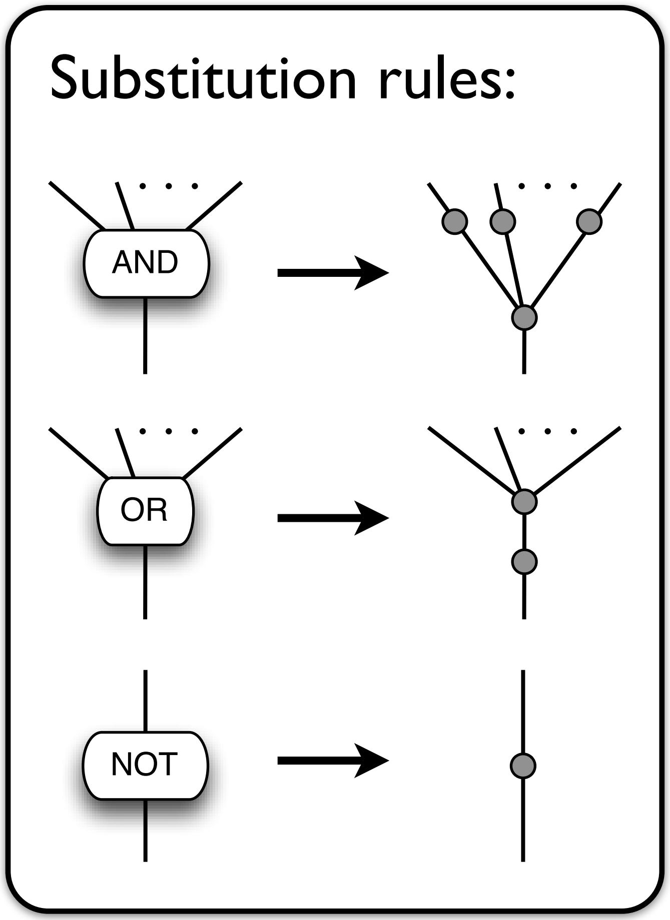
Convert formula φ into a graph G(φ) → Define classical random walk on G(φ) → Quantize that walk

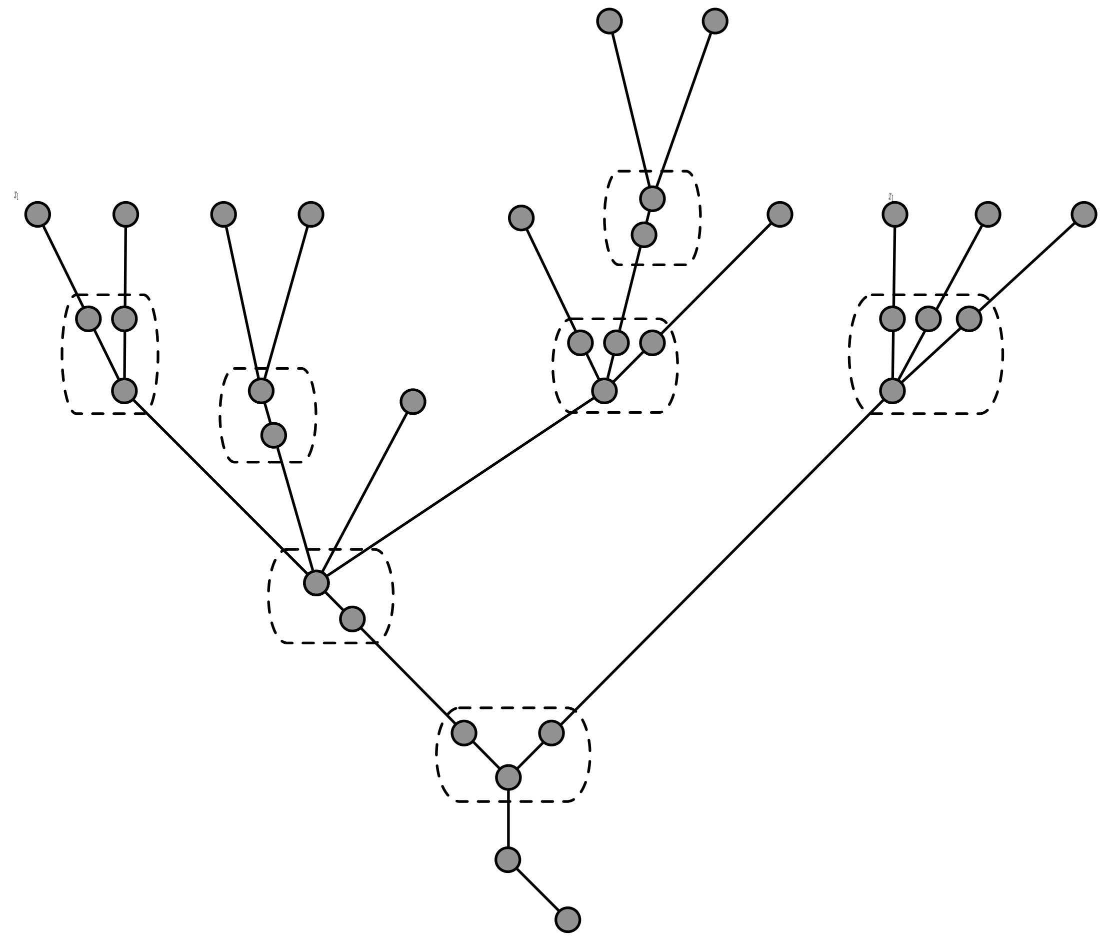
Substitution rules:



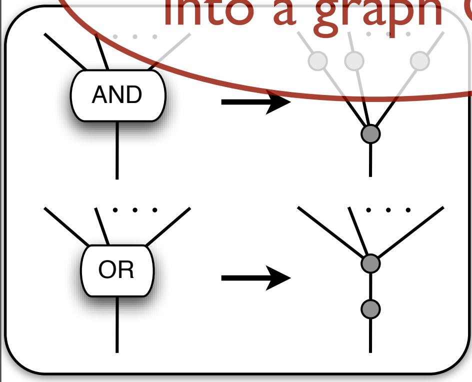


Convert formula φ into a graph G(φ) → Define classical random walk on G(φ) → Quantize that walk

