# Quantum and Stochastic Branching Programs of Bounded Width

Farid Ablayev* Cristopher Moore† Christopher Pollett‡

**Abstract**

In this paper we show that one qubit polynomial time computations are at least as powerful as $NC^1$ circuits. More precisely, we define syntactic models for quantum and stochastic branching programs of bounded width and prove upper and lower bounds on their power. 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 $NC^1$.

## 1 Preliminaries

Interest in quantum computation has steadily increased since Shor's discovery of a polynomial time quantum algorithm for factoring [17]. 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; see for example the book by Wegener [20]. In addition, branching programs are a convenient model for nonuniform computation with varying restrictions.

*Dept. of Theoretical Cybernetics, Kazan State University. ablayev@ksu.ru ; work done in part while visiting Institute of Advanced Study and Max-Plnak Institute for Mathematics, supported by RFBR grant 03-01-00769

†Computer Science Department, University of New Mexico, Albuquerque and the Santa Fe Institute moore@cs.unm.edu; supported by NSF grant EIA-0218563

‡Dept. of Computer Science San Jose State University. pollett@cs.sjsu.edu

Even oblivious branching programs of constant width — the non-uniform equivalent of finite-state automata — are surprisingly powerful. Indeed, Barrington [5] showed that branching programs of width 5 are already as powerful as circuits of logarithmic depth.

Moreover, branching programs are a very natural model for comparing the power of quantum computation with classical computation, both deterministic and randomized. Recently, several models of quantum branching programs have been proposed [1, 2, 4, 12].

In this paper we define and consider syntactic models of stochastic and quantum branching programs. For this syntactic model we present several results for quantum branching programs of bounded width [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 language in $\mathrm{NC}^1$ exactly. Note that such programs are equivalent to a nonuniform automaton whose only storage device is 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 [21] showed that width-2 deterministic programs require superpolynomial length to compute the majority function.

Next, we show that bounded-error (syntactic) quantum and stochastic programs can be simulated by deterministic programs of the same length and larger, but still bounded, width. Therefore the class of languages recognized by these programs coincides with (nonuniform) $\mathrm{NC}^1$. This also implies that, for bounded-width quantum programs, exact acceptance is just as strong as acceptance with bounded error.

To give some flavour of what our syntactic model is, consider the usual computation of a branching program. When we query a variable we do one of two actions depending on its values. If we query the variable again we expect to see the same value. An inconsistent state is one that is reached by querying a variable more than once and using a different value each time. Our syntactic models allow for the inclusion of inconsistent final states when the final bounded error acceptance property is calculated. This enables us to show that final consistent states of stochastic and quantum programs recognizing languages with bounded error recognition have a specific metric property: sets of accepting and rejecting states are isolated from each other. We define syntactic programs as programs which satisfy the property that the set of all (consistent as well as inconsistent) final states form a partition into two sets isolated from each other.

We call this property a syntactic property in analogy with the notion of syntactic classical read-$k$-times programs. The definition of classical syntactic read-$k$-times branching programs [7] includes all their nonconsistent paths in the general read-$k$-times variables testing property.

We use the techniques of our result to show that polynomial-length width-2 stochastic programs that accept with probability $1/2 + \epsilon$ cannot compute the majority function if $\epsilon > 1/4$. In addition, we show that polynomial-length stochastic programs with width 2 and $\epsilon > 1/8$, width 3 and $\epsilon > 1/3$, or width 4 and $\epsilon > 3/8$ can only recognize languages in ACC.

## 2 Branching Programs

We begin by discussing the classical model of branching programs and then generalize it to the quantum setting. A good source of information on branching programs is Wegener's book [20], and for an introduction to quantum computation see Nielsen and Chuang [13].

