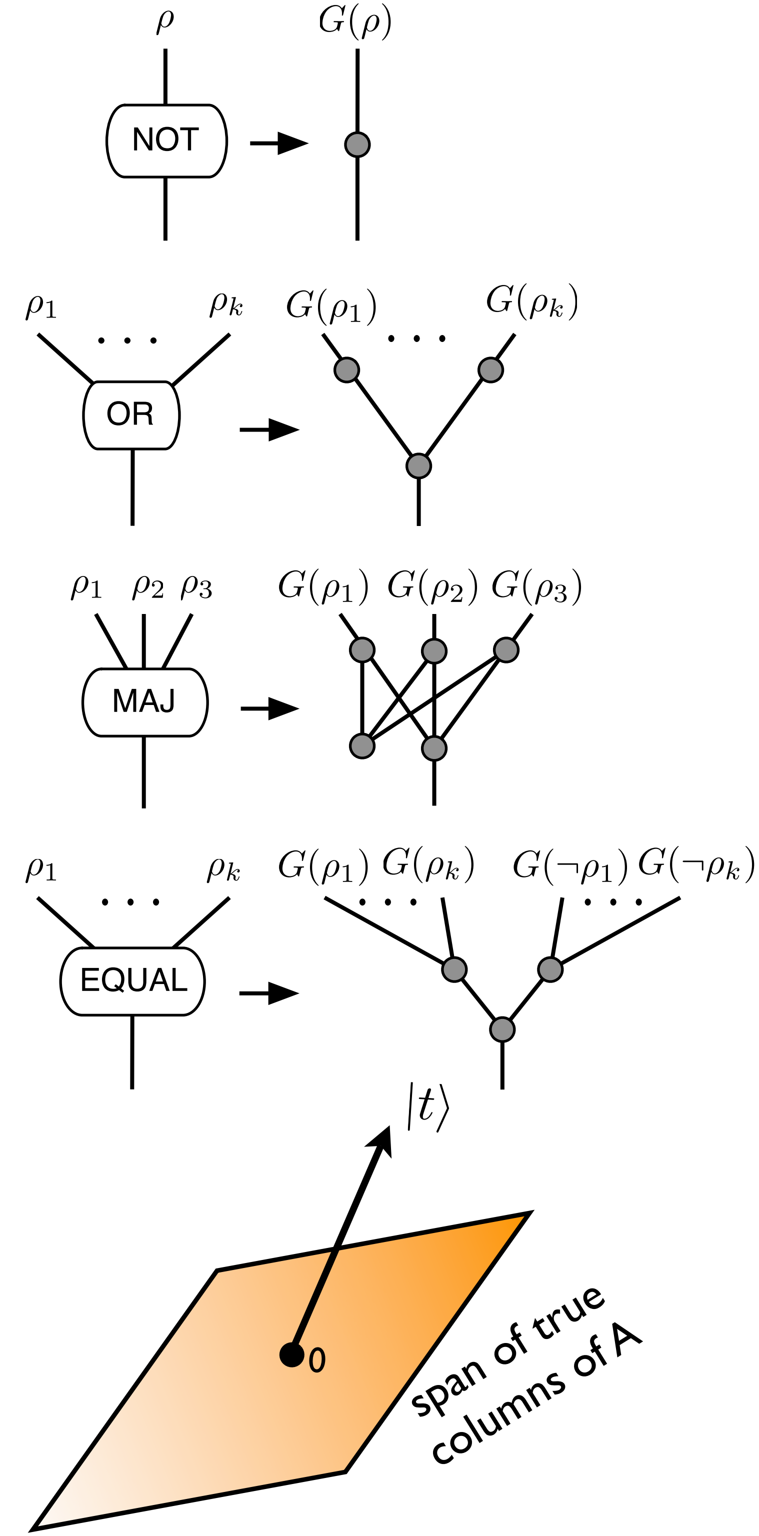
Span-program-based quantum algorithm for formula evaluation

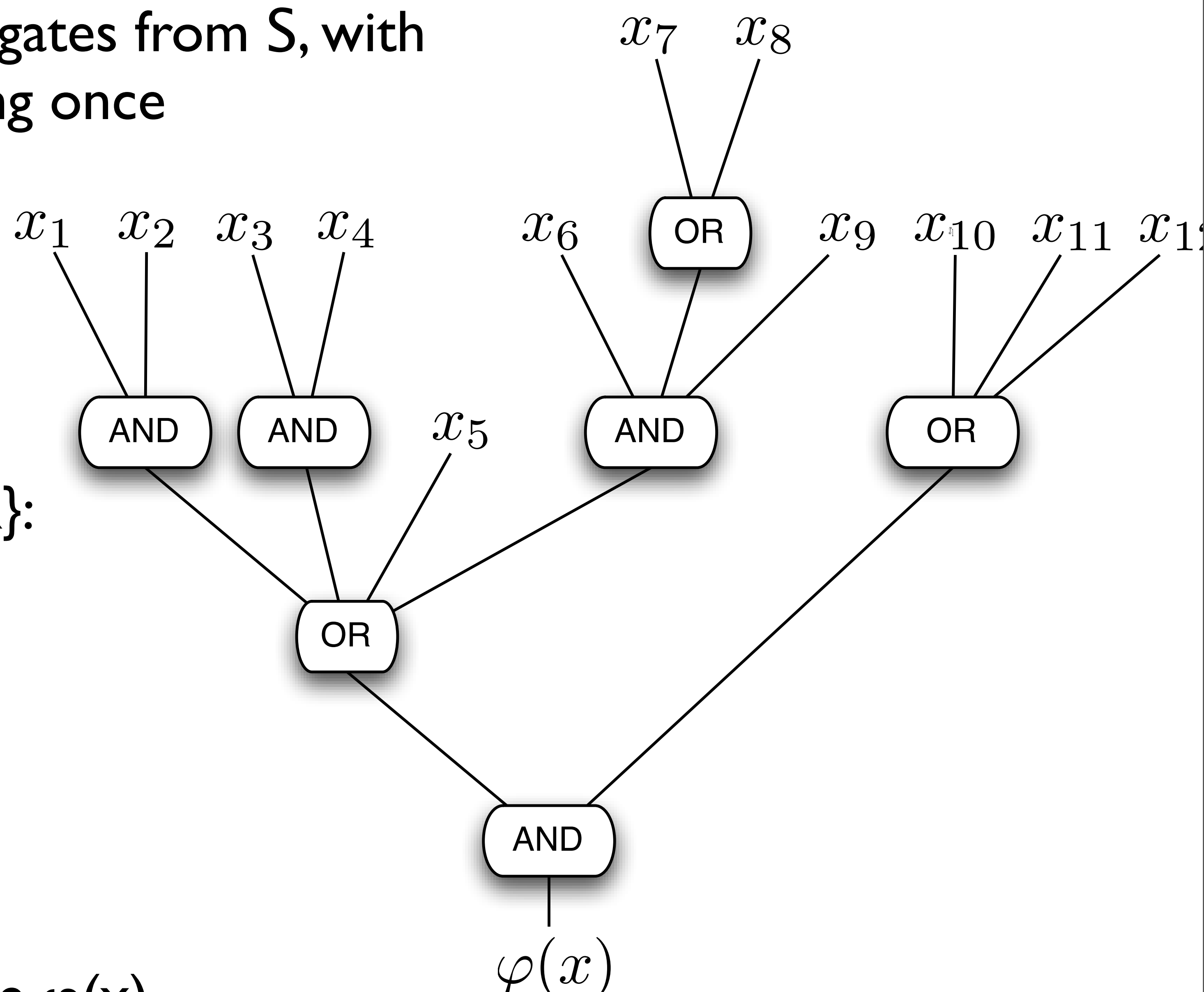
Ben Reichardt
Caltech

Robert Špalek
Google



Def: Read-once formula φ on gate set S
= Tree of nested gates from S, with each input appearing once

