1 What are interactive proofs

Think of a "prover" trying to convince a "verifier" that a statement is correct. For example, that a certain CNF formula is satisfiable. How much power does interaction gives? We will assume the verifier is poly-time, while the prover is all powerful. Think of the prover trying to "prove" to the verifier that \( x \in L \).

**Definition 1.1.** Let \( L \subset \{0,1\}^* \) be a language. An interactive protocol between a verifier \( V \) and a prover \( P \), to check whether \( x \in L \), is a sequence of messages:

1. \( V \) sends to \( P \) message \( m_1 = f_1(x) \).
2. \( P \) sends to \( V \) message \( m_2 = g_1(x, m_1) \).
3. \( V \) sends to \( P \) message \( m_3 = f_2(x, m_1, m_2) \).
4. \( P \) sends to \( V \) message \( m_4 = g_2(x, m_1, m_2, m_3) \).
5. ...

We assume the functions \( f_i \) are poly-time computable (either deterministic or randomized), and the functions \( g_i \) are arbitrary. At the end, the verifier either accepts (e.g says "\( x \in L \)") or rejects ("\( x \notin L \)") based on the sequence of messages (transcript), e.g. \( V(m_1, \ldots, m_k) \in \{0,1\} \). We would like that

- If \( x \in L \) then there exists messages for the prover that will convince the verifier that \( x \in L \), e.g. make the verifier accept (maybe whp).
- If \( x \notin L \) then for any possible messages of the prover, the verifier that should reject, e.g. declare \( x \notin L \) (maybe whp).

Deterministic interactive protocols are equivalent to \( NP \), which can be viewed as a one-round protocol (where the prover sends the proof to the verifier, which then verifies it).
Claim 1.2. $L$ can be decided by a interactive protocol with deterministic prover and verifier iff $L \in \text{NP}$.

Proof. Clearly if $L \in \text{NP}$ then it is decided by a one-round protocol. On the other side, if $L$ is decided by a protocol with deterministic verifier and prover, the prover can compute in advance all the messages that will be sent, and send the all together. That is, the proof is the sequence of message $(m_1, m_2, \ldots, m_k)$, and the verifier checks that $m_{2i-1} = f_i(x, m_1, \ldots, m_{2i-2})$ for all $i$. □

However, it turns out that allowing the verifier to be randomized gives much more power.

Definition 1.3 (IP). A language $L \in \text{IP}$ if there exists an interactive protocol with randomized poly-time verifier and unbounded prover, such that

1. If $x \in L$ then there exist a prover for which $\Pr[V \text{ accepts}] \geq 2/3$.
2. If $x \not\in L$ then for any possible prover $\Pr[V \text{ rejects}] \geq 2/3$.

We say $L \in \text{IP}[k]$ if the protocol exchanges at most $k$ messages. We may allow $k$ to be a function of $|x|$. Note that we always limit $k \leq |x|^{O(1)}$ to keep the verifier poly-time. In this notation, $\text{IP} = \text{IP}[\text{poly}]$.

We will prove later the following surprising theorem.

Theorem 1.4 (LFKN’90, Shamir’90). $\text{IP} = \text{PSPACE}$.

Let us first verify that $\text{IP} \subset \text{PSPACE}$. In particular, even though we allow the prover to be all-powerful, it can actually be implemented in PSPACE. Also, we can assume the prover is deterministic.

Lemma 1.5. $\text{IP} \subset \text{PSPACE}$.

Proof. Let $L \in \text{IP}$. That is $x \in L$, where $|x| = n$, iff there exist answers from the prover $m_2(x, m_1), m_4(x, m_1, m_2, m_3), \ldots, m_k(x, m_1, \ldots, m_{k-1})$ for which $\Pr[V(x, m_1, \ldots, m_k) = 1] \geq 2/3$. The probability is over the randomness used by the verifier, and $k = k(x) \leq n^{O(1)}$.

