Week 5: Quantum Query Complexity

Stacey Jeffery

March 6, 2025

5.1 Introduction

Quantum query complexity is perhaps the most important measure for understanding the power of quantum computers, for the following reasons:

1. It is a reasonable (though imperfect) proxy for the time required by a quantum computer to solve a problem.
2. It is sufficiently simple and structured that we can actually say something about it.

In fact, quantum query complexity is even better understood than its classical counterparts, deterministic query complexity and randomized query complexity, because, as we will see, we have a perfect characterization of it as a semidefinite program (optimization problem, see Appendix .1). Quantum query *upper bounds* are obtained from algorithms.[1] In Section 5.3, we will see several methods of proving quantum query *lower bounds*.

More generally, to try to understand the power of quantum computers relative to classical computers, some concrete questions we can ask are:

1. How does quantum query complexity $\mathsf{Q}(\mathrm{F})$ compare to classical measures of complexity? For example, can we show that *for all* problems F with a certain property, the quantum query complexity of F can't be *that much* better than, say, the randomized query complexity (or some other measure)?
2. What is the biggest separation between $\mathsf{Q}(\mathrm{F})$ and $\mathsf{R}(\mathrm{F})$ (randomized query complexity) for *any* F?

We will also study these questions this week.

5.2 Measures of Complexity

In this section, we will define several measures of complexity, and compare them with quantum query complexity, either in analogy (i.e., they can be derived as somewhat similar looking expressions) or giving concrete bounds.

We will talk a lot about optimization problems in this section and the next. A review of some basics can be found in Appendix .1. For the most part, the details of which optimization problems are linear and which are semidefinite are not too important, but it is useful to understand, at least at a high level, the concept of *duality*.

Two natural measures of complexity to compare to $\mathsf{Q}(\mathrm{F})$ are deterministic and randomized query complexity. A deterministic query algorithm is a classical deterministic algorithm that accesses the

[1] Even algorithms that are not fully worked out, in the sense that only their query complexity is analyzed, such as those obtained from span programs.

input $x \in \{0,1\}^n$ by querying the value of x_i for some chosen $i \in [n]$ at each step. Deterministic query complexity is sometimes called *decision tree complexity*, because a deterministic algorithm can be viewed as a rooted tree, with root s, where every node has an associated index i, and two outgoing edges, labelled by 0 and 1. Every leaf is either an accepting leaf or a rejecting leaf, and a decision tree is evaluated by starting from s, and at any node with index i, query x_i and use its value to decide which edge to traverse, until a leaf is reached, which determines whether to accept or reject. A decision tree is similar to a branching program, except that it must be a tree. In the same way, any input x induces a unique path from s to a leaf, whose length is the query complexity of the decision tree on input x. The maximum query complexity over all inputs gives the query complexity of the decision tree, and the minimum query complexity for any decision tree that decides F is its deterministic query complexity, denoted $\mathsf{D}(\mathrm{F})$. Similarly, we can define the randomized query complexity of F as the minimum query complexity of any *randomized* algorithm that decides F with bounded error.

$\mathsf{D}(\mathrm{F})$: deterministic query complexity of F
$\mathsf{R}(\mathrm{F})$: (bounded-error) randomized query complexity of F

We immediately have

$$\mathsf{Q}(\mathrm{F}) \leq \mathsf{R}(\mathrm{F}) \leq \mathsf{D}(\mathrm{F})$$

The natural classical analogue of Q is R, but we can also gain some insight from other classical measures, as we will see shortly.

5.2.1 From certificates to witnesses

Another measure of complexity called *certificate complexity* is less operational, but quite intuitive. We first need the notion of a certificate.

Definition 5.2.1 (Certificate)**.** *For a function* $\mathrm{F} : D \to \{0,1\}$ *with* $D \subseteq \{0,1\}^n$*, and input* $x \in \{0,1\}^n$*, a* certificate *for* x *(wrt* F*) is a set* $S \subseteq [n]$ *such that: for all* $y \in \{0,1\}^n$ *such that for all* $i \in S$*,* $y_i = x_i$*,* $\mathrm{F}(y) = \mathrm{F}(x)$*. In other words, it's a set of indices into* x *such that if you check all those bits of* x*, you will know the value of* $\mathrm{F}(x)$*.*

Example 5.2.2. Let $x \in \{0,1\}^n$ be such that $\textsc{Or}(x) = 1$. Then for any i such that $x_i = 1$, $\{i\}$ (or any set that contains i) is a certificate for x with respect to Or.

Example 5.2.3. Let G be a graph given in the adjacency matrix model, so we can query $G_{u,v}$ for $u < v$ to learn if $\{u,v\} \in E(G)$. Let F be the problem Ustcon, defined in Section 4.4. Then if $s = u_0, \ldots, u_\ell = t$ is a path from s to t in G, then $S = \{\{u_0,u_1\}, \ldots, \{u_{\ell-1}, u_\ell\}\}$ is a certificate for G with respect to F.

Exercise 5.2.1. *Show that for any total* $\mathrm{F} : \{0,1\}^n \to \{0,1\}$*, if* S_x *is a certificate for* $x \in \mathrm{F}^{-1}(0)$*, and* S_y *is a certificate for* $y \in \mathrm{F}^{-1}(1)$*, then* $S_x \cap S_y \cap \{i : x_i \neq y_i\} \neq \emptyset$*.*

Definition 5.2.4 (Certificate Complexity)**.** *For a function* $\mathrm{F} : D \to \{0,1\}$ *with* $D \subseteq \{0,1\}^n$*, and input* $x \in D$*, the* certificate complexity *of* x *with repect to* F*,* $\mathsf{C}(x, \mathrm{F})$ *is defined as the smallest* $|S|$ *such that* S *is a certificate for* x *with respect to* F*. The certificate complexity of* F *is defined* $\mathsf{C}(\mathrm{F}) = \max_{x \in D} \mathsf{C}(x, \mathrm{F})$*. For* $b \in \{0,1\}$*, we can also define* $\mathsf{C}_b(\mathrm{F}) = \max_{x \in \mathrm{F}^{-1}(b)} \mathsf{C}(x, \mathrm{F})$*.*

Example 5.2.5. In the case of $\textsc{Or}_n$, for any $i \in [n]$ such that $x_i = 1$, $\{i\}$ is a certificate for x, so $\mathsf{C}_1(\textsc{Or}) = 1$. It is easy to see that no set smaller than $[n]$ could be a certificate for 0^n, so $\mathsf{C}_0(\textsc{Or}) = n$.

Certificate complexity could equivalently be called non-deterministic query complexity, since it is precisely the number of queries needed by a non-deterministic algorithm for F. That is, a non-deterministic algorithm that tries all sets of $\mathsf{C}(F)$ queries will be convinced of the answer in at least one path, for

every input; and it is also easy to see that if a non-deterministic algorithm has some computation path that makes at most T queries, for every input, then every input has a certificate of size at most T. This immediately gives the inequality:

$$\forall \mathrm{F},\ \mathsf{C}(\mathrm{F}) \leq \mathsf{D}(\mathrm{F}) \tag{3}$$

However, certificate complexity is also related to deterministic complexity for *total* functions by the inequality:

$$\forall \text{ total } \mathrm{F},\ \mathsf{D}(\mathrm{F}) \leq \mathsf{C}(\mathrm{F})^2.$$

To prove this inequality, we will show how any set of certificates for F can be used to design a deterministic algorithm for F (as long as F is total). In fact, we will see something slightly stronger. For $b \in \{0,1\}$, we have

$$\mathsf{C}_b(\mathrm{F}) = \max_{x \in \mathrm{F}^{-1}(b)} \mathsf{C}(x, \mathrm{F}),$$

so that clearly $\mathsf{C}(\mathrm{F}) = \max\{\mathsf{C}_0(\mathrm{F}), \mathsf{C}_1(\mathrm{F})\}$. We will show the following.

Theorem 5.2.6. *For all total functions* $\mathrm{F} : \{0,1\}^n \to \{0,1\}$*,*

$$\mathsf{D}(\mathrm{F}) \leq \mathsf{C}_0(\mathrm{F})\mathsf{C}_1(\mathrm{F}). \tag{5.1}$$

Proof. For each $x \in \{0,1\}^n$, let S_x be a certificate for x with respect to F with $|S_x| \leq \mathsf{C}_b(\mathrm{F})$ whenever $\mathrm{F}(x) = b$. Consider the following algorithm that takes input $z \in \{0,1\}^n$:

Algorithm 1. Certificate Finding

1. Let $\mathcal{X} \leftarrow \mathrm{F}^{-1}(0)$ and $\mathcal{Y} \leftarrow \mathrm{F}^{-1}(1)$.
2. While $\mathcal{X} \neq \emptyset$ and $\mathcal{Y} \neq \emptyset$:
   (a) Choose $y \in \mathcal{Y}$ and query z_i for all $i \in S_y$.
   (b) Remove from $\mathcal{X}$ any strings x that are not consistent with the queried values.
   (c) Remove from $\mathcal{Y}$ any strings y that are not consistent with the queried values.
3. If $\mathcal{X} = \emptyset$, output 1, otherwise output 0.

The algorithm starts with all strings in $\mathcal{X} \cup \mathcal{Y}$ as candidates for z, and every time it queries a bit of z, it eliminates all strings that are not consistent with that query. It's clear that this algorithm computes $\mathrm{F}(z)$, since it doesn't stop until all of $\mathrm{F}^{-1}(0)$ or $\mathrm{F}^{-1}(1)$ has been eliminated.

We will argue that the total number of queries made by this algorithm is at most $\mathsf{C}_0(\mathrm{F})\mathsf{C}_1(\mathrm{F})$, which will prove the theorem statement. First notice that any iteration of the while loop makes at most $|S_y| \leq \mathsf{C}_1(\mathrm{F})$ queries. We claim the folllowing.

Claim 5.2.7. *Let* $Q \subset [n]$*,* $\overline{Q} = [n] \setminus Q$*, and define*

$$\mathcal{X} = \{x \in \mathrm{F}^{-1}(0) : \forall i \in Q, x_i = z_i\}$$

$$\mathcal{Y} = \{y \in \mathrm{F}^{-1}(1) : \forall i \in Q, y_i = z_i\}.$$

For all $y \in \mathcal{Y}$ *and* $x \in \mathcal{X}$*, there is some* $i \in S_x \cap S_y$ *such that* $i \in \overline{Q}$*.*

The set Q represents the set of indices already queried at the start of any iteration of the loop – and so $\mathcal{X}$ and $\mathcal{Y}$ are precisely as stated in the claim. The y in the claim can be the y chosen in the first step of the loop. Then the claim says that for each $x \in \mathrm{F}^{-1}(0)$ that remains in $\mathcal{X}$ in any given iteration, a new $i \in S_x$ is queried – $i \in \overline{Q}$ means it has not yet been queried, and since it is in S_y, it will now be queried for the first time. This implies that any x can be in $\mathcal{X}$ for at most $|S_x| \leq \mathsf{C}_0(\mathrm{F})$ iterations.

