Quantum Computing Notes

Alex Creiner

September 14, 2021

1 The Linear Transformation of a Turing machine

Consider an arbitrary Turing machine. This machine has a countably infinite number of configurations associated with it. Enumerate all such configurations $\{c_i\}_{i\in\omega}$. Note that our machine being deterministic, it follows that c_i will yield c_j for any i. That is, for any i, there is a unique j such that c_j is the configuration yielded in one step by the machine. Consider the "matrix" (linear transformation) M, whose i^{th} column is all 0's, except for the entry in the j^{th} row, which is a 1, representing that i^{th} configuration yields the j^{th} configuration in a single step my the machine. This transformation M then to some extent *is* the machine, in the sense that given an initial configuration $\vec{v_i}$, which we can represent as itself a vector of the sort described, then the configuration yielded by the machine in a single step is $M\vec{v_i}$, and so forth and so on. We have thus discovered a new representation of Turing machines - as linear transformations.

What kind of a matrix is M? It certainly seems at first glance like it might be unitary, since each column is already a unit vector of all 0's except for a single 1. However, it need not be, since Turing machines need not be *reversible*, in the sense that there may easily be several configurations which all yield the same configuration in a step, in which case our matrix would have several of the same column. The following lemmas aim to show that any Turing machine can be thought of as reversible with negligible efficiency loss, and that the property of reversibility is indeed sufficient in this initial deterministic case to showing that the associated transformations are unitary [TODO]

2 Probabilistic Turing machines

There are several ways to define formally what probabilistic Turing machines 'are'. We will start with the one most closely related to the 'standard' theory.

Definition 2.1. A **probabilistic Turing machine** N is a nondeterministic Turing machine with the additional properties of being precise (that is, all computational paths are the exact same length), and whose transition relation yields exactly two possibilities for any configuration encountered (with the exception, perhaps, of halting configurations).

Informally, what we are saying here is that probabilistic Turing machines are simply deterministic machines with the added feature of being able to 'flip a coin' at each step. (Not flipping a coin here would simply mean having the same transition whether the coin is tails or heads.) From these machines we can define several interesting complexity classes immediately:

Definition 2.2. Let L be a language. Suppose there exists a probabilistic Turing machine N which operates in polynomial time $p(n)$, with the following relation to L: If $x \in L$, then *at least half* of the $2^{p(n)}$ computations on x halt in acceptance, and if $x \notin L$, then *all* computations halt in rejection. Then we call N a polynomial time **Monte Carlo machine** for L. We denote the class of problems with polynomial time Monte Carlo machines $\boldsymbol{RP}$.

(Define the others)

The behavior of a probabilistic Turing machine (indeed of any Turing machine) is that it is essentially memoryless, in the sense that it's behavior at any particular step is completely determined by it's present configuration. Past configurations have no influence at all - this information has no bearing at all on the next step. In this sense, there is a Markov process implicit to every Turing machine. Specifically, if we label the countable set of configurations $\{c_1, c_2, ...\}$, then we can imagine a countably infinite dimensional stochastic matrix (that is, a matrix whose entries sum to one for each column vector) M, where the i^{th} column of M specifies the probability distribution of the configurations which may be entered on the next step, given the machine is presently in configuration c_i. To be more specific, let

$$M = (\vec{p}_1, \vec{p}_2, \ldots)$$

where for each i we have

$$\vec{p}_i = \begin{pmatrix} p_{i,1} \\ p_{i,2} \\ \vdots \end{pmatrix}$$

and the entry $p_{i,j}$ represents the probability that the machine enters configuration c_i given it is presently in configuration c_j. Obviously initially the machine is in a single, known configuration, c_1. Thus we have the initial 'probability distribution'

$$\vec{e} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

(The choice of making the initial configuration the first in our indexing is arbitrary and only assumed for descriptive convenience.) The probability distribution after a single step of the machine will then be $\vec{p_1} = M\vec{e}$.

Suppose we have reached step n of the computation of the machine N. We at this step have a probability distribution of the configurations

$$\vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \end{pmatrix}$$

What is the distribution of the machine after the next step? Let $\vec{r}$ denote the probability distribution for the machine after this present step. Consider r_i, the probability that the machine will be in configuration i. By the law of total probability, we have that

$$r_i = p_{i,1}q_1 + p_{i,2}q_2 + \ldots$$

That is to say, the inner product of the i^{th} row of M with $\vec{q}$. Thus we have that $\vec{r} = M\vec{q}$, and can say in general that the probability distribution of configurations evolves via repeated multiplication by the matrix M.

While every Turing machine and probabilistic Turing machine has an associated stochastic matrix, not every stochastic matrix has an associated Turing machine. To immediately understand why, note that based on our definition of a probabilistic Turing machine, the column vector $\vec{p}_i$ only has two nonzero entries, and thus despite it being infinite dimensional, the matrix M is very sparse and qualitatively finite in a certain sense. For a conventional Turing machine, the matrix is even sparser: each vector is just a standard basis vector. It's worth noting in this most simple of cases that the matrix is *not* necessarily a permutation matrix, and not necessarily unitary: since two configurations can easily end up yielding the same configuration in the next step, we will likely have the same basis vector appearing as multiple columns of M. Whether ordinary Turing machines can be without loss of generality assumed to have unitary matrix representations will become an important consideration soon.

The passage from Turing machines to quantum Turing machines can be seen as a *careful* generalization of these matrices to a broader set of stochastic matrices, and then a substitution of real numbers for complex 'amplitudes'. At present, probabilities in this matrix are bound to be in the set $\{0, \frac{1}{2}, 1\}$. The most general relaxation of this restriction - to all real numbers - would yield magic machines which can compute many

uncomputable problems, since this would allow hard or impossible to compute quantities into the machine. For instance, consider the real number x whose binary expression has the property that the i^{th} bit of x tells us whether or not the a machine M halts on a particular input, with i indexing the particular machine input pair. With this one amplitude hard-coded into the machine we would immediately imagine how to construct machines which use this to solve the halting problem. It turns out that this set of three probabilities is sufficient.

3 Quantum Turing Machines

Our passage from Turing machines to quantum Turing machines lies in the generalization of this matrix representation to complex amplitudes. For the sake of simplicity, we will begin with a finite alphabet Σ (usually $\{0, 1, B\}$, with the blank symbol specifically designated), assume the machine has a single two-way infinite tape, and that the machine's tape head cannot 'stay still' (i.e. will always move left or right.) We will index the cells by $\mathbb{Z}$, and standardize initial configurations such that the tape head always begins at cell 0, in state q_0, and has the input $x \in (\Sigma - B)^*$ written in positions $0, 1, 2, ..., |x|$. Given two states $|\phi\rangle$, $|\phi'\rangle$, we can speak of the probability distributions $\mathcal{D}$ and $\mathcal{D}'$ such that measurement produces samples from $\mathcal{D}$ and $\mathcal{D}'$ respectively. Recall that the total variation distance between two distributions is largest possible difference between the probabilities that measurement produces a particular result. The following relates the Euclidean difference between states directly to the similarity of their accompanying distributions. Recall also that when the set of possible outcomes is countable, the total variation distance can be also expressed as $\frac{1}{2}\sum_{\text{outcomes } \omega} |\mathcal{D}(\omega) - \mathcal{D}'(\omega)|$.

