Electronic Colloquium on Computational Complexity, Report No. 13 (2002)

Quantum and Stochastic Branching Programs of Bounded Width

Farid Ablayev[1], Cristopher Moore[2], and Christopher Pollett[3]

[1] Dept. of Theoretical Cybernetics
Kazan State University
420008 Kazan, Russia
ablayev@ksu.ru

[2] Computer Science Department
University of New Mexico
Albuquerque, New Mexico 87131
moore@cs.unm.edu

[3] Dept. of Math and Computer Science
San Jose State University
One Washington Square
San Jose, California 95192
pollett@mathcs.sjsu.edu

Abstract. We prove upper and lower bounds on the power of quantum and stochastic branching programs of bounded width. We show any NC^1 language can be accepted exactly by a width-2 quantum branching program of polynomial length, in contrast to the classical case where width 5 is necessary unless $NC^1 = ACC$. This separates width-2 quantum programs from width-2 doubly stochastic programs as we show the latter cannot compute the middle bit of multiplication. Finally, we show that bounded-width quantum and stochastic programs can be simulated by classical programs of larger but bounded width, and thus are in NC^1.

ISSN 1433-8092

1 Introduction

Interest in quantum computation has been steadily increasing since Shor's discovery of a polynomial time quantum algorithm for factoring [13]. A number of models of quantum computation have been considered, including quantum versions of Turing machines, simple automata, circuits, and decision trees. The goal of much of this research has been to understand in what ways quantum algorithms do and do not offer a speed-up over the classical case, and to understand what classical techniques for proving upper and lower complexity bounds transfer to the quantum setting.

Branching programs have proven useful in a variety of domains, such as hardware verification, model checking, and other CAD applications [14]. Recently, several models of quantum branching programs have been proposed [1, 10]. Ablayev, Gainutdinova, and Karpinski [1] gave a matrix-based definition of quantum branching programs as a natural generalization of quantum finite automata [7, 8]. In contrast to what had been shown about one-way quantum automata [4], they showed that arbitrary boolean functions can be computed by one-way quantum branching programs. They exhibited a symmetric boolean function which could be computed with log-width, leveled, oblivious read-once quantum programs which require linear width for classical programs of this type. Finally, they gave a lower bound on the width of a read-once quantum program computing a boolean function in terms of the minimal width of a classical ordered binary decision diagram (OBDD) for the same function. Nakanishi, Hamaguchi, and Kashiwabara [10] took a graph-based approach to defining quantum branching programs. They give a language L_{Half} which can be recognized by ordered bounded-width quantum branching programs of polynomial length but which cannot be recognized by probabilistic programs of this type.

In this paper we prove several new results for quantum branching programs of bounded width. After reviewing the definition of [1], we show that width-2 quantum programs are more powerful than width-2 doubly stochastic programs, and are as strong as deterministic branching programs of width 5. Specifically, we show that polynomial-length, width-2 quantum branching programs can recognize any NC^1 language exactly. This is surprising, since such programs act on a single qubit. On the other hand, we show that polynomial-length, width-2 doubly stochastic programs cannot compute the middle bit of the multiplication function. In the classical case, Yao [15] showed that width-2 deterministic programs require superpolynomial length to compute the majority function, and Barrington [5] showed that width 5 is sufficient for deterministic programs to capture NC^1.

Finally, we improve the result of Ablayev, Gainutdinova, and Karpinski [1] by showing that bounded-probability quantum and stochastic programs can be simulated by deterministic programs of the same length and larger, but still bounded, width. Therefore these classes are contained in NC^1, and in fact for bounded-width quantum programs exact acceptance is just as strong as acceptance with bounded probability. We use the techniques of this result to show that polynomial-length width-2 stochastic programs accepting with er-

ror margin more than $\epsilon = 1/4$ cannot compute majority. In addition, we show that polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/8$ and polynomial-length width-3 stochastic programs accepting with error margin more than $\epsilon = 1/\sqrt{5}$ must compute functions in ACC.

2 Preliminaries

We begin by discussing the classical model of branching programs and then show how to quantize it. A good source of information on branching programs is Wegener's book [14], and for an introduction to quantum computation see Nielsen and Chuang [11].

