CSE 291 / Math 277A - Quantum Complexity Theory November 8th, 2022

Lecture 14

Lecturer: Daniel Grier *Scribe: Christopher Ye*

1 Warmup Exercise

Recall from a previous lecture the complexity class NC^0 of classical circuits with bounded fan-in and constant depth. In this warmup, we introduce a similar class AC^0 with the modification that the gates are now allowed to have unbounded fan-in. However, we restrict the size of the circuits to be polynomial to avoid the trivial case that any function can be solved by a depth-2 circuit of unbounded fan-in and unbounded size.

Definition 1. *(AC^0) The class of languages $L \subseteq \{0,1\}^*$ such that there exists a uniform family of poly-size, constant-depth classical circuits $C_n \colon \{0,1\}^n \to \{0,1\}$ built from* AND*,* OR*, and* NOT *gates with unbounded fan-in where $x \in L$ if and only if $C_n(x) = 1$.*

We show that this class is strictly larger than NC^0.

Proposition 2. $\mathsf{NC}^0 \subsetneq \mathsf{AC}^0$

Proof. There are two things to prove: every NC^0 language is in AC^0, and there is some language in AC^0 that is not in NC^0. For the first requirement, notice that AC^0 contains a larger set of possible gates than NC^0, so AC^0 contains NC^0 so long as NC^0 circuits are also restricted to polynomial size (recall that we did not make this size restriction explicit). However, this is actually just a consequence of the fact that NC^0 circuits have constant depth. Since each output bit depends on at most constantly-many input bits and gates, the rest of the circuit can be dropped. Therefore, every NC^0 circuit can be made to have linear size.

To show that AC^0 is strictly larger than NC^0, consider the OR function. Clearly it is in AC^0 by simply taking an OR gate on all inputs. However, once again, every output bit of an NC^0 circuit depends only on a constant number of inputs, so it cannot possibly compute the OR function correctly on arbitrarily many inputs. □

2 Hierarchy of NC^k and AC^k

Above, we saw that adding unbounded fan-in increases the power of NC^0. However, below we will see that there is a relation problem solved by QNC^0 circuits that is not solved by AC^0 circuits. A key ingredient in this result is famous result that parity is not computable by AC^0 circuits.

Definition 3.

$$\text{PARITY} = \left\{ x \in \{0,1\}^* \mid \bigoplus_{i=1}^{n} x_i = 0 \right\}$$

Theorem 4 (Ajtai [Ajt83], Furst-Saxe-Sipser [FSS84])**.** PARITY $\notin \mathsf{AC}^0$

Proof. For a nice write-up of this result, see [Kat] and [BS90]. □

We can also generalize the definitions of NC^0 and AC^0 to allow for increasingly larger depth circuits.

Definition 5. *NC^k is the class of languages computable by a uniform family of poly-size circuits with $O(\log^k n)$-depth and bounded fan-in gates. AC^k is the class of languages computable by a uniform family of poly-size circuits with $O(\log^k n)$-depth and unbounded fan-in* AND*,* OR*, and* NOT *gates.*

Note that PARITY $\in \mathsf{NC}^1$ since we can build a $O(\log n)$-depth circuit computing parity by building a tree of XOR gates. In general, we have the following set of relations,

$$\mathsf{NC}^0 \subsetneq \mathsf{AC}^0 \subsetneq \mathsf{NC}^1 \subseteq \mathsf{AC}^1 \subseteq \dots$$

where the only strict inclusions known are the first two.

Above, we showed that NC^1 is separated from AC^0 by the PARITY function. What if we forcibly include the PARITY function into AC^0? This gives the motivation for defining the class $\mathsf{AC}^0[p]$, which is like AC^0 with the additional MOD_p gate. $\mathsf{MOD}_p \colon \{0,1\}^* \to \{0,1\}$ is a Boolean gate that maps the input to 1 if the sum of the inputs is a multiple of p, and outputs 0 otherwise. Note that the parity gate is a MOD_2 gate.

Definition 6. *($\mathsf{AC}^0[p]$) The class of languages $L \subseteq \{0,1\}^*$ such that there exists a uniform family of poly-size, constant-depth classical circuits $C_n \colon \{0,1\}^n \to \{0,1\}$ built from* AND, OR, NOT, *and* MOD_p *gates with unbounded fan-in where $x \in L$ if and only if $C_n(x) = 1$.*

In particular, we have that PARITY $\in \mathsf{AC}^0[2]$. Furthermore, we have the following fact on the distinctness of $\mathsf{AC}^0[p]$ for different primes p.

