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 (promise) decision problem (YES, NO) is in NP if there is a polynomial-time verifier V and a polynomial p such that

$$\begin{aligned} \text{if } x \in YES, &\quad \text{then there is a } y, |y| \leq p(|x|) \text{ such that } V(x,y) = 1, \text{ and} \\ \text{if } x \in NO, &\quad \text{then for all } y, |y| \leq p(|x|), V(x,y) = 0. \end{aligned}$$

So far we have been thinking of y as a witness or a certificate: This is the satisfying assignment for a boolean formula, or the perfect matching in a graph. Today we will think of this witness as an object furnished by an external entity called the *prover*. Then verification can be viewed as a process: On input x, the *verifier* V asks to see a proof that $x \in YES$. The prover tries to provide such a proof. If the prover is honest, it will always be able to provide a correct proof (provided it exists). The verifier should be *complete*, namely recognize such proofs as correct. On the other hand, if the verifier tries to provide a bogus proof (for instance an assignment a such that $\phi(a)$ is false) the verifier should be *sound* and detect the mistake.

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 for which there exists a TM V (called a *verifier*) whose running time is polynomial in the length of x and a computationally unbounded function $P\colon \{0,1\}^* \to \{0,1\}^*$ (called the *honest prover*) such that

$$\begin{aligned} \text{if } x \in YES, &\quad \text{then } V(x, P(x)) = 1 \\ \text{if } x \in NO, &\quad \text{then for all } P^*, V(x, P^*(x)) = 0. \end{aligned}$$

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).

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 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. (y_1 may be empty if the other party goes first.) 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 (YES, NO) is a pair of Turing Machines (V, P) where V runs in time polynomial in the input x, and

$$\begin{aligned} \text{if } x \in YES, &\quad \text{then } (V,P)(x) \text{ accepts} \\ \text{if } x \in NO, &\quad \text{then for all } P^*,\ (V,P^*)(x) \text{ rejects.} \end{aligned}$$

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.

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.

Definition 1. A *(polynomial-time) interactive proof* for (YES, NO) is a pair of Turing Machines (V, P) where V is a randomized TM that runs in time polynomial in the input x, and

$$\begin{aligned} \text{if } x \in YES, &\quad \Pr[(V,P)(x) \text{ accepts}] \geq 2/3 \\ \text{if } x \in NO, &\quad \Pr[(V,P^*)(x) \text{ accepts}] \leq 1/3 \quad \text{for all } P^*. \end{aligned}$$

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. What we can do instead is run the protocol $p(n)$ times *in parallel* (using independent randomness), and accept if a suitable fraction of runs accept. This amplifies the probabilities from any pair of constants $c > s$ to $1 - 2^{-\Omega(n)}$ and $2^{-\Omega(n)}$, respectively. We won't give the 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_0, G_1), 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_0 and G_1 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_0, G_1), 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_0, G_1):

V: Choose a random $i \in \{0, 1\}$ 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 0 if G is isomorphic to G_0 and 1 if G is isomorphic to G_1.

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_0 and G_1 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_0 and G_1 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_0 and G_1 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. 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 f has an r-round interactive proof, then it has a two-round public-coin interactive proof.*

To prove this theorem and explain what "public-coin" means, 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_0 and G_1 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 relies 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. A *nondeterministic circuit* $C\colon \{0,1\}^n \to \{0,1\}$ is a circuit that, in addition to its regular input $x \in \{0,1\}^n$, a witness w. We say such a circuit C accepts x (or x is a satisfying assignment of C) if there exists a w such that $C(x,w) = 1$.

MSAT (MANY SATISFYING ASSIGNMENTS):
Input: A nondeterministic circuit C and numbers s (in binary) and 1^a (in unary),
Yes instances: (C, s) such that C has at least s satisfying assignments.
No instances: (C, s) such that C has at most $(1 - 1/a)s$ satisfying assignments.

MSAT is at least as hard as SAT, since if we set $s = 1$ and say $a = 2$ it asks exactly for the presence of a satisfying assignment. If we set s to say $2^{n/2}$ then the problem is not known to be in NP.

Claim 3. MSAT *has a two-round public-coin interactive proof.*

Proof. We first show how to do this when $1 - 1/a$ is replaced by $1/8$. In the first round, the prover and verifier choose a random hash function $h\colon \{0,1\}^n \to \{0,1\}^k$, where $2^{k-2} \leq s \leq 2^{k-1}$. Recall that h can be specified by a random string of length $2n$. 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 Lemma 7 from Lecture 6, for a yes instance such an x exists with probability at least $1/8$.

For a no instance, the probability that the verifier can provide an x such that $C(x) = 1$ and $h(x) = 0$ is at most the number of satisfying assignments to C times 2^{-k} by a union bound. This is at most $(s/8)2^{-k} \leq 1/16$, so the probability that the prover can choose such an x is less than $1/16$.

To do this for larger values of a, the verifier and prover run the above protocol on the circuit

$$C'(x_1, \ldots, x_{3a}) = C(x_1) \text{ AND } \cdots \text{ AND } C(x_{3a})$$

with size parameter s^{3a}. If C has at least s satisfying assignments then C' has at least s^{3a} satisfying assignment and the protocol accepts with probability $1/8$. If C has fewer than $(1-1/a)s$ satisfying assignments, then C' has fewer than $(1 - 1/a)^{3a}s^{3a} \leq s^m/8$ satisfying assignments. □

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.