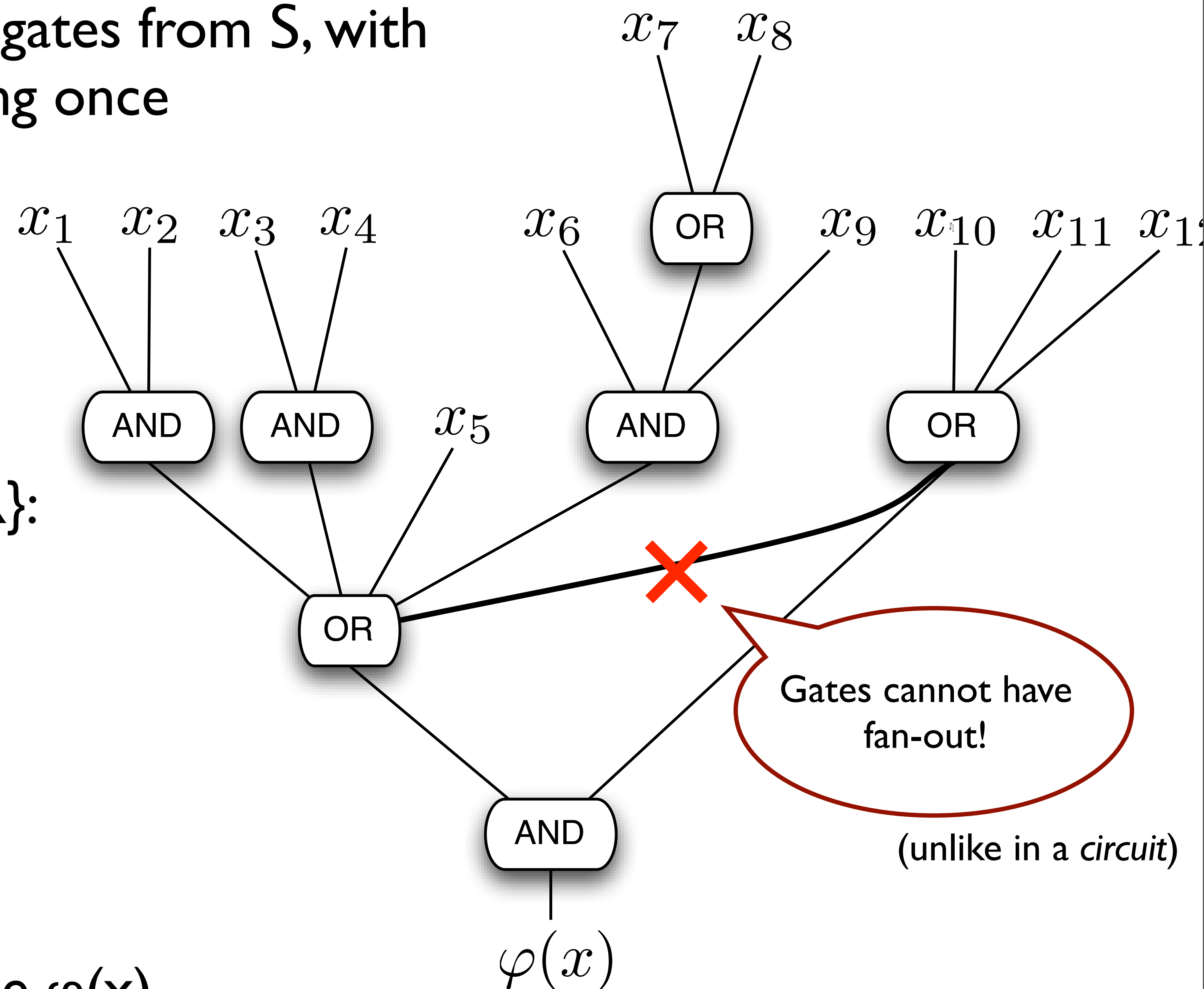
Ex: S = {AND, OR}:



Problem: Evaluate φ(x).

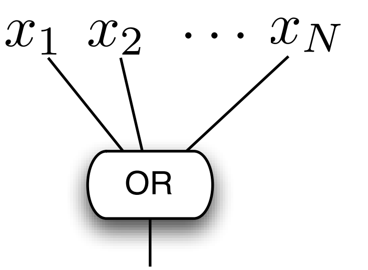
Def: Read-once formula φ on gate set S
= Tree of nested gates from S, with each input appearing once

Ex: S = {AND, OR}:



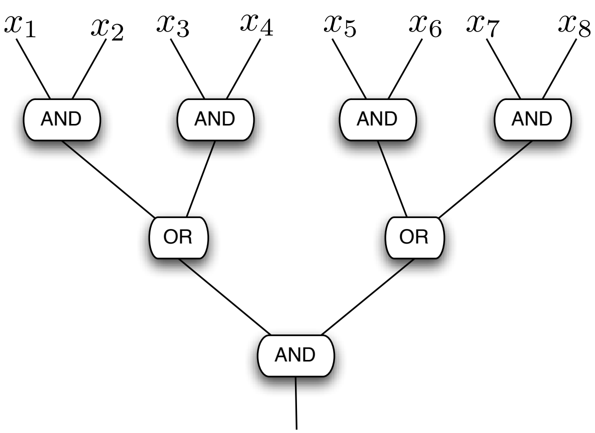
Problem: Evaluate φ(x).

Classical complexity of formula evaluation



$\Theta(N)$

Balanced AND-OR



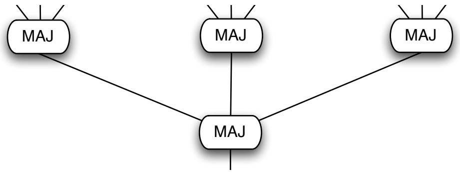
$\Theta(N^{0.753\ldots})$

(fan-in two)

[Snir '85, Saks & Wigderson '86, Santha '95]

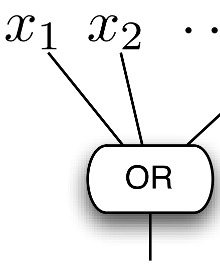
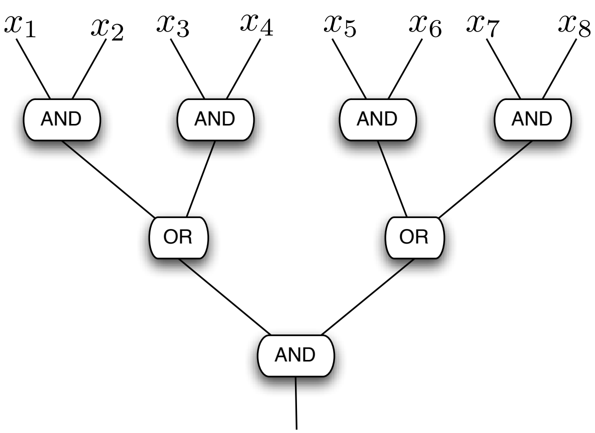
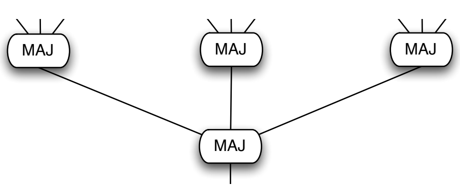
$\vdots$

Balanced MAJ_3

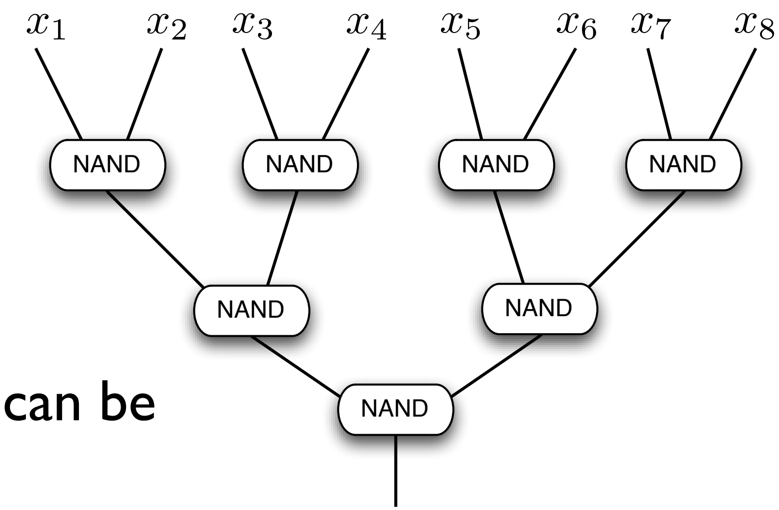


$\Omega((7/3)^{\text{depth}}) = R_2(f) = O((2.6537\ldots)^{\text{depth}})$

[Jayram, Kumar, Sivakumar '03]

	Classical	**Quantum**
	$\Theta(N)$	$\Theta(\sqrt{N})$ [Grover '96]
Balanced AND-OR 	$\Theta(N^{0.753\ldots})$ (fan-in two) [S'85, SW'86, S'95]	$\Theta(\sqrt{N})$ [Farhi & Goldstone & Gutmann '07, ACRŠZ '07]
⋮ Balanced MAJ$_3$ 	$\Omega((7/3)^d)$, $O((2.6537\ldots)^d)$ [JKS '03]	$\Theta(2^d = N^{\log_3 2})$ [RŠ '07]

Two generalizations of [FGG '07] AND-OR algorithm:



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

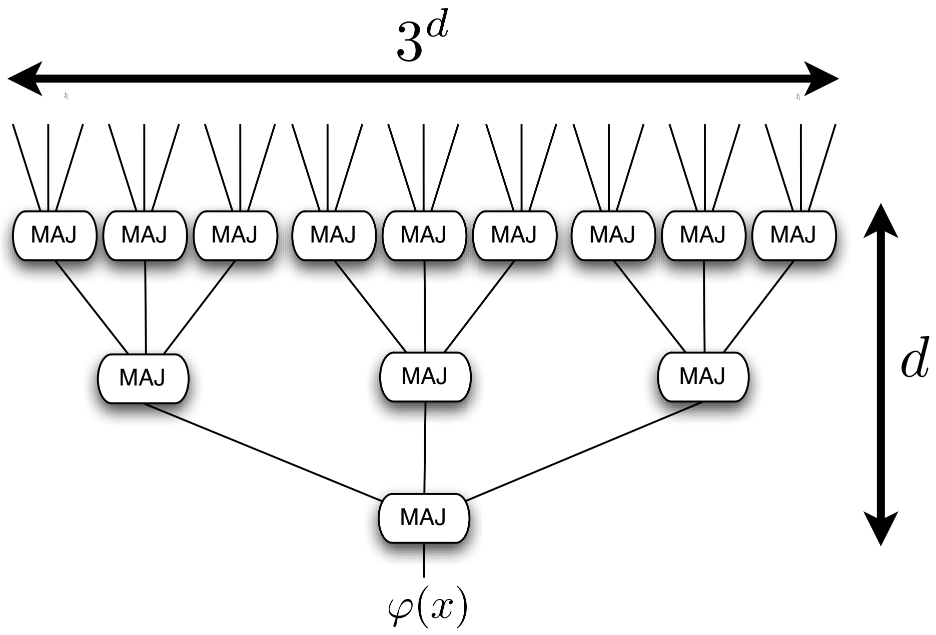
Unbalanced AND-OR [ACRŠZ, FOCS'07]

- **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 $N^{½+o(1)}$ queries.

Balanced, more gates [RŠ, STOC'08]

- **Theorem:** A balanced ("adversary-bound-balanced") formula φ over a gate set including all three-bit gates (and more…) can be evaluated in O(ADV(φ)) queries (optimal!).

