

Notes on OpenAI’s solution to the Unitary Synthesis Problem (DRAFT)

Fermi Ma

This write-up is the result of my attempt to understand OpenAI’s solution [OAI] to the Unitary Synthesis problem [AK]. GPT Astra came up with the linear-query algorithm described here after I prompted it to improve the query complexity of OpenAI’s original solution. OpenAI’s paper does not clearly state its query complexity, though Astra estimated it to be around $O(n^{52})$.

All of the following text was written by me, with extensive help from Devin, an AI agent that had access to a PDF generated by Astra. The process went as follows: I would write a bullet, send it to Devin, and ask if it was the correct next step. If I got something wrong, I’d ask Devin to explain the issue, and then I’d rewrite the bullet and check again. Occasionally, I’d get completely stuck and just ask Devin to tell me what the next bullet should be, and then I’d write it up in my own words. The following is the result of repeating this process dozens of times.

My contribution was mostly to steer the proof in a direction that made sense to me. In particular, after struggling to understand Astra’s original approach, I suggested using the compressed-oracle perspective to simplify the arguments in Sections 3 and 4.

Note: this document is not intended to be a complete proof, though I believe it contains essentially all the important steps. Some technical details are omitted/unresolved. The goal of this document is to make the main ideas behind the unitary synthesis algorithm easier to understand. Any frontier model can fill in the remaining proof details if needed.

1 Reducing d -dim synthesis to $\alpha \cdot d$ -dim synthesis

- We’ll show how to reduce synthesizing a d -dim Haar-random unitary to synthesizing k -dim unitaries, plus $O(1)$ additional oracle calls. Provided $k = \alpha \cdot d$ for a constant $\alpha < 1$, this recursion gives an $O(n)$ -query algorithm.
- Let’s first relax our goal: suppose we only want to synthesize U on *some k -dimensional subspace P* . How can we use the ability to synthesize k -dim unitaries to accomplish this?
- Here is a natural idea that we don’t know how to make work, but that will give us some useful intuition. Suppose we can find k -dimensional subspaces P and $Q = U \cdot P$, together with an orthonormal basis $|p_1\rangle, \dots, |p_k\rangle$ of P and an orthonormal basis $|q_1\rangle, \dots, |q_k\rangle$ of Q , such that we can efficiently (or with at most $O(1)$ oracle queries) apply both
 - the partial isometry $\phi_P := \sum_{j \in [k]} |j\rangle\langle p_j|$ (which maps from P to $\mathbb{C}^k$), as well as
 - the isometry $\phi_Q^\dagger := \sum_{j \in [k]} |q_j\rangle\langle j|$ (which maps from $\mathbb{C}^k$ to Q).

If we could find such P and Q , then we could synthesize U on the subspace P as follows:

- **Step 1: Map the input to $\mathbb{C}^k$.** First, given a d -dim vector $|\psi\rangle \in P$, we apply ϕ_P , obtaining a k -dim vector $\phi_P |\psi\rangle$.
- **Step 2: Apply the k -dim unitary V .** Next, we use k -dim unitary synthesis to apply the unitary $V := \sum_{i,j \in [k]} |i\rangle\langle j| \cdot \langle q_i| U |p_j\rangle$.
- **Step 3: Map back to $\mathbb{C}^d$.** Finally, we apply $\phi_Q^\dagger$ to map from $\mathbb{C}^k$ back up to $\mathbb{C}^d$ (in particular, the subspace $Q \subset \mathbb{C}^d$).

This would amount to synthesizing U (on the P subspace) because

$$\phi_Q^\dagger \cdot V \cdot \phi_P |\psi\rangle = \left(\sum_{i \in [k]} |q_i\rangle\langle i| \right) \left(\sum_{i,j \in [k]} |i\rangle\langle j| \cdot \langle q_i| U |p_j\rangle \right) \left(\sum_{j \in [k]} |j\rangle\langle p_j| \right) |\psi\rangle \quad (1)$$

$$= \sum_i |q_i\rangle\langle q_i| U \sum_j |p_j\rangle\langle p_j| |\psi\rangle \quad (2)$$

$$= U |\psi\rangle. \quad (3)$$

- Unfortunately, it's completely unclear how to pick P and Q such that the isometries ϕ_P and $\phi_Q^\dagger$ are efficiently/ $O(1)$ -query computable.
- The real idea is inspired by the above approach, but has a couple twists. The first twist is: we're going to relax the requirement that ϕ_P (the map from P to $\mathbb{C}^k$) and $\phi_Q^\dagger$ (the map from $\mathbb{C}^k$ to Q) are isometries. Equivalently, we are not going to require that the bases $\{|p_i\rangle\}_{i \in [k]}$ and $\{|q_i\rangle\}_{i \in [k]}$ be orthonormal.

