Introduction to Complexity Theory*
Lecture 11: Interactive Proof Systems

Notes taken by Danny Harnik, Tzvika Hartman and Hillel Kugler

March 1, 1999

Summary: We introduce the notion of interactive proof systems and the complexity class IP, emphasizing the role of randomness and interaction in this model. The concept is demonstrated by giving an interactive proof system for the graph non-isomorphism language. We discuss the power of the class IP and prove that co$\mathcal{NP} \subseteq \mathcal{IP}$. We discuss issues regarding the number of rounds allowed in a proof system and introduce the class AM capturing languages recognized by Arthur-Merlin games.

1 Introduction

A proof is a way of convincing a party of a certain claim. When talking about proofs, we consider two parties: the *prover* and the *verifier*. Given an assertion, the prover's goal is to convince the verifier of it's validity, whereas the verifier's objective is to accept only a correct assertion. In mathematics, for instance, the prover provides a fixed sequence of claims and the verifier checks that they are truthful and that they imply the theorem. In real life, however, the notion of a proof has a much wider interpretation. A proof is a process rather than a fixed object, by which the validity of the assertion is established. For instance, a job interview is a process in which the candidate tries to convince the employer that she should hire him. In order to make the right decision, the employer carries out an interactive process. Unlike a fixed set of questions, in an interview the employer can adapt her questions according to the answers of the candidate, and therefore extract more information, and lead to a better decision. This example exhibits the power of a proof process rather than a fixed proof. In particular it shows the benefits of interaction between the parties.

In many contexts, finding a proof requires creativity and originality, and therefore attracts most of the attention. However, in our discussion of proof systems, we will focus on the task of the verifier - the verification process. Typically the verification procedure is considered to be relatively easy while finding the proof is considered a harder task. The asymmetry between the complexity of verification and finding proofs is captured by the complexity class NP.

We can view NP as a proof system, where the only restriction is on the complexity of the verification procedure (the verification procedure must take at most polynomial-time). For each language $L \in$NP there exists a polynomial-time recognizable relation R_L such that:

$$L = \{x : \exists y \text{ s.t. } (x,y) \in R_L\}$$

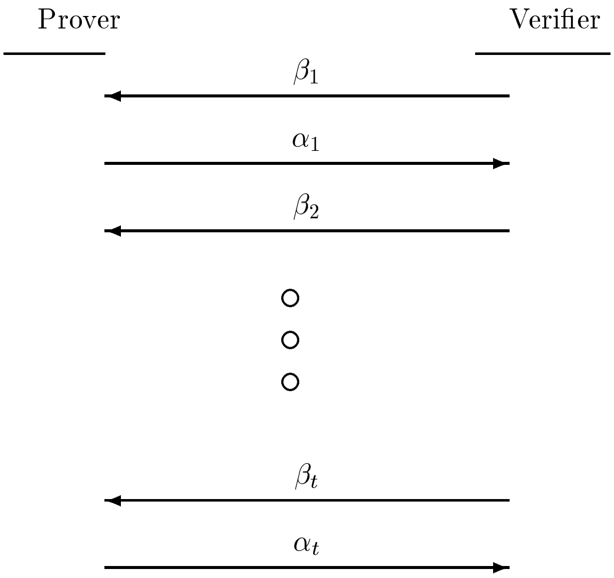
* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

and $(x, y) \in R_L$ only if $|y| \le poly(|x|)$. In a proof system for an NP language L, a proof for the claim "$x \in L$" consists of the prover sending a witness y, and the verifier checking in polynomial-time whether $(x, y) \in R_L$. Such a witness exists only if the claim is true, hence, only true assertions can be proved by this system. Note that there is no restriction on the time complexity of finding the proof (witness). A good proof system must have the following properties:

1. The verifier strategy is efficient (polynomial-time in the NP case).
2. Correctness requirements:
   - **Completeness** : For a true assertion, there is a convincing proof strategy (in the case of NP, if $x \in L$ then a witness y exists).
   - **Soundness** : For a false assertion, no convincing proof strategy exists (in the case of NP, if $x \notin L$ then no witness y exists).