Recursive 3-bit majority tree



- Best quantum lower bound is
  $\Omega(\mathrm{ADV}(\varphi) = 2^d)$ [LLS'05]

- Expand majority into {AND, OR} gates:
  $$\mathrm{MAJ}_3(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee (x_3 \wedge (x_1 \vee x_2))$$

∴ {AND, OR} formula size is $\leq 5^d$

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

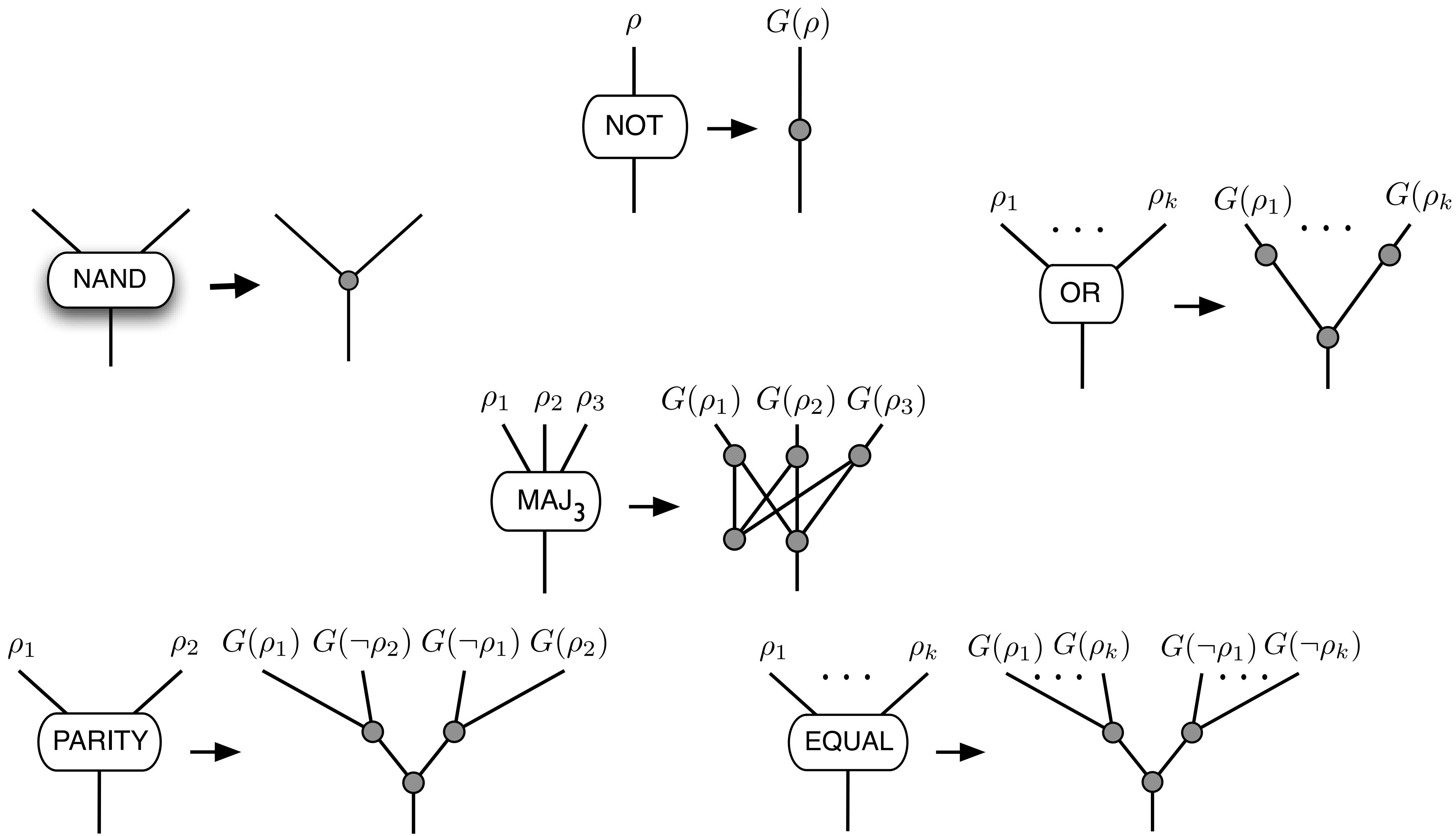
[FGG, ACRŠZ '07]

[RŠ '07] algorithm

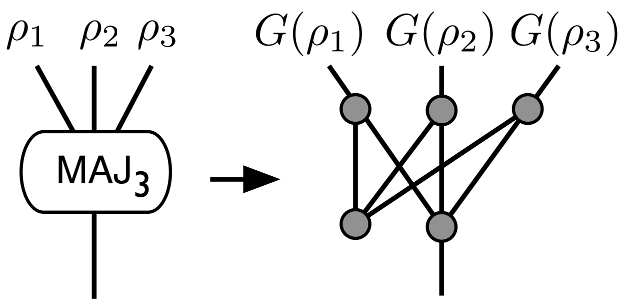
- **Theorem:** A balanced ("adversary-bound-balanced") formula φ over a gate set including all three-bit gates (and more…) can be evaluated in O(ADV(φ)) queries (optimal!).

- New: $O(2^d)$-query quantum algorithm

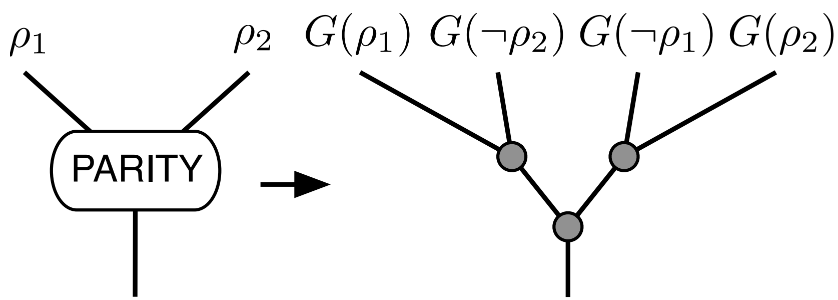
Converting formula into a tree

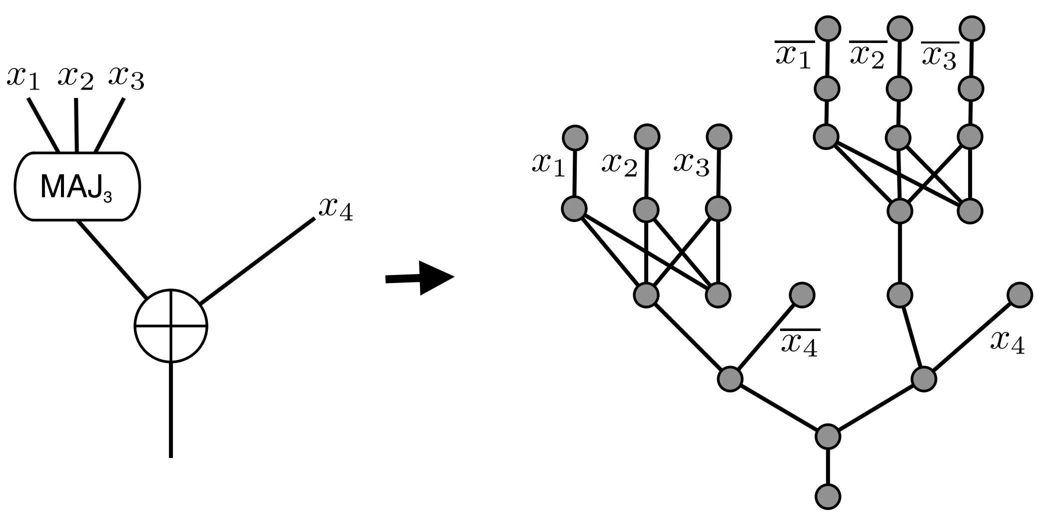


(with appropriate edge weights)



- **Main Theorem:**
  - $\varphi(x)=1 \Rightarrow A_G$ has $\lambda=0$ eigenvector with $\Omega(1)$ support on the root.
  - $\varphi(x)=0 \Rightarrow A_G$ has no eigenvectors overlapping the root with $|\lambda|<1/O(\text{ADV}(\varphi))$.





- **Main Theorem:**
  - $\varphi(x)=1 \Rightarrow A_G$ has $\lambda=0$ eigenvector with $\Omega(1)$ support on the root.
  - $\varphi(x)=0 \Rightarrow A_G$ has no eigenvectors overlapping the root with $|\lambda|<1/O(ADV(\varphi))$.

quantitative bounds needed to analyze the running time

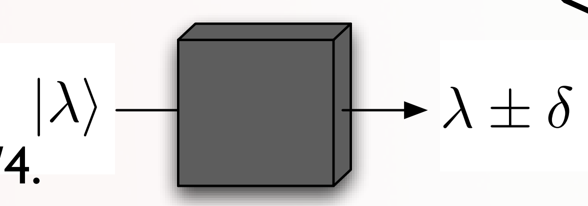
- **Main Theorem:**
  - $\varphi(x)=1 \Rightarrow A_G$ has $\lambda=0$ eigenvector with $\Omega(1)$ support on the root.
  - $\varphi(x)=0 \Rightarrow A_G$ has no eigenvectors overlapping the root with $|\lambda|<1/O(\mathrm{ADV}(\varphi))$.

Fast Quantum Algorithm:

- Start at the root
- Apply phase estimation to the quantum walk with precision $1/O(\mathrm{ADV}(\varphi))$
- If measured phase is 0, output "$\varphi(x)=1$." Otherwise, output "$\varphi(x)=0$."