Lemma 3.1. *If two unit length superpositions $|\phi\rangle$ and $|\psi\rangle$ are within Euclidean distance ϵ of each other (i.e. $||\phi - \psi|| \leq \epsilon$), then the total variation distance between the probability distributions resulting from measurement is at most 4ϵ.*

Proof. Let $\phi = \sum_i \alpha_i |i\rangle$ and $\psi = \sum_i \beta_i |i\rangle$. That is, observing ϕ gives $|i\rangle$ with probability $|\alpha_i|^2$, and observing ψ gives $|i\rangle$ with probability $|\beta_i|^2$. Let $\sigma = \psi - \phi = \sum_i (\beta_i - \alpha_i) |i\rangle = \sum_i \sigma_i |i\rangle$. Note that $(\alpha_i + \sigma_i)(\alpha_i + \sigma_i)^* = \beta_i \beta_i^* = |\beta_i|^2$. Using this we can see

$$|\beta_i|^2 = |\alpha_i|^2 + |\sigma_i|^2 + \alpha_i \sigma_i^* + \sigma_i \alpha_i^*$$

For notational convenience define for a vector $a = \sum_i a_i |i\rangle$, the vector $abs(a) = \sum_i |a_i| |i\rangle$, and note that for any vector we have $|a| = |abs(a)|$. Now by the identity we mentioned above, the total variation is

$$\begin{aligned}
\frac{1}{2}\sum_i ||\beta_i|^2 - |\alpha_i|^2| = \frac{1}{2}\sum_i |\sigma_i|^2 + |\alpha_i \sigma_i^*| + |\sigma_i \alpha_i^*| &= \sum_i |\sigma_i|^2 + \sum_i |\alpha_i||\sigma_i^*| + \sum_i |\sigma_i||\alpha_i^*| && (1)\\
&= ||\sigma||^2 + \sum_i |\alpha_i||\sigma_i| + \sum_i |\sigma_i||\alpha_i| && (2)\\
&= ||\sigma||^2 + 2\langle abs(\alpha)|abs(\sigma)\rangle && (3)\\
&\leq \epsilon^2 + 2||abs(\alpha)||||abs(\sigma)|| && (4)\\
&= \epsilon^2 + 2||\sigma|| && (5)\\
&\leq \epsilon^2 + 2\epsilon && (6)
\end{aligned}$$

But since α and β are unit vectors, $\epsilon \leq 2$, and so $\epsilon^2 \leq 2\epsilon$, completing the proof. □

For the following, in order to define a test language we will use the length preserving function ξ_A defined in the main research notes. For the following lemmas, it will also help to define, for a string y and a state $|\phi\rangle$, the value $q_y(\phi)$ to be the sum of the squared magnitudes of the amplitudes of ϕ corresponding to configurations in which y is to be queried in the next step.

Lemma 3.2. *Fix a quantum Turing machine M, an oracle A, and an input x. For any natural number i, let $|\phi_i\rangle$ denote the current superposition for $M^A(x)$ at step i. Let $\epsilon > 0$. Let $F \subseteq [0, T-1] \times \{0,1\}^*$ be a set of time-string pairs such that $\sum_{(i,y)\in F} q_y(\phi_i) \leq \frac{\epsilon^2}{2T}$. (That is to say, we have in mind a certain*

collection of strings which M^A will have relatively little 'need' to consult it's oracle about within the first T steps.) Suppose that we replace A with a new oracle A' which is identical to A except that for any y such that $(i, y) \in F$ for some i, A' consistently gives (possibly) the opposite answer to A. (Even this is too specific. The proof will show that we are allowed to change the oracle answers for any pairs in the set F, regardless of whether this even results in a consistent oracle. This generality will be necessary later, when we run a machine relativized to a slightly different oracle for every step.) Let $|\phi'_i\rangle$ be the superposition of $M^{A'}(x)$ at time i. Then $| \, |\phi_T\rangle - |\phi'_T\rangle \, | \leq \epsilon$.

Proof. This proof will hinge on some specific technicalities of how oracle queries work. In particular, our oracle queries are somewhat deterministic compared to a quantum machine's normal behavior. Enumerate basis configurations by $|1\rangle, |2\rangle$, and so forth. Consider the set

$$J_y = \{|j\rangle : j \text{ indexes a configuration with intent to query } y \text{ on the next step}\}$$

and fix a $|j\rangle$. Normally, a basis state would map to a superposition of combinations of printing characters, moving the cursor, and changing states. However, in the case of $|j\rangle$, we first of all must be in a specific pre-query state q_q, and second of all will always map only to the post-query state q_a, with either a 0 or a 1 printed to the right of the query string on the query tape. The cursor will not move at all (neither on the work tape nor the query tape), and nothing at all will be printed on the work tape. With all of this said, we can say with certainty that the basis state $|j\rangle$ will map to a *single* other basis state, and *not* a linear combination of basis states as would normally be the case. Let U be the unitary time evolution operator corresponding to M^A. For an arbitrary basis state $|j\rangle$, we could only claim that there is an integer N dependent on the number of states of the machine M and a collection of $2N$ indexes of basis configurations $\{m_1^j, m_2^j, ..., m_N^j, m_{N+1}^j, ..., m_{2N}^j\}$ such that the resulting superpositions is a finite sum:

$$U|j\rangle = \sum_{i=1}^{N} u_k^j |m_k^j\rangle$$

However, if $|j\rangle$ is a basis state in in a pre-query state, then we can simply assume that there is another basis configuration $|j'\rangle$ such that

$$U|j\rangle = |j'\rangle$$

Now, we will begin the proof. To 'get to' A', for each $i = 0, ..., T-1$, let A_i be the oracle which only negates answers to strings y such that $(i, y) \in F$. and let U_i be that corresponding to A_i for each i. Consider the difference between $U|j\rangle$ and $U_i|j\rangle$, where $|j\rangle$ is a basis configuration in J. Then via the above discussion, there are basis states a $|j'\rangle$ and $|j_i\rangle$ such that $U|j\rangle = |j'\rangle$ and $U_i|j\rangle = |j_i\rangle$, and so

$$(U - U_i)|j\rangle = |j\rangle - |j_i\rangle$$

Now, consider the magnitude of this vector. These two basis configurations are nearly identical - the only difference will possibly be the character printed on the query string. In that case, this is the difference of two orthogonal unit vectors, and therefore has magnitude $\sqrt{2}$. Otherwise, it is simply the zero vector, and has magnitude 0. Let us also denote the amplitudes of the $|\phi_i\rangle$'s:

$$|\phi_i\rangle = \sum_{j \in \omega} \alpha_j^i |j\rangle$$

and define

$$|E_i\rangle = U|\phi_i\rangle - U_i|\phi_i\rangle$$

With our observations about basis configurations, we can better understand this 'error vector' $|E_i\rangle$. We'll assume the worst case - that everything in F has been altered. Then:

$$|E_i\rangle = \sum_{j \in \omega} \alpha_j^i (U - U_i)|j\rangle \tag{7}$$

$$= \sum_{y:(i,y)\in F} \sum_{j \in J_y} \alpha_j^i (|j\rangle - |j_i\rangle) \tag{8}$$

And so

$$|\,|E_i\rangle\,|^2 = \sum_{y:(i,y)\in F}\sum_{j\in J_y}\sum_{k=1}^{2N}|\alpha_j^i|^2|u_k^j|^2 = \sum_{y:(i,y)\in F}\sum_{j\in J_y}|\alpha_j^i|^2\sum_{k=1}^{2N}|u_k^j|^2 \tag{9}$$

$$= \left[\sum_{y:(i,y)\in F}\sum_{j\in J_y}|\alpha_j^i|^2\right]2 \tag{10}$$

$$= 2\sum_{y:(i,y)\in F} q_y(\phi_i) \tag{11}$$

Thus by our hypothesis we have that the sum of the squared magnitudes of these error vectors is bounded above by $\frac{\epsilon^2}{T}$:

$$\sum_{i=0}^{T-1}|\,|E_i\rangle\,|^2 = 2\sum_{i=0}^{T-1}\sum_{y:(i,y)\in F} q_y(\phi_i) \tag{12}$$

$$= 2\sum_{(i,y)\in F} q_y(\phi_i) \leq 2\frac{\epsilon^2}{2T} \tag{13}$$

Next, for convenience let $U_T = I$ (the identity matrix), and note that by definition of $|E_i\rangle$, we have the following:

$$|\phi_T\rangle = U\,|\phi_{T-1}\rangle = U_{T-1}\,|\phi_{T-1}\rangle - |E_{T-1}\rangle \tag{14}$$

$$= U_{T-1}U\,|\phi_{T-2}\rangle - |E_{T-1}\rangle = U_{T-1}[U_{T-2}\,|\phi_{T-2}\rangle - |E_{T-2}\rangle] - |E_{T-1}\rangle \tag{15}$$

$$= U_{T-1}U_{T-2}\,|\phi_{T-2}\rangle - (U_T\,|E_{T-1}\rangle + U_TU_{T-1}\,|E_{T-2}\rangle) \tag{16}$$

$$= \ldots = U_{T-1}U_{T-2}\ldots U_1U_0\,|\phi_0\rangle - \sum_{i=1}^{T-1}U_T\ldots U_{i+1}\,|E_i\rangle \tag{17}$$