**Definition 1** *A* branching program *is a finite directed acyclic graph which accepts some subset of $\{0,1\}^n$. Each node (except for the sink nodes) is labeled with an integer $1 \leq i \leq n$ and has two outgoing arrows labeled* 0 *and* 1. *This pair of edges corresponds to querying the $i$'th 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 a subset* Accept *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 node in* Accept, *and the set of such inputs is the language accepted by the program. A branching program is* oblivious *if the nodes can be partitioned into levels $V_1, \ldots, V_\ell$ and a level $V_{\ell+1}$ such that the nodes in $V_{\ell+1}$ are the sink nodes, nodes in each level $V_j$ with $j \leq \ell$ 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 $\ell$, *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 $\ell$ *is a string of $\ell$* instructions*; the $j$'th 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 recognized by the program.*

Such programs are often called *non-uniform deterministic finite automata* (NUDFAs). A computation in a deterministic finite automaton can be thought of as taking a product in its syntactic monoid; in a NUDFA we

generalize this by allowing the same variable to be queried many times, and allowing "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 final states, where the $k$ states correspond to the $k$ vertices (according to an arbitrary ordering) 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 $A \subset \{1, \ldots, k\}$.*

# 3 Bounded Width Branching Programs

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 $\mathrm{NC}^1$, the class of languages recognizable by Boolean circuits of polynomial width and logarithmic depth [14]. The version of Barrington's result that we will use is:

**Theorem 1 ([5, 11])** *Let $G$ be a non-Abelian simple group, and let $a \neq 1$ be any non-identity element. Then any language $L$ in $\mathrm{NC}^1$ 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 permutation $a$ that maps 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 $\ell$ 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 $\subset$ NC$^1$, so we have

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

In the stochastic and quantum cases, the state of the program will be described by a $k$-dimensional vector $\mu$. The $j$'th step of the program will query a variable $x_{i_j}$, and apply a transition matrix $M_j(x_{i_j})$.

**Definition 5** *We call state (state vector) $\mu$ of branching program a consistent state if there exists an input $x$ that induces a chain of transitions leading to the state $\mu$ from the initial state $\mu_0$. Otherwise, we call the $\mu$ an inconsistent state.*

If $x = x_1, \ldots, x_n$ is the input of a program of length $\ell$, then the final consistent state of the program will be

$$|\mu(x)\rangle = \prod_{j=\ell}^{1} M_j(x_{i_j})|\mu_0\rangle \ .$$

(This product is in reverse order since we think of each step of the program as a left multiplication of the state by $M_j$.) In the deterministic case, state $\mu$ is a Boolean vector with exactly one 1 and the $M_j$ correspond to elements of $T_k$ and so have exactly one 1 in each column. For branching programs over groups this is true of the rows as well, in which case the $M_j$ are permutation matrices. We can generalize this by letting $\mu$ be a probability distribution, and letting the $M_j$ be *stochastic* matrices, i.e., matrices with non-negative entries where each column sums to 1, or *doubly stochastic* where both the rows and columns sum to 1. Recall that, by Birkhoff's Theorem [19], doubly stochastic matrices are convex combinations of permutation matrices [**?**].

In the deterministic and stochastic cases for sink state vector $\mu \in V_{\ell+1}$, we define

$$\Pr(\mu) = \sum_{i \in \text{Accept}} \langle i \mid \mu \rangle = \|\Pi_{\text{Accept}}\mu\|_1 \qquad (1)$$

and we define the probability of acceptance as $Pr(x) = Pr(\mu(x))$. Here $|i\rangle$ is the basis vector with support on the state $i$, and $\Pi_{\text{Accept}}$ is a projection operator on the *accepting subspace* span$\{|i\rangle : i \in \text{Accept}\}$.

For *quantum* branching programs, $\mu$ is a complex state vector with $\|\mu\|_2 = 1$, and the $M_j$ are complex-valued unitary matrices. Then we define $Pr(\mu)$ for sink state vector $\mu \in V_{\ell+1}$ as

$$\Pr(\mu) = \sum_{i \in \text{Accept}} |\langle i \mid \mu \rangle|^2 = \|\Pi_{\text{Accept}}\mu\|_2^2 \qquad (2)$$

and the probability of acceptance as $Pr(x) = Pr(\mu(x))$ which is the probability that, that if we measure $\mu(x)$, we will observe it in the accepting subspace. Note that this is a "measure-once" model analogous to the model of quantum finite automata in [10], 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 $\mathcal{A} = \{\mu : Pr(\mu) > 1/2\}$ the state as a density matrix and making the $M_j$ superoperators; we do not consider this here.