Running time is $O(\mathrm{ADV}(\varphi))$

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.



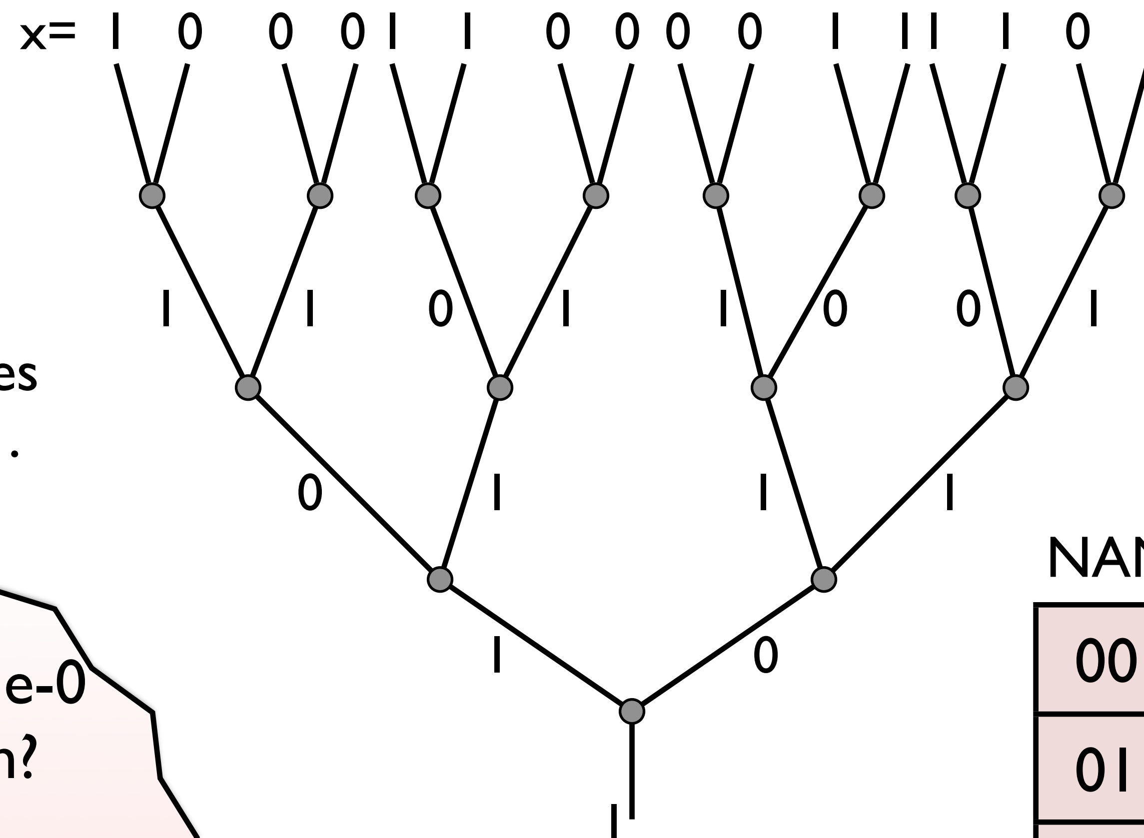
Computation of formula ⟺ Eigenvalue-zero eigenvector of tree

Input dependence

- Substitutions define $G(0^N)$; to define graph $G(x)$, delete edges to all leaves evaluating to $x_i=1$.

Q: What is an eigenvalue-0 eigenvector of a graph?

A: Assignment of coefficients to each vertex, such that sum of neighboring coefficients adds up to 0.



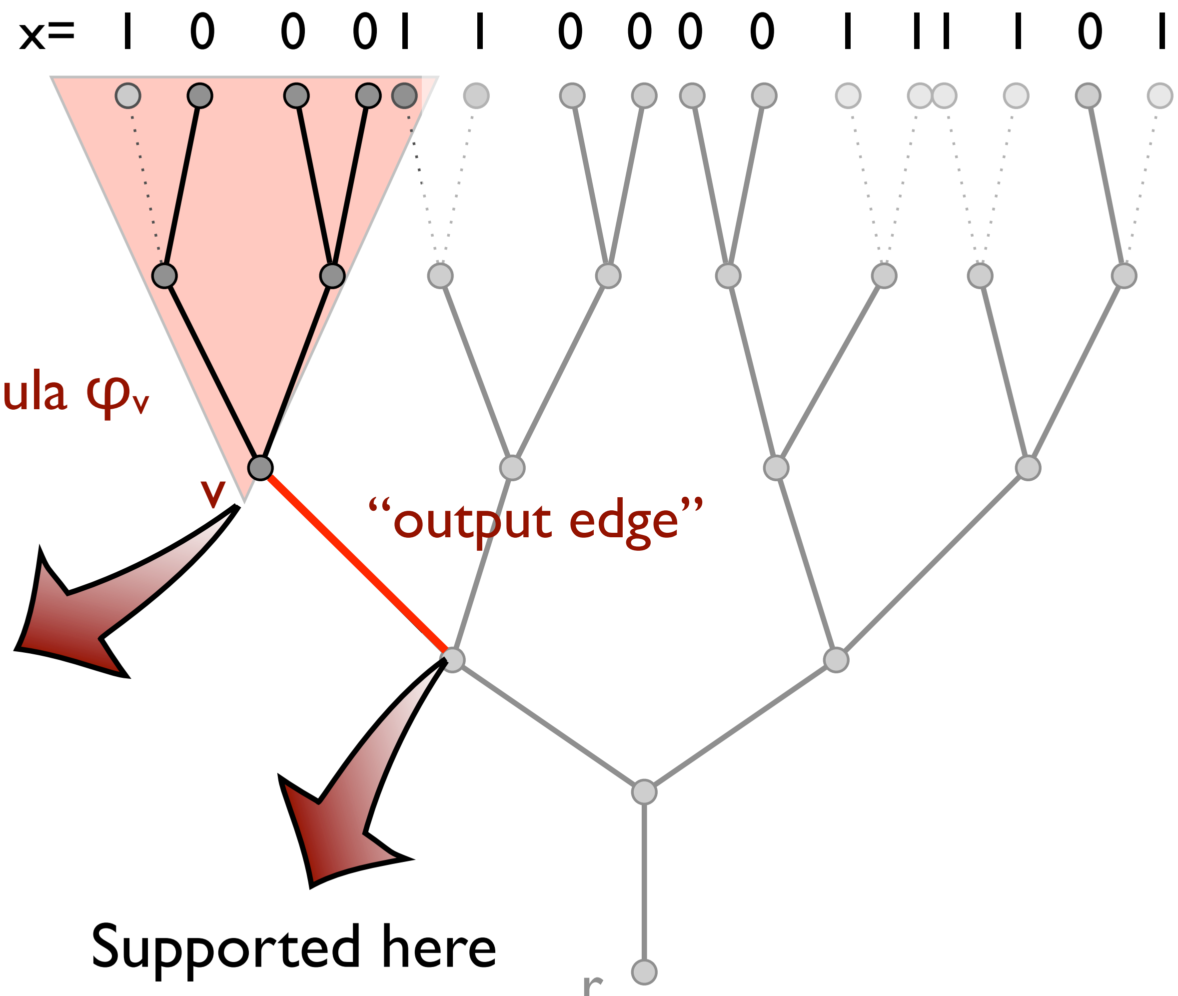
NAND

00	1
01	1
10	1
11	0

(each edge labeled by the evaluation of the NAND sub-formula above it)

Computation of formula ⇔ Eigenvalue-zero eigenvector of tree

Induction Claim: Each edge gives a "dual-rail" encoding for the evaluation of the sub-formula above that edge…

