SIAM J. COMPUT.
Vol. 39, No. 6, pp. 2513–2530

ANY AND-OR FORMULA OF SIZE N CAN BE EVALUATED IN TIME $N^{1/2+o(1)}$ ON A QUANTUM COMPUTER*

A. AMBAINIS†, A. M. CHILDS‡, B. W. REICHARDT§, R. ŠPALEK¶, AND S. ZHANG‖

Abstract. Consider the problem of evaluating an AND-OR formula on an N-bit black-box input. We present a bounded-error quantum algorithm that solves this problem in time $N^{1/2+o(1)}$. In particular, approximately balanced formulas can be evaluated in $O(\sqrt{N})$ queries, which is optimal. The idea of the algorithm is to apply phase estimation to a discrete-time quantum walk on a weighted tree whose spectrum encodes the value of the formula.

Key words. quantum computation, quantum query complexity, formula evaluation, AND-OR trees, quantum walk

AMS subject classifications. 68Q10, 81P68

DOI. 10.1137/080712167

1. Introduction.

Consider a Boolean formula φ on N binary inputs $x_1, \ldots, x_N$, using a gate set S consisting of either AND, OR, and NOT gates or, equivalently, NAND gates. The formula φ corresponds to a tree where each internal node is a gate from S on its children. If the same variable is fed into different inputs of φ, we treat each occurrence separately, so that N counts variables with multiplicity. The variables x_i are accessed by querying a quantum oracle, which we can take to be the unitary operator

$$O_x : |b, i\rangle \mapsto (-1)^{bx_i}|b, i\rangle, \tag{1}$$

where $b \in \{0, 1\}$ and $i \in \{1, \ldots, N\}$ label the control qubit and query index, respectively. In this paper, we show the following.

THEOREM 1. *Let φ be an arbitrary AND-OR formula of size N. After efficient (i.e., time* poly(N)*) classical preprocessing that does not depend on the input x, $\varphi(x)$ can be evaluated with error at most $1/3$ using $N^{1/2+O(1/\sqrt{\log N})}$ queries to O_x. The running time is also $N^{1/2+O(1/\sqrt{\log N})}$, assuming unit-cost coherent access to the result*

*Received by the editors January 2, 2008; accepted for publication (in revised form) July 16, 2009; published electronically April 30, 2010.

http://www.siam.org/journals/sicomp/39-6/71216.html

†Department of Computer Science, University of Latvia, LV-1459 Riga, Latvia (ambainis@lu.lv). This author's research was supported by CIAR, ARO, NSERC, MITACS, and an IQC University Professorship as well as by a University of Latvia Research grant ZB01-100 and an FP7 Marie Curie International Reintegration grant PIRG02-GA-2007-224886.

‡Department of Combinatorics & Optimization and Institute for Quantum Computing, University of Waterloo, Waterloo, Waterloo, ON, N2L 3G1, Canada (amchilds@uwaterloo.ca). This author's research was supported by NSF grant PHY-0456720, ARO grant W911NF-05-1-0294, and by NSERC, MITACS, Quantum Works, and the US ARO/DTO.

§School of Computer Science and Institute for Quantum Computing, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (breic@uwaterloo.ca). This author's research was suported by NSF grants CCF-0524828, CCF-0515342, and PHY-0456720, and by ARO grant W911NF-05-1-0294, and by NSERC.

¶Google, Inc., Mountain View, CA 94043 (spalek@google.com). This author's research was supported by NSF grant CCF-0524837 and ARO grant DAAD 19-03-1-0082.

‖Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N. T., Hong Kong (syzhang@cse.cuhk.edu.hk). This author's research was supported by NSF grant PHY-0456720, ARO grant W911NF-05-1-0294 and Hong Kong grant RGCGRF-419309.

of the preprocessing. For an approximately balanced formula (see Definition 2), the query complexity is only $O(\sqrt{N})$ and the running time is only $\sqrt{N}(\log N)^{O(1)}$.

Our algorithm is inspired by the recent $N^{1/2+o(1)}$-time algorithm of Farhi, Goldstone, and Gutmann [FGG08] for the case in which $S = \{\text{NAND}\}$, each NAND gate in φ has exactly two inputs, and φ is *balanced*—i.e., $N = 2^n$ and φ has depth n. Our algorithm requires no preprocessing in this special case of a balanced binary NAND tree. For a balanced or even an "approximately balanced" NAND tree, our algorithm requires only $O(\sqrt{N})$ queries, which is optimal [BS04]. As in [FGG08], we analyze the algorithm in terms of the spectrum of a Hermitian matrix, but in general this matrix involves weights that compensate for the formula's imbalances.

This result almost resolves in the positive a problem posed by Laplante, Lee, and Szegedy [LLS06]. They asked whether the square of the bounded-error quantum query complexity $Q(f)$ of evaluating a Boolean function f is a lower bound on the size $L(f)$ of the smallest formula evaluating that function. Theorem 1 says that $Q(f)$ is at most $L(f)^{1/2+o(1)}$ or, equivalently, that the formula size of f is at least $Q(f)^{2-o(1)}$.

Our algorithm also almost solves a problem of O'Donnell and Servedio [OS03], who conjectured that every Boolean formula of size N has a polynomial threshold function of degree $\sqrt{N}$. Our result implies that every Boolean formula of size N has a polynomial threshold function of degree $N^{1/2+o(1)}$, because a T-query quantum algorithm implies an upper bound of $2T$ for the corresponding polynomial threshold function [BBC+01]. By previous results [KS04, KOS04], this in turn implies that the class of Boolean formulas of size N can be classically learned in time $N^{N^{1/2+o(1)}} = 2^{N^{1/2+o(1)}}$, in both the probably approximately correct model and the online model of learning from adversarially generated examples.

Note that evaluating an AND-OR tree is the decision version of evaluating a MIN-MAX tree; the latter can be solved using any algorithm for the former with at most a logarithmic slowdown.

History of the problem and related work. Grover showed in 1996 [Gro97, Gro02] how to search a general unstructured database of size N, represented by a black-box oracle function, in $O(\sqrt{N})$ oracle queries and $O(\sqrt{N}\log\log N)$ time on a quantum computer. Grover's search algorithm can be used to compute the logical OR of N bits in the same time, by simply searching for a 1 in the input string. By applying Grover search recursively, one can speed up the computation of more general logical formulas. For example, a two-level AND-OR tree, with one AND gate of fan-in $\sqrt{N}$ and $\sqrt{N}$ OR gates of the same fan-in as its children, can be evaluated in $O(\sqrt{N}\log N)$ queries. Here the logarithmic factor comes from amplifying the success probability of the inner quantum search to be polynomially close to one, so that the total error is at most constant. By iterating the same argument, regular AND-OR trees of depth d can be evaluated with constant error in time $O(\sqrt{N}\log^{d-1} N)$ [BCW98].

Høyer, Mosca, and de Wolf [HMW03] showed that Grover search can be applied even if the input variables are noisy, so the log factor is not necessary. Consequently, a depth-d AND-OR tree can be computed in $O(\sqrt{N}\cdot c^d)$ queries, where c is a constant that comes from their algorithm. It follows that constant-depth AND-OR trees can be computed in $O(\sqrt{N})$ queries. Unfortunately, their algorithm is too slow for the balanced binary AND-OR tree of depth $\log_2 N$ (although it does give some speedup over classical computation for sufficiently large constant fan-ins).

Classically, one can compute the value of a balanced binary AND-OR tree with zero error in expected time $O(N^{\log_2[(1+\sqrt{33})/4]}) = O(N^{0.754})$ [Sni85, SW86] using a

technique called alpha-beta pruning. This running time is optimal even for bounded-error algorithms [San95]. For a long time, no quantum algorithm was known that performed better than this classical zero-error algorithm, despite the fact that the best known lower bound, from the adversary method, is only $\Omega(\sqrt{N})$ [BS04].

Recently, Farhi, Goldstone, and Gutmann [FGG08] presented a ground-breaking quantum algorithm for the balanced binary case using continuous-time quantum walks. Their algorithm is based on the concept of scattering and runs in time $O(\sqrt{N})$ in an unconventional, continuous-time query model. Shortly afterward, Childs et al. [CCJY09] pointed out that this algorithm can be discretized into the conventional oracle query model with a small slowdown, to run in time $N^{1/2+o(1)}$.

The present paper is based on a merged version of the technical reports [CRŠZ07, Amb07], which appeared in a joint conference proceeding [ACR+07].

Two of the authors have since generalized the framework of this paper by incorporating the concept of span programs and have thereby given algorithms for evaluating formulas over more general gate sets [RŠ08]. For example, span programs provide an optimal algorithm for evaluating a balanced recursive three-majority formula. One of the authors has investigated generalizations and further applications of a correspondence employed in our algorithm between continuous- and discrete-time quantum walks [Chi08]. Also, it has been shown that the formula evaluation algorithm of this paper can be applied recursively to find a certificate for a balanced, regular NAND formula in nearly optimal time [ACLT09].

