Building and bounding quantum Bernoulli factories

Theodore J. Yoder[1]

[1]*Department of Physics, MIT, Cambridge, MA 02139*

A Bernoulli factory formalizes the notion of using one binary random variable (or coin) with unknown distribution to simulate another binary random variable with some desired distribution. This problem has been extensively studied classically, and recently Dale et. al. [1] have defined a quantum generalization that uses pure state qubits, called quoins, instead of coins and characterized the power of their factories when unlimited quoins are available. However, there are other generalizations of Bernoulli factories to the quantum world, and some are even interesting in the case of limited resources (e.g. quoins). Here we define a variety of resource limited quantum Bernoulli factories, place lower bounds on their power, and characterize a particular kind of factory that we call GCF$_1$. GCF$_1$ is a small part of the Bernoulli factory landscape, but it is not unimportant – we also show that the problem of evaluating a symmetric boolean function in the standard quantum query model reduces to GCF$_1$. We also give a method to convert GCF$_1$ into quoined factories of the sort of Dale et. al.

I. INTRODUCTION

Von-Neumann famously proposed [2] the following problem: provided with a coin that lands heads with probability λ, simulate a fair coin that lands heads with probability 1/2. His solution was to flip the biased coin twice, and upon observing the results (heads, tails) or (tails, heads) declare the output to be heads or tails, respectively. In the cases of observing two heads or two tails, repeat the procedure.

Since von-Neumann's trick, the problem of generating a desired coin from an unknown coin has been generalized considerably. The generic problem is to construct (the classical term is to "simulate") a coin of bias $f(\lambda)$ for some function of the unknown coin's bias λ. Necessary and sufficient conditions on the function $f(\lambda)$ were proven by Keane and O'Brien [3] such that this construction is possible, when one is allowed a potentially unbounded number of coin flips.

With the introduction of quantum mechanics, we are no longer limited to just discussing λ-coins, which in the quantum notation may be written $(1-\lambda)|0\rangle\langle 0| + \lambda|1\rangle\langle 1|$. We also have objects called "λ-quoins" [1], which are qubits in the state $\sqrt{1-\lambda}|0\rangle + \sqrt{\lambda}|1\rangle$. Upon measurement in the z-basis, a quoin looks like a coin, but it might offer even more power to a Bernoulli factory. Indeed, Dale, Jennings, and Rudolph (DJR) [1] have found that quoins allow one to construct $f(\lambda)$-coins for an even larger class of functions f than classical coins allow, when one is allowed a potentially unbounded number of quoins.

In fact, once we allow quoins there are many more kinds of Bernoulli factories to consider because now the inputs or outputs can both be either coins or quoins. Moreover, we might continue to broaden the scope and allow as input to the factory, instead of a coin or quoin, the use of a λ-quoin preparation operator $A_\lambda = \begin{pmatrix} \sqrt{1-\lambda} & -\sqrt{\lambda} \\ \sqrt{\lambda} & \sqrt{1-\lambda} \end{pmatrix}$, which is perhaps an equally good generalization of a classical coin – if it is applied to $|0\rangle$ and followed by complete decoherence, A_λ also simulates a coin flip. We will refer to A_λ as a λ-oracle. Similar use of a state preparation oracle is considered in [4], but the oracle there is built to hide quantum states rather than a probability.

The kinds of Bernoulli factories considered in this paper are included in Table I. Some types are newly defined and others have already been studied in the literature. We rename some types, removing from their names the word "Bernoulli" and replacing it with a description of the output type of the factory. For instance, the "quantum Bernoulli factory" of DJR becomes a "quantum coin factory" in our terminology since it uses quoins to make coins. We will use the term "Bernoulli factory" to refer generally to algorithms that use a black-box input hiding a probability distribution, be it a state or operator.

One of the key distinctions in Bernoulli factories is whether one is limited in copies of the input (say at most L queries can be used) or whether one is allowed an unlimited number of copies. In the unlimited case, such as that considered classically by Keane and O'Brien and quantumly by DJR, functions $f(\lambda)$ are efficiently constructible if the average number of queries is small. In the limited case, which we will consider exclusively in this paper, bounds are proven on the exact number of copies L need to construct a function $f(\lambda)$.

Bernoulli factories can also be split along the lines of which free resources are allowed. For instance, one might allow as free resources a supply of $|0\rangle$ states, sampling from λ-independent distributions, or any λ-independent single-qubit gates. These distinctions sometimes disappear in the unlimited case, however. For instance, copies of $|0\rangle$ can be gotten by measuring enough quoins in the 0/1 basis and λ-independent distributions can be created from coins or quoins using (necessarily) unbounded procedures similar to von-Neumann's.

Here we seek to better understand the myriad Bernoulli factories in the case of limited resources. We explore one type of Bernoulli factor most substantially, what we call a Groverlike coin factory (GCF) in Table I, but meanwhile discover relations with the other types of factories. We characterize exactly the $f(\lambda)$-coins that can be created by GCFs restricted to one qubit of mem-

name	input	output	gates allowed	other names
classical coin factory (CCF)	λ-coin	$f(\lambda)$-coin	classical	Bernoulli factory [3]
quantum coin factory (QCF)	λ-quoin	$f(\lambda)$-coin	quantum	quantum Bernoulli factory [1]
quantum quoin factory (QQF)	λ-quoin	$f(\lambda)$-quoin	quantum	—
oracular coin factory (OCF)	λ-oracle	$f(\lambda)$-coin	quantum	—
oracular quoin factory (OQF)	λ-oracle	$f(\lambda)$-quoin	quantum	—
Groverlike coin factory (GCF)	λ-oracle	$f(\lambda)$-coin	phase	—
Groverlike quoin factory (GQF)	λ-oracle	$f(\lambda)$-quoin	phase	—

TABLE I: Some types of Bernoulli factories considered in this report. Each uses some black-box input parameterized by λ and creates an output that depends on $f(\lambda)$. For instance, von-Neumann's coin problem is a CCF with $f(\lambda) = 1/2$. There are of course other interesting problems of this Bernoulli factory form. Examples of converting coins to quoins can be found in [5] or [6] for instance. Though not distinguished in the table, there is a difference between query unlimited factories and the query limited factories. Unless otherwise stated, in the main text these three-letter acronyms refer to the limited case.

ory (a subclass of GCF denoted GCF$_1$) with an optimal constructive proof and note that evaluation of symmetric boolean functions in the standard quantum query model reduces to this case. GCFs use the λ-oracle to create coins, and so we place lower bounds on how many queries are needed to create a coin with given bias and also give interesting examples of GCF$_1$ algorithms. In addition, we lower bound quantum coin factories (QCFs), which use quoins to create coins, using known bounds on state discrimination and show how approximate QCFs can be designed from GCF$_1$s.

Finally, we would like to list some differences between our work and DJR [1]. First, as mentioned above, our results, both upper and lower bounds, are in the resource limited scenario, as opposed to the resource unlimited case. One consequence of this is that unlike DJR, we do not have the ability to create perfect λ-independent distributions, which require an unbounded number of queries to construct and which DJR use heavily in their proofs. Also unlike DJR, we consider a quantum generalization of the classical factory that uses the λ-oracle resource rather than a λ-quoin, though after constructing factories in this model, we do convert some of our algorithms to (approximate) algorithms in the resource limited quoin model using a Hamiltonian simulation technique due to Lloyd et. al. [7]. We also define and study Bernoulli factories restricted to applying λ-independent quantum gates diagonal in the computational basis. Our motivation is to avoid trivializing the creation of λ-independent distributions (such as that constructed in the von-Neumann coin problem) to just applying an X or Y rotation to $|0\rangle$. In contrast, DJR use non-diagonal single-qubit gates in their constructions.

II. QUERY ALGORITHMS AND SYMMETRIC BOOLEAN FUNCTIONS

We start with reviewing a well-studied field, that of quantum query algorithms, so that we can, first, use some of the same proof techniques and, second, so that we can compare and contrast with oracular coin and quoin factories. Query algorithms are designed with the goal of calculating a function on N-bits $g : \{0,1\}^N \to \{0,1\}$ and queries are counted as the number of bits of the input x that need to be inspected before $g(x)$ can be determined. Random algorithms and quantum algorithms are allowed to fail, calculating $f(x)$ correctly only $\geq 2/3$ of the time averaged over the algorithm's internal randomness. In quantum algorithms, the oracle can be queried in superposition, so its effect on a three register basis state — index, (single-qubit) ancilla, and bystander — is $U|i\rangle|a\rangle|b\rangle = |i\rangle|a \oplus x_i\rangle|b\rangle$. In the query complexity model, the maximum number of queries ever needed is N.