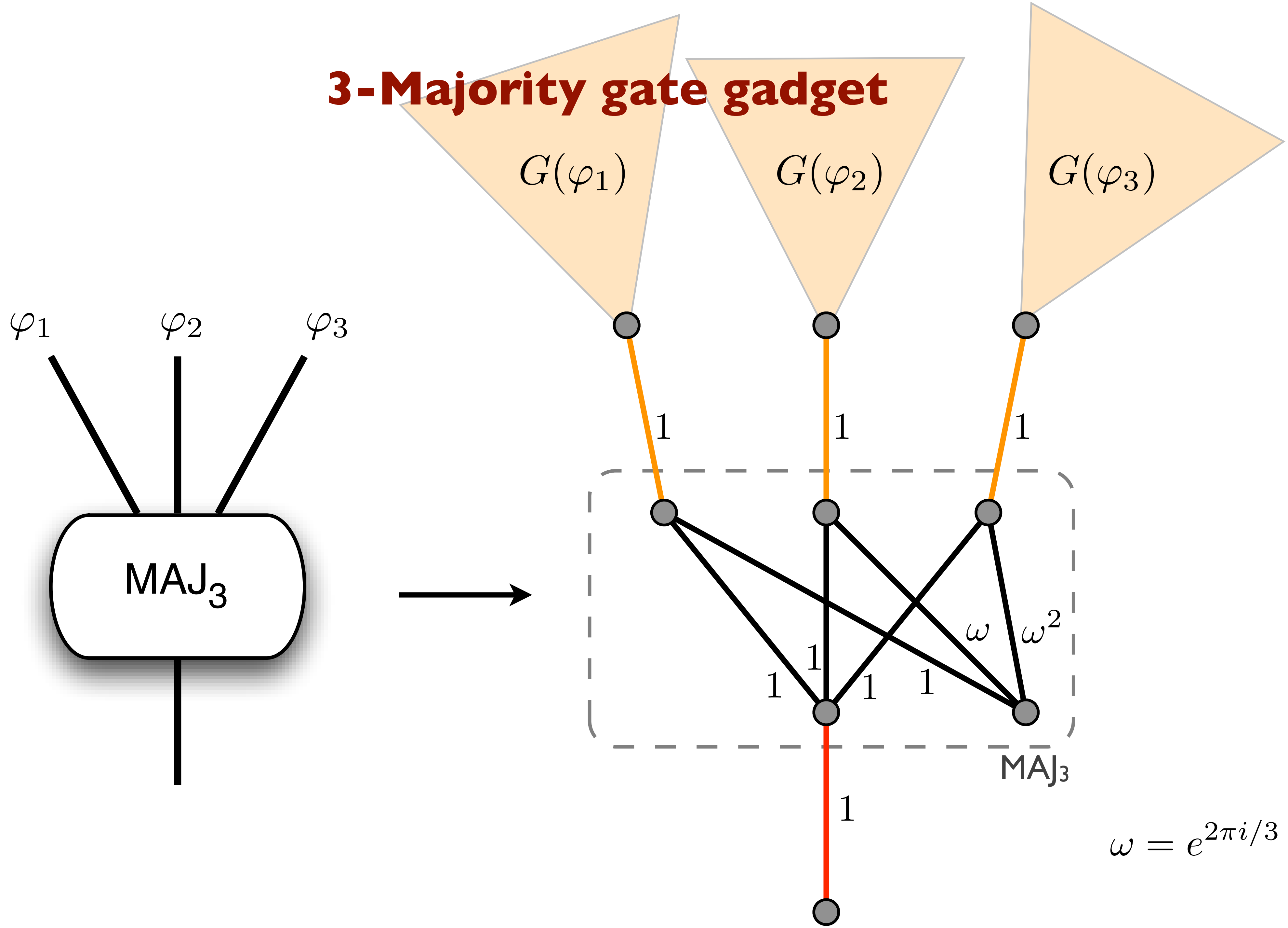


The $\lambda=0$ eigenvector of $G(\varphi_v, x)$ is:

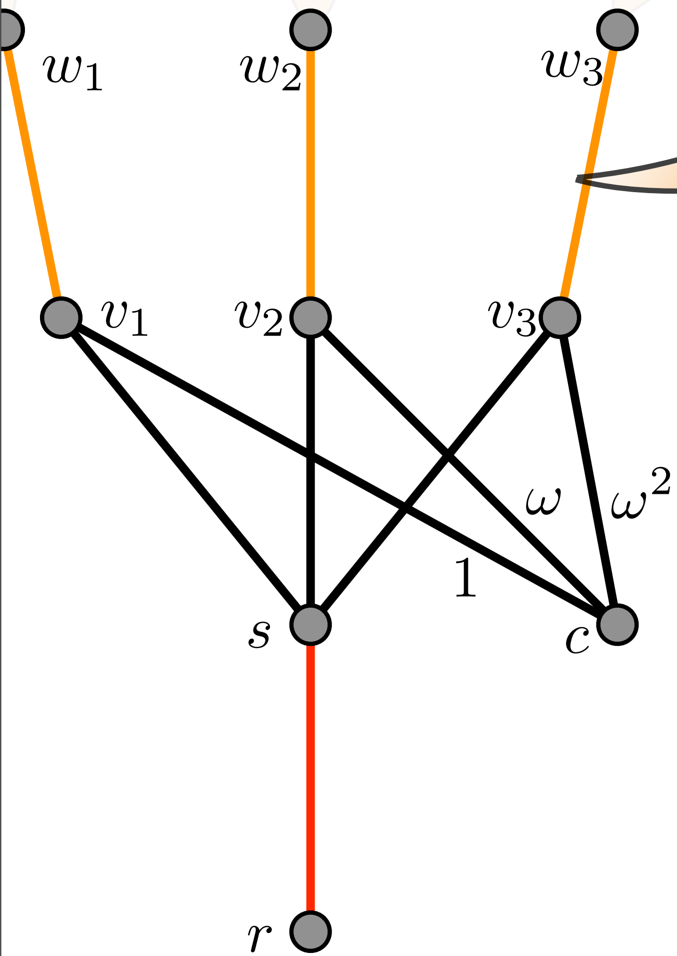
Supported here ⇔ $\varphi_v(x)$=false

Supported here ⇔ $\varphi_v(x)$=true

3-Majority gate gadget



Computation of MAJ₃ gate $\Longleftrightarrow$ Eigenvalue-zero eigenvector of graph



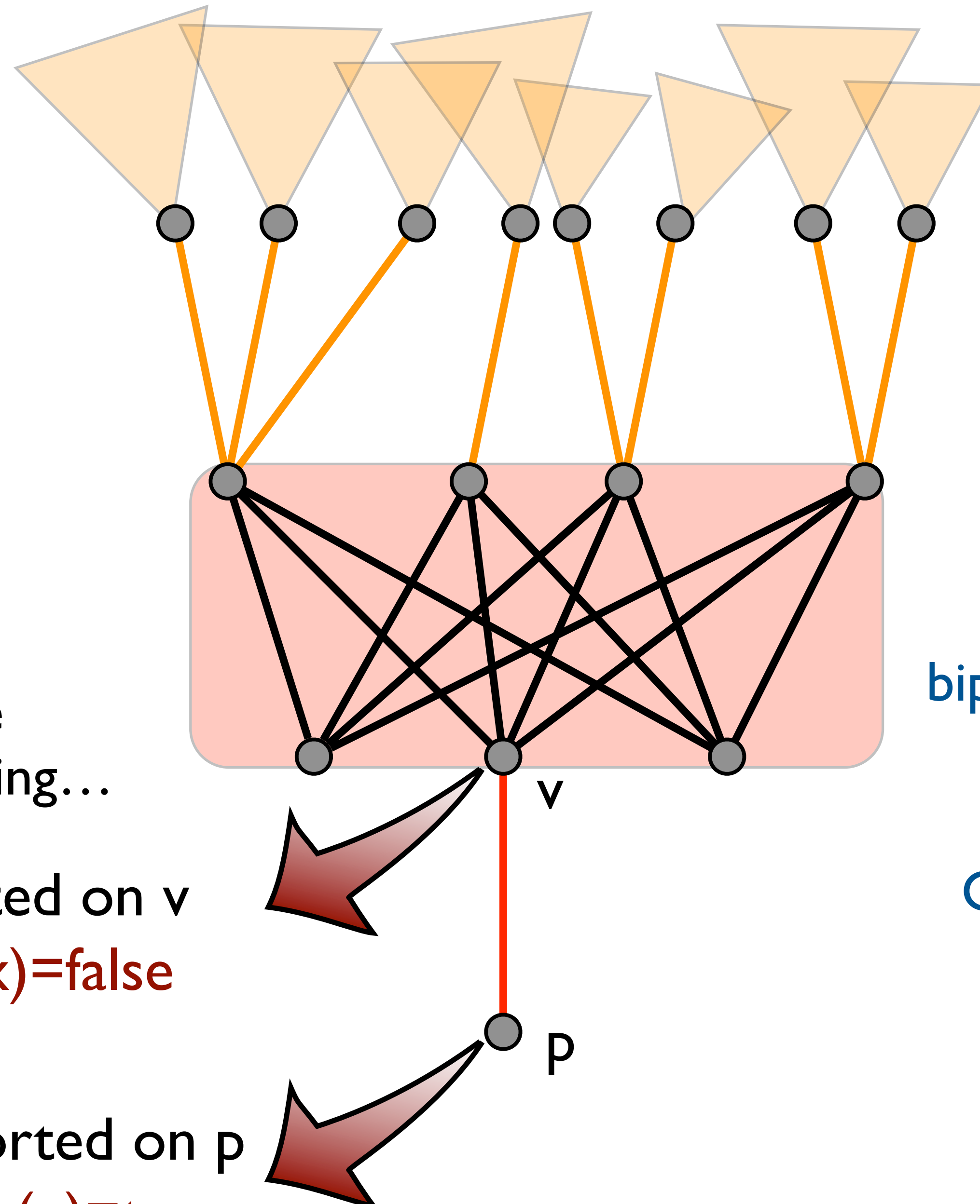
- Induction hypothesis: λ=0 eigenvectors on sub-formula graphs G(φ$_i$) compute the sub-formulas

- Constraints

$$\underbrace{\begin{pmatrix}1 & 1 & 1 & 1\\ 0 & 1 & \omega & \omega^2\end{pmatrix}}_{A_G}\begin{pmatrix}\alpha_r\\ \alpha_{v_1}\\ \alpha_{v_2}\\ \alpha_{v_3}\end{pmatrix} = 0 = \begin{pmatrix}1 & 1 & 1 & 0 & 0\\ 1 & \omega^2 & 0 & 1 & 0\\ 1 & \omega & 0 & 0 & 1\end{pmatrix}\begin{pmatrix}\alpha_s\\ \alpha_c\\ \alpha_{w_1}\\ \alpha_{w_2}\\ \alpha_{w_3}\end{pmatrix}$$

- When can α$_r$ be nonzero (i.e., gadget evaluates to true)?
  1. Only depends on first constraint eq.'s $(\alpha_{v_1}, \alpha_{v_2}, \alpha_{v_3})$
  2. Need $\alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3} \neq 0$, but
     $$\alpha_{v_1} + \omega\alpha_{v_2} + \omega^2\alpha_{v_3} = 0$$
  3. Can only have $\alpha_{v_i} \neq 0$ if input i evaluates to true
- At least two inputs φ$_i$ must be true to satisfy both constraints nontrivially. ✓MAJ₃

General graph gadgets



Induction Claim: Each edge (p,v) gives a "dual-rail" encoding…

The $\lambda=0$ eigenvector of $G(\varphi_v,x)$ is:

Supported on v $\Leftrightarrow \varphi_v(x)=\text{false}$

Supported on p $\Leftrightarrow \varphi_v(x)=\text{true}$

Span program definition

- Substitution rules defining G come from *span programs*. [Karchmer, Wigderson '93]
- **Def:** A span program P is:
  - A target vector t in vector space V over **C**,
  - Input vectors v_j each associated with a literal from $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$

Span program P computes $f_P: \{0,1\}^n \rightarrow \{0,1\}$,
$f_P(x) = 1 \Leftrightarrow$ t lies in the span of { true v_j }

- **Ex. 1:** P:

$$t = \binom{1}{0} \quad \overset{x_1}{\binom{1}{a}} \quad \overset{x_2}{\binom{1}{b}} \quad \overset{x_3}{\binom{1}{c}}$$

with a,b,c distinct and nonzero.

