IAS/PCMI Summer Session 2000
Clay Mathematics Undergraduate Program
Basic Course on Computational Complexity

Lecture 13: Interactive Proofs

David Mix Barrington and Alexis Maciel
August 2, 2000

1. Overview

We have now explored a number of prover-verifier scenarios, defining a number of complexity classes. With polynomial-time computation, Bob can decide membership in a P language on his own. With a clever Alice to supply him a proof, he can be convinced of membership in an NP language. With randomization, he can decide languages in BPP. We've also seen the power of interaction between a clever Alice and a clever Bob, which can define any language in AP = PSPACE. What about interaction between a clever Alice and an ordinary Bob? Here we see that allowing Bob randomization has a great effect on the power of the proof system. When Bob can pose random questions and get clever answers, he can be convinced of membership in any language in the class IP. In our next lecture, we will see that IP is actually equal to PSPACE, so that these *interactive proofs* can define all the same languages as can alternating machines in poly-time.

- To provide our first example of an interactive proof, we consider the problem of telling whether two graphs are *isomorphic*. The language GRAPH-ISO of pairs of isomorphic graphs is in NP, because Alice can prove the two graphs isomorphic by presenting an isomorphism. We show that Alice can also prove two graphs to be *not* isomorphic, using a poly-time *interactive proof.*
- We formally define interactive proofs and the class IP of languages for which there exists a poly-time interactive proof system. We consider the relationship between IP and other classes, proving that IP is contained in PSPACE.
- We discuss the notion of a *zero-knowledge proof* and its potential uses in communication.

- We give an example of a zero-knowledge proof for 3-colorability of graphs, where Alice can convince Bob that a graph is 3-colorable, but Bob at the end of the interaction knows no more about a potential 3-coloring than he did before. This proof uses a *cryptographic assumption*, that the RSA cryptosystem is secure.

2. Graph Isomorphism

Let G and H be directed graphs, each with n vertices. An *isomorphism* from G to H is a one-to-one, onto function f from the vertices of G to the vertices of H such that (x, y) is an edge of G iff $(f(x), f(y))$ is an edge of H. If such an f exists we say that G and H are *isomorphic*.

How hard is it to decide the language GRAPH-ISO, consisting of the pairs of graphs $\langle G, H\rangle$ such that G and H are isomorphic? No poly-time deterministic algorithm is known. GRAPH-ISO is pretty clearly in NP, because Alice could send Bob a description of f and Bob could then check the isomorphism condition on each of the n^2 possible edges. For technical reasons, GRAPH-ISO is thought not to be NP-complete, and many people would not be too surprised were it to be found to be in P (though this would be an exciting result).

Consider the following protocol between a clever Alice, who wants to convince Bob that a pair of graphs G and H are *not* isomorphic. There is no obvious proof of this fact she could send. (It's possible that G and H differ on some easily verifiable property that is invariant under isomorphism, like the number of edges. But no set of such properties are known that characterize isomorphism classes, so it is possible that two non-isomorphic graphs agree on any particular set of such properties.) Bob will be assumed to have poly-time computation *and randomization* available to him. We will also assume that, clever though she may be, Alice cannot see the results of Bob's coin flips. (This assumption can be removed due to a theorem of Goldwasser and Sipser.)

Bob begins the protocol by flipping a private coin c. He then flips enough coins to choose a *random permutation* σ from the uniform distribution of permutations of n elements. Then he sends a graph to Alice. If c was heads, he sends $\sigma(G)$, otherwise he sends $\sigma(H)$. Alice then sends back a bit telling Bob which graph he permuted to get the graph he sent her. Bob accepts iff she is right. In general, they play k rounds of this protocol and Bob accepts iff Alice is right every time.

This is a fairly familiar sort of experiment. If someone claims to be able to tell Coke from Pepsi and you think they can't, you might well subject them to a blind taste test. If they correctly identify the Coke 100 times in a row, you will begin to think that they are right (or are successfully cheating). Of course, you wouldn't be silly enough to put the Coke to your left every time — you would flip a secret coin to decide which side to put it on.

Why should Bob be convinced in the case of the graphs? Consider the two random variables $\sigma(G)$ and $\sigma(H)$, each producing an n-vertex graph. If G and H are not isomorphic, these two distributions are disjoint, in that no graph appears in both with nonzero probability. (If a single graph I were isomorphic both to G and to H, then G and H would be isomorphic to each other. This fact is too easy to put in the exercises.) Thus a clever Alice, presented with one of these graphs, can easily tell which distribution it came from.

But what if G and H are isomorphic, by some permutation τ from G to H? In that case a graph I could be equal to both $\rho(H)$ and to $\rho(\tau(G))$. It will appear as $\sigma(G)$ exactly if the random σ happens to be $\rho \circ \tau$, an event with probability $1/n!$. It will appear as $\sigma(H)$ exactly if the random σ happens to be ρ, another event of probability $1/n!$. The two distributions are exactly the same! (Technically, we should probably consider the possibility of *automorphisms* of G, so that a single graph I could occur in the distribution as a result of several different permutations of G. But the number of times I occurs as $\sigma(G)$ is the same as the number of times it occurs as $\sigma(H)$.)

