“Math” Section 5 + 6

10/25/2010

Abstract

These notes cover section 5, which reviewed course material in preparation for the midterm, and section 6, where we discussed interactive and zero knowledge proofs.

Contents

1 Midterm prep problems

1. Prove $\{a^n b^2 n\}$ is not regular.
2. Consider two strings in a regular language. Both can be decomposed into xyz by the pumping lemma. We denote the first string's decomposition $x_1 y_1 z_1$ and the second's $x_2 y_2 z_2$. Is $|y_1| = |y_2|$? Why or why not? Why does the pumping lemma not bound the size of z? Does this suggest the possibility of a more general pumping lemma?
3. Prove $L = \{a^n \mid n \text{ is a prime}\}$ is not a regular language. It may be useful to know the distance between two consecutive primes is roughly $\ln n$, where n is the length of one of the primes in question.
4. "A grand hotel." Arriving at a hotel with an infinite number of rooms, each with a unique natural number, you're initially dismayed to hear all the rooms are taken by guests. Assuming no guests are willing to share a room, how do you tell the hotel manager to make room for you to stay? Now consider the hotel is still full but some of your friends are coming to town and would like to stay in it, too. How many friends can the hotel accomodate, and what do you tell the hotel manager to do to fit them all?
5. A friend informs you they're studying "the smallest number which cannot be expressed in fewer than fourteen English words." Which number is your friend studying?
6. With a partner, create a question which requres students to build a DFA.

2 Some last questions on context free languages and early decidability

1. Show the language of palindromes over $\{0, 1\}$ with an equal number of zeros and ones is not context free. What about the language which doesn't require an equal number of zeros and ones?
2. Make the pumping lemma stronger by proving both v and y must be nonempty.
3. Consider a "prefix closed," infinite, context-free language. Prove this language contains an infinite regular subset.
4. With a partner, create a language to be recognized by a PDA which is interesting.
5. Consider the language of $L = \{a^n \mid n \text{ is the smallest number such that the twin primes conjecture doesn't hold true}\}$. Is this language decidable? The twin prime conjecture: there are infinitely many primes p s.t. p + 2 is also prime.

3 Complexity review problems

1. Prove NP is in PSPACE.

A: We can generate and test every polynomial size warrant.

2. Prove coNP is in P^{NP}

   A: We'll show TAUTOLOGY can be decided by P with an NP oracle. First we take the 3-SAT expression and ask if there is a solution. If there isn't, return 0. Then we negate the 3-SAT expression and query the oracle again to decide if there is a solution. If there is, return 0. Return 1.

3. Prove if P = NP, NP = coNP

   A:

   $$P = NP$$

   $$L \in P \leftrightarrow \bar{L} \in P$$

   (P is closed under complement)

   $$L \in NP \leftrightarrow \bar{L} \in NP$$

   (so is NP now)

   $$coNP = NP$$

   (since coNP is defined as the complement of NP)

4. Prove the union of two decidable languages is decidable.

   A: Create a NTM with epsilon transitions to the TMs for each of the decidable languages as its start state.

5. We know the halting problem is undecidable. Is it also undecidable whether a program will output a particular string?

   A: Yes it is. Replace halts without outputting this string then halting.

4 An "early" introduction to complexity

Complexity is a vast research field which often plays fast and loose with core concepts—this requires our understanding of these concepts be immaculate to avoid confusion. We'll spend considerable time reviewing basic definitions and, like any mathematical system, when dealing with complexity concepts referring back to base definitions will be your most powerful tool—so remember them.

We are foreshadowing the class but we need to describe these basic concepts to have a discussion. The most commonly heard of complexity class is P. P is a class of languages or decision problems whose membership is always decidable in time polynomial in the length of the input. This is usually written as algorithms of the form $O(n^k)$. So let's be clear. These are decision problems or alternatively problems where we're deciding if a particular string is in a language. The output of these functions or algorithms is just zero or one. Questions like, "what is the prime factorization of n?" are not in P by definition.

There are many other classes, NP, P#, ... We will not discuss BQP, the quantum class similar to P, but we may discuss randomized classes like

BPP, advice classes like P/poly, interactice classes like IP and definitely nondeterministic classes like NP.

NP is the class of languages decidable in worst case polynomial time with a string of polynomial size in the input acting as a warrant, witness or proof. More formally, a language is in NP if there EXISTS a warrant polynomial in the size of the input where a polynomial time TM considering the string and the warrant together decides the language. This is not random, it's possibly much more powerful.

Later we'll describe languages like 3SAT in more detail, but 3SAT or 3CNF is the canonical NP-complete problem, which means it's in NP and that every language in NP "reduces" to an instance of this language in a log space or polynomial time reduction. Later we may discuss some details of reductions.

5 Nondeterminism, randomness, verification and advice

(Section 6 started here)

Last week we briefly mentioned the classes P and NP. NP is the class of languages where a deterministic polynomial time machine can decide membership with access to a polynomial-size string dependent on the input string s.

Q: Why doesn't the string simply state where a string is or isn't in the language?

A: Because the DPTM must DECIDE membership in polynomial time. It cannot be fooled.

This is distinct from randomness, which will review later. Randomness can also be considered an additional string representing coinflips (1 = heads, 0 = tails), but the string is generated according to a distribution. While NP demands the existence of these strings, randomized complexity classes will demand a high percentage of all strings (under the distribution) result in the correct answer for a problem.

Today we're looking at verification. Verification involves ideas from both nondeterminism and randomness. A prover with unbounded power will attempt to prove something to a verifier such that with high probability they will be successful if it's true or unsuccessful if it's not.