In the following discussion we introduce the notion of *interactive proofs.* To do so, we generalize the requirements from a proof system, adding interaction and randomness.

Roughly speaking, an interactive proof is a sequence of questions and answers between the parties. The verifier asks the prover a question β_i and the prover answers with message α_i. At the end of the interaction, the verifier decides based the knowledge he acquired in the process whether the claim is true or false.



2 Definition of IP

Definition 1 (interactive proof systems): *An* interactive proof system *for a language L is a two-party game between a verifier and a prover that interact on a common input in a way satisfying the following properties:*

1. *The verifier strategy is a probabilistic polynomial-time procedure* (where time is measured in terms of the length of the common input).

2. *Correctness requirements:*

- **Completeness** *: There exists a prover strategy P, such that for every $x \in L$, when interacting on the common input x, the prover P convinces the verifier with probability at least $\frac{2}{3}$.*
- **Soundness** *: For every $x \notin L$, when interacting on the common input x, any prover strategy P^* convinces the verifier with probability at most $\frac{1}{3}$.*

Note that the prover strategy is computationally unbounded.

Definition 2 (The IP Hierarchy): *The complexity class IP consists of all the languages having an interactive proof system.*

We call the number of messages exchanged during the protocol between the two parties, the number of rounds *in the system.*

For every integer function $r(\cdot)$, the complexity class IP($r(\cdot)$) consists of all the languages that have an interactive proof system in which, on common input x, at most $r(|x|)$ rounds are used.

For a set of integer functions R, we denote

$$\mathcal{IP}(R) = \bigcup_{r \in R} \mathcal{IP}(r(\cdot))$$

2.1 Comments

- Clearly $\mathcal{NP} \subseteq \mathcal{IP}$.
- The number of rounds in IP cannot be more than a polynomial in the length of the common input, since the verifier strategy must run in polynomial-time. Therefore, if we denote by `poly` the set of all integer polynomial functions, then $\mathcal{IP} = \mathcal{IP}(\texttt{poly})$.
- The requirement for completeness, can be modified to require perfect completeness (acceptance probability 1). In other words, if $x \in L$, the prover can always convince the verifier. These two definitions are equivalent. Unlike this, if we require perfect soundness, interactive proof systems collapse to NP-proof systems. These results will be shown in Section 5.
- Much like in the definition of the complexity class BPP, the probabilities $\frac{2}{3}$ and $\frac{1}{3}$ in the completeness and soundness requirements can be replaced with probabilities as extreme as $1 - 2^{-p(\cdot)}$ and $2^{-p(\cdot)}$, for any polynomial $p(\cdot)$. In other words the following claim holds:

  Claim 2.1 *Any language that has an interactive proof system, has one that achieves error probability of at most $2^{-p(\cdot)}$ for any polynomial $p(\cdot)$.*

  Proof: We repeat the proof system sequentially for k times, and take a majority vote. Denote by z the number of accepting votes. If the assertion holds, then z is the sum of k independent Bernoulli trials with probability of success at least $\frac{2}{3}$. An error in the new protocol happens if $z < \frac{1}{2}k$.

  Using Chernoff's Bound :

$$\Pr[z < (1-\delta)E(z)] < e^{-\frac{\delta^2 E(z)}{2}}$$

We choose $k = O(p(\cdot))$ and $\delta = \frac{1}{4}$ and note that $E(z) = \frac{2}{3}k$ (so that $\frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$) to get:

$$\Pr[z < \frac{1}{2}k] < 2^{-p(\cdot)}$$

The same argument holds for the soundness error (as due to the sequential nature of the interaction we can assert that in each of the k iterations, for any history of prior interactions, the success probability of any cheating strategy is bounded by 1/3). ■

The proof above uses sequential repetition of the protocol to amplify the probabilities. This suffices for showing that the class IP is invariant under the various definitions discussed. However, this method increases the number of rounds used in the proof system. In order to show the invariance of the class IP($r(\cdot)$), an analysis of the parallel repetition version should be given. (Such an argument is given in Appendix C.1 of Oded's book.)