We now prove the claim. Define $\mathrm{F}_{\overline{Q}} : \{0,1\}^{\overline{Q}} \to \{0,1\}$ as follows. $\mathrm{F}_{\overline{Q}}(x) = \mathrm{F}(x')$, where $x'_i = x_i$ for all $i \in \overline{Q}$, and $x'_i = z_i$ for all $i \in [n] \setminus \overline{Q}$. Here we are using the fact that F is a total function, because if it were a partial function, the value $\mathrm{F}(x')$ would not be defined for all strings x' formed this way. So we have basically defined a total function on a set of shorter strings, where the 0-inputs are in one-to-one correspondance with $\mathcal{X}$, and 1-inputs are in one-to-one correspondance with $\mathcal{Y}$ ($\mathrm{F}_{\overline{Q}}$ is called a *sub-function* of F). Then $S_x \setminus Q$ is a certificate for x wrt $\mathrm{F}_{\overline{Q}}$, and $S_y \setminus Q$ is a certificate for x wrt $\mathrm{F}_{\overline{Q}}$, so we must have

$$(S_x \setminus Q) \cap (S_y \setminus Q) = (S_x \cap S_y) \setminus Q = S_x \cap S_y \cap \overline{Q}$$

non-empty by Exercise 5.2.1. □

For reasons that will soon become clear, let us write $\mathsf{C}(\mathrm{F})$ for $\mathrm{F} : D \to \{0,1\}$, $D \subseteq \{0,1\}^n$, as an optimization problem:

$$\begin{aligned} \mathsf{C}(\mathrm{F}) = \text{minimize: } & \max_{x \in D} \sum_{i=1}^{n} w_{x,i} \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \{0,1\} \\ & \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \left(\sum_{i:x_i \neq y_i} w_{x,i}\right)\left(\sum_{i:x_i \neq y_i} w_{y,i}\right) \geq 1. \end{aligned} \tag{5.2}$$

Above, $w_{x,i}$ represents a bit indicating if $i \in S_x$ (the certificate for x), and so $\sum_{i=1}^{n} w_{x,i} = |S_x|$. The constraint is equivalent to $S_x \cap \{i : x_i \neq y_i\} \neq \emptyset$ and $S_y \cap \{i : x_i \neq y_i\} \neq \emptyset$, which is equivalent to y not agreeing with x on S_y (otherwise we would have $\mathrm{F}(x) = \mathrm{F}(y)$) and x not agreeing with y on S_x (or again, we would have $\mathrm{F}(x) = \mathrm{F}(y)$).

For *total* functions F, we can replace the constraints $\sum_{i:x_i \neq y_i} w_{x,i}$ with the seemingly stronger constraints $\sum_{i:x_i \neq y_i} w_{x,i} w_{y,i}$, which is equivalent to $S_x \cap S_y \cap \{i : x_i \neq y_i\} \neq \emptyset$, which we know by Exercise 5.2.1 is true for certificates of a *total* function. That is, for all *total* F:

$$\begin{aligned} \mathsf{C}(\mathrm{F}) = \text{minimize: } & \max_{x \in D} \sum_{i=1}^{n} w_{x,i} \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \{0,1\} \\ & \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i \neq y_i} w_{x,i} w_{y,i} \geq 1. \end{aligned} \tag{5.3}$$

Fractional Certificate Complexity A natural relaxation of the definition of $\mathsf{C}(f)$ in (5.2) is to allow the $w_{x,i}$ values to be in $\mathbb{R}$, yielding what we call the *fractional certificate complexity* of F:

$$\begin{aligned} \mathsf{FC}(\mathrm{F}) = \text{minimize: } & \max_{x \in D} \sum_{i=1}^{n} w_{x,i} \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\ & \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \left(\sum_{i:x_i \neq y_i} w_{x,i}\right)\left(\sum_{i:x_i \neq y_i} w_{y,i}\right) \geq 1. \end{aligned} \tag{5.4}$$

So a fractional certificate for x with respect to F is a set of non-negative real values $\{w_{x,i}\}_{i=1}^n$ such that for all y with $\mathrm{F}(x) \neq \mathrm{F}(y)$, $\sum_{i:x_i \neq y_i} w_{x,i} \geq 1$. The optimization problem in (5.4) is a (linear) *relaxation* (see Appendix .1) of (5.2), so clearly we have $\mathsf{FC}(\mathrm{F}) \leq \mathsf{C}(\mathrm{F})$ for all F.

A more operational definition of FC is in terms of a game, played between Alice and a referee. The referee chooses a bit $b \in \{0,1\}$ and an input pair $(x,y) \in \mathrm{F}^{-1}(b) \times \mathrm{F}^{-1}(1-b)$, and sends x to Alice. Alice wins if she can output $i \in [n]$ such that $x_i \neq y_i$. An optimal (up to constants) strategy for Alice is to choose i with probability proportional to $w_{x,i}$, which gives her winning probability $1/\mathsf{FC}(\mathrm{F})$.

Expectational Certificate Complexity Along similar lines, we can define the *expectational certificate complexity* as follows:

$$\begin{aligned} \mathsf{EC}(\mathrm{F}) = \text{minimize: } & \max_{x \in D} \sum_{i=1}^{n} w_{x,i} \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in [0,1] \\ & \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i \neq y_i} w_{x,i} w_{y,i} \geq 1. \end{aligned} \tag{5.5}$$

It is no longer possible to talk about an "expectational certificate for x with respect to f" – it is necessary to talk about a *set* of expectational certificates, $\mathcal{W} = \{\{w_{x,i}\}_{i=1}^n\}_{x \in D}$, as the constraints involving x depend on the values $w_{y,i}$ for other inputs y.[2] While it is possible to define "a certificate for x" or (fractional certificate for x) without defining certificates for all other inputs, this is not true for expectational certificates.

The quantity EC was introduced in [JKK+20] to study other complexities. It is known that

$$\mathsf{EC}(\mathrm{F}) \leq \mathsf{C}(\mathrm{F}) \leq \mathsf{EC}(\mathrm{F})^2$$

and

$$\mathsf{FC}(\mathrm{F}) \leq \mathsf{EC}(\mathrm{F}) \leq \mathsf{FC}(\mathrm{F})^{3/2}$$

for all F. It is an open problem whether or not EC and FC are always equal up to constants.

To try to interpret this quantity in an operational way, note that the $w_{x,i}$ are in $[0,1]$, so we can interpret them as probabilities. Then, for example, it is shown in [JKK+20] that[3] for all F, $\mathsf{R}(\mathrm{F}) = O(\mathsf{EC}(\mathrm{F})^2)$. This is accomplished using an algorithm like Algorithm 1, except that the algorithm is randomized: instead of querying all of S_y, an index is chosen according to the distribution $\Pr[i] \propto w_{y,i}$. (Showing that $O(\mathsf{EC}(\mathrm{F})^2)$ iterations are sufficient is non-trivial).

Certificate Game Complexity Consider the following game, for some fixed $\mathrm{F} : D \to \{0,1\}$ with $D \subseteq \{0,1\}^n$, played between a referee, and cooperating players Alice and Bob. The referee gives Alice an input $x \in D$, and Bob an input $y \in D$ such that $\mathrm{F}(x) \neq \mathrm{F}(y)$, and they must both output some $i \in [n]$ (the same i) such that $x_i \neq y_i$. If we view this game as a *communication* task, we can ask how many bits of information must be exchanged between Alice and Bob for them to win the game, which turns out to be characterized, up to constants, by the *circuit depth* of F for any *total* F [KW90]. We will instead be interested in this game in the setting where no communication is allowed, and we will ask what is the worst-case probability ω_{F}, over all inputs, that Alice and Bob win this game using an optimal strategy. We will not allow Alice and Bob to use any shared resources, such as shared randomness, or entanglement (although these variations can also be considered) [CGL+23]. Then we let $\mathsf{CG}(\mathrm{F}) = 1/\omega_{\mathrm{F}}$. It turns out that $\mathsf{CG}(\mathrm{F})$ can be written as the following optimization problem,

[2] While the constraints in (5.2) and (5.4) also involve an x and y simultaneously, this is because we are stating two constraints at once. See the equivalent formulation in Figure 5.1.

[3] In fact, a stronger statement replacing R with *one-sided* randomized error is shown.

which looks intriguingly similar to expectational certificate complexity in (5.5).

$$\begin{aligned}\mathsf{CG}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\left(\sum_{i=1}^{n} w_{x,i}\right)^2 \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\ & \forall (x,y) \in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1.\end{aligned}$$

Let us massage the above optimization problem, ever so slightly, replacing the square of the ℓ_1-norm in the objective function with the square of the ℓ_2-norm, to get a quantity we will call W' for now:

$$\begin{aligned}\mathsf{W}'(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i}^2 \\ \text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\ & \forall (x,y) \in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1.\end{aligned} \tag{5.6}$$

While the optimization problem for FC in (5.4) is a linear relaxation of the optimization problem for C in (5.2), the above optimization problem is a *semidefinite relaxation* of the optimization problem in (5.3) that defines $\mathsf{C}(\mathrm{F})$ whenever F is a *total* function. To see this, note that if $\{w_{x,i}\}_{x,i}$ is any feasible solution for the optimization problem for C, then since $w_{x,i} \in \{0,1\}$, $w_{x,i}^2 = w_{x,i}$, so we could replace the objective function in (5.2) with $\max_{x\in D}\sum_i w_{x,i}^2$, as in (5.6).

Exercise 5.2.2. *Show that for all total functions* F, $\mathsf{W}'(\mathrm{F}) \leq \sqrt{\mathsf{C}_0(\mathrm{F})\mathsf{C}_1(\mathrm{F})}$.

There are two reasons we are interested in the quantity W'. First, it turns out to be equal to something somewhat interesting, as we will see in Section 5.3.1. Second, we are building up to something. In the interest of this building up, you will show that the optimization problem in (5.6) has the same objective value even if you allow the variables $w_{x,i}$ to be vectors:

Exercise 5.2.3. *Show that for any* F,

$$\begin{aligned}\mathsf{W}'(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} \||w_{x,i}\rangle\|^2 \\ \text{subject to: } & \forall x \in D, i \in [n],\ |w_{x,i}\rangle \in \mathbb{R}^d \\ & \forall (x,y) \in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} \langle w_{x,i}|w_{y,i}\rangle \geq 1.\end{aligned} \tag{5.7}$$

Witness Complexity Now we will define what I think of as the true quantum analogue of certificates (though this is probably somewhat a matter of opinion). Define:

$$\begin{aligned}\mathsf{W}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} \||w_{x,i}\rangle\|^2 \\ \text{subject to: } & \forall x \in D, i \in [n],\ |w_{x,i}\rangle \in \mathbb{R}^d \\ & \forall (x,y) \in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} \langle w_{x,i}|w_{y,i}\rangle = 1.\end{aligned} \tag{5.8}$$

This is extremely similar to (5.7)[4], but it is more constrained, meaning that $\mathsf{W}(\mathrm{F}) \geq \mathsf{W}'(\mathrm{F})$. In analogy with certificates – or perhaps expectational certificates, where certificates for all inputs x must be defined simultaneously – we will say that:

[4]For easy comparison, we summarize all these optimization problems in Figure 5.1, and the relationship between various quantities in Figure 5.2.

a set of vectors $W = \{|w_x\rangle\}_{x\in D}$ is a *set of witnesses* for $\mathrm{F} : D \to \{0,1\}$ if for each $y \in \mathrm{F}^{-1}(1)$, $|w_y\rangle$ is of the form

$$\sum_{i\in[n]} |i, y_i\rangle|w_{y,i}\rangle$$

for some $\{|w_{y,i}\rangle\} \subset \mathbb{R}^d$; and for each $x \in \mathrm{F}^{-1}(0)$, $|w_x\rangle$ is of the form:

$$\sum_{i\in[n]} |i, x_i \oplus 1\rangle|w_{x,i}\rangle$$

for some $\{|w_{x,i}\rangle\} \subset \mathbb{R}^d$, and for all $(x, y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$,

$$\langle w_x|w_y\rangle = \sum_{i:x_i\neq y_i} \langle w_{x,i}|w_{y,i}\rangle = 1.$$

The *complexity* of the set of witnesses is then $\mathcal{C}(W) = \max_{x\in D} \||w_x\rangle\|^2 = \max_{x\in D} \sum_{i=1}^{n} \||w_{x,i}\rangle\|^2$. **[4]**

How does this concept relate to the witnesses you have seen so far? It turns out that these are basically just span program witnesses: that is, a set of witnesses defines a span program (Lemma 5.2.9 below), and any span program's witnesses can be turned into a set of witnesses as described above (Exercise 5.2.4 below). One implications of this, by Theorem 4.2.6, is that $\mathsf{W}(\mathrm{F}) = \mathsf{SPC}(\mathrm{F}) = \Theta(\mathsf{Q}(\mathrm{F}))$ – so unlike in the case of classical certificates which are only loosely related to query complexity, quantum witnesses capture quantum query complexity perfectly. Let us now see this more precisely.

