Mathematical proofs are immutable objects: You read the proof, and if it is convincing, you accept it as correct. Outside mathematics "proofs" may involve interaction. A criminal suspect who wants to prove their innocence in court is subjected to a cross-examination. What happens in the interaction with layers affects the decision of the jury. The complexity-theoretic analogue of a cross-examination is called an interactive proof.

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

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

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

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

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 the verifier is either convinced that the prover is correct and accepts, or thinks the prover's 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 $(y_1, z_1, \dots, y_k, z_k)$ which represent the messages sent and received in the first k rounds of interaction. Here, $y_1, \dots, y_k$ are the messages sent by the machine itself, and $z_1, \dots, z_k$ 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, the interaction (A, B) on input x consists of the sequence of messages $y_1 = A(x), z_1 = B(x, y_1), y_2 = A(x, y_1, z_1), z_2 = B(x, y_1, z_1, y_2), \dots,$ exchanged between A and B before A decides to accept or reject. The *round complexity* is the number of messages exchanged before A reaches a decision.

A *(polynomial-time) deterministic interactive proof* for a decision problem $(\mathit{YES}, \mathit{NO})$ is a pair of Turing Machines (V, P) where V runs in time polynomial in the input x, and

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

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

This model is at least as powerful as NP, which requires no interaction. Is it more powerful? A thought experiment 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 single prover message, so the problem decided by the proof system must be in NP.

2 Interaction and randomness

Now let's allow the verifier to use randomness. It is no more the case that the prover can predict the verifier's questions and answer them before they are asked.

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

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

The *round complexity* of the proof system is the maximum round complexity of the interaction (over the choice of x and the verifier's randomness).

As in the definition of BPP here is nothing special about the constants 2/3 and 1/3; any two constants $c > s$ will do. The gap $c - s$ can be amplified by repetition. If prover and verifier repeat the protocol t times and verifier outputs the majority of answers, the probabilities are amplified to $1 - 2^{-\Omega(t)}$ and $2^{-\Omega(t)}$, respectively. Sequential repetition would increase the number of rounds by a factor of t. Parallel repetition is preferable as it preserves the round complexity.

Here is a decision problem that has a two-round interactive proof, but is not (provably) known to be in NP. Graphs G_0 and G_1 on n vertices are *isomorphic* if there exists a permutation π of vertices that, when applied to G_0, outputs exactly G_1, namely

$$(u, v) \text{ is an edge of } G_0 \text{ if and only if } (\pi(u), \pi(v)) \text{ is an edge of } G_1. \tag{1}$$

GI (GRAPH ISOMORPHISM): Given a pair of graphs (G_0, G_1), are they isomorphic?

There are some natural tests you can try for isomorphism, for example 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). G_0 and G_1 can pass these tests and still be isomorphic. No general polynomial-time test for isomorphism is known (though in 2016 Babai discovered a quasipolynomial, i.e. $n^{O(\log n)}$-time, test). Problem GI is in NP because given π, condition (1) can be checked efficiently. In the context of interaction the complementary problem is more interesting:

$\overline{\text{GI}}$ (GRAPH NONISOMORPHISM): Given a pair of graphs (G_0, G_1), are they *non*-isomorphic?

This does not look like an NP problem. How would you convince someone that two graphs are not isomorphic without going over all candidate $n! = n^{\Omega(n)}$ isomorphisms and verifying that they all fail? In fact $\overline{\text{GI}}$ is not known to be in NP. However it admits the following one-round *interactive* proof:

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.

This is an interactive proof for $\overline{\text{GI}}$ for the following reason: 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 bit i. Therefore no matter what the prover does, the chances that he guesses the correct value of i is exactly $1/2$.

So we have some evidence that interaction helps in proofs. If one-round of interaction is more powerful than no interaction, how about two rounds? It turns out that any fixed number of additional rounds does not help:

Theorem 2. *For every constant r, if f has an r-round interactive proof, then it has a one-round public-coin interactive proof.*

3 Public-coin Proofs and Pairwise Independence

Let's first explain what public-coin means. In the graph non-isomorphism proof it is important that the bit i chosen by the verifier is hidden from the prover: If the prover knew i he could make the verifier accept with probability 1 even if G_0 and G_1 were not isomorphic.. 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 and flip some random coins. The coin flips serve as the verifier's first question. The prover then answers this question, and then they flip some coins again to come up with the verifier's second question. And so on until the verifier makes her decision based on the coin flips and prover's replies.