A few general purpose techniques are known for proving lower bounds on such query algorithms, including the quantum adversarial [8, 9] and quantum polynomial methods [10]. Most relevant to the results of this paper, Beals et. al. [10] have completely determined the query complexity of quantum algorithms for symmetric boolean functions, those for which there exists a function $g_\sim$ of a single variable such that $g(x) = g_\sim(|x|)$ for $|x|$ the hamming weight of x. Their result is:

Theorem II.1. *(Beals et. al. [10])*
If $g : \{0,1\}^N \to \{0,1\}$ is non-constant and symmetric, then the quantum query complexity $Q(g) = \Theta\left(\sqrt{N(N-\Gamma(g))}\right)$ where

$$\Gamma(g) = \min\left\{|2k - N + 1| : g_\sim(k) \neq g_\sim(k+1), \quad 0 \leq k \leq N-1\right\}. \tag{II.1}$$

One of their main proof tools is Paturi's lemma, which bounds the approximate degree of symmetric boolean functions like f. The approximate degree $\widetilde{deg}(g)$ is the degree of the smallest degree single-variate polynomial p such that $|g(x) - p(|x|)| \leq 1/3$ for all x.

Lemma II.2. *(Paturi [11])*
If $g : \{0,1\}^N \to \{0,1\}$ is non-constant and symmetric, then $\widetilde{deg}(g) = \Theta\left(\sqrt{N(N-\Gamma(g))}\right)$.

We will find Paturi's lemma and Theorem II.1 make interesting comparisons with some of our lower bounds on Bernoulli factories. Also useful will be the Markov brother's inequality, which relates the degree of a polynomial to its extreme values on a domain.

Lemma II.3. *(Markov brother's inequality [12]) If P is a single-variate polynomial of degree d, then*

$$\frac{\max_{0\le y\le 1}|P^{(k)}(y)|}{\max_{0\le y\le 1}|P(y)|}\le 2^k\frac{d^2(d^2-1^2)\ldots(d^2-(k-1)^2)}{1\cdot 3\ldots(2k-1)}. \tag{II.2}$$

For all derivatives k, equality is achieved by the Chebyshev polynomials of the first kind, which will appear again in an optimal factory for a quantum version of von-Neumann's coin that we present in Section V C.

III. COIN AND QUOIN FACTORIES

An oracular coin factory (OCF) uses the quoin preparation operator

$$A_\lambda=\begin{pmatrix}\sqrt{1-\lambda} & -\sqrt{\lambda}\\ \sqrt{\lambda} & \sqrt{1-\lambda}\end{pmatrix}. \tag{III.1}$$

and quantum operations independent of λ to create, for some specified function f, coins

$$\rho_\lambda=(1-f(\lambda))|0\rangle\langle 0|+f(\lambda)|1\rangle\langle 1|. \tag{III.2}$$

When we want to clarify a coin's bias, we will also call these $f(\lambda)$-coins. An oracular quoin factory (OQF) has the same set of allowed operations, but uses them to create $f(\lambda)$-quoins

$$|\psi_\lambda\rangle=\sqrt{1-f(\lambda)}|0\rangle+\sqrt{f(\lambda)}|1\rangle. \tag{III.3}$$

Notice that a coin factory reduces to a quoin factory, since a state $|\psi_\lambda\rangle$ is a coin ρ_λ upon measurement in the z-basis. As such, we will prove lower bounds on coin factories which will apply also to quoin factories.

OCFs differ from the standard oracle model discussed in Section II by the type of oracle. Indeed the standard oracle U exists only to provide (coherent) access to an input bit string x, while in the OCF model there is no such input string – the oracle A_λ, a quantum operator, is itself the input! This means, for instance, that there is no natural upper bound on the number of queries required to make a $f(\lambda)$-coin, since the input can never be completely "read" as is the case with N queries of U.

Notice that oracular coin and quoin factories allow the use of arbitrary quantum gates, and therefore one effectively has access to coins (or even quoins) of any bias independent of λ that one wishes. This may appear somewhat contrary to the spirit of, for instance, von-Neumann's problem, where the very goal is to create a coin with bias independent of λ. Therefore, we also define, what we call, Groverlike coin and quoin factories (GCFs and GQFs) which still use the oracle A_λ to create $f(\lambda)$-coins and -quoins respectively, but which limit the allowed λ-independent quantum gates to phase gates, diagonal in the computational basis. With only phase gates, $|0\rangle$ initial states, measurement in the computational basis, and classical postprocessing, it is impossible to create coins with arbitrary constant bias $p\neq 0,1$ without using the resource A_λ. Note that a tomographic solution in the GCF model is also disallowed, because even if one were to somehow use $f(\lambda)$ to estimate λ and then calculate $f(\lambda)$, creating a coin of bias $f(\lambda)$ with only phase gates is still impossible. The connection of GCFs to Grover's algorithm [13] should hopefully be made clear later in Section V.

IV. OCF LOWER BOUNDS

A general algorithm for a coin factory is the alternation of the oracle A_λ with λ-independent quantum operations, applied to an initial state $|0\rangle^{\otimes n}$ on n-qubits. That is, for a L-query algorithm,

$$U_{L+1}\left(A_\lambda\otimes I\right)U_L\left(A_\lambda\otimes I\right)\ldots\left(A_\lambda\otimes I\right)U_1|0\rangle^{\otimes n}, \tag{IV.1}$$

where I is the identity operator on $n-1$ qubits. Call the output of this factory, $|\Psi\rangle=\sum_x\alpha_x|x\rangle$. Using an inductive method, it can easily be argued that the amplitudes α_x have the form $\sum_{k=0}^{L}\beta_{k,x}\sqrt{1-\lambda}^{L-k}\sqrt{\lambda}^{k}$. At the end of the algorithm, without loss of generality, we can obtain a coin by measuring a single qubit in the z-basis. The probability that this coin is heads (i.e. $|1\rangle$) will have the form $f(\lambda)=a(\lambda)+\sqrt{\lambda(1-\lambda)}b(\lambda)$, where a is a real polynomial of degree at most L and b is real polynomial of degree at most $L-1$. This can be seen by squaring the amplitude – some terms in the sum pair up so that the powers of $\sqrt{1-\lambda}$ and $\sqrt{\lambda}$ are both even, while other pairs result in them both being odd powers.

This correspondence between output coin bias and polynomials yields lower bounds on how many queries to A_λ are required to make a $f(\lambda)$-coin, through use of the Markov brother's inequality.

Theorem IV.1. *(OCF lower bounds) If an OCF produces $f(\lambda)$-coins, and $f(\lambda)=a_0(\lambda)+\sqrt{\lambda(1-\lambda)}a_1(\lambda)$ where a_0 and a_1 are polynomials, then it must use A_λ at least*

$$L\ge\max_{b\in\{0,1\}}\left(\sqrt{\frac{1}{2}\frac{\max_{0\le\lambda\le 1}|a_b'(\lambda)|}{\max_{0\le\lambda\le 1}|a_b(\lambda)|}}+b\right) \tag{IV.2}$$

times.

Proof. We have already argued above that an OCF making L queries can only make $f(\lambda)$-coins for f of the form stated where a_0 has degree at most L and a_1 has degree at most $L-1$. A polynomial of degree L must satisfy Eq. (IV.2) by the Markov brother's inequality, Lemma II.2. Since OCFs reduce to OQFs, QCFs, and QQFs the same lower bound applies to those Bernoulli factories as well. ■

Actually, although Theorem IV.1 applies to GCFs and GQFs as well by the same reductive argument, we can also make a slightly simpler statement in those cases. This is because the output $\sum_x \alpha_x|x\rangle$ of a general GCF making L queries can be shown to have amplitudes,

$$\alpha_x = \sum_{\substack{k=0 \\ k-|x| \text{ even}}}^{L} \beta_{k,x}\sqrt{1-\lambda}^{L-k}\sqrt{\lambda}^{k}. \tag{IV.3}$$

That is, the sum is only over values of k with the same parity as the hamming weight of x. This implies that $|\alpha_x|^2$ is a polynomial in λ for any x, and therefore an output coin from a GCF has a *polynomial* bias. As for Theorem IV.1, this means that when applied to GCFs or GQFs, Eq. IV.2 may be simplified to the $b = 0$ case.

Next, we move on to actually implementing Bernoulli factories, but concentrate on the least complex and, as we have seen, polynomial-biased type of factory, the GCF.

