6.896 Quantum Complexity Theory	November 6, 2008

Lecture 19

Lecturer: Scott Aaronson *Scribe: Joshua Horowitz*

On Election Day, November 4, 2008, the people voted and their decision was clear: Prof. Aaronson would talk about quantum computing with closed time-like curves. But before we move into strange physics with even stranger complexity-theoretic implications, we need to fill in a more basic gap in our discussion so far – quantum space-complexity classes.

1 BQPSPACE

Recall that PSPACE is the class of all decision problems which can be solved by some classical Turing machine, whose tape-length is bounded by some polynomial of the input-length. To make BQPSPACE, we quantize this in the most straightforward way possible. That is, we let BQPSPACE be the class of all decision problems which can be solved with a bounded rate of error (the B) by some quantum Turing machine (the Q), whose tape-length is bounded by some polynomial of the input-length (the PSPACE). As we've seen before, allowing quantum computers to simulate classical ones gives the relationship PSPACE $\subseteq$ BQPSPACE.

If we believe that BQP is larger than P, we might suspect by analogy that BQPSPACE is larger than PSPACE. But it turns out that the analogy isn't such a great one, since space and time seem to work in very different ways. As an example of the failure of the time-space analogy, take PSPACE vs. NPSPACE. We certainly believe that nondeterminism gives classical machines exponential time speed-ups for some problems in NP, making P $\neq$ NP at least in our minds. But Savitch's theorem demonstrates that the situation is different in the space-domain: PSPACE = NPSPACE. That is, nondeterministic polyspace algorithms can be simulated in deterministic polyspace. It turns out even more than this is true. Ladner proved in 1989 that PSPACE = PPSPACE, establishing PSPACE as a truly solid rock of a class.

This is the result we need to identify BQPSPACE. Using the same technique we used to prove BQP $\subseteq$ PP, we can prove that BQPSPACE $\subseteq$ PPSPACE, and then using PPSPACE = PSPACE $\subseteq$ BQPSPACE we have BQPSPACE = PSPACE.

2 Talking to Physicists

Some physicists have been known to react to the stark difference between time- and space-complexity with confusion and incredulity. "How can PTIME $\neq$ PSPACE," they ask, "if time and space are equivalent, as suggested by Einstein's theory of relativity?"

The basic answer to this question is that time and space are in fact not perfectly equivalent. Einstein's theory relates them in a non-trivial way, but distinctions remain. At the very least, they correspond to opposite signs in the metric tensor. And, for whatever reason, though objects can move about space at will, movement through time is constrained to the "forward" direction. Thus, our inability to travel back in time is in a sense the reason why (we believe) P $\neq$ PSPACE!

3 Time Travel

But are we really so sure that time travel is impossible? Special relativity says that it takes infinite energy to accelerate faster than the speed of light and into the past. But general relativity extends our concept of space-time, allowing it to take the form of whatever sort of topologically crazy 4D manifold we want (more or less). Is it possible that such a manifold could have loops of space-time which extended forwards in time but ended up bringing us back to where we started? That is, can the universe have *closed time-like curves* (CTCs)?

In an amusing coincidence for a class on complexity theory, one of the first people to investigate this question was Kurt Gödel, who in 1949 constructed a solution to the general-relativistic field equations which included CTCs, and gave it to Einstein as a birthday gift. Like all good birthday gifts, this left Einstein somewhat troubled. The solution was fairly exotic, however, involving such strange constructions as infinitely long massive cylinders.

The problem came up again in the 1980s, when Carl Sagan asked his physicist friend Kip Thorne if there was a scientific basis for the time travel in his novel-in-progress *Contact*. Though Thorne initially dismissed the possibility, he later began to look into whether wormholes could be used for time travel. The problem turned out to be surprisingly difficult and subtle. The conclusion reached by Thorne and his collaborators was that such wormholes were theoretically possible, though they would require the presence of negative energy densities. Given that such negative energy densities have been known to arise in quantum situations such as the Casimir effect, Thorne's work effectively linked the question of wormhole time-travel to that of quantum gravity (the "P vs. NP" of theoretical physics).

Of course, it is not the place of computer scientists to say whether something actually exists or not. They need only ask "what if?". So we will suppose that we have computers capable of sending information back in time, and see where that takes us.

4 Computing with CTCs: The Naïve Proposal

The first scheme many people think of for speeding up computations with CTCs is very simple:

1. Perform your computation however you like.

2. Send the answer back in time to whenever you wanted it.

At first glance, this sounds pretty good. It makes every computation constant-time! Conceivably, even negative-time...

But there are problems with this scheme, which is fortunate for anyone hoping time-travel computation would be anything other than completely trivial.