This looks like a strange thing to do. How much can the verifier learn by asking random questions? 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 $C(x, w) = 1$. Nondeterministic circuits are to NP what ordinary circuits are to P: Just like all problems in P admit polynomial-size circuit families, all problems in NP admit nondeterministic polynomial-size circuit families.

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; if we set $s = 1$ and $a = 2$, yes and no instances are satisfiable and unsatisfiable circuits. But if for example $s = 2^{n/2}$ then it is not known to be in NP. (The natural certificate would consist of an exponentially long list of assignments.)

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

The proof of this theorem makes use of a non-cryptographic type of a pseudorandom function called a pairwise-independent hash function. A keyed function $H\colon \{0,1\}^n \to \{0,1\}^k$ is *pairwise independent* if for any pair of distinct inputs $x_1 \neq x_2$, the pair $(H(x_1), H(x_2))$ is a random independent pair of values in $\{0,1\}^k \times \{0,1\}^k$. In other words, H is pseudorandom against any two-query distinguisher.

One example of an efficiently computable pairwise independent hash function with one bit of output is $H(x) = IP(a, x) + b$ where IP is the inner product mod 2 function, a is a random n-bit string, and b is a random bit. Each individual output $H(x)$ is a random bit owing to b and every pair of outputs $(H(x_1), H(x_2))$ is uncorrelated because $H(x_1) + H(x_2) = IP(a, x_1 + x_2)$ is also a random bit. This is only possible if $H(x_1)$ and $H(x_2)$ are independent. To obtain more output bits the construction can be repeated independently, i.e., $H(x) = (IP(a_1, x) + b_1, \dots, IP(a_k, x) + b_k)$. This can be viewed as the $\mathbb{F}_2$ matrix-vector product expression $H(x) = Ax + b$, where A is a random $k \times n$ Boolean matrix and b is a random k-bit Boolean (column) vector.

One reason pairwise independence is important is because it isolates unique elements of large sets:

Lemma 4. *Let $H\colon \{0,1\}^n \to \{0,1\}^k$ be a pairwise independent hash function. For every subset X of $\{0,1\}^m$ of size between 2^{k-2} and 2^{k-1}, the probability that there is a (unique) x in X such that $H(x) = 0$ is at least $1/8$ (over the choice of H).*

Proof. The event "there is a unique $x \in X$ such that $H(x) = 0$" is a *disjoint* union of the $|X|$ events "$H(x) = 0$ and $H(x') \neq 0$ for every $x' \in Y - \{x\}$", one for each x in X. Therefore

$$\begin{aligned}
&\Pr[\text{there is a unique } x \in X \text{ with } H(x) = 0] \\
&\qquad = \sum_{x \in X} \Pr[H(x) = 0 \text{ and } H(x') \neq 0 \text{ for all } x' \in X - \{x\}] \\
&\qquad = \sum_{x \in X} \Pr[H(x') \neq 0 \text{ for all } x' \in X - \{x\} \mid H(x) = 0] \Pr[H(x) = 0] \\
&\qquad = \sum_{x \in X} (1 - \Pr[H(x') = 0 \text{ for some } x' \in X - \{x\} \mid H(x) = 0]) \Pr[H(x) = 0] \\
&\qquad \geq \sum_{x \in X} \Bigl(1 - \textstyle\sum_{x' \in X - \{x\}} \Pr[H(x') = 0 \mid H(x) = 0]\Bigr) \Pr[H(x) = 0] \\
&\qquad \geq |X| \cdot (1 - |X| \cdot 2^{-k}) \cdot 2^{-k}.
\end{aligned}$$

Since Y has size between 2^{k-2} and 2^{k-1} the last expression is at least $(1/4) \cdot (1 - 1/2) = 1/8$. □

To understand Lemma 4 notice that if H was a purely random function then the probability that $H(x) = 0$ for some $x \in X$ is exactly $1 - (1 - 2^{-k})^{|X|} \geq 1 - (1 - 2^{-k})^{2^{k-2}} \approx 1 - e^{-1/4} \approx 0.22$. Lemma 4 says that when H is merely pairwise independent this probability is not much smaller.