If G and H are isomorphic, Alice is presented with a random graph from a fixed distribution which does not depend on Bob's initial coin flip. Clever though she is, she has no way to tell how Bob produced the graph, and she can do no better than to guess randomly. If they play k independent rounds of the protocol with an isomorphic G and H, her chance of fooling Bob is only $1/2^k$.

3. Interactive Proof Systems and IP

Let's now generalize from this particular proof to define an *interactive proof system*. We have a protocol governing how Alice and Bob should send messages to each other. Alice is clever, and Bob is limited to poly-time computation except for his random choices. When all messages have been sent (a polynomial number of message bits in all) the decision as to whether the proof is convincing is made by a deterministic poly-time algorithm. If the input is in the language, *there exists* a proof that Bob will definitely find convincing. If the input is not in the language, *no possible proof* will convince Bob with more than a certain probability.

Just as with RP and BPP, we have a choice of two possible definitions of interactive proof systems, a "one-sided error" definition where we require valid proofs to be absolutely convincing, and a "two-sided error" definition where Bob has a small chance of being convinced by an invalid proof. Later we will show that poly-time one-sided error proof systems exist for all languages in PSPACE, so the weakening of the class is only apparent and the two definitions lead to the same poly-time complexity class. Later, with probabilistically checkable proofs, it will be convenient to insist on the one-sided definition. It's also true that all current interactive proof systems seem to operate in a similar way: Alice makes a number of statements, often answers to Bob's questions, and Bob checks them for some sort of internal consistency. If Alice is proving a true statement, it is generally no problem for her to give the true answers to Bob's questions, which are consistent. The trick in designing the protocol is to prevent her from convincing Bob of a false statement, by ensuring that there are few or no sequences of answers that are consistent in the case that Bob should not be convinced.

For now, however, it is convenient to use the two-sided error definition. We define IP to be the set of languages A such that a protocol exists for a clever Alice and a randomized Bob where:

- if $x \in A$, there exists a proof strategy for Alice that convinces Bob with probability at least $2/3$, and
- if $x \notin A$, for any proof strategy by Alice, the probability that Bob is convinced is less than $1/3$.

We can see immediately that IP contains both NP and BPP. An NP prover-verifier system fits into the IP framework, with Bob asking no questions and flipping no coins. A BPP algorithm can be thought of as a proof system, where Alice contributes no input and Bob just runs his randomized algorithm. (It is true, but harder to prove, that BPP $\subseteq$ IP in the one-sided error version as well — of course this will follow from PSPACE $\subseteq$ IP.) The GRAPH-NON-ISO problem is not known to be in either NP or BPP. This was one of the original facts motivating the study of IP — how much bigger than NP and BPP could it be?

There are limits on the power of IP, as it is not too hard to prove that:

Theorem 1 *IP $\subseteq$ PSPACE.*

Proof Let $A \in$ IP have an interactive proof system and consider the following recursive algorithm that provides an optimal strategy for Alice. At any point in the protocol, the algorithm will evaluate the probability that Bob will be convinced given optimal play on Alice's part. If this is a final configuration, we run the poly-time referee to get a probability of one or zero. If it is a Bob-move configuration, we compute the probability of each of Bob's moves by simulating Bob, recursively evaluate the probability of Bob accepting in each of the resulting configurations, and take a weighted sum of these probabilities. If this is an Alice-move configuration, we recursively evaluate the probability of Bob accepting in each of the successor configurations and find the maximum.

The action of this algorithm at any point uses polynomial space, and the recursion is to only polynomial depth because there are only polynomially many moves in any run of the protocol. (We assume that there is a clock, and Bob rejects if time runs out.) So the overall algorithm is in PSPACE. At the end of a run of this algorithm with the start configuration as input, we know Bob's probability of acceptance given optimal play by Alice, so we know whether $x \in A$. $\square$

4. Zero-Knowledge and Its Uses

There are many applications when you want to prove some fact to somebody, but not give them the capacity to prove the same thing to someone else. Giving your credit card number to a vendor is usually sufficient to prove to their satisfaction that you are authorized to use the account with that number. Usually this works, but in principle it is a terrible solution to the problem, because the vendor (or their phone operator) now knows your credit card number and could possibly use it for their own gain, "proving" *them* to be an authorized user of your account.

Dealings between humans over a telephone tend to have a certain level of protection against such nefarious activities. (For example, large companies record their operators' business conversations and could fairly easily detect their use of a customer's account.) But dealings between machines are more problematic, because machines often do what they are "told to do" without question. Certainly banks must think about the possibility of an evildoer recording and duplicating the signal that tells an automatic teller machine to dispense cash. It would be nice to have a technological solution to this problem, that would be usable in electronic communication.

A possible model for such a solution is the *zero-knowledge proof*. This is an interactive proof as defined above, with one additional property. After Alice proves to Bob that $x \in A$, Bob should know *nothing* new about x other than the fact that it is in A. There are several possible formal definitions of this, but one popular one says that Bob's conversations with Alice, as a random variable, come from a distribution that Bob could sample from himself without input from Alice. An example of a zero-knowledge proof should help to make this definition clearer.