Definition 1 *A* branching program *is a finite directed acyclic graph which recognizes some subset of $\{0,1\}^n$. Each node (except for the sink nodes) is labelled with an integer $1 \leq i \leq n$ and has two outgoing arrows labelled 0 and 1. This corresponds to querying the ith bit x_i of the input, and making a transition along one outgoing edge or the other depending on the value of x_i. There is a single source node corresponding to the start state, and there is a subset A of the sink nodes corresponding to accepting states. An input x is* accepted *if and only if it induces a chain of transitions leading to a sink node in A.*

A branching program is oblivious *if the nodes can be partitioned into levels $V_1, \ldots, V_l$ such that the nodes in V_l are the sink nodes, nodes in each level V_j with $j < l$ have outgoing edges only to nodes in the next level V_{j+1}, and all nodes in a given level V_j query the same bit x_{i_j} of the input. Such a program is said to have* length *l, and* width *k if each level has at most k nodes.*

Oblivious branching programs have an elegant algebraic definition. Recall that a *monoid* is a set with an associative binary operation $\cdot$ and an identity 1 such that $1 \cdot a = a \cdot 1 = a$ for all a.

Definition 2 *Let M be a monoid and $S \subset M$ an accepting set. Let x_i, $1 \leq i \leq n$ be a set of Boolean variables. A* branching program over M of length l *is a string of l* instructions*; the jth instruction is a triple $(i_j, a_j, b_j) \in \{1, \ldots, n\} \times M \times M$, which we interpret as a_j if $x_{i_j} = 0$ and b_j if $x_{i_j} = 1$. Given an input x, the* yield *$Y(x)$ of the program is the product in M of all its instructions. We say that the input x is* accepted *if $Y(x) \in S$, and the set of such inputs is the language L recognized by the program.*

Such programs are often called *non-uniform deterministic finite automata* (NUDFAs); a computation over a deterministic finite automaton consists of taking a product in its syntactic monoid, while in a NUDFA we allow the same variable to be queried many times, and for "true" and "false" to be mapped into a different pair of monoid elements in each query.

A common monoid is T_k, the set of functions from a set of k objects into itself. Then the program makes transitions among k states, and we can equivalently define oblivious, width-k branching programs by choosing an initial state and a

set of accepting states, where the k states correspond, according to an arbitrary ordering, to the k vertices in each level V_j.

Definition 3 *An* oblivious width-k branching program *is a branching program over T_k, where the accepting set $S \subset T_k$ consists of those elements of T_k that map an initial state $s \in \{1, \ldots, k\}$ to a final state $t \in A$ for some subset $A \subset \{1, \ldots, k\}$.*

We define language classes recognized by (non-uniform) families of bounded-width branching programs whose length increases polynomially with n:

Definition 4 *k-*BWBP *is the class of languages recognized by polynomial-length branching programs of width k, and* BWBP $= \cup_k$ *k-*BWBP.

Recall that a *group* is a monoid where every element has an inverse, and a group is *Abelian* if $ab = ba$ for all a, b. A subgroup $H \subseteq G$ is *normal* if the left and right cosets coincide, $aH = Ha$ for all $a \in G$. A group is *simple* if it has no normal subgroups other than itself and $\{1\}$.

Barrington [5] studied branching programs over the permutation group on k objects $S_k \subset T_k$; such programs are called *permutation programs*. He showed that polynomial-length programs over S_5, and therefore width-5 branching programs, can recognize any language in NC^1, the class of languages recognizable by boolean circuits of polynomial width and logarithmic depth. The version of Barrington's result that we will use is:

Theorem 1 ([5, 9]). *Let G be a non-Abelian simple group, and let $a \neq 1$ be any non-identity element. Then any language L in NC^1 can be recognized by a family of polynomial-length branching programs over G such that their yield is $Y(x) = a$ if $x \in L$ and 1 otherwise.*