2. Summary of results and methods. We design an algorithm whose running time depends on the structure of the NAND tree. For arbitrary balanced trees, the algorithm uses $O(\sqrt{N})$ queries. More generally, if the fan-in of each NAND gate in our formula is bounded by a constant, our algorithm uses $O(\sqrt{N}d)$ queries, where d is the depth of the formula. If the depth is large, we can use a rebalancing procedure [BCE95, BB94] to construct an equivalent formula with depth $2^{O(\sqrt{\log N})}$. This implies that any NAND formula of size N can be evaluated using $N^{1/2+O(1/\sqrt{\log N})}$ queries.

Idea of the algorithm. Starting from a tree representing the formula φ with input x (see Figure 1), we define a weighted adjacency matrix H such that the eigen-

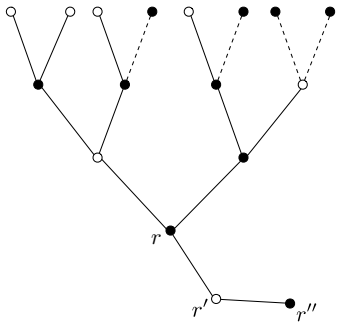

FIG. 1. *The balanced binary NAND formula of depth three*

$$\varphi(x) = \big((x_1 \mathbin{\overline{\wedge}} x_2) \mathbin{\overline{\wedge}} (x_3 \mathbin{\overline{\wedge}} x_4)\big) \mathbin{\overline{\wedge}} \big((x_5 \mathbin{\overline{\wedge}} x_6) \mathbin{\overline{\wedge}} (x_7 \mathbin{\overline{\wedge}} x_8)\big),$$

where $a \mathbin{\overline{\wedge}} b = \overline{(a \wedge b)}$*, is represented by the subtree rooted at* r*. The leaves (at the top of the tree) correspond to the input, which is encoded by deleting edges to leaves evaluating to* 1*, as indicated by a dashed line. Internal vertices correspond to NAND gates on their children. A vertex is represented by an open circle if it evaluates to* 0 *and by a filled circle if it evaluates to* 1*. In this example, the input is* $x = 00010111$*, and* $\varphi(x) = 1$*. Below* r*, we add two more vertices* r' *and* r''*; then* r'' *also evaluates to* $\varphi(x)$*.*

states of H with small eigenvalues encode $\varphi(x)$. In particular,

- if $\varphi(x) = 0$, then H has a normalized eigenstate with eigenvalue zero and with large amplitude on the root.
- if $\varphi(x) = 1$, then any eigenstate of H with nonzero amplitude on the root has an eigenvalue bounded away from zero.

We carefully design H so that quantitative versions of both of these properties can be established inductively.

Considering H as the Hamiltonian of a quantum system, these properties mean that $\varphi(x)$ can be evaluated by applying phase estimation to the unitary operator e^{-iHt}, with the root as the starting state.[1] However, that approach requires simulating the dynamics generated by H, a continuous-time quantum walk on the tree, and this introduces unnecessary overhead. We avoid this overhead by converting the process into a corresponding discrete-time quantum walk with similar behavior, using Szegedy's correspondence between classical random walks and discrete-time quantum walks. We reinterpret Szegedy's correspondence in order to construct a discrete-time quantum walk from an arbitrary continuous-time quantum walk with positive weights such that the spectral properties of the two walks are closely related.

Figure 2 presents a simplified version of the overall algorithm for the balanced binary case.

Organization. The remainder of the paper is organized as follows.

For any NAND formula φ, section 3 defines a weighted undirected tree $T(\varphi)$, where the weights of edges to the leaves depend on the input x and where a short tail is added to the root.

In section 4 and section 5, we establish spectral properties of this weighted tree. Section 4 considers only the eigenstates with eigenvalue zero. We show that when $\varphi(x) = 0$, there is a normalized zero-eigenvalue eigenstate with substantial overlap on the root. Conversely, if $\varphi(x) = 1$, any such eigenstates have no overlap on the root and hence can be neglected. Section 5 then shows that in the case $\varphi(x) = 1$, any eigenvectors with small nonzero eigenvalues can also be neglected. Essentially, these properties follow because the ratio of eigenstate amplitudes between a vertex and its parent depend on the evaluation of NAND gates.

We then apply the spectral analysis to construct the algorithm. Section 6 reviews Szegedy's correspondence theorem, which we use to construct a discrete-time quantum walk whose eigenvalues and eigenvectors are closely related to those of the weighted tree. In section 7, we explain how to evaluate φ by applying phase estimation to this discrete-time quantum walk.

We conclude the paper by describing some applications to evaluating iterated functions in section 8 and by presenting some open problems in section 9.

3. Weighted NAND formula tree. The NAND gate on inputs $y_1, \ldots, y_k \in \{0, 1\}$ evaluates to $1 - \prod_{i=1}^{k} y_i$. In particular, a NAND gate on a single input is simply a NOT gate.

Consider a NAND formula φ of size N, i.e., on N variables, counting multiplicity. Represent φ by a rooted tree $T = T(\varphi)$, in which the leaves correspond to variables and other vertices correspond to NAND gates on their children. (Because φ is a formula,

[1] Note that phase estimation has also been used to resolve the eigenvalue gap of a quantum walk—albeit for a different purpose, namely, to implement a reflection operator used in each step of a search algorithm—in [MNRS07].

1. **Initialization.** Let $\tau = 320\lfloor\sqrt{N}\rfloor$. Prepare three quantum registers in the state

$$\Big(\frac{1}{\sqrt{\tau}}\sum_{t=0}^{\tau-1}(-i)^t|t\rangle\Big)\otimes|r''\rangle|\text{left}\rangle.$$

   The first register is a counter for quantum phase estimation, the second register holds a vertex index, and the third register is a qutrit "coin" holding "down," "left," or "right."
2. **Quantum walk.** If the first register is $|t\rangle$, perform t steps of the following discrete-time quantum walk U. Denote the last two registers by $|v\rangle|c\rangle$.
   - **Diffusion step.**
     (a) If v is a leaf, apply a phase flip $(-1)^{x_v}$ using one controlled call to the input oracle.
     (b) If v is an internal degree-three vertex, apply the following diffusion operator on coin $|c\rangle$:

$$\text{Reflection}_{|u\rangle} = 2|u\rangle\langle u| - 1,$$

     where $|u\rangle = \frac{1}{\sqrt{3}}(|\text{down}\rangle + |\text{left}\rangle + |\text{right}\rangle)$.
     (c) If $v = r'$, apply the following diffusion operator on $|c\rangle$:

$$\text{Reflection}_{|u'\rangle} = 2|u'\rangle\langle u'| - 1,$$

     where $|u'\rangle = \frac{1}{\sqrt[4]{N}}|\text{down}\rangle + \sqrt{1-\frac{1}{\sqrt{N}}}|\text{left}\rangle$.
     (d) If $v = r''$, do nothing.
   - **Walk step.**
     (a) If $c =$ "down," then walk down to the parent of v and set c to either "left" or "right," depending on which child v is.
     (b) If $c \in \{$"left," "right"$\}$, then walk up to the corresponding child of v and set c to 'down'.

   Note that the walk step operator is a permutation that simply flips the direction of each oriented edge.
3. **Quantum phase estimation.** Apply the inverse quantum Fourier transform (modulo τ) on the first register and measure it in the computational basis. Return 0 if and only if the outcome is 0 or $\tau/2$.

FIG. 2. *An optimal quantum algorithm to evaluate the balanced binary NAND formula using $O(\sqrt{N})$ queries. The algorithm runs quantum phase estimation on top of the quantum walk of Figure* 1.

not a circuit, each gate has fan-out one, so there are no loops in the associated graph.) Attach to the root r a "tail" of two vertices r' and r'', as in Figure 1.

DEFINITION 1. *For a vertex v, let s_v be the number of inputs of the subformula rooted at v, counting multiplicity. In particular, $s_r = s_{r'} = s_{r''} = N$; if v is a leaf, then $s_v = 1$. Let $\overline{\wedge}(v)$ denote the value of the subformula rooted at v, so $\varphi(x) = \overline{\wedge}(r) = \overline{\wedge}(r'')$.*

To track error terms through the analysis, it will also be helpful to define

$$\sigma_-(v) = \max_{\xi} \sum_{w \in \xi} \frac{1}{\sqrt{s_w}}, \qquad \sigma_+(v) = \max_{\xi} \sum_{w \in \xi} s_w, \tag{2}$$

with the maximum in each case taken over all paths ξ from v up to a leaf of T. Let $\sigma_-(\varphi) = \sigma_-(r)$ and $\sigma_+(\varphi) = \sigma_+(r)$.