Consider the following tree. Its vertices are possible partial transcripts $\tau = (x, m_1, \ldots, m_t)$ for $t \leq k$. Define the value $v(\tau)$ as the maximal probability that a prover can make the verifier accept, assuming that the messages $m_1, \ldots, m_t$ were sent. Clearly, $x \in L$ iff $v(x) \geq 2/3$. Now, as odd messages are sent by the verifier, if $i$ is even then

$$v(x, m_1, \ldots, m_i) = \sum_{m_{i+1}} v(x, m_1, \ldots, m_i) \cdot \Pr[V \text{ sent message } m_{i+1} \text{ given } x, m_1, \ldots, m_i],$$

and even messages are sent by the prover, then if $i$ is odd then

$$v(x, m_1, \ldots, m_i) = \max_{m_{i+1}} v(x, m_1, \ldots, m_i).$$

Note also that the length of messages is at most $n^{O(1)}$. Note that we can compute $v(x)$ be a DFS-like process, at every step computing recursively the value, while only using polynomial space. A crucial point is since we only need to distinguish if $v(x) \leq 1/3$ or $v(x) \geq 2/3$, it suffices to use polynomial many precision bits for the probabilities. Hence, we can decide if $x \in L$ using only polynomial space. □
2 Examples

2.1 Graph non-isomorphism

The Graph Isomorphism problem is to decide whether two graphs $G_1, G_2$ are isomorphic. It is clearly in NP. The Graph Non-Isomorphism problem is in co-NP and is not known to be in NP. However, there is a simple randomized interactive protocol which can prove that $G_1, G_2$ are non-isomorphic.

1. The verifier chooses uniformly $b \in \{0, 1\}$ and a permutation $\pi \in S_n$ and sends $G = \pi(G_b)$ to the prover.

2. The prover returns $r \in \{0, 1\}$ such that $G$ is isomorphic to $G_r$.

3. The verifier accepts if $b = r$.

**Lemma 2.1.** Graph Non-Isomorphism is in $IP$[2] If $G_1, G_2$ are non-isomorphic, there is a prover which makes $V$ accepts always. If $G_1, G_2$ are isomorphic, then for any prover, the probability that the verifier accepts is at most $1/2$.

**Proof.** If $G_1, G_2$ are non-isomorphic the verifier can know if $G_b$ is isomorphic to $G_1$ or $G_2$, hence always return $r = b$. However, the main point is that if $G_1, G_2$ are isomorphic, the distribution of $\pi(G_1)$ and $\pi(G_2)$ is identically the same, hence the prover has no way to knowing whether $V$ chose $G_1$ or $G_2$. \qed

2.2 Quadratic non-residue

A number $1 \leq x \leq n$ is a quadratic residue modulo $n$ if $x = y^2 \pmod{n}$. Clearly checking if $x$ is a quadratic residue is in NP, as the prover can give $y$. However, we do not know how to show non-quadratic reside in NP. There is a randomized interactive protocol which does that.

1. The verifier chooses $y \in Z_n^*$ randomly, and a bit $b \in \{0, 1\}$, sends $y^2 x^b \pmod{n}$ to the prover.

2. The prover sends $r \in \{0, 1\}$ to the verifier.

3. The verifier accepts if $r = b$.

Note that the verifier can sample $y \in Z_n^*$ efficiently by sampling $y \in Z_n$ randomly, and if $gcd(y, n) > 1$ then resample. It can be shown that $Pr_{y \in Z_n}[y \in Z_n^*] \geq 1/\log(n)$.

**Lemma 2.2.** Quadratic Non-Residue is in $IP$[2]. If $x$ is a quadratic non-reside then there is a strategy for the prover which makes the verifier accepts always. If $x$ is a quadratic reside then for any prover, the verifier accepts with probability at most $1/2$. 

3
Proof. If $x$ is a quadratic non-residue then the support of $\{y^2 : y \in \mathbb{Z}_n^*\}$ and $\{y^2x : y \in \mathbb{Z}_n^*\}$ is disjoint, hence the prover can decide the value of $b$. If $x$ is a quadratic residue then $\{y^2 : y \in \mathbb{Z}_n^*\} = \{y^2x : y \in \mathbb{Z}_n^*\}$ hence the distribution of $y^2x^b \pmod n$ is the same whether $b = 0$ or $b = 1$. Hence the prover cannot reply with the correct value with probability better than $1/2$. 