Since the smallest non-Abelian simple group is $A_5 \subset S_5$, the group of even permutations of 5 objects, and since we can choose a to map some initial state s to some other final state t, width 5 suffices. Conversely, note that we can model a width-k branching program as a boolean product of l transition matrices of dimension k, and a simple divide-and-conquer algorithm allows us to calculate this product in $O(\log l)$ depth. Thus BWBP $\subset \mathrm{NC}^1$, so we have

$$5\text{-BWBP} = \mathrm{BWBP} = \mathrm{NC}^1 \ .$$

To define stochastic and quantum branching programs, we write the probability of acceptance as an inner product. Let e_s and e_t be k-dimensional vectors whose entries are 1 for the initial state and accepting final states respectively and 0 otherwise, and M_j the matrix corresponding to the jth instruction. Then write

$$P(x) = \left\langle e_s \left| \prod_{j=1}^{l} M_j \right| e_t \right\rangle$$

In the deterministic case $P = 1$ or $P = 0$ for all x, since the transition matrix corresponding to an element of T_k has exactly one 1 in each column. For a group this is true of the rows as well, in which case the M_j are permutation matrices. We can generalize this by letting the M_j be *stochastic* matrices, i.e. matrices with non-negative entries where each column sums to 1, and letting e_s be an initial probability distribution over the set of states. Then P is the probability that the program accepts. If the transpose of a stochastic matrix is also stochastic the matrix is called *doubly stochastic*. If all the matrices in a program are doubly stochastic then we say the program is a *doubly stochastic* program.

In the quantum case, we let the M_j be complex-valued and *unitary*, i.e. $M_j^{-1} = M_j^\dagger$ where $\dagger$ denotes the Hermitian conjugate, and e_s and e_t be initial and final state vectors with $|e_s|^2 = |e_t|^2 = 1$. Then the program accepts with probability

$$P(x) = \left| \left\langle e_s \left| \prod_{j=1}^{l} M_j \right| e_t \right\rangle \right|^2$$

Note that this is a "measure-once" model analogous to the quantum finite automata of [8], in which the system evolves unitarily except for a single measurement at the end. We could also allow multiple measurements during the computation, by representing the state as a density matrix and making the M_i superoperators instead of purely unitary operators; we do not do this here.

We can define recognition in several ways for the quantum case. We say that a language L is accepted *with bounded probability* if there is some $\epsilon > 0$ such that $P(x) > 1/2 + \epsilon$ if $x \in L$ and $P(x) < 1/2 - \epsilon$ if $x \notin L$, and accepted *exactly* if $P(x) = 1$ if $x \in L$ and $P(x) = 0$ if $x \notin L$ as in the deterministic case.

We denote by $B\cdot$ (where we will drop the '$\cdot$' when clear) the language classes recognized with bounded probability, and $E\cdot$ those recognized exactly. Writing SBP and QBP for stochastic and quantum branching programs respectively, we define the classes of languages recognized by width-k stochastic and quantum programs of polynomial length k-BSBP, k-BQBP, and k-EQBP. Note that we remove "BW" to avoid acronym overload. We write BSBP for $\cup_k k$-BSBP and define BQBP and EQBP similarly. We have

$$\text{BWBP} \subseteq \text{EQBP} \subseteq \text{BQBP}$$

and

$$\text{BWBP} \subseteq \text{BSBP}$$

but in principle k-BSBP could be incomparable with k-EQBP or k-BQBP.

3 Width-2 Doubly Stochastic and Quantum Programs

In this section we show that width-2 quantum programs with exact acceptance contain NC^1 and also show that these programs are stronger than width-2 doubly stochastic programs.

First we note that stochastic programs are stronger than permutation programs for width 2. It is easy to see that any program over $\mathbb{Z}_2$ simply yields the parity of some subset of the x_i. The AND_n function, which accepts only the input with $x_i = 1$ for all i, is not of this form, and so this language cannot be recognized by a width-2 permutation program. However, it can easily be recognized by a stochastic program P with bounded error which queries each variable once as follows: for $i < n$ it maps $x_i = 1$ and 0 to the identity $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the matrix $\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$ respectively, and for x_n it maps 1 and 0 to $\begin{pmatrix} 3/4 & 1/4 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 3/8 & 5/8 \\ 3/8 & 5/8 \end{pmatrix}$ respectively. Taking the first state to be both the initial and final state, P accepts with probability 3/4 if $x_i = 1$ for all i and 3/8 otherwise. Note that except for one matrix this is in fact a doubly stochastic program. If we had treated the variable x_n in the same fashion as the other variables we would have gotten a doubly stochastic program accepting AND_n with one-sided error.