This seems obviously wrong at first, since if they're not isometries, then how do we implement $\phi_Q^\dagger \cdot V \cdot \phi_P$? This is where the second twist comes in: we're going to use many pairs of subspaces (P_r, Q_r) in superposition, and only require that the **global map** is an isometry.

- In more detail, we're going to have M pairs of subspaces $\{(P_r, Q_r)\}_{r \in [M]}$, where $Q_r = U \cdot P_r$. For each r , we fix a basis $\{|p_{r,j}\rangle\}_{j \in [k]}$ of P_r and a basis $\{|q_{r,j}\rangle\}_{j \in [k]}$ of Q_r , which need not be orthonormal. As before, we define the maps

$$\phi_{P_r} := \sum_{j \in [k]} |j\rangle\langle p_{r,j}|, \quad (4)$$

$$\phi_{Q_r}^\dagger := \sum_{j \in [k]} |q_{r,j}\rangle\langle j|. \quad (5)$$

Note: we have not yet said how we'll pick P_r, Q_r , or their bases. This will be filled in later.

For each r , we'll also define a $k \times k$ matrix V_r , which describes how U maps P_r to Q_r in terms of our chosen bases. In particular the (i, j) entry of V_r is defined to be the coefficient on $|q_{r,i}\rangle$ when we write $U |p_{r,j}\rangle$ as a linear combination of the basis vectors $|q_{r,1}\rangle, \dots, |q_{r,k}\rangle$. That is, for every $j \in [k]$,

$$U |p_{r,j}\rangle = \sum_{i \in [k]} (V_r)_{ij} |q_{r,i}\rangle. \quad (6)$$

Note that since the bases $\{|p_{r,j}\rangle\}_{j \in [k]}$ and $\{|q_{r,j}\rangle\}_{j \in [k]}$ need not be orthonormal, the definition of V_r does not guarantee that it is a unitary. **However, we will later show how to pick P_r and Q_r and their bases so that V_r will be a unitary.**

- Now, we'll apply a similar template to the one we had above, but with every r in superposition.
 - **Step 1: Map the input to $\mathbb{C}^M \otimes \mathbb{C}^k$.** First, given a d -dim vector $|\psi\rangle$, we apply the map

$$\text{Split} := \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \phi_{P_r}, \quad (7)$$

obtaining an Mk -dim vector

$$\frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \phi_{P_r} |\psi\rangle \quad (8)$$

(Here, ϕ_{P_r} is defined on all of $\mathbb{C}^d$, and not just on P_r . In particular, it maps $|\psi\rangle$ to $\sum_{j \in [k]} \langle p_{r,j} | \psi \rangle |j\rangle$. Note that it acts as 0 on the subspace orthogonal to P_r .)

- **Step 2: Apply the controlled k -dim unitary V_r .** Next, we use k -dim unitary synthesis (in superposition) to apply the controlled- V_r unitary

$$\sum_{r \in [M]} |r\rangle\langle r| \otimes V_r. \quad (9)$$

- **Step 3: Map back to $\mathbb{C}^d$.** Finally, we apply the map

$$\text{Merge} := \frac{1}{\sqrt{M}} \sum_{r \in [M]} \langle r| \otimes \phi_{Q_r}^\dagger \quad (10)$$

to map from $\mathbb{C}^{Mk}$ back to $\mathbb{C}^d$.