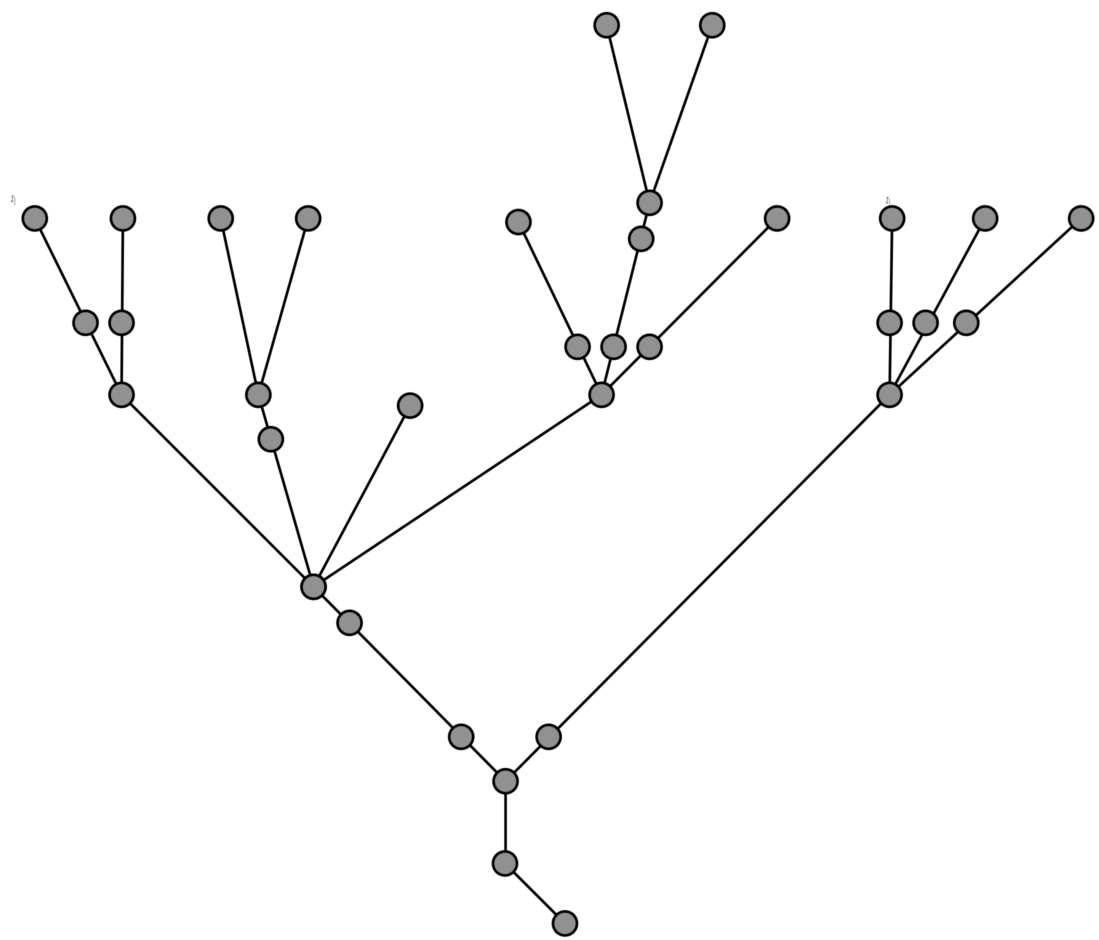




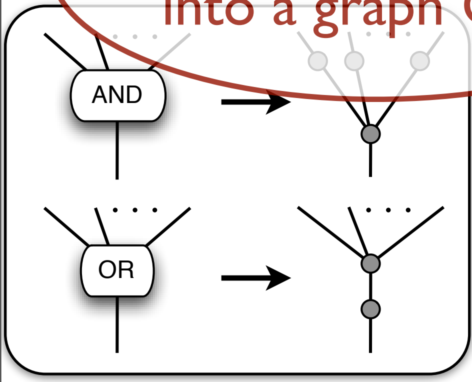
Convert formula φ into a graph G(φ) → Define classical random walk on G(φ) → Quantize that walk



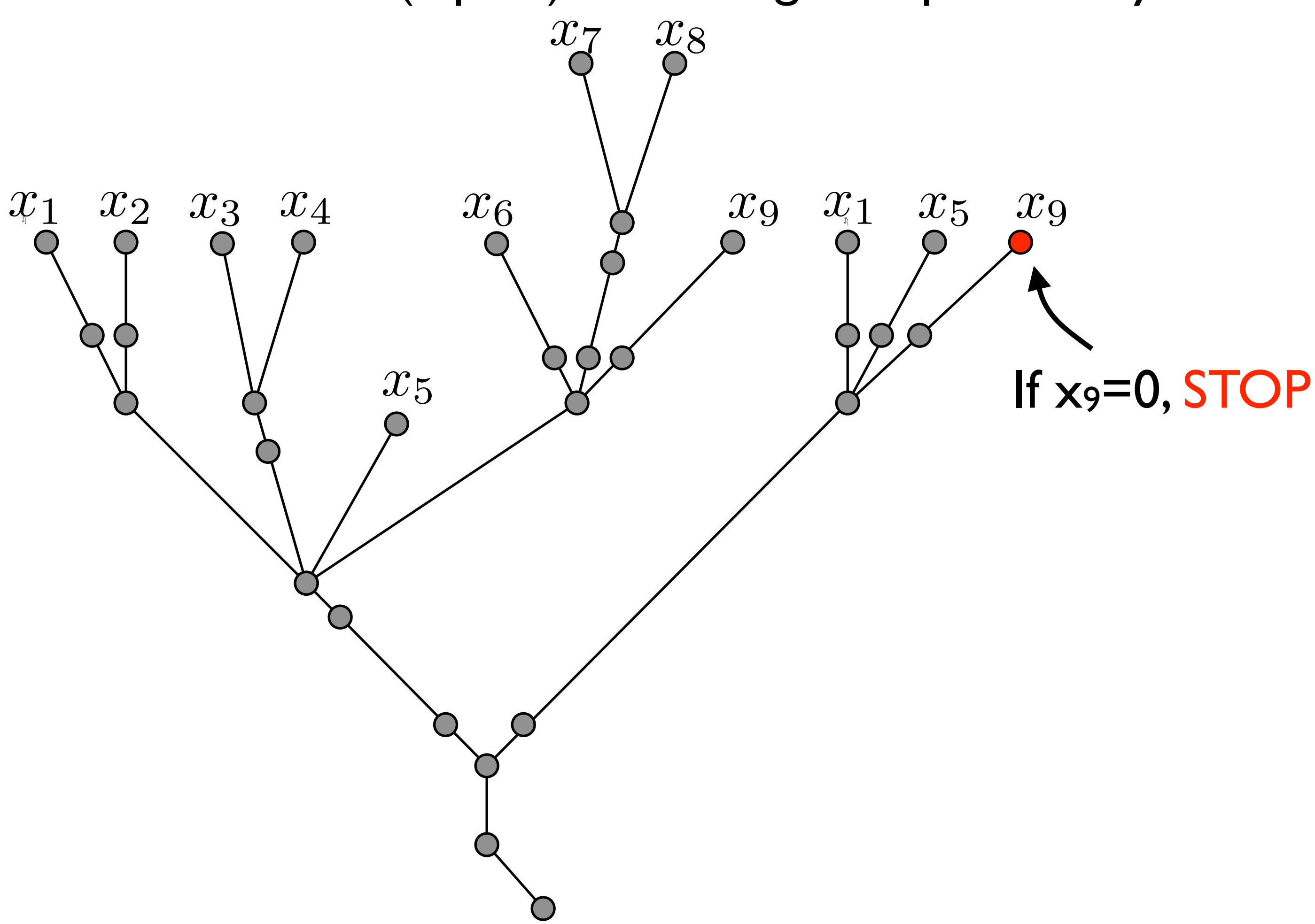
- P(stepping to subtree) ∝ √(size of that subtree)
  - (For a balanced tree, walk is uniform)



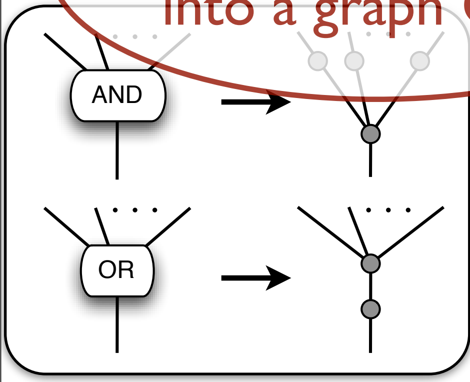
Convert formula φ into a graph G(φ) → Define classical random walk on G(φ) → Quantize that walk



- P(stepping to subtree) ∝ √(size of that subtree)
  - (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.