V. GCF ALGORITHMS

A. Grover's algorithm

Our first example of a GCF is, fittingly, the eponymous Grover's algorithm [13]. Although Grover's algorithm is usually framed as computing the boolean function OR on N-bits (indexed by the state of $n = \lceil \log N \rceil$ qubits), it is well-known that it acts non-trivially only within an SU(2) subspace of the full N-dimensional Hilbert space. This SU(2) subspace, called $\mathcal{T}$ from now on, is spanned by the (normalized) states $|\bar{t}\rangle = \frac{1}{\sqrt{N-M}}\sum_{x_i=0}|i\rangle$ and $|t\rangle = \frac{1}{\sqrt{M}}\sum_{x_i=1}|i\rangle$. If we associate these states with the vectors $(1,0)$ and $(0,1)$, respectively, then Grover's phase oracle acts as $Z = \left(\begin{smallmatrix}1 & 0\\ 0 & -1\end{smallmatrix}\right)$ and the initial state preparation operator $H^{\otimes n}$ takes the form $\left(\begin{smallmatrix}\sqrt{1-\lambda} & -\sqrt{\lambda}\\ \sqrt{\lambda} & \sqrt{1-\lambda}\end{smallmatrix}\right)$, which is exactly A_λ.

Grover tells us that to amplify the state $|t\rangle$, beginning with the start state $|s\rangle = A_\lambda|0\rangle = \frac{1}{\sqrt{N}}\sum_x|x\rangle = \sqrt{1-\lambda}|\bar{t}\rangle + \sqrt{\lambda}|t\rangle$, we should repeatedly apply the "Grover iterate" G which is the phase oracle Z, followed by S, a reflection about the start state. The reflection about the start state can be implemented using A_λ by $S = A_\lambda Z A_\lambda^\dagger = A_\lambda^2 Z$, since $Z A_\lambda Z = A_\lambda^\dagger$. But this means $G = SZ = A_\lambda^2$, and so Grover's algorithm with l iterates is a GCF making $L = 2l+1$ queries like $A_\lambda^{2l+1}|0\rangle = \cos\left((2l+1)\theta/2\right)|0\rangle + \sin\left((2l+1)\theta/2\right)|1\rangle$ for θ defined such that $\cos(\theta/2) = \sqrt{1-\lambda}$ and $\sin(\theta/2) = \sqrt{\lambda}$.

B. Memory limited GCFs

Motivated by Grover's algorithm, we will now consider GCF algorithms of a specialized form, using only a single qubit of memory, which we call GCF$_1$. Despite the space limitation, we find that such algorithms are remarkably powerful, in that $f(\lambda)$-quoins can be created for a wide variety of functions f. We can, in fact, exactly characterize the functions constructible in GCF$_1$. Our proof also gives the algorithm for each such function f. The single qubit memory limitation is similar to that considered by Aaronson and Drucker [14] in which they find that quantum automaton with just two states can be made arbitrarily sensitive to a input coin's bias, though we use our quantum memory for a different purpose.

When limited to a single qubit, a Groverlike factory takes the following general form,

$$\begin{aligned}\mathcal{G}|0\rangle &= A_\lambda P_{L-1}\dots A_\lambda P_2 A_\lambda P_1 A_\lambda|0\rangle \\ &= \sqrt{1-f(\lambda)}|0\rangle + e^{i\chi(\lambda)}\sqrt{f(\lambda)}|1\rangle,\end{aligned} \tag{V.1}$$

where $P_j = \exp(-i\phi_j Z)$. This is the most general GCF$_1$ algorithm.

Actually, GCF$_1$ is not completely irrelevant to the standard query model. In fact, the evaluation of symmetric boolean functions (such as Grover's algorithm for OR) reduce to this class of Bernoulli factories.

Lemma V.1. *(symmetric booleans reduce to GCF$_1$) Evaluating a symmetric boolean function $g : \{0,1\}^N \to \{0,1\}$ reduces to constructing an $f(\lambda)$-coin in the GCF$_1$ model with f approximating $g_\sim$ (i.e. $|f(j/N) - g_\sim(j/N)| \leq 1/3$ for all $j \in \{0,1,\dots,N\}$).*

This reduction works essentially because a GCF$_1$ algorithm in the form of Eq. (V.1) can be converted to a function evaluation problem restricted to the subspace $\mathcal{T}$ defined in Section V A. A complete argument is presented in Appendix A.

By choosing the phases ϕ_j, what biases $f(\lambda)$ can be constructed by a GCF$_1$? Or from the point of view of query complexity of symmetric boolean functions, how powerful are operations restricted to $\mathcal{T}$? Our next theorem characterizes the power of these algorithms.

Theorem V.2. *(GCF$_1$ characterization) A $f(\lambda)$-coin is constructible by a GCF$_1$ if and only if f is a polynomial, $f(\lambda) \in [0,1]$ for all $\lambda \in [0,1]$, and, if f has odd degree,*

1. *$f(\lambda) \leq 0$ for all $\lambda < 0$*
2. *$f(\lambda) \geq 1$ for all $\lambda > 1$*

or, if f has even degree,

1. *$f(\lambda) \leq 0$ for all $\lambda < 0$*
2. *$f(\lambda) \leq 0$ for all $\lambda > 1$.*

The proof of this theorem is a bit involved and so is presented in Appendix B. The constructive part of the proof uses a provably optimal number of queries (if f has degree L, then L queries are made), which is perhaps interesting as the characterizing constructions for resource unlimited QCFs and resource unlimited CCFs by Dale et. al. [1] and Keane, O'Brien [3] are non-optimal.

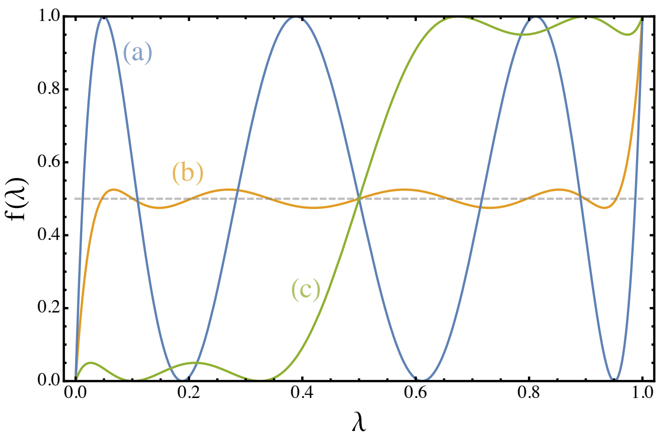

FIG. 1: What you can do with $L = 9$ queries – some examples of $f(\lambda)$-coins constructible in the GCF$_1$ model. (a) Grover's algorithm (b) A von-Neumann coin that approximates a fair coin over a wide range of λ (c) An approximate majority algorithm with success probability $1-\delta^2$ determining whether $\lambda \leq 1/2 - \epsilon$ or $\lambda \geq 1/2 + \epsilon$ for $\epsilon \approx 0.1$ and $\delta^2 = 0.05$.

Because of Lemma V.1, we might also interpret Theorem V.2 as a partial converse of the polynomial method of Beals et. al. in the case of computing symmetric boolean functions. It is not a complete converse because we restricted to algorithms acting wholly in the subspace $\mathcal{T}$. Nevertheless, Theorem V.2 allows us to create query algorithms, or GCF$_1$ algorithms, by specifying only an appropriate polynomial with the requisite properties. We demonstrate next.

C. A quantum von-Neumann coin

The largest family of GCF$_1$ algorithms we present is a quantum version of von-Neumann's coin when one is limited to L queries. Let L be odd. The polynomials we propose are

$$f_{\rm vN}(\lambda) = p + \frac{1}{2}\delta^2 - \delta^2 T_L\left[\sqrt{\lambda_0 + (\lambda_1 - \lambda_0)\lambda}\right]^2, \quad \text{(V.2)}$$

where $p \in (0,1)$ is the desired bias of the output coin, $\delta \in (0,1]$ is an error parameter satisfying $\delta^2/2 \leq p$ and $\delta^2/2 \leq 1-p$,

$$\lambda_0 = \cosh\left(\frac{1}{L}\cosh^{-1}\left(\frac{\sqrt{p+\delta^2/2}}{\delta}\right)\right)^2 \quad \text{(V.3)}$$

$$\lambda_1 = -\sinh\left(\frac{1}{L}\sinh^{-1}\left(\frac{\sqrt{1-(p+\delta^2/2)}}{\delta}\right)\right)^2, \quad \text{(V.4)}$$

and $T_L[x] = \cos(L\cos^{-1}(x))$ are the Chebyshev polynomials of the first kind.