- We still haven't said what these subspaces P_r (or its bases) will be, or shown that they can be picked so that **Split** and **Merge** are $O(1)$ -query computable isometries. But, assuming that all this can be done, then the above is a valid unitary synthesis procedure. That is, applying the above three steps to an arbitrary state $|\psi\rangle \in \mathbb{C}^d$ yields $U |\psi\rangle$:

$$\text{Merge} \cdot \left(\sum_{r \in [M]} |r\rangle\langle r| \otimes V_r \right) \cdot \text{Split} \cdot |\psi\rangle \quad (11)$$

$$= \left(\frac{1}{\sqrt{M}} \sum_{r \in [M]} \langle r| \otimes \phi_{Q_r}^\dagger \right) \cdot \left(\sum_{r \in [M]} |r\rangle\langle r| \otimes V_r \right) \cdot \left(\frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \phi_{P_r} \right) \cdot |\psi\rangle \quad (12)$$

$$= \frac{1}{M} \sum_{r \in [M]} \phi_{Q_r}^\dagger \cdot V_r \cdot \phi_{P_r} |\psi\rangle \quad (13)$$

$$= \frac{1}{M} \sum_{r \in [M]} U \cdot \phi_{P_r}^\dagger \cdot \phi_{P_r} |\psi\rangle \quad (\text{see below for an explanation of this step})$$

$$= U \cdot \underbrace{\left(\frac{1}{M} \sum_{r \in [M]} \phi_{P_r}^\dagger \cdot \phi_{P_r} \right)}_{=\text{Split}^\dagger \text{Split}} |\psi\rangle \quad (14)$$

$$= U \cdot |\psi\rangle. \quad (15)$$

In the middle, we used the fact that $\phi_{Q_r}^\dagger V_r = U \phi_{P_r}^\dagger$. To see why this is true, recall that

$$\phi_{Q_r}^\dagger := \sum_{j \in [k]} |q_{r,j}\rangle\langle j|, \quad \phi_{P_r}^\dagger = \sum_{j \in [k]} |p_{r,j}\rangle\langle j|. \quad (16)$$

Then, for all $j \in [k]$,

$$\phi_{Q_r}^\dagger V_r |j\rangle = \sum_{i \in [k]} (V_r)_{ij} \phi_{Q_r}^\dagger |i\rangle = \sum_{i \in [k]} (V_r)_{ij} |q_{r,i}\rangle = U |p_{r,j}\rangle = U \phi_{P_r}^\dagger |j\rangle, \quad (17)$$

where the third equality uses the definition of V_r .

- All that's left is to fill in the missing pieces. Let's take stock of everything we need. We need to pick:
 - some M of size at most $2^{\text{poly}(n)}$ (or else the above won't be efficient),
 - for each r :
 - * a k -dimensional subspace P_r and a corresponding basis $\{|p_{r,j}\rangle\}_{j \in [k]}$
 - * a basis $\{|q_{r,j}\rangle\}_{j \in [k]}$ of Q_r (the subspace $Q_r = U \cdot P_r$ is determined once we pick P_r),

such that:

- **Condition 1:** for each r , V_r is unitary,
- **Condition 2:** Split is an isometry (which is used above to show correctness of the 3-step synthesis procedure),
- **Condition 3:** Split and Merge are both implementable in $O(1)$ oracle queries.

2 Satisfying condition 1

- Let's now see how to pick P_r and the bases such that the resulting V_r is a unitary. For this section, we'll drop the subscript r , and just write P , Q , $|p_j\rangle$, $|q_j\rangle$, and V .

Recall that we defined V to be the $k \times k$ matrix that describes how U maps P to Q in terms of the bases $\{|p_j\rangle\}_{j \in [k]}$ and $\{|q_j\rangle\}_{j \in [k]}$. In the previous section, we showed that this definition implies:

$$U \cdot \phi_P^\dagger = \phi_Q^\dagger \cdot V, \quad \text{where } \phi_P^\dagger = \sum_{j \in [k]} |p_j\rangle\langle j|, \quad \phi_Q^\dagger = \sum_{j \in [k]} |q_j\rangle\langle j|. \quad (18)$$

- For V to be a unitary, we have to ensure that $\|V|y\rangle\| = \||y\rangle\|$ for all $|y\rangle \in \mathbb{C}^k$. The equation above (applied to $|y\rangle$), tells us that

$$U \cdot \phi_P^\dagger |y\rangle = \phi_Q^\dagger \cdot V |y\rangle. \quad (19)$$

Taking the norm of both sides and using the fact that U is unitary (in particular, $\|U|w\rangle\| = \||w\rangle\|$ for all $|w\rangle$), we get

$$\|\phi_P^\dagger |y\rangle\| = \|\phi_Q^\dagger \cdot V |y\rangle\|. \quad (20)$$

If we could somehow “delete” the $\phi_P^\dagger$ and $\phi_Q^\dagger$, then this would say that V is a unitary. But since we're not requiring these operators to be isometries, we can't simply do that.