3 Arthur-Merlin protocols

It seems that the interactive protocols for Graph Non-Isomorphism or Quadratic Residue depended crucially on the verifier using private randomness, e.g. using random bits not revealed to the prover. What happens if we disallow this?

Definition 3.1 (Arthur-Merlin games). An AM protocol is a protocol where the verifier messages are simply random bits, e.g. if $|x| = n$ then for some $t = n^{O(1)}$,

1. $V$ picks $r_1 \in \{0,1\}^t$ uniform and sends $m_1 = r_1$.
2. $P$ responds $m_2 = g_1(x, r_1)$.
3. $V$ sends $r_2 \in \{0,1\}^t$ uniform and sends $m_3 = r_2$.
4. $P$ responds $m_4 = g_2(x, r_1, r_2)$.
5. . . .

At the end, the verifier accepts or rejects based on the transcript (via some poly-time function). As before, AM is the set of languages which can be decided probabilistically by an AM protocol, and $AM[k]$ is those where the protocol uses at most $k$ rounds.

Theorem 3.2 (Goldwasser-Sipser’87). $IP[k] \subset AM[k+2]$.

We will prove a baby version of this.

Theorem 3.3. Graph non-isomorphism is in $AM[2]$.

Let $G_1, G_2$ be graphs. Assume that $G_1, G_2$ have no nontrivial automorphisms, and consider

$$S = \{H : H \text{ isomorphic to } G_1 \text{ or } G_2\}.$$

Then:

- If $G_1, G_2$ isomorphic then $|S| = n!$.
- If $G_1, G_2$ non-isomorphic then $|S| = 2n!$.

So, the prover will convince the verifier then $|S| \geq 2n!$. In the general case, where $G_1$ or $G_2$ may have nontrivial automorphisms, the same idea will work with

$$S = \{(H, \pi) : H \text{ isomorphic to } G_1 \text{ or } G_2, \quad \pi \in Aut(H)\}.$$
3.1 Size checking protocol

Let \( S \subset \{0,1\}^n \) be a set, where we need to decide whether \(|S| \leq K\) or \(|S| \geq 2K\). Assume furthermore, that the verifier can efficiently test given \( x \) whether \( x \in S \). We will construct an \( AM[2] \) protocol such that for some \( \alpha \geq 1/8 \),

- If \(|S| \geq 2K\) then there exist a prover which will make the verifier accept with probability \( \geq (3/2)\alpha \).

- If \(|S| \leq K\) then for any prover, the probability the verifier accepts is at most \( \alpha \).

This is not quite what we need, but we will see how to amplify the errors. We will need the definition of pairwise-independent hash functions.

**Definition 3.4.** A family of functions \( H_{n,k} : \{h : \{0,1\}^n \rightarrow \{0,1\}^k\} \) is pairwise-independent if for any \( x \neq x' \in \{0,1\}^n \) and any \( y, y' \in \{0,1\}^k \),

\[
\Pr_{h \in H}[h(x) = y, h(x') = y'] = 2^{-2k}.
\]

**Lemma 3.5.** Let \( \mathbb{F}_{2^n} \) be the finite field of size \( 2^n \). Then

\[
H_{n,n} = \{h(x) = ax + b : a, b \in \mathbb{F}_{2^n}\}
\]

is a pairwise-independent hash function family of size \( 2^{2n} \). In particular, for any \( k \leq n \) we can take

\[
H_{n,k} = \{h(x) = \text{first } k \text{ bits of } ax + b : a, b \in \mathbb{F}_{2^n}\}
\]

to be a pairwise-independent hash function family of size \( 2^{2n} \) mapping \( n \) bits to \( k \) bits.