Despite being stronger than their permutation counterparts, the next result shows width-2 doubly stochastic branching programs are not that strong. Let $MULT_k^n$ be the boolean function which computes the kth bit of the product of two n-bit integers. Define $MULT^n$ to be $MULT_{n-1}^n$. i.e., the middle bit of the product. We will argue that this function requires at least exponential lengthed width 2 stochastic programs.

Lemma 1. *Any width-2 doubly stochastic program on n variables is equivalent to one which queries each variable once and in the order 1,2,3, ... ,n.*

Proof. Any 2×2 stochastic matrix can be written as $\begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$ for some $p \in [0,1]$. It is easy to verify that matrices of this kind commute. Hence, if we have a product of such matrices $\prod_j^n M_{x_{j_i}}$ we can rewrite it so that we first take the product of all the matrices that depend on x_1, then those that depend on x_2, and so on. To finish the proof we note that products of doubly stochastic matrices are again doubly stochastic, so we can use a single doubly stochastic matrix for the product of all the matrices that depend on a given x_i.

The above lemma shows we can convert any width-2 doubly stochastic program into one which is read-once and with a fixed variable ordering. i.e. a randomized ordered binary decision diagram (OBDD). The next result is proved in Ablayev and Karpinski [2].

Theorem 2. *A BP-*OBDD *that correctly computes $MULT^n$ has length at least $2^{\Omega(n/\log n)}$.*

So by Lemma 1 we have immediately:

Corollary 1 *Any width 2 doubly stochastic program correctly computing $MULT^n$ with bounded error has length at least $2^{\Omega(n/\log n)}$.*

So width-2 stochastic programs are not that strong. However, width-2 *quantum* programs are surprisingly strong, as the next result shows. Note that a width-2 quantum program has a state space equivalent to a single qubit such as a single spin-1/2 particle [11].

Theorem 3. NC^1 *is contained in* 2-EQBP.

Proof. First, recall that A_5, the smallest non-Abelian simple group, is the set of rotations of the icosahedron. Therefore, the group $SO(3)$ of rotations of $\mathbb{R}^3$, i.e. the 3×3 orthogonal matrices with determinant 1, contains a subgroup isomorphic to A_5.

There is a well-known 2-to-1 mapping from $SU(2)$, the group of 2×2 unitary matrices with determinant 1, to $SO(3)$. Consider a qubit $a|0\rangle + b|1\rangle$ with $|a|^2 + |b|^2 = 1$; we can make a real by multiplying by an overall phase. The *Bloch sphere* representation (see e.g. [11]) views this state as the point on the unit sphere with latitude θ and longitude ϕ , i.e. $(\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta)$, where $a = \cos\theta/2$ and $b = e^{i\phi}\sin\theta/2$.

Given this representation an element of $SU(2)$ is equivalent to some rotation of the unit sphere. Recall the *Pauli matrices*

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Then we can rotate an angle α around the x, y or z axes with the following operators:

$$\begin{aligned} R_x(\alpha) = e^{i(\alpha/2)\sigma_x} &= \begin{pmatrix} \cos\alpha/2 & -i\sin\alpha/2 \\ -i\sin\alpha/2 & \cos\alpha/2 \end{pmatrix} \\ R_y(\alpha) = e^{i(\alpha/2)\sigma_y} &= \begin{pmatrix} \cos\alpha/2 & -\sin\alpha/2 \\ \sin\alpha/2 & \cos\alpha/2 \end{pmatrix}, \text{ and} \\ R_z(\alpha) = e^{i(\alpha/2)\sigma_z} &= \begin{pmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{pmatrix} . \end{aligned}$$

