Today we talk about interactive proofs. Usually we think of proofs as immutable objects: A proof is something you read in a book, think about, and decide if it is correct or not. But suppose you are suspected of a murder and want to prove your innocence in court. Instead of submitting a written proof of your story, you are subjected to a cross-examination of lawyers at the end of which a jury will decide if you are innocent or guilty. How should you go about preparing your case?

1 Interactive proofs

To explain interactive proofs, let's go back to our definition of NP. A decision problem L is in NP if there is a polynomial-time verifier V and a polynomial p such that

$$\begin{aligned} &\text{if } x \in L, && \text{then there is a } y, |y| \leq p(|x|) \text{ such that } V(x,y) = 1, \text{ and} \\ &\text{if } x \notin L, && \text{then for all } y, |y| \leq p(|x|), V(x,y) = 0. \end{aligned}$$

So far we have been thinking of y as a *witness* or *certificate* for L: This is the satisfying assignment for a boolean formula, or the perfect matching in a graph. Today we think of it a bit differently: Instead of thinking of the witness as something intrinsic to the problem, we think of it as being furnished by an external entity, a *prover*. So we can think of NP verification as the following kind of process: On input x, the *verifier* V asks to see a proof that $x \in L$. The prover tries to provide such a proof. If the prover is honest, it will always provide a witness y when $x \in L$. However, the prover may try to cheat by providing a bogus proof when $x \notin L$: For instance, given a CNF ϕ, it can try to give an assignment a such that $\phi(a)$ is false. The job of the verifier is to look at the prover's claimed proof and decide if $x \in L$ or not.

Formally, an NP-*prover* P is any (computationally unbounded) function that maps inputs $x \in \{0,1\}^*$ to proofs $y \in \{0,1\}^*$. Then NP is the class of all decision problems L for which there exists a TM V (called a *verifier*) whose running time is polynomial in the length of x and a computationally unbounded TM P: $\{0,1\}^* \to \{0,1\}^*$ (called the *honest prover*) such that

$$\begin{aligned} &\text{if } x \in L, && \text{then } V(x, P(x)) = 1 \\ &\text{if } x \notin L, && \text{then for all TM } P^*, V(x, P^*(x)) = 0. \end{aligned}$$

Now consider the following extension of NP-proofs: After receiving the purported proof, the verifier is not quite convinced that the prover is correct and asks to see more detail. The prover may then send a new message to the verifier elaborating on his case. The two keep going back and forth until at the end of the day, the verifier is either convinced that the proof is correct (and accepts), or thinks the whole argument is bogus (and rejects).

Notice that in this setting the prover and verifier are *adaptive*: The question that the verifier asks at any given round of interaction may depend on the answers it received from the prover in previous

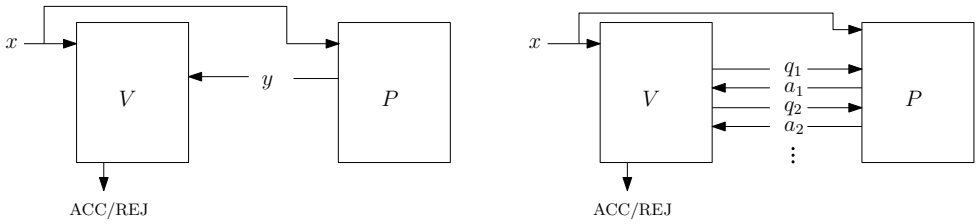

Figure 1: Introducing interaction in proofs

rounds. However, the prover himself may choose to adapt his answers based on the previous queries made by the verifier.

To formalize it we need to introduce *interactive Turing Machines*. This is a Turing Machine that, in addition to its input x, receives additional inputs $(y_1, z_1, \ldots, y_k, z_k)$ which represent the messages sent and received in the first $2k$ rounds of interaction. Here, $y_1, \ldots, y_k$ are the messages sent by the machine itself, and $z_1, \ldots, z_k$ are the responses received from the other party. On this input, the machine produces the next message y_{k+1} or possibly accepts/rejects.