- It encourages "massive deficit spending": Life may be easy for you, now that you've received the answer to your NP-complete question in no time at all, but the accounting of time as $O(0)$ ignores the possible millennia of computation your computer still has to do to get that answer. Keeping a computer running that long could be extraordinarily costly or even impossible if the run-time exceeds the lifetime of the universe itself.

- Complexity theory is all about knowing exactly how much of each computational resource you need in order to solve a problem. However, in the above treatment, we are completely ignoring the CTC as a computational resource. If we accounted for its length, we would soon discover just how blithely we were constructing exponentially long wormholes in our naïve attempt to defeat complexity itself.

- There must be a fundamental flaw in the way we are modeling time travel, since, the way we've been talking, there's nothing to prevent "grandfather" paradoxes[1]. As a computational example of this, suppose we had a computer which took its input, NOTed it, and then sent that back in time to serve as its own input. If the machine sent a 0, it would receive a 0 and thus send a 1, and visa versa, so the machine's existence creates a physical contradiction.

5 Deutsch's Solution and $\mathsf{P}_{\mathsf{CTC}}$

Of the problems mentioned above, the most pressing one to solve is problem the last. How can the universe allow time travel at all if this creates time-loop paradoxes?

In 1991, David Deutsch proposed a resolution to this problem. He claimed that time-loop paradoxes such as the NOTing computer above (or the grandfather paradox itself) can only occur in classical universes. In a quantum universe, such as our own, the universe can always find a way to maintain consistency (according to Deutsch).

As an example of this sort of resolution, take the grandfather paradox, and represent the two possibilities as a vector $\vec{v}$ which can be either

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix} \leftarrow \text{you kill your grandfather} \quad \text{or} \quad \begin{bmatrix} 0 \\ 1 \end{bmatrix} \leftarrow \text{you do not kill your grandfather} .$$

Your murderous time-travelling adventure puts a constraint on $\vec{v}$:

$$\vec{v} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \vec{v}.$$

Clearly, neither the two possibilities mentioned above satisfies this fixed-point equation; the two possibilities flip into each other. This is just a restatement of the grandfather paradox. But this restatement presents an interesting idea: what if we could have $\vec{v} = \frac{1}{2}\begin{bmatrix} 1 \\ 1 \end{bmatrix}$? This "superposition" of the two states is a fixed point of the matrix map above. As suggested by the normalization $\frac{1}{2}$ out in front, we are interpreting it as a vector of probabilities. (Truly quantum states will come later; for now we will use a semi-classical or probabilistic approach.)

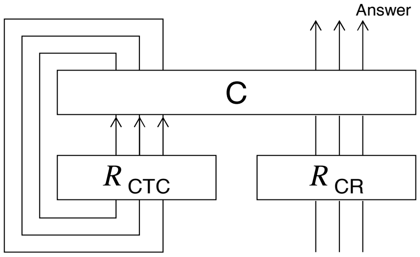
Does turning every time-loop constraint into a matrix and looking for general fixed-point vectors always work? That is, if some computation in a CTC establishes the constraint $\vec{v} = M\vec{v}$, is there always a solution for $\vec{v}$? This certainly isn't true for any matrix M, but matrices which come from time-loop constraints will satisfy additional properties, since they have to take probability vectors to probability vectors. A necessary and sufficient pair of conditions which guarantees that M does this is that M's entries are non-negative and that each of its columns sums to one. A matrix with these properties is said to be a *stochastic matrix.*

Fortunately for us, it is well-known that every stochastic matrix has at least one probability vector as a fixed point. This is exactly what we need to ensure that the universe can find a

[1] The story is that, by traveling back in time and killing your own grandfather, you prevent your own birth, thus preventing your going back in time to kill your grandfather, thus enabling your own birth, etc.

probability distribution over the classical states which is preserved by the time loop! So in our model, our computer's design will determine for Nature a stochastic matrix M, and Nature will in turn provide the computer with some fixed-point probability distribution which it can sample by observation. (As good computer scientists, we will assume that Nature chooses this distribution as an adversary.)

Let's make this rigorous: We will have some classical computer C. It will have two input registers, one for the CTC-looping bits, called R_{CTC}, and one for the standard "causality-respecting" bits, called R_{CR}:



Given a set of inputs to R_{CR}, C determines a map from R_{CTC} to R_{CTC}. This map can be represented by a stochastic matrix, which we know must have some fixed-point probability distribution. Each such probability distribution gives a probability distribution for values of C's answer bits.