$$= |\phi_T'\rangle - \sum_{i=1}^{T-1}U_T\ldots U_{i+1}\,|E_i\rangle \tag{18}$$

Therefore, *crucially*, since all of these U_i's are unitary

$$|\,|\phi_T\rangle - |\phi_T'\rangle\,|^2 = \left|\sum_{i=1}^{T-1}U_T\ldots U_{i+1}\,|E_i\rangle\right|^2 \tag{19}$$

$$\leq \left[\sum_{i=1}^{T-1}|U_T\ldots U_{i+1}\,|E_i\rangle\,|\right]^2 \tag{20}$$

$$= \left[\sum_{i=1}^{T-1}|\,|E_i\rangle\,|\right]^2 \tag{21}$$

$$\leq T\left[\sum_{i=1}^{T-1}|\,|E_i\rangle\,|^2\right] \tag{22}$$

$$\leq T\frac{\epsilon^2}{T} = \epsilon^2 \tag{23}$$

Where the transition from (22) to (23) is due to the fact that for and set of real numbers $x_1, ..., x_n$, it is always the case due to convexity that $\left(\sum_{i=1}^n x_i\right)^2 \leq n\left(\sum_{i=1}^n x_i^2\right)$. □

The following corollary is fairly specific, it needs to be generalized and very obviously can be to a startling degree. For an arbitrary oracle A, $n \in \omega$, and an arbitrary string y of length n, let A_y be the oracle created by modifying A so that $y10^i$ for $i = 0, ..., n-1$. (Note that this has the function of fixing $\xi_{A_y}(y) = 1^n$ regardless of A.)

Corollary 3.1. *Fix a quantum Turing machine M, an oracle A, an input x, and a number of steps T. Given a y, let $|\phi_i\rangle$ denote the time i superposition of $M^A(x)$, and $|\phi_i\rangle^{(y)}$ the same for $M^{A_y}(x)$. Then for any $\epsilon > 0$. There is a set S of cardinality at most $\frac{2T^2}{\epsilon^2}$ such that for all $y \notin S$, $| |\phi_T\rangle - |\phi_T\rangle^{(y)} | \leq \epsilon$.*

Proof. Since states have unit length, it is clearly the case that

$$\sum_{i=0}^{T-1} \sum_{y \in \{0,1\}^*} q_y(\phi_i) \leq T$$

Let $\epsilon > 0$. Let S be the set of strings y such that

$$\sum_{i=0}^{T-1} \sum_{j=0}^{n-1} q_{y10^j}(\phi_i) \geq \frac{\epsilon^2}{2T}$$

Clearly then $|S| \leq \frac{2T^2}{\epsilon^2}$, since otherwise

$$\sum_{i=0}^{T-1} \sum_{y \in \{0,1\}^*} q_y(\phi_i) \geq \sum_{i=0}^{T-1} \sum_{y \in S} q_y(\phi_i) \tag{24}$$

$$\geq \sum_{y \in S} \frac{\epsilon^2}{2T} \tag{25}$$

$$> \frac{2T^2}{\epsilon^2} \frac{\epsilon^2}{2T} = T \tag{26}$$

which we just observed cannot be the case. By definition then if $y \notin S$, we have that $\sum_{i=0}^{T-1} \sum_{j=0}^{n-1} q_{y10^j}(\phi_i) < \frac{\epsilon^2}{2T}$. But then considering $F = \{0, ..., T-1\} \times \{y, y10, \ldots, y10^{n-1}\}$, this is exactly the criteria we need to apply the previous lemma, from which it follows that $| |\phi_T\rangle - |\phi_T\rangle^{(y)} | \leq \epsilon$. □

Theorem 3.1. *For any randomly drawn oracle A, and any $T(n)$ which is $o(2^{\frac{n}{2}})$, $P(\boldsymbol{NP}^A \subseteq \boldsymbol{BQTime(T(n))}^A) = 1$. In particular, $P(\boldsymbol{BQP}^A \neq \boldsymbol{NP}^A) = 1$.*

Proof. Before getting started on the proof itself it will be beneficial to note some properties of the function ξ_A. For a random oracle A, we must suppose that the probability that any particular bit is a 1 is exactly $\frac{1}{2}$, and that these events are all independent. For an integer n, consider the experiment of drawing a random $n-bit$ string y, and let X_n be the random variable representing the number of strings x such that $\xi_A(x) = y$. Note that $X_n = \sum B_n^x$, for each string x of length n, B_n^x equals 1 if $\xi_A(x) = y$ and 0 otherwise. Now, for any particular x, $\xi_A(x) = y$ iff $A(x1) = y_0$ and $A(x10) = y_1$ and ... and $A(x10^{n-1}) = y_{n-1}$, each of which is a probability $\frac{1}{2}$. Thus $P(\xi_A(x) = y) = (\frac{1}{2})^n$. Thus, considerations of independence aside, X_n is a sum of 2^n identically distributed Bernoulli random variables, and thus at least strongly approximates a Binomial random variable, i.e. X_n $Binomial(2^n, (\frac{1}{2})^n)$. Since the number of trials here is quite high while the probability of success is quite low, it follows that X_n approximates a Poisson distribution with $\lambda = 1$, the mean of X_n. Thus $P(X_n = k) \approx \frac{1}{ek!}$. These approximations get better as the length of the string drawn gets large. Of particular interest for us will be the fact that $P(X_n = 0) = P(X_n = 1) \approx \frac{1}{e}$, which represents the probabilities that a randomly drawn string will have either no inverses images or exactly 1 inverse image under ξ_A, respectively.

Consider the test language $L_A = \{y : \exists x \xi_A(x) = y\}$ (This is exactly the same test language $RANGE_A$ from the original proof). Clearly $L_A \in \boldsymbol{NP}^A$. As before, the test language and the class $BQTime(T(n))^A$ satisfy the oracle conditions discussed before, and so to show that $P(L_A \in \boldsymbol{BQTime(T(n))}) = 0$ it will suffice to show that for any quantum machine M^A in this class, there is a probability of at least $\frac{1}{8}$ that the language determined by the machine M^A, which we will denote $L(M^A)$, does not equal L_A. Thus we fix a machine M^A, and also fix an $n \in \omega$ big enough that $T(n) \leq \frac{2^{\frac{n}{2}}}{20}$ for all $n' \geq n$. Of course if M^A's output on 1^n disagrees with L_A, then $L(M^A) \neq L_A$.

(Maybe move to later) To simplify the problem further, we will fix a set of oracles $\mathcal{C}$ such that for any $A, B \in \mathcal{C}$, if $x \in \{0,1\}^*$ and $|x| \neq n$, $\xi_A(x) = \xi_B(x)$.

Consider the following two events:

$$\mathcal{A} = \{A \in 2^\omega : 1^n \text{ has no inverse image under } \xi_A\}$$

$$\mathcal{B} = \{A \in 2^\omega : 1^n \text{ has exactly one inverse image under } \xi_A\}$$

Note that by the earlier discussion about ξ_A, we have that

$$P(\mathcal{A}) = P(X_n = 0) = P(X_n = 1) = P(\mathcal{B}) \approx \frac{1}{e} \geq \frac{1}{4}$$

It is also clear that $\mathcal{A}$ and $\mathcal{B}$ are mutually exclusive. Moreover, conditioning on either of these events makes explicit what the machine M^A must return on the input 1^n. In order to be correct, $M^A(1^n)$ must output a 1 in the case of event $\mathcal{B}$ and a 0 in the case of $\mathcal{A}$ (by *output* we really mean that observing the output string after $T(n)$ steps returns the appropriate outputs at least $\frac{2}{3}$ of the time. If this is the case we will simply write $M^A(1^n) = 1$.) Putting these observations together and applying the multiplication rule gives us the following:

$$\begin{aligned} P(L(M_A) \neq L_A) &\geq P(A \in \mathcal{B} \wedge M^A(1^n) = 0) + P(A \in \mathcal{A} \wedge M^A(1^n) = 1) && (27)\\ &= P(M^A(1^n) = 0|\mathcal{B})P(\mathcal{B}) + P(M^A(1^n) = 1|\mathcal{A})P(\mathcal{A}) && (28)\\ &\geq \frac{1}{4}[P(M^A(1^n) = 0|\mathcal{B}) + P(M^A(1^n) = 1|\mathcal{A})] && (29)\\ &= \frac{1 + \alpha_{\mathcal{A}} - \alpha_{\mathcal{B}}}{4} && (30) \end{aligned}$$

Where $\alpha_{\mathcal{A}} = P(M^A(1^n) = 1|\mathcal{A})$ and $\alpha_{\mathcal{B}} = P(M^A(1^n) = 1|\mathcal{B})$ (so the $\mathcal{B}$ term is really $1 - \alpha_{\mathcal{B}}$, hence where the 1 comes from). What we need to show is now clear: that $\alpha_{\mathcal{A}}$ is comparable enough in size to $\alpha_{\mathcal{B}}$ that this final number remains a solid bound for all machines M^A.