Theorem 7 (Razborov [Raz87], Smolensky [Smo87])**.** *Let p, q be distinct primes. Then $\mathsf{AC}^0[p] \neq \mathsf{AC}^0[q]$.*

3 How hard is it to classically simulate QNC^0 circuits?

In the previous lecture, we saw that there was a relation task solved by a constant-depth quantum circuit (QNC^0) that cannot be solved by NC^0 circuits. How far can we push this separation? In other words, can we show that even larger classes of classical circuits (e.g., AC^0, NC^1) cannot simulate these simple quantum circuits?

It turns out that you can! A results of Bene Watts, Kothari, Schaeffer, and Tal [WKST19] states that there is a relational problem in QNC^0 that cannot be solved by AC^0 circuits. The task is essentially identical to the GRID problem we saw in the previous lecture.

Today, we will try to go a step further. Namely, we will show that any classical circuit computing (a modification of) this task can also be used to compute NC^1-hard problems.

Theorem 8 (Grier, Schaeffer [GS20])**.** *Suppose there is a classical circuit $\mathcal{O}$ solving (a modification) of the* GRID *problem. Then this circuit $\mathcal{O}$ can be leveraged to solve NC^1-hard problems. Formally, $\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathcal{O}}$.*

Let's see how we might go about using this theorem to prove separations. Suppose that the circuit $\mathcal{O}$ could be instantiated with an AC^0 circuit. By the theorem, we have

$$\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathsf{AC}^0} = \mathsf{AC}^0$$

where the last equality follows from the observation that giving an AC^0 circuit the "additional" ability to solve AC^0 problems doesn't increase its power. Recall, however, that PARITY is a problem in NC^1 that is not in AC^0 (Theorem 4), so it is impossible for AC^0 to contain NC^1. This contradiction implies that there must not have been an AC^0 circuit that solved the modified GRID problem.

We can generalize this argument to show that not even $\mathsf{AC}^0[p]$ circuits can solve this problem. We get

$$\mathsf{NC}^1 \subseteq (\mathsf{AC}^0)^{\mathsf{AC}^0[p]} = \mathsf{AC}^0[p]$$

since we can replace any oracle access with a constant-depth circuit with MOD_p gates. However, $\mathsf{NC}^1 \not\subseteq \mathsf{AC}^0[q]$ for all primes q by Theorem 7. Therefore, $\mathcal{O} \notin \mathsf{AC}^0[p]$ for any prime p. In other words, we have exhibited a problem that can be computed by QNC^0 circuits but not by $\mathsf{AC}^0[p]$ circuits. We will see much more "impressive" separations later between quantum and classical complexity classes, but these separations will rely on unproven conjectures. This is one of the largest-known *unconditional* separations between natural classes of quantum and classical circuits.

4 Sketching the proof of Theorem 8

There are five main ingredients that go into the proof of Theorem 8. We start with the definition of the problem itself.

4.1 Ingredient 1: Making the GRID problem interactive

The main modification we make to GRID problem is *interactivity.* Namely, we start as usual with the $N \times N$ grid, but then we give the input (i.e., the measurement bases) in two distinct rounds. We imagine these questions being asked by a classical verifier. In round 1, the verifier gives measurement bases for some subset of the qubits. The prover is expected to make those measurements, and output measurement results that are consistent with the bases specified by the verifier. In round 2, the verifier gives measurement bases for the remainder of the qubits, and the prover must once again return measurement outcomes that are consistent with both these bases given in round 2 and its previously returned answers in round 1. It's worth emphasizing that this problem is both interactive and relational, that is, the prover can output any responses that have non-zero probability in the true distribution that arises from measuring the quantum graph state.

For a quantum circuit, the interactive and non-interactive problems are essentially identical—the circuit simply defers certain measurements until round 2. However, imagine that you had a classical instantiation of the prover. In this case, the classical circuit could overcome the quantum no-cloning limitation, and can *copy* its state after the first step. Therefore, it can perform the second stage arbitrarily many times by "rewinding" the state of the machine to the end of the first stage. This intuitively suggests that a classical circuit playing the same game has to be somehow more powerful than its quantum counterpart.

On a final technical note, the grid that we will be using for this lecture is actually the $2 \times N$ grid not the $N \times N$ grid. It turn out that using the $N \times N$ grid will actually give us a stronger result (showing that the circuit can be leveraged to solve problems in a complexity class called $\oplus\mathsf{L}$). That said, the $2 \times N$ case will still have most of the main technical ideas, so we will stick with that.