Proof of Theorem 3. We first show how to do this when $1 - 1/a$ is replaced by $1/8$. The prover and verifier choose a random hash function $H\colon \{0,1\}^n \to \{0,1\}^k$, where $2^{k-2} \leq s \leq 2^{k-1}$, namely the matrix A and the vector b. The prover then sends any x, w such that C accepts x with witness w and $H(x) = 0$. The verifier checks these two conditions and accepts iff they both hold.

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

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

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. □

Theorem 3 can be used to turn *any* private-coin protocol into a public-coin one.

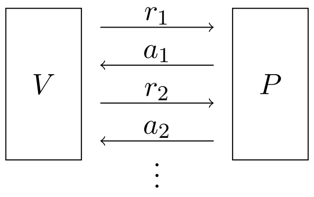
Theorem 5 (Goldwasser and Sipser)**.** *If f has an r-round interactive proof, then it has a public-coin interactive proof with at most $r+1$ rounds.*

We won't show how to do this in general. Instead here is some intuition about how Theorem 3 can be used to turn the graph non-isomorphism proof into a public-coin proof. Let us assume that we have the additional 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 $2 \cdot n!$ possible messages, $n!$ for permutations of G_0 and $n!$ for permutations of G_1. Let C be the nondeterministic circuit that accepts input G if 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.

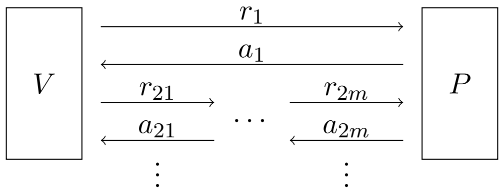
4 Round Reduction

Now we sketch how to turn an r-round public-coin interactive proof for any constant $r > 2$ into a one-round public-coin proof. This is done 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.



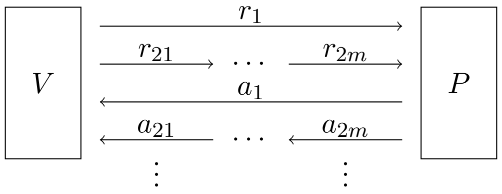
Consider what happens if in the third round of the protocol, the verifier "forks" m independent executions of it in parallel (we'll give the value of m later): Namely, instead of asking a single question r_2, it asks m such questions $r_{21}, \dots, r_{2m}$ independently at random. It then expects m answers $a_{21}, \dots, a_{2m}$ 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 the response $a_1 = P(x, r_1)$ provided by the honest prover P, 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 $2^{-m/6}$, so the overall probability that the verifier accepts is at least $2/3 - 2^{-m/6}$.

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 accepts with probability at most 1/3. 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 $2^{-m/6}$. As there are at most 2^k possible responses a_1, 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 $m = 6k + 19$ to make this probability as small as 1/9.

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 at most $1/3 + 1/9 = 4/9$. So the message a_1 can be *delayed* to come after the questions $r_{21}, \dots, r_{2m}$. The order of the second and third message can then be swapped. After this swap the first two rounds reduce to a single round.



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 transformation that we won't show is one that turns any 1-round interactive proof into one with perfect completeness (yes-instances are always accepted as in the graph non-isomorphism proof).

Definition 6. The class AM consists of those decision problems that admit a one-round public-coin interactive proof (V, P) such that

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

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.

Applying the simulation of Turing Machines by circuits from Lecture 6, the AM verifier can be efficiently compiled into a circuit family $\{C_m\}$ such that

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

After fixing x, this transformation yields an efficient reduction from any AM problem to MSAT: *YES* and NP instances of the AM problem are mapped to nondeterministic circuits that accept all of their inputs and at most half of their inputs, respectively. As all the other reductions (private to public coins, round reduction) are also efficient, MSAT is complete for all promise problems that have polynomial-time constant-round interactive proofs.

Theorem 7. *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).*

5 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. Using amplification as in the proof of Adleman's theorem in Lecture 6, the randomness from (2) can be eliminated giving the following consequence:

Theorem 8. *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. The Nisan-Wigderson generator can itself be used to derandomize proofs given a sufficiently strong hardness assumption.

Theorem 9. *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* $\mathrm{AM} = \mathrm{NP}$*, even for promise problems.*

If you buy into the assumption (as most of us complexity theorists do), the conclusion sounds a bit 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 9 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, which is the topic of 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 3 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. Lemma 4 was proved by Valiant and Vazirani.

The derandomization of interactive proofs was first discussed by Klivans and van Melkebeek. 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. In 2016 Babai showed that graph isomorphism and non-isomorphism have *quasi-polynomial* time algorithms.