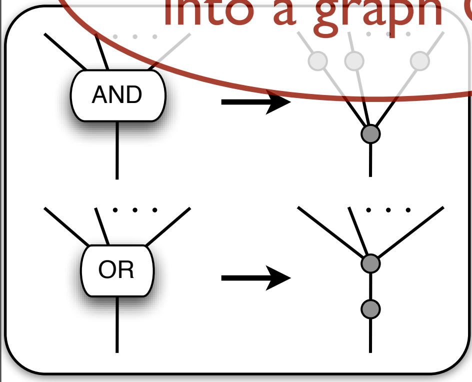
Transition probabilities

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

U = reflection about the state

$$\sqrt{p_1}|\text{⚀}\rangle + \sqrt{p_2}|\text{⚁}\rangle + \sqrt{p_3}|\text{⚂}\rangle + \sqrt{p_4}|\text{⚃}\rangle + \sqrt{p_5}|\text{⚄}\rangle + \sqrt{p_6}|\text{⚅}\rangle$$

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

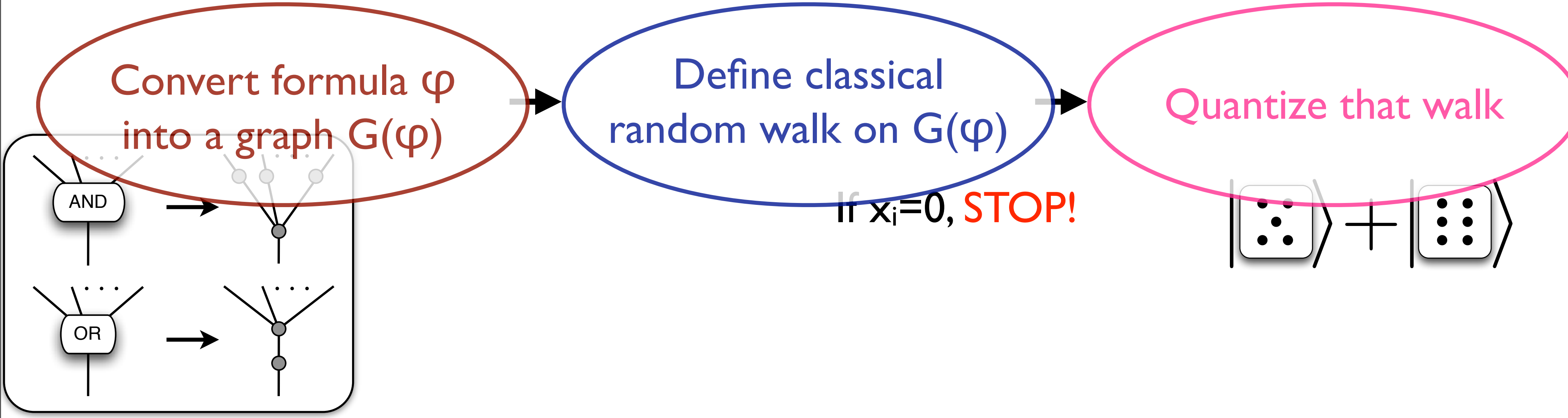
Transition probabilities

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

⟶

U = reflection about the state

$$\sqrt{p_1}\,|\text{⚀}\rangle + \sqrt{p_2}\,|\text{⚁}\rangle + \sqrt{p_3}\,|\text{⚂}\rangle + \sqrt{p_4}\,|\text{⚃}\rangle + \sqrt{p_5}\,|\text{⚄}\rangle + \sqrt{p_6}\,|\text{⚅}\rangle$$



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, output "φ(x)=1"
  - Otherwise output "φ(x)=0"

Formula evaluation algorithm

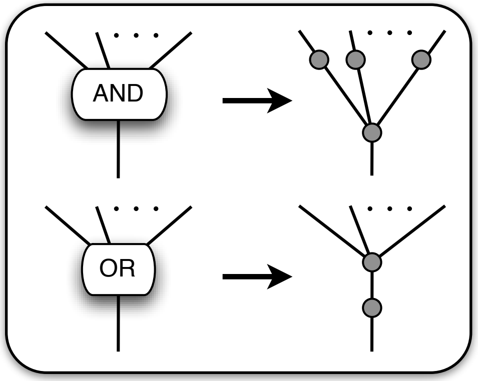
Convert formula φ into a graph G(φ)

→

Define classical random walk on G(φ)

→

Quantize that walk



P(stepping to subtree)

∝ √(size of that subtree)

If x_i=0, STOP!

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

↓

$$\sqrt{p_1}|\text{⚀}\rangle + \sqrt{p_2}|\text{⚁}\rangle + \sqrt{p_3}|\text{⚂}\rangle + \sqrt{p_4}|\text{⚃}\rangle + \sqrt{p_5}|\text{⚄}\rangle + \sqrt{p_6}|\text{⚅}\rangle$$

2. Why It Works

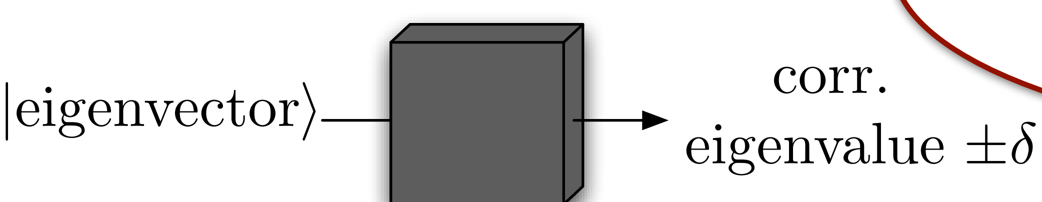
The Algorithm:

- 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 eigenvalue is 0, output "$\varphi(x)=1$"
  - Otherwise output "$\varphi(x)=0$"

Note:

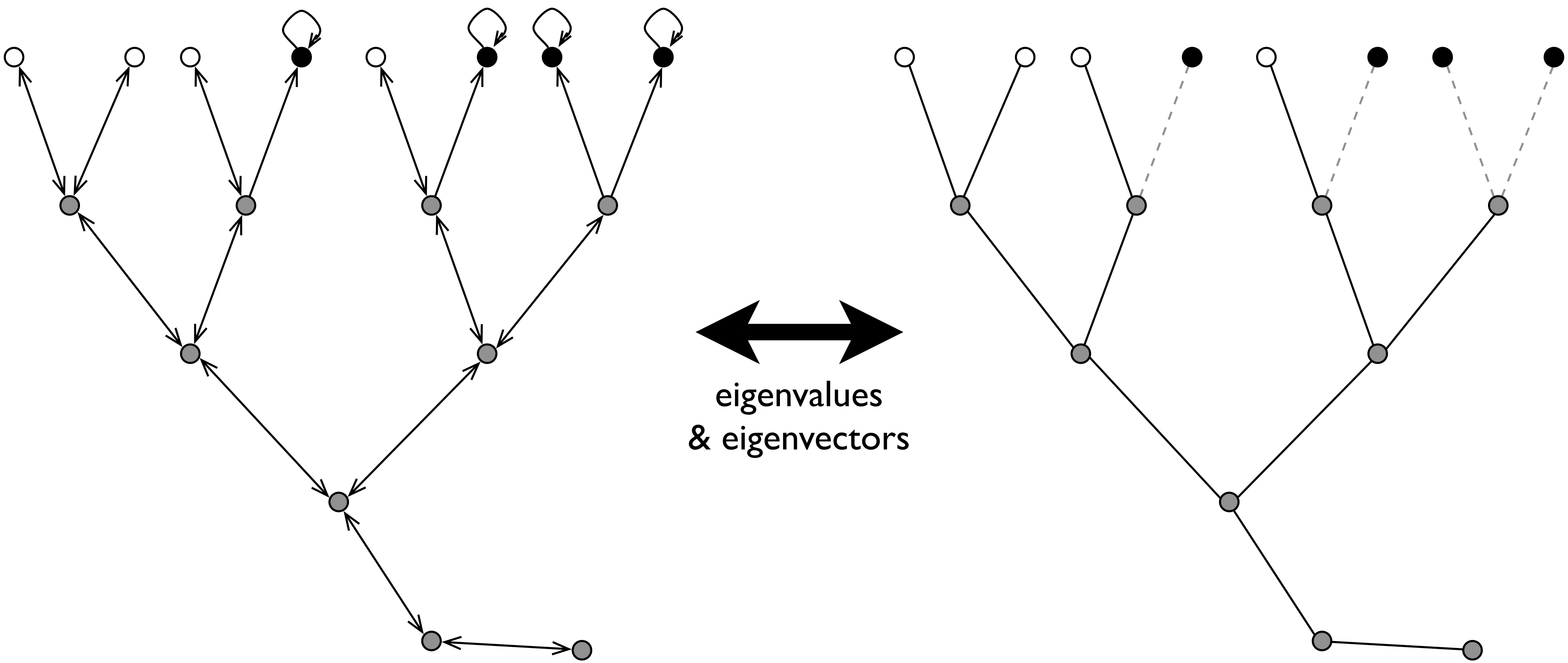
Precision-δ phase estimation on a unitary U, starting at an e-state, returns the e-value to precision δ, except w/ prob. 1/4. It uses $O(1/\delta)$ calls to c-U.