- What does applying $\phi_P^\dagger$ and $\phi_Q^\dagger$ do to the norm of a state? We'll spend the next few bullets answering this.

For arbitrary P and arbitrary $\{|p_j\rangle\}$ this can be hard to analyze. So instead, we'll consider certain structured P and $\{p_j\}$, for which the effect of $\phi_P^\dagger$ is easier to analyze, and from that we'll derive constraints under which V is guaranteed to be unitary.

- It will turn out to be easiest to consider k -dimensional subspaces P that have the form

$$P = \left\{ \begin{pmatrix} |y\rangle \\ W|y\rangle \end{pmatrix} : |y\rangle \in \mathbb{C}^k \right\} \quad (21)$$

where W is some $s \times k$ matrix, where $s = d - k$. We'll set $|p_j\rangle = (|j\rangle, W|j\rangle)$ (where this notation means $\begin{pmatrix} |j\rangle \\ W|j\rangle \end{pmatrix}$). Then since $\phi_P^\dagger = \sum_{j \in [k]} |p_j\rangle\langle j|$, we have

$$\phi_P^\dagger |y\rangle = \begin{pmatrix} |y\rangle \\ W|y\rangle \end{pmatrix}. \quad (22)$$

- When we choose subspaces P that have the form above, what does Q look like? Let's write $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$, where A is a $k \times k$ matrix and D is $s \times s$. Then, applying U to any vector in P , we get

$$U \begin{pmatrix} |y\rangle \\ W|y\rangle \end{pmatrix} = \begin{pmatrix} (A + BW)|y\rangle \\ (C + DW)|y\rangle \end{pmatrix}. \quad (23)$$

Thus, Q is the set of vectors that can appear on the RHS.

Now, suppose for a moment that $(A + BW)$ is invertible. Define $W' = (C + DW) \cdot (A + BW)^{-1}$. Substituting $|y'\rangle = (A + BW)|y\rangle$ in the RHS above, the top part becomes $|y'\rangle$ and the bottom part becomes $W'|y'\rangle$. So, Q is the set of all vectors of the form $(|y'\rangle, W'|y'\rangle)$. This has the same form as vectors in P , except with a different matrix W' .

We'll set $|q_j\rangle = (|j\rangle, W'|j\rangle)$. Then since $\phi_Q^\dagger = \sum_{j \in [k]} |q_j\rangle\langle j|$, we have

$$\phi_Q^\dagger |y'\rangle = \begin{pmatrix} |y'\rangle \\ W'|y'\rangle \end{pmatrix}. \quad (24)$$

- Combining the last three equations, we have

$$U \cdot \phi_P^\dagger |y\rangle = U \cdot \begin{pmatrix} |y\rangle \\ W|y\rangle \end{pmatrix} = \begin{pmatrix} (A + BW)|y\rangle \\ W' \cdot (A + BW)|y\rangle \end{pmatrix} = \phi_Q^\dagger \cdot (A + BW)|y\rangle. \quad (25)$$

But recall that

$$U \cdot \phi_P^\dagger |y\rangle = \phi_Q^\dagger \cdot V |y\rangle. \quad (26)$$

This implies that $V = A + BW$.

- Now, we can answer the question we asked earlier, which was: under what conditions does $\|\phi_P^\dagger |y\rangle\| = \|\phi_Q^\dagger \cdot V |y\rangle\|$ imply $\| |y\rangle \| = \|V |y\rangle\|$ (i.e., V is unitary)?

Using (25) and unitarity of U , the equation $\|\phi_P^\dagger |y\rangle\| = \|\phi_Q^\dagger \cdot V |y\rangle\|$ implies

$$\| |y\rangle \|^2 + \|W |y\rangle\|^2 = \|V |y\rangle\|^2 + \|W' \cdot V |y\rangle\|^2. \quad (27)$$

Thus, V is unitary as long as

$$\|W |y\rangle\| = \|W' \cdot V |y\rangle\| = \|(C + DW) |y\rangle\| \quad (28)$$

for all $|y\rangle$.