Given two interactive Turing Machines A and B we define the interactive computation of (A, B) in the natural way: On input x, $A(x)$ outputs y_1, $B(x, y_1)$ outputs z_1, $A(x, y_1, z_1)$ outputs y_2, and so on, until A accepts or rejects.

A *(polynomial-time) deterministic interactive proof* for a decision problem L is a pair of Turing Machines (V, P) where V runs in time polynomial in the input x, and

$$\begin{array}{ll} \text{if } x \in L, & \text{then } (V,P)(x) \text{ accepts} \\ \text{if } x \notin L, & \text{then for all TM } P^*,\ (V,P^*)(x) \text{ rejects.} \end{array}$$

The verifier's messages are called *questions*, and the prover's messages are called *answers*.

Clearly the model we just introduced is at least as powerful as NP, which requires no interaction. But is it really any more powerful? A simple argument shows that it is not, at least in the case when V is deterministic. The reason is that on input x, the prover can predict in advance which questions the verifier is going to ask, so it can answer all of them in the first round. Therefore the whole interaction can be emulated by a one-round interaction, and the decision problem in question is in NP. (See Figure 2.)

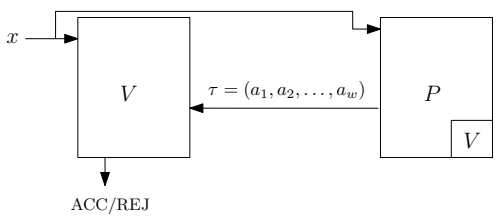

Figure 2: Deterministic verifiers are just like NP

2 Interaction and randomness

Now let's supposed that the verifier is randomized. In this case, it is no more the case that the prover can predict the verifier's questions. It turns out that this model is a lot more interesting.

Definition 1. A *(polynomial-time) interactive proof* for a decision problem L is a pair of Turing Machines (V, P) where V is a randomized TM that runs in time polynomial in the input x, and

$$\begin{aligned} &\text{if } x \in L, && \Pr[\text{then for all TM } P^*,\ (V, P^*)(x) \text{ accepts}] \geq 2/3 \\ &\text{if } x \notin L, && \Pr[\text{then for all TM } P^*,\ (V, P^*)(x) \text{ accepts}] \leq 1/3. \end{aligned}$$

An *r-round interactive proof* is one in which the prover and verifier exchange at most r messages on any input and any setting of the randomness.

As usual there is nothing special about the constants 2/3 and 1/3, by the same argument we had for BPP. However there is one subtlety: Repeating the protocol several times may increase the number of rounds. It turns out that what we can do instead is repeat the protocol *in parallel*, and amplify the gap in probabilities without affecting the number of rounds.

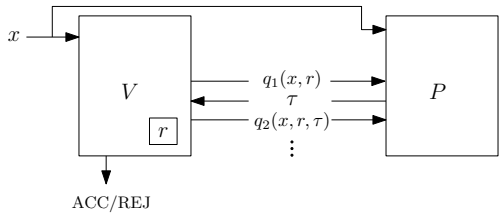

Figure 3: An interactive proof

We now give an example of a problem that has a two-round interactive proof, but is not known to be in NP. Two graphs G_1 and G_2 on n vertices are *isomorphic* if there is a permutation of the vertices of G_1 so that, after permuting them, they both look the same. Formally, the permutation π has the property that (u, v) is an edge of G_1 if and only if $(\pi(u), \pi(v))$ is an edge of G_2. Look at the following question:

GI (GRAPH ISOMORPHISM): Given a pair of graphs (G_1, G_2), are they isomorphic?