In the case of decision problems, C has exactly one answer bit. We say that our CTC computer accepts (resp. rejects) some input if every non-zero-probability value of R_{CTC} in every fixed-point probability distribution causes C to give an answer of 1 (resp. accepts). Since we are permitting no uncertainty in the actual output of the computer, these concepts of accepting/rejecting are appropriate for defining the complexity class $\mathsf{P}_{\mathsf{CTC}}$ of polynomial-time computers with closed time-like curves. The only other subtlety is that we should make sure we only use a polynomially large number of CTC bits.

5.1 An Example: Search

To illustrate the power of CTCs, let us use them to solve an NP-complete problem: search. That is, given a polytime computable function $f : \{0,1\}^n \to \{0,1\}$, we want to find an $x \in \{0,1\}^n$ such that $f(x) = 1$, or to know that no such x exists. One approach to solving this with CTCs is to find a map for which fixed points correspond to xs with $f(x) = 1$. The simplest candidate is

$$C(x) = \begin{cases} x & f(x) = 1 \\ x+1 \pmod{2^n} & f(x) = 0 \end{cases}$$

and it turns out that this works perfectly! If there is a x with $f(x) = 1$, the fixed-point distributions of C will be non-zero only on such solutions (since probability flows towards these solutions under the action of the matrix M). If there is no such x, then C acts as a 2^n-cycle on the values of R_{CTC}, and the only fixed-point distribution is the uniform distribution over all possibilities. Thus, by taking the value of R_{CTC} and testing the value of f at that value, we can determine conclusively whether that value is a solution or that there is no solution at all. (Note that, though the value of the R_{CTC} bits is in general probabilistic or undetermined, the final answer is always exactly what we want.)

Therefore, the search problem is in $\mathsf{P}_{\mathsf{CTC}}$. Since the search problem is NP-complete, we have $\mathsf{NP} \subseteq \mathsf{P}_{\mathsf{CTC}}$. Assuming $\mathsf{P} \neq \mathsf{NP}$, this gives us a separation between P and $\mathsf{P}_{\mathsf{CTC}}$.

6 A New Sort of Paradox?

We know that by allowing probabilistic superpositions of states, we have successfully averted the problem of time-loop paradoxes (fixed-point-less CTC constraints). But in its stead, we have produced a new sort of problem. CTCs allow us to make programs which can find the answers to problems "out of thin air", without doing any of the work to find the solution. This is much like the time-travel paradox of going back in time to 1904 and giving Einstein his own papers, which he looks over with great interest and proceeds to furtively publish a year later. Who actually wrote the papers?

Of course, this is not a true paradox of mathematical or physical contradiction (a *falsidical paradox*), but one of a derived result contradicting natural intuition (a *veridical paradox*). It seems that if we are to allow time travel, we must abandon some of our intuitions about causality and information. "Plays get written, plexiglass gets invented, and NP-complete problems get solved," without anyone having to put in the work.

7 $\mathsf{P}_{\mathsf{CTC}}$ Vs. PSPACE

We would like to understand exactly what is possible in polynomial time using closed time-like curves. That is, we would like tight bounds on $\mathsf{P}_{\mathsf{CTC}}$. We already have a lower bound of $\mathsf{NP} \subseteq \mathsf{P}_{\mathsf{CTC}}$.

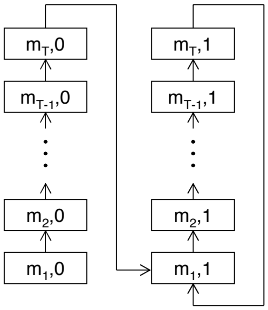
There is also good upper bound: PSPACE. Simulating a $\mathsf{P}_{\mathsf{CTC}}$ with a PSPACE machine is fairly straightforward: We can visualize the action of our computation on R_{CTC} as a directed graph in every vertex has out-degree 1. Such a graph will have the structure of a collection of disjoint cycles, possibly with trees of edges flowing into the cycles. Thus, starting with any vertex, repeated application of the computation will eventually bring you to a vertex in one of the cycles, which will have non-zero probability on a fixed-point distribution. Since it's no fun to detect whether a vertex is in a cycle or not, we will just play it safe and iterate the computation on the arbitrarily chosen point 2^n times: we must end up in a cycle, or there would be no cycle, which is impossible.

The more surprising fact is that we can improve $\mathsf{P}_{\mathsf{CTC}}$'s lower bound from NP to PSPACE, thus proving that $\mathsf{P}_{\mathsf{CTC}} = \mathsf{PSPACE}$. Apparently CTCs allow us to interchange time and space, like the physicists suspected!