4.2 Ingredient 2: A NC^1-Hard Problem - Clifford Multiplication

Recall that our goal is to leverage any classical simulator for the interactive GRID problem above to solve NC^1-hard problems. Here, we present this NC^1-hard problem:

Clifford Multiplication:
Input: List of 2-qubit Clifford gates $g_1, g_2, \dots, g_n$
Output: Their product $g_n g_{n-1} \cdots g_1$

Fact 9. Clifford Multiplication *is* NC^1*-hard, even under the promise that the output is* $I \otimes I$ *or* $H \otimes H$*.*

We omit the proof but note that this is a consequence of Barrington's Theorem, namely that multiplying over the symmetric group S_5 is an NC^1-hard problem. Indeed, there is a subgroup of the Clifford group (the Clifford group mod the Pauli group) that is isomorphic to S_6, so that multiplication over the Clifford gates must be at least as hard as multiplication over S_5.

4.3 Ingredient 3: Measurement-Based Quantum Computation

Above, we have presented a problem that seems in some sense inherently sequential, or at the very least cannot be computed in constant depth. However, we now introduce *measurement-based quantum computation*, a trick we will use to parallelize the sequential computation via quantum measurements. We use the following proposition without proof. A similar statement will be shown in the following lecture.

Proposition 10. *For all 2-qubit Clifford gates g, there exists a set of* 38 *measurements on the* 2×20 *grid graph state that leaves the unmeasured qubits in the state $Pg|00\rangle$ for some Pauli matrix P, where P depends on the outcomes of the measurements.*

In fact, we claim without proof that you can string these gadgets together, so that if you have a graph state of size $2 \times (20n)$ and a list of gates $g_1, \dots, g_n$, then there is a set of measurements on the graph state that creates the state

$$P_n g_n P_{n-1} g_{n-1} \dots P_1 g_1 |00\rangle$$

where the Pauli terms $P_1, \dots, P_n$ depend on the measurement results obtained in each one of the gadgets.

Next, recall that the Pauli matrices are a normal subgroup of the Clifford group (i.e., conjugating a Pauli operator by a Clifford operator yields another Pauli). This implies that you can "push" Pauli elements through Clifford elements:

$$gP = gP(g^\dagger g) = (gPg^\dagger)g = P'g$$

where P' is just some other Pauli operator. Using this idea, we can propagate all the Pauli operators to the front of our expression:

$$P_n g_n \dots P_1 g_1 = P'_n P'_{n-1} \dots P'_1 g_n \dots g_1 = P^* g_n g_{n-1} \dots g_1 |00\rangle$$

for some Pauli operator P^*.

Recall that we can use the promise that $g_n g_{n-1} \cdots g_1$ is either $I \otimes I$ or $H \otimes H$. By our previous observation, we just need to distinguish between these cases where the state has been corrupted by some unknown Pauli P^*. (Note that in principle, one could compute P^* from the measurement outcomes. Unfortunately, this computation is itself NC^1-hard, so we can't do this in our reduction).

To circumvent this issue, let's first consider the stabilizer groups for our ending state $|00\rangle$ and $|{+}{+}\rangle$, respectively:

$$\begin{pmatrix} II \\ ZI \\ IZ \\ ZZ \end{pmatrix}, \qquad \begin{pmatrix} I\,I \\ X\,I \\ I\,X \\ XX \end{pmatrix}$$

Finally, applying P^* to the state is equivalent to conjugating by P^* in the stabilizer group. The key fact is that conjugation by Pauli matrices can only change the sign. In particular, the output of the measurement based quantum computation is

$$\begin{pmatrix} I\,I \\ a\,PI \\ b\,IP \\ abPP \end{pmatrix}$$

for some $P \in \{X, Z\}$ and $a, b \in \{\pm 1\}$. Our goal then is to distinguish $P = X, P = Z$.

4.4 Ingredient 4: Randomizing the Output

In the next two sections, we will treat the prover as a sort of adversary. Recall that our claim is that we can leverage the simulator to solve the Clifford multiplication problem. In particular, this should be true regardless of which answers the simulator gives us. Therefore, we should expect that the simulator gives us the maximally useless answers at all times. Or, in other words, that the simulator is acting adversarially. To be clear, the simulator must still be correct, but it can still try to answer correctly, without being useful.