Letting $\mathrm{depth}(\varphi)$ denote the depth of the formula φ, it is clear that, for all vertices v, $\sigma_-(v) \le \sigma_-(\varphi) \le \mathrm{depth}(\varphi)$ and $\sigma_+(v) \le \sigma_+(\varphi) \le N\,\mathrm{depth}(\varphi)$.[2]

DEFINITION 2. *The formula φ is* approximately balanced *if $\sigma_-(\varphi) = O(1)$ and $\sigma_+(\varphi) = O(N)$.*

The simplest example of an approximately balanced tree is the balanced binary tree with N leaves. In this case, $\sigma_-(\varphi) < 2$ and $\sigma_+(\varphi) < 2N$ are both geometric series. More generally, a tree is approximately balanced if s_v decreases sufficiently rapidly from the root toward the leaves. For example, if for a fixed $\epsilon \in (0, \frac{1}{2}]$, for every vertex p and every grandchild c of p, $s_c \le (1-\epsilon)s_p$, then $\sigma_-(\varphi) = O(1/\epsilon)$ and $\sigma_+(\varphi) = O(N/\epsilon)$.

DEFINITION 3. *Let H be a weighted, symmetric adjacency matrix of the graph consisting of T and the attached tail. Letting $h_{pv} = \langle p|H|v\rangle = \langle v|H|p\rangle$ denote the weight on the edge between a vertex v and its parent p, we have*

$$H|v\rangle = h_{pv}|p\rangle + \sum_c h_{vc}|c\rangle, \tag{3}$$

where the sum is over the children of v. (If v has no parent or no children, the respective terms are zero.) The edge weights depend on the structure of the tree and are given by

$$h_{pv} = \left(\frac{s_v}{s_p}\right)^{1/4}, \tag{4}$$

with two exceptions:

1. *If a leaf v evaluates to $\overline{\wedge}(v) = 1$, set $h_{pv} = 0$, i.e., effectively remove the edge (p, v) by setting its weight to zero.*
2. *Set $h_{r''r'} = 1/(\sqrt{\sigma_-(\varphi)}N^{1/4})$.*

Our algorithm relies on spectral properties of H.

THEOREM 2. *The weighted adjacency matrix H has the following properties:*

- *If $\varphi(x) = 0$, then there exists a zero-eigenvalue eigenvector $|a\rangle$ of H with $\||a\rangle\| = 1$ and overlap $|\langle r''|a\rangle| \ge 1/\sqrt{2}$.*
- *If $\varphi(x) = 1$, then every eigenvector with support on r' or r'' has corresponding eigenvalue at least $1/\big(9\sigma_-(\varphi)\sqrt{\sigma_+(\varphi)}\big)$ in absolute value.*

The following two sections contain the proof of Theorem 2.

Remark 1. *Although we have specified H with particular weights for the sake of concreteness, there is considerable flexibility in the choice of these weights. For a leaf*

[2] In fact, $\sigma_-(\varphi) = O(\sqrt{\mathrm{depth}(\varphi)})$, because s_w must increase by at least one every two levels toward the root; two NOT gates in a row would be redundant. Slightly stronger bounds can be given for trees preprocessed according to the rebalancing procedure of Lemma 9, but $\mathrm{poly}(\mathrm{depth}(\varphi))$ and $\mathrm{poly}(\log N)$ factors here won't significantly change the running time.

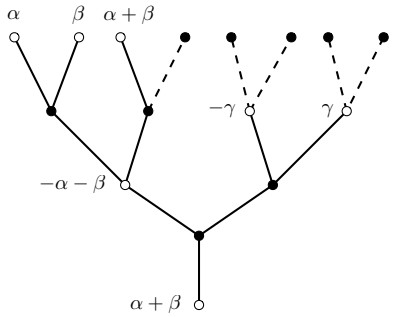

FIG. 3. *An example NAND tree to illustrate Lemmas* 3 *and* 4. *As in Figure* 1, *a vertex is filled or not according to whether it evaluates to* 1 *or* 0, *respectively. The amplitudes* $\langle v|a\rangle$ *of a zero-eigenvalue eigenstate* $|a\rangle$ *of the adjacency matrix* H *are also labeled, with* α, β, γ *free variables, assuming* $h_{pv}=1$ *for every edge* (p,v). *The eigenvalue condition means that the amplitudes of the neighbors of any vertex sum to zero. The existence of such an* $|a\rangle$ *is promised by Lemma* 4. *As required by Lemma* 3, $\langle v|a\rangle=0$ *if* $\overline{\wedge}(v)=1$, *so vertices evaluating to* 1 *are not labeled.*

v evaluating to 0, *one can check that it is sufficient for* h_{pv} *to satisfy* $h_{pv}\geq 1/s_p^{1/4}$. *One can also verify that for any fixed* $\beta\in(0,1/2)$, *Theorem* 2 *holds if the weights* h_{pv} *are defined by* $h_{pv}={s_v}^{\beta}/{s_p}^{\frac{1}{2}-\beta}$ *and* $h_{r''r'}=1/(\sigma_-^{\beta}(\varphi)^{\frac{1}{2}}N^{\frac{1}{2}-\beta})$, *where* $\sigma_-^{\beta}(\varphi)=\max_{\xi}\sum_{w\in\xi:\,\overline{\wedge}(w)=0}{s_w}^{-2\beta}$. *We have fixed* $\beta=1/4$ *to simplify notation.*

4. Zero-eigenvalue eigenstates of H. In this section, we consider the eigenstates of H with eigenvalue zero. We first prove the second half of Theorem 2 in the special case where the eigenvalue is zero and then prove the first half of the theorem.

Recall that in a NAND tree T, internal vertices are interpreted as NAND gates on their children. As Definition 3 puts zero weight on the parental edge of a leaf evaluating to 1, such leaves can be regarded as disconnected. Thus all leaves connected to the root component can be interpreted as 0s.

DEFINITION 4. *Let* T_v *denote the subtree of* T *consisting of* v *and all its descendants. The restriction to* T_v *of a vector* $|a\rangle$ *on* T *is denoted* $|a_{T_v}\rangle$. *That is, for a subset* S *of the vertices, define the projection* $\Pi_S=\sum_{s\in S}|s\rangle\langle s|$; *then* $|a_{T_v}\rangle=\Pi_{T_v}|a\rangle$. *Also let* $a_v=\langle v|a\rangle$, *and let* $H_S=\Pi_S H$.

LEMMA 3. *For an internal vertex* p *in the NAND tree* T, *if* $\overline{\wedge}(p)=1$ *and* $H_{T_p}|a\rangle=0$, *then* $a_p=0$.

Proof. Since $\overline{\wedge}(p)=1$, there exists a child v of p having $\overline{\wedge}(v)=0$. If v is a leaf, then $0=\langle v|H|a\rangle=h_{pv}a_p$, as asserted. Otherwise, all children c of v must have $\overline{\wedge}(c)=1$, implying by induction that $a_c=0$. Then

$$0=\langle v|H|a\rangle=h_{pv}a_p+\sum_c h_{vc}a_c=h_{pv}a_p,$$

so $a_p=0$, as claimed. □

Lemma 3 constrains the existence of zero-eigenvalue eigenstates supported on the root r when the NAND formula evaluates to 1. However, there may be zero-eigenvalue eigenstates that are not supported on the root; for example, consider the right subtree in Figure 3.

LEMMA 4. *Consider a vertex* p *in NAND tree* T. *If* $\overline{\wedge}(p)=0$, *then there exists an* $|a\rangle=|a_{T_p}\rangle$ *with* $H_{T_p}|a\rangle=0$, $\||a\rangle\|=1$, *and* $a_p\geq 1/(\sqrt{\sigma_-(p)}s_p^{1/4})$.

Proof. The proof is by induction on the height of the tree. For the base case when p is a leaf, let $|a\rangle=|p\rangle$.

If p is not a leaf, then $\overline{\wedge}(p) = 0$ implies $\overline{\wedge}(v) = 1$ for every child v of p. Thus, each v has at least one particular child, denoted c^v, satisfying $\overline{\wedge}(c^v) = 0$. Then construct $|a\rangle$ as follows. Set $a_v = 0$ for all children v of p, and let $|a_{T_c}\rangle = 0$ for every grandchild $c \notin \{c^v : v \text{ is a child of } p\}$. By induction, for each v construct $|\tilde{a}_{T_{c^v}}\rangle$ satisfying $\||\tilde{a}_{T_{c^v}}\rangle\| = 1$, $H_{T_{c^v}}|\tilde{a}_{T_{c^v}}\rangle = 0$ and $\tilde{a}_{c^v} \geq 1/(\sqrt{\sigma_-(c^v)}s_{c^v}^{1/4})$.