- An easy way to ensure this condition is to first pick an s -dimensional unitary Z , and then set $W = Z(C + DW)$; if we solve this equation for W , we get $W = (I - ZD)^{-1}ZC$. If we do this, then (as we just argued), V is unitary, and since $V = A + BW$, our earlier assumption that $A + BW$ is invertible is justified.
- To summarize: any choice of unitaries $\{Z_r\}_{r \in [M]}$ will induce a choice of $\{W_r\}_{r \in [M]}$, as well as subspaces $\{P_r\}$ (each with a corresponding basis), and matrices $\{V_r\}_{r \in [M]}$ such that each V_r is unitary.

3 Satisfying condition 2

- Next, we'll identify conditions on the unitaries $\{Z_r\}$ under which **Split** is an isometry.

But first, let's collect all the definitions we'll need from the previous sections:

- $U = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ where A is $k \times k$ and D is $s \times s$, and $s = d - k$.
- W_r is a $s \times k$ matrix that we set to be $W_r = (I - Z_r D)^{-1} Z_r C$ to satisfy Condition 1.
- The isometry **Split** was defined so that for all $|x\rangle \in \mathbb{C}^k$ and $|w\rangle \in \mathbb{C}^s$,

$$\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \phi_{P_r} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix}. \quad (29)$$

Since $\phi_{P_r} = \sum_{j \in [k]} |j\rangle \langle p_{r,j}|$ where $|p_{r,j}\rangle = \begin{pmatrix} |j\rangle \\ W_r |j\rangle \end{pmatrix}$, plugging this into the definition of **Split** gives:

$$\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes (|x\rangle + W_r^\dagger |w\rangle), \quad (30)$$

- Actually, the inverse in the definition of $W_r^\dagger$ is annoying to work with. Let's expand it as a geometric series:

$$(I - D^\dagger Z_r^\dagger)^{-1} = I + D^\dagger Z_r^\dagger + (D^\dagger Z_r^\dagger)^2 + (D^\dagger Z_r^\dagger)^3 \dots, \quad (31)$$

which converges as long as $\|D^\dagger Z_r^\dagger\| = \|D\| < 1$ (the first equality is because Z_r is a unitary). Using this expansion, we can rewrite **Split** one last time as:

$$\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \left(|x\rangle + \sum_{m=0}^{\infty} C^\dagger Z_r^\dagger (D^\dagger Z_r^\dagger)^m |w\rangle \right). \quad (32)$$

- The question is, how do we pick $\{Z_r\}$ to make the above map an isometry?
- The RHS of the above expression is scary, but there's a change of perspective that will (in my opinion) make it easier to reason about. We can view $\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix}$ as the result of the following process:

- Let $|+M\rangle := \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle$. Start with the unit-norm state $|+M\rangle \otimes \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix}$.
- Repeat the following steps m times (in a moment, we'll consider the limit as $m \rightarrow \infty$):
 1. Apply a controlled- $Z_r^\dagger$ that acts only to the bottom s coordinates (note that this step is unitary):

$$|r\rangle \otimes \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} \mapsto |r\rangle \otimes \begin{pmatrix} |x\rangle \\ Z_r^\dagger |w\rangle \end{pmatrix}. \quad (33)$$

2. Apply the (not necessarily unitary) map

$$|r\rangle \otimes \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} \mapsto |r\rangle \otimes \begin{pmatrix} |x\rangle + C^\dagger |w\rangle \\ D^\dagger |w\rangle \end{pmatrix}. \quad (34)$$

After m steps, the state we get is

$$\frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \begin{pmatrix} |x\rangle + \sum_{j=0}^{m-1} C^\dagger Z_r^\dagger (D^\dagger Z_r^\dagger)^j |w\rangle \\ (D^\dagger Z_r^\dagger)^m |w\rangle \end{pmatrix}. \quad (35)$$

As $m \rightarrow \infty$, the norm of the $(D^\dagger Z_r^\dagger)^m |w\rangle$ component goes to 0. And $\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix}$ is the “top” component.