- Introducing both interaction and randomness in the IP class is essential.
  - By adding interaction only, the interactive proof systems collapse to NP-proof systems. Given an interactive proof system for a prover and a deterministic verifier, we construct an NP- proof system. The prover can predict the verifier's part of the interaction and send the full transcript as an NP witness. The verifier checks that the witness is a valid and accepting transcript of the original proof system. An alternative argument uses the fact that interactive proof systems with perfect soundness are equivalent to NP-proof systems (and the fact that a deterministic verifier necessarily yields perfect soundness).
  - By adding randomness only, we get a proof system in which the prover sends a witness and the verifier can run a BPP algorithm for checking its validity. We obtain a class IP(1) (also denoted MA) which seems to be a randomized (and perhaps stronger) version of NP.

2.2 Example - Graph Non-Isomorphism (GNI)

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are called *isomorphic* (denoted $G_1 \cong G_2$) if there exists a 1-1 and onto mapping $\pi : V_1 \to V_2$ such that $(u,v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$. The mapping π, if existing, is called an *isomorphism* between the graphs. If no such mapping exists then the graphs are *non-isomorphic* (denoted $G_1 \ncong G_2$).

GNI is the language containing all pairs of non-isomorphic graphs. Formally :

$$GNI = \{(G_1, G_2) : G_1 \ncong G_2\}$$

An interactive proof system for GNI:

- G_1 and G_2 are given as input to the verifier and the prover. Assume without loss of generality that $V_1 = V_2 = \{1, 2, ..., n\}$
- The verifier chooses $i \in_R \{1, 2\}$ and $\pi \in_R S_n$ (S_n is the group of all permutations on $\{1, 2, ..., n\}$).

  He applies the mapping π on the graph G_i to obtain a graph H

  $$H = (\{1, 2, ..., n\}, E_H) \text{ where } E_H = \{(\pi(u), \pi(v)) : (u,v) \in E_i\}$$

  and sends the graph H to the prover.

- The prover sends $j \in \{1, 2\}$ to the verifier.
- The verifier accepts iff $j = i$.

Motivation : if the input graphs are non-isomorphic, as the prover claims, then the prover should be able to distinguish (not necessarily by an efficient algorithm) isomorphic copies of one graph from isomorphic copies of the other graph. However, if the input graphs are isomorphic, then a random isomorphic copy of one graph is distributed identically to a random isomorphic copy of the other graph and therefore, the best choice the prover could make is a random one. This fact enables the verifier to distinguish between the two cases. Formally:

Claim 2.2 *The above protocol is an interactive proof system for GNI.*

Comment: We show that the above protocol is an interactive proof system with soundness probability at most $\frac{1}{2}$ rather than $\frac{1}{3}$ as in the formal definition. However, this is equivalent by an amplification argument (see Claim 2.1).
Proof: We have to show that the above system satisfies the two properties in the definition of interactive proof systems:

- The verifier's strategy can be easily implemented in probabilistic polynomial time. (The prover's complexity is unbounded and indeed, he has to check isomorphism between two graphs, a problem not known to be solved in probabilistic polynomial time.)
-
  - Completeness : In case $G_1 \ncong G_2$, every graph can be isomorphic to at most one of G_1 or G_2 (otherwise, the existence of a graph isomorphic to both G_1 and G_2 implies $G_1 \cong G_2$). It follows that the prover can always send the correct j (i.e. a j such that $j = i$), since $H \cong G_i$ and $H \ncong G_{3-i}$.
  - Soundness : In case $G_1 \cong G_2$ we show that the prover convinces the verifier with probability at most $\frac{1}{2}$ (the probability ranges over all the possible coin tosses of the verifier, i.e. the choice of i and π). Denote by H the graph sent by the verifier. $G_1 \cong G_2$ implies that H is isomorphic to both G_1 and G_2. For k = 1,2 let