For each v, in order to satisfy $\langle v|H|a\rangle = 0$, we need $h_{pv}a_p = -h_{vc^v}a_{c^v}$. To achieve this, we *rescale* the vectors $|\tilde{a}_{T_v}\rangle$. Let $a_p = 1$, and let $|a_{T_{c^v}}\rangle = -\frac{h_{pv}}{h_{vc^v}}\frac{1}{\tilde{a}_{c^v}}|\tilde{a}_{T_{c^v}}\rangle$. It remains only to verify that $\||a\rangle\|^2 \leq \sqrt{s_p}\sigma_-(p)$, so that when we renormalize, $a_p/\||a\rangle\| = 1/\||a\rangle\|$ is still large. Indeed,

$$\begin{aligned}
\||a\rangle\|^2 &= a_p^2 + \sum_v \frac{h_{pv}^2}{h_{vc^v}^2(\tilde{a}_{c^v})^2}\||\tilde{a}_{T_{c^v}}\rangle\|^2 \\
&\leq 1 + \sum_v \frac{h_{pv}^2}{h_{vc^v}^2}\sqrt{s_{c^v}}\sigma_-(c^v) \\
&= 1 + \sum_v \frac{s_v}{\sqrt{s_p}}\sigma_-(c^v) \\
&\leq \sqrt{s_p}\left(\frac{1}{\sqrt{s_p}} + \max_v \sigma_-(c^v)\right) \\
&\leq \sqrt{s_p}\sigma_-(p). \qquad \square
\end{aligned}$$

The key step in the above proof, which motivates the choice of weights h_{pv}, is $\sum_v s_v = s_p$.

Lemma 4 is a strong converse of Lemma 3, as it does not merely assert that a_p can be set nonzero; it also puts a quantitative lower bound on the achievable magnitude. Lemma 4 lets us say that there *exists* a zero-eigenvalue eigenstate with large overlap on the root r when $\overline{\wedge}(r) = 0$.

Now in the case $\varphi(x) = \overline{\wedge}(r) = 0$, let us extend $|a_{T_r}\rangle$ from Lemma 4 into a zero-eigenvalue eigenvector $|a\rangle$ over the whole tree, to see that the overlap $|\langle r''|a\rangle|/\||a\rangle\|$ is large. In order to satisfy $H|a\rangle = 0$, we must have $a_{r'} = 0$ and $-a_{r''} = \frac{h_{r'r}}{h_{r''r'}}a_r = \sqrt{\sigma_-(\varphi)}N^{1/4}a_r \geq 1$. Therefore, we lower bound

$$\frac{|\langle r''|a\rangle|}{\||a\rangle\|} \geq \frac{1}{\sqrt{1 + \||a_{T_r}\rangle\|^2}} = \frac{1}{\sqrt{2}}.$$

This completes the proof of the first half of Theorem 2.

5. Spectral gap of H in the case $\varphi(x) = 1$. To prove the second half of Theorem 2, we must consider the case $\varphi(x) = 1$ and investigate the eigenvectors of H corresponding to eigenvalues E close to zero. As T is a bipartite graph, the spectrum of H is symmetric around zero. Let

$$|E\rangle = \sum_v a_v|v\rangle$$

be an eigenvector of H with eigenvalue $E > 0$.

From (3) we obtain

$$\langle v|H|E\rangle = E\alpha_v = h_{pv}a_p + \sum_c h_{vc}a_c. \tag{5}$$

The analysis depends on the fact that a_v/a_p is either large or small in magnitude, depending on whether $\overline{\wedge}(v) = 0$ or 1.

LEMMA 5. *Let* $0 < E \le 1/\left(5\sigma_-(\varphi)\sqrt{\sigma_+(\varphi)}\right)$. *For vertices* $v \ne r''$ *in* T, *define* y_{0v} *and* y_{1v} *by*

$$\begin{aligned} y_{0v} &= (1+k_v)\sigma_-(v)\sqrt[4]{s_v s_p}, \\ y_{1v} &= (1+k_v)\sigma_-(v)\frac{s_v^{3/4}}{s_p^{1/4}}, \end{aligned} \tag{6}$$

where p *is the parent of* v *and* k_v *is defined by*

$$k_v = 4E^2\sigma_-(v)^2\sigma_+(v).$$

Then for every vertex $v \ne r''$ *in* T, *either* $a_v = a_p = 0$ *or*

$$\begin{aligned} \overline{\wedge}(v) = 0 \quad &\Rightarrow \quad 0 < a_p/a_v \le y_{0v}E, \\ \overline{\wedge}(v) = 1 \quad &\Rightarrow \quad 0 > a_v/a_p \ge -y_{1v}E. \end{aligned} \tag{7}$$

Note that, because of our assumption on E, we always have $k_v \le \frac{4}{25} = 0.16$.

Proof. The proof is by induction on the height of the tree. For the base case, we have $\overline{\wedge}(v) = 0$ for every leaf v, and by (5), $Ea_v = h_{pv}a_p$. Thus either $a_v = a_p = 0$ or

$$\frac{a_p}{a_v} = \frac{E}{h_{pv}} = \sqrt[4]{\frac{s_p}{s_v}}E = \sqrt[4]{s_p s_v}E \le y_{0v}E.$$

The induction proceeds as follows:

- If $\overline{\wedge}(v) = 0$, then all children c of v evaluate to $\overline{\wedge}(c) = 1$. First assume $a_v \ne 0$. Dividing both sides of (5) by $a_v h_{pv}$, using the induction hypothesis, and rearranging terms gives

$$\begin{aligned} \frac{a_p}{a_v} &= \frac{1}{h_{pv}}\left(E - \sum_c h_{vc}\frac{a_c}{a_v}\right) \\ &\le \frac{1}{h_{pv}}\left(1 + \sum_c h_{vc}y_{1c}\right)E. \end{aligned}$$

  Using the inductive assumption about y_{1c} and substituting the expressions for h_{pv}, h_{vc} in terms of a_p, a_v, a_c, we can upper bound the coefficient of E by

$$\frac{1}{h_{pv}} + \sum_c (1+k_c)\sigma_-(c)\frac{s_c s_p^{1/4}}{s_v^{3/4}}.$$

  Since $\sum_c s_c = s_v$ and $k_c \le k_v$ for any c, this is at most

$$\begin{aligned} &(1+k_v)\left(\sqrt[4]{\frac{s_p}{s_v}} + \max_c \sigma_-(c)\sqrt[4]{s_v s_p}\right) \\ &= (1+k_v)\sqrt[4]{s_v s_p}\left(\max_c \sigma_-(c) + \frac{1}{\sqrt{s_v}}\right) \\ &= (1+k_v)\sigma_-(v)\sqrt[4]{s_v s_p}. \end{aligned}$$

  The induction hypothesis also gives that $a_p/a_v \ge E/h_{pv} > 0$. If $a_v = 0$, then the induction hypothesis gives that all a_c are zero, so also $a_p = 0$ by (5).

- If $\overline{\wedge}(v) = 1$, then there is at least one child c with $\overline{\wedge}(c) = 0$. We may assume $a_v \neq 0$, since otherwise $a_p = 0$ and the condition holds trivially. Then, again dividing (5) by $a_v h_{pv}$ and using the induction hypothesis gives

$$\begin{aligned}\frac{a_p}{a_v} &= \frac{E}{h_{pv}} - \sum_c \frac{h_{vc}}{h_{pv}} \frac{a_c}{a_v} \\ &\leq \frac{E}{h_{pv}} + \sum_{c:\overline{\wedge}(c)=1} \frac{h_{vc} y_{1c}}{h_{pv}} E - \sum_{c:\overline{\wedge}(c)=0} \frac{h_{vc}}{h_{pv} y_{0c} E}. \qquad (8)\end{aligned}$$

Because of the previous case, we can upper bound the first sum by

$$\begin{aligned}\sum_c \frac{h_{vc} y_{1c}}{h_{pv}} E &\leq \max_c (1 + k_c)\sigma_-(c) \sqrt[4]{s_v s_p} E \\ &\leq (1 + k_v)\sigma_-(v) \sqrt[4]{s_v s_p} E. \qquad (9)\end{aligned}$$

We lower bound the second sum by one of its terms (since there is at least one c with $\overline{\wedge}(c) = 0$):

$$\frac{h_{vc}}{h_{pv} y_{0c} E} \geq \frac{s_p^{1/4}}{(1 + k_c)\sigma_-(v) s_v^{3/4} E}.$$

Finally, the first term on the right-hand side of (8) is $\frac{E}{h_{pv}} = \sqrt[4]{\frac{s_p}{s_v}} E$, which is less than the right-hand side of (9). Therefore, (8) is at most

$$\begin{aligned}&2(1 + k_v)\sigma_-(v) \sqrt[4]{s_v s_p} E - \frac{s_p^{1/4}}{(1 + k_c)\sigma_-(v) s_v^{3/4} E} \\ &\quad = \frac{-s_p^{1/4} \left[1 - 2(1 + k_v)(1 + k_c)\sigma_-(v)^2 s_v E^2\right]}{(1 + k_c)\sigma_-(v) s_v^{3/4} E}.\end{aligned}$$