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

# 4 Syntactic Variants of Stochastic and Quantum Programs

For unbounded and bounded error stochastic and quantum branching programs we define two subsets $\mathcal{A}$ and $\mathcal{R}$ of the set of sink state vectors as follows: For an unbounded error programs, we define

$$\mathcal{A} = \{\mu \in V_{\ell+1} : Pr(\mu) > 1/2\} \quad \text{and} \quad \mathcal{R} = \{\mu \in V_{\ell+1} : Pr(\mu) \leq 1/2\};$$

for bounded error programs,we define

$$\mathcal{A} = \{\mu \in V_{\ell+1} : Pr(\mu) \geq 1/2 + \epsilon\} \quad \text{and} \quad \mathcal{R} = \{\mu \in V_{\ell+1} : Pr(\mu) \leq 1/2 - \epsilon\}$$

We call $\mathcal{A}$ the accepting set of state vectors and call $\mathcal{R}$ the rejecting set of state vectors.

**Definition 6** *We call a stochastic or a quantum branching program a syntactic program if its accepting and rejecting set of state vectors form a partition of the set of sink state vectors for the program. That is* $V_{\ell+1} = \mathcal{A} \cup \mathcal{R}$.

Observe that deterministic, unbounded error stochastic and quantum branching programs are syntactic programs. But in the case of bounded error branching programs it might happen that $V_{\ell+1} \neq \mathcal{A} \cup \mathcal{R}$. That is, it might happen that a inconsistent final state vector $\mu \in V_{\ell+1}$ has the property that $1/2 - \epsilon < Pr(\mu) < 1/2 + \epsilon$.

We denote by $B\cdot$ the language classes recognized by standard (nonsyntactic) programs with bounded error and denote by $E\cdot$ those recognized

exactly. The notations SBP and QBP stand for stochastic and quantum branching programs, respectively. We denote the classes of languages recognized by width-$k$ stochastic and quantum programs of polynomial length as $k$-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. Clearly 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.

# 5 Width-2 Doubly Stochastic and Quantum Programs

In this section we show that width-2 syntactic quantum programs with exact acceptance contain $\text{NC}^1$, and also that this class of programs is stronger than width-2 syntactic doubly 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 $x_1, x_2, \ldots, x_n$.*

*Proof.* Any $2 \times 2$ 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 form commute. Hence, if we have a product of such matrices $\prod_{j=n}^{1} M_j(x_{i_j})$ 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). Hence in the case of width-2 syntactic and nonsyntactic models of programs are equivalent.

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 function 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 $\begin{pmatrix} 3/4 & 0 \\ 1/4 & 1 \end{pmatrix}$ and $\begin{pmatrix} 3/8 & 3/8 \\ 5/8 & 5/8 \end{pmatrix}$ 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 syntactic 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 $k$'th 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 any width-2 stochastic program that calculates this function (i.e., that recognizes the set of inputs for which $MULT^n = 1$) requires at least exponential width.

In [3] Ablayev and Karpinski investigated randomized OBDDs, i.e., those which accept with bounded error.

**Theorem 2 ([3])** *Any randomized* OBDD *that correctly computes* $MULT^n$ *has width at least* $2^{\Omega(n/\log n)}$.

So by Lemma 1 we have immediately:

**Corollary 1** $MULT^n$ *can not be computed by a width-2 doubly stochastic program.*

While width-2 doubly stochastic programs are quite weak, the next result shows that width-2 *quantum* programs are surprisingly strong. Note that a width-2 quantum program has a state space equivalent to a single qubit, such as a single spin-1/2 particle.

**Theorem 3** $\mathrm{NC}^1$ *is contained in syntactic* 2-EQBP.

*Proof.* Recall that the smallest non-Abelian simple group $A_5$ is isomorphic to the set of rotations of the icosahedron. Therefore, the group $SO(3)$ of rotations of $\mathbb{R}^3$, i.e., the $3 \times 3$ 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 $2 \times 2$ 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. [13]) views this state as the point on the unit sphere with latitude $\theta$ and longitude $\phi$ , 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 $\alpha$ 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 $a$ and $b$ around two adjacent vertices of an icosahedron. Since two such vertices are an angle $\tan^{-1} 2$ apart, if one lies on 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 $\pi$ around the midpoint of the edge connecting these two vertices. In $SU(2)$, this maps each of the eigenvectors of $\sigma_y$ to the other times an overall phase. These eigenvectors are