- For what choices of $\{Z_r\}$ is the above process norm-preserving?
- Taking inspiration from compressed oracles, suppose we pick $Z_r = \omega_M^r \cdot I$, where $\omega_M = e^{2\pi i/M}$. Let $|\hat{c}\rangle := \frac{1}{\sqrt{M}} \sum_{r \in [M]} \omega_M^{-cr} |r\rangle$ denote the M -ary Fourier basis. (**Note:** This is just one choice of Z_r that satisfies Condition 2. In the next section, we'll use a different choice to satisfy Condition 3, but essentially the same argument will apply to show that Condition 2 holds for that choice of Z_r .) Then $|+M\rangle = |\hat{0}\rangle$. Moreover, applying controlled- $Z_r^\dagger$ to the bottom component increments the value (mod M) on the $|r\rangle$ register in the Fourier basis: $|\hat{c}\rangle |w\rangle \mapsto |\widehat{c+1}\rangle |w\rangle$.
- To see why this works, let's write out the first few rounds explicitly. Initially, our state is

$$|\hat{0}\rangle \otimes \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} = \begin{pmatrix} |\hat{0}\rangle \otimes |x\rangle \\ |\hat{0}\rangle \otimes |w\rangle \end{pmatrix}. \quad (36)$$

After the first round, the state is

$$\begin{pmatrix} |\hat{0}\rangle \otimes |x\rangle + |\hat{1}\rangle \otimes C^\dagger |w\rangle \\ |\hat{1}\rangle \otimes D^\dagger |w\rangle \end{pmatrix}. \quad (37)$$

After the second round, the state is

$$\begin{pmatrix} |\hat{0}\rangle \otimes |x\rangle + |\hat{1}\rangle \otimes C^\dagger |w\rangle + |\hat{2}\rangle \otimes C^\dagger D^\dagger |w\rangle \\ |\hat{2}\rangle \otimes (D^\dagger)^2 |w\rangle \end{pmatrix}. \quad (38)$$

And after the third round, the state is

$$\begin{pmatrix} |\hat{0}\rangle \otimes |x\rangle + |\hat{1}\rangle \otimes C^\dagger |w\rangle + |\hat{2}\rangle \otimes C^\dagger D^\dagger |w\rangle + |\hat{3}\rangle \otimes C^\dagger (D^\dagger)^2 |w\rangle \\ |\hat{3}\rangle \otimes (D^\dagger)^3 |w\rangle \end{pmatrix}. \quad (39)$$

- With this perspective, it's easy to show that every step is norm-preserving. Suppose the state at the beginning of round j is $\begin{pmatrix} |T\rangle \\ |\widehat{j-1}\rangle \otimes |v\rangle \end{pmatrix}$, where $|T\rangle$ denotes the entire top component. At the end of round j , it is $\begin{pmatrix} |T\rangle + |\hat{j}\rangle \otimes C^\dagger |v\rangle \\ |\hat{j}\rangle \otimes D^\dagger |v\rangle \end{pmatrix}$. Crucially, $|\hat{j}\rangle \otimes C^\dagger |v\rangle$ is orthogonal to $|T\rangle$, since $|T\rangle$ is a sum of terms whose Fourier labels are smaller than j (this is easiest to see by just plugging in the examples above).
- Thus, the squared norm of the state before round j is $\langle T|T\rangle + \langle v|v\rangle$, and afterwards it is $\langle T|T\rangle + \langle v|CC^\dagger|v\rangle + \langle v|DD^\dagger|v\rangle = \langle T|T\rangle + \langle v|v\rangle$, where the equality follows from $CC^\dagger + DD^\dagger = I$ (which is implied by unitarity of U). This means each round preserves the norm. Moreover, since the bottom component goes to 0, **Split** is (approximately) an isometry.
- (Technically, this only shows that **Split** is approximately an isometry. But the error is extremely small.)

4 Satisfying Condition 3

- All we have to do now is show how to pick $\{Z_r\}$ so that **Split** and **Merge** can be implemented in $O(1)$ oracle queries. We'll focus on **Split**, since the argument for **Merge** is analogous.
- Using state synthesis, e.g., [Ros], we can implement the map $|t\rangle \mapsto |t\rangle \otimes \text{Split } |t\rangle$ in $O(1)$ queries, for any standard basis state $t \in [d]$. Thus, to implement **Split**, it suffices to show how to compute t given **Split** $|t\rangle$ so we can uncompute it.