A *real* span program is a span program as in Definition 4.2.1, except that H and V are vector spaces over $\mathbb{R}$ instead of $\mathbb{C}$. The following claim shows that we can restrict our attention to real span programs without loss of generality.

Claim 5.2.8. *For any (complex) span program P, there is a real span program P' such that for any y that is accepted by P, it is also accepted by P', and $w_+(y, P') \leq w_+(y, P)$; and for any x that is rejected by P, it is also rejected by P', and $w_-(x, P') \leq w_-(x, P)$.*

Exercise 5.2.4. *Let P be a* real *span program that decides F. For all $x \in \mathrm{F}^{-1}(0)$, let $|\omega_x\rangle$ be an optimal negative witness, and define $|w_x\rangle = (\langle\omega_x|A)^\dagger$; and for all $y \in \mathrm{F}^{-1}(1)$, let $|w_y\rangle$ be an optimal positive witness.*

1. *Show that $\{|w_x\rangle\}_{x\in D}$ is a set of witnesses for F. (You can assume the span program does not use the spaces H_{true} or H_{false}).*

2. *Show that there is a positive real α such that if we scale all witnesses $|w_x\rangle$ for $x \in \mathrm{F}^{-1}(0)$ by $\sqrt{\alpha}$, and all witnesses $|w_y\rangle$ for $y \in \mathrm{F}^{-1}(1)$ by $1/\sqrt{\alpha}$, then we get a set of witnesses W with $\mathcal{C}(W) = \mathcal{C}(P)$. Thus $\mathsf{W}(\mathrm{F}) \leq \mathsf{SPC}(\mathrm{F})$.*

Lemma 5.2.9. *Let W be a set of witnesses for F. Then there is a span program P that decides F with complexity $\mathcal{C}(P) \leq \mathcal{C}(W)$. Thus $\mathsf{SPC}(\mathrm{F}) \leq \mathsf{W}(\mathrm{F})$.*

Proof. For any $i \in [n]$ and $b \in \{0,1\}$, define

$$H_{i,b} = \mathrm{span}\{|i, b\rangle\} \otimes \mathbb{R}^d,$$

where d is the dimension from W. Thus, for all $y \in \mathrm{F}^{-1}(1)$,

$$|w_y\rangle = \sum_{i\in[n]} |i, y_i\rangle|w_{y,i}\rangle \in H(y), \tag{5.9}$$

and for all $x \in \mathrm{F}^{-1}(0)$,

$$|w_x\rangle = \sum_{i\in[n]} |i, x_i \oplus 1\rangle |w_{x,i}\rangle \in H(x)^{\perp}. \tag{5.10}$$

Next, define

$$V = \mathrm{span}\{|x\rangle : x \in \mathrm{F}^{-1}(0)\} \quad \text{and} \quad |\tau\rangle = \sum_{x\in\mathrm{F}^{-1}(0)} |x\rangle.$$

Finally, define

$$A = \sum_{x\in\mathrm{F}^{-1}(0)} |x\rangle\langle w_x|.$$

We now argue that this span program decides F, and analyze its complexity. First, for any $y \in \mathrm{F}^{-1}(1)$, we have $|w_y\rangle \in H(y)$ (by (5.9)) and:

$$A|w_y\rangle = \sum_{x\in\mathrm{F}^{-1}(0)} |x\rangle\langle w_x|w_y\rangle = \sum_{x\in\mathrm{F}^{-1}(0)} |x\rangle = |\tau\rangle,$$

so $|w_y\rangle$ is a positive witness for y, and so

$$\mathcal{W}_+(P) \leq \max_{y\in\mathrm{F}^{-1}(1)} \||w_y\rangle\|^2 .$$

Next, we will show that for any $x \in \mathrm{F}^{-1}(0)$, $|x\rangle$ is a negative witness for x. This follows from $\langle x|\tau\rangle = 1$, and

$$\langle x|A\Pi_{H(x)} = \langle w_x|\Pi_{H(x)} = 0,$$

by (5.10). Thus

$$\mathcal{W}_-(P) \leq \max_{x\in\mathrm{F}^{-1}(0)} \|\langle x|A\|^2 = \max_{x\in\mathrm{F}^{-1}(0)} \|\langle w_x|\|^2 .$$

Thus

$$\mathcal{C}(P) \leq \sqrt{\max_{x\in\mathrm{F}^{-1}(0)} \||w_x\rangle\|^2 \max_{y\in\mathrm{F}^{-1}(1)} \||w_y\rangle\|^2} \leq \max_{x\in D} \||w_x\rangle\|^2 = \mathcal{C}(W). \qquad \square$$

A span program of the form in the proof of Lemma 5.2.9 is called a *canonical form* span program, and by the proof of Lemma 5.2.9 and Exercise 5.2.4, any span program can be converted into canonical form. The optimization problem in (5.8) is essentially just optimizing over all canonical form span programs. In Section 5.3.1, we will see that this optimization version of span programs is not only interesting for its similarity to classical certificate complexity, but also because of its relationship to query *lower bounds*.

Let us compare $\mathsf{W}(\mathrm{F})$ with $\mathsf{C}(\mathrm{F})$. We can relate $\mathsf{C}(\mathrm{F})$ to $\mathsf{D}(\mathrm{F})$ by using a set of certificates for F to design a deterministic algorithm for F whose query complexity depends on their sizes. Similarly, we can use a set of *witnesses* for F to design a *quantum* algorithm for F, via span programs, whose query complexity is precisely (up to constants) their witness complexity, and this is even optimal.

5.3 Lower Bounds

Just as important as finding new quantum algorithms is proving *lower bounds*, which are limitations on how fast a quantum computer can solve a certain problem. These tell us what kinds of algorithms we can possibly hope for, and give us valuable insights into the types of problems for which we can hope for quantum speedups (and how much those speedups could be).

The easiest way to prove a lower bound on the number of steps needed for a quantum computer to solve a particular problem F is to show a lower bound on $\mathsf{Q}(\mathrm{F})$. Unlike in the case of upper bounds

$$\begin{aligned}
\mathsf{C}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i} = \max_{x\in D}\sum_{i=1}^{n} w_{x,i}^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \{0,1\} \\
& \forall (x,y) \in \mathrm{F}^{-1}(b) \times \mathrm{F}^{-1}(1-b),\ \sum_{i:x_i\neq y_i} w_{x,i} \geq 1
\end{aligned}$$

$$\begin{aligned}
\forall \text{ total } \mathrm{F},\ \mathsf{C}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i}^2 = \max_{x\in D}\sum_{i=1}^{n} w_{x,i}^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \{0,1\} \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1.
\end{aligned}$$

$$\begin{aligned}
\mathsf{FC}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i} \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\
& \forall (x,y) \in \mathrm{F}^{-1}(b) \times \mathrm{F}^{-1}(1-b),\ \sum_{i:x_i\neq y_i} w_{x,i} \geq 1.
\end{aligned}$$

$$\begin{aligned}
\mathsf{EC}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i} \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in [0,1] \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1.
\end{aligned}$$

$$\begin{aligned}
\mathsf{CG}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\left(\sum_{i=1}^{n} w_{x,i}\right)^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1.
\end{aligned}$$

$$\begin{aligned}
\mathsf{Adv}^{+}(\mathrm{F}) = \mathsf{W}'(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} w_{x,i}^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ w_{x,i} \in \mathbb{R}_{\geq 0} \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} w_{x,i}w_{y,i} \geq 1. \\
= \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} \||w_{x,i}\rangle\|^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ |w_{x,i}\rangle \in \mathbb{R}^d \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} \langle w_{x,i}|w_{y,i}\rangle \geq 1.
\end{aligned}$$

$$\begin{aligned}
\mathsf{Adv}^{\pm}(\mathrm{F}) = \mathsf{W}(\mathrm{F}) = \text{minimize: } & \max_{x\in D}\sum_{i=1}^{n} \||w_{x,i}\rangle\|^2 \\
\text{subject to: } & \forall x \in D, i \in [n],\ |w_{x,i}\rangle \in \mathbb{R}^d \\
& \forall (x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1),\ \sum_{i:x_i\neq y_i} \langle w_{x,i}|w_{y,i}\rangle = 1.
\end{aligned}$$

Figure 5.1: A summary of optimization problems we have seen so far, including their relationships with Adv and $\mathsf{Adv}^{\pm}$, defined in Section 5.3.1.

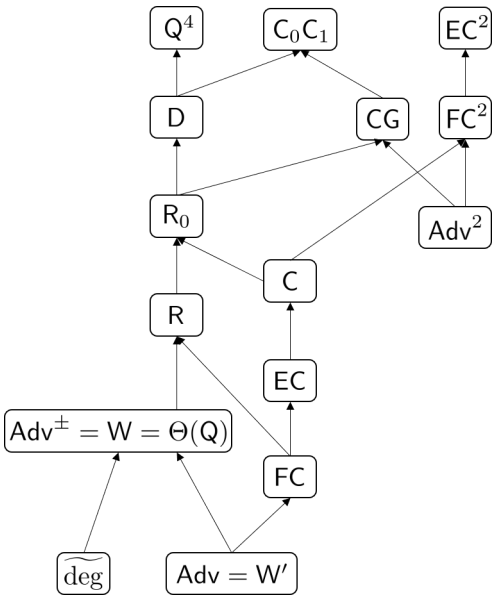

Figure 5.2: Known relationships between various complexity measures introduced here (R_0 is the *zero-error* randomized query complexity) for *total* functions. An arrow from A to B indicates that $A(\mathrm{F}) \leq B(\mathrm{F})$ for all total F. The larger quantities are at the top of the figure, and the smallest at the bottom. Figure based on [CGL+23, Figure 1].

on $\mathsf{Q}(\mathrm{F})$, lower bounds on $\mathsf{Q}(\mathrm{F})$ are actual lower bounds on the time needed by a quantum computer to solve F (with bounded error).

The first systematic method for proving quantum query lower bounds was the *polynomial method*, developed right here in Amsterdam [BBC+01], although there were quantum query lower bounds proven before this using more ad hoc methods. We will discuss the polynomial method briefly in Section 5.3.2. Later, the quantum *adversary bound* was proposed. We will discuss the adversary bound in detail in Section 5.3.1.

Progress Measures Before we begin discussing the adversary bound, we outline a general technique for proving lower bounds, which is the use of a *progress measure*. A progress measure for an algorithm $\mathcal{A}$, S_t is a function of the algorithm (we leave this dependence implicit) and the number of queries[5] made by the algorithm so far. We want to show that:

1. For any algorithm, before any queries have been made, no progress has been made, so $S_0 = 0$;
2. For any algorithm, after its final query, there must have been a lot of progress if we want to get the right answer, so $S_T \geq B$ for some $B > 0$;
3. For any algorithm, a single query can increase the progress by at most δ, for some $\delta > 0$ – that is, for all t, $S_{t+1} - S_t \leq \delta$.

If we can show statements of this form, then they imply that any algorithm must make at least B/δ queries – that's the only way to possibly get from 0 to B progress.

Example 5.3.1. Consider any deterministic algorithm for $\textsc{Or}_n$. Let S_t be the number of distinct i such that the algorithm has queried x_i, after t queries. Then of course $S_0 = 0$. If the algorithm is to succeed in the worst case, then we must have $S_T = n$, because otherwise, there is some i such that the algorithm has not queried x_i, meaning the algorithm cannot distinguish between the all-0s input, and the input that is 0 everywhere except $x_i = 1$, and so it cannot possibly be correct on both of those inputs. Finally, it is easy to see that $S_{t+1} - S_t \leq 1$ – we can't increase the number of distinct indices

[5] Or other complexity we're trying to lower bound.

queried by more than one with a single query. Thus $D(\text{OR}_n) \geq n$.