Towards this end we make use of our lemma's from earlier. Recall the small oracle transformation $A \mapsto A_y$ given a string y. Setting $\epsilon = \frac{1}{13}$, by corollary 3.1 there exists a set of strings S of cardinality at most $338T^2(n)$ such that for all $y \notin S$, $|\,|\phi_T\rangle - |\phi_T\rangle^{(y)}\,| \leq \frac{1}{13}$. How many strings y are there which are not in S? Since $T(n) \leq \frac{2^{\frac{n}{2}}}{20}$, and there are 2^n strings of length n in total, we have that there are at least

$$2^n - \frac{338}{400}2^n = \frac{62}{400}2^n \geq 2^{n-3}$$

Which is a bound we will need soon. For now though observe that in conjunction with lemma 3.1, if $y \notin S$, $|P(M^A(1^n) = 0) - P(M^A(1^n) = 0)| \leq \frac{4}{13} < \frac{1}{3}$. Suppose $A \in \mathcal{A}$, and the machine M^A successfully accepts the language L_A. Then $P(M^A(1^n) = 0) \geq \frac{2}{3}$, and so what we have from this lemma is that $P(M^{A_y}(1^n) = 0) > \frac{1}{3}$, i.e. that $P(M^{A_y}(1^n) = 1) < \frac{2}{3}$, meaning that $\boldsymbol{M^{A_y}}$ **cannot possible accept** $\boldsymbol{L_{A_y}}$.

Fix a set of values for xi_A on all inputs not of length n. Now, on this fixed slice, for any oracle A such that M^A succeeds (that is to say, $M^A(1^n) = 1)$), there are at least 2^{n-3} strings y such that M^{A_y} fails, with A_y in B. Furthermore these A_y are clearly unique. Thus for any $A \in \mathcal{A}$ on which M^A succeeds, there are at least 2^{n-3} oracles in $\mathcal{B}$ such that M^{A_y} fails. Furthermore, there are $2^n - 1$ total unique mappings $A \mapsto A_y$ to choose from. (One for whatever string y we are choosing for which to make $\xi_{A_y}(y) = 1^n$.) Thus, if we were to consider the experiment of picking a random A from those oracles in $\mathcal{A}$ on which M^A succeeds and the values of xi_A are fixed for strings not of length n, and then picking a random $y \neq 1^n$, there is at least a $\frac{2^{n-3}}{2^n-1} \geq \frac{1}{8}$ chance that M^{A_y} fails. And since this will be constant for fixings of values of ξ_A not of length n, we can assume that this is a bound on the probability without that caveat.

Picking a random oracle from $\mathcal{B}$ is equivalent as a probability experiment to picking a random oracle from $\mathcal{A}$ and a random string $y \neq 1^n$, and considering A_y. Let Y_A be the set of strings y such that M^{A_y} fails. Thus

$$\begin{aligned} P(M^{A_y}(1^n) = 0|A_y \in \mathcal{B}) &= P(M^A(1^n) = 0 \wedge y \in Y_A|A \in \mathcal{A}) && (31)\\ &\geq \frac{1}{8}P(M^A(1^n) = 0|A \in \mathcal{A}) && (32) \end{aligned}$$

Thus $1 - \alpha_{\mathcal{B}} \geq \frac{1}{8}(1 - \alpha_{\mathcal{A}})$ □

Next we would like to find a set of oracles separating $\boldsymbol{BQP}$ from not just $\boldsymbol{NP}$, but $\boldsymbol{NP} \cap \boldsymbol{coNP}$. To do this, we need to restrict the set of oracles that we are drawing from at random to a particular set of **permutation oracles**. Previously we had chosen oracles and defined length preserving functions in terms of them. The reverse construction is also possible. Consider a standard encoding method which takes binary strings to pairs of binary strings $x \mapsto (y, i)$, which is an efficiently computable bijection, so that strings can be immediately interpreted as pairs of strings. Given a length preserving function f from strings to strings, an oracle A_f can be defined by having $A(x) = 1$ iff the i^{th} bit of y is a 1 (where again $x \mapsto (y, i)$, and we are interpreting i as an integer.) The set of permutation oracles can be thought of as the set of oracles A_f defined in terms of functions as above, such that the function f itself is a permutation on strings of length n, for all n. (What is the measure of this set?)

Theorem 3.2. *Consider a permutation oracle A, drawn at random. Then with probability 1, and let $T(n) \in o(2^{\frac{n}{3}})$. Then with probability 1, $\boldsymbol{BQTime}^A(T(n))$ does not contain $\boldsymbol{NP}^A \cap \boldsymbol{coNP}^A$. In particular $\boldsymbol{BQP}$ does not contain $\boldsymbol{NP}^A \cap \boldsymbol{coNP}^A$.*

Proof. Note first that given a permutation oracle A, we can always assume that there is an underlying function f. Pick n large enough that $T(n) \leq \frac{2^{\frac{n}{3}}}{100}$. We will show, like the previous proofs, that for an arbitrary machine M^A running in time at most $T(n)$, there is probability at least $\frac{1}{8}$ that the machine does not accept the language L_A, where

$$L_A = \{y : \text{first bit of } f^{-1}(y) \text{ is } 1\}$$

(Note that this test language would even not be well defined outside the context of permutation oracles. Substituting f for ξ_A would not work, since ξ_A is not necessarily a permutation.) Our proof will require an expanded probability space. Rather than simply drawing a random oracle from the set of permutation oracles, we are going to instead do the following: first, draw T strings of length n uniformly at random $x_0, x_1, ..., x_T$. Then, draw a permutation π_0 uniformly at random from the set of permutation functions such that $\pi_0(x_0) = 1^n$. Then define $T - 1$ more permutations $\pi_1, \pi_2, ..., \pi_T$ by $\pi_i = \pi_{i-1} \circ \tau$, where τ is the transposition (x_{i-1}, x_i), i.e. for say, π_1, we are letting $\pi_1(y) = \pi_0(y)$ for all strings other than x_0 and x_1, and letting $\pi_1(x_0) = \pi_0(x_1)$, and $\pi_1(x_1) = \pi_0(x_0) = 1^n$. Thus these T permutations are defined deterministically given the randomly drawn first permutation, are quite similar to one another, with the only difference for each being that two of the outputs are swapped, and with the 1^n output in particular sliding down so that $\pi_i(x_i) = 1^n$ for each i. Finally, permutation oracles $A_0, A_1, ..., A_T$ are created deterministically to correspond to these permutations. Presuming a fixed initial definition of the permutation outputs for strings not of length n, we have a finite and fairly large, albeit very messily sized probability space. Notably it can easily be confirmed that the final oracle, A_T, on it's own has the same probability distribution as one would obtain from selecting uniformly at random an arbitrary permutation oracle.