Let $\delta = \sigma_-(v)^2 s_v E^2$. Since $k_c \leq k_v \leq 0.16$, we can lower bound the expression in square brackets by

$$1 - 2 \cdot 1.16^2 \delta \geq 1 - 2.7\delta.$$

This means that

$$\frac{a_p}{a_v} \leq \frac{-s_p^{1/4}}{(1 + k_c)\sigma_-(v) s_v^{3/4} E} (1 - 2.7\delta).$$

To complete the proof that $\frac{a_p}{a_v} \leq -\frac{1}{y_{1v} E}$ (and hence $\frac{a_v}{a_p} \geq -y_{1v} E$), it suffices to show that

$$\frac{1 + k_c}{1 - 2.7\delta} \leq 1 + k_v.$$

We have

$$\begin{aligned}\frac{1 + k_c}{1 - 2.7\delta} &= 1 + k_c + (1 + k_c)\left(\frac{1}{1 - 2.7\delta} - 1\right) \\ &\leq 1 + k_c + 1.16\left(\frac{1}{1 - 2.7\delta} - 1\right). \qquad (10)\end{aligned}$$

We now observe that $\delta \le \sigma_-(v)^2\sigma_+(v)E^2 \le 0.04$. If $0 \le \delta \le 0.04$, the last term of (10) is always upper bounded by 4δ. Therefore, the entire right-hand side of (10) is upper bounded by

$$\begin{aligned} 1 + k_c + 4\delta &= 1 + 4\sigma_-(v)^2(\sigma_+(c) + s_v)E^2 \\ &\le 1 + k_v. \qquad \square \end{aligned}$$

Now we are ready to complete the proof of the second half of Theorem 2. Assume $|E\rangle$ is an eigenvector of H with eigenvalue $E \in \left(0, 1/\left(5\sigma_-(\varphi)\sqrt{\sigma_+(\varphi)}\right)\right]$. We want to show $a_{r'} = a_{r''} = 0$. We have

$$\begin{aligned} Ea_{r'} &= h_{r'r''}a_{r''} + h_{rr'}a_r \\ &\ge \frac{h_{r'r''}^2}{E}a_{r'} - h_{rr'}y_{1r}Ea_{r'}, \end{aligned}$$

with the inequality following from $Ea_{r''} = h_{r'r''}a_{r'}$ and Lemma 5. If $a_{r'} \neq 0$, we can divide both sides by $Ea_{r'}$. Then, moving the second term from the right-hand side to the left gives us

$$1 + h_{rr'}y_{1r} \ge \frac{h_{r'r''}^2}{E^2}.$$

Substituting the values of $h_{rr'}$ and $h_{r'r''}$ and applying the assumed upper bound on E gives us

$$1 + y_{1r} \ge \frac{25\sigma_-(\varphi)\sigma_+(\varphi)}{\sqrt{N}} \ge 25\sigma_-(\varphi)\sqrt{N}. \tag{11}$$

By Lemma 5, we have

$$y_{1r} = (1 + k_r)\sigma_-(\varphi)\frac{s_r^{3/4}}{s_{r'}^{1/4}} \le 1.16\,\sigma_-(\varphi)\sqrt{N}.$$

Substituting this into (11) gives a contradiction.

Therefore, $a_{r'} = 0$, and, because $Ea_{r''} = h_{r'r''}a_{r'}$, we also have $a_{r''} = 0$. Together with Lemma 3 for the case when $E = 0$, this completes the proof of the second half of Theorem 2.

6. Discrete-time quantum walk. As mentioned in section 2, one way of obtaining a formula evaluation algorithm is to apply phase estimation directly to the continuous-time quantum walk generated by H. However, simulating that walk has an overhead that can be avoided by instead considering a corresponding discrete-time quantum walk.

To construct this walk, we first briefly review Szegedy's procedure for quantizing classical random walks. Theorem 6, adapted from [Sze04] (see also [MNRS07]), relates the eigensystem of the discrete-time quantum walk to that of the original classical walk.

THEOREM 6 (see [Sze04]). *Let $\{|v\rangle : v \in V\}$ be an orthonormal basis for $\mathcal{H}_V$. For each $v \in V$, let*

$$|\tilde{v}\rangle = |v\rangle \otimes \sum_{w \in V} \sqrt{p_{vw}}|w\rangle \in \mathcal{H}_V \otimes \mathcal{H}_V,$$

where $p_{vw} \geq 0$ *and* $\langle\tilde{v}|\tilde{v}\rangle = \sum_w p_{vw} = 1$. *Let* $T = \sum_v |\tilde{v}\rangle\langle v|$ *be an isometry mapping* $|v\rangle$ *to* $|\tilde{v}\rangle$, *and let* $\Pi = TT^\dagger = \sum_v |\tilde{v}\rangle\langle\tilde{v}|$ *denote the projection onto the span of the* $|\tilde{v}\rangle$*s. Let* $S = \sum_{v,w} |v,w\rangle\langle w,v|$ *denote the swap operator. Let*

$$M = T^\dagger ST = \sum_{v,w\in V} \sqrt{p_{vw}p_{wv}}|v\rangle\langle w|$$

(a real symmetric matrix), and let $\{|\lambda_a\rangle\}$ *denote a complete set of orthonormal eigenvectors of* M *with corresponding eigenvalues* λ_a.

The spectral decomposition of $U = (2\Pi - 1)S$ *is determined by that of* M *as follows. Let* $R_a = \mathrm{span}\{T|\lambda_a\rangle, ST|\lambda_a\rangle\}$. *Then* $R_a \perp R_{a'}$ *for* $a \neq a'$; *let* $R = \bigoplus_a R_a$. U *fixes the spaces* R_a *and is* $-S$ *on* $R^\perp$. *The eigenvectors of* U *within* R_a *are given by* $(1 - \beta_{a,\mp}S)T|\lambda_a\rangle$, *with corresponding eigenvalues* $\beta_{a,\pm} = \lambda_a \pm i\sqrt{1-\lambda_a^2}$.

Proof. First assume $a \neq a'$, and let us show $R_a \perp R_{a'}$. Indeed, $\langle\lambda_a|T^\dagger T|\lambda_{a'}\rangle = \langle\lambda_a|\lambda_{a'}\rangle = 0$, as $T^\dagger T = 1$. Since $S^2 = 1$, similarly, $ST|\lambda_a\rangle$ is orthogonal to $ST|\lambda_{a'}\rangle$. Finally, $\langle\lambda_a|T^\dagger ST|\lambda_{a'}\rangle = \langle\lambda_a|M|\lambda_{a'}\rangle = 0$. Therefore, the decomposition $\mathcal{H}_V \otimes \mathcal{H}_V = (\bigoplus_a R_a) \oplus R^\perp$ is well defined.

R is the span of the images of ST and T. $2\Pi - 1$ is $+1$ on the image of T and -1 on its complement; therefore, U is $-S$ on $R^\perp$.

Finally, $\Pi T = TT^\dagger T = T$ and $\Pi ST = TT^\dagger ST = TM$, so

$$U(ST|\lambda_a\rangle) = (2\Pi - 1)T|\lambda_a\rangle = T|\lambda_a\rangle,$$
$$U(T|\lambda_a\rangle) = (2\Pi - 1)ST|\lambda_a\rangle = (2\lambda_a - S)T|\lambda_a\rangle;$$

U fixes the subspaces R_a. To determine its eigenvalues on R_a, let $|\beta\rangle = (1+\beta S)T|\lambda_a\rangle$. Then $U|\beta\rangle = (2\lambda_a + \beta)T|\lambda_a\rangle - ST|\lambda_a\rangle$ is proportional to $|\beta\rangle$ if $\beta(2\lambda_a + \beta) = -1$; i.e., $\beta = -\lambda_a \pm i\sqrt{1-\lambda_a^2}$. (If $\lambda_a = \pm 1$, note that $T|\lambda_a\rangle = \pm ST|\lambda_a\rangle$, so R_a is one-dimensional, corresponding to a single eigenvector of U.) ∎

To connect this theorem to classical and quantum walks, start with an undirected graph $G = (V, E)$. Choose the $p_{v,w}$ to be the transition probabilities $v \to w$ of a classical random walk on this graph (i.e., with the constraint $p_{v,w} = 0$ if $(v,w) \notin E$). Then $U = (2\Pi - 1)S$ can be considered a quantization of the classical walk, taking place on the *directed edges* of G. First, the swap S switches the direction of an edge. Then, when the first register is $|v\rangle$, $2\Pi - 1$ acts as a reflection about $|\tilde{v}\rangle = |v\rangle \otimes \sum_{w\sim v} \sqrt{p_{v,w}}|w\rangle$; it is a "coin flip" that mixes the directed edges leaving v. Therefore, although U acts on $\mathcal{H}_V \otimes \mathcal{H}_V$, it preserves the subspace spanned by $|v,w\rangle$ and $|w,v\rangle$ for $(v,w) \in E$. An alternative basis for this subspace is to give a vertex v together with an edge index to describe an edge leaving v. If the graph has maximum degree D, then U can be implemented on $\mathcal{H}_V \otimes \mathbb{C}^D$ instead of $\mathcal{H}_V \otimes \mathcal{H}_V$.