Of course, the same argument works if we don't require S_0 to be 0, but just smaller than some bound A – we then get a lower bound of $(B - A)/\delta$ queries; and a similar argument holds if S_0 begins high, and must decrease by S_T; concretely, if we can show:

1. For any algorithm, before any queries have been made, $S_0 \geq A$, for some bound A;
2. For any algorithm, after its final query, $S_T \leq B$ for some $B < A$;
3. For any algorithm, a single query can decrease the progress measure by at most δ, for some $\delta > 0$ – that is, for all t, $S_t - S_{t+1} \leq \delta$;

then we can give a lower bound on the query complexity of $(A - B)/\delta$.

Example 5.3.2. We will again consider deterministic algorithms. Let $\mathrm{F} : D \to \{0,1\}$ be any problem, with $D \subseteq \{0,1\}^n$, and fix a set $R \subseteq \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$. For each $i \in [n]$, let

$$\Delta_i = \{(x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1) : x_i \neq y_i\}$$

be the set pairs $(x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$ that are distinguished by querying the i-th bit. Let S_t be the number of pairs $(x,y) \in R$ for which we have not yet queried some i such that $(x,y) \in \Delta_i$. Then we start off having queried nothing, so $S_0 = |R|$. By the end of any algorithm that decides F, we must have $S_T = 0$. Otherwise, if there is some $(x,y) \in R$ such that we have not queried any i such that $x_i \neq y_i$, then our algorithm cannot distinguish them, so it outputs either $\mathrm{F}(x) = 0 \neq \mathrm{F}(y)$ on both inputs, so it is wrong on y, or it outputs 1 on both inputs, so it is wrong on x. Finally, we have $S_t - S_{t+1} \leq \max_i |R \cap \Delta_i|$. Thus we have

$$\mathsf{D}(\mathrm{F}) \geq \frac{|R|}{\max_{i \in [n]} |R \cap \Delta_i|}. \tag{5.11}$$

The above examples gives a method for proving lower bounds on $\mathsf{D}(F)$, by exhibiting a set $R \subseteq F^{-1}(0) \times F^{-1}(1)$ for which the right-hand side of (5.11) is large.

Example 5.3.3. Let $\mathrm{F} : \{0,1\}^{n^2} \to \{0,1\}$ be the function AND of ORs, defined

$$\mathrm{F}(x^{(1)}, \ldots, x^{(n)}) = \bigwedge_{i=1}^{n} \bigvee_{j=1}^{n} x_j^{(i)},$$

where each $x^{(i)} \in \{0,1\}^n$. We will show a lower bound on $\mathsf{D}(\mathrm{F})$ by exhibiting a set R as in Example 5.3.2. Let R be defined

$$R = \{(x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1) : \forall i \in [n], |y^{(i)}| = 1; \exists i' \in [n], |x^{(i')}| = 0; \forall i \in [n] \setminus \{i'\}, x^{(i)} = y^{(i)}\}.$$

Then we leave it as an exercise to show that $|R| = n^{n+1}$ and for all $i,j \in [n]$, $|\Delta_{i,j} \cap R| = n^{n-1}$. Thus, we have proven the lower bound $\mathsf{D}(\mathrm{F}) \geq n^2$.

5.3.1 The Adversary Bound

The Unweighted Adversary

The first version of the quantum adversary bound was introduced in a paper entitled "Quantum lower bounds by quantum arguments" [Amb02]. A key feature of this method is that there is a nice quantum intuition behind why it works, which we will try to convey.

Entanglement View Suppose you have a quantum algorithm that decides F with bounded error, consisting of unitaries $U_1(x), \ldots, U_T(x)$ on a space

$$H_{\mathcal{A}} = \text{span}\{|i\rangle|z\rangle : i \in [n] \cup \{0\}, z \in \mathcal{Z}\},$$

where for any odd t, $U_t = U_t(x)$ is input independent, and for every even t, $U_t(x) = \mathcal{O}_x^{\pm}$ is a phase query. Then, in principle, it would be possible to run this algorithm on a superposition of different inputs stored in some additional register $\mathcal{I}$, replacing each call to $\mathcal{O}_x^{\pm}$ with a call to $\mathcal{O}$ acting on $H_{\mathcal{I}} \otimes H_{\mathcal{A}}$ as:

$$\mathcal{O}|x\rangle_{\mathcal{I}}|i,z\rangle_{\mathcal{A}} = (-1)^{x_i}|x\rangle_{\mathcal{I}}|i,z\rangle_{\mathcal{A}}.$$

Such an algorithm would begin in a state:

$$\sum_x \alpha_x|x\rangle_{\mathcal{I}}|\psi_0\rangle_{\mathcal{A}}, \tag{5.12}$$

where $|\psi_0\rangle$ is the input-independent initial state of the algorithm. The state in (5.12) is a *separable* state – there is no entanglement between $\mathcal{I}$ and $\mathcal{A}$. Said another way, if you trace out the register $\mathcal{A}$, you have the pure state $\sum_x \alpha_x|x\rangle$ in register $\mathcal{I}$. However, by the end of the algorithm, there must be entanglement between the two registers, assuming $\sum_x \alpha_x|x\rangle$ has non-zero overlap with both 0-inputs and 1-inputs to F. That's because in order for the algorithm to be correct, the final state must depend on x. Concretely, fix sets $X \subseteq \mathrm{F}^{-1}(0)$ and $Y \subseteq \mathrm{F}^{-1}(1)$ of "hard" inputs, and let

$$\sum_x \alpha_x|x\rangle_{\mathcal{I}} = \frac{1}{\sqrt{2}} \underbrace{\sum_{x\in X} \frac{1}{\sqrt{|X|}}|x\rangle_{\mathcal{I}}}_{=:|\phi_0\rangle} + \frac{1}{\sqrt{2}} \underbrace{\sum_{y\in Y} \frac{1}{\sqrt{|Y|}}|y\rangle_{\mathcal{I}}}_{=:|\phi_1\rangle}, \tag{5.13}$$

and suppose the algorithm ends in the state $|0, \mathrm{F}(x)\rangle$ (with no error, for simplicity) on input x. Then the final state is

$$\sum_x \alpha_x|x\rangle_{\mathcal{I}}|0,\mathrm{F}(x)\rangle_{\mathcal{A}} = \frac{1}{\sqrt{2}}|\phi_0\rangle_{\mathcal{I}}|0,0\rangle_{\mathcal{A}} + \frac{1}{\sqrt{2}}|\phi_1\rangle_{\mathcal{I}}|0,1\rangle_{\mathcal{A}}.$$

This is a maximally entangled state – not if we consider entanglement between the systems $\mathcal{I}$ and $\mathcal{A}$, but if we consider entanglement between the single-qubit subsystem of $\mathcal{I}$, $\text{span}\{|\phi_0\rangle, |\phi_1\rangle\}$, and the single-qubit subsystem of $\mathcal{A}$, $\text{span}\{|0,0\rangle, |0,1\rangle\}$; so in particular, if we trace out $\mathcal{A}$, we get a mixed state that is $|\phi_0\rangle$ with probability $1/2$ and $|\phi_1\rangle$ with probability $1/2$. The adversary bound works by upper bounding how much the entanglement between $\mathcal{I}$ and $\mathcal{A}$ can increase with one query, and lower bounding how much entanglement must have amassed by the end of the algorithm in order for the algorithm to be correct, which gives a lower bound on the number of queries needed – that is, it is based on a progress measure that measures the amount of entanglement across $\mathcal{I}$ and $\mathcal{A}$.

Concretely, let $|\psi_0(x)\rangle = |\psi_0\rangle$ be the initial state of the algorithm, which is independent of the input, and for any $t \in [T]$, let $|\psi_t(x)\rangle = U_t|\psi_{t-1}(x)\rangle$ be the state of the algorithm right after the t-th unitary is performed (so for even t, it is the state right after the $t/2$-th query has been performed). Then the state of the whole system, including the input register, after $\lfloor t/2 \rfloor$ queries is:

$$\sum_{x\in D} \alpha_x|x\rangle_{\mathcal{I}}|\psi_t(x)\rangle_{\mathcal{A}}.$$

Tracing out $\mathcal{A}$ gives the mixed state:

$$\rho_{\mathcal{I}}^t := \mathrm{Tr}_{\mathcal{A}}\left[\sum_{x,y\in D} \alpha_x\alpha_y^*|x\rangle\langle y|_{\mathcal{I}} \otimes |\psi_t(x)\rangle\langle\psi_t(y)|_{\mathcal{A}}\right] = \sum_{x,y\in D} \alpha_x\alpha_y^*\langle\psi_t(y)|\psi_t(x)\rangle|x\rangle\langle y|_{\mathcal{I}}.$$

State Divergence View Let's view this from a different angle. Suppose again that the initial superposition is

$$\sum_{x\in X}\frac{1}{\sqrt{2|X|}}|x\rangle_{\mathcal{I}}+\sum_{y\in Y}\frac{1}{\sqrt{2|Y|}}|y\rangle_{\mathcal{I}}$$

for some sets $X\subseteq \mathrm{F}^{-1}(0)$ and $Y\subseteq \mathrm{F}^{-1}(1)$. The entries of $\rho_{\mathcal{I}}^t$ for $(x,y)\in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1)$ are then proportional to

$$\rho_{\mathcal{I}}^t[x,y]=\alpha_x\alpha_y^*\langle\psi_t(x)|\psi_t(y)\rangle\propto\langle\psi_t(x)|\psi_t(y)\rangle.$$

When $t=0$, these are all 1, but by the end of the algorithm, for any $(x,y)\in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1)$, we should have $|\psi_T(x)\rangle$ and $|\psi_T(y)\rangle$ almost orthogonal, because there should be a measurement that outputs 0 on $|\psi_T(x)\rangle$ and 1 on $|\psi_T(y)\rangle$, so $\langle\psi_t(x)|\psi_t(y)\rangle$ must decrease from 1 to sufficiently small as t increases from 0 to T. How small is sufficiently small?

Lemma 5.3.4. *If $|\psi_T(x)\rangle$ and $|\psi_T(y)\rangle$ are the final states of an algorithm that decides* F *with bounded error ε, on inputs x and y, respectively, and if $\mathrm{F}(x)\neq\mathrm{F}(y)$, then*

$$|\langle\psi_T(x)|\psi_T(y)\rangle|\leq 2\sqrt{\varepsilon(1-\varepsilon)}.$$

Proof. If the algorithm has success probability $1-\varepsilon$, then there is a measurement that outputs $\mathrm{F}(x)$ when applied to $|\psi_T(x)\rangle$ with probability at least $1-\varepsilon$, and that measurement outputs something else with probability at least $1-\varepsilon$ when applied to $|\psi_T(y)\rangle$, meaning there is some orthogonal projector Π (corresponding to the output $\mathrm{F}(x)$) such that:

$$\|\Pi|\psi_T(x)\rangle\|^2\geq 1-\varepsilon,\quad\text{and}\quad\|(I-\Pi)|\psi_T(y)\rangle\|^2\geq 1-\varepsilon, \tag{5.14}$$

or equivalently:

$$\|(I-\Pi)|\psi_T(x)\rangle\|^2\leq \varepsilon,\quad\text{and}\quad\|\Pi|\psi_T(y)\rangle\|^2\leq \varepsilon. \tag{5.15}$$

Then we have:

$$\begin{aligned}
&|\langle\psi_T(x)|\psi_T(y)\rangle| \\
&\leq |\langle\psi_T(x)|\Pi|\psi_T(y)\rangle|+|\langle\psi_T(x)|(I-\Pi)|\psi_T(y)\rangle| && \text{by the triangle ineq.}\\
&= |\langle\psi_T(x)|\Pi\Pi|\psi_T(y)\rangle|+|\langle\psi_T(x)|(I-\Pi)(I-\Pi)|\psi_T(y)\rangle| && \text{since } \Pi,\ I-\Pi \text{ are proj.}\\
&\leq \|\Pi|\psi_T(x)\rangle\|\,\|\Pi|\psi_T(y)\rangle\|+\|(I-\Pi)|\psi_T(x)\rangle\|\,\|(I-\Pi)|\psi_T(y)\rangle\| && \text{by Cauchy-Schwarz}\\
&\leq 2\sqrt{\varepsilon(1-\varepsilon)} && \text{by (5.14) and (5.15).}
\end{aligned}$$

