CMSC 858L: Quantum Complexity

Instructor: Daniel Gottesman

Spring 2023

27 MIP* = RE

27.1 References

The result that MIP* = RE comes from Ji, Nataranjan, Vidick, Wright, and Yuen, "MIP* = RE," arXiv:2001.04383. There are some nice blog posts by Yuen (https://quantumfrontiers.com/2020/03/01/the-shape-of-mip-re/#fn-14879-6) and Vidick (https://mycqstate.wordpress.com/2020/01/14/a-masters-project/) that talk about the background and history leading up to the result. For a summary of the technical aspects, see either the introductory sections of the paper or a series of 3 talks at a workshop at the Simons Institute for the Theory of Computing, (https://simons.berkeley.edu/workshops/quantum-protocols-testing-quantum-pcps/videos#simons-tabs)

27.2 Overview

We wish to prove that MIP* = RE.

Recall that MIP* is the class of languages for which there exists a multiprover interactive proof with a classical verifier (who only has to do polynomial-time computations), polynomial-size messages, and where the two provers can share entanglement. The provers cannot communicate with each other but are computationally unbounded, and can in fact solve uncomputable problems and share infinite amounts of entanglement (where the overall Hilbert space is a tensor product of the two provers' Hilbert spaces).

RE is the class of languages whose "yes" instances can be listed by a Turing machine. It includes uncomputable problems, and in fact the halting problem is an RE-complete problem.

We have a straightforward upper bound that MIP* $\subseteq$ RE, so the goal now is to prove that RE $\subseteq$ MIP*. In particular, we want to come up with a multiprover interactive proof (with the provers sharing entanglement) to solve the halting problem. The proof is long and complicated, so I will just talk about some of the main ideas.

The type of multiprover interactive proof we will use is a non-local game, which we discussed briefly last time. To recap, in a non-local game, the Verifier asks Prover 1 $a \in A$ and asks Prover 2 $b \in B$, where A and B are finite sets of possible questions; the Verifier may be asking correlated questions (a, b) but the exact choice of questions is random. Each returns an answer $x \in X$ and $y \in Y$. Then the verifier computes some function $f(a, b, x, y)$ and accepts if $f(a, b, x, y) = 1$.

Specifically, the proof works by constructing, given the description of a Turing machine M, a polynomial-size non-local game such that:

- If M halts, then the provers have a strategy for which the verifier accepts with probability 1. (So, perfect soundness)
- If M does not halt, then for any strategy of the provers, the verifier will accept with probability at most 1/2.

Note that this is a particular simple type of multiprover interactive proof, with only 2 provers and 2 rounds of communication (verifier to provers, and then provers to verifier). However, since the proof shows that non-local games have RE-complete problems, this is good enough to capture all of MIP*.

The first step of the proof is basically to adapt the classical result MIP = NEXP by creating a non-local game for NEXP-complete problems and showing that it is still sound when the provers have access to entanglement. The specific game used is a version of the classical proof, but the full analysis for soundness against quantum provers is quite involved.

The main idea of the rest of the proof is to compress this protocol so that the verifier can ask about even larger systems that would be outside NEXP. For instance, there was a precursor result which showed that MIP* contains NEEXP, which is again like NP, but now the witness and checking circuit are *doubly*-exponential in size. This is so large that the verifier cannot even ask a prover to send a single bit of the witness, say, because the witness is so long that specifying which bit the verifier wants would already take an exponential-length message.

The intuition of how to do the compression is that the verifier asks the provers to run the whole NEEXP interactive proof protocol, including the verifier part, and report the result. This is obviously a terrible idea, since the provers can just report whatever they like, and indeed, classically this does not work at all (which is why the classical MIP is only equal to NEXP and not anything larger). The verifier needs a way to check that the provers have correctly run the big protocol, and in particular that they have generated genuine random numbers for the simulated verifier's questions and have not shared those numbers with the other prover.

Classically, there is no way to guarantee that the provers are using random numbers, and that is why it fails, but the non-locality of quantum mechanics makes it possible using similar ideas to those for device-independent quantum random number generation. In particular, using self-testing techniques, it is possible for the verifier to test whether the provers share maximally entangled states and to force them to make certain measurements on those states if they want to pass the test. The main complication over the usual random number generation context is that the verifier needs to test that the provers have *exponentially many* entangled states.

Once you have a compression procedure, then the result follows by using it recursively. The compression doesn't work on arbitrary protocols, so you have to make sure that the first non-local game is in the right form and that the output of the compression procedure is also in the right form to be compressed again, but once you have done that, nothing stops you from repeating the procedure again and again to apply to triply-exponential size problems, then quadruply-exponential, and so on up and up without limit.

The actual application to the halting problem is done through the following protocol: Given a Turing machine M,

1. Verifier runs M for n steps and accepts if it halts in that time.

2. If it hasn't halted, the verifier asks the provers to run steps 1 and 2 replacing n with 2^n and to use the compression procedure to prove that they have done so correctly. The verifier then accepts if the provers manage to convince the verifier that the simulated protocol accepts.

Then, if M halts in n steps, the verifier accepts without using the provers. If it halts in more than n but up to 2^n steps, the provers determine this without further recursion and convince the verifier that they simulated the Turing machine for 2^n steps and found that it halts. If it halts in more than 2^n steps but no more than 2^{2^n} steps, then the provers simulate themselves simulating the verifier running the Turing machine for 2^{2^n} steps, and they simulate convincing the single-exponential size verifier that the doubly-exponential size verifier accepted, and then they prove to the real verifier that the single-exponentially-sized verifier was convinced. No matter how many steps it takes for the Turing machine to halt, the verifiers will get there through enough levels of compression.