Discretization of continuous-time quantum walks. Szegedy's Theorem 6 relates the eigenvalues and eigenvectors of the quantum walk U to those of the matrix $M = \sum_{v,w} \sqrt{p_{v,w}p_{w,v}}|v\rangle\langle w|$. If $P = \sum_{v,w} \sqrt{p_{v,w}}|v\rangle\langle w|$ is the elementwise square root of the transition matrix of a classical random walk, then M is the elementwise product $P \circ P^T$. But M can also be regarded as the Hamiltonian for a continuous-time quantum walk on the vertices of the underlying graph.

In our case, we are given H and desire a factorization $H = h\,P \circ P^T$ for some normalization factor $h > 0$ such that P has all row norms exactly one. Then Theorem 6

with $M = H/h$ relates the eigensystem of H to that of the discrete-time quantum walk corresponding to P. To obtain this factorization, it is convenient to enlarge the Hilbert space by adding one isolated vertex to the graph, denoted $\varnothing$. Then, for a large class of Hamiltonians, we can choose P as follows.

CLAIM 7. *Let H be an $N \times N$ symmetric matrix with nonnegative entries. Define $\|H\|_1 = \max_v \sum_w |H_{v,w}|$, and suppose that $h \geq \|H\|_1$. Then $P \circ P^T = (H \oplus 0)/h$, where P is the $(N+1) \times (N+1)$ matrix with nonnegative entries $P_{v,w} = \sqrt{H_{v,w}/h}$, $P_{v,\varnothing} = \sqrt{1 - \sum_w H_{v,w}/h}$, $P_{\varnothing,w} = 0$, and $P_{\varnothing,\varnothing} = 1$. By construction, all row norms of P are equal to one.*

Remark 2. Szegedy's Theorem 6, with Claim 7, serves as a general method for relating an arbitrary positive-weighted continuous-time quantum walk on the vertices of G to a discrete-time quantum walk on the directed edges of G. In particular, the eigenvalues of the discrete walk $-iU$ are given by $\pm\sqrt{1-\lambda_a^2} + i\lambda_a$ (i.e., $e^{i \arcsin \lambda_a}$ and $-e^{-i \arcsin \lambda_a}$), whereas the continuous walk e^{iM} has eigenvalues $e^{i\lambda_a}$. The spectral gaps from zero of the continuous walk and the discrete walk are equal up to third order.

7. The algorithm. We now establish the main result.

THEOREM 1. *Let φ be an arbitrary AND-OR formula of size N. After efficient (i.e., time $\mathrm{poly}(N)$) classical preprocessing that does not depend on the input x, $\varphi(x)$ can be evaluated with error at most $1/3$ using $N^{1/2+O(1/\sqrt{\log N})}$ queries to O_x. The running time is also $N^{1/2+O(1/\sqrt{\log N})}$, assuming unit-cost coherent access to the result of the preprocessing. For an approximately balanced formula (see Definition 2), the query complexity is only $O(\sqrt{N})$ and the running time is only $\sqrt{N}(\log N)^{O(1)}$.*

By "unit-cost coherent access," we mean the following. Our algorithm begins by classically preprocessing the formula, giving some string y. We assume that there exists an oracle O_y for accessing y as in (1) such that applying O_y takes unit time.

Proof. The proof of Theorem 1 is as follows.

Preprocessing. First, apply Lemma 8 to expand out gates so each NAND gate has $O(1)$ fan-in.

LEMMA 8. *For any NAND formula φ, one can efficiently construct an equivalent NAND formula φ' of the same size such that all NAND gates have fan-in at most two, $\sigma_+(\varphi') = O(\sigma_+(\varphi))$, and $\sigma_-(\varphi') = O(\sigma_-(\varphi))$.*

Bounding the gate fan-in is needed to bound the norm of the weighted adjacency matrix from Definition 3. A proof of Lemma 8 is given in the appendix.

Next, if $\sigma_-(\varphi)\sqrt{\sigma_+(\varphi)} = N^{\frac{1}{2}+\omega(1/\sqrt{\log N})}$, then apply the formula rebalancing procedure of [BCE95, BB94] with parameter k to be determined.

LEMMA 9 (see [BB94, Theorem 4]). *For any NAND formula φ of size N and for all $k \geq 2$, one can efficiently construct an equivalent NAND formula φ' with gate fan-ins at most two and satisfying*[3]

$$\mathrm{depth}(\varphi') \leq (9 \ln 2) k \log_2 N,$$

$$\mathrm{size}(\varphi') \leq N^{1+1/\log_2 k}.$$

Let φ' be the preprocessed formula, and let H be the Hamiltonian corresponding to φ' according to Definition 3. We would like to define U as a discrete-time quantum walk corresponding to $H/n(H)$ (where $n(H)$ is some upper bound on $\|H\|_1$) via

[3]The constant in the depth bound is $9 \ln 2$ instead of the $3 \ln 2$ in [BB94, Theorem 4] because we lose a constant converting an {AND, OR, NOT} formula to a NAND formula.

Claim 7. Obtaining this U takes a little care, since H depends on the oracle. Let H_0 denote the Hamiltonian from Definition 3, assuming that all leaves evaluate to 0. By applying Claim 7 as part of the preprocessing, we obtain a U_0 corresponding to $H_0/\|H_0\|_1$. Then let $U = O_x U_0$, where O_x is the oracle for the input as in (1), acting on the leaf vertices. Since O_x introduces a phase of $(-1)^{x_i}$ conditioned on the current vertex being leaf i, this U is also a discrete-time quantum walk as in Theorem 6. We claim that in Theorem 6, U corresponds to $M = H/\|H_0\|_1$. To see this, note that the only difference between U and U_0 is on the $|\tilde{i}\rangle$s for leaf vertices i with $x_i = 1$: in U_0, $|\tilde{i}\rangle = |i, p\rangle$ (where p is the parent of i), whereas in U, $|\tilde{i}\rangle = |i, i\rangle$ (i.e., $p_{i,i} = 1$). Therefore, the M corresponding to U differs only from H_0 in the coefficients involving leaves i with $x_i = 1$ and $\langle i|M|p\rangle = \langle\tilde{i}|S|\tilde{p}\rangle = 0$, so $M = H/\|H_0\|_1$ as claimed. Note that since the tree has positive weights and constant maximum degree, $\|H_0\|_1 = O(\|H_0\|) = O(1)$.

For each vertex v, compute a sequence of $O((\log N)^2 \log\log N)$ elementary gates that approximate to within $1/N$ the reflection about $|\tilde{v}\rangle$ induced by U_0, using the Solovay–Kitaev theorem [KSV02]. (With this approximation, the algorithm's total error probability increases only by $o(1)$.) Store the descriptions of these gate sequences in a classical string, which we assume the algorithm can access coherently at unit cost. To apply U_0 at vertex v, the algorithm looks up the corresponding gate sequence and applies it to the coin register $|c\rangle$.

Algorithm.

1. Prepare $|\tilde{r}''\rangle = |r'', r'\rangle$.
2. "Measure the energy according to H." In other words, apply quantum phase estimation to $-iU = -iO_x U_0$. Use precision $\delta_p = 1/(10\sigma_-(\varphi')\sqrt{\sigma_+(\varphi')})$ and error probability $\delta_e \le 1/4$.
3. Output 0 if and only if the measured phase is 0 or π.

Figure 2 lays out the steps of the algorithm in complete detail for the case of a balanced binary NAND tree. We did not use Claim 7 to derive U in Figure 2, because in this special case it is clear that applying Theorem 6 to U gives H, except with larger weights to leaves evaluating to 0 (see Remark 1).

Correctness. The correctness follows from Theorems 2 and 6. If $\varphi(x) = 0$, then there exist two eigenvectors of U given by $(1 \pm iS)T|a\rangle$ with eigenvalues $\pm i$, respectively. Their overlaps with the initial state $|\tilde{r}''\rangle$ are $|\langle\tilde{r}''|(1 \pm iS)T|a\rangle| = |a_{r''} \pm i a_{r'} \frac{h_{r'',r'}}{\|H\|}| \ge 1/\sqrt{2} - O(1/N^{1/4})$. Since the norm of $(1 \pm iS)T|a\rangle$ is at most $2\||a\rangle\| = 2$, we find that the probability of outputting 0 is at least $2\big(\frac{1}{2}(\frac{1}{\sqrt{2}} - o(1))\big)^2 = 1/4 - o(1)$.