□

For any pair of inputs $(x,y)\in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1)$, it is easy to distinguish x and y with one query – just query an index where they are not the same. The difficulty is that the algorithm must decrease $|\langle\psi_t(x)|\psi_t(y)\rangle|$ for *all* $(x,y)\in \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1)$. It can simplify things to restrict our attention to some hard to distinguish set of pairs $R\subseteq \mathrm{F}^{-1}(0)\times \mathrm{F}^{-1}(1)$. Then we will define, as our first progress measure:

$$S_t:=\sum_{(x,y)\in R}|\rho_{\mathcal{I}}^t[x,y]|=\sum_{(x,y)\in R}\frac{|\langle\psi_t(x)|\psi_t(y)\rangle|}{2\sqrt{|X||Y|}}, \tag{5.16}$$

[5]

still assuming our initial superposition over inputs is as in (5.13)[6]. We can immediately see that:

$$S_0=\sum_{(x,y)\in R}\frac{|\langle\psi_0(x)|\psi_0(y)\rangle|}{2\sqrt{|X||Y|}}=\sum_{(x,y)\in R}\frac{|\langle\psi_0|\psi_0\rangle|}{2\sqrt{|X||Y|}}=\frac{|R|}{2\sqrt{|X||Y|}}, \tag{5.17}$$

and as a corollary to Lemma 5.3.4, any algorithm that decides F with bounded error ε must have:

$$S_T = \sum_{(x,y)\in R} \frac{|\langle\psi_T(x)|\psi_T(y)\rangle|}{2\sqrt{|X||Y|}} \leq \sum_{(x,y)\in R} \frac{2\sqrt{\varepsilon(1-\varepsilon)}}{2\sqrt{|X||Y|}} = \frac{\sqrt{\varepsilon(1-\varepsilon)}|R|}{\sqrt{|X||Y|}}. \tag{5.18}$$

For example, if $\varepsilon = 1/3$, we must have S_0 decrease from $|R|/(2\sqrt{|X||Y|})$ to

$$S_T \leq \frac{\sqrt{2}}{3}\frac{|R|}{\sqrt{|X||Y|}} \approx .94\frac{|R|}{2\sqrt{|X||Y|}}.$$

The final thing we need for lower bounds on quantum query complexity is an upper bound on the amount that progress can increase with a single query. This is the most difficult part. We give such a bound in the proof of the following theorem.

Theorem 5.3.5 ([Amb02])**.** *Fix any problem* $\mathrm{F} : D \to \{0,1\}$ *with* $D \subseteq \{0,1\}^n$*. Let* $X \subseteq \mathrm{F}^{-1}(0)$*,* $Y \subseteq \mathrm{F}^{-1}(1)$ *and* $R \subseteq X \times Y$ *be such that:*

1. *For all* $x \in X$*,* $|\{y \in Y : (x,y) \in R\}| \geq m$*.*
2. *For all* $y \in Y$*,* $|\{x \in X : (x,y) \in R\}| \geq m'$*.*
3. *For all* $x \in X$*,* $i \in [n]$*,* $|\{y \in Y : (x,y) \in R, x_i \neq y_i\}| \leq \ell$*.*
4. *For all* $y \in Y$*,* $i \in [n]$*,* $|\{x \in X : (x,y) \in R, x_i \neq y_i\}| \leq \ell'$*.*

Then $\mathrm{Q}(\mathrm{F}) = \Omega\left(\sqrt{\frac{mm'}{\ell\ell'}}\right)$*.*

Proof. By (5.17) and (5.18), we have

$$S_0 - S_T \geq (1 - 2\sqrt{\varepsilon(1-\varepsilon)})\frac{|R|}{2\sqrt{|X||Y|}}. \tag{5.19}$$

Note that $|R| = \sum_{x\in X} |\{y \in Y : (x,y) \in R\}|$ is at least $|X|m$, and similarly, $|R|$ is at least $|Y|m'$. We have

$$|R| \geq \max\{|X|m, |Y|m'\} \geq \sqrt{|X|m|Y|m'}. \tag{5.20}$$

Thus, (5.19) and (5.20) combine to give:

$$S_0 - S_T \geq (1 - 2\sqrt{\varepsilon(1-\varepsilon)})\frac{\sqrt{mm'}}{2}.$$

We will complete the proof by showing that for all $t \in \{0,\ldots,T-1\}$,

$$S_t - S_{t+1} \leq \sqrt{\ell\ell'}, \tag{5.21}$$

from which it follows that we must have

$$\frac{1 - 2\sqrt{\varepsilon(1-\varepsilon)}}{2}\sqrt{mm'} \leq S_0 - S_T = \sum_{t=0}^{T-1}(S_t - S_{t+1}) \leq T\sqrt{\ell\ell'}.$$

[6] We could just as well have defined the progress measure without the $2\sqrt{|X||Y|}$ in the denominator, but we leave the $1/2\sqrt{|X||Y|} = \alpha_x\alpha_y^*$ for consistency with later definitions.

Since $\lfloor T/2 \rfloor$ is the number of queries made by the algorithm, and any bounded error quantum algorithm for F must have

$$T \geq \frac{1 - 2\sqrt{\varepsilon(1-\varepsilon)}}{2}\sqrt{\frac{mm'}{\ell\ell'}} = \Omega\left(\sqrt{\frac{mm'}{\ell\ell'}}\right),$$

$\mathsf{Q}(\mathrm{F}) = \Omega\left(\sqrt{\frac{mm'}{\ell\ell'}}\right)$.

We now proceed with the proof of (5.21). We have:

$$\begin{aligned} S_t - S_{t+1} &= \sum_{(x,y)\in R} \frac{1}{2\sqrt{|X||Y|}} \left(|\langle\psi_t(x)|\psi_t(y)\rangle| - |\langle\psi_{t+1}(x)|\psi_{t+1}(y)\rangle|\right) \\ &= \sum_{(x,y)\in R} \frac{1}{2\sqrt{|X||Y|}} \left(|\langle\psi_t(x)|\psi_t(y)\rangle| - |\langle\psi_t(x)|U_{t+1}(x)^\dagger U_{t+1}(y)|\psi_t(y)\rangle|\right). \end{aligned} \tag{5.22}$$

If t is even, $U_{t+1}(x) = U_{t+1}(y)$ is an input-independent unitary, so $\langle\psi_t(x)|U_{t+1}(x)^\dagger U_{t+1}(y)|\psi_t(y)\rangle = \langle\psi_t(x)|\psi_t(y)\rangle$, and $S_t - S_{t+1} = 0$. That is: there is no change in the progress function on steps where we don't make a query.

If t is odd, then $U_{t+1}(x) = \mathcal{O}_x^{\pm}$ is a phase query. For $i \in [n]$, let $\Pi_i = |i\rangle\langle i| \otimes I$ be the orthogonal projector onto $|i\rangle$ in the query register. Let $\Pi_x = \sum_{i:x_i=1} \Pi_i$. Then we can express $\mathcal{O}_x^{\pm}$ as $\mathcal{O}_x^{\pm} = I - 2\Pi_x$, so:

$$\begin{aligned} U_{t+1}(x)^\dagger U_{t+1}(y) = \mathcal{O}_x^{\pm}\mathcal{O}_y^{\pm} &= I - 2\Pi_x - 2\Pi_y + 4\Pi_x\Pi_y \\ &= I - 2(\Pi_x - \Pi_x\Pi_y + \Pi_y - \Pi_x\Pi_y) = I - 2(\Pi_x(I-\Pi_y) + (I-\Pi_x)\Pi_y) \\ &= I - 2\left(\sum_{i:x_i=1}\Pi_i \sum_{i:y_i=0}\Pi_i + \sum_{i:x_i=0}\Pi_i \sum_{i:y_i=1}\Pi_i\right) \\ &= I - 2\sum_{i:x_i\neq y_i}\Pi_i. \end{aligned}$$

Continuing from (5.22), and using the triangle inequality, we have:

$$\begin{aligned} S_t - S_{t+1} &= \frac{1}{2\sqrt{|X||Y|}} \sum_{(x,y)\in R} \left(|\langle\psi_t(x)|\psi_t(y)\rangle| - \left|\langle\psi_t(x)|\psi_t(y)\rangle - 2\sum_{i:x_i\neq y_i}\langle\psi_t(x)|\Pi_i|\psi_t(y)\rangle\right|\right) \\ &\leq \frac{1}{2\sqrt{|X||Y|}} \sum_{(x,y)\in R} 2\sum_{i:x_i\neq y_i} |\langle\psi_t(x)|\Pi_i|\psi_t(y)\rangle|. \end{aligned}$$

Rearranging terms, and applying Cauchy-Schwarz, we get:

$$S_t - S_{t+1} \leq \frac{1}{\sqrt{|X||Y|}} \sum_{i\in[n]} \sum_{(x,y)\in R: x_i\neq y_i} \|\Pi_i|\psi_t(x)\rangle\| \, \|\Pi_i|\psi_t(y)\rangle\|. \tag{5.23}$$

By Cauchy-Schwarz (again), we have:

$$\begin{aligned} \sum_{(x,y)\in R:x_i\neq y_i} \frac{\|\Pi_i|\psi_t(x)\rangle\|}{\sqrt{|X|}} \frac{\|\Pi_i|\psi_t(y)\rangle\|}{\sqrt{|Y|}} &\leq \sqrt{\sum_{(x,y)\in R:x_i\neq y_i} \frac{\|\Pi_i|\psi_t(x)\rangle\|^2}{|X|}} \sqrt{\sum_{(x,y)\in R:x_i\neq y_i} \frac{\|\Pi_i|\psi_t(y)\rangle\|^2}{|Y|}} \\ &= \sqrt{\sum_{x\in X} \frac{\|\Pi_i|\psi_t(x)\rangle\|^2}{|X|} \sum_{\substack{y:(x,y)\in R:\\ x_i\neq y_i}} 1} \sqrt{\sum_{y\in Y} \frac{\|\Pi_i|\psi_t(y)\rangle\|^2}{|Y|} \sum_{\substack{x:(x,y)\in R:\\ x_i\neq y_i}} 1} \\ &\leq \sqrt{\sum_{x\in X} \frac{\|\Pi_i|\psi_t(x)\rangle\|^2}{|X|}\ell} \sqrt{\sum_{y\in Y} \frac{\|\Pi_i|\psi_t(y)\rangle\|^2}{|Y|}\ell'} \end{aligned}$$

which we can plug into (5.23) to get, using the AMGM inequality, which upper bounds the geometric mean by the arithmetic mean:[7]

$$\begin{aligned} S_t - S_{t+1} &\leq \sum_{i\in[n]} \sqrt{\sum_{x\in X} \frac{\|\Pi_i|\psi_t(x)\rangle\|^2}{|X|}\ell} \sqrt{\sum_{y\in Y} \frac{\|\Pi_i|\psi_t(y)\rangle\|^2}{|Y|}\ell'} \\ &\leq \sqrt{\ell\ell'} \sum_{i\in[n]} \frac{1}{2}\left(\sum_{x\in X} \frac{\|\Pi_i|\psi_t(x)\rangle\|^2}{|X|} + \sum_{y\in Y} \frac{\|\Pi_i|\psi_t(y)\rangle\|^2}{|Y|}\right) \\ &= \frac{\sqrt{\ell\ell'}}{2}\left(\sum_{x\in X} \frac{1}{|X|}\sum_{i\in[n]} \|\Pi_i|\psi_t(x)\rangle\|^2 + \sum_{y\in Y} \frac{1}{|Y|}\sum_{i\in[n]} \|\Pi_i|\psi_t(y)\rangle\|^2\right) \\ &= \frac{\sqrt{\ell\ell'}}{2}\left(\sum_{x\in X} \frac{1}{|X|} + \sum_{y\in Y} \frac{1}{|Y|}\right) = \sqrt{\ell\ell'}, \end{aligned}$$

establishing (5.21). □

Example 5.3.6. Let $\mathrm{F} : \{0,1\}^n \to \{0,1\}$ be defined by $\mathrm{F}(x) = \bigwedge_{i=1}^{\sqrt{n}} \bigvee_{j=1}^{\sqrt{n}} x_{i,j}$ for any $x \in \{0,1\}^n \equiv \{0,1\}^{\sqrt{n}\times\sqrt{n}}$. That is, an input x consists of $\sqrt{n}$ blocks of $\sqrt{n}$ bits each, and we take the OR of each block to get $\sqrt{n}$ bits, and output the AND of those bits. Let Y be the set of all n-bit strings in which each of the $\sqrt{n}$ blocks contains exactly one 1. Let X be the set of strings such that one block is all 0s, and every other block contains exactly one 1. Let

$$R = \{(x,y) \in X \times Y : \exists \text{ a unique } i \text{ such that } x_i \neq y_i\}.$$

Then it is not difficult to convince yourself that $m = m' = \sqrt{n}$ and $\ell = \ell' = 1$ satisfy the conditions of Theorem 5.3.5, and thus $\mathsf{Q}(\mathrm{F}) = \Omega(\sqrt{n})$.