There are some obvious tests you can try for isomorphism, for instance checking that they have the same number of edges and the same degree sequence (i.e. they have the same number of vertices of any given degree). However it should be easy to convince yourself that two graphs can pass these tests and still not be isomorphic. There are more sophisticated tests like checking that G_1 and G_2 have the same eigenvalues. Yet it turns out that they can still be non-isomorphic.

On the other hand, GI is clearly in NP – the isomorphism permutation can serve as a witness. But what about the opposite problem:

$\overline{\text{GI}}$ (GRAPH NONISOMORPHISM): Given a pair of graphs (G_1, G_2), are they *non*-isomorphic?

While $\overline{\text{GI}}$ is not known to be in NP, there is a simple two-round interactive protocol for it:

Interactive proof for graph non-isomorphism

On input (G_1, G_2):

V: Choose a random $i \in \{1, 2\}$ and a random permutation π on n elements. Create a graph G by applying π to the vertices of G_i and permuting its edges accordingly (i.e., $(\pi(u), \pi(v))$ is an edge in G iff (u, v) is an edge in G_i). Send G to the prover.

P: Answer 1 if G is isomorphic to G_1 and 2 if G is isomorphic to G_2.

V: If the prover answered i accept, otherwise reject.

We now argue that the above is indeed an interactive proof for $\overline{\text{GI}}$. If G_1 and G_2 are not isomorphic, then G is isomorphic to G_i, so it cannot be isomorphic to the other graph. So the honest prover will always answer i, and the verifier will always accept. Now assume G_1 and G_2 are not isomorphic. We have to argue that no matter what the prover P^* answers, V will reject with probability at least $1/2$. The key insight is that when G_1 and G_2 are isomorphic, the random graph G is (statistically) independent of the random value i. Therefore no matter what the prover does, the chances that he guesses the correct value of i is exactly $1/2$.

3 Round reduction and public-coin proofs

So we seem to be getting evidence that interaction helps us prove more things. At this point the situation looks a bit like what we had for the polynomial-time hierarchy: We have an example of a problem (graph non-isomorphism) that has a 2-round interactive proof, but no known non-interactive proof.

It seems reasonable to think that if two rounds of interaction are more powerful than no interaction, then three rounds should be even more powerful than two rounds. However, it turns out that this is not the case:

Theorem 2. *For every (constant) r, if L has an r-round interactive proof, then it has a 2-round interactive proof.*

To prove this theorem, it turns out it is convenient to first convert the interactive proof in a special form. To explain what this special form is, let's look back at our example of graph non-isomorphism. The reason the verifier is convinced that G_1 and G_2 are not isomorphic after running this protocol is that the if they were isomorphic, then the value i chosen by the verifier at the very beginning is completely hidden from the prover. This is an example of a *private coin* protocol: The soundness of the protocol hinges on the fact that the verifier has some private (random) value that the prover cannot guess.

We can also consider a more restricted kind of protocol, where everything the verifier does is in plain view of the prover. Without loss of generality, this kind of protocol, called a *public coin* protocol, works in the following way: The prover and the verifier get together in a room and toss

some random coins together. The coin tosses serve as the verifier's first question. The prover then answers this question, and they get together again, toss some coins, and this is the second question. After sufficiently many rounds of interaction the verifier has to accept or reject.

This seems like a very strange thing to do – how much can the verifier learn by asking random questions? But here is an example of an interesting promise problem that can has a two-round public-coin proof:

Input: A circuit $C\colon \{0,1\}^n \to \{0,1\}$ and a number $0 \le s \le 2^n$.
Yes instances: (C, s) such that C has at least s satisfying assignments.
No instances: (C, s) such that C has at most $s/8$ satisfying assignments.

This problem has a very simple protocol: In the first round, the prover and verifier choose a random hash function $h\colon \{0,1\}^n \to \{0,1\}^i$, where $2^{i-1} \le s \le 2^{i-2}$. The prover is then supposed to send x such that C accepts x and $h(x) = 0$. If such an x is sent, the verifier accepts, otherwise it rejects. By the analysis we did in the last lecture, for a yes instance such an x exists with probability at least $1/8$, and for a no instance no x exists with probability $1/16$.