Conversely, if $\varphi(x) = 1$, then every eigenstate of H with support on r' or r'' has an eigenvalue at least $1/(5\sigma_-(\varphi')\sqrt{\sigma_+(\varphi')})$ in magnitude. Every eigenstate of U with support on $|r'', r'\rangle = |\tilde{r}''\rangle = T|r''\rangle$ must be of the form $(1 + \beta_{a,\pm}S)T|\lambda_a\rangle = (1 + (-\lambda_a \pm i\sqrt{1-\lambda_a^2})S)T|\lambda_a\rangle$. The terms which can overlap $T|r''\rangle$ are either $\langle r''|\lambda_a\rangle$ (via T) or $\langle r'|\lambda_a\rangle$ (via ST). But for $|\lambda_a| < 1/(10\sigma_-(\varphi')\sqrt{\sigma_+(\varphi')})$, both coefficients must be zero. Therefore, our algorithm outputs 0 with probability less than $\delta_e < 1/4$. This constant gap can be amplified as usual.

Query and time complexity. To obtain precision δ_p and error probability at most δ_e, phase estimation requires applying $O(\|H_0\|/(\delta_p\delta_e)) = O\big(\sigma_-(\varphi')\sqrt{\sigma_+(\varphi')}\big)$ controlled-U operations [CEMM98].

- If φ is approximately balanced, this is $O(\sqrt{N})$, as desired.

- Otherwise, due to the bounds on $\sigma_\pm$ below Definition 1,

$$\begin{aligned}\sigma_-(\varphi')\sqrt{\sigma_+(\varphi')} &= O\big(\sqrt{\operatorname{size}(\varphi')}\operatorname{depth}(\varphi')^{3/2}\big)\\ &= O(N^{\frac{1}{2}(1+1/\log_2 k)}(k\log_2 N)^{3/2})\end{aligned}$$

from Lemma 9 with parameter k. Setting $k = 2^{\sqrt{(\log_2 N)/3}}$ optimizes this bound, giving $N^{1/2+\sqrt{(3+o(1))/\log N}}$ queries to O_x.

Finally, since the Solovay–Kitaev theorem produces a sequence of only $(\log N)^{O(1)}$ elementary gates that approximate each walk step within $1/N$, the total running time is only polylogarithmically larger than the number of queries. □

8. Evaluating iterated functions. Our algorithm can be used to evaluate an arbitrary Boolean function by first writing a NAND formula for the function and then evaluating that formula. For functions defined recursively, this approach is particularly natural. Of course, it will only be advantageous when the formula size is sufficiently small.

In particular, our algorithm gives improved upper bounds for evaluating an iterated "all-equal" function studied in [Amb06]. Define $f : \{0,1\}^3 \to \{0,1\}$ by $f(x_1,x_2,x_3) = 1$ if $x_1 = x_2 = x_3$ and $f(x_1,x_2,x_3) = 0$ otherwise. Define a sequence of iterated functions $f_n : \{0,1\}^{3^n} \to \{0,1\}$, for $n = 1,2,3,\ldots$, by $f_1 = f$ and

$$f_n(x_1,\ldots,x_{3^n}) = f_{n-1}\big(f(x_1,x_2,x_3), f(x_4,x_5,x_6),\ldots,f(x_{3^n-2},x_{3^n-1},x_{3^n})\big)$$

for $n \geq 2$. The functions f_n have attracted interest in the context of relating polynomial degree and quantum query complexity of Boolean functions. The polynomial degree of a Boolean function f is always a lower bound on its quantum query complexity [BBC+01]. The function f_n is one of the known functions for which this lower bound is not optimal. The polynomial degree of f_n is 2^n, while the quantum query complexity of f_n is lower bounded by $\Omega\big((\frac{3}{\sqrt{2}})^n\big) = \Omega(2.12^n)$ [Amb06].

Our approach gives the first quantum algorithm for evaluating f_n with $o(3^n)$ queries. To see this, note that f can be represented by a NAND formula of size six as follows:

$$f(x_1,x_2,x_3) = \overline{\wedge}(\overline{\wedge}(x_1,x_2,x_3), \overline{\wedge}(\bar{x}_1,\bar{x}_2,\bar{x}_3)).$$

Using this formula, we can inductively construct a size-6^n NAND formula for f_n. Namely, suppose we are given a NAND formula for f_{n-1} with size 6^{n-1}. Then substituting for each variable the size-six NAND formula for f gives a NAND formula for f_n with size 6^n. The formula is clearly approximately balanced. Thus, using our algorithm for NAND tree evaluation, we can evaluate f_n using $O(\sqrt{6^n}) = O(2.45^n)$ quantum queries.

Another function for which our algorithm gives an improved upper bound is the iterated three-majority function. Let $g(x_1,x_2,x_3)$ be 1 if $x_1 + x_2 + x_3 \geq 2$ and 0 otherwise. Classically, the query complexity of evaluating the iterated function g_n is only known to lie between $\Omega\big((7/3)^n\big)$ and $O\big((2.6537\ldots)^n\big)$ [JKS03], and no better quantum algorithm was known. However, since $g(x_1,x_2,x_3) = (x_1 \wedge x_2) \vee ((x_1 \vee x_2) \wedge x_3)$, the function g_n can be evaluated in $O(\sqrt{5^n}) = O(2.24^n)$ quantum queries.

Two of the authors have developed a formula evaluation algorithm that in particular evaluates the above two iterated formulas optimally [RŠ08]. This quantum algorithm uses $O\big((3/\sqrt{2})^n\big)$ queries to evaluate f_n and $O(2^n)$ queries to evaluate g_n.

9. Open problems. We conclude by mentioning some open problems.

- Our algorithm needs to know the full structure of the formula beforehand to determine the quantum walk transition amplitudes (i.e., the biases of the "quantum coin") at each internal vertex. However, these values need not be computed exactly, because there is some freedom in the recurrence on y_{0v} and y_{1v}. It would be interesting to know if a different choice of coefficients, or a relaxed calculation thereof, would allow for faster preprocessing. Furthermore, it would be interesting to know on what kinds of structured inputs the preprocessing can be done in time $N^{1/2+o(1)}$.
- Numerical simulations indicate that the formula can be evaluated by running the quantum walk from the initial state and measuring whether the final quantum state has large overlap with $\frac{1}{\sqrt{2}}(|r', r''\rangle + |r'', r'\rangle)$ or $\frac{1}{\sqrt{2}}(|r', r''\rangle - |r'', r'\rangle)$. If this is true, then we can avoid the phase estimation on top of the quantum walk, simplifying the algorithm.
- What kinds of noisy oracles can this algorithm, or an extended version, tolerate? For example, [HMW03] extends a Grover search to the case where input values are computed by a bounded-error quantum subroutine.
- Are there hard instances of formulas for which the rebalancing provided by Lemma 9 is tight? If so, are these also hard instances for our algorithm? For example, the most unbalanced formula $\varphi(x) = x_1 \,\overline{\wedge}(x_2 \,\overline{\wedge}(x_3 \,\overline{\wedge}(\ldots \overline{\wedge}\, x_N)))$ is not a hard instance. It can be rebalanced by a different procedure, giving an equivalent formula with depth $O(\log N)$ and size $O(N \log N)$ and can be evaluated with $O(\sqrt{N})$ queries.

Appendix A. Proof of Lemma 8. Let φ be a NAND formula, and consider a NAND gate in φ with fan-in $k > 2$. Rewrite this gate as a NOT gate, i.e., a NAND gate of fan-in one, applied to an AND gate of fan-in k. Let ψ denote the subformula rooted at that AND gate. Then the following recursive procedure can be used to expand the AND gate into a tree of AND gates, each of fan-in two.

PROCEDURE EXPAND(ψ).

Input: A formula ψ that begins with an AND gate of fan-in k.
Output: An equivalent formula in which the original AND gate of fan-in k is implemented by AND gates of fan-in at most two.

1. If $k \le 2$, then return ψ.
2. Otherwise, let $\psi_1, \ldots, \psi_k$ be the input subformulas of sizes $s_1 \ge \cdots \ge s_k$. Let $i^* = \min\{i : \sum_{j=1}^{i^*} s_j \ge \alpha \sum_{j=1}^{k} s_j\}$, where $\alpha = \frac{1}{2}(3 - \sqrt{5})$. Return

$$\mathrm{AND}\big(\mathrm{Expand}\big(\,\mathrm{AND}(\psi_1, \ldots, \psi_{i^*})\big), \mathrm{Expand}\big(\,\mathrm{AND}(\psi_{i^*+1}, \ldots, \psi_k)\big)\big).$$

Since the input subformulas are ordered by their sizes, note that $i^* < k$, so the procedure terminates.