Exercise 5.3.1. *Let* $\textsc{Par}_n(x) = x_1 \oplus \cdots \oplus x_n$. *Use Theorem 5.3.5 to show that* $\mathsf{Q}(\textsc{Par}_n) = \Omega(n)$.

The Weighted Adversary

The simple idea behind the *weighted* adversary [Amb03] is to modify the progress function to give some particularly difficult-to-distinguish pairs $(x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$ more *weight* than others. This was already partially captured in the progress measure defined in (5.16), as we selected just a subset R of $\mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$, but we can take this further by choosing non-negative real numbers $\Gamma_{x,y}$ for each $(x,y) \in \mathrm{F}^{-1}(0) \times \mathrm{F}^{-1}(1)$, and defining the progress measure:

$$S_t := \sum_{(x,y)\in \mathrm{F}^{-1}(0)\times\mathrm{F}^{-1}(1)} \Gamma_{x,y}\rho_{\mathcal{I}}^t[x,y]. \tag{5.24}$$

If we restrict $\Gamma_{x,y}$ to be in $\{0,1\}$, this gives almost the same types of progress measures as (5.16), except we have dropped the absolute values (this is going to make things a bit cleaner). We can view Γ as a matrix in $\mathbb{R}^{D\times D}$ with the additional constraints that Γ is symmetric and $\Gamma_{x,y} = 0$ whenever $\mathrm{F}(x) = \mathrm{F}(y)$, then we have:

$$S_t = \frac{1}{2}\sum_{x,y\in D} \Gamma_{x,y}\rho_{\mathcal{I}}^t[x,y] = \frac{1}{2}\sum_{x,y\in D} \Gamma_{x,y}\alpha_x\alpha_y^*\langle\psi_t(x)|\psi_t(y)\rangle$$

[7] That is, $\sqrt{AB} \leq \frac{A+B}{2}$.

where we recall that the α_x values are the amplitudes on different inputs in the register $\mathcal{I}$, from (5.12). Then similar to (5.17), we have:

$$S_0 = \frac{1}{2}\sum_{x,y\in D}\Gamma_{x,y}\alpha_x\alpha_y^*\langle\psi_0(x)|\psi_0(y)\rangle = \frac{1}{2}\sum_{x,y\in D}\Gamma_{x,y}\alpha_x\alpha_y^* = \frac{1}{2}\langle\alpha|\Gamma|\alpha\rangle, \tag{10}$$

where $|\alpha\rangle = \sum_{x\in D}\alpha_x|x\rangle_{\mathcal{I}}$ is the initial state on $\mathcal{I}$. To maximize S_0, we can assume that $|\alpha\rangle$ is chosen to maximize $\langle\alpha|\Gamma|\alpha\rangle$, so that

$$S_0 = \frac{1}{2}\|\Gamma\|. \tag{11}$$

Above, $\|\Gamma\|$ is the *spectral norm* of Γ, which is equal to the largest absolute value of any of its eigenvalues, since Γ is symmetric.[8] On the other hand, similar to (5.18), we have:

$$S_T = \frac{1}{2}\sum_{x,y\in D}\Gamma_{x,y}\alpha_x\alpha_y^*\langle\psi_T(x)|\psi_T(y)\rangle \leq \frac{1}{2}\sum_{x,y\in D}\Gamma_{x,y}\alpha_x\alpha_y^* 2\sqrt{\varepsilon(1-\varepsilon)} = \sqrt{\varepsilon(1-\varepsilon)}\,\|\Gamma\|. \tag{12}$$

For the final property we need to make this a useful progress measure, we claim:

Claim 5.3.7. *For any $i \in [n]$, let $\Delta_i = \sum_{x,y\in D:x_i\neq y_i}|x\rangle\langle y|$, and let $\circ$ denote the entrywise matrix product: $[A\circ B][x,y] = A[x,y]B[x,y]$. Then for any $t \in \{0,\ldots,T-1\}$, $|S_t - S_{t+1}| \leq \max_{i\in[n]}\|\Gamma\circ\Delta_i\|$.*

We will not prove this claim, but we give some intuition. $\|\Gamma\|$ measures the "progress" in the task of diverging the pairs of state $|\psi_t(x)\rangle, |\psi_t(y)\rangle$ such that $\Gamma_{x,y} \neq 0$ without making any queries. If we query i, then this completely distinguishes all such pairs of states for which $x_i \neq y_i$.

Let $\mathsf{Q}_\varepsilon(\mathrm{F})$ denote the minimum query complexity of any quantum algorithm that computes F with bounded error ε, so $\mathsf{Q}_{1/3}(\mathrm{F}) = \mathsf{Q}(\mathrm{F})$. From (5.25), (5.26), and Claim 5.3.7 we can conclude that

$$\mathsf{Q}_\varepsilon(\mathrm{F}) \geq \frac{S_0 - S_T}{\max_t|S_t - S_{t+1}|} \geq \frac{\left(\frac{1}{2} - \sqrt{\varepsilon(1-\varepsilon)}\right)\|\Gamma\|}{\max_{i\in[n]}\|\Gamma\circ\Delta_i\|}, \tag{13}$$

for any $\varepsilon \in (0, 1/2)$.

Thus, if we define[9]:

$$\begin{aligned}\mathsf{Adv}(\mathrm{F}) = \text{maximize: } & \frac{\|\Gamma\|}{\max_{i\in[n]}\|\Gamma\circ\Delta_i\|} \\ \text{subject to: } & \Gamma \in \mathbb{R}^{D\times D},\ \text{symmetric} \\ & \forall x,y\in D,\ \Gamma_{x,y}\geq 0 \\ & \forall x,y\in D \text{ s.t. } \mathrm{F}(x)=\mathrm{F}(y),\ \Gamma_{x,y}=0,\end{aligned} \tag{5.27}$$

then we have

[8] In general, the spectral norm is defined $\|\Gamma\| = \max_{|u\rangle}\frac{\|\Gamma|u\rangle\|}{\||u\rangle\|}$ (also called the *operator norm*).

[9] For intuition, compare this objective function with (5.11).

$$\mathsf{Q}(\mathrm{F}) = \Omega\left(\mathsf{Adv}(\mathrm{F})\right). \tag{5.28}$$

An important question is then: What is the power of this lower bound technique? Can we always show an *optimal* lower bound on $\mathsf{Q}(\mathrm{F})$, for any F? Said another way, is it the case that $\mathsf{Q}(\mathrm{F}) = O(\mathsf{Adv}(\mathrm{F}))$ for all F? The answer is no. To understand this, we make note of the following claim.

Claim 5.3.8. *For all* F, $\mathsf{Adv}(\mathrm{F}) = \mathsf{W}'(\mathrm{F})$, *where* W' *is as in (5.7).*

We will not prove this claim, but it follows from the fact that the optimization problem in (5.7), which is a semidefinite optimization problem, is the *dual* (see Appendix .1) of the optimization problem in (5.27) (which is also semidefinte), and moreover, these satisfy the conditions of *strong duality*.

You have already shown in Exercise 5.2.2 that $\mathsf{W}'(\mathrm{F}) \leq \sqrt{\mathsf{C}_0(\mathrm{F})\mathsf{C}_1(\mathrm{F})}$ for all total F. In the following exercise, you will show that there exist total problems for which $\sqrt{\mathsf{C}_0(\mathrm{F})\mathsf{C}_1(\mathrm{F})} = o(\mathsf{Q}(\mathrm{F}))$, meaning $\mathsf{Adv}(\mathrm{F}) = o(\mathsf{Q}(\mathrm{F}))$. Thus Adv is not a tight lower bound in general.

Exercise 5.3.2. *Let* ED_n *be the* element distinctness *problem, in which we interpret an input* $x \in \{0,1\}^{2n\log n}$ *as* n *integers* $x_1, \ldots, x_n \in [n^2]$*, and we want to decide if they are all distinct or not. It is known via a quantum walk algorithm and a matching query lower bound using the polynomial method that* $\mathsf{Q}(\mathrm{ED}_n) = \Theta(n^{2/3})$*. Show that* $\sqrt{\mathsf{C}_0(\mathrm{ED}_n)\mathsf{C}_1(\mathrm{ED}_n)} = o(\mathsf{Q}(\mathrm{ED}_n))$.

Negative Weights Make Adversaries Stronger

If we modify the semidefinite optimization problem in (5.27) by removing the constraint that all entries of Γ be positive, we get:

$$\begin{aligned} \mathsf{Adv}^{\pm}(\mathrm{F}) = \text{maximize: } & \frac{\|\Gamma\|}{\max_{i\in[n]} \|\Gamma \circ \Delta_i\|} \\ \text{subject to: } & \Gamma \in \mathbb{R}^{D\times D}, \text{ symmetric} \\ & \forall x, y \in D \text{ s.t. } \mathrm{F}(x) = \mathrm{F}(y),\ \Gamma_{x,y} = 0. \end{aligned} \tag{5.29}$$

While using positive weights makes intuitive sense, it turns out that none of the claims we made about the the progress measure in the previous section relied on the weights being non-negative (although the proof of Claim 5.3.7 becomes somewhat more complicated when weights can be negative), and thus, similar to (5.28), we also have [HLŠ07]:

$$\mathsf{Q}(\mathrm{F}) = \Omega\left(\mathsf{Adv}^{\pm}(\mathrm{F})\right). \tag{5.30}$$

Let us compare (5.29) with the expression for Adv in (5.27). Adv is more constrained than $\mathsf{Adv}^{\pm}$, so it will generally be smaller, making $\mathsf{Adv}^{\pm}$ a (potentially) tighter lower bound on Q. If we take the dual of the optimization problem for $\mathsf{Adv}^{\pm}$, we should get something more constrained than the dual of Adv, which is W' (see (5.7)), and in fact, we get precisely the optimization problem for W in (5.8). It turns out (again, for reasons we will not discuss) that these again satsify strong duality for all F, meaning that [Rei09]:

Claim 5.3.9. *For all problems* F, $\mathsf{Adv}^{\pm}(\mathrm{F}) = \mathsf{W}(\mathrm{F})$.