While this protocol seems quite specialized, it turns out that the idea can be used to turn *any* private-coin protocol into a public-coin one. We won't show how to do this but here is the general result.

Theorem 3. *If L has an r-round interactive proof, then it has an $r+3$ round public-coin interactive proof.*

Now we sketch how to turn an r-round public-coin interactive proof for any constant $r > 2$ into a two-round public-coin proof. We will do it iteratively, reducing the number of rounds one by one. In such a protocol, the verifer starts by asking a random question r_1, then the prover answers by r_1 and the verifier responds with r_2. We will sketch how to flip the order of the second and third round of interaction without affecting the completeness and soundness of the protocol. The result is a protocol with one less round.

Let's assume that each message is k bits long, where k grows at a rate polynomial in the input size.

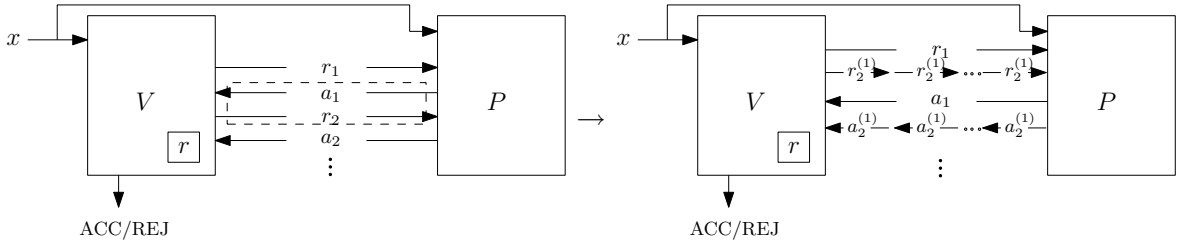

Figure 4: Reversing a pair of question and answer to reduce the number of rounds

Consider what happens if in the third round of the protocol, the verifier "forks" m independent executions of it in parallel (we'll give the value of m later): Namely, instead of asking a single question r_2, it asks m such questions $r_{21}, \ldots, r_{2m}$ independently at random. It then expects m

answers $a_{21}, \ldots, a_{2m}$ from the prover, and so forth. At the end, it computes m different answers, and accepts if the majority of them are accepting.

To analyze what happens it helps to assume the probability of accepting for the yes instances is at least $8/9$ and the probability of rejecting the no instances is at most $1/9$. Let's look at the yes instances first. It then follows that for at least a $2/3$ fraction choices of the first message r_1, for any fixed response a_1 by the prover, the probability that the verifier accepts in the rest of the interaction is at least $2/3$. Let's fix a message r_1 with this property. Then by the Chernoff bound the probability that fewer than half of the parallel interactions accept is at most $2^{-m/6}$.

Now let's look at the no instances. These accept with probability at most $1/9$, so there is at least a $2/3$ fraction of messages r_1 such that for any fixed response a_1 by the prover, the rest of the interaction rejects with probability $2/3$. Let's fix a message r_1 with this property. Again by the Chernoff bound the probability that more than half of the forked interactions accept is at most $2^{-m/6}$.

In either case, as long as r_1 is "good", we have that for any fixed prover response a_1, the probability that fewer than half of the forked interactions do the right thing is at most $2^{-m/6}$. But there are at most 2^k possible responses a_1; so by a union bound we get that the probability that *there exists* a_1 that makes fewer than half of the forked interactions do the right thing is at most $2^{k-m/6}$. We choose $m = 6k + 18$ to make this probability as small as $1/8$.

But now look at what we proved: Regardless of what the prover's message a_1 is, the rest of the protocol succeeds with probability $7/8$. So we can now *delay* the message a_1 to come after the questions $r_{21}, \ldots, r_{2m}$. (In fact we could make it the very last message of the interaction if we wanted to.) This allows us to combine the first and second rounds of questions into a single round (and even the first two rounds of answers into a single round) and reduce the number of rounds by at least one.