It turns out, we won't *quite* be able to do this. However, let's analyze how to recover $|t\rangle$ from **Split** $|t\rangle$. We'll get something close, and then at the end of this section, we'll see how the real algorithm works by synthesizing a slightly different state (a "reweighted" version of **Split** $|t\rangle$). This sounds like it will be very confusing, but bear with me.

- Let's recall the expression for **Split** we derived in the previous section. For

$$\text{Split} \cdot \begin{pmatrix} |x\rangle \\ |w\rangle \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \left(|x\rangle + \sum_{m=0}^{\infty} C^\dagger Z_r^\dagger (D^\dagger Z_r^\dagger)^m |w\rangle \right). \quad (40)$$

- There are two cases to consider:

– In the first case, $|t\rangle = \begin{pmatrix} |j\rangle \\ 0 \end{pmatrix}$ for some $j \in [k]$. Then the formula gives

$$\text{Split} \cdot \begin{pmatrix} |j\rangle \\ 0 \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes |j\rangle. \quad (41)$$

In this case, we can recover $|t\rangle$ from simply looking at the second register. So this case is easy.

– In the second case, $|t\rangle = \begin{pmatrix} 0 \\ |a\rangle \end{pmatrix}$ for some $a \in [s]$. Then

$$\text{Split} \cdot \begin{pmatrix} 0 \\ |a\rangle \end{pmatrix} = \frac{1}{\sqrt{M}} \sum_{r \in [M]} |r\rangle \otimes \left(\sum_{m=0}^{\infty} C^\dagger Z_r^\dagger (D^\dagger Z_r^\dagger)^m |a\rangle \right). \quad (42)$$

This is the tricky case. As a spoiler, we won't actually be able to recover a exactly, but we will be able to come up with a short list of candidates for a , and then we'll need an amplitude amplification step to uncompute a .

Next, we'll show how to pick $\{Z_r\}$ so that, given the above state, we can output a short list of candidates that contains a (where the short list is in constant-size in expectation, which will turn out to be enough).

- Again, we'll use the compressed-oracle-style framing from the previous section. There, we thought of Split as acting in rounds, where in each round we apply a controlled- $Z_r^\dagger$ to the bottom component, and simultaneously add 1 to the Fourier label in every round.

The intuition is that now, we'll pick the $Z_r^\dagger$ matrices so that, instead of adding 1 to the Fourier label in every round, we add a value μ_b that depends on which bottom coordinate $b \in [s]$ we're at, so that in the Fourier basis, the first register records more information, which will help us in reconstructing a . In particular, the bottom component starts at a , and so the first round will add μ_a to the Fourier label. Then applying $D^\dagger$ turns the bottom component into a superposition over other bottom coordinates $b \in [s]$, then the second round adds μ_b to the Fourier label, and so on.

Concretely, we'll just pick $Z_r = \text{diag}(\omega_M^{\mu_{1r}}, \dots, \omega_M^{\mu_{sr}})$ where $\mu_1, \dots, \mu_s \in \{0, 1, \dots, M-1\}$ are integers that we'll set in a moment.

Now, step 1 of each round corresponds to mapping $|\widehat{c}\rangle |b\rangle \mapsto |\widehat{c + \mu_b}\rangle |b\rangle$ for each bottom coordinate $b \in [s]$.

- After L rounds, the Fourier label is the sum of the μ 's of all the bottom coordinates we visit. But we know at least one of these coordinates is a . So, we'll simply pick the μ 's such that this sum determines which coordinates were visited, as long as the list contains at most L elements (Any set of integers with this property will suffice; the AI suggests "power sums"). This will give us our candidate list that contains a .
- This sounds scary: we continue this process for infinitely many rounds, so won't a get buried in an infinitely long list? It turns out this won't be the case. Any term that gets added to the top component in round m has a list of size at most m (which will not get larger in any

subsequent round). And crucially, this term is $C^\dagger$ applied to the bottom component after $m-1$ rounds, and that term has squared norm at most $\|D\|^{2(m-1)}$. (For intuition, it is helpful to look back at how the examples in the previous section work.)