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

so we have

$$|\langle e_+| \pm c|e_-\rangle|^2 = 1$$

while, since they are orthogonal,

$$|\langle e_+|1|e_-\rangle|^2 = 0 \ .$$

Now, Theorem 1 tells us that for any language $L$ in $\mathrm{NC}^1$ 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. Using the embedding of $A_5$ in $SO(3)$ and then lifting to $SU(2)$ gives a program which yields $\pm c$ or 1. If we take the initial state to be $\mu_0 = e_-$ and the accepting subspace to be that spanned by $e_+$, this program accepts $L$ exactly. □

# 6 Deterministic Simulations of Syntactic Stochastic and Quantum Branching Programs

In this section we give general results on simulating syntactic stochastic and quantum programs with deterministic ones. Specifically, Theorem 4 shows that syntactic stochastic and quantum programs that accept with bounded error can be simulated by deterministic programs of the same length and larger (but still bounded) width. Below we use this to place upper bounds on the computational power of stochastic programs with various widths and error thresholds.

**Theorem 4** *Let $P$ be a syntactic stochastic or quantum branching program of width $k$ and length $\ell$ that recognizes a language $L$ with probability $1/2+\epsilon$. Then, there exists a deterministic branching program $P'$ of width $k'$ and length $\ell$ that recognizes $L$, where*

$$k' \leq \left(\frac{1}{\epsilon}\right)^{k-1} \tag{3}$$

*if $P$ is stochastic, and*

$$k' \leq \left(\frac{2}{\epsilon}\right)^{2k} \tag{4}$$

*if $P$ is quantum.*

*Proof.* Our proof is a simple generalization of arguments of Rabin [15] and Kondacs and Watrous [9] to the non-uniform case. For each step of the program, we define an equivalence relation on state vectors, where two state vectors are equivalent if they lead to the same outcome (acceptance or rejection). Since $P$ recognizes $L$ with bounded error, inequivalent states must be bounded away from each other, and since the state space is compact the number of equivalence classes is finite. These equivalence classes then become the states of our deterministic program $P'$.

First, we construct a much larger deterministic branching program $P''$ whose states at each level $j$ consist of the state vectors (consistent and

nonconsitent) $\mu$ that $P$ can reach after $j$ steps (computational or noncomputational). $V_1''$ consists of the initial state $\mu_0$, and for all $1 \leq j \leq \ell$ each $\mu \in V_j''$ has two outgoing edges to $M_j(0)\mu, M_j(1)\mu \in V_{j+1}''$. The syntactic property provides partition the final states $\mu \in V_{\ell+1}''$ into accepting and rejecting subsets $\mathcal{A}$ and $\mathcal{R}$ according to equations (1) and (2).

In the stochastic case, $\mu \in \mathcal{A}$ (resp. $\mu \in \mathcal{R}$) if $\|\Pi_{\text{Accept}}\mu\|_1 \geq 1/2 + \epsilon$ (resp. $\|\Pi_{\text{Accept}}\mu\|_1 \leq 1/2 - \epsilon$), and similarly in the quantum case except that $\|\cdot\|_1$ is replaced with $\|\cdot\|_2^2$. Clearly if $\mathcal{A}$ is the accepting subset of $V_{\ell+1}''$, then $P''$ recognizes $L$ deterministically.

Now we inductively define an equivalence relation $\equiv_j$ on $V_j''$ for each level $j$. First, we let $\mathcal{A}$ and $\mathcal{R}$ be the equivalence classes of $\equiv_{\ell+1}$ (note that $V_{\ell+1}'' = \mathcal{A} \cup \mathcal{R}$ since $P$ recognizes $L$ with bounded error). Then, for each $1 \leq j \leq \ell$, define

$$\mu \equiv_j \mu' \;\Leftrightarrow\; M_j(0)\mu \equiv_{j+1} M_j(0)\mu' \text{ and } M_j(1)\mu \equiv_{j+1} M_j(1)\mu' \ .$$

We now define $P'$ as the branching program whose states $V_j'$ for each level $j$ are the equivalence classes of $\equiv_j$ and whose accepting subset is the singleton $\{\mathcal{A}\}$. Clearly $P'$ is well-defined and recognizes $L$ deterministically; it just remains to show that the number of equivalence classes for each $j$ is bounded.