LEMMA 10. *In the tree of AND gates returned by Expand(ψ), the size of the grandparent of any AND gate v_1 is at least a factor of* $\min\big(\frac{1}{1-\alpha}, \frac{1}{\sqrt{\alpha}}\big) = \frac{1+\sqrt{5}}{2}$ *larger than the size of v_1.*

Proof. Let $f = \frac{1}{\sqrt{\alpha}} - 1$. Order the leaves of the tree from left to right with s_1 on the left and s_k on the right. Consider a node v_1 inside the tree, with size S_1. Let v_2 be v_1's sibling, with size S_2.

- If v_2 is to the left of v_1, then by construction $S_2 \ge \alpha(S_1 + S_2)$, so $\frac{S_1+S_2}{S_1} \ge \frac{1}{1-\alpha}$.

- If v_2 is to the right of v_1 with size $S_2 < fS_1$, then we have no good lower bound on $\frac{S_1+S_2}{S_1}$. Let p be the parent of v_1 and v_2, with sibling v_3 having size S_3. We claim that $S_3 \geq \min\left(\frac{\alpha}{1-\alpha}, f\right) \cdot (S_1 + S_2)$. To see this, assume otherwise, that v_3 is to the right of p_1 with size $S_3 < f(S_1+S_2)$. Let s_i be the rightmost leaf above v_2. Thus $S_1+S_2-s_i < \alpha(S_1+S_2+S_3)$. Rearranging this equation,

$$\begin{aligned} s_i &> (1-\alpha)(S_1+S_2) - \alpha S_3 \\ &> (1-\alpha-\alpha f)(S_1+S_2) \\ &> (1-\alpha-\alpha f)\Big(\frac{1}{f}+1\Big)S_2 \\ &= S_2, \end{aligned}$$

  which is a contradiction.

Thus, $\max\left(\frac{S_1+S_2+S_3}{S_1+S_2}, \frac{S_1+S_2}{S_1}\right) \geq \min\left(\frac{1}{1-\alpha}, \frac{1}{\sqrt{\alpha}}\right)$, as claimed. ∎

Proof of Lemma 8. Let $\tilde{T}$ be the tree of AND gates returned by a call to Expand(ψ) with input subformulas $\psi_1, \ldots, \psi_k$ of sizes $s_1, \ldots, s_k$, with $S = \sum_{i=1}^{k} s_i$. Let ξ_i be the path from the root AND gate up to the subformula ψ_i. Then by Lemma 10, letting $\tau = \min\left(\frac{1}{1-\alpha}, \frac{1}{\sqrt{\alpha}}\right) = \frac{1+\sqrt{5}}{2}$,

$$\sum_{w \in \xi_i} s_w < 2S \sum_{j=0}^{\infty} \tau^{-j} < 6S$$

and

$$\sum_{w \in \xi_i} \frac{1}{\sqrt{s_w}} < \frac{2}{\sqrt{s_i}} \sum_{j=0}^{\infty} \tau^{-j/2} < \frac{10}{\sqrt{s_i}}.$$

After calling Expand on every NAND gate in φ with fan-in more than two, rewrite each AND gate as a NOT gate on a NAND gate, and let φ' be the resulting NAND formula. The above equations imply that $\sigma_+(\varphi') < 12\sigma_+(\varphi)$ and $\sigma_-(\varphi') < 20\sigma_-(\varphi)$. This completes the proof. ∎

Acknowledgments. R. Špalek would like to thank the Institute for Quantum Information at Caltech for hospitality. We thank Ronald de Wolf for comments on an early draft of the paper, and we thank Jérémie Roland for thoughtful remarks.

This work was conducted while A. Ambainis was at the University of Waterloo, A. M. Childs, B. W. Reichardt, and S. Zhang were at the California Institute of Technology, and R. Špalek was at the University of California, Berkeley.

REFERENCES

[ACLT09] A. Ambainis, A. M. Childs, F. Le Gall, and S. Tani, *The quantum query complexity of certification*, Quantum Information and Computation, to appear.

[ACR+07] A. Ambainis, A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, *Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer*, in Proceedings of the 48th IEEE FOCS, 2007, pp. 363–372.

[Amb06] A. Ambainis, *Polynomial degree vs. quantum query complexity*, J. Comput. System Sci., 72 (2006), pp. 220–238.

[Amb07] A. Ambainis, *A nearly optimal discrete query quantum algorithm for evaluating NAND formulas*, arXiv quant-ph/0704.3628, 2007.

[BB94] M. L. Bonet and S. R. Buss, *Size-depth tradeoffs for Boolean formulae*, Inform. Process. Lett., 49 (1994), pp. 151–155.

[BBC+01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf, *Quantum lower bounds by polynomials*, J. ACM, 48 (2001), pp. 778–797.

[BCE95] N. H. Bshouty, R. Cleve, and W. Eberly, *Size-depth tradeoffs for algebraic formulas*, SIAM J. Comput., 23 (1995), pp. 682–705.

[BCW98] H. Buhrman, R. Cleve, and A. Wigderson, *Quantum vs. classical communication and computation*, in Proceedings of the 30th ACM STOC, 1998, pp. 63–68.

[BS04] H. Barnum and M. Saks, *A lower bound on the quantum query complexity of read-once functions*, J. Comput. System Sci., 69 (2004), pp. 244–258.

[CCJY09] A. M. Childs, R. Cleve, S. P. Jordan, and D. Yeung, *Discrete-query quantum algorithm for NAND trees*, Theory of Computing, 5 (2009), pp. 119–123.

[CEMM98] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, *Quantum algorithms revisited*, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci., 454 (1969), pp. 339–354.

[Chi08] A. M. Childs, *On the relationship between continuous- and discrete-time quantum walk*, Commun. Math. Phys., to appear.

[CRŠZ07] A. M. Childs, B. W. Reichardt, R. Špalek, and S. Zhang, *Every NAND formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer*, arXiv quant-ph/0703015, 2007.

[FGG08] E. Farhi, J. Goldstone, and S. Gutmann, *A quantum algorithm for the Hamiltonian NAND tree*, Theory Comput., 4 (2008), pp. 169–190.

[Gro97] L. K. Grover, *A fast quantum mechanical algorithm for database search*, Phys. Rev. Lett., 79 (1997), pp. 325–328.

[Gro02] L. K. Grover, *Tradeoffs in the quantum search algorithm*, arXiv quant-ph/0201152, 2002.

[HMW03] P. Høyer, M. Mosca, and R. de Wolf, *Quantum search on bounded-error inputs*, in Proceedings of the 30th ICALP, 2003, pp. 291–299. Lecture Notes in Comput. Sci. 2719, Springer, New York.

[JKS03] T. S. Jayram, R. Kumar, and D. Sivakumar, *Two applications of information complexity*, in Proceedings of the 35th ACM STOC, 2003, pp. 673–682.

[KOS04] A. R. Klivans, R. O'Donnell, and R. A. Servedio, *Learning intersections and thresholds of halfspaces*, J. Comput. System Sci., 68 (2004), pp. 808–840.

[KS04] A. R. Klivans and R. A. Servedio, *Learning DNF in time $2^{\tilde{O}(n^{1/3})}$* J. Comput. System Sci., 68 (2004), pp. 303–318.

[KSV02] A. Y. Kitaev, A. H. Shen, and M. N. Vyalyi, *Classical and Quantum Computation*, Grad. Stud. Math. 47, American Mathematical Society, Providence, RI, 2002.

[LLS06] S. Laplante, T. Lee, and M. Szegedy, *The quantum adversary method and classical formula size lower bounds*, Comput. Complexity, 15 (2006), pp. 163–196.

[MNRS07] F. Magniez, A. Nayak, J. Roland, and M. Santha, *Search via quantum walk*, in Proceedings of the 39th ACM STOC, 2007, pp. 575–584.

[OS03] R. O'Donnell and R. A. Servedio, *New degree bounds for polynomial threshold functions*, in Proceedings of the 35th ACM STOC, 2003, pp. 325–334.

[RŠ08] B. W. Reichardt and R. Špalek, *Span-program-based quantum algorithm for evaluating formulas*, in Proceedings of the 40th ACM STOC, 2008, pp. 103–112.

[San95] M. Santha, *On the Monte Carlo decision tree complexity of read-once formulae*, Random Structures Algorithms, 6 (1995), pp. 75–87.

[Sni85] M. Snir, *Lower bounds on probabilistic linear decision trees*, Theoret. Comput. Sci., 38 (1985), pp. 69–82.

[SW86] M. Saks and A. Wigderson, *Probabilistic Boolean decision trees and the complexity of evaluating game trees*, in Proceedings of the 27th IEEE FOCS, 1986, pp. 29–38.

[Sze04] M. Szegedy, *Quantum speed-up of Markov chain based algorithms*, in Proceedings of the 45th IEEE FOCS, 2004, pp. 32–41.