5. A Zero-Knowledge Proof for 3-Colorability

First note that we need some sort of unproven assumption to make zero-knowledge proofs work, because our knowledge about complexity classes is insufficient to rule out possible worlds in which they don't and can't. (If P = PSPACE, for example, interactive proofs are only possible for languages in P, making the definition of zero-knowledge at best problematic.) In this example we assume that the *RSA cryptosystem* is secure, an unproven but widely believed hypothesis. (Though note that a practical implementation of Shor's quantum factoring algorithm would break RSA.)

Describing RSA is outside our scope here, but we should say what me mean by the system being secure. Using RSA, Alice (for example) can encrypt a message using a public key e, and unless the system is broken, no one can decrypt it, or even obtain partial information about it, without using her secret key d. In our interactive protocol, Alice will use RSA to irrevocably *commit* to a choice without revealing it. An important property of RSA that we will need is that Bob can recognize d as the proper decrypting key for e, so that Alice cannot change her mind about what she has encrypted once she has sent it to Bob.

Suppose we have a three-colorable graph G with n nodes and Alice knows a three-coloring c, a function from the nodes of the graph to $\{1,2,3\}$. To prove to Bob that G is three-colorable, she first randomly picks a permutation σ of the set $\{1,2,3\}$, from the uniform distribution. She then chooses n RSA encryption keys e_i, and sends Bob the message $\langle e_1(\sigma(c(1))),\ldots,e_n(\sigma(c(n)))\rangle$. Bob then chooses an edge (i,j) uniformly from the edges of G. Alice sends Bob d_i and d_j. Bob uses these to determine $\sigma(c(i))$ and $\sigma(c(j))$, and accepts iff these are different.

We must prove three things: that this protocol is *sound*, *complete*, and *zero-knowledge*. Soundness means that it cannot prove false things with high probability. Since Alice must send an encrypted coloring of G, if G is not three-colorable she must send a coloring where at least one edge connects two nodes with the same color. With probability at least $1/n^2$, Bob will choose this edge and reject. By a polynomial number of repeated trials, the probability of false acceptance can be made very low.

Completeness means that Alice can prove three-colorability of any graph that is really three-colorable. This follows in the usual way, because Alice can send an encryption of an actual coloring if one exists and there is *no* chance of Bob rejecting it.

The zero-knowledge property relies on the security of the encryption. If we assume that Bob cannot gain any information from Alice's messages other than the decrypted colors, we can show that he learns nothing about G other than the fact that it is three-colorable. In a repeated set of trials in which Bob accepts, he sees two distinct colors from $\{1, 2, 3\}$ on each trial. Furthermore, he is *equally likely* to see any of the six possible sequences of two distinct colors, since whatever the original colors were, the six possibilities for σ permute them to each of the six possible sequences with equal probability. Bob thus sees a sequence of independent samples from the uniform distribution on six elements, a distribution he could certainly sample himself.

6. Exercises

1. Suppose you want to choose uniformly from among m options, and m is not a power of two. You are given only fair two-sided coins. Show that you can do this in time $O(\log n)$, where you are allowed to fail with up to a fixed constant probability $\epsilon > 0$.

2. Suppose that G and H are undirected graphs with no cycles (forests). Show that Bob can determine, in deterministic poly-time with no help, whether G and H are isomorphic.

3. The COLOR-ISO problem takes as input two graphs G and H, where each vertex in G and H has a label (color) from the set $\{1, \ldots, n\}$. A color isomorphism f must take a vertex x in G to a vertex $f(x)$ of the same color in H. Show that COLOR-ISO $\leq_P$ GRAPH-ISO.

4. Suppose that Bob is limited to deterministic poly-time computation and that Alice is clever, but that Alice and Bob may exchange a polynomial number of messages rather than just one from Alice to Bob. Show that proof systems of this type define only languages in NP.

5. Suppose IP were defined so that Bob's accepting probability was merely known to be above 1/2 for input $x \in A$ and an optimal proof, and below 1/2 for $x \notin A$ and any proof. Show that this seemingly more powerful IP class, which we might call IPP, would still be contained in PSPACE.

6. Is the interactive protocol we presented for GRAPH-NON-ISO a zero-knowledge protocol? Why or why not?

7. Suppose that $A \in$ BPP, and we have a probabilistic algorithm that determines whether $x \in A$, for an n-bit x, in poly time with a probability of error that

is at most 2^{-2n}. That is, there is a poly-time language B such that if $x \in A$, $\langle x, y \rangle \in B$ for all but a 2^{-2n} fraction of the possible y's, and if $x \notin A$, $\langle x, y \rangle \in B$ for only a 2^{-2n} fraction of the possible y's. Show that there exists a single string z such that $x \in A$ iff $\langle x, z \rangle \in B$. (The string z is able to "derandomize" the randomized algorithm.)

8. Let A be any language in BPP. Define a protocol where Alice proves $x \in A$ by sending Bob the string z constructed in the previous problem and letting Bob test whether $\langle x, z \rangle \in B$. Alice has proved membership in A to a poly-time deterministic Bob. Have we proven that BPP $\subseteq$ NP?