Self-Reflection

Ben Little

Sep 25, 2023

Planning

Goals

My goal for this class is to learn more about the nature of proofs in complexity theory. Relationships between complexity classes are expressed as theorems and their corresponding proofs. How do these theorems and proofs relate to others used in mathematics? Can studying these data types yield insight into unexpected areas of math and science?

In order to answer these questions, I need a lot more context about complexity theory. The definitions, the proof techniques, etc. That's what I'm hoping to get out of this class.

Project Topic

Going forward with what I wrote above, I think one area that might be good to investigate is the operator syntax for complexity classes, starting with polynomial hierarchy. It would be very interesting to cast PH as a category and interpret the properties of that category. Perhaps some subset of PH forms a symmetric monoidal category, or a cartesian closed category, or even a compact closed category! Who knows? But this type of information might yield insight into how complexity classes relate to other domains of science.

Meta-cognition

> What concepts have you learned so far and how comfortable do you feel with them? What questions do you still have about them?

I'm pretty comfortable with the definitions of P, NP, and coNP. The recent focus on PH has put that into perspective. I still struggle a bit with coNP, as I'm not sure what the universal witness shows. I mean if it's universal, why can't I just ignore it? If every poly-size witness works, then I don't see how it's helping me verify anything.

I'm also feeling a lot more comfortable with circuit families. At first I was a little suspicious because arguments involving an infinite number of arbitrary choices

remind me of the C in ZFC and that bothers me a bit. But I understand now the historical significance of these tools because initially it wasn't clear which theory of computing would be mainstream.

The big area I'm really struggling is the formal logic. I feel like I can hand-wave my way through a lot of proofs, but there's always something nagging at me that it might all fall apart when I try to express a machine-verifiable proof.

> What proof techniques have you attempted (successfully or not) or learned/relearned? Where else might such proof techniques apply?

Simulation is one. Being able to show that one machine is strictly more powerful than another by showing that the first can simulate the second in poly-time has shown up a lot. This reminds me a lot of abstraction refinement in abstract interpretation, where one refinement is shown to be at-least-as expressive as another if it anything that is true in the second is true in the first.

Another technique is "pushing the work onto someone else" as in the form of circuit families, TMs with advice, and tapes with strange topologies. This definitely has some real-world applications in hierarchically structured social organizations. (Yes this one is a joke.)

> What have you found most and least exciting about the course so far? Most and least valuable?

This is a tough question. I've enjoyed most of what we've talked about.

I'm probably least interested in the non-constructive proof techniques. That just doesn't vibe with the way I prefer to do math.

I'm most interested in the syntactic techniques that let us use "shorthand" for proofs, like the operators in PH. I think it's easier to learn things the short way and then dig into the full proofs when something interesting is happening.

Most valuable is probably PH. This put NP and coNP in context for me and demonstrated that complexity classes can form inductive data types, which is very relevant to research in programming languages and verification.

Least valuable... again picking on non-constructive families of machines. Like sure, I could just design a machine to solve every instance of the problem, but that doesn't tell me anything about how to do it, so why is it useful?

> How is your learning log going? What can I do to help you make the process more smooth or more valuable?

My learning log is mostly the problem sets I've been turning in. I'm trying to be a little more verbose than I would if I were just turning something in for feedback. I'm also considering revising these as we progress through the material and I get more context.

I'm managing this in a get repo and hosting it publicly on GitHub.

Meta-ungrading

I think I'm comfortable assiging myself a grade for this class. I like that I am accountable to my own learning needs and don't have to do all this extraneous thinking about how someone else is going to perceive my work.