The compression procedure consists of a large battery of tests (although still a constant number of test types) to ensure that the provers are acting the way they are supposed to. If the provers behave correctly and the Turing machine M halts, they can pass all the tests and have their answer accepted with 100% probability. Because there are many tests and all are needed, if M *doesn't* halt, the provers could try to adopt a strategy that fails one of the tests and passes the others. This will work unless the verifier chooses that particular test, which happens to with constant (albeit small) probability. This constant chance of failure can be magnified using a form of parallel repetition.

27.3 Self-Testing and the Quantum Low-Degree Test

The basic idea of the self-testing is that certain non-local games are *rigid*, meaning that not only does the best quantum strategy beat any classical strategy, but there is only a single quantum strategy (up to change of basis) that works to get the optimal success probability. The CHSH game is one such game — to succeed with the optimal quantum success probability, you must essentially make measurements specified by the questions on the two qubits of a maximally entangled state. The CHSH game wouldn't give perfect completeness, so in this case, we actually use a different game called the *Magic Square* game.

The Magic Square game consists of a 3×3 grid of values ± 1, and one prover is asked to return values for either a column or a row of the grid. The other prover is asked for the value of a single cell that appears in that row or column. The verifier accepts if the value of the cell matches the value returned by the first prover, and if the product of all three values returned by the first prover multiply to $+1$ (if the prover was asked for a row) or -1 (if asked for a column).

In the classical case, the provers cannot succeed with probability greater than $8/9$, because to definitely agree on the value of a cell, the provers need to have prearranged answers for all the squares. However, if they do so, then the product of the values for all 9 cells has to be $+1$ because the product of each column is $+1$. Unfortunately, the product of all 9 values *also* has to be -1 because the product of each column is -1. Thus, the provers can't possibly succeed with 100% probability.

In the quantum case, the provers can succeed with 100% probability if they share two EPR pairs $|00\rangle + |11\rangle$. Whenever either is asked to return the value of a cell, they measure a Pauli determined by the cell on their two qubits (each being half of one EPR pair) and return the measured value:

$$\begin{array}{ccc} I \otimes X & X \otimes I & X \otimes X \\ Z \otimes I & I \otimes Z & Z \otimes Z \\ -Z \otimes X & -X \otimes Z & Y \otimes Y \end{array}$$

Each row and column commutes, so the prover who has to return a full row and column can do so, and because both provers are measuring on the qubits of the shared EPR pairs, they will get the same result. It turns out that the magic square game is rigid, so that the only way the provers can pass with close to 100% probability is do something close to this quantum strategy. This relies on the provers not knowing which question the other one is being asked, so for instance, when a prover is asked for a column, the other prover might be asked for any of the three entries in the column, and so must make the correct measurement for all three entries.

The verifier can test for more EPR pairs by using a battery of different Pauli measurement requests to the provers. Again, the provers don't know which question the other one is being asked, and that forces them to stick to making the requested measurements and nothing else. To test exponentially many EPR pairs, the verifier must use a version of the classical low-degree test used in the proof that MIP = NEXP.

27.4 Introspection

The compression procedure has two pieces: The questions being asked by the verifier need to be reduced from exponential to polynomial size (*question reduction*), and the answers being provided by the provers also need to be reduced from exponential to polynomial size (*answer reduction*). These two parts use different techniques. The approach used for question reduction is known as *introspection*.

Because the provers are forced to make the correct measurements on genuine entangled states, the specific measurement results they get are random and can be used as the source of the simulated verifier's questions. We can also throw in additional Paulis mixed in with the measurements for the low-degree test in order to generate the questions needed for the simulation. The low-degree test forces the provers to measure the correct Paulis, and so these measurements can generate exponentially many random bits using only questions on polynomially many bits.

By picking the measurements to be done carefully, we can make sure that the provers get the same random bits where they need to but that the correct information is hidden from them so that they don't know what the other prover is being asked in the simulated protocol. Note that the provers don't necessarily know when they are measuring a Pauli that is used to generate questions and when it is one of the Paulis being used for cross-checking, so they need to go ahead and simulate the full protocol and report the answers in either case. The verifier can then ignore the simulation results when the actual questions don't correspond to simulated questions.

27.5 Answer Reduction

The last step is to reduce the size of the provers' answers. Instead of providing the full answers to the exponentially large non-local game, the provers create a probabilistically checkable proof (PCP) and send samples from that (as requested by the verifier). This lets the verifier evaluate the exponentially large answers by only looking at a small number of bits.

However, there is a problem in that the PCP needs to contain the verifier's full checking procedure for the simulated game, and that requires the answers from both provers — which means that as things currently stand, neither prover knows everything they need to compute the PCP. The solution is a process called *oracularization*, in which one prover runs the full simulation, including *both* provers. This again seems like a terrible idea, since that prover could then cheat, but we can prevent this by having the other prover run the simulation but as only *one* of the simulated provers — and it could be either one, chosen at random. Then in addition to requiring that the verifier accept the PCP version of the answers to the simulated protocol, the verifier also checks that the answer provided by the second prover matches the corresponding answer provided by the first prover. The second prover needs to provide a short version of the answer as well, such as a hash of the answer, to keep the answers to polynomial size.