First we show that two inequivalent state vectors in $V_j''$ must be far apart, using the following standard argument [15, 9].

**Lemma 2** *Suppose $\mu, \mu' \in V_j''$ and $\mu \not\equiv_j \mu'$. Then*

$$\|\mu - \mu'\|_1 \geq 4\epsilon$$

*if $P$ is stochastic, and*

$$\|\mu - \mu'\|_2 \geq 2\epsilon$$

*if $P$ is quantum.*

*Proof.* Since stochastic and unitary matrices both preserve or decrease the appropriate norm, it suffices to show this for the last step. Therefore, suppose that $j = \ell + 1$, $\mu \in \mathcal{A}$ and $\mu' \in \mathcal{R}$. We can decompose both vectors into their components inside the accepting subspace and transverse to it, writing $\mu = \mu_A + \mu_R$ where $\mu_A = \Pi_{\text{Accept}}\mu$ and $\mu_R = (1 - \Pi_{\text{Accept}})\mu$ and similarly for $\mu'$. In the stochastic case, $\|\mu_A\|_1 \geq 1/2 + \epsilon$ and $\|\mu_A'\|_1 \leq 1/2 - \epsilon$, and so

$$\begin{aligned} \|\mu - \mu'\|_1 &= \|\mu_A - \mu_A'\|_1 + \|\mu_R - \mu_R'\|_1 \\ &\geq 2\left[(1/2 + \epsilon) - (1/2 - \epsilon)\right] \\ &= 4\epsilon \ . \end{aligned}$$

In the quantum case, $\|\mu_A\|_2 \geq \sqrt{1/2+\epsilon}$ and $\|\mu'_A\|_2 \leq \sqrt{1/2-\epsilon}$, so

$$\begin{aligned}
\|\mu - \mu'\|_2^2 &= \|\mu_A - \mu'_A\|_2^2 + \|\mu_R - \mu'_R\|_2^2 \\
&\geq 2\left[\sqrt{1/2+\epsilon} - \sqrt{1/2-\epsilon}\right]^2 \\
&= 2\left(1 - \sqrt{1-4\epsilon^2}\right) \\
&\geq 4\epsilon^2
\end{aligned}$$

so $\|\mu - \mu'\|_2 \geq 2\epsilon$. □

It follows that the width $k'$ of $P'$ is at most the largest number of balls of radius $2\epsilon$ or $\epsilon$ (in the stochastic and quantum case respectively) one can fit inside the state space. In the stochastic case, the state space is a $(k-1)$-dimensional simplex. Its $L_1$-diameter is 2, so each ball of radius $2\epsilon$ covers a fraction at least $(1/\epsilon)^{k-1}$ of its volume, yielding (3). This bound is crude in that it assumes that the center of each ball is at a corner of the simplex; balls whose center are in the interior of the simplex cover up to $2^{k-1}$ times as much volume. In particular, if $k = 2$ then $k' \leq 1 + 1/(2\epsilon)$.

In the quantum case, the state space is isomorphic to the surface of the $2k$-dimensional sphere of radius 1. The crude bound of (4) comes from noticing that this sphere, and the balls of radius $\epsilon$ whose centers lie on its surface, are all contained in a $2k$-dimensional ball of radius 2. □

Theorem 4 shows that bounded-error syntactic stochastic and quantum programs of constant width can be simulated by deterministic programs of constant (albeit exponentially larger) width, and are therefore contained in $\mathrm{NC}^1$. Conversely, we showed in Theorem 3 that $\mathrm{NC}^1$ is contained in width-2 syntactic quantum programs. Therefore, the following classes all coincide with $\mathrm{NC}^1$:

**Corollary 2** *For syntactic programs,*

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

Of all the program classes discussed in this paper, the only ones *not* included in this collapse are stochastic programs of width less than 5. Theorem 4 allows us to place upper bounds on their computational abilities if their error margins are sufficiently large. For instance, since Yao [21] showed that width-2 deterministic programs require superpolynomial length to compute the majority function, we have

**Corollary 3** *For the syntactic case, width-2 stochastic branching programs of polynomial length cannot recognize the majority function with probability* $1/2 + \epsilon$ *if* $\epsilon > 1/4$.