What do the functions $f_{\rm vN}(\lambda)$ look like? It is easy to see that $f_{\rm vN}(0) = 0$ and $f_{\rm vN}(1) = 1$ and that they are degree L polynomials in λ. More interesting is that

$$\frac{\lambda_0 - 1}{\lambda_0 + \lambda_1} \leq \lambda \leq \frac{\lambda_0}{\lambda_0 + \lambda_1} \quad \Rightarrow \quad |f_{\rm vN}(\lambda) - p| \leq \frac{1}{2}\delta^2. \quad \text{(V.5)}$$

For very large L and small δ, the scaling of $w_0 = (\lambda_0 - 1)/(\lambda_0 + \lambda_1)$ and $w_1 = \lambda_0/(\lambda_0 + \lambda_1)$ becomes

$$w_0 \sim \frac{\log^2(2\sqrt{p}/\delta)}{L^2} \quad \text{(V.6)}$$

$$w_1 \sim \frac{\log^2(2\sqrt{1-p}/\delta)}{L^2}, \quad \text{(V.7)}$$

which implies that we can achieve a von-Neumann coin with bias $p \pm \delta^2/2$ for all $w_0 \leq \lambda \leq w_1$ using the λ-oracle A_λ

$$L = O\left(\frac{\log(1/\delta)}{\sqrt{\min(w_0, w_1)}}\right) \quad \text{(V.8)}$$

times. Additionally, this is optimal scaling in the parameters w_0 and w_1, because of Theorem IV.1, Eq. (IV.2).

We plot $f_{\rm vN}$ in Fig. 1 along with Grover's GCF$_1$ and a constructible polynomial that computes approximate Majority by rising quickly at $\lambda = 1/2$. See also Appendix C for examples of approximating some non-polynomial biases, like $f(\lambda) = \sqrt{\lambda}$, in the $L \to \infty$ limit.

D. GCF$_1$ ⊂ GCF and CCF ⊂ GCF

Consider the bias $f_2(\lambda) = 4\lambda + (-6 + \sqrt{2})\lambda^2 + (2 - \sqrt{2})\lambda^3$. This polynomial does not satisfy the conditions of Theorem V.2 – it has odd degree but $f_2(0) = f_2(1) = 0$. Nevertheless, f_2 can be created by a GCF, shown in Fig. 2. Thus, GCF must be strictly larger than GCF$_1$.

Can $f_2(\lambda)$ be constructed by a CCF? We will argue briefly that it cannot and so GCF is also strictly larger than CCF. The most general CCF limited to L queries can be constructed by sorting the strings $\{0,1\}^L$ into two sets for the output 0 and output 1 of the coin being simulated. To write the simulated bias $f_{\rm CCF}(\lambda)$, choose non-negative integers $a_k \leq \binom{L}{k}$ for all $k \in \{0, 1, \ldots L\}$, so then $f_{\rm CCF}(\lambda) = \sum_k a_k (1-\lambda)^{L-k}\lambda^k$. Therefore, a CCF is restricted to constructing coins with polynomial biases with integer coefficients. Evidently, f_2 is not one of those.

Finally, to demonstrate that CCF is incomparable with GCF$_1$ we provide two more examples. Indeed, a coin constructible in GCF$_1$ but not in CCF is that with bias $f_{\rm vN}$ from Section V C, since for some choices of δ and p this polynomial does not have integer coefficients. A function constructible in CCF but not in GCF$_1$ is the simple $f_{\rm 2heads}(\lambda) = \lambda^2$ function, which note does not satisfy the conditions of Theorem V.2 to be GCF$_1$ constructible.

VI. CONVERTING GCF$_1$ INTO QCF

Our constructions of Bernoulli factories have so far been only in the GCF$_1$ model, having only used the quoin

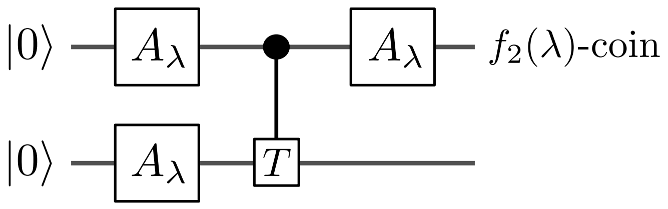

FIG. 2: A GCF circuit that makes a $f_2(\lambda)$-coin by measuring the top qubit. Here the controlled-T phase gate is $\mathrm{diag}(1,1,1,\exp(i\pi/4))$. Neither GCF$_1$ nor CCF can construct such coins, as described in the text.

preparation operator A_λ. It might be argued that this is also not in the spirit of von-Neumann's coin problem. The resource we should perhaps rather be using is a quoin $|\psi_\lambda\rangle = \sqrt{1-\lambda}|0\rangle + \sqrt{\lambda}|1\rangle$.

Our main result in this section is that the output $f(\lambda)$-coin of a GCF$_1$ making L queries to A_λ can be approximated arbitrarily closely by a QCF using asymptotically at most L^2 copies of $|\psi_\lambda\rangle$.

Theorem VI.1. *(queries from quoins)*
The GCF$_1$ algorithm $\mathcal{G}|0\rangle = A_\lambda P_{L-1}\ldots A_\lambda P_2 A_\lambda P_1 A_\lambda|0\rangle$ making L queries to A_λ alternatively with phase gates $P_j = e^{-i\phi_j Z}$ can be implemented with error ϵ using $O(L^2/\epsilon)$ copies of $|\psi_\lambda\rangle$.

Proof. To show this, we use a trick devised by Lloyd, Mohseni, and Rebentrost (LMR) [7] in the context of quantum principal component analysis. There, they simulate a positive semi-definite Hamiltonian (i.e. a density matrix) given copies of that density matrix. The general case is

$$\begin{aligned}\mathrm{Tr}_1\left[e^{-iSt}(\rho\otimes\sigma)e^{iSt}\right] &= \sigma - it[\rho,\sigma] + O(t^2) \qquad \text{(VI.1)}\\ &= e^{-i\rho t}\sigma e^{i\rho t} + O(t^2),\end{aligned}$$

where S is the swap operation between the two registers initially containing states ρ and σ and Tr_1 is the partial trace over the first register. To simulate the Hamiltonian ρ for a longer time T, break T into $t = T/m$ sized pieces. The accumulated error in the simulation of $e^{-i\rho T}$ will then be $\epsilon = O(mt^2)$, from which we get the relation $m = O(1/\epsilon)$ [7].

For our purposes, we must simply note that pairs of oracle calls can be replaced by this LMR algorithm. That is, $A_\lambda P_j A_\lambda^\dagger = A_\lambda P_j Z A_\lambda Z = e^{-i\phi_j|\psi_\lambda\rangle\langle\psi_\lambda|}$. Also, if L is odd, the QCF algorithm will start with $|\psi_\lambda\rangle = A_\lambda|0\rangle$ rather than from $|0\rangle$.

■

VII. QCF LOWER BOUNDS

Ultimately, what are the limits on the number of copies of a λ-quoin $|\psi_\lambda\rangle$ needed to create a single $f(\lambda)$-coin? Using Helstrom's bound on state discrimination, we will answer this question in this section.

Lemma VII.1. *(Helstrom's bound [15]) Any test to determine the identity of an unknown state promised that it is either $|\psi\rangle$ or $|\phi\rangle$ with success probability $1-\epsilon$ and overlap $|\langle\psi|\phi\rangle|^2 = 1-J$, must use*

$$k \geq \frac{\log 4\epsilon(1-\epsilon)}{\log|\langle\psi|\phi\rangle|^2} \qquad \text{(VII.1)}$$

copies of the unknown state. For small J (i.e. states that are close together), this bound is asymptotically

$$k = \Omega\left(\frac{1}{J}\log\left(\frac{1}{4\epsilon(1-\epsilon)}\right)\right). \qquad \text{(VII.2)}$$

The main theorem of this section is the following.

Theorem VII.2. *(QCF lower bound) A QCF must use at least $\Omega\left(\max_\lambda\left[f'(\lambda)^2\lambda(1-\lambda)\right]\right)$ quoins to produce a single $f(\lambda)$-quoin.*

In Appendix D, we prove this theorem using Helstrom's bound. We also provide examples for some exemplary functions $f(\lambda)$. Our lower bound on QCFs is actually saturated for some functions f, such as the trivial $f_{\mathrm{heads}}(\lambda) = \lambda^L$ and also the non-trivial, Eq (D.7).

VIII. CONCLUSION

In this paper, we have attempted to map some of the territory of Bernoulli factories. We have considered several different types of such factories, depending on what resource is given as input, what free resources and gates are allowed within the factory, and what output is demanded. The other general distinction we made was between unbounded algorithms, that might run forever with small probability, and query-limited algorithms.