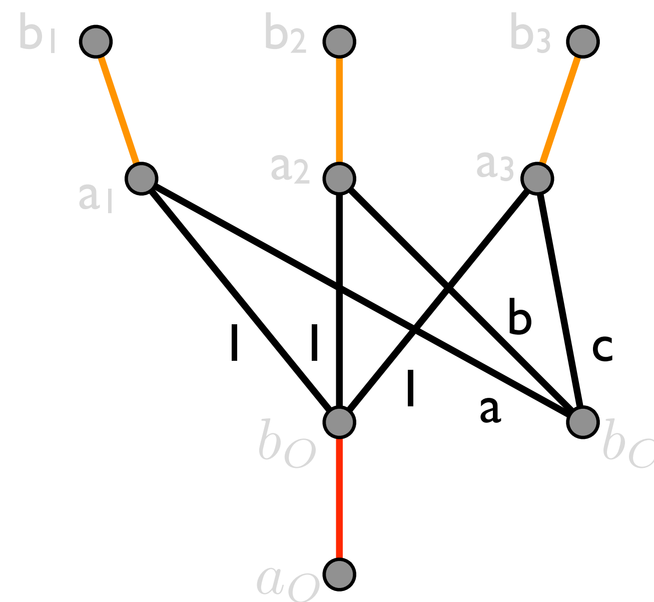
➡ $f_P = MAJ_3$

Span program ⇔ Bipartite graph gadget

with t=(1,0,…,0)

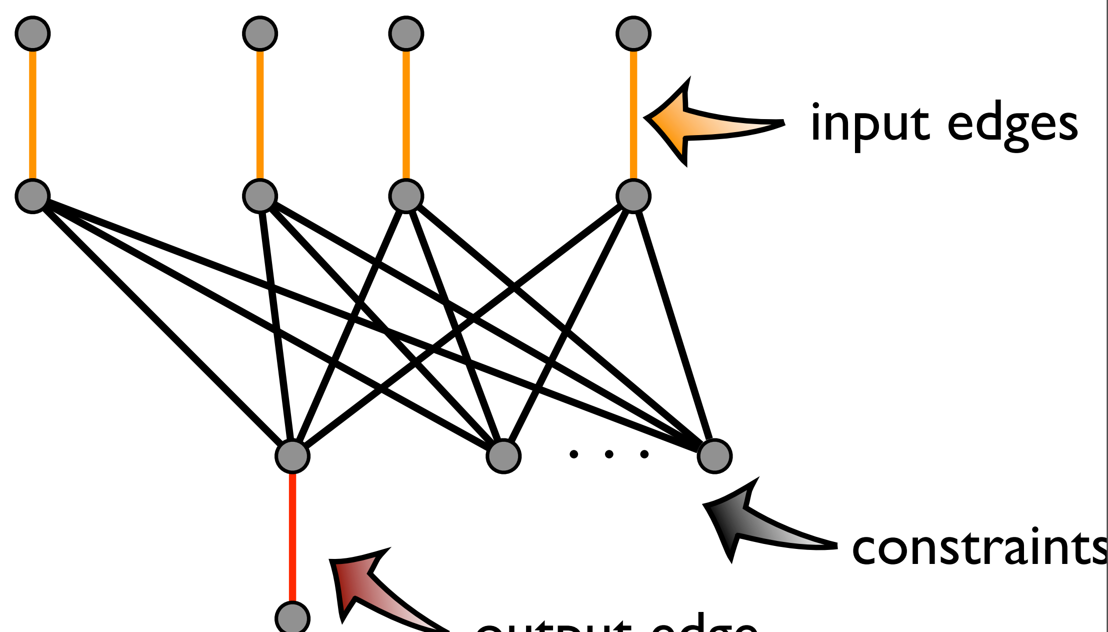
E.g., MAJ3:

$$t = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \begin{matrix} x_1 & x_2 & x_3 \\ 1 & 1 & 1 \\ a & b & c \end{matrix}$$



In general:

$$t = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \qquad \mathbf{A}$$



Composing span programs

- Given span programs for g, h_1, …, h_k, immediately get s.p. for

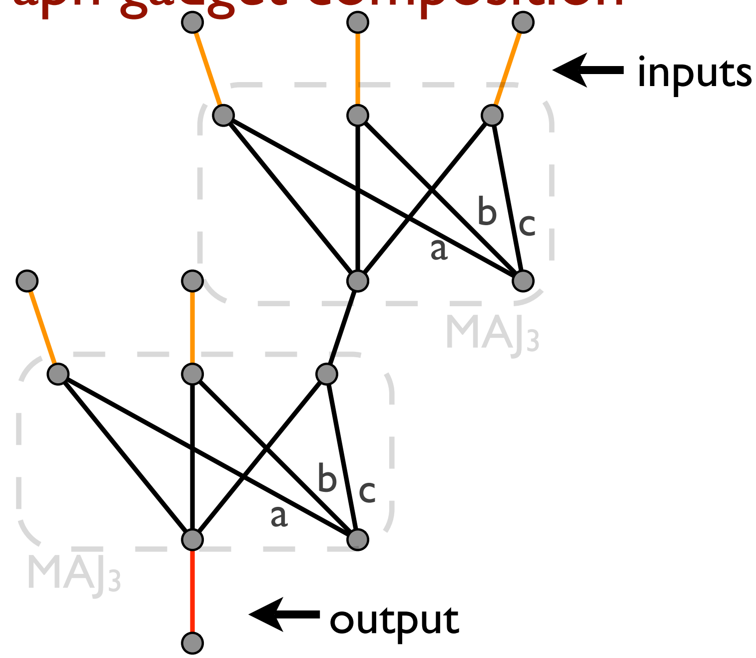
$$f = g \circ (h_1, \ldots, h_k)$$

SP composition

- Ex.: $MAJ_3(x_1, x_2, MAJ_3(x_4, x_5, x_6))$:

$\underline{\underline{t}}$	x_1	x_2	1	x_3	x_4	x_5
1	1	1	1	0	0	0
0	a	b	c	0	0	0
0	0	0	1	1	1	1
0	0	0	0	a	b	c

Graph gadget composition



Composing span programs

- Given span programs for g, h_1, …, h_k, immediately get s.p. for

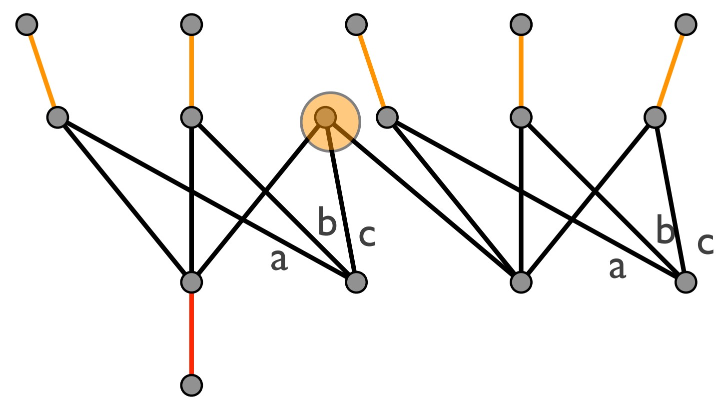
$$f = g \circ (h_1, \ldots, h_k)$$

SP composition

Graph gadget composition

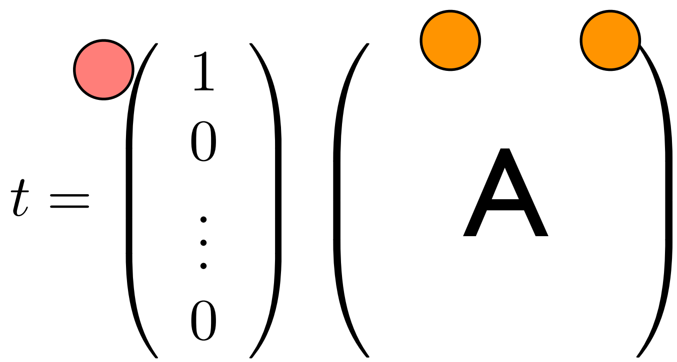
- Ex.: $MAJ_3(x_1, x_2, MAJ_3(x_4,x_5,x_6))$:

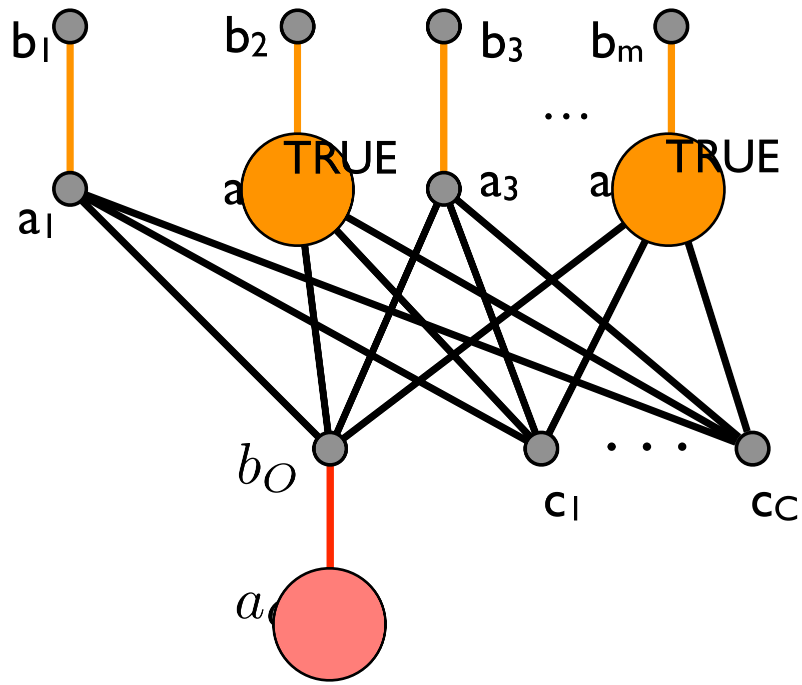
t	x_1	x_2	1	x_3	x_4	x_5
1	1	1	1	0	0	0
0	a	b	c	0	0	0
0	0	0	1	1	1	1
0	0	0	0	a	b	c



Eigenvalue-zero lemmas

- **Define:** $G_P(x)$ by deleting edges to true input literals
- **Lemma:** $f_P(x)=1 \Leftrightarrow \exists\ \lambda=0$ eigenvector of $A_{G_P(x)}$ supported on a_O.