Also notable is that the final *two* oracles A_T and A_{T-1} on their own have the same probability distribution which one would obtain by selecting a random permutation oracle A (where, say, x is the string such that the underlying permutation function $f(x) = 1^n$), and random length n string $y \neq x$ and defining the permutation oracle A' by the composition of f and the transposition (x, y). (I.e. for the oracle A' with underlying permutation g, $g(x) = f(y)$, and $g(y) = 1^n$.) By way of contradiction, let us assume that the probability of the corresponding machine the machine $M^{A_T}(1^n)$ outputting the correct answer (i.e. if the first bit of $g^{-1}(1^n)$ is a 1, then $M^{A_T}(1^n)$ has at least a $\frac{2}{3}$ chance of outputting a 1, etcetera) is at least $\frac{7}{8}$. Here our assumption only applies to the experiment of drawing a random permutation oracle. To consider the consequence of this assumption for the machine $M^{A_{T-1}}$, we must see this drawing as a different probability experiment on the product space of random oracles and random transpositions τ of the sort discussed above. In this context, our assumption effectively says that the conditional probability of $M^{A_{T-1}}(1^n)$ being correct given a particular choice of τ is always $\frac{7}{8}$, for any such transposition. But of course it follows then that the probability of the machine $M^{A_{T-1}}(1^n)$ is correct is itself also at least $\frac{7}{8}$. Thus, assuming one machine has a $\frac{7}{8}$ chance of being correct necessitates that the other machine also has at least $\frac{7}{8}$ chance of being correct. It follows that the probability of *both* machines being correct on 1^n is at least $\frac{7}{8} + \frac{7}{8} - 1 = \frac{3}{4}$.

Next consider the probability of 1^n being in the languages L_{A_T} and $L_{A_{T-1}}$. Recall that the strings x_{T-1} and x_T were chosen uniformly at random, that these strings are the inverse images of 1^n for the permutations implicit to the oracles A_{T-1} and A_T respectively. Thus the question of whether the first bit of

the inverse image of 1^n is in both cases the question of the output of a simply coin flip, and these two flips are independent. Thus the probability that 1^n is in exactly one of the two languages $L_{A_{T-1}}$ and L_{A_T} is $\frac{1}{2}$. Therefore the probability that both machines M^{A_T} and $M^{A_{T-1}}$ are correct on 1^n *and* 1^n is in exactly one of the languages is at least $\frac{3}{4} + \frac{1}{2} - 1 = \frac{1}{4}$. But for these machines to be correct if 1^n is in one language but not the other, they must give *different* answers upon final measurement with probability at least $\frac{2}{3}$. What we will show, in order to arrive at a contradiction, is that the close relationship between the oracles A_T and A_{T-1} necessitates that the probability distributions of the final superpositions are themselves close, and that from this it becomes impossible that there is such a high probability of both machines outputting different answers.

Let $|\phi_i\rangle$ denote the time i superposition of $M^{A_T}(1^n)$, and $|\phi'_i\rangle$ the time i superposition of $M^{A_{T-1}}(1^n)$. We will show that $E\left[\,|\,|\phi_T\rangle - |\phi'_{T-1}\rangle\,|\,\right] \leq \frac{1}{50}$. Recall Markov's inequality: given any $a > 0$, and any random variable X,

$$P(X \leq a) \geq 1 - \frac{E(X)}{a}$$

Therefore, setting $a = \frac{2}{25}$, this bound on the expected value gives that

$$P(|\,|\phi_T\rangle - |\phi'_{T-1}\rangle\,| \leq \frac{2}{25}) \geq 1 - \frac{25}{100} = \frac{3}{4}$$

Furthermore if it is the case that $|\,|\phi_T\rangle - |\phi'_{T-1}\rangle\,| \leq \frac{2}{25}$, then the acceptance probability of M^{A_T} and $M^{A_{T-1}}$ differ by at most $\frac{8}{25} < \frac{1}{3}$. Suppose in this case that M^{A_T} accepts 1^n. Then the probability that we measure a 1 is at least $\frac{2}{3}$. In this case of the superpositions being this close then, it is impossible for the other machine to reject, since it's probability of acceptance must be greater than $\frac{1}{3}$, meaning that the probability of rejection is less than $\frac{2}{3}$. Likewise if the superpositions are this close, and M^{A_T} rejects 1^n, it is impossible for $M^{A_{T-1}}$ to accept, for the same reason. Thus, in order for both machines to give different answers, we must have the distance between superpositions greater than $\frac{2}{25}$, which we say occurs with probability less than $\frac{1}{4}$. Assuming we have shown the claim about the expected value, we then have the following:

(1) The probability that both machines are correct and give different answers is at least $\frac{1}{4}$.

(2) The probability that both machines are capable of giving different answers (let alone are correct) is less than $\frac{1}{4}$.

Since the latter event is obviously a prerequisite for the former, it must be a superset, and have a higher probability, but this is the exact opposite of what we are left with, and we therefore have our contradiction. We can conclude then that the probability of the machine M^{A_T} getting the correct answer on 1^n is less than $\frac{7}{8}$, i.e. the probability of it being incorrect on this input, and thus fails to decide the language L_A, is at least $\frac{1}{8}$, completing the proof. All that remains then is to prove the inequality for our expected value.

To accomplish this, we will make use of the many other intermediary oracles that we discussed at the beginning. We will show that both superpositions $|\phi_T\rangle$ and $|\phi'_T\rangle$ are close to another superposition, $|\psi_T\rangle$. This superposition is created by running M on input 1^n with a different oracle at each step. At step i, run M relativized to A_i, and so forth. At each step we have a superposition then $|\psi_i\rangle$. Now, if we recall how these A_i's were defined, we'll note oracle A_i can only possibly differ from oracle A_j for $j \geq i$ on the strings $x_i, x_{i+1}, ...$ Thus consider the set of time string pairs $S = \{(i, x_j) : j \geq i, 0 \leq i \leq T\}$. This set essentially populates an upper diagonal of a square matrix of size $T(n)$, i.e. $|S| \leq T(n)^2$, and clearly by the way these oracles are defined, the machines M^{A_i} and M^{A_j} can only differ on the set S, for any $i \leq j \leq T(n)$.

Now, consider the query magnitude $q_{x_j}(|\psi_i\rangle)$ corresponding to one of these pairs. This is a random variable, dependent on the randomly chosen x_j (indeed upon all of the x_i randomly chosen prior to x_j as well as on the random permutation.) We need an upper bound for the expected value of this variable. To obtain this number, we would need to sum over the range of $q_{x_j}(|\psi_i\rangle)$, the values $q_{x_j}(|\psi_i\rangle)$ (all of which are obviously less than 1), and for each of those possible values, sum the probabilities for each outcome resulting in that query magnitude. Recall what we're actually doing probabilistically: drawing uniformally at random $T(n)$ different strings of length n, and also drawing a random permutation on strings of length n. Now, it might seem that the set of combinations of these outcomes that result in a particular query magnitude $q_{x_j}(|\psi_i\rangle)$ is extremely intricately determined, to the point where there is no simple justification for why the probability should generally be bounded below $\frac{1}{2^n}$. However, the uniformity of the situation

makes things very simple. Whatever else is going on, we can take the situation at face value - we can envision the superposition $|\phi\rangle$ as fixed, look through all possible query magnitudes for that superposition, and for each such magnitude consider the probability that the x_j is that particular string x which yields the query magnitude we're talking about, which is always just $\frac{1}{2^n}$. Since the sum of all such query magnitudes must be less than 1 (since we have the superposition fixed), and thus the expected query magnitude must be less than $\frac{1}{2^n}$. Assuming this is the case, then if α is the sum of all query magnitudes of pairs in S, we would then have that $E(\alpha) \leq \frac{T(n)^2}{2^n}$. Define $\epsilon = \sqrt{2T(n)\alpha}$, so that $\alpha = \frac{\epsilon^2}{2T(n)}$. Then the sum of the query magnitudes of pairs in S is by definition less than or equal to $\frac{\epsilon^2}{2^n}$ (since it is literally defined to be equal to this), meaning that lemma 3.2 applies, and we can confidently assume that regardless of what ϵ turns out to be, $|\,|\psi_{T(n)}\rangle - |\phi_{T(n)}\rangle\,| < \epsilon$, and $|\,|\psi_{T(n)}\rangle - |\phi'_{T(n)}\rangle\,| < \epsilon$, and so of course the expected values of both of these as random variables must be less then $E(\epsilon)$. Moreover by the triangle inequality, the random variable $|\,|\phi_T\rangle - |\phi_T\rangle\,| \leq 2\epsilon$, and so it's expected value is less than 2 times the expected value of ϵ. The proof can now finally conclude: we have first of all

$$E\left(\frac{\epsilon^2}{2T(n)}\right) \leq \frac{T(n)^2}{2^n} \tag{33}$$

$$\implies E(\epsilon^2) \leq \frac{T(n)^3}{2^{n-1}} \tag{34}$$

But $E(X)^2 \leq E(X^2)$ for any random variable X, and $T(n) \leq \frac{2^{\frac{n}{3}}}{100}$, so we have

$$E(\epsilon) \leq \sqrt{E(\epsilon^2)} \leq \frac{\sqrt{\left(\frac{2^{\frac{n}{3}}}{100}\right)^3}}{2^{\frac{n-1}{2}}} \tag{35}$$

$$= \frac{\sqrt{2}}{1000} \tag{36}$$

$$\leq \frac{1}{100} \tag{37}$$

Thus we have that $E(|\,|\phi_T\rangle - |\phi_T\rangle\,|) \leq \frac{1}{50}$, completing the proof. □

That's the proof proposed by the original paper, but it seems overly complicated, to a point where simplifying it might yield stronger results.