    $$S_{G_k} = \{\sigma \in S_n \mid \sigma G_k = H\}$$

    This means that when choosing $i = k$, the verifier can obtain H only by choosing $\pi \in S_{G_k}$.
    Assume $\tau \in S_n$ is an isomorphism between G_2 and G_1, i.e. $G_1 = \tau G_2$. For every $\sigma \in S_{G_1}$ it follows that $\sigma\tau \in S_{G_2}$ (because $\sigma\tau G_2 = \sigma G_1 = H$). Therefore, τ is a 1-1 mapping from S_{G_1} to S_{G_2} (since S_n is a group). Similarly, τ^{-1} is a 1-1 mapping from S_{G_2} to S_{G_1}. Combining the two arguments we get that $|S_{G_1}| = |S_{G_2}|$. Therefore, given that H was sent, the probability that the verifier chose $i = 1$ is equal to the probability of the choice $i = 2$. It follows that for every decision the prover makes he has success probability $\frac{1}{2}$ and therefore, his total probability of success is $\frac{1}{2}$.

■

The above interactive proof system is implemented with only 2 rounds. Therefore,

Corollary 3 *GNI* $\in$ *IP(2).*

3 The Power of IP

We have already seen that NP$\subseteq$ IP. The above example suggests that the power of IP is even greater. Since GNI is not known to be in NP we conjecture that NP$\subset$ IP (strict inclusion). Furthermore, the class of languages having interactive proof systems is shown to be equivalent to the powerful complexity class PSPACE. Formally,

Theorem 4 *IP = PSPACE*

We will not give a complete proof of the theorem. One direction of the proof (IP $\subseteq$ PSPACE) is shown in the following discussion:

The Optimal Prover: Given a fixed verifier strategy , there exists an optimal prover strategy, i.e. for every input x, the optimal strategy has the highest probability of convincing the verifier. This is due to the fact that given the verifier strategy and the verifier's coin tosses, a prover can simulate the whole interaction and it's outcome for any prover strategy. The computationally unbounded prover can enumerate all possible outcomes of the verifier's coin tosses, and count how many times each strategy succeeds. The optimal strategy for each input, is one that yields the highest number of successes. Furthermore, this can be done in polynomial-space :

Claim 3.1 *The optimal prover strategy can be computed in polynomial-space.*

Proof: We assume without loss of generality that the verifier tosses all his coins before the interaction begins. We also assume that the verifier plays first.
Let β_i be the i^{th} message sent by the verifier and α_i be the i^{th} message sent by the prover.
Let r be the outcome of all the verifier's coin tosses.
Let $R_{\beta_1,\alpha_1,...,\alpha_{i-1},\beta_i}$ be the set of all r (outcome of coin tosses) that are consistent with the interaction $\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i$.
Let $F(\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i)$ be the probability that an interaction (between the optimal prover and the fixed verifier) beginning with $\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i$ will result in acceptance. The probability is taken uniformly over the verifier's relevant coin tosses (only r such that $r \in R_{\beta_1,\alpha_1,...,\alpha_{i-1},\beta_i}$).

Suppose an interaction between the two parties consists of $\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i$ and it is now the prover's turn to play. Using the function F, the prover can find the optimal move. We show that a polynomial-space prover can recursively compute $F(\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i)$. Furthermore, in the process, the prover finds an α_i that yields this probability and hence, an α_i that is an optimal move for the prover.

The best choice for α_i is one that gives the highest expected value of $F(\beta_1, \alpha_1, ..., \alpha_i, \beta_{i+1})$ over all of the possiblities of verifier's next message (β_{i+1}). Formally :

$$(1) \qquad F(\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i) = \max_{\alpha_i} E_{\beta_{i+1}}[F(\beta_1, \alpha_1, ..., \alpha_i, \beta_{i+1})]$$

Let $V(r, \alpha_1, ..., \alpha_i)$ be the message β_{i+1} that the verifier sends after tossing coins r and receiving messages $\alpha_1, ..., \alpha_i$ from the prover.