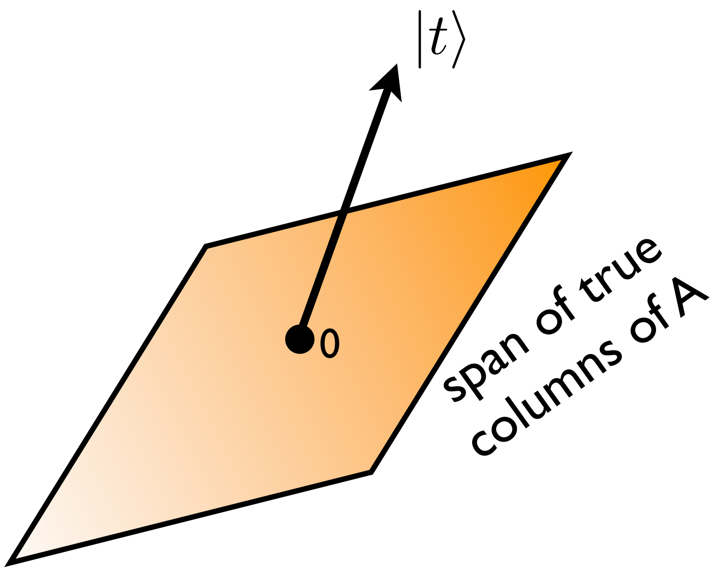


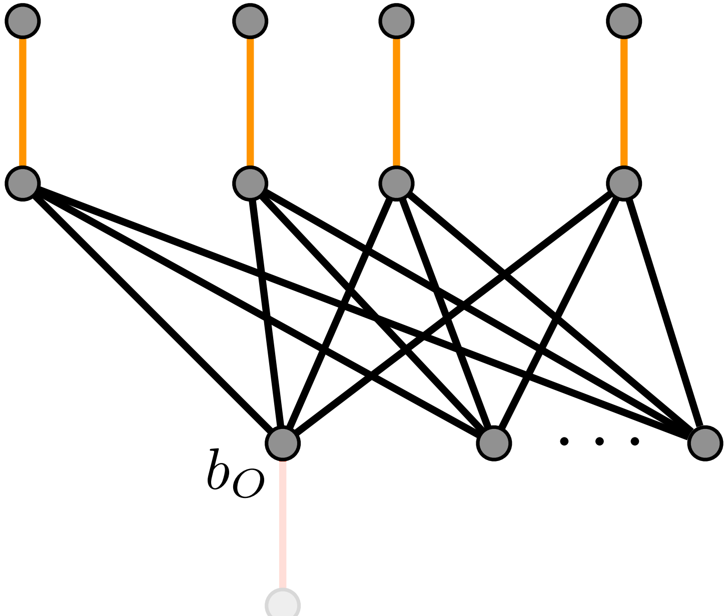


Eigenvalue-zero lemmas

- **Define:** $G_P(x)$ by deleting edges to true input literals
- **Lemma:** $f_P(x)=1 \Leftrightarrow \exists\ \lambda=0$ eigenvector of $A_{G_P(x)}$ supported on a_O.
- **Lemma:** Delete output edge (a_O, b_O). Then
  $f_P(x)=0 \Leftrightarrow \exists\ \lambda=0$ eigenvector supported on b_O.

Proof: $f_P(x)$ is false $\Leftrightarrow |t\rangle$ *not* in span of true columns of A



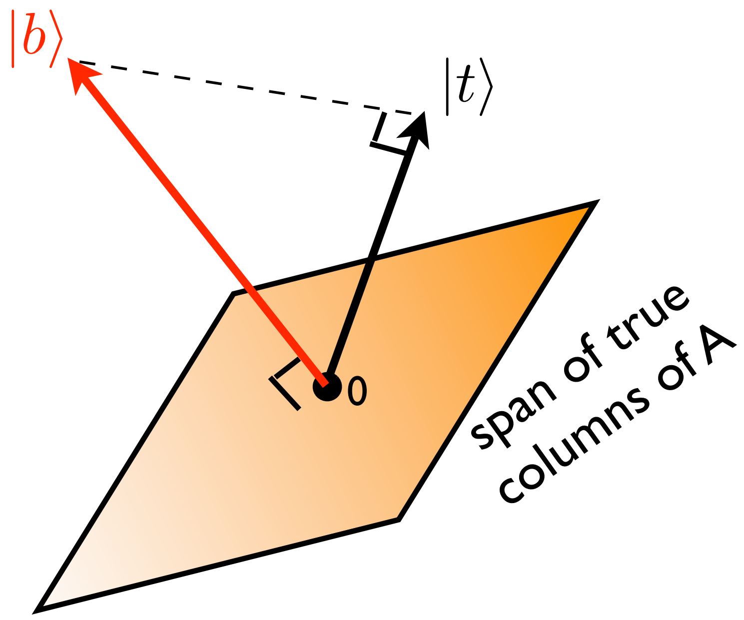


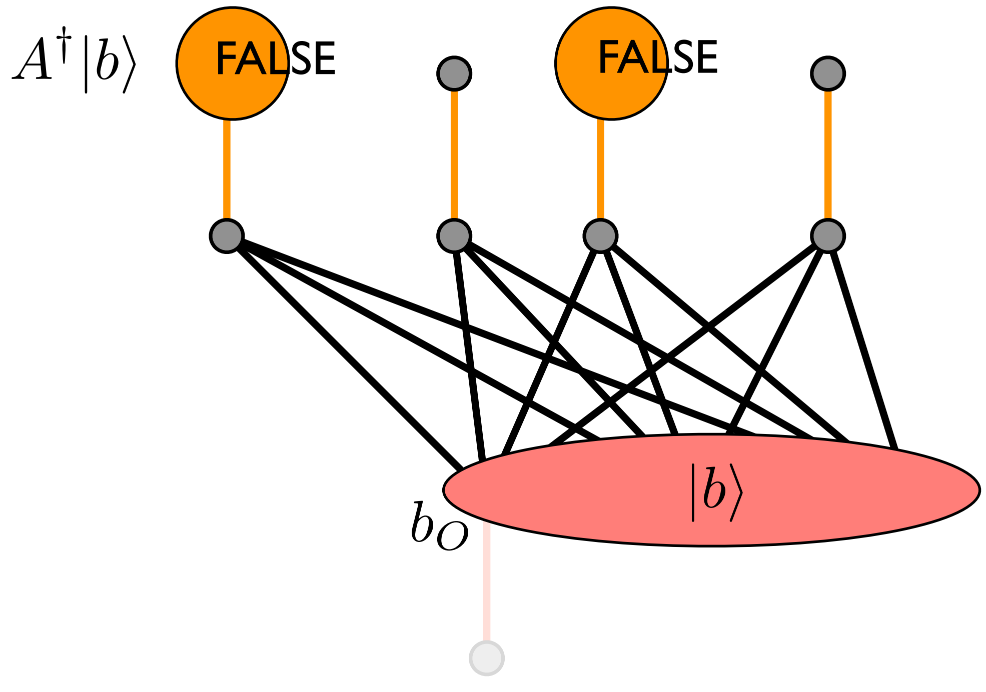
Eigenvalue-zero lemmas

- **Define:** $G_P(x)$ by deleting edges to true input literals
- **Lemma:** $f_P(x)=1 \Leftrightarrow \exists\ \lambda=0$ eigenvector of $A_{G_P(x)}$ supported on a_O.
- **Lemma:** Delete output edge (a_O, b_O). Then
  $f_P(x)=0 \Leftrightarrow \exists\ \lambda=0$ eigenvector supported on b_O.