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



Szegedy eigenvalue and eigenvector correspondence [FOCS '04]

Quantum coined walk U(x): $\sqrt{P \circ P^T}$ W'ted adj. matrix $A_{G(x)}$ of G(x):



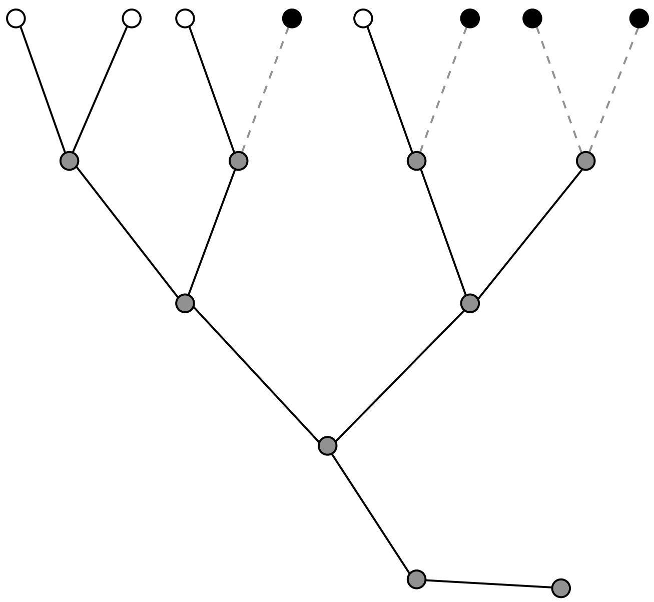
2|E| dimensions

|V| dimensions

○ =1

● =0

Note: Much like the [FGG] algorithm, edges to input vertices evaluating to 1 are deleted in G(x).



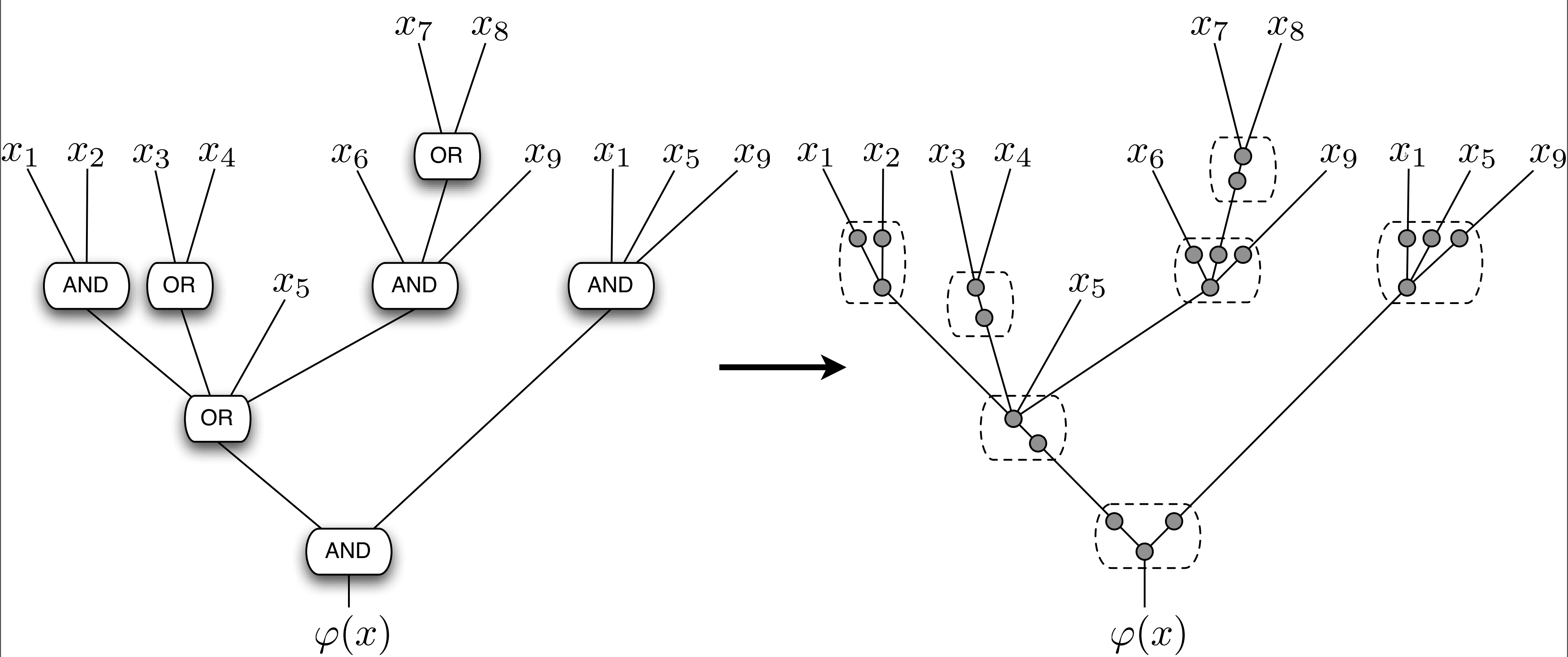
- **Main Theorem:**
  - $\varphi(x)=1 \Rightarrow A_{G(x)}$ has eigenvalue-0 e.v. with Ω(1) support on the root.
  - $\varphi(x)=0 \Rightarrow A_{G(x)}$ has no eigenvectors overlapping the root with $|\text{eigenvalue}|<2/\sqrt{N}$.

The Algorithm:

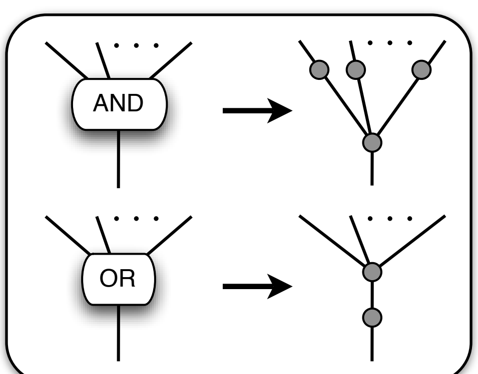
- Start at the root
- Apply phase estimation to the walk with precision $1/\sqrt{N}$
  - If e-value is 0, output "$\varphi(x)=1$"
  - Otherwise output "$\varphi(x)=0$"

∴ Algorithm is correct, except w/ error rate <1/4 (say)

- **Theorem:** $\varphi(x)=1 \iff \exists$ a $\lambda=0$ eigenstate of $A_{G(x)}$ supported on root r.



Proof: By induction, we argue that for every v, a vector α satisfying constraints for vertices above v must satisfy:



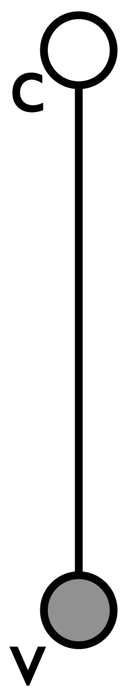
Induction hypothesis:

- $\varphi_v(x)=0 \Rightarrow \alpha_v=0$
- $\varphi_v(x)=1 \Rightarrow \alpha_v$ *can* be $\neq 0$

- **Induction hypothesis:**
  - $\varphi_v(x)=0 \Rightarrow \alpha_v=0$
  - $\varphi_v(x)=1 \Rightarrow \alpha_v$ *can* be $\neq 0$

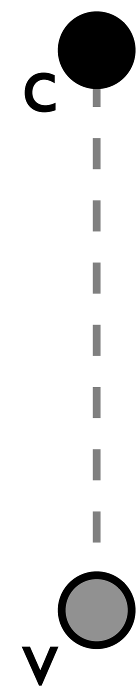
Base case: v an input

$x_i=0$:



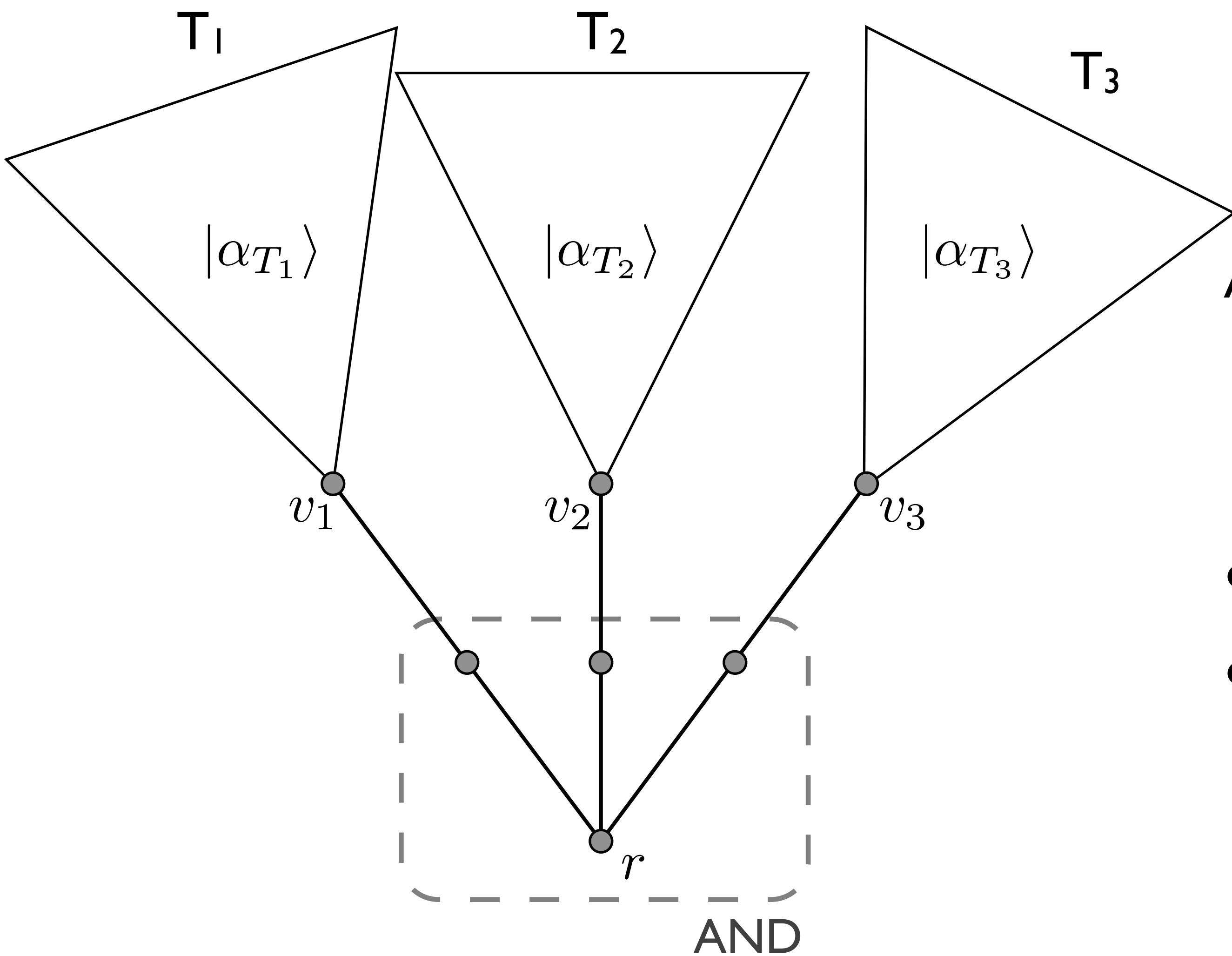
$\lambda=0$ eigenvector constraint at c is $\alpha_v=0$. ✓

$x_i=1$:



v and c are not connected in G(x), so α_v is not constrained.✓

- **Induction hypothesis:**
  - $\varphi_v(x)=0 \Rightarrow \alpha_v=0$
  - $\varphi_v(x)=1 \Rightarrow \alpha_v$ *can* be $\neq 0$



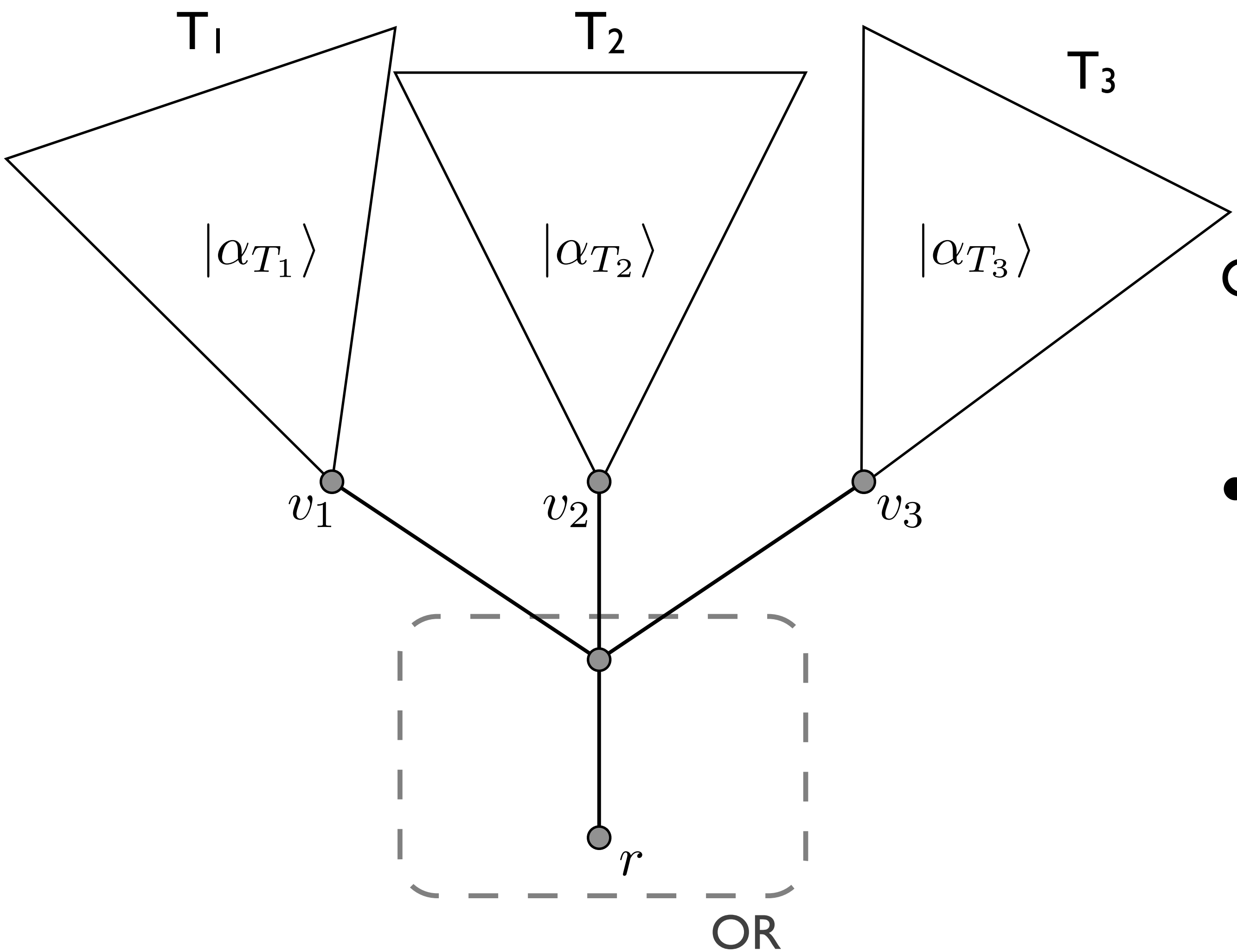
AND gate gadget constraints:

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

- 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 $\alpha_{v1}=\alpha_{v2}=\alpha_{v3}\neq 0$, then set $\alpha_r=-\alpha_{v_i}\neq 0$

✓ AND

- **Induction hypothesis:**
  - $\varphi_v(x)=0 \Rightarrow \alpha_v=0$
  - $\varphi_v(x)=1 \Rightarrow \alpha_v$ *can* be $\neq 0$



OR gate gadget constraint:

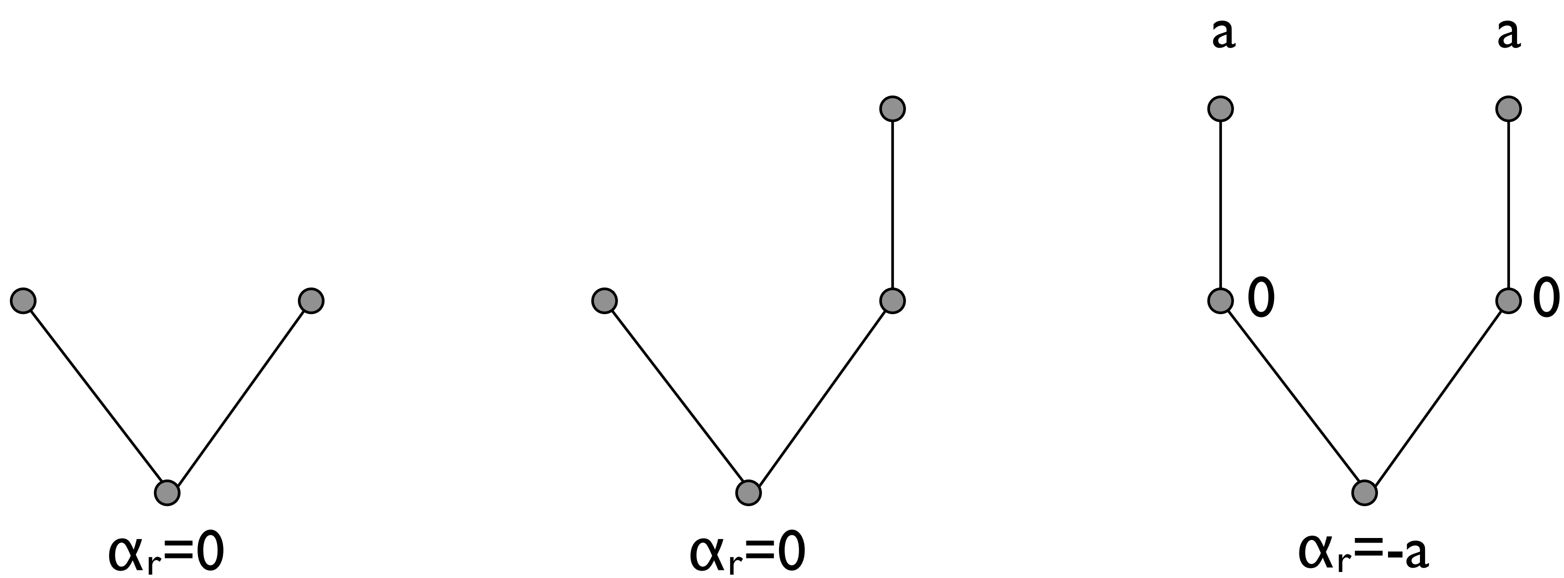
$$\alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3} + \alpha_r = 0$$

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

✓ OR

Just in case...

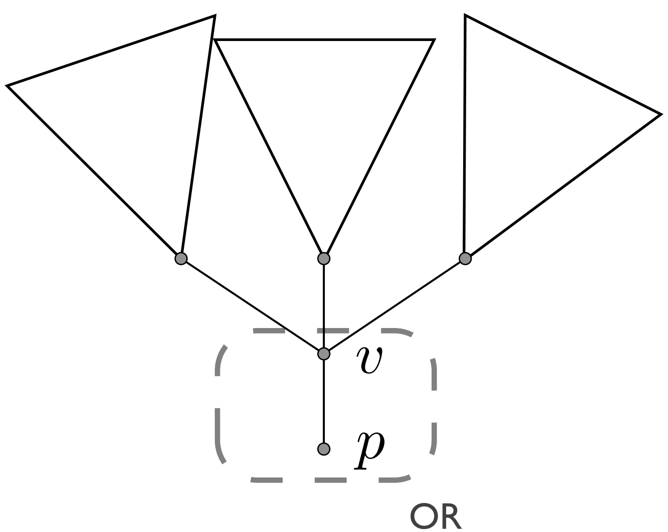
AND(0,0)=0 AND(0,1)=0 AND(1,1)=1



- **Theorem:** φ(x)=1 $\Longleftrightarrow$ ∃ a λ=0 eigenstate of $A_{G(x)}$ supported on root r. ✓

- **Main Theorem:**
  - φ(x)=1 $\Longrightarrow$ $A_{G(x)}$ has eigenvalue-0 e.v. with Ω(1) support on the root.
  - φ(x)=0 $\Longrightarrow$ $A_{G(x)}$ has no eigenvectors overlapping the root with |eigenvalue|<1/√N.

- Remains to show support α_r is large (Ω(1)) when φ(r)=0, and that there is a large spectral gap (1/√N) away from E=0 when φ(r)=1.
- Proofs by same induction but quantitative.

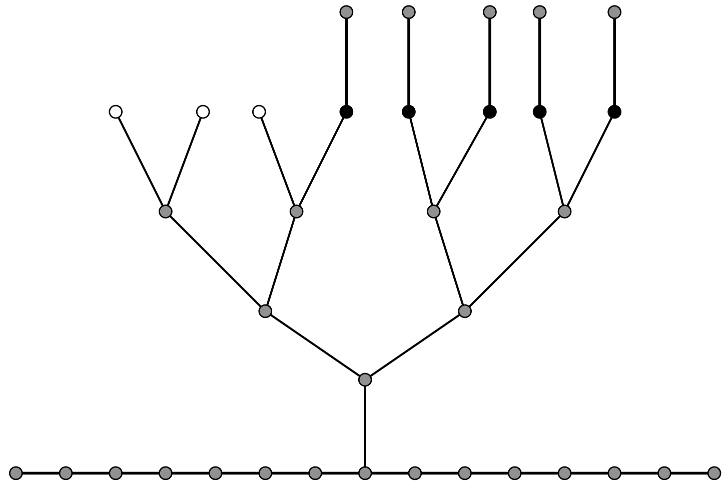


$$\frac{\alpha_p}{\alpha_v} \in (0, s_v \lambda) \quad \text{if true}$$

$$-\frac{\alpha_v}{\alpha_p} \in (0, s_v \lambda) \quad \text{if false}$$

with $s_v = \sqrt{s_{v_1}^2 + \cdots + s_{v_3}^2} = \sqrt{\mathrm{size}(\varphi_v)}$

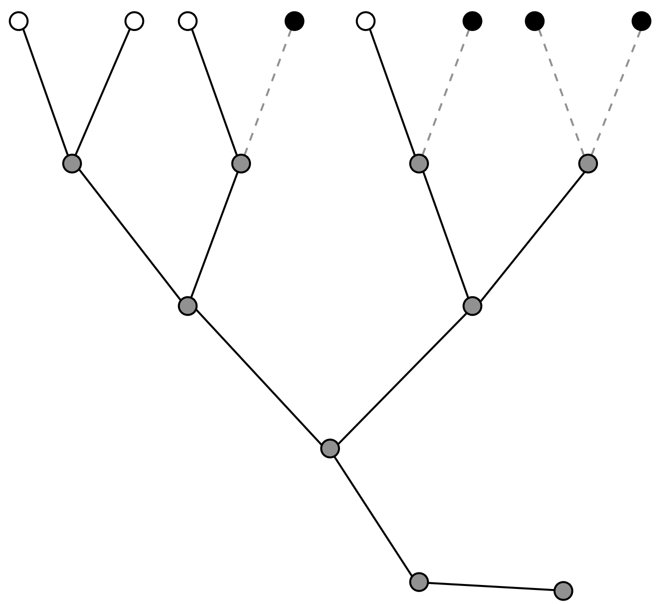
[FGG '07] algorithm



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

  Analysis by scattering theory.