This makes $SU(2)$ a *double cover* of $SO(3)$, where each element of $SO(3)$ corresponds to two elements $\pm U$ in $SU(2)$. (Note that angles get halved by this mapping.) Therefore, $SU(2)$ has a subgroup which is a double cover of A_5. One way to generate this subgroup is with $2\pi/5$ rotations around two adjacent vertices of an icosahedron. Since two such vertices are an angle $\tan^{-1}2$ apart, if one is pierced by the z axis and the other lies in the x-z plane we have

$$\begin{aligned} a &= R_z(2\pi/5) = \begin{pmatrix} e^{i\pi/5} & 0 \\ 0 & e^{-i\pi/5} \end{pmatrix} \\ b &= R_y(\tan^{-1}2)\cdot a \cdot R_y(-\tan^{-1}2) \\ &= \frac{1}{\sqrt{5}} \begin{pmatrix} e^{i\pi/5}\tau + e^{-i\pi/5}\tau^{-1} & -2i\sin\pi/5 \\ -2i\sin\pi/5 & e^{-i\pi/5}\tau + e^{i\pi/5}\tau^{-1} \end{pmatrix} \end{aligned}$$

where $\tau = (1+\sqrt{5})/2$ is the golden ratio. Now consider the group element $c = a \cdot b \cdot a$; this rotates the icosahedron by π around the midpoint of the edge connecting these two vertices. In $SU(2)$, this maps each of the eigenvectors of σ_y to the other times an overall phase. Taking these as the initial and final state,

$$e_s = \frac{|0\rangle + i|1\rangle}{\sqrt{2}}, \quad e_t = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}$$

we have

$$|\langle e_s|c|e_t\rangle|^2 = 1$$

while, since the two eigenvectors are orthogonal,

$$|\langle e_s|1|e_t\rangle|^2 = 0 \ .$$

Now, Theorem 1 tells us that for any language in NC^1 we can construct a polynomial-length program over A_5 that yields the element equivalent to c if the input is in the language and 1 otherwise. Mapping this language to $SU(2)$ gives a program which yields $\pm c$ or 1, and accepts with probability 1 or 0.

4 Classical Simulations of Stochastic and Quantum Branching Programs

In this section we give general results on simulating stochastic and quantum programs by classical ones. We use this to show that width-2 quantum programs can be simulated by bounded-width deterministic programs, and to show that if $\epsilon > 1/4$ then width-2 stochastic programs for majority require super-polynomial length. We also get that polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/8$ and polynomial-length width-3 stochastic programs accepting with error margin more than $\epsilon = 1/\sqrt{5}$ must compute functions in ACC.

Theorem 4. *If a language is recognized with bounded probability $1/2+\epsilon$ by a width-w stochastic branching program, then it is also recognized by a deterministic branching program of the same length, and width*

$$w_C \leq \lfloor \epsilon^{-(w-1)} \rfloor$$

Similarly, if a language is recognized with bounded probability $1/2+\epsilon$ by a width-w_Q quantum branching program, it is recognized by a deterministic program of the same length, and width

$$w_C \leq \lfloor (\epsilon/2)^{-(2w-1)} \rfloor$$

The proof uses essentially the same techniques as were used for Theorem 3 of [12] and Proposition 6 of [7], which show that stochastic and quantum finite state automata that accept with bounded probability can be simulated by

deterministic finite state automata and therefore accept regular languages. Here we are extending this method to branching programs where the automaton is non-uniform; this was done for read-once branching programs in [1].

The idea is that if two state vectors are within a distance θ of each other, where θ depends on ϵ, then if the same operators are applied to both they must either both accept or both reject, and so are equivalent. Since only a finite number of balls of radius $\theta/2$ can fit into the state space we end up with a finite number of states. We start with the following lemmas. Note that we use the L_1 norm $|v|_1 = \sum_i |v_i|$ in the stochastic case, and the L_2 norm $|v|_2 = \sqrt{\sum_i |v_i|^2}$ in the quantum case.

As shorthand, say that a final probability distribution or state vector is *accepting* if it causes the program to accept with probability at least $1/2 + \epsilon$ and *rejecting* if it causes the program to accept with probability at most $1/2 - \epsilon$.