To do this, we must find a way to perform every PSPACE computation in polytime with CTCs. The basic idea is to store the PSPACE machine's tape in the the CTC register and have the $\mathsf{P}_{\mathsf{CTC}}$ machine perform a single step of the PSPACE machine's computation. But we have to be careful to define our computation so that we can read the final result of the simulated PSPACE computation off of the fixed-point distribution.

We do this by adding an auxiliary bit to the CTC register which represents the outcome of the computation. Usually this auxiliary bit remains untouched, but we redirect every halting state to the initial state with the appropriate auxiliary bit (1 if the halting state was accepting, 0 if it was rejecting).

The directed graph of this operation is shown below (in a situation where the machine accepts):



As this diagram shows, the only cycle in the graph will be one going from the initial state all the way to a halting state and then back to the start, with every state having the auxiliary bit determined by whether that halting state was accepting or rejecting. Thus, by giving us a fixed-point distribution, Nature is doing the entire computation for us and telling us how it went.

Notice exactly how our scheme exploits the relationship between PSPACE and $\mathsf{P}_{\mathsf{CTC}}$. Since our tapes have polynomial length, a single computation step can be done in polynomial time. In contrast, we cannot use this technique to put $\mathsf{EXPSPACE}$ inside $\mathsf{P}_{\mathsf{CTC}}$, which is good, since $\mathsf{P}_{\mathsf{CTC}} \subseteq \mathsf{PSPACE}$ and $\mathsf{EXPSPACE}$ is certainly not a subclass of PSPACE.

8 Quantum Computation and Time Travel: $\mathsf{BQP}_{\mathsf{CTC}}$

Though our model of time-travelling computers was inspired by quantum physics, we haven't actually given our computers the quantum capabilities which we've been studying all semester. What would that look like?

When we had classical computers with time-loops, the time-loop constraint was of the form $\vec{v}_{\text{CTC}} = M\vec{v}_{\text{CTC}}$ for some stochastic matrix M. What sort of time-loop constraint could a quantum computer make? A good first guess would be $|\psi_{\text{CTC}}\rangle = U|\psi_{\text{CTC}}\rangle$ for some unitary U. But this isn't quite right. Though our computation as a whole, from [input bits + CTC bits] to [output bits + CTC bits] can be represented by a unitary U, restricting this to a map from [CTC bits] to [CTC bits] does not give a unitary. In fact, the possibility of entanglement between the output bits and the CTC bits means that it doesn't make any sense to think of the state of the CTC bits as a pure quantum state: $|\psi_{\text{CTC}}\rangle$ doesn't exist.

The right way to represent a substate of a quantum state is with a *density matrix* $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. How are we allowed to transform density matrices into each other? Well, every unitary map on a pure-state supersystem restricts to a transformation on the mixed-state subsystem. There is an exact mathematical characterization of the form such transformations can take:

$$\rho \mapsto \sum_i E_i \rho E_i^\dagger \text{ where } \sum_i E_i^\dagger E_i = I.$$

This characterization is exact in the sense that the restriction of any unitary to a mixed-state subsystem gives such a transformation, and every such transformation comes from the restriction of a unitary on a pure-state supersystem. We call these transformations *superoperators* (the analogy to remember is "Pure state : Mixed state :: Unitary : Superoperator").

So, in general, we can expect our quantum computer to create a constraint of the form $\rho_{\mathrm{CTC}} = S(\rho_{\mathrm{CTC}})$ for some superoperator S. Fortunately for the coherence of space-time, such an equation is always solvable! That is, every superoperator has a fixed point. So if we do the same sort of thing we did above to define $\mathsf{P}_{\mathsf{CTC}}$ but on quantum computers instead, and require only a bounded rate of error rather than perfection, we'll get $\mathsf{BQP}_{\mathsf{CTC}}$.

Now to bound this complexity class. Quite trivially, $\mathsf{P}_{\mathsf{CTC}} \subseteq \mathsf{BQP}_{\mathsf{CTC}}$, and $\mathsf{P}_{\mathsf{CTC}} = \mathsf{PSPACE}$, so we can bound $\mathsf{BQP}_{\mathsf{CTC}}$ below by PSPACE. We expect that adding quantum capabilities will extend our abilities, though, so what's the upper bound?

Well, it turns out that our expectation is wrong: $\mathsf{BQP}_{\mathsf{CTC}}$ is also bounded above by PSPACE. Once again, we are face-to-face with the solidity of the rock that is PSPACE! The fact that $\mathsf{BQP}_{\mathsf{CTC}} = \mathsf{PSPACE}$ was proven by Aaronson & Watrous in 2008. The title of their paper, "Closed Timelike Curves Make Quantum and Classical Computing Equivalent", raises an interesting question: why are we spending so much time trying to get quantum computers to work if they would be rendered totally obsolete by CTC computers?