Claim 3.1. *For any $T(n)$ which is $o(2^{\frac{n}{3}})$, the set of oracles A such that $\boldsymbol{NP}^A \cap \boldsymbol{coNP}^A \not\subseteq \boldsymbol{BQTime(T(n))}$ is at least Lebesgue measure $\frac{1}{e}$.*

Proof. We first want to eliminate any business existing which mentions "permutation oracles". To do this, we return to our function ξ_A, defined exactly as earlier, and consider the set of oracles:

$$C = \{A : |\xi_A^{-1}(1^n)| = 1\}$$

Where n has been fixed such that $T(n) \leq 2\frac{2^{\frac{n}{2}-1}}{50\sqrt{n}}$. That is, the set of oracles such that 1^n has exactly one inverse image under xi_A. Recall that this set has Lebesgue measure at least $\frac{1}{e}$. We will draw oracles exclusively from this set, and slightly redefine the language L_A so that it makes sense in this new context.

$$L_A = \{y : \xi_A^{-1}(y) \neq \varnothing \wedge \text{first bit of the least } x \text{ such that } \xi_A(x) = y \text{ is 1}\}$$

Clearly such an oracle is well defined for any oracle, permutation or otherwise, but carries a special promise in the case of drawing from our set C. Our experiment will be significantly simpler from before. Simply draw an oracle from C. Then there is guaranteed an x such that $\xi_A(x) = 1^n$, and the probability that the first bit of this x is a 1 is exactly $\frac{1}{2}$. Following this, draw a random $y \neq x$, revise A to create an A' such that $\xi_{A'}^{-1}(1^n) = y$ rather than x, ie $\xi_{A'}(y) = 1^n$, and $\xi_{A'}(x) = \xi_A(y)$. This can be done without too much trouble,

in an algorithmic way: First, to fix $\xi_{A'}(y) = 1^n$, we simply add all strings of the form $y1$, $y10$,...,$y10^{n-1}$ which weren't in A before. To make

$$\xi_{A'}(x) = \xi_A(y) = A(y1)A(y10)\dots A(y10^{n-1})$$

we clearly need to satisfy $A(x1) = A(y1)$, $A(x10) = A(y10)$, and so forth. Thus, for $0 \leq k \leq n-1$, if $x10^k \in A$ but $y10^k \notin A$, then we remove $x10^k$ from A, and if $x10^k \notin A$ but $y10^k \in A$, then we add $x10^k$ to A. Altogether, this make at most $2n$ changes to A. Let D denote the set of strings on which A and A' differ - we'll need it later.

Like before, we will show that if upon drawing a random oracle A from C, an arbitrary quantum machine $M^A \in \boldsymbol{BQTime^A(T(n))}$ has probability at least $\frac{1}{8}$ of not accepting the language L_A, by virtue of high probability of being wrong on the particular input 1^n. As before, we will assume by way of contradiction that the machine has at least a $\frac{7}{8}$ chance of outputting the correct answer on 1^n (whatever that may be). Note from this assumption, it follows that the machine $M^{A'}$ has at least a $\frac{7}{8}$ chance of being correct as well: since 'drawing' the oracle A' in the secondhand way we are doing it is identical to drawing the oracle A' directly as we did A. **(The only real question I have about this is what guarantees the machine $M^{A'} \in \boldsymbol{BQTime^{A'}(T(n))}$?)** As before, my worry about both proofs aside, it follows then that the probability of both machines being correct on 1^n is at least $\frac{3}{4}$. Also just as before, since the strings x and y are chosen uniformally at random, the probability of 1^n being in exactly one of the languages L_A is $\frac{1}{2}$, and so the probability that both machines are correct on 1^n and that 1^n is in only one of the languages is at least $\frac{1}{4}$.

Also just as before, we define $|\phi\rangle$ and $|\phi'\rangle$ denote the time $T(n)$ superpositions of $M^A(1^n)$ and $M^{A'}(1^n)$ respectively. We will show that $E(|\,|\phi\rangle - |\phi'\rangle\,|) \leq \frac{1}{50}$, and by a word for word repetition of the paragraph explaining it in the previous proof, this will contradict the conclusion we arrived at in the previous paragraph, concluding the proof. Consider the set of time string pairs $F = T(n) \times D$, and note that $|F| \leq 2nT(n)$. Just as before, the expected query magnitude of any time string pair in F is at most $\frac{1}{2^n}$ for the same reason as before, and so the expected total query magnitude of all pairs in F, call it α, is at most $\frac{2nT(n)}{2^n} = \frac{nT(n)}{2^{n-1}}$. As before, if we define $\epsilon = \sqrt{2T(n)\alpha}$ so that $\alpha = \frac{\epsilon^2}{2T(n)}$, then the sum of the query magnitudes is by definition less than $\frac{epsilon^2}{2^n}$, and so by 3.2 $|\,|\phi\rangle - |\phi'\rangle\,| \leq \epsilon$. (I also really don't see the point of singling out a state in between, not just comparing them directly. Then

$$E\left(\frac{\epsilon^2}{2T(n)}\right) \leq \frac{nT(n)}{2^{n-1}} \tag{38}$$

$$\implies E(\epsilon^2) \leq \frac{nT(n)^2}{2^{n-2}} \tag{39}$$

But then

$$E(\epsilon) \leq \sqrt{E(\epsilon^2)} \leq \sqrt{\frac{nT(n)^2}{2^{n-2}}} \tag{40}$$

$$\leq \frac{2\sqrt{n}T(n)}{2^{n-2}} \tag{41}$$

$$\leq \frac{1}{50} \tag{42}$$

Thus the proof is complete. I really don't see what's wrong with this. □

More generally I also don't see what is wrong with replacing the $2^{\frac{n}{3}}$ bound with $2^{\frac{n}{2}}$. Just like the proof I just gave, I cannot see a single problem with drawing a single permutation oracle and a single string to transpose with 1^n, in which the words of the proof I gave could be repeated. In this case it's even simpler, the difference between oracles in a single string, and the set of time-string discrepancy pairs is just criminality $T(n)$, not $T(n)^2$.

Finally, I want to consider the implications of this proof for the deterministic time classes. Since a deterministic Turing machine is a special case of a quantum Turing machine, if $\boldsymbol{BQTime(2^{\frac{n}{2}})}$ (need to do)

Theorem 3.3. *Blah blah*

Proof. Let $f(n)$ be a sufficiently fast growing function, tentatively such that $f(n+1) > f(n)^{\log(f(n))}$ for all $n \in \omega$. Fix an enumeration over all quantum Turing machines. We construct the oracle A in steps. At step n, run the n^{th} machine, call it M, relativized to the partially constructed oracle, on the input $1^{f(n)}$ for $f(n)^{\log(f(n))}$ many steps. Suppose the machine rejects the input. That is to say $P(M(1^{f(n)}) = 0) \geq \frac{2}{3}$. In this case, we would like to add a string x of length $f(n)$, without altering the machine so as to accept $1^{f(n)}$, and also without altering the behavior of any prior machine. The latter concern is a nonissue, since no prior machine could possible run for enough steps to query a string of length $f(n)$. Pick an x of length $f(n)$ at random. We would like to show that upon adding this string to the oracle A, and running the same machine M relativized to this new oracle, will not be significantly different than before. Let $|\phi_i\rangle$ denote the time i superposition of the machine relativized to the oracle without the string x, and $|\phi_i'\rangle$ the time i superposition of the machine relativized to the oracle which does include the string x. In particular, we would like to show that

$$\sum_{i=1}^{f(n)^{\log(f(n))}} q_x(|\phi_i'\rangle) \leq \frac{1}{288 f(n)^{log(f(n))}}$$