One part of the quantum Bernoulli factory territory was particularly mappable, that of coin factories with limited uses of a resource oracle acting on a single-qubit of memory, which we called GCF$_1$. This is also an interesting part of the landscape from the viewpoint of query complexity, as quantum algorithms for symmetric boolean functions reduce to GCF$_1$ algorithms. We managed to completely characterize the possible coins constructible by GCF$_1$s. We also converted GCF$_1$ algorithms into QCF algorithms, which use quoins as a resource rather than an oracle, and proved a lower bound on QCF algorithms using Helstrom's bound.

Further questions abound, however. Can all symmetric boolean functions be evaluated optimally as GCF$_1$ algorithms, that is, by the Groverlike method of alternating reflections about the start state and target state in the subspace we call $\mathcal{T}$? More broadly, can characterization theorems like the one we proved for GCF$_1$ be proved for the other types of Bernoulli factories in Table I (or for others unlisted)? Finally, the connection between resource limited Bernoulli factories and those using unlimited resources is worth exploring.

Acknowledgements

The author would like to thank Guang Hao Low for useful discussion and for the proof of Lemma B.2. Special thanks go to Scott Aaronson for encouraging this project and teaching the course!

[1] H. Dale, D. Jennings, and T. Rudolph, Nature communications **6** (2015).
[2] J. Von Neumann, Appl. Math Ser. **12**, 36 (1951).
[3] M. Keane and G. L. O'Brien, ACM Transactions on Modeling and Computer Simulation (TOMACS) **4**, 213 (1994).
[4] M. Ozols, M. Roetteler, and J. Roland, ACM Transactions on Computation Theory (TOCT) **5**, 11 (2013).
[5] L. Grover and T. Rudolph, arXiv preprint quant-ph/0208112 (2002).
[6] G. H. Low, T. J. Yoder, and I. L. Chuang, Physical Review A **89**, 062315 (2014).
[7] S. Lloyd, M. Mohseni, and P. Rebentrost, Nature Physics **10**, 631 (2014).
[8] A. Ambainis, in *Proceedings of the thirty-second annual ACM symposium on Theory of computing* (ACM, 2000), pp. 636–643.
[9] P. Hoyer, T. Lee, and R. Spalek, in *Proceedings of the thirty-ninth annual ACM symposium on theory of computing* (ACM, 2007), pp. 526–535.
[10] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. De Wolf, Journal of the ACM (JACM) **48**, 778 (2001).
[11] R. Paturi, in *Proceedings of the twenty-fourth annual ACM symposium on theory of computing* (ACM, 1992), pp. 468–474.
[12] W. Markoff and J. Grossmann, Mathematische Annalen **77**, 213 (1916).
[13] L. K. Grover, in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing* (ACM, 1996), pp. 212–219.
[14] S. Aaronson and A. Drucker, in *Automata, Languages and Programming* (Springer, 2011), pp. 61–72.
[15] C. W. Helstrom, Journal of Statistical Physics **1**, 231 (1969).
[16] M. Marshall, *Positive polynomials and sums of squares*, 146 (American Mathematical Soc., 2008).
[17] K. Weierstrass, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin **2**, 633 (1885).
[18] F. Herzog and J. Hill, American Journal of Mathematics pp. 109–124 (1946).
[19] T. J. Yoder, G. H. Low, and I. L. Chuang, Physical review letters **113**, 210501 (2014).

Appendix A: Proof of Lemma V.1

Proof. A generic L-query GCF$_1$ algorithm can be written in the form of Eq. (V.1), which we repeat here for convenience.

$$\begin{aligned}\mathcal{G}|0\rangle &= A_\lambda P_{L-1}\dots A_\lambda P_2 A_\lambda P_1 A_\lambda|0\rangle && \text{(A.1)}\\ &= \sqrt{1-f(\lambda)}|0\rangle + e^{i\chi(\lambda)}\sqrt{\lambda}|1\rangle, && \text{(A.2)}\end{aligned}$$

where $P_j = \exp(-i\phi_j Z)$. Given this algorithm, we aim to construct a standard query algorithm for a symmetric boolean function g on N-bits that accepts string x with probability $f(|x|/N)$.

The idea is to use the subspace $\mathcal{T}$ that was so useful in Grover's algorithm, Section V A. Let $\lambda = |x|/N$, $n = \log N$ (for simplicity assume N is a power of two), and H be the single-qubit Hadamard gate. Also let U be the standard quantum oracle (i.e. $U|i\rangle|a\rangle|b\rangle = |i\rangle|a\oplus 1\rangle|b\rangle$). The correspondence of states and operators between GCF$_1$ and the standard oracle model is as follows:

$$\begin{aligned}|0\rangle &\leftrightarrow |\bar{t}\rangle = \frac{1}{\sqrt{N-|x|}}\sum_{x_i=0}|i\rangle = |\text{Rej}\rangle && \text{(A.3)}\\ |1\rangle &\leftrightarrow |t\rangle = \frac{1}{\sqrt{|x|}}\sum_{x_i=1}|i\rangle = |\text{Acc}\rangle && \text{(A.4)}\\ A_\lambda|0\rangle &\leftrightarrow (H|0\rangle)^{\otimes n} = \sqrt{1-\lambda}|\bar{t}\rangle + \sqrt{\lambda}|t\rangle = |s\rangle && \text{(A.5)}\\ e^{-i\theta Z} &\leftrightarrow U(I\otimes e^{-i\theta Z}\otimes I)U, && \text{(A.6)}\end{aligned}$$

where the tripartite system implicit in the final equation is the index, ancilla, bystander partition used in the definition of U. If we define also the reflections, $R_s(\alpha) = I-(1-e^{-i\alpha})|s\rangle\langle s|$ and $R_t(\beta) = I-(1-e^{i\beta})|t\rangle\langle t|$ where $|s\rangle = \sqrt{1-\lambda}|\bar{t}\rangle+\sqrt{\lambda}|t\rangle$, then Eqs. (A.5) and (A.6) imply also the correspondences,

$$\begin{aligned}A_\lambda e^{i\alpha Z/2}A_\lambda^\dagger &\longleftrightarrow e^{i\alpha/2}R_s(\alpha) && \text{(A.7)}\\ e^{-i\beta Z/2} &\longleftrightarrow e^{-i\beta/2}R_t(\beta). && \text{(A.8)}\end{aligned}$$

Using Eqs. (A.7) and (A.8) repeatedly converts the GCF$_1$ algorithm in Eq. (A.1) into a standard query algorithm $\mathcal{C}$ taking place in $\mathcal{T}$. Output accept or reject by applying the oracle U, and measuring the ancilla – 0 implies reject and 1 implies accept. If L is odd, use Eq. (A.5) to convert the initial state $A_\lambda|0\rangle$ into $|s\rangle$. If L is even, then applying the oracle U to $|s\rangle$ and measuring the ancilla will collapse the state to either $|t\rangle$ or $|\bar{t}\rangle$, and either can be used as the initial state. (But, note, if $|t\rangle$ is obtained then proceed with the same algorithm $\mathcal{C}$, but reject if $|\text{Acc}\rangle$ is found and accept if $|\text{Rej}\rangle$ is.)

■

Appendix B: Proof of Theorem V.2

The characterization of GCF$_1$ is the most involved proof of this report. First, we introduce a well-known result from mathematics on polynomials and provide a proof following [16].

Lemma B.1. *(polynomial SoS)*
Suppose $f : \mathbb{R} \to \mathbb{R}$ is a real polynomial. Then $f(x) = g(x)^2 + h(x)^2$ for real polynomials g, h iff $f(x) \geq 0$ for all $x \in \mathbb{R}$.

Proof. The forward implication is obvious. For the reverse implication, we begin by factoring f into irreducible polynomials over the reals as

$$f(x) = \alpha \prod_i (x - \beta_i)^{k_i} \prod_j ((x - \gamma_j)^2 + \delta_j^2)^{l_j}. \tag{B.1}$$

Because $f(x) \geq 0$ for all x, we must have k_i even for all i and also $\alpha \geq 0$. Therefore, $\sqrt{\alpha}, (x - \beta_i)^{k_i/2} \in \mathbb{R}$ for all x and i. Furthermore, the second product in the factorization of f can be reduced to a single sum of two squares by repeated application of the identity (called the 'two squares identity' in [16])

$$(r^2 + s^2)(t^2 + u^2) = (rt \pm su)^2 + (ru \mp st)^2, \tag{B.2}$$

where the possible choice of upper signs or lower signs implies that the decomposition of f into a sum of two squares is not unique.