Lemma 2. *Consider a stochastic or quantum branching program. Let v, v' be two probability distributions or state vectors such that v is accepting and v' is rejecting. Then $|v - v'|_1 \geq 4\epsilon$ in the stochastic case, or $|v - v'|_2 \geq 2\epsilon$ in the quantum case.*

Proof. We prove the stochastic case first. As in the definition above, let $A \subset \{1, \ldots, k\}$ be the set of accepting states. By hypothesis $\sum_{i \in A} v_i \geq 1/2 + \epsilon$ and $\sum_{i \notin A} v_i \leq 1/2 - \epsilon$, and vice versa for v'. Then

$$|v - v'|_1 = \sum_i |v_i - v'_i| \geq \left| \sum_{i \in A} v_i - \sum_{i \in A} v'_i \right| + \left| \sum_{i \notin A} v_i - \sum_{i \notin A} v'_i \right| \geq 4\epsilon \ .$$

In the quantum case, let e_t be the accepting final state. Write $v_{\text{acc}} = \langle v | e_t \rangle e_t$ and $v_{\text{rej}} = v - v_{\text{acc}}$ for the components of v parallel to e_t and orthogonal to it, and similarly for v'. By hypothesis $|v_{\text{acc}}|_2^2 \geq 1/2 + \epsilon$ and $|v_{\text{rej}}|_2^2 \leq 1/2 - \epsilon$, and vice versa for v'. Then

$$|v_{\text{acc}} - v'_{\text{acc}}|, \ |v_{\text{rej}} - v'_{\text{rej}}| \geq \sqrt{1/2 + \epsilon} - \sqrt{1/2 - \epsilon}$$

so

$$|v - v'|_2 = \sqrt{|v_{\text{acc}} - v'_{\text{acc}}|_2^2 + |v_{\text{rej}} - v'_{\text{rej}}|_2^2} = \sqrt{1 + 2\epsilon} - \sqrt{1 - 2\epsilon} \geq 2\epsilon$$

where the final inequality comes from some simple algebra.

Given a set D in a metric space (see e.g. [3]) with metric ρ, we say two points x, y in D are *θ-equivalent* if they are connected by a chain of points where each step has distance less than θ; that is, $x \sim y$ if there are $z_0, \ldots, z_m \in D$ such that $x = z_0$, $y = z_m$, and $\rho(z_i, z_{i+1}) < \theta$ for all $0 \leq i < m$. Call the equivalence classes of this relation the *θ-components* of D. Then:

Lemma 3. *Suppose a stochastic or quantum branching program P accepts a language with bounded probability $1/2+\epsilon$. Then it is equivalent to a deterministic branching program of the same length where the states are the θ-components of the set of possible states at each step, where θ is given by Lemma 2.*

Proof. Since stochastic matrices never increase the L_1 distance, and unitary matrices preserve the L_2 distance, the θ-component of the P's state on a given step is a function only of the θ-component it was in on the previous step, and the value of variable queried on that step. This defines a deterministic branching program D on the set of θ-components of the same length as P. Lemma 2 shows that if two final states are in the same θ-component where $\theta = 4\epsilon$ or 2ϵ as applicable, they either both accept or both reject. Therefore, whether the final state is accepting or rejecting depends only on its θ-component, and D accepts if and only if P does.

Now all that remains is to bound the number of θ-components, or equivalently the maximum number of balls of radius $\theta/2$, that can fit in the state space of a stochastic or quantum program of width w. The w-dimensional probability distributions form a flat $(w-1)$-dimensional manifold of volume V and diameter 2, and a ball of radius $\theta/2 = 2\epsilon$ has volume at least $\epsilon^{w-1}V$. Similarly, the set of w-dimensional state vectors v with $|v|^2 = 1$ forms the complex sphere C^w, a $(2w-1)$-dimensional manifold of volume V and diameter 2. Due to the positive curvature of C^w, a ball of radius $\theta/2 = \epsilon$ has volume at least $(\epsilon/2)^{2w-1}V$. This completes the proof of Theorem 4 (note that both these bounds can be significantly improved with a little more work).

While we may need an exponentially larger width to simulate a stochastic or quantum branching program, the width is still constant. Thus bounded-width programs of all three types are equivalent, and since bounded-width classical programs are contained in NC^1 they all are. Conversely, we showed in Theorem 3 that NC^1 is contained in width-2 quantum programs, so we have

Corollary 2.

$$2\text{-EQBP} = 2\text{-BQBP} = \mathrm{BQBP} = \mathrm{BSBP} = \mathrm{BWBP} = \mathrm{NC}^1 \ .$$

In other words, width-2 quantum programs with exact acceptance are exactly as strong as quantum or stochastic programs of arbitrary bounded width, with exact or bounded-probability acceptance.