From here on out let $T = f(n)^{log(f(n))}$. If this were the case, then it would follow that $\| |\phi_T\rangle - |\phi_T'\rangle \| < \frac{1}{12}$, and from this it would follow that the probability of the machine relativized to the new oracle, when run on the input 1^n, has no less than a $\frac{1}{3}$ chance of rejection, meaning that it cannot accept. Suppose for now that we have shown this to be the case. Then on the condition of simply not adding anything to A if the machine either accepts or does not halt, we have spelled out the construction of an oracle A. Let's inspect this oracle. Consider the language

$$\{L_A = \{1^n : \text{there exists an } x \text{ in } A \text{ of length } n\}$$

Clearly, this problem is in $\boldsymbol{NP}^A$. Is it in $\boldsymbol{BQP}^A$? Suppose it were. Then we have a machine M^A which decides L_A. During the construction of A, the machine M would certainly have been encountered at some step, say step i, and suppose that the machine halts in time exactly n^k. Consider the behavior of $M^A(1^{f(i)})$. For starters, we are presuming the enumeration of the machines is such that each machine is encountered infinitely often. Thus, we can safely assume i is such that $f(i)^{\log(f(i))} > f(i)^k$. Since M^A decides L_A, we know that it must halt in either acceptance or rejection prior to $f(i)^{\log(f(i))}$ many steps. If it accepts, then we can be sure that the same machine accepted during the construction of A, since we didn't add anything to it and change the oracle in any way that would affect the computation. But precisely because of this, it cannot be the case that $1^{f(i)} \in L_A$, and so the machine is making an error. Thus it must be the case that the machine rejects the input. Suppose the machine accepted or failed to halt on the input during the construction. Then we wouldn't have added a string, and so the machine must continue to accept, which we just said wasn't the case. So it must be the case that the machine rejected the input during the construction of A as well. *(Need to read the convention for quantum machines halting again, confusing myself a bit here.)* So we have that during the construction the machine M relativized to the oracle without any strings of length $f(i)$, halted. Suppose it halted with neither a $\frac{2}{3}$ chance or acceptance nor rejection. Then we would not have added a string of length $f(i)$ to A, and so the machine M would be doing the same thing, which is impossible since we said it had to reject. So during the construction it had to have halted and furthermore it had to have halted with either a $\frac{2}{3}$ chance of acceptance or a $\frac{2}{3}$ chance or rejection. Suppose it were rejection. Then we would have added a string of length $f(i)$, meaning that $1^{f(i)} \in L_A$, and the machine would be making an error. The only way for the machine to have not made an error, then, is for the original machine, *prior* to adding a string of length $1^{f(i)}$, halted in acceptance... but then we wouldn't have added a string. So this is only going to be even slightly an issue in the case of trying to construct a homogeneous set. I was going to say that the machine halted in acceptance, we added a string x, and this turned it over to halting in rejection (which would have been impossible).

Furthermore it has to be rejection, since if it were acceptance, then we would not have added a string, the machine would be no different in the current situation, and we would have the current machine accepting too. Thus in the construction, when relativized to only the partial oracle which had not yet potentially added any strings of length $f(i)$ to A, the machine had halted and rejected. Thus, we added a string of length $f(i)$. □

Theorem 3.4. *There exists a homogeneous set for the collection of oracles C relative to which $\boldsymbol{NP}^C \nsubseteq \boldsymbol{BQP}^C$.*

Proof. The strategy before was to create a collection A in which exactly one string of every length in the range of $f(n)$, which we could sure "always worked". Let $f(n)$ be a sufficiently fast growing function, tentatively such that $f(n+1) > f(n)^{\log(f(n))}$ for all $n \in \omega$. Fix an enumeration over all quantum Turing machines. We construct the oracle A in steps. At step n, run all $m < n$ machines, relativized to A' for all $A' \subseteq A$, where A is the partially constructed oracle, on the input $1^{f(n)}$ for $f(n)^{\log(f(n))}$ many steps. Suppose that one of these machines halts. Before, in the context of $\boldsymbol{NP}$ machines, we wanted to add a string x which was never encountered along at least one accepting path of any of the machine/oracle subset pairs, among those machines that had accepting paths. If a machine ever accepts, we don't want the string we have available to add to change that acceptance into a rejection. In the context of quantum machines, we have the same goal. At step n, suppose we have a machine M_i and a subset of the partially constructed oracle A_j, $i < n, j < 2^n$. Suppose that for one of the machines M and one of the partial subsets $A' \subseteq A$, we have that $P(M^{A'}(1^{f(n)}) = 1) \geq \frac{2}{3}$, and that the machine halts within $f(n)^{\log(f(n))}$ steps.

Define the set $S_{M,A'}$ to be the set of strings y such that

$$\sum_{i=0}^{T-1} q_y(\phi_i) \geq \frac{1}{338T}$$

Suppose it were the case that $|S_{M,A'}| > 338T^2$. Then we would have

$$T \geq \sum_{i=0}^{T-1} \sum_{y \in \{0,1\}^*} q_y(\phi_i) \geq \sum_{i=0}^{T-1} \sum_{y \in S_{M,A'}} q_y(\phi_i) \tag{43}$$

$$\geq \sum_{y \in S_{M,A'}} \frac{1}{338T} \tag{44}$$

$$> 338T^2 \frac{1}{338T} = T \tag{45}$$

Thus it must be the case that $|S_{M,A'}| \leq 338T^2$. Thus, if $y \notin S_{M,A'}$, we have that $\sum_{i=0}^{T-1} q_y(\phi_i) < \frac{1}{338T}$. But then considering $F = \{0, ..., T-1\} \times \{y\}$, and with $\epsilon = \frac{1}{13}$, we have that $\frac{\epsilon^2}{2T} = \frac{1}{338T}$, and so by lemma 3.2, if $|\phi_T\rangle^{(y)}$ is the time T superposition of the machine M on input $1^{f(n)}$ and relativized to $A' \cup \{y\}$, then $\| |\phi\rangle_T - |\phi\rangle_T^{(y)} \| \leq \frac{1}{13}$. For any string y in the complement of $S_{M,A'}$, then, we have that if y is added to the oracle, it will not cause enough changes to make the machine reverse it's decision, since the probabilities of acceptance or rejection must be within $\frac{4}{13} < \frac{1}{3}$ of each other. We wish to add such a string, but have it specifically be a string of length $f(n)$. In the worst case, all strings in the set $S_{M,A'}$ will be of this length.

At stage n of the construction of A, we have such a set $S_{M,A'}$, as described above, for all machines M_i for $i < n$, and all subsets A' of the partially constructed A, of which there are 2^n many. Consider the union of all of these sets. This itself is a set of size at most $n2^n 338T^2 = n2^n 338 f(n)^{2\log(f(n))}$, which provided $f(n)$ grows sufficiently fast is easily much smaller than $2^{f(n)}$, the size of the set of all strings of length $f(n)$. Thus the intersection of the complements of all of these sets, itself intersected with the set of strings of length $f(n)$, will be nonempty, assuring the existence of a string x that we can add at each step which will not flip any of the decisions made by any Turing machines simulated up to that step of the construction, regardless of whatever subset of the partially constructed oracle that the machines are relativized to. This x we add to A. This describes the construction of A. It will have a single string of length $f(n)$, for every $n \in \omega$.

Now, let $B \subseteq A$. We will construct a $C \subseteq B$ such that the language

$$L_C = \{1^n : \text{there exists an } x \text{ in } C \text{ of length } n\}$$

is in $\boldsymbol{NP}^C$ (which is trivially clear) but not in $\boldsymbol{BQP}^C$. Again we construct C in steps. Let $g(n)$ be the length of the n^{th} element of B. At step n, we run the n^{th} quantum machine relativized to the partially constructed C for $g(n)^{\log(g(n))}$ many steps. Suppose we find a rejecting machine. By construction, we have exactly one string of length $g(n)$ available to possible add to C. Further there exists a $k \geq n \in \omega$ such that

$g(n) = f(k)$. Thus at stage k of the construction of A, we ran the machine M_n for a number of steps greater than or equal to the number of steps we are running it for now, in this construction. Furthermore we ran it relativized to every partial subset of A, one of which would have the be our own partially constructed C. At this point we were sure that the partial oracle C with the x of length $g(n) = f(k)$ added would not change the machine output to a degree necessary for it to be an accepting machine (although it might no longer reject).