■

Now we can begin studying L-query GCF$_1$ algorithms, whose general form we repeat here,

$$\mathcal{G} = A_\lambda P_{L-1} \dots A_\lambda P_2 A_\lambda P_1 A_\lambda, \tag{B.3}$$

where recall $P_j = \exp(-i\phi_j Z)$ are phase gates. Because $\mathcal{G}$ will be applied to the Z-eigenstate $|0\rangle$ and the final output measured in the Z-basis, we can drop leading and trailing Z-rotations, and convert $\mathcal{G}$ into the following form,

$$\mathcal{G} = R_{\phi_L}(\lambda) R_{\phi_{L-1}}(\lambda) \dots R_{\phi_1}(\lambda), \tag{B.4}$$

where $R_\phi(\lambda) = \sqrt{1-\lambda} I - i\sqrt{\lambda}(\cos(\phi)X + \sin(\phi)Y)$, and we used the facts that A_λ is a single qubit Y-rotation (i.e. $A_\lambda = \exp(-i\theta Y/2)$ for θ defined such that $\sqrt{\lambda} = \sin(\theta/2)$ and $\sqrt{1-\lambda} = \cos(\theta/2)$) and also that

$$e^{-i(\phi/2+3\pi/4)Z} e^{-i\theta Y/2} e^{i(\phi+3\pi/4)Z} = R_\phi(\lambda). \tag{B.5}$$

Now we have the following Lemma.

Lemma B.2. *(1-qubit compiling)*
A unitary $U = a(\lambda)I - ib(\lambda)Z - ic(\lambda)X - id(\lambda)Y$ can be written as a sequence of L rotations around axes in the XY-plane

$$U = R_{\phi_L}(\lambda) R_{\phi_{L-1}}(\lambda) \dots R_{\phi_1}(\lambda), \tag{B.6}$$

if and only if the Pauli coefficients can be written as

$$a(\lambda) = \sum_{\substack{k=0 \\ k \text{ even}}}^{L} a_k B_{L,k}(\lambda), \quad b(\lambda) = \sum_{\substack{k=0 \\ k \text{ even}}}^{L} b_k B_{L,k}(\lambda),$$
$$c(\lambda) = \sum_{\substack{k=0 \\ k \text{ odd}}}^{L} c_k B_{L,k}(\lambda), \quad d(\lambda) = \sum_{\substack{k=0 \\ k \text{ odd}}}^{L} d_k B_{L,k}(\lambda), \tag{B.7}$$

where $B_{L,k}(\lambda) = \sqrt{1-\lambda}^{L-k} \sqrt{\lambda}^k$ and a_k, b_k, c_k, d_k are real numbers, satisfying $a(\lambda)^2 + b(\lambda)^2 + c(\lambda)^2 + d(\lambda)^2 = 1$.

Proof. The forward direction proceeds by induction. Let $a^{(p)}(\lambda), b^{(p)}(\lambda), c^{(p)}(\lambda), d^{(p)}(\lambda)$ be the Pauli coefficients of the unitary formed from the first p rotations. That is,

$$a^{(p)} I + ib^{(p)} Z + ic^{(p)} X + id^{(p)} Y = R_{\phi_p}(\lambda) \dots R_{\phi_1}(\lambda). \tag{B.8}$$

Then, by multiplying out the unitaries, one can easily check that (suppressing function arguments λ),

$$\begin{aligned}
a^{(p+1)} &= \sqrt{1-\lambda} a^{(p)} + \sqrt{\lambda} \cos\phi_{p+1} c^{(p)} + \sqrt{\lambda} \sin\phi_{p+1} d^{(p)}, \\
b^{(p+1)} &= \sqrt{1-\lambda} b^{(p)} + \sqrt{\lambda} \cos\phi_{p+1} d^{(p)} - \sqrt{\lambda} \sin\phi_{p+1} c^{(p)}, \\
c^{(p+1)} &= \sqrt{1-\lambda} c^{(p)} - \sqrt{\lambda} \cos\phi_{p+1} a^{(p)} + \sqrt{\lambda} \sin\phi_{p+1} b^{(p)}, \\
d^{(p+1)} &= \sqrt{1-\lambda} d^{(p)} - \sqrt{\lambda} \cos\phi_{p+1} b^{(p)} - \sqrt{\lambda} \sin\phi_{p+1} a^{(p)}.
\end{aligned} \tag{B.9}$$

The base case is $p = 1$, for which $a^{(1)} = \sqrt{1-\lambda}, b^{(1)} = 0, c^{(1)} = -\sqrt{\lambda}\cos\phi_1$, and $d^{(1)} = -\sqrt{\lambda}\sin\phi_1$ evidently satisfy Eq. (B.7) of the lemma. The condition that the squares sum to one, follows directly from unitarity of U.

The reverse implication is more complicated, but reverts ultimately to algebra once one has the right idea. We have to find phases ϕ_j that will appropriately construct U given the Pauli coefficients a, b, c, d of the form specified in Eq. (B.7). Roughly, the trick is to again use the recursion relations for a, b, c, d given in Eq. (B.9), but with the goal of *reducing* the polynomial degree of a, b, c, d. That is, find the Pauli coefficients of $R_{\phi_1}(\theta) U^\dagger$ using the recursion relations, and choose ϕ_1 to cancel the highest order terms (in λ) in those coefficients. Repeat L times to find all the phases. We make this procedure explicit next.

Note first that writing $a(\lambda), b(\lambda), c(\lambda)$, and $d(\lambda)$ in the power monomial basis is more convenient for canceling leading orders in λ. That is, for $L = 2l + 1$ odd,

$$a(\lambda) = \sum_{j=0}^{l} \tilde{a}_{2j+1} \sqrt{1-\lambda}^{2j+1}, \tag{B.10}$$

$$b(\lambda) = \sum_{j=0}^{l} \tilde{b}_{2j+1} \sqrt{1-\lambda}^{2j+1}, \tag{B.11}$$

$$c(\lambda) = \sum_{j=0}^{l} \tilde{c}_{2j+1} \sqrt{\lambda}^{2j+1}, \tag{B.12}$$

$$d(\lambda) = \sum_{j=0}^{l} \tilde{d}_{2j+1} \sqrt{\lambda}^{2j+1}, \tag{B.13}$$

where $\tilde{a}_h, \tilde{b}_h, \tilde{c}_h, \tilde{d}_h$ are simple linear combinations of a_h, b_h, c_h, d_h, the coefficients from Eq. (B.7). For $L = 2l$

even, we likewise define tilde coefficients such that,

$$a(\lambda) = \sum_{j=0}^{l} \tilde{a}_{2j} \sqrt{1-\lambda}^{2j}, \tag{B.14}$$

$$b(\lambda) = \sum_{j=0}^{l} \tilde{b}_{2j} \sqrt{1-\lambda}^{2j}, \tag{B.15}$$

$$c(\lambda) = \sqrt{\lambda(1-\lambda)} \sum_{j=0}^{l-1} \tilde{c}_{2j} \sqrt{\lambda}^{2j}, \tag{B.16}$$

$$d(\lambda) = \sqrt{\lambda(1-\lambda)} \sum_{j=0}^{l-1} \tilde{d}_{2j} \sqrt{\lambda}^{2j}. \tag{B.17}$$

Note that in this form, whether L is even or odd, the unitarity condition that $a(\lambda)^2 + b(\lambda)^2 + c(\lambda)^2 + d(\lambda)^2 = 1$ implies that $\tilde{a}_L^2 + \tilde{b}_L^2 = \tilde{c}_L^2 + \tilde{d}_L^2$.

Now let $a^{(p')}(\lambda), b^{(p')}(\lambda), c^{(p')}(\lambda), d^{(p')}(\lambda)$ be the Pauli coefficients of $R_{\phi_p}(\lambda) \ldots R_{\phi_1}(\lambda) U^\dagger$. For instance, $a^{(0')} = a, b^{(0')} = b, c^{(0')} = c$, and $d^{(0')} = d$ is the base case, and the recursion relation Eq. (B.9) (replace unprimed superscripts with primed superscripts) defines the rest.