In the future we may look at advice, which is distinct from these prior concepts. Advice is a trustable string but is limited in that it's defined relative to the length of a string. That is, if we could tailor advice to every string uniquely advice WOULD solve all our problems, but if we tailor it to every string of length 5 it may not be able to. Any unary problem, however, is decidable with advice, including undecidable problems like the halting problem. The advice analogue of P, NP, BPP, AM, etc. is P/poly.

6 AM and coAM

Arthur-Merlin protocols have a polynomial time verifier, Arthur, and unbounded prover, Merlin, with public randomization. Merlin would like to prove something to Arthur, but Arthur needs to verify the proof is correct in a finite number of rounds.

Completeness If x is in the language, Arthur accepts with probability greater than or equal to $\frac{2}{3}$.

Soundness If x is not in the language, Arthur accepts with probability less than or equal to $\frac{1}{3}$.

The class MA of protocols has Merlin sending Arthur a string and Arthur deciding whether to accept or reject. This is similar to NP, which accepted a warrant, except Arthur can use randomization. The coins are flipped only after Merlin's message. NP $\subseteq$ MA is clear.

AM has Arthur flip coins, Merlin respond with a message and then Arthur deterministically verify the result.

Interesting facts: this protocol can simulate any number of additional rounds, too. The private coin and public coin versions decide the same languages.

Open questions: Is AM different from MA?

7 IP

IP, or interactive polynomial time, also has a prover and verifier, and completeness and soundness are the same as in AM. Here the number of rounds is polynomial, however. Since public coins are as powerful as private coins we can consider this a private coin technique, and it turns out that IP = PSPACE (which is in Sipser).

Here is a broad proof intuition for that statement. First, IP is a subset of PSPACE. For every set of coin tosses consider all possible responses by Merlin to Arthur, and simulate Arthur as before. This allows us to know the likehilood Arthur will accept if we're careful with the accounting, since for every set of cointosses there should be some response on at least 2/3rds of these Merlin can give which makes Arthur accept.

The other direction shows an IP protocol for TQBF, but that would require what defining TQBF is, so take a look in Sipser if you're curious (or we may return to it later).

Interestingly, a quantum verifier is no more powerful than a determinstic verifier with regards to interactive proofs.

8 Zero knowledge proofs

Big idea: Prove you know something without revealing what it is.

Fundamental caveat: Demonstrate such that it's unlikely you don't know.

Thee properties:

Completeness If the statement is true the honest verifier will be convinced by an honest prover.

Soundness If the statement is false no cheating prover can convince the verifier it's true all of the time.

Zero-knowledge No cheating verifier learns more than the statement is true.

What's a simple example? Consider you develop a polynomial solver for problems in NP, like 3-SAT and factoring. This is incredibly valuable and you'd like to sell your solver without revealing its secrets. To prove you the capability without giving it away you solicit a set of difficult problems and return their answers in polynomial time. If you were a liar you might be able to do this by guessing which problems would be returned or only being able to solve some portion of problems in NP in polynomial time, but as we test your solver more and more this becomes increasingly unlikely. The verifier only observes the time, not how your device works, so the proof is zero knowledge, and if you do have this technique you'll succesfully complete every task so it's complete.

Here's a more complete and intuitive example for this class of problems, which Wikipedia suggested: knowledge of a Hamiltonian cycle. The prover commits to an isomorphism of the original graph and the verifier requests either proof it's an isomorphism or proof this new graph contains a Hamiltonian cycle. Note it can't compute either on its own because this may take longer than polynomial time. If the prover is cheating then one of these must be false, so half the time they're caught. If not cheating they're always correct. Now to verify it's zero-knowledge we construct a simulator. A simulator is something the verifier can create that provides the same distribution as answers as the prover—this shows the proof is zero knowledge. If the verifer would ask for proof it's an isomorphism he can generate his own, and if he asks for proof there's a Hamiltonian cycle in the graph he simply generates a graph with a Hamiltonian cycle. Done.

An interesting crypotgraphic problem is the prover demonstrating knowledge of a quadratic residue mod n. For this proof we only need to know that a quadratic residue mod n is a number x where $x = y^2 \mod n$, and the product of two quadratic residues is also a quadratic residue, the product of two non-residues is a residue, and a residue and a non-residue is a non-residue (the same as adding even and odd numbers). It is believed hard to determine if a number is a quadratic residue without knowing the factorization of n. The group of quadratic residues is denoted $\mathbb{Z}_n^*$.

Proof that x is a quadratic residue mod n, without revealing w where $x = w^2 \mod n$.

$$P \to V \; y = (u \leftarrow_R \mathbb{Z}_n^*)^2 \tag{1}$$
$$V \to P \; b \leftarrow_R \; \{0,1\} \tag{2}$$
$$b = 0 \tag{3}$$
$$P \to V \; (b == 0 \; ? \; u : w \cdot u \mod n) \tag{4}$$

P accepts when $b = 0, z^2 = y \mod n$ or $b = 1, z^2 = xy \mod n$.

Intuitvely, the prover commits to a random quadratic residue, y, and the verifer requests either that the prover demonstrate y is a quadratic residue or that xy is a quadratic residue.

Formally proving this is sound, complete and zero knowledge is non trivial. We'll review at a high level.

Completeness: An honest verifier never makes a mistake since it may always run the protocol. Soundness: Consider that x is a non-residue. Then either xy is a non-residue or y is a non-residue. However, half the time the verifier selects which one of these has a mismatch. Zero-knowledge: To prove something is zero knowledge requires us to be able to simulate the "proof" without the prover and obtain the same distribution over outcomes. This shows the verifier couldn't have obtained any additional information. Do to time we won't discuss this.

Q: Is the half chance of catching a cheating prover a problem?

A: No. By repetition we can boost this arbitrarily. For k rounds the probability of success of a cheater is 2^{-k}.