Similarly, recall that $\mathrm{ACC} = \cup_p \mathrm{ACC}[p]$ where $\mathrm{ACC}[p]$ is the class of languages recognizable by constant-depth circuits with AND, OR, and mod-$p$ counting gates of arbitrary fan-in. It is known that $\mathrm{ACC}[p] \subsetneq \mathrm{NC}^1$ for prime $p$ [16, 18], and strongly believed, but not known, that $\mathrm{ACC} \subsetneq \mathrm{NC}^1$. Since its is known [6] that deterministic programs of width less than 5 recognize languages in ACC[6], we have

**Corollary 4** *Suppose $L$ is recognized with probability $1/2 + \epsilon$ by a width-$k$ stochastic syntactic branching program of polynomial length. If $k = 2$ and $\epsilon > 1/8$, or $k = 3$ and $\epsilon > 1/3$, or $k = 4$ and $\epsilon > 3/8$, then $L \in \mathrm{ACC}$.*

*Proof.* For each $k$ we consider the problem of how small $\epsilon$ has to be to fit 5 points into the $(k-1)$-dimensional simplex with an $L_1$ distance $4\epsilon$ between them. While these values of $\epsilon$ are smaller that those given by (3), they follow easily from assuming without loss of generality that $k$ of the points lie on the simplex's corners. □

However, we conjecture that stochasticity doesn't greatly increase the power of bounded-width branching programs, in the following sense:

**Conjecture 1** *If $L$ is recognized with bounded error by a stochastic branching program of width less than 5, then $L \in \mathrm{ACC}$.*

**Acknowledgments.** We grateful to Sasha Razborov for numeric suggestions on the results presentations. We thank Denis Thérien for helpful discussions on ACC and Sasha Shen for productive discussions on syntactic model.

## References

[1] F. Ablayev, C. Moore, and C. Pollett, Quantum and stochastic branching programs of bounded width. *Proc. 29th Intl. Colloquium on Automata, Languages and Programming* 343–354, 2002.

[2] F. Ablayev, A. Gainutdinova, and M. Karpinski, On the computational power of quantum branching programs. *Proc. FCT 2001*, Lecture Notes in Computer Science 2138: 59–70, 2001.

[3] 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`

[4] A. Ambainis, L. Schulman, and U. Vazirani, Computing with highly mixed states. *Proc. 32nd ACM Symp. on Theory of Computing* 697–704, 2000.

[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. Borodin, A. Razborov, and R. Smolensky, On lower bounds for read-$k$-times branching programs, *Computational Complexity*, 3, (1993), 1-18.

[8] J.G. Kemeny and J.L. Snell, *Finite Markov Chains.* Van Nostrand, 1960.

[9] A. Kondacs and J. Watrous, On the power of quantum finite automata. *Proc. 38th IEEE Conf. on Foundations of Computer Science* 66–75, 1997.

[10] C. Moore and J.P. Crutchfield, Quantum automata and quantum grammars. *Theoretical Computer Science* 237: 275–306, 2000.

[11] 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.

[12] 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 Intl. Conf. on Computing and Combinatorics (COCOON)* Lecture Notes in Computer Science 1858: 467–476, 2000.

[13] M.A. Nielson and I.L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[14] C. Papadimitriou, *Computational Complexity.* Addison-Wesley, 1994.

[15] M. Rabin, Probabilistic automata. *Information and Control* 6: 230–245, 1963.

[16] A.A. Razborov, Lower bounds for the size of circuits of bounded depth with basis $\{\&, \oplus\}$. *Math. Notes Acad. Sci. USSR* **41(4)** 333–338, 1987.

[17] P. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26(5): 1484–1509, 1997.

[18] R. Smolensky, Algebraic methods in the theory of lower bounds for Boolean circuit complexity. *Proc. 19th ACM Symposium on the Theory of Computing* 77–82, 1987.

[19] J. H. van Lint and R. M. Wilson. *A Course in Combinatorics*, 2nd ed. Cambridge. 2001.

[20] Ingo Wegener, *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[21] A.C. Yao, Lower Bounds by Probabilistic Arguments. *Proc. 24th IEEE Conf. on Foundations of Computer Science* 420–428, 1983.