Now we go through finding ϕ_1 in the case that $L = 2l + 1$ is odd. The even case is very similar, and finding all ϕ_p for p larger than 1 also follows the same logic. First, expand the Pauli coefficients,

$$\begin{aligned} a^{(1')} =& (1-\lambda)^{l+1} \left(\tilde{a}_L - (-1)^l (\tilde{c}_L \cos\phi_1 + \tilde{d}_L \sin\phi_1) \right) \\ &+ O\left((1-\lambda)^l\right), \\ b^{(1')} =& (1-\lambda)^{l+1} \left(\tilde{b}_L - (-1)^l (\tilde{d}_L \cos\phi_1 - \tilde{c}_L \sin\phi_1) \right) \\ &+ O\left((1-\lambda)^l\right), \\ c^{(1')} =& \sqrt{\lambda(1-\lambda)} \big(\lambda^l (\tilde{c}_L - (-1)^l (\tilde{a}_L \cos\phi_1 - \tilde{b} \sin\phi_1)) \\ &+ O(\lambda^{l-1}) \big), \\ d^{(1')} =& \sqrt{\lambda(1-\lambda)} \big(\lambda^l (\tilde{c}_L - (-1)^l (\tilde{b}_L \cos\phi_1 + \tilde{a} \sin\phi_1)) \\ &+ O(\lambda^{l-1}) \big). \end{aligned}$$

Now it is easy to check that making the leading order terms disappear simply requires choosing ϕ_1 so that,

$$e^{-i\phi_1} = (-1)^l \frac{\tilde{a}_L + i\tilde{b}_L}{\tilde{c}_L + i\tilde{d}_L}. \tag{B.18}$$

Note also that $\phi_1 \in \mathbb{R}$ because $\tilde{a}_L^2 + \tilde{b}_L^2 = \tilde{c}_L^2 + \tilde{d}_L^2$.

In general (and showing this just requires doing the algebra similar to the $p = 1$, L odd case above), one should choose ϕ_{j+1} according to

$$e^{-i\phi_{j+1}} = (-1)^{\lceil (L-j)/2 \rceil - 1} \frac{\tilde{a}_{L-j}^{(j')} + i\tilde{b}_{L-j}^{(j')}}{\tilde{c}_{L-j}^{(j')} + i\tilde{d}_{L-j}^{(j')}}, \tag{B.19}$$

where $\tilde{a}_{L-j}^{(j')}$ for instance is the largest degree coefficient of $a^{(j')}(\lambda)$ when $a^{(j')}$ is expanded as in Eq. (B.10) or Eq. (B.14) depending on the parity of j. This is an iterative procedure for finding the phases ϕ_j.

■

Because Lemma B.2 is concerned with making a single-qubit unitary from the oracle A_λ, it is in some sense a characterization of Groverlike Oracle Factory (GOF) restricted to one qubit of memory (GOF$_1$), the first example we have given and discussed of a Bernoulli factory that makes operators rather than states. The possibility of creating quoins as outputs (therefore GQF$_1$s) using Lemma B.2 should also be obvious.

In any case, returning to coin factories, we can now finally complete the proof of Theorem V.2 and therefore the characterization of GCF$_1$ by combining Lemmas B.1 and B.2.

Proof of Theorem V.2. To use the notation of Lemma B.2, note that $f(\lambda) = c(\lambda)^2 + d(\lambda)^2$, which implies that $1 - f(\lambda) = a(\lambda)^2 + b(\lambda)^2$. Thus, by Lemma B.2, to prove that f is GCF$_1$ constructible we must only find functions a, b, c, d of the form given in Eq. (B.7). There are two cases to consider – deg(f) odd and deg(f) even – which are similar, but we will treat them separately for clarity.

Odd degree: In this case, L must be odd – even L cannot generate an even degree polynomial bias f, as can be checked by inspecting Eq. (B.7). In the odd L case, we also note that $c(\lambda) = \tilde{c}(\sqrt{\lambda})$ for an odd polynomial $\tilde{c}$. Likewise, there are odd polynomials $\tilde{a}, \tilde{b}$, and $\tilde{d}$ such that $a(\lambda) = \tilde{a}(\sqrt{1-\lambda})$, $b(\lambda) = \tilde{b}(\sqrt{1-\lambda})$, and $d(\lambda) = \tilde{d}(\sqrt{\lambda})$.

For the forward implication of the theorem, we need to prove the stated conditions on f assuming f is GCF$_1$ constructible. First, f is certainly a polynomial in λ because $\tilde{c}$ and $\tilde{d}$ are both odd polynomials in $\sqrt{\lambda}$. Also, $f(\lambda) \in [0, 1]$ for all $\lambda \in [0, 1]$, because $a(\lambda), b(\lambda), c(\lambda)$ and $d(\lambda)$ are real valued for all $\lambda \in [0, 1]$. For all $\lambda < 0$, we see $c(\lambda)^2, d(\lambda)^2 \leq 0$ and for all $\lambda > 1$, we have $a(\lambda)^2, b(\lambda)^2 \leq 0$. This implies $f(\lambda) \leq 0$ for all $\lambda < 0$ and $f(\lambda) \geq 1$ for all $\lambda > 1$.

For the reverse implication things are only slightly more complicated. We will first show the existence of odd-polynomials $\tilde{c}, \tilde{d}$ such that $f(\lambda) = \tilde{c}(\sqrt{\lambda})^2 + \tilde{d}(\sqrt{\lambda})^2$. It is easy to show that any odd polynomials can be put into the form required by Eq. (B.7).

To show that $\tilde{c}, \tilde{d}$ exist, factor $f(\lambda)$ as

$$f(\lambda) = \alpha \prod_i (\lambda - \beta_i)^{k_i} \prod_j \left((\lambda - \gamma_j)^2 + \delta_j^2 \right)^{l_j}. \tag{B.20}$$

Now we know that $f(\lambda) \geq 0$ for all $\lambda \geq 0$, so all k_i are even if $\beta_i > 0$. On the other hand, if any k_i is odd while $\beta_i < 0$ then $f(\beta_i) = 0$ and the k_i-derivative $f^{(k_i)}(\beta_i) \neq 0$. But this would mean that somewhere near β_i (say at $\beta_i + \epsilon < 0$) we would have $f(\beta_i + \epsilon) > 0$, which violates condition (1). Likewise, we must have a factor of λ^{k_0} in the factorization of f, and k_0 better be odd – otherwise, f would not cross zero at $\lambda = 0$ and so violate either condition (1) or the hypothesis that $f(\lambda) \in [0, 1]$ for all $\lambda \in [0, 1]$.

Having settled the parities of the k_i, it is easy to break f into the sum of two squares of odd polynomials in $\sqrt{\lambda}$, making use of the two squares identity, Eq. (B.2), on the latter factors of Eq. (B.20).

Condition (2) provides, by a similar argument, the proof that $\tilde{a}$ and $\tilde{b}$ exist and are odd polynomials. Simply factor $g(1-\lambda) = 1 - f(\lambda)$ instead.

Even degree: In this case, L is even. Also, inspection of Eq. (B.7) shows that $c(\lambda)$ and $d(\lambda)$ may be written as $c(\lambda) = \sqrt{\lambda(1-\lambda)}\tilde{c}(\sqrt{\lambda})$ and $d(\lambda) = \sqrt{\lambda(1-\lambda)}\tilde{d}(\sqrt{\lambda})$ for even polynomials $\tilde{c}$ and $\tilde{d}$. Likewise, $a(\lambda) = \tilde{a}(\sqrt{\lambda})$ and $b(\lambda) = \tilde{b}(\sqrt{\lambda})$ for even polynomials $\tilde{a}$ and $\tilde{b}$.

Now we prove the forward implication. Evidently, $f(\lambda)$ is a polynomial and $f(\lambda) \in [0,1]$ for all $\lambda \in [0,1]$. Also, because $f(\lambda) = \lambda(1-\lambda)\left(\tilde{c}(\sqrt{\lambda})^2 + \tilde{d}(\sqrt{\lambda})^2\right)$ and $\tilde{c}, \tilde{d}$ are even polynomials (and so actually functions of λ), we see that $f(\lambda) \leq 0$ for all $\lambda < 0$ and $\lambda > 1$.

For the reverse implication, we have to use the polynomial SoS Lemma B.1 again. Let $g(\lambda) = f(\lambda)/\lambda(1-\lambda)$. By the hypotheses, f has roots at $\lambda = 0$ and $\lambda = 1$, so g is also a polynomial. Indeed, we know also that $g(\lambda) \geq 0$ for all $\lambda \in \mathbb{R}$. Thus, by the SoS lemma there are two polynomials in λ such that $g(\lambda)$ is the sum of their squares. This gives us $\tilde{c}$ and $\tilde{d}$.

To get $\tilde{a}$ and $\tilde{b}$, simply note that $1 - f(\lambda) \geq 0$ for all $\lambda \in \mathbb{R}$. Thus, the SoS lemma implies the existence of $\tilde{a}$ and $\tilde{b}$. Similar to the odd case, once $\tilde{a}, \tilde{b}, \tilde{c}$, and $\tilde{d}$ are found it is a matter of a linear transformation of their polynomial coefficients to put them in the form required by Eq. (B.7).

■

Appendix C: Bernstein approximations

In the query limited case, we have already seen that the bias $f(\lambda)$ of an output coin will always be a polynomial in the input coin's bias λ. However, we may suspect that because of Weierstrass' theorem [17], which states that arbitrary continuous functions can be well approximated by large degree polynomials, that we can find a sequence of query limited algorithms, making say $L_1 < L_2 < L_3 < \dots$ queries whose outputs get arbitrarily close to a coin with desired non-polynomial bias.