Proof: $f_P(x)$ is false $\Leftrightarrow |t\rangle$ *not* in span of true columns of A

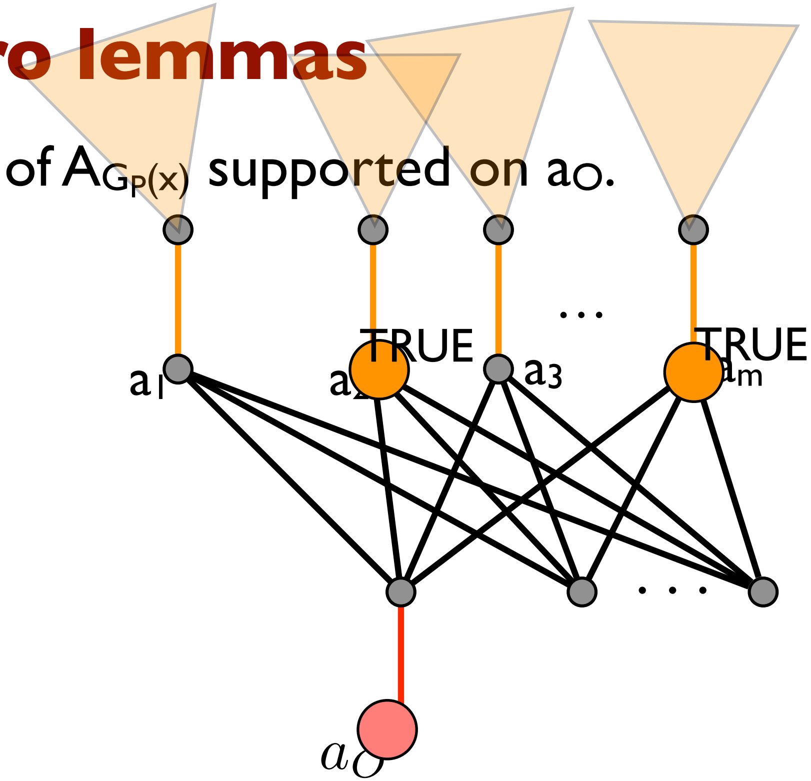
$\Leftrightarrow \exists\ |b\rangle$ with $\langle b|t\rangle=1$, orthogonal to all true columns of A





Quantitative Eigenvalue-zero lemmas

- **Lemma:** $f_P(x)=1 \Leftrightarrow \exists$ $\lambda=0$ eigenvector of $A_{G_P(x)}$ supported on a_O.



$$t = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \begin{pmatrix} \\ A \\ \\ \end{pmatrix}$$

- Assume that f(x)=1, and that for all true inputs i, we have constructed normalized $\lambda=0$ eigenvectors with squared support $\geq \gamma$ on a_i.
  Q: How large can we make $|a_O|^2$ in a normalized $\lambda=0$ eigenvector?

- **Answer:** Fix $a_O=1$ and try to minimize the eigenvector's norm. We want the shortest witness vector:

$$\min_{|w\rangle : \substack{\Pi|w\rangle = |w\rangle \\ A|w\rangle = |t\rangle}} \| |w\rangle \|^2 = \|(A\Pi)^- |t\rangle\|^2$$

:= wsize(P, x)

Π = projection onto true input coords.

Quantitative Eigenvalue-zero lemmas

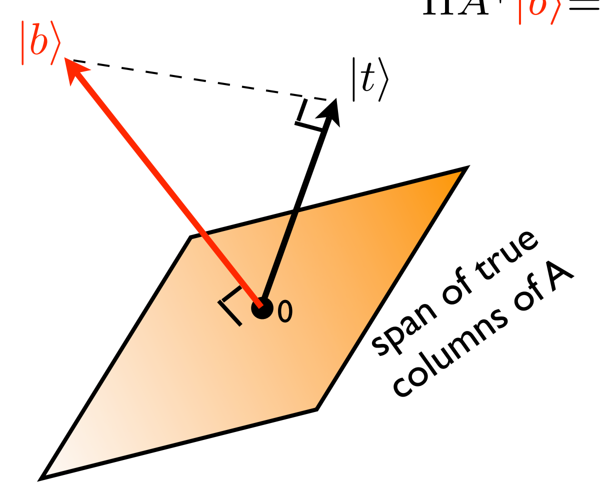
- **Assume:** For all inputs i we have constructed normalized λ=0 eigenvectors with squared support ≥ γ on a_i or b_i.

- **Lemma:** f(x)=1 ⇒ ∃ unit-normalized λ=0 eigenvector with

$$|a_O|^2 \geq \frac{\gamma}{\mathrm{wsize}(P,x)} \qquad \mathrm{wsize}(P,x) := \min_{|w\rangle : \substack{\Pi|w\rangle = |w\rangle \\ A|w\rangle = |t\rangle}} \| |w\rangle \|^2$$

- **Lemma:** f(x)=0 ⇒ ∃ unit-normalized λ=0 eigenvector with

$$|b_O|^2 \geq \frac{\gamma}{\mathrm{wsize}(P,x)} \qquad \mathrm{wsize}(P,x) := \min_{|b\rangle : \substack{\langle t|b\rangle = 1 \\ \Pi A^\dagger |b\rangle = |t\rangle}} \| A^\dagger |b\rangle \|^2$$



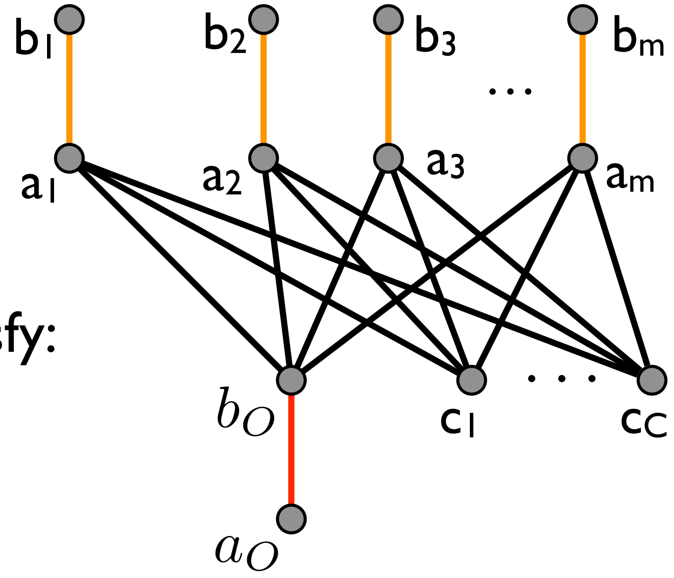
- **Def:** Witness size of P

$$\mathrm{wsize}(P) = \max_x \mathrm{wsize}(P,x)$$

Small λ≠0 analysis

- Construct the eigenvectors starting at the leaves, and working down. Eigenvector equations are

$$\lambda b_C = A_{CJ} a_J$$
$$\lambda b_O = A_{OJ} a_J + a_O$$
$$\lambda a_J = {A_{IJ}}^\dagger b_I + {A_{OJ}}^\dagger b_O + {A_{CJ}}^\dagger b_C$$



- **Induction assumption:** Input ratios $r_i = a_i/b_i$ satisfy:

i false $\Rightarrow$ $r_i \in (0, s_i \lambda)$

i true $\Rightarrow$ $r_i \in \left(-\infty, \frac{-1}{s_i \lambda}\right)$

- Solve equations for $r_O = a_O/b_O$, apply Woodbury identity, expand the Taylor series in λ of the matrix inverse (on the range and its Schur complement separately), bound the higher-order terms, QED.
  - The first-order term is the same as the factor wsize(P, x) lost in the λ=0 analysis (not so surprisingly)

Framework for quantum algorithms based on span programs:

- Quantum algorithm for evaluating "span programs":

Span program P ⟷ Graph G_P with detectable spectral gap ⟶ Algorithm using a quantum walk

  - Behaves well under composition/recursion:

Span programs for gates	⟶ compose	Composed span program for φ
↕		↕
Gate graph gadgets	⟶ compose	Graph G(φ)

- Possible extensions: Interesting quantum algorithms based directly on asymptotically large span programs?

Summary of technical results

- **Def:** Let S' = { arbitrary two- or three-bit gates, O(1)-fan-in EQUAL gates}
  Let S = { O(1)-size {AND, OR, NOT, PARITY} formulas on inputs that are themselves possibly elements of S' }
- E.g., $\mathrm{MAJ}_3(x_1, x_2, x_3) \wedge (x_4 \oplus x_5 \oplus \cdots \oplus (x_{k-1} \vee x_k))$
- (Idea: Gates other than AND, OR, PARITY need to have balanced inputs.
  AND, OR, PARITY gates can have constant-factor unbalanced inputs)

- **Def:** Read-once formula φ is "adversary-bound-balanced" if for each gate g, the adversary bounds for its input sub-formulas are all the same.
- **Main Theorem:** Any adversary-balanced formula φ over gate set S can be evaluated in O(ADV(φ)) queries.
  Time complexity is the same, up to poly-log N factor, in coherent RAM model after preprocessing.

Questions?

3-bit gates

Gate	Adversary lower bound
0	0
x_1	1
$x_1 \wedge x_2$	$\sqrt{2}$
$x_1 \oplus x_2$	2
$x_1 \wedge x_2 \wedge x_3$	$\sqrt{3}$
$x_1 \oplus x_2 \oplus x_3$	3
$x_1 \oplus (x_2 \wedge x_3)$	$1 + \sqrt{2}$
$x_1 \vee (x_2 \wedge x_3)$	$\sqrt{3}$
$(x_1 \wedge x_2) \vee (\overline{x_1} \wedge x_3)$	2
$x_1 \vee (x_2 \wedge x_3) \vee (\overline{x_2} \wedge \overline{x_3})$	$\sqrt{5}$
$\mathrm{MAJ}_3(x_1, x_2, x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$	2
$\mathrm{MAJ}_3(x_1, x_2, x_3) \vee (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3})$	$\sqrt{7}$
$\mathrm{EQUAL}(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_2} \wedge \overline{x_3})$	$3/\sqrt{2}$
$(x_1 \wedge x_2 \wedge x_3) \vee (\overline{x_1} \wedge \overline{x_2})$	$\sqrt{3 + \sqrt{3}}$

Fact: $A(f \oplus g) = A(f) + A(g)$, $A(f \wedge g) = \sqrt{(A(f)^2 + A(g)^2)}$ if f, g have disjoint inputs.