Unfortunately, it will turn out that if we only have two possible outcomes (the $I \otimes I$ outcome and the $H \otimes H$ outcome), then adversarial outcomes will prevent us from learning which state we have. To overcome this, let us briefly describe how we can "randomize" the output of the previous step. Before we compute $g_n \cdots g_1$, we can generate random Clifford gates $h_0, \dots, h_n$ and redefine $g'_i = h_i g_i h_{i-1}^{-1}$. Then,

$$g'_n g'_{n-1} \dots g'_1 = h_n g_n h_{n-1}^{-1} h_{n-1} g_{n-1} h_{n-2}^{-1} \dots h_1 g_1 h_0^{-1} = h_n g_n g_{n-1} \dots g_1 h_0^{-1}$$

Thus, rather than the two specific stabilizer groups above, we now must distinguish between two stabilizer groups that are random! Since we chose h_i, we know which outcome with correspond to $I \otimes I$ and $H \otimes H$, but what we have essentially done is made the input uniformly random among all sets of n Clifford gates. The adversary does not know.

4.5 Ingredient 5: The Magic Square

Our goal is to measure our unknown 2-qubit state to learn what it is. It will turn out that any single measurement cannot give us enough information to determine this state. So, this is the first place we will use interactivity and the rewinding ingredient shown earlier. Still, even with many measurements, it's a bit unclear which measurements we should make.

Our measurements will correspond to the rows/columns of the following matrix called the *Magic Square*:

$$\begin{pmatrix} XX & YY & ZZ \\ YZ & ZX & XY \\ ZY & XZ & YX \end{pmatrix}$$

Let's note some important properties of this arrangement of Pauli operations:

1. The Pauli operators within each column and row commute. This means that we can "measure" all 3 Pauli operations simultaneously, i.e., rotate to the Z-basis and then measure.

2. The product of each row is $-II$. That is, the sum of measurement results on a row must be odd.

3. The product of each column is II. That is, the sum of measurement results on a column must be even.

We now need two key observations:

Fact 11. *If a measured Pauli operator happens to be in the stabilizer group of the unknown state, then the measurement result for that Pauli is equal to the sign of the Pauli in the stabilizer group (This is just a generalization of Lemma 2 from the previous lecture).*

Fact 12. *There is no consistent way to label a* 3×3 *matrix with* $0/1$ *values such that the sum of each row is odd and the sum of each column is even.*

Therefore, if we make all measurements corresponding to each row and each column, there must be some measurement output which was different for a particular Pauli (Fact 12). On the other hand, if we happened to measure some Pauli that was in the stabilizer group, then we could not have gotten different results. Together, the 6 measurements reveal at least one Pauli element which is *not* in the stabilizer group of the measured state.

4.6 Putting It All Together

Let's first consider the problem from the perspective of the quantum algorithm. In round 1, it uses measurement-based quantum computation (ingredient 3) to prepare the state $P^* g_n \dots g_1 |00\rangle$ in constant depth. The gates $g_1, \dots, g_n$ correspond to the input of some 2-qubit Clifford multiplication problem (ingredient 2). In round 2, it measures that state in the specified basis.

Now consider a classical simulator achieving the same output as the quantum algorithm. After round 1, the simulator has some classical internal state which captures the true quantum state $P^* g_n \dots g_1 |00\rangle$. Using the "rewinding" technique on this intermediate state (ingredient 1), it can make many possible measurements in round 2. Armed with this power, it measures all the magic square Paulis (ingredient 5). In this process, it learns some non-stabilizer of the state.

Since we randomized our inputs (ingredient 4), this non-stabilizer will appear in one of the two stabilizer groups for the states we are trying to distinguish with some constant probability. In this case, we learn exactly which one of our two states we have, and so we have solved the Clifford multiplication problem.

There are many details left to check here, but this is most of the main ideas. For example, we must check that this reduction can indeed be done with AC^0 circuits, which upon closer inspection are actually randomized, and so there are more subtleties to handle.

References

[Ajt83] Miklós Ajtai. Σ_1^1-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983.

[BS90] Ravi B Boppana and Michael Sipser. The complexity of finite functions. In *Algorithms and complexity*, pages 757–804. Elsevier, 1990.

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984.

[GS20] Daniel Grier and Luke Schaeffer. Interactive shallow Clifford circuits: Quantum advantage against NC^1 and beyond. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 875–888, 2020.

[Kat] Jonathan Katz. Lecture on parity.

[Raz87] Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82, 1987.

[WKST19] Adam Bene Watts, Robin Kothari, Luke Schaeffer, and Avishay Tal. Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 515–526, 2019.