We present some preliminary results towards this goal in this section using Bernstein approximations. The L^{th} Bernstein approximation to a function $f(\lambda)$ is the polynomial

$$B_L\left[f(\lambda)\right] = \sum_{j=0}^{L} f(j/L)\binom{L}{j}\lambda^j(1-\lambda)^{L-j}. \qquad \text{(C.1)}$$

As L increases, the Bernstein approximations approach $f(\lambda)$ uniformly, even for some discontinuous functions f [18]. The Bernstein approximations are polynomials so we at least have a chance of constructing them within GCF$_1$ for any function f and any L. But whether, for all $f(\lambda)$, there exist $L_1 < L_2 < L_3 < \dots$ such that $B_{L_j}\left[f(\lambda)\right]$ are constructible polynomials (e.g. within OCF, GCF, or GCF$_1$) is non-obvious.

At least for specific functions, we can find a sequence of Bernstein approximations that are constructible in GCF$_1$. Take the classic Bernoulli factory example $f_{\text{sqrt}}(\lambda) = \sqrt{\lambda}$. The Bernoulli approximations $B_L[f_{\text{sqrt}}]$ with odd L are 0 at $\lambda = 0$, are 1 at $\lambda = 1$, and have non-negative derivative. Therefore, they satisfy the conditions of Theorem V.2 and therefore $B_L[f_{\text{sqrt}}]$-coins are constructible in GCF$_1$.

We know of numerous other examples of GCF$_1$ constructible Bernstein approximations. For instance, the von-Neumann coin functions f_{vN} in the $p = 1 - \delta^2/2$ and $\delta \to 0$ limit become Bernstein approximations to the function $f_{\text{OR}} = \left\{\begin{smallmatrix} 0, & \lambda=0 \\ 1, & \lambda>0 \end{smallmatrix}\right.$. Other patterns of constructible L_j (rather than simply "L odd") also exist. For instance, the Bernstein approximations to $f_{\text{MAJ}}(\lambda) = \left\{\begin{smallmatrix} 0, & \lambda\leq 1/2 \\ 1, & \lambda>1/2 \end{smallmatrix}\right.$ are constructible in GCF$_1$ for all $L \equiv 1 \pmod 4$. A general case encompassing at least these two cases and possibly worth further consideration might be the Bernstein approximations to the threshold functions $\theta_t(\lambda) = \left\{\begin{smallmatrix} 0, & \lambda\leq t \\ 1, & \lambda>t \end{smallmatrix}\right.$.

Appendix D: Proof and examples of the QCF lower bound

Here we prove Theorem IV.1.

Proof. Consider two input quoins $|\psi_{\lambda_1}\rangle$ and $|\psi_{\lambda_2}\rangle$ with $\lambda_2 = \lambda_1 + \Delta\lambda$, and say we have a QCF producing $f(\lambda)$-coins using k quoins. Given km copies of an unknown state promised to be either $|\psi_{\lambda_1}\rangle$ or $|\psi_{\lambda_2}\rangle$, we can determine which we have by sampling m coins using the QCF, getting binary values $x_1, x_2, \dots x_m$. If $(\sum_i x_i)/m$ is closer to λ_1 than λ_2, we report that we were given the state $|\psi_{\lambda_1}\rangle$, and otherwise we report $|\psi_{\lambda_2}\rangle$. This procedure has error probability $\epsilon \leq e^{-m\Delta f^2/2}$ bounded by Hoeffding's inequality, where $\Delta f = f(\lambda_2) - f(\lambda_1)$.

This procedure for distinguishing states must be limited by Helstrom's bound, Lemma VII.1. In this case, $1 - |\langle\psi_{\lambda_1}|\psi_{\lambda_2}\rangle|^2 = \Delta\lambda^2/4\lambda(1-\lambda) + O(\Delta\lambda^3)$, where we have set $\lambda = \lambda_1$. So by Hoeffding and Helstrom we have $mk = 2k\ln(1/\epsilon)/\Delta f^2 \geq 4\lambda(1-\lambda)\ln(1/4\epsilon(1-\epsilon))/\Delta\lambda^2$. Thus,

$$k \geq 2\frac{\ln(4\epsilon(1-\epsilon))}{\ln(\epsilon)}\left(f'(\lambda)^2\lambda(1-\lambda)\right). \qquad \text{(D.1)}$$

Treating ϵ as a (small) constant and noticing that the bound on k holds for all λ, we arrive at the theorem.

■

Now we provide some examples of using this bound.

Ex. 1 Consider the bias function for L odd,

$$f(\lambda) = \sum_{j=0}^{(L-1)/2}\binom{L}{j}(1-\lambda)^j\lambda^{L-j}, \qquad \text{(D.2)}$$

which is the L^{th} Bernstein approximation to the function $f_{\text{MAJ}}(\lambda)$ introduced at the end of Section C. It is simple to check that the maximum of $f'(\lambda)$ always occurs at $\lambda = 1/2$ (this is where f_{MAJ} changes most rapidly after all). Also, the maximum of $\lambda(1-\lambda)$ occurs at $\lambda = 1/2$. Therefore, the bound on the number of copies k of the quoin $|\psi_\lambda\rangle$ needed to create a $f(\lambda)$-coin is

$$\begin{aligned} k &= \Omega\left(\max_{\lambda}\left(f'(\lambda)^2\lambda(1-\lambda)\right)\right) && \text{(D.3)} \\ &= f'(1/2)^2/4 = \Omega(L). && \text{(D.4)} \end{aligned}$$

This is in fact an optimal lower bound, because the algorithm to create $f(\lambda)$ is a classical one – simply sample k quoins (they may just as well be coins) and report 0 or 1 depending on the majority of those samples.

Example 1 gave a trivial bound, because the degree of f was L and so we already know that at least L copies are required to construct it. Example 2 is not so trivial.

Ex. 2 Let the bias be the Grover bias,

$$f(\lambda) = \sin^2\left(L\cos^{-1}\sqrt{1-\lambda}\right). \tag{D.5}$$

Therefore, $f'(\lambda) = \frac{L\sin(2L\cos^{-1}\sqrt{1-\lambda})}{2\sqrt{\lambda(1-\lambda)}}$, and so we obtain the bound

$$k = \Omega\left(L^2\right). \tag{D.6}$$

This bound implies that the quadratic speedup of Grover's algorithm is lost if instead of A_λ we have only the start states $|\psi_\lambda\rangle$ as our resources (i.e. in the QCF model rather than the OCF model). Similar to Example 1, this bound is satisfiable by a naive classical algorithm – simply sample k quoins and report 1 if any of them are found to be 1.

However, the satisfying algorithm is not even classical if we generalize the bias in Eq. (D.5) to create a bias

$$f_{\text{fp}}(\lambda) = 1 - \delta^2 T_L\left[T_{1/L}[1/\delta]\sqrt{1-\lambda}\right]^2, \tag{D.7}$$

which is the fixed-point algorithm of [19] and becomes the Grover bias in the $\delta = 1$ limit. In this case, the lower bound on the number of quoins needed can also be shown to be $k = \Omega(L^2)$, but f_{fp} is not even in CCF for some values of δ (because the coefficients in the polynomial f_{fp} will be non-integer). So in this case, an optimal (approximate) algorithm can be created using the method of Theorem VI.1 to convert the $O(L)$ query GCF$_1$ algorithm into a $O(L^2)$ query QCF algorithm.

Ex. 3 Say we have a non-constant symmetric boolean function $g : \{0,1\}^N \to \{0,1\}$ and we want a $f(\lambda)$-coin such that $f(\lambda)$ approximates g. Let's also say that g changes value at the hamming weights $h_1, h_2, \ldots$ (i.e. $g_\sim(h_j - 1) \neq g_\sim(h_j)$) and that h^* is the hamming weight h_j closest to $N/2$. By the mean-value theorem, there is some $\lambda^* \in [(h^*-1)/N, h^*/N]$ such that $f'(\lambda^*) = 1/3N$. So,

$$\max_{\lambda}\left[f'(\lambda)^2\lambda(1-\lambda)\right] \geq 9N^2\lambda^*(1-\lambda^*). \tag{D.8}$$

This lower bound on QCFs is essentially the square of the lower bound proved by Beals et. al. Theorem II.1 on oracle queries in the query model. Note that in a Bernoulli factory, there is no such thing as an input string consisting of N-bits, so there is no natural upper bound of N queries to "learn" the input; the input is instead the coin, quoin, or oracle, which depends on a continuous bias parameter λ that can never be exactly learned.