Suppose $L_C \in \boldsymbol{BQP}^C$. Then there exists a polynomial time quantum machine which decides it, say in time n^k. Furthermore this machine must be M_i for some i in our indexing, and we can assume that we've chosen an index i such that $g(i)^k < g(i)^{\log(g(i))}$. Consider the machine's behavior on the input $1^{g(i)}$, ie $M^C(1^{g(i)})$. Since M^C decides L_C, it must halt within n^k steps with a two thirds chance of observing either a 0 or a 1. Suppose it is a 1. Consider step i of the construction of C, in which the machine M_i is run relativized to the subset $C' \subseteq C$ which consists of all of the strings of lengths less than $g(i)^{\log(g(i))}$ that are in C. Let x denote the string of length $g(i)$ that we added at this stage of the construction. Suppose that during the simulation of M_i, the machine halted in acceptance just as it does now, or did not halt at all. In those cases, we did not add any strings to C from B, and thus the machine is making an error. Thus is must have halted in rejection. But this too creates a contradiction when we think back to the construction of A. During *that* construction, we simulated, among others, the machine M_i on the input $1^{g(i)}$, relativized to every partial subset of the master set up to that point. We should be clear about what is meant by "that point". As before there exists a $j \geq i$ such that $f(j) = g(i)$. Specifically at that stage of the construction of A, we ran the machine M_i (among many others) on the input $1^{f(j)} = 1^{g(i)}$, relativized to every subset of strings of from the finished A minus strings of length greater than or equal to $f(j)$. One of these subsets must correspond to the oracle C minus strings of length greater than or equal to $g(i)$. Since the parameters are identical, the machine during the construction of A must have also halted in rejection. But consider the x added to the oracle at that point. Clearly this must be the same x which was added to C, and we know that it was added from B since we are assuming that M_i^C decides L_C, and halted in acceptance. But when we originally added x we observed that this would not be sufficient to flip a rejection into acceptance. Thus no matter how we assume the machine M_i behaved during the construction of C, we have no way of reconciling that with an accepting output on the string $1^{g(i)}$.

Thus we are led to assume that the machine M_i outputs a 0, meaning that there is no x in C of length $g(i)$. Again, we consider the cases of what M_i was doing during the construction of C. Suppose first that it halted and accepted. This is clearly impossible, since if this had been the case, we of course would not have added a string to C of length $g(i)$, which we just said cannot happen. Suppose that it rejected. Then of course we just as immediately have a contradiction, since this means that we added a string of length $g(i)$ to C, meaning that M^i is in error. Our final option is that the machine simply did not halt at all. But again, immediately this means that our machine is in error, since in this case we would not have added a string of length $g(i)$, meaning that M_i^C would have identical behavior, and we just said that it halted within $g(i)^k < g(i)^{\log(g(i))}$ steps. If it didn't halt in $g(i)^{\log(g(i))}$ many steps when relativized to a partial oracle consisting of every string that can possibly be queried within that many steps that is also in C, then it certainly cannot now be halting within $g(i)^k$ steps. Thus it is impossible to reconcile the construction with an output of 0 either. We conclude that it cannot possibly be the case that M_i actually decides L_C. Thus it cannot be the case that $L_C \in \boldsymbol{BQP}^C$. □

Need to pursue the opposite direction. Could also pursue separating other levels of the polynomial hierarchy (ie second vs first or something)

Lemma 3.3. *Let*

$$\sigma(y) = \exists^P x_1 \forall^P x_2 \dots Q^P x_i R^A(x_1, ..., x_i, y)$$

be a sentence in $\Sigma_i^{P,A}$. Then there is an equivalent sentence

$$\bar{\sigma}(y) = \exists^P x_1 \forall^P x_2 \dots Q^P x_{i+2} R^A(x_1, ..., x_{i+2}, y)$$

in $\Sigma_{i+2}^{P,A}$ where $\bar{R}^A(x_1, ..., x_{i+2}, y)$ is a predicate that can be computed by a Turing machine that makes at most one oracle query and runs in polynomial time.

Essentially what this lemma is saying is that given an oracle machine at any level of the polynomial hierarchy, we can descend a bit further down the hierarchy to find the same problem decided by, at its

bottom, a machine which only makes a single query, shunting the multiple queries within the quantification itself.

Proof. The intuitive explanation above gives us a clear idea of how to proceed. We show that $R^A(x_1, ..., x_n, y)$ can be expressed as $\exists^P r \forall^P s \bar{R}^A(x_1, ..., x_{i+2}, r, s, y)$ where $\bar{R}^A(x_1, ..., x_{i+2}, r, s, y)$ can be computed by a machine which makes only one oracle query and runs in polynomial time. Substituting this quantified sentence for R^A in σ then clearly gives us the new expression $\bar{\sigma}$. Let $M_R^A(x_1, ..., x_i, r, s, y)$ be a polynomial time machine that computes the predicate $R^A(x_1, ..., x_i, y)$. That is to say, given the inputs $x_1, ..., x_i$, the machine, on input y, runs a polynomial number of steps. At step, say, j, the machine might query the string q_j, and receive an answer r_j. r_j is really a single bit here, representing whether or not $q_j \in A$. One can then, for a fixed input y, visualize a binary tree T, whose nodes represent the query strings q_j, and whose edges represent the path to the next step's query given whether or not $q_j \in A$. Specifying an oracle A then really specifies a polynomial length path through this tree, with leaves representing whether or not the machine itself accepts.

Note that the predicate $R^A(x_1, ..., x_i, y)$ is equivalent to the sentence "there exists an accepting path in the tree T whose path is consistent with A" (that is to say, in which every query response dictating an edge is consistent with A). To formalize this into the sentence we want, let r range over all possible polynomial length sequences of (q_j, r_j) (query, response) pairs (clearly each of which is polynomial length), and s range over all possible queries q_s (each of which is of course polynomial length since the original machine runs in polynomial time). Let $\bar{R}^A(x_1, ..., x_i, r, s, y)$ be the predicate "$r = (q_1, r_1), ..., (q_t, r_i)$" is an accepting branch in T, and if q_s appears as a query node along the branch r, then the associated edge (q_s, r_s) is consistent with A (that is to say, if $r_s = 1$, then $q_s \in A$, and if $r_s = 0$, then $q_s \notin A$). The first part, verifying the path, can be done in polynomial time deterministically without an oracle by simply simulating M with these oracle answers. The second part, verifying (q_s, r_s), can be done with a single oracle query. Clearly then we have

$$R^A(x_1, ..., x_i, y) \iff \exists^P r \forall^P s \bar{R}^A(x_1, ..., x_i, r, s, y)$$

completing the proof. □

Note in the above proof that the choice of structuring our sentence as a $\exists\forall$ alternation was arbitrary - it could just as easily have been $\forall\exists$. Thus by writing the two quantifier sentence such that the first quantifier matches Q_i, we can merge those two quantifiers and accomplish the same task with only a single other alteration. Thus we actually have the stronger corollary:

Corollary 3.2. *Let*

$$\sigma(y) = \exists^P x_1 \forall^P x_2 \dots Q^P x_i R^A(x_1, ..., x_i, y)$$

be a sentence in $\Sigma_i^{P,A}$. Then there is an equivalent sentence

$$\bar{\sigma}(y) = \exists^P x_1 \forall^P x_2 \dots Q^P x_{i+1} R^A(x_1, ..., x_{i+2}, y)$$

in $\Sigma_{i+1}^{P,A}$ where $\bar{R}^A(x_1, ..., x_{i+1}, y)$ is a predicate that can be computed by a Turing machine that makes at most one oracle query and runs in polynomial time.

Lemma 3.4. *Consider the test language*

$$L_A = \{1^n : \text{ the number of strings of length } n \text{ in } A \text{ is odd}\}$$

If this language is in $\Sigma_i^{P,A}$ for some i, then for some fixed k, there exists a depth $i+1$, $O(n^{\log^k(n)})$-size circuit to compute the parity of n variables.

Theorem 3.5. *There exists an oracle A $\Sigma_1^{P,A} \cup \Pi_1^{P,A} \not\subseteq \Sigma_2^{P,A}$.*