**Proof.** Let \( x \neq x' \) and \( y, y' \) be elements in \( \mathbb{F}_{2^n} \). The number of \( a, b \in \mathbb{F}_{2^n} \) such that \( ax + b = y \) and \( ax' + b = y' \) is exactly one, as this is a nonsingular system of linear equations. \( \square \)

Let \( k \) be such that \( 2K \leq 2^k \) and fix a pairwise-independent family \( H_{n,k} \) mapping \( n \) bits to \( k \) bits. Consider the following \( AM[2] \) protocol:

1. The verifier sends a uniform \( h \in H_{n,k} \) and \( y \in \{0,1\}^k \) to the prover.
2. The prover responds with \( x \in \{0,1\}^n \).
3. The verifier accepts if \( x \in S \) and \( h(x) = y \).

**Lemma 3.6.** Let \( p = |S|/2^k \) where \( p \leq 1 \). Then

\[
p - p^2/2 \leq \Pr_{h,y}[\exists x \in S, h(x) = y] \leq p
\]
Proof. Clearly
\[
\Pr_{h,y}[\exists x \in S, h(x) = y] = \mathbb{E}_{h,y}[y \in h(S)] = 2^{-k}\mathbb{E}[|h(S)|].
\]
The upper bound follows since \(|h(S)| \leq |S|\) and hence \(2^{-k}\mathbb{E}[|h(S)|] \leq |S|/2^{k} \leq p\). For the lower bound, we have
\[
|h(S)| \geq |S| - \sum_{x \neq x' \in S} 1_{h(x) = h(x')}.
\]
Hence,
\[
\mathbb{E}_{h}[|h(S)|] \geq |S| - \left(\frac{|S|}{2}\right)2^{-k} \geq |S| - \frac{|S|^2}{2}2^{-k},
\]
which gives
\[
2^{-k}\mathbb{E}[|h(S)|] \geq p - p^2/2.
\]

Let us fix \(k\) so that \(2^{k-2} \leq 2K \leq 2^{k-1}\).

Corollary 3.7. Let \(\alpha = K/2^k\) where \(1/8 \leq \alpha \leq 1/4\). Then
1. If \(|S| = 2K\) then the verifier accepts with probability at least \(2\alpha - (2\alpha)^2/2 \geq (3/2)\alpha\).
2. If \(|S| = K\) then the verifier accepts with probability at most \(\alpha\).

We can now describe the full protocol. Let \(t = O(1)\) large enough. Then
1. The verifier sends uniform \(h_1, \ldots, h_t \in H_{n,k}\) and \(y_1, \ldots, y_t \in \{0,1\}^k\) to the prover.
2. The prover responds with \(x_1, \ldots, x_t \in S\).
3. Let \(T = \{i : x_i \in S, h(x_i) = y_i\}\). The verifier accepts if \(|T| \geq (5/4)\alpha t\).

Theorem 3.8. By choosing \(t\) a large enough constant,

• If \(G_1, G_2\) are isomorphic then \(\Pr[V\text{ accepts}] \leq 1/3\).

• If \(G_1, G_2\) are non-isomorphic then \(\Pr[V\text{ accepts}] \geq 2/3\).

Proof. If \(G_1, G_2\) are isomorphic then \(|S| = K = n!\). Hence, the probability that exists \(x_i \in S\) for which \(h_i(x_i) = y_i\) is at most \(\alpha\). By Chernoff (or Chebychev), by choosing \(t\) large enough we get that
\[
\Pr[W \geq (5/4)\alpha t] = \Pr[W \geq (5/4)\mathbb{E}[W]] \leq 1/3.
\]
On the other hand, if \(G_1, G_2\) are non-isomorphic the prover can send good \(x_i\) for each \(i\) with probability \((3/2)\alpha\). Hence, by choosing \(t\) large enough we get that
\[
\Pr[W \geq (5/4)\alpha t] = \Pr[W \geq (5/6)\mathbb{E}[W]] \geq 2/3.
\]
4 Interactive protocols for co-NP

We will prove that co-NP has an interactive protocol, which was proven by [LFKN]. The same idea was generalized by Shamir to prove that $IP = PSPACE$.

Let $\phi(x) = C_1 \land \ldots \land C_m(x)$ be a 3-CNF. The prover wants to convince the verifier that there are no satisfying assignments to $\phi$. More generally, we will build a protocol where the prover would prove that $\phi$ has exactly $k$ satisfying assignments.