But we have seen, in Lemma 5.2.9, that any feasible solution for W with objective value T, can be turned into a span program with complexity T, meaning

$$\mathsf{SPC}(\mathrm{F}) = O(\mathsf{W}(\mathrm{F})) = O\left(\mathsf{Adv}^{\pm}(\mathrm{F})\right) = O(\mathsf{Q}(\mathrm{F})),$$

by (5.30). This proves the other direction (that we have not yet seen) of Theorem 4.2.6, and explains how we can have a proof that $\mathsf{SPC}(\mathrm{F}) = O(\mathsf{Q}(\mathrm{F}))$ that is not constructive, since it goes through semidefinite program duality.

We summarize what we have seen with the following.

Theorem 5.3.10. *For all* F*,*

$$\mathsf{Adv}^{\pm}(\mathrm{F}) = \mathsf{W}(\mathrm{F}) = \mathsf{SPC}(\mathrm{F}) = \Theta(\mathsf{Q}(\mathrm{F})).$$

We conclude by remarking that applying the negative weights version of the adversary bound is generally quite difficult. Despite this, there have been some notable successes, including a tight lower bound for the k-sum problem [BŠ13].

Function Composition

Recall from Section 4.3.4 that if $\mathrm{F} \circ \mathrm{G}$ denotes the composed function:

$$(x^{(1)}, \ldots, x^{(n)}) \mapsto \mathrm{F}(\mathrm{G}(x^{(1)}), \ldots, \mathrm{G}(x^{(n)})),$$

then $\mathsf{Q}(\mathrm{F} \circ \mathrm{G}) = O(\mathsf{Q}(\mathrm{F})\mathsf{Q}(\mathrm{G}))$. This is actually tight:

$$\mathsf{Q}(\mathrm{F} \circ \mathrm{G}) = \Theta(\mathsf{Q}(\mathrm{F})\mathsf{Q}(\mathrm{G})),$$

which follows from the fact that

$$\mathsf{Adv}^{\pm}(\mathrm{F} \circ \mathrm{G}) = \mathsf{Adv}^{\pm}(\mathrm{F})\mathsf{Adv}^{\pm}(\mathrm{G}), \tag{5.31}$$

and Theorem 5.3.10. That $\mathsf{Q}(\mathrm{F} \circ \mathrm{G}) = \Omega(\mathsf{Q}(\mathrm{F})\mathsf{Q}(\mathrm{G}))$ is not surprising. However, note that (5.31) is quite strong, since it holds perfectly, without even a constant factor. One implication of this is that if we let

$$\mathrm{F}^k = \underbrace{\mathrm{F} \circ \mathrm{F} \circ \cdots \circ \mathrm{F}}_{k \text{ times}},$$

then

$$\mathsf{Adv}^{\pm}(\mathrm{F}^k) = \mathsf{Adv}^{\pm}(\mathrm{F})^k.$$

This would not hold for non-constant k, even up to constants, if (5.31) only held up to some constant $c > 1$, since there would be a non-constant factor of c^k. So although this implies that

$$\mathsf{Q}(\mathrm{F}^k) = \Theta(\mathsf{Q}(\mathrm{F}))^k,$$

we cannot conclude that $\mathsf{Q}(\mathrm{F}^k) = \Theta(\mathsf{Q}(\mathrm{F})^k)$. In the remainder of this section, we will work through an example that illustrates this subtle difference.

Exercise 5.3.3. *Let* $\textsc{Par}_n$ *denote* n *bit parity:* $\textsc{Par}_n(x) = x_1 \oplus \cdots \oplus x_n$*. Prove that for any even* n*,* $\mathsf{Q}_E(\textsc{Par}_n) \leq \frac{n}{2}$ *where* Q_E *is the* exact *query complexity (that is, your algorithm should work without error). Hint: It is sufficient to look at the* $n = 2$ *case.*

In the above exercise, you worked out an exact quantum algorithm that computes the parity of 2 bits in 1 query. It is easy to show that $\textsc{Par}_n = \textsc{Par}_2^{\log n}$ for any n that is a power of 2. In other words: We can express n-bit parity as a $\log(n)$-depth tree of 2-bit $\oplus$ gates. So it now seems obvious that we can compute $\textsc{Par}_n$ using 1 query, as follows: Run a quantum algorithm that computes $\textsc{Par}_n$ using one query to an oracle for $\textsc{Par}_{n/2}$, which itself uses 1 query to $\textsc{Par}_{n/4}$, etc. until finally $\textsc{Par}_2$ makes one query to the input. This would appear to imply $\mathsf{Q}_E(\textsc{Par}_n) \leq 1$ from which it would follow that $\mathsf{Q}(\textsc{Par}_n) \leq 1$, however, there is a subtle but crucial flaw to this argument!

The algorithm you derived for $\textsc{Par}_2$ in the exercise above did not implement the map

$$|0\rangle \mapsto |x_1 \oplus x_2\rangle$$

using a single query to $\mathcal{O}_x$. Instead, it probably did something like the following:

$$\mathcal{A}(x) : |0\rangle \mapsto (-1)^{x_1}|x_1 \oplus x_2\rangle.$$

From the state $(-1)^{x_1}|x_1 \oplus x_2\rangle$, we can learn $x_1 \oplus x_2$ with certainty, but it is not a perfect query of the bit $|x_1 \oplus x_2\rangle$, which you will see if you try to use $\mathcal{A}(\cdot)$ as an oracle to compute the parity of $(x_1 \oplus x_2) \oplus (x_3 \oplus x_4)$.

There is a standard trick we can use to get rid of the unwanted phase $(-1)^{x_1}$: we simply run $\mathcal{A}$, copy out the answer, and then uncompute $\mathcal{A}$:

$$|0\rangle|0\rangle \overset{\mathcal{A}(x)\otimes I}{\mapsto} (-1)^{x_1}|x_1 \oplus x_2\rangle|0\rangle \overset{\mathsf{CNOT}}{\mapsto} (-1)^{x_1}|x_1 \oplus x_2\rangle|x_1 \oplus x_2\rangle \overset{\mathcal{A}(x)^\dagger\otimes I}{\mapsto} |0\rangle|x_1 \oplus x_2\rangle.$$

Since $\mathcal{A}$ is exact, this works perfectly. However, we have doubled the query complexity from 1 to 2, so if we try to compose this new algorithm for $\textsc{Par}_2$ to depth $\log(n)$ (which will work) we get an algorithm for $\textsc{Par}_n$ that uses $2^{\log(n)} = n$ queries, and it turns out we can't improve this by more than a constant factor.

Exercise 5.3.4. *Use the fact that* $\mathsf{Q}(\textsc{Par}_n) = \Theta(n)$ *to show that* $\mathsf{Adv}^{\pm}(\textsc{Par}_n) = n$.

5.3.2 Polynomial Lower Bounds

This section closely follows [dW19, Section 11.2]. An n-variate multilinear polynomial $p : \mathbb{C}^n \to \mathbb{C}$ is a function of the form:

$$p(x_1, \ldots, x_n) = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i,$$

for some coefficients $a_S \in \mathbb{C}$. The degree of p is $\deg(p) = \max\{|S| : a_S \neq 0\}$. Every function $\mathrm{F} : \{0,1\}^n \to \mathbb{C}$ has a unique representation as an n-variate multilinear polynomial, and we let $\deg(\mathrm{F})$ denote the degree of this polynomial.

Example 5.3.11. The polynomial

$$p(x_1, \ldots, x_n) = x_1 \ldots x_n$$

represents the n-bit AND, $\textsc{And}_n$. The polynomial

$$p(x_1, \ldots, x_n) = 1 - (1 - x_1) \ldots (1 - x_n)$$

represents the n-bit OR, $\textsc{Or}_n$.

If we are happy with a polynomial that *approximates* F, then the representation is no longer unique, and might have significantly smaller degree. We call the smallest degree of any polynomial ε-approximating F (we have not defined precisely what we mean by this) the *approximate degree*, $\widetilde{\deg}_\varepsilon(\mathrm{F})$.

Theorem 5.3.12 ([BBC+01])**.** *Let $U_1, \ldots, U_T$ be a quantum algorithm where for each odd t, $U_t(x) = U_t$ is independent of the input, and for each even t, $U_t(x) = \mathcal{O}_x^{\pm}$. Suppose without loss of generality that the final measurement $\mathcal{M}$ is a computational basis measurement. Let $|\psi_T(x)\rangle = \sum_{i,z} \alpha_{i,z}(x)|i,z\rangle$ be the state of a quantum algorithm after the last unitary $U_T(x)$ has been applied. Then for every i and z, $\alpha_{i,z}(x)$ is a multilinear polynomial of degree at most $\lfloor T/2 \rfloor$. It follows that the acceptance probability, which is the sum of $|\alpha_{i,z}(x)|^2$ over some subset of the i, z, is a multilinear polynomial of degree at most $2\lfloor T/2 \rfloor$.*

We will give a sketch of the proof. For details, see [dW19, Section 11.2] or [BBC+01].

Proof Sketch: The proof is by induction on T. For $T = 0$, Note that the amplitudes in $|\psi_0\rangle$ are independent of x, so they are constant functions of x, which are degree 0 polynomials.

Suppose $T > 0$ is odd. In that case, $U_T(x) = U_T$ is independent of the input. By the induction hypothesis,

$$|\psi_{T-1}(x)\rangle = \sum_{i,z} \alpha_{i,z}(x)|i,z\rangle$$

for some multilinear polynomials of degree $\lfloor (T-1)/2 \rfloor = (T-1)/2$. Then the amplitudes of $|\psi_T(x)\rangle = U_T|\psi_{T-1}(x)\rangle$ are just linear functions of the amplitudes of $|\psi_{T-1}(x)\rangle$, and so they are also polynomials of degree at most $(T-1)/2 = \lfloor T/2 \rfloor$.

Suppose $T > 0$ is even. In that case, $U_T(x) = \mathcal{O}_x^{\pm}$. Thus:

$$|\psi_T(x)\rangle = \mathcal{O}_x^{\pm}|\psi_{T-1}(x)\rangle = \sum_{i,z} \alpha_{i,z}(x)(-1)^{x_i}|i,z\rangle = \sum_{i,z} \alpha_{i,z}(x)(1-2x_i)|i,z\rangle.$$

Since $\alpha_{i,z}(x)$ is a multilinear polynomial of degree $\lfloor (T-1)/2 \rfloor = T/2 - 1$ by the induction hypothesis, $\alpha_{i,z}(x)(1-2x_i)$ is a polynomial of degree at most $(T/2-1)+1 = T/2 = \lfloor T/2 \rfloor$. It is multilinear because $x_i^2 = x_i$ for all i. □

The other important observation is that for any algorithm that computes a problem F with bounded error ε, the polynomial p that is the acceptance probability of the algorithm is an ε-approximating polynomial for F, so it must be the case that $T \geq \widetilde{\deg}(\mathrm{F})$. The technique is then to prove a lower bound on $\widetilde{\deg}_{1/3}(\mathrm{F})$, which is then a lower bound on $\mathsf{Q}(\mathrm{F})$.

An important consequence of this technique was to show that for all *total* F,

$$\mathsf{D}(\mathrm{F}) = O\left(\mathsf{Q}(\mathrm{F})^6\right),$$

meaning we cannot get better than polynomial speedups over even *deterministic algorithms* if we restrict to total functions [BBC+01]. This result has since been strengthened to [ABDK+21]

$$\mathsf{D}(\mathrm{F}) = O\left(\mathsf{Q}(\mathrm{F})^4\right). \tag{5.32}$$

This is tight, as there exists a total function F such that $\mathsf{D}(\mathrm{F}) = \widetilde{\Omega}(\mathsf{Q}(\mathrm{F})^4)$ [ABB+17]. Note that (5.32) also implies that $\mathsf{R}(\mathrm{F}) = O(\mathsf{Q}(\mathrm{F})^4)$ for all total functions. It is an open question whether this can be improved, and by how much. For a while it was conjectured that $\mathsf{R}(\mathrm{F}) = O(\mathsf{Q}(\mathrm{F})^2)$ for all total functions (Grover's algorithm would be a counter example to any stronger conjecture), but better separations were found, with the current best being a total function F such that $\mathsf{R}(\mathrm{F}) \geq \mathsf{Q}(\mathrm{F})^{2.66}$ [Tal20].