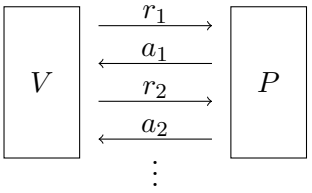
Theorem 4. *If f has an r-round interactive proof, then it has a public-coin interactive proof with at most $r + 2$ rounds.*

We won't show how to do this, but let us give some intuition about how this theorem relates to the graph non-isomorphism protocol. Let us make the additional assumption that we have the promise the graphs G_0 and G_1 have no automorphisms. If G_0 and G_1 are isomorphic, then there are $n!$ possible first messages sent by the verifier, one for each permutation of the vertices. If they are not then there are $2n!$ possible messages, $n!$ for permutations of G_0 and $n!$ for permutations of G_1. Now let C be the nondeterministic circuit that on input G accepts if and only there exists an isomorphism between G and G_0 or between G and G_1. Then proving G_0 and G_1 non-isomorphic amounts to proving that $(C, 2n!, 2)$ is a yes-instance of MSAT.

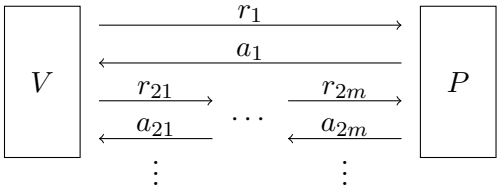
Round reduction Now we sketch how to turn an r-round public-coin interactive proof for any constant $r > 2$ into a two-round public-coin proof. We will do it iteratively, reducing the number of rounds one by one. In such a protocol, the verifer starts by asking a random question r_1, then

the prover answers by some string a_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.



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.

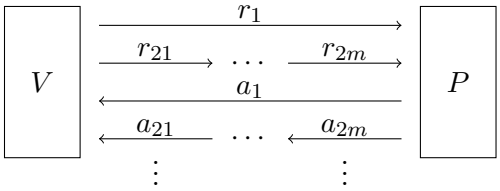


To analyze what happens it helps to assume the probability of accepting for the yes instances is at least $8/9$ and the probability of rejecting the no instances is at most $1/9$. Let's look at the yes instances first. It then follows that for at least a $2/3$ fraction choices of the first message r_1, 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 $r_{21}, \ldots, r_{2m}$. 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 two.



By repeating this transformation 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. One last simplification we can make is to turn any proof into one with perfect completeness.

Definition 5. The class AM consists of those decision problems that admit a two-round interactive proof (V, P) such that

$$\begin{aligned} \text{if } x \in \mathit{YES}, &\quad \Pr[(V,P)(x) \text{ accepts}] = 1 \\ \text{if } x \in \mathit{NO}, &\quad \Pr[(V,P^*)(x) \text{ accepts}] \leq 1/2 \quad \text{for all } P^*. \end{aligned}$$

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.

Moreover, since all the transformations we described are efficient, we obtain the following. A *reduction* from promise decision problem A to promise decision problem B is a function that maps yes instances of A to yes instances of B and no instances of A to no instances of B.

Theorem 6. *Any f that has a constant-round interactive proof polynomial-time reduces to* MSAT *even if we fix s to equal the number of all possible assignments and a to one.*

4 Derandomizing interactive proofs

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. By our standard simulation of Turing Machines by circuits, every f in AM has a polynomial-size family of *randomized nondeterministic circuits* $\{C_m\}$, namely

$$\begin{aligned} \text{if } x \in \mathit{YES}, &\quad \Pr_r[C_m(x,r,w) \text{ accepts for some } w] = 1 \\ \text{if } x \in \mathit{NO}, &\quad \Pr[C_m(x,r,w) \text{ accepts for some } w] \leq 1/2. \end{aligned}$$

The randomness of these circuits can be eliminated just like in the proof of Theorem 12 in Lecture 5, leading to the following analogue of that result.

Theorem 7. *Every decision problem in* AM *has a polynomial-size family of nondeterministic circuits.*

So in the circuit model, randomness and a constant number of interaction rounds do not increase the power of polynomial-size verifiers. For algorithms, we can attempt to replace the randomness of C_m by the output of a suitable pseudorandom generator. An analysis analogous to the proof of Theorem 3 in Lecture 7 gives the following result.[1]

[1] A weaker-looking assumption is in fact sufficient.

Theorem 8. *If there is a problem f decidable in time $2^{O(t)}$ but not decidable by nondeterministic circuits of size $2^{\delta t}$ even on a $1/2 + 2^{-\delta t/2}$ fraction of uniformly chosen inputs for some $\delta > 0$ then* AM = NP*, even for promise problems.*

If you believe the assumption this sounds quite strange. Think of the example of graph non-isomorphism. We saw a simple *interactive* method that can be used to prove any two given graphs are non-isomorphic, but nobody knows if there are short written proofs of graph non-isomorphism. Theorem 8 gives evidence that such proofs are in fact likely to exist!

One last word of warning: Our discussion here applies only to proof systems whose number of rounds is constant, that is independent of input size. Proofs of unbounded round complexity are much more powerful. In particular they can be used to refute the existence of solutions to NP problems, something we will talk about in the next lecture.

References

Interactive proofs were invented independently by Goldreich, Micali, and Wigderson and by Babai, who only considered public-coin proofs. The MSAT problem was introduced and Theorem 2 was proved by Goldwasser and Sipser; it also holds when the number of rounds is unbounded. Round reduction was proved by Babai and Moran. They show, more generally, that the number of rounds can be halved with a polynomial increase in verifier complexity. The derandomization of interactive proofs was first proposed by Klivans and van Melkebeek in a paper titled *Graph Nonisomorphism Has Subexponential Size Proofs Unless The Polynomial-Time Hierarchy Collapses.* The hardness assumption was simplified and the running time of the derandomization improved in a sequence of works leading to a general result of Shaltiel and Umans. Last year Babai showed that graph isomorphism and non-isomorphism have *quasi-polynomial* time algorithms.