A more careful analysis of the number of θ-components can also be used to derive lower bounds on the capabilities of width-2 stochastic programs. For instance:

Corollary 3. *Any width-2 stochastic program recognizing majority with bounded probability $1/2 + \epsilon$ where $\epsilon > 1/4$ must have superpolynomial length.*

Proof. In the width-2 case the state space of a stochastic program consists of the pairs of points on the line from $(1,0)$ to $(0,1)$ in the plane. This line has L_1-length 2. If we allow θ-chains which might be centered on the end points, the maximum number of θ-components that can fit into this space is bounded by $\lfloor (2+4\epsilon)/4\epsilon \rfloor = \lfloor 1+1/2\epsilon \rfloor$. When $\epsilon > 1/4$ this gives 2, so using Lemma 3 a width-2 stochastic program with $\epsilon > 1/4$ can be simulated by a width-2 deterministic program of the same length. Finally, Yao [15] showed that deterministic width-2 programs for majority require super-polynomial length.

Recall that ACC[k] is the class of langauges computed by families of polynomial-sized, unbounded fan-in, AND, OR, NOT, MOD_k circuits of constant depth. The class ACC is $\cup_k$ACC[k]. Our techniques above can also be used to get the following result.

Corollary 4. *Polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/8$ and polynomial-length width-3 stochastic programs accepting with error margin more than $\epsilon = 1/\sqrt{5}$ must compute functions in* ACC.

Proof. By Barrington and Therien [6], we know that polynomial length width-4 deterministic programs compute functions in ACC. By our analysis in Corollary 3, if $\lfloor 1 + 1/2\epsilon \rfloor <= 4$ then a polynomial-length width-2 stochastic program can be simulated by a width-4 deterministic program of the same length. This inequality holds provided $\epsilon > 1/8$. For the width-3 case we apply Theorem 4 to get the bound on simulating width-3 stochastic programs by width-4 deterministic ones.

Acknowledgments. We are grateful to Alexander Russell, Eric Allender, David Mix Barrington, and Azaria Paz for helpful conversations and e-mails. C.M. is supported by NSF grant PHY-0071139 and the Sandia University Research Program.

References

1. F. Ablayev, A. Gainutdinova, and M. Karpinski. On computational Power of quantum branching programs. *Proc. FCT 2001*, Lecture Notes in Computer Science 2138: 59–70, 2001.
2. F. Ablayev and M. Karpinski. A lower bound for integer multiplication on randomized read-once branching programs. *Electronic Colloquium on Computational Complexity* TR 98-011, 1998. `http://www.eccc.uni-trier.de/eccc`
3. P. Alexandrov. *Introduction to set theory and general topology.* Berlin, 1984.
4. A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weakness, and generalizations. *Proc. 39th IEEE Symp. on Foundations of Computer Science (FOCS)*, 332–342, 1998.
5. D.A. Barrington. Bounded-width polynomial branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences* 38(1): 150–164, 1989.
6. D.A. Barrington and D. Therien. Finite Monoids and the Fine Structure of NC^1 *Journal of the ACM* 35(4): 941–952, 1988.
7. A. Kondacs and J. Watrous On the power of quantum finite automata. *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, 66–75, 1997.
8. C. Moore and J.P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science* 237: 275–306, 2000.
9. C. Moore, D. Thérien, F. Lemieux, J. Berman, and A. Drisko. Circuits and Expressions with Non-Associative Gates. *Journal of Computer and System Sciences* 60: 368–394, 2000.

10. M. Nakanishi, K. Hamaguchi, and T. Kashiwabara. Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. *Proc. 6th Annual International Conference on Computing and Combinatorics (COCOON)* Lecture Notes in Computer Science 1858: 467–476, 2000.
11. M.A. Nielson and I.L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press. 2000.
12. M. Rabin. Probabilistic automata. *Information and Control* 6: 230–245, 1963.
13. P. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26(5): 1484–1509, 1997.
14. Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications. 2000.
15. A.C. Yao. Lower Bounds by Probabilistic Arguments *Proc. of the 24th IEEE Symp. on Foundations of Computer Science (FOCS)*, 420–428, 1983.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'