For partial functions, we can do much better, as we will see shortly.

5.4 Forrelation: A Maximal Separation

Until now we have talked about the limits of quantum computers, and techniques for proving such limits. What can we say about their power? Concretely, what is the largest possible separation between Q and R for *any* F? We will now address this question using a problem called Forrelation. In fact, there is a version of this problem for any constant $k \geq 2$. First, for functions $f_1, \ldots, f_k : \{0,1\}^n \to \{0,1\}$, define:

$$\Phi_{f_1,\ldots,f_k} := \frac{1}{2^{(k+1)n/2}} \sum_{x_1,\ldots,x_k \in \{0,1\}^n} f_1(x_1)(-1)^{x_1 \cdot x_2} f_2(x_2)(-1)^{x_2 \cdot x_3} \ldots (-1)^{x_{k-1} \cdot x_k} f_k(x_k),$$

where $x \cdot x' = \sum_{i=1}^n x_i x'_i$. Then k-fold Forrelation is the following problem.

Problem: $\textsc{Forr}_{k,n}$

Input: Oracles $\mathcal{O}_{f_1}, \ldots, \mathcal{O}_{f_k}$ such that either $|\Phi_{f_1,\ldots,f_k}| \leq \frac{1}{100}$ or $|\Phi_{f_1,\ldots,f_k}| \geq \frac{3}{5}$
Output: 0 if $|\Phi_{f_1,\ldots,f_k}| \leq \frac{1}{100}$, and 1 if $|\Phi_{f_1,\ldots,f_k}| \geq \frac{3}{5}$

This is a promise problem (i.e. a partial function), since the condition that either $|\Phi_{f_1,\ldots,f_k}| \leq \frac{1}{100}$ or $|\Phi_{f_1,\ldots,f_k}| \geq \frac{3}{5}$ excludes many inputs. If it were a total problem, there would be no hope of getting a superpolynomial separation between Q and R. It is not a problem that comes up in any known practical scenario, but instead it is designed to be as easy as possible for quantum computers, while taking advantage of their power as much as possible.

Exercise 5.4.1. *Describe a quantum algorithm that decides* $\textsc{Forr}_{k,n}$ *with bounded error using a single call to each oracle, and prove that it works. (Hint: It's basically the simplest quantum algorithm imagineable).*

In fact, it is possible for a quantum algorithm to decide $\textsc{Forr}_{k,n}$ with bounded error using $\lceil k/2 \rceil$ queries. While this may seem impossible since each of the k oracles must be involved in the algorithm somehow, we can actually consider the input as a *single* oracle

$$\mathcal{O} = \sum_{i=1}^{k} |i\rangle\langle i| \otimes \mathcal{O}_{f_i}.$$

If we think of the inputs f_i as strings, then this is essentially just viewing the full input as their concatenation, and indexing into the long string by a pair of indices $i \in [k]$ and $x \in \{0,1\}^n$. Thus, we can make fewer than k calls to $\mathcal{O}$, on superpositions of inputs, to get an algorithm that depends non-trivially on all f_i. This algorithm is also not particularly complicated. The much more difficult thing is showing a classical lower bound. We state the following without proof, where $N = 2^n$.

Theorem 5.4.1. *For any* $k \geq 2$, $\mathsf{Q}(\textsc{Forr}_k) \leq \lceil k/2 \rceil$ *[AA15], and* $\mathsf{R}(\textsc{Forr}_k) = \Omega(N^{1-1/k})$ *[BS21].*

This is close to the best possible separation we could hope for. Assymptotically, $\mathsf{Q}(\textsc{Forr}_{k,n}) = O(1)$ is as small as possible, and since the input size is $k2^n = kN = O(N)$, the strongest classical lower bound possible would be $\Omega(N)$, and we can get arbitrarily close to that by increasing the constant k. So this is almost the best we could hope for, but it's a little worse. Can we do better? Nope [BGGS21]. It turns out that any quantum algorithm making t queries to a $O(N)$-bit input can be simulated by a classical algorithm making $O(N^{1-1/(2t)})$ queries to the input, meaning this is, in fact, the best possible separation.

Bibliography

[AA15] Scott Aaronson and Andris Ambainis. Forrelation: A problem that optimally separates quantum from classical computing. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC)*, pages 307–316, 2015. arXiv: 1411.5729 22

[ABB+17] Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *Journal of the ACM*, 64:1–24, 2017. 21

[ABDK+21] Scott Aaronson, Shalev Ben-David, Robin Kothari, Shravas Rao, and Avishay Tal. Degree vs. approximate degree and quantum implications of huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC 2021)*, page 1330–1342. Association for Computing Machinery, 2021. 21

[Amb02] Andris Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64(4):750–767, 2002. Earlier version in STOC'00. arXiv: quant-ph/0002066 11, 14

[Amb03] Andris Ambainis. Polynomial degree vs. quantum query complexity. pages 230–239, 2003. arXiv: 0305028 16

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. Earlier version in FOCS'98. arXiv: quant-ph/9802049 10, 20, 21

[BGGS21] Sergey Bravyi, David Gosset, Daniel Grier, and Luke Schaeffer. Classical algorithms for forrelation. arXiv: 2102.06963, 2021. 22

[BŠ13] Aleksandrs Belovs and Robert Špalek. Adversary lower bound for the k-sum problem. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 323–328, 2013. arXiv: 1206.6528 19

[BS21] Nikhil Bansal and Makrand Sinha. k-forrelation optimally separates quantum and classical query complexity. In *Proceedings of the 53rd ACM Symposium on the Theory of Computing (STOC)*, pages 1303–1316, 2021. 22

[CGL+23] Sourav Chakraborty, Anna Gál, Sophie Laplante, Rajat Mittal, and Anupa Sunny. Certificate games. In *Proceedings of the 14th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 32:1–32:24, 2023. arXiv: 2211.03396 5, 10

[HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th ACM Symposium on the Theory of Computing (STOC)*, pages 526–535, 2007. arXiv: quant-ph/0611054 18

[JKK+20] Rahul Jain, Hartmut Klauck, Srijita Kundu, Troy Lee, Miklos Santha, Swagato Sanyal, and Jevgenijs Vihrovs. Quadratically tight relations for randomized query complexity. *Theory of Computing Systems*, 64:101–119, 2020. arXiv: 1708.00822 5

[KW90] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require superlogarithmic depth. *SIAM Journal on Discrete Mathematics*, 3:255–265, 1990. 5

[Mir23] Piotr Mironowicz. Semi-definite programming and quantum information. arXiv: 2306.16560, 2023. 24

[Rei09] Ben W. Reichardt. Span programs and quantum query complexity: The general adversary bound is nearly tight for every Boolean function. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 544–551, 2009. arXiv: 0904.2759 18

[Tal20] Avishay Tal. Towards Optimal Separations between Quantum and Randomized Query Complexities . In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS 2020)*, pages 228–239. IEEE Computer Society, 2020. 21

[dW19] Ronald de Wolf. Quantum computing lecture notes. arXiv: 1907.09415v5, 2019. 20

.1 Linear and Semidefinite Optimization

An optimization problem consists of an *objective function* on some variables, and a set of *constraints* on those variables. The value of the problem is the minimum or maximum value the objective function may take, over all settings of the variables such that all constraints are satisfied – such a setting of the variables is called a *feasible solution*.

In a *linear* optimization problem, the objective function is a linear function in the variables, and the constraints are equalities or inequalities of linear functions. For example:

$$\begin{aligned} \text{minimize: } & \sum_{i=1}^{k} a_i z_i \\ \text{subject to: } & \forall i \in [k],\ z_i \in \mathbb{R} \\ & \forall j \in [m],\ \sum_{i \in [k]} b_{i,j} z_i \geq c_j \end{aligned} \tag{33}$$

where $\{a_i\}_{i\in[k]}$, $\{b_{i,j}\}_{i\in[k],j\in[m]}$ and $\{c_j\}_{j\in[m]}$ are some fixed sets of scalars.

A *semidefinite* optimization problem is so-called because of its *semidefinite constraints*. Rather than explaining what this means, we give the canonical form in which any semidefinite minimization problem can be written. For $n \times n$ matrices A and B, define

$$\langle A, B \rangle := \sum_{i,j \in [n]} A_{i,j} B_{i,j}.$$

Then for $n \times n$ matrices A, $\{B_j\}_{j\in[m]}$ and scalars $\{c_j\}_{j\in[m]}$, the following is a semidefinite optimization problem:

$$\begin{aligned} \text{minimize: } & \langle A, Z \rangle \\ \text{subject to: } & X \in \mathbb{R}^{n \times n},\ \text{symmetric} \\ & \forall j \in [m],\ \langle B_j, Z \rangle \leq c_j \\ & Z \succeq 0, \end{aligned} \tag{34}$$

where $Z \succeq 0$ means that Z is positive semidefinite. The semidefinite optimization problems in this note look varying degrees of different from this standard form, so I recommend not distracting yourself with trying to convince yourself something is semidefinite if you are not already familiar with this subject. Instead, you could consult [Mir23] for more details.

Relaxations Naturally, if we add constraints to a minimization problem, it can only make its value go up, and if we remove or weaken constraints, it can only make the value go down. Similarly, making a maximization problem more constrained can only make its value go down, and make it less constrained can only make its value go up. A *relaxation* of an optimization problem is a problem with weaker constraints. Sometimes it is useful to take an optimization problem of an exotic nature, with exotic constraints, and replace those constraints with weaker ones that are, for example, linear. This gives a much easier to understand optimization problem, and its value lower bounds the value of the original, in the case of a minimization problem.

Duality For any linear minimization problem, P, we can define its *dual*, $P^\dagger$, which is a linear maximization problem such that if P has an optimal solution, then $P^\dagger$ has an optimal solution, and they have the same value. The dual of the optimization problem in (33) is:

$$\begin{aligned} \text{maximize: } & \sum_{j=1}^{m} c_j z_j \\ \text{subject to: } & \forall j \in [m],\ z_j \in \mathbb{R} \\ & \forall i \in [k],\ \sum_{j \in [m]} b_{j,i} z_j \geq a_i. \end{aligned} \tag{35}$$

Thus any feasible solution to the dual gives a lower bound on its value, and thus on the value of P (called the *primal*). If we take the dual of the above program, $P^\dagger$, we get P back. The dual has a constraint for every variable (i.e. degree of freedom) in the primal, and a degree of freedom for every

constraint in the primal. That is, the more constrained the primal, the less constrained its dual, and vice versa.

Similarly, for any semidefinition minimization problem, P, we can define a dual. The dual of the semidefinite optimization problem in (34) is:

$$\begin{aligned} \text{maximize: } & \sum_{j=1}^{m} c_j z_j \\ \text{subject to: } & \forall j \in [m],\ z_j \in \mathbb{R} \\ & \forall i \in [k],\ \sum_{j \in [m]} z_j B_j \preceq A, \end{aligned} \tag{36}$$

where $A \succeq B$ means that $A - B$ is positive semidefinite. This looks quite different from the problem in (34), but it turns out that it is also a semidefinite optimization problem. The most important point we will need to understand the contents of this week's material is the following:

> **Duality:** Let P be a semidefinite minimization problem, and $P^\dagger$ its dual. Then it always holds that the value of P is at least the value of $P^\dagger$ (weak duality) and if certain conditions are met, then P and $P^\dagger$ have the same value (strong duality). **[16]**

We will not go into detail on the required conditions, but all semidefinite programs in these notes will have strong duality. So when we have a minimization problem P, feasible solutions give upper bounds on the value, and feasible solutions to its dual give lower bounds on its value, and in the case of strong duality, we can find the optimal value (and be certain we have found it) by exhibiting matching upper and lower bounds via optimal dual feasible and feasible solutions.