[ACRŠZ '07] algorithm



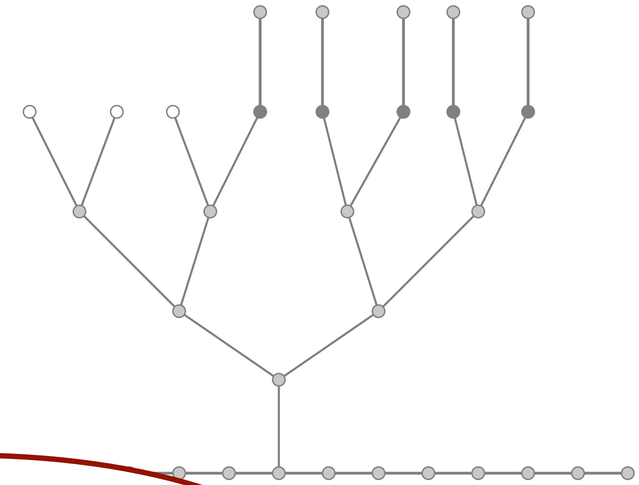
- **Theorem**:
  - An "approximately balanced" AND-OR formula can be evaluated with $O(\sqrt{N})$ queries (optimal for read-once!).
  - A general AND-OR formula can be evaluated with $N^{1/2+o(1)}$ queries.

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

[FGG '07] algorithm

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

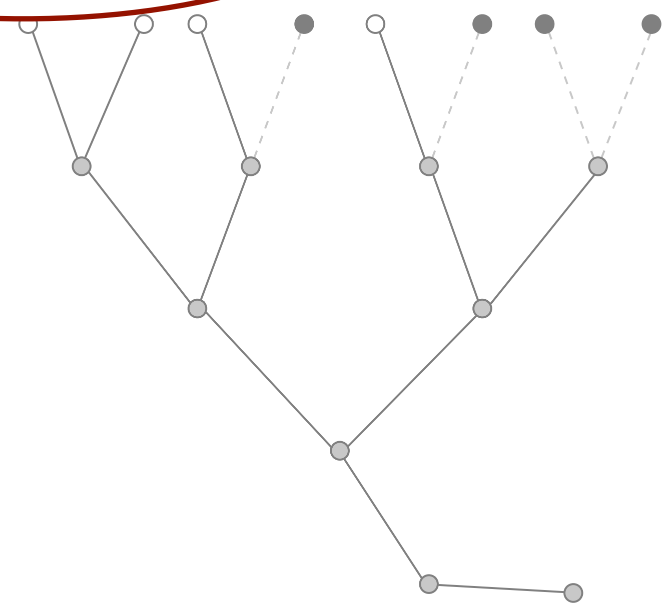
  Analysis by scattering theory.



Where do o(1) terms come from?

[ACRŠZ '07] algorithm

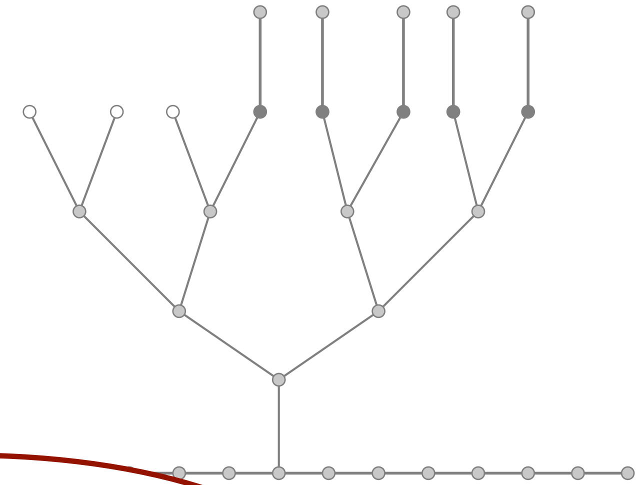
- **Theorem**:
  - An "approximately balanced" AND-OR formula can be evaluated with $O(\sqrt{N})$ queries (optimal for read-once!).
  - A general AND-OR formula can be evaluated with $N^{\frac{1}{2}+o(1)}$ queries.



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

[FGG '07] algorithm

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



Analysis by scattering theory.

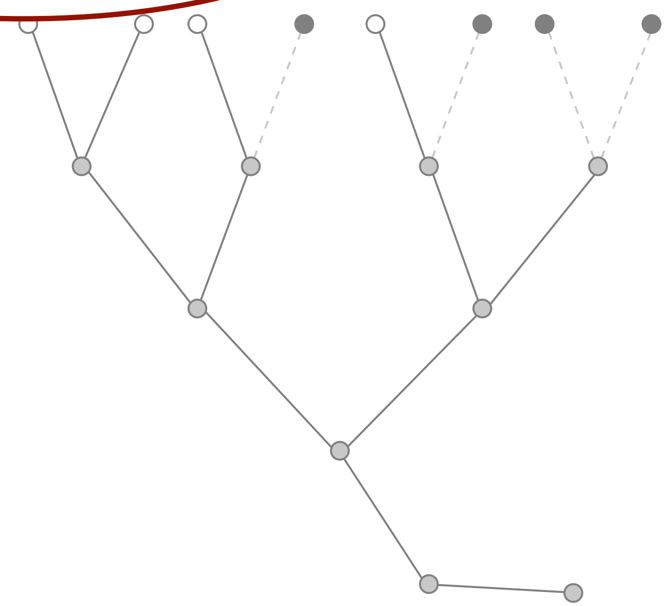
Where do o(1) terms come from?

Fixed, by working with coined quantum walks (via Szegedy corr.) instead of continuous-time qu. walks

[ACRSZ '07] algorithm

Theorem:

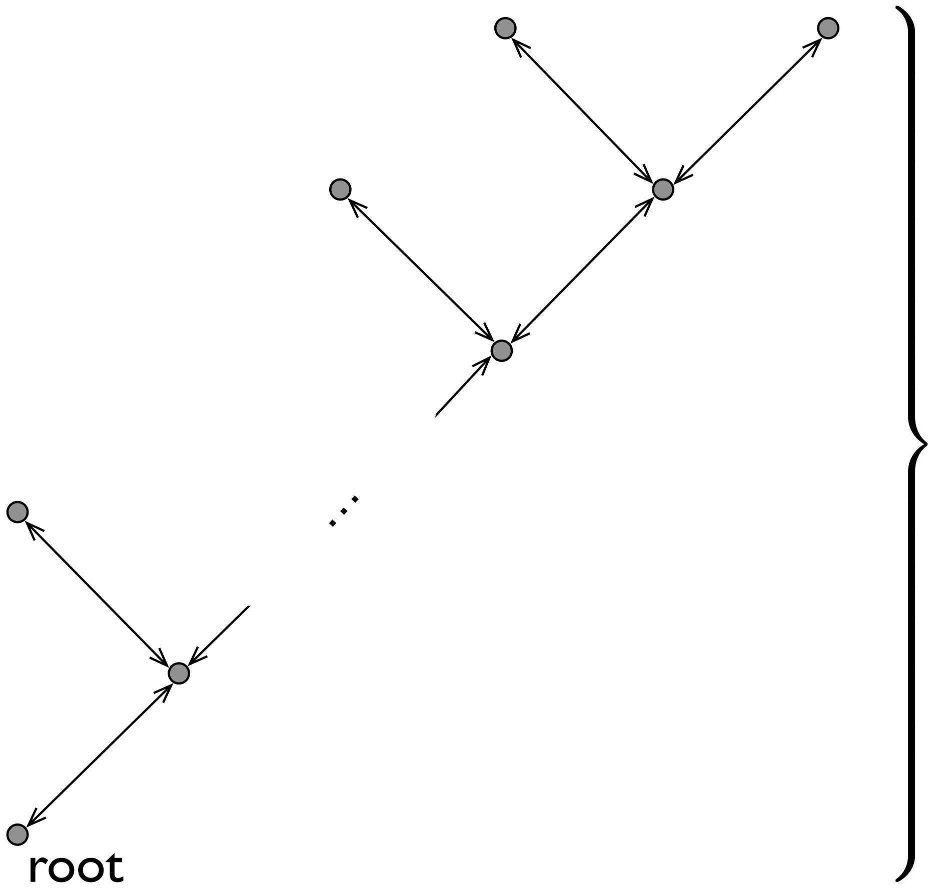
- An "approximately balanced" AND-OR formula can be evaluated with $O(\sqrt{N})$ queries (optimal for read-once!).
- A general AND-OR formula can be evaluated with $N^{1/2+o(1)}$ queries.