4.1 Simplified protocol

The prover wants to prove that $\sum_{x \in \{0,1\}^n} \phi(x) = k$. Consider the following protocol:

1. Prover sends the two numbers $k_0 = \sum_{x' \in \{0,1\}^{n-1}} \phi(0, x')$ and $k_1 = \sum_{x' \in \{0,1\}^{n-1}} \phi(1, x')$.
2. Verifier checks that $k_0 + k_1 = k$.
3. Verifier chooses a random bit $b \in \{0,1\}$ and asks the prover to prove that $\sum_{x' \in \{0,1\}^{n-1}} \phi(b, x') = k_b$.
4. If there are no free variables left, the verifier accepts if the formula is correct.

Claim 4.1. If $\sum_{x \in \{0,1\}^n} \phi(x) = k$ there is a strategy for the prover that will make the verifier accept always. If not, then for any strategy of the prover, the verifier will reject with probability at least $2^{-n}$.

Proof. The strategy for an honest prover is obvious. If however $\sum_{x \in \{0,1\}^n} \phi(x) \neq k$ then the prover is cheating on either $k_0$ or $k_1$ (or both). Hence the verifier will continue in a branch where the prover is giving the wrong number of assignments with probability $\geq 1/2$. Hence, the probability the verifier will continue in such a path until there are no more free variables is $\geq 2^{-n}$, where he can check for himself the statement. \qed

4.2 Actual protocol

We would like a protocol in which, when the prover is lying, we will continue in a "lying branch" with good probability. This is done by using polynomials. Let $2^n < p < 2^{n+1}$ be a prime.

Claim 4.2. There exists a polynomial $f(x_1, \ldots, x_n)$ over $\mathbb{F}_p$ such that

1. $f(x) = \phi(x)$ for all $x \in \{0,1\}^n$.
2. It is easy (e.g. poly-time) to evaluate $f(a_1, \ldots, a_n)$ for any $a_1, \ldots, a_n \in \mathbb{F}_p$.
3. The degree of $f$ is $\leq 3m$.

Proof. We can clearly compute any clause $C_i(x_a, x_b, x_c)$ by a polynomial $f_i(x_a, x_b, x_c)$ of degree at most 3, on inputs in $\{0,1\}^3$. Set $f(x) = \prod_{i=1}^m f_i(x)$. \qed
The prover now needs to convince the verifier that \( \sum_{x \in \{0,1\}^n} f(x) = k \pmod{p} \). Note that since \( p > 2^n \) this will be good for us, since the number of assignments is between 0 and \( 2^n \). Consider the following protocol:

1. Prover sends the univariate polynomial \( g(x_1) = \sum_{x' \in \{0,1\}^{n-1}} f(x_1, x') \).
2. The verifier verifies that \( \deg(g) \leq 3m \) and \( g(0) + g(1) = k \).
3. The verifier chooses a random \( a \in \mathbb{F}_p \) and asks the prover to prove that \( \sum_{x' \in \{0,1\}^{n-1}} f(a, x') = g(r) \).
4. If there are no free variables left, the verifier checks if \( f(a_1, \ldots, a_n) \) is equal to the value claims by the prover.

**Claim 4.3.** If \( \sum_{x \in \{0,1\}^n} f(x) \neq k \) then, doesn’t matter what strategy the prover takes, the verifier will accept with probability \( \leq O(mn/p) \).

**Proof.** In the first round, let \( g(x_1) \) be the polynomial sent by the prover, and let \( g'(x_1) = \sum_{x' \in \{0,1\}^{n-1}} f(x_1, x') \) be the correct polynomial. Since \( g(0) + g(1) = k \neq g'(0) + g'(1) \) the two polynomials are different. However, since each is of degree \( \leq 3m \), they can agree on at most \( 6m \) inputs \( a \in \mathbb{F}_p \). Hence, with probability \( 1 - (6m/p) \), the value \( a \) chosen by the verifier will force the prover to “continue cheating”.

\[ \square \]