To ensure the list contains at most L elements, we can either consider the truncation of the round-by-round implementation of **Split** after L rounds (and get an approximation of **Split** that has error $\|D\|^L$), or implement **Split** exactly, and just accept that if we get really unlucky and measure a list of length $> L$, we might not be able to recover a .

- The expected list size is something like $\sum_{m \geq 0} \|D\|^{2m} \leq 2$ if $\|D\| \leq 1/\sqrt{2}$. (Where does this assumption on D come from? One can show it's satisfied with very high probability for Haar-random unitaries U . Moreover, when we recurse, I believe the inner V_r unitaries will also be Haar-random, so this assumption will continue to hold.)
- What we've shown so far is that if we choose $\{Z_r\}$ as above, and then view the first register of **Split** $|t\rangle$ in the Fourier basis, it is a superposition of the form

$$\text{Split } |t\rangle = \sum_S |\{a\} \cup S\rangle \otimes |\psi_{\{a\} \cup S}\rangle \quad (43)$$

where the sum is over multisets S of bottom coordinates in $[s]$ (it is a multiset since a coordinate can be visited more than once). The states $|\psi_{\{a\} \cup S}\rangle$ are **unnormalized**. Moreover, the total squared norm of the terms with $|S| = m$ is at most $\|D\|^{2m}$.

- This looks promising, but recall that we need to compute a with extremely high probability.
- Let's see what happens when we try to uncompute $|a\rangle$ using the information in the $|\{a\} \cup S\rangle$ register. After state synthesis, our state is

$$|a\rangle \otimes \underbrace{\sum_S |\{a\} \cup S\rangle \otimes |\psi_{\{a\} \cup S}\rangle}_{=\text{Split } |t\rangle} \quad (44)$$

Our goal is to cleanly erase $|a\rangle$, i.e., set it back to $|0\rangle$, without disturbing the rest of the state.

- But the natural way to uncompute a is as follows: controlled on $\{a\} \cup S$, apply to the first register the inverse of the unitary that maps $|0\rangle$ to the uniform superposition over the distinct elements of $\{a\} \cup S$. This maps $|a\rangle$ to $|0\rangle$ with amplitude $\frac{1}{\sqrt{|\{a\} \cup S|}}$ (here I'm taking $|\{a\} \cup S|$ to mean the number of unique elements in $\{a\} \cup S$). The result of this is

$$|0\rangle \otimes \sum_S \frac{1}{\sqrt{|\{a\} \cup S|}} |\{a\} \cup S\rangle \otimes |\psi_{\{a\} \cup S}\rangle + \text{orthogonal terms}, \quad (45)$$

where the orthogonal terms are orthogonal because their first register is not 0.

- This looks concerning for two reasons. First, we have these annoying orthogonal terms. Second, the state in tensor product with $|0\rangle$ is not **Split** $|t\rangle$, but some weird, reweighted version of it.
- Let's ignore the first issue for the moment and focus on the second one. To resolve this problem, we'll have to change the state we synthesize. Instead of synthesizing **Split** $|t\rangle$, we'll synthesize the following *reweighted* version of **Split** $|t\rangle$:

$$\frac{1}{2} \sum_S \sqrt{|\{a\} \cup S|} \cdot |\{a\} \cup S\rangle \otimes |\psi_{\{a\} \cup S}\rangle. \quad (46)$$

(Okay, it turns out this state has norm less than 1, so technically you need to synthesize this state plus a clearly labeled orthogonal junk state.)

- Now, when we perform the uncomputation step, the $\sqrt{|\{a\} \cup S|}$ factors cancel out, and the resulting state is $\frac{1}{2} |0\rangle \otimes \text{Split} |t\rangle + \text{orthogonal terms}$. (Note that if we didn't have the $1/2$ factor, the state we'd be writing down at this step would have norm greater than 1.)
- The last step is to perform one step of amplitude amplification to amplify the $1/2$ to 1. This gives us a constant blow-up in the number of queries, but we can tolerate that.

References

- [AK] Scott Aaronson and Greg Kuperberg. Quantum versus classical proofs and advice. *Theory of Computing*, 3:129–157, 2007.
- [Ros] Gregory Rosenthal. Efficient quantum state synthesis with one query. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2508–2534, 2024. arXiv:2306.01723.
- [OAI] OpenAI. Polynomial-time unitary synthesis from a Boolean oracle. Manuscript, October 5, 2026.