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

Algorithm for very unbalanced trees

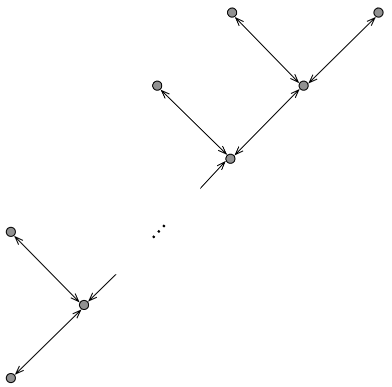
- Problem: We lose control of recursion fudge factors in a very deep formula.
- Intuition: Walk from root will not even *reach* the farthest leaves in time √N.



E.g., if depth is N, then gap could be only 1/N

Algorithm for very unbalanced trees

- Problem: Walk might not even reach the bottom of a deep formula in time $\sqrt{N}$



- Solution: Rebalance the formula tree (in preprocessing)

  Theorem: ([Bshouty, Cleve, Eberly '91, Bonet & Buss '94]) For any NAND formula φ and k ≥ 2, can efficiently construct an equivalent NAND formula φ' with

  - depth(φ') = O(k log N)
  - size(φ') ≤ $N^{1+1/\log k}$

  size-depth tradeoff (set $k=2^{\sqrt{(\log N)}}$ to balance size*depth)

- 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') ≥ $N^{1+1/\log k}$?
- Open: What is the effect of general formula rebalancing on the ADV bound?

Remarks on formula evaluation algorithms:

Classical vs. Quantum

- Classical complexity of evaluating balanced k-ary alternating AND-OR tree is $(k/2)^{\text{depth}} = N^{\sim(1-1/\log_2 k)}$ — 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 $\Omega\left(\left(7/3\right)^d\right)$ and $o\left(\left(8/3\right)^d\right)$
  - (the generalization of the optimal AND-OR algorithm is *not* optimal when applied to MAJ_3 trees)

[Jayram, Kumar, Sivakumar '03]

- Quantumly, complexity is $\sqrt{N}$ 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
- Expanding MAJ_3 into AND-OR gates gives $O(\sqrt{5^d})$ quantumly.
- Also, the algorithm *generalizes* to give optimal algorithm for evaluating iterated f, where f is any 3-bit function

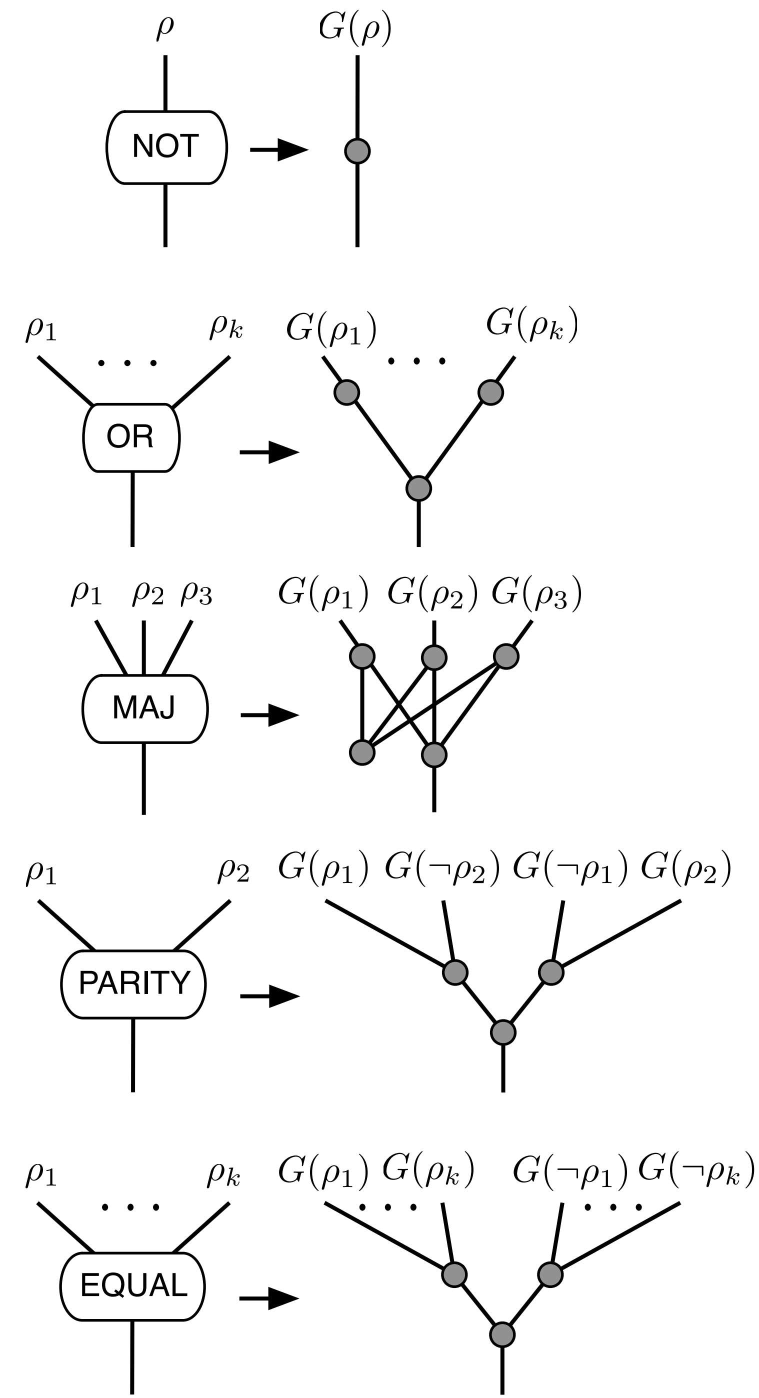
Span-program-based quantum algorithm for formula evaluation

Ben Reichardt
Caltech

Robert Špalek
Google

[quant-ph/0710.2630]

We present a time-efficient and query-optimal quantum algorithm for evaluating adversary-bound-balanced formulas on an extended gate set. The allowed gates include arbitrary two- and three-bit gates, as well as bounded fan-in AND, OR, PARITY and EQUAL gates. The technique behind the formula evaluation algorithm is a new framework for quantum algorithms based on span programs. For example, the classical complexity of evaluating the balanced ternary majority formula is unknown, and the natural generalization of the standard balanced AND-OR formula evaluation algorithm is known to be suboptimal. In contrast, a generalization of the optimal quantum {AND, OR, NOT} formula evaluation algorithm is optimal for evaluating the balanced ternary majority formula.



span programs [Karchmer, Wigderson '93],...

Classical learning theory:

Corollary: AND-OR formulas of size N are (classically) PAC-learnable in time $2^{N^{1/2+o(1)}}$. [O'Donnell & Servedio '03]

Open problems

- Is the phase estimation needed, or can the walk be run directly?
- Is the eigenstate useful as a witness?
- 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') ≥ $N^{1+1/\log k}$?
  - Effect of rebalancing on the adversary lower bound
- Optimal algorithm for more formula types, more span-program-based quantum algorithms; see [quant-ph/0710.2630]

(and many more...)

link