So it turns out that any polynomial-time interactive proof with a *constant* number of rounds can be turned into a polynomial-time interactive proof with two rounds and public coins. There is one last simplification that we can make, giving rise to the following simplified definition:

Definition 4. The class AM consists of those decision problems that admit a two-round interactive proof (P, V) such that

$$\begin{aligned} &\text{if } x \in L, && \Pr[\text{for all TM } P^*,\ (V, P^*)(x) \text{ accepts}] = 1 \\ &\text{if } x \notin L, && \Pr[\text{for all TM } P^*,\ (V, P^*)(x) \text{ accepts}] \leq 1/2. \end{aligned}$$

So any constant-round interactive proof can be "compiled" into this very special form: The verifier asks a single random question; for a yes instance, the prover can always make the verifier accept, but for a no instance, the verifier will reject with high probability.

The shorthand AM stands for Arthur-Merlin games. This notation is inspired by the historic parable in which the fool Merlin asks wise King Arthur completely random questions, yet at the end Arthur manages to convince Merlin of some deep truth.

4 Interactive proofs, derandomization, and the polynomial-time hierarchy

After all these simplifications it is natural to ask if constant-round interactive proofs are really all that much more powerful than ordinary (non-interactive) proofs. Just as BPP could be simulated by polynomial-size circuit families, a look at the definition of AM and an application of the same argument shows that AM can be simulated by *nondeterministic* polynomial-size circuit families: A nondeterministic circuit NC gets two inputs x and y, and we say NC accepts x if there is some choice of y that makes $NC(x, y) = 1$.

This looks awfully close to NP. In fact we can do more: Recall that we showed BPP = P using the Nisan-Wigderson generator, once we were willing to make an assumption that certain circuit lower bounds hold. In fact it is possible to show that NP = AM under a stronger but still believable assumption:[1]

Theorem 5. *If there is a problem L decidable in time $2^{O(n)}$ but not decidable by nondeterministic circuits of size $2^{\delta(n)}$ for some $\delta > 0$, then* AM = NP*.*

This is quite strange: Think of the example of graph non-isomorphism. We saw this problem is in AM, but nobody knows if it is in NP. What this theorem tells us is that if we are willing to believe its assumption, then interaction is not required to prove graph non-isomorphism!

If we are not willing to make any such assumptions, we can still obtain "derandomizations" of AM within the polynomial-time hierarchy, as well as evidence that interactive proofs and interactive refutations are different. The situation is pretty similar to the one for BPP, and the results are proved in much the same way. We summarize the various results of this kind in the following table.

$\mathrm{BPP} \subseteq \mathrm{P/poly}$	$\mathrm{AM} \subseteq \mathrm{NP/poly}$
$\mathrm{BPP} \subseteq \Sigma_2 \cap \Pi_2$	$\mathrm{AM} \subseteq \Pi_2$
$\mathrm{NP} \subseteq \mathrm{BPP} \Rightarrow \Sigma_2 = \Pi_2$	$\mathrm{AM} \subseteq \mathrm{coAM} \Rightarrow \Sigma_2 = \Pi_2$

One consequence of the last relation is that graph isomorphism is unlikely to be NP-complete: Since graph isomorphism is in coAM, if graph isomorphism were NP-complete, then we would have NP $\subseteq$ coAM and thefore $\Sigma_2 = \Pi_2$.

What if we consider interactive proofs with no restriction on the number of rounds? The class of decision problems for which such proofs exist is denoted by IP. Noam Nisan showed that $\mathrm{P}^{\#\mathrm{SAT}} \subseteq \mathrm{IP}$, so this kind of interactive proofs appear to be much more powerful.

[1] In fact a somewhat weaker assumption is sufficient.