The probability for each possible message β_{i+1} to be sent by after $\beta_1, \alpha_1, ..., \alpha_i$ is the portion of possible coins $r \in R_{\beta_1,\alpha_1,...,\alpha_{i-1},\beta_i}$ that yield the message β_{i+1} (i.e. $\beta_{i+1} = V(r, \alpha_1, ..., \alpha_i)$). This yields the following equation for the expected probability :

$$(2) \qquad E_{\beta_{i+1}}[F(\beta_1, \alpha_1, ..., \alpha_i, \beta_{i+1})] = \frac{1}{|R_{\beta_1,\alpha_1,...,\beta_i}|} \sum_{r \in R_{\beta_1,\alpha_1,...,\beta_i}} F(\beta_1, \alpha_1, ..., \alpha_i, V(r, \alpha_1, ..., \alpha_i))$$

Combining (1) and (2) we get the recursion formula

$$F(\beta_1, \alpha_1, ..., \alpha_{i-1}, \beta_i) = \max_{\alpha_i} \frac{1}{|R_{\beta_1,\alpha_1,...,\beta_i}|} \sum_{r \in R_{\beta_1,\alpha_1,...,\beta_i}} F(\beta_1, \alpha_1, ..., \alpha_i, V(r, \alpha_1, ..., \alpha_i))$$

We now show how to compute the function F in polynomial-space:
For each potential α_i, we enumerate all possible values of r. For each r, all of the following can be done in polynomial-space:

- Checking if $r \in R_{\beta_1,\alpha_1,...,\beta_i}$ by simulating the verifier in the first i interactions (when given r the verifier strategy is polynomial).
- Calculating $\beta_{i+1} = V(r, \alpha_1, ..., \alpha_i)$ again by simulating the verifier.
- Recursively computing $F(\beta_1, \alpha_1, ..., \alpha_i, \beta_{i+1})$.

  In order for the recursion to be polynomial-space computable, we need to show that the recursion stops after polynomially many stages, and that this stage can be computed in polynomial-space. The recursion stops when reaching a full transcript of the proof system. In such a case the prover can enumerate r and find the probability of acceptance among all consistent r by simulating the verifier. Clearly, this can be done in polynomial-space. Also the depth of the recursion must be at most polynomial, which is obviously the case here, since it is bounded by the number of rounds.

Using polynomial-size counters, we can sum the probabilities for all consistent r, and find the expected probability for each α_i. By repeating this for all possible α_i we can find one that maximizes the expectation. Altogether, the prover's optimal strategy can be calculated in polynomial-space.

Note: All the probabilities are taken over the verifier's coin tosses (no more than a polynomial number of coins). This enables us to use polynomial-size memory for calculating all probabilities with exact resolution (by representing them as rational numbers - storing the nominator and denominator separately). ■

Corollary 5 *IP* $\subseteq$ *PSPACE*

Proof: If $L \in$ IP then there exists an interactive proof system for L and hence there exists a polynomial-space optimal prover strategy. Given input x and the verifier's coin tosses, we can simulate (in polynomial-space) the interaction between the optimal prover and the verifier and determine this interaction's outcome. We enumerate over all the possible verifier's coin tosses and accept only if more than $\frac{2}{3}$ of the outcomes are accepting. Clearly, we accept if and only if $x \in L$ and this can be implemented in polynomial-space. ■

As mentioned above, we will not prove that PSPACE $\subseteq$ IP. Instead, we prove a weaker theorem, using a similar approach of employing algebraic methods.

Theorem 6 co$\mathcal{NP} \subseteq \mathcal{IP}$

Proof: We prove the theorem by presenting an interactive proof system for the coNP-complete problem $\overline{3SAT}$ (the same method can work for the problem $\overline{SAT}$ as well). $\overline{3SAT}$ is the set of non-satisfiable $3CNF$ formulae: Given a $3CNF$ formula ϕ, it is in the set if no truth assignment to it's variables satisfies the formula.

The proof uses an arithmetic generalization of the boolean problem, which allows us to apply algebraic methods in the proof system.

The Arithmetization of a Boolean CNF formula: Given the formula ϕ with variables $x_1, ..., x_n$ we perform the following replacements:

Boolean		Arithmetic
T	$\longrightarrow$	positive integers
F	$\longrightarrow$	0
x_i	$\longrightarrow$	x_i
$\overline{x_i}$	$\longrightarrow$	$(1 - x_i)$
$\vee$	$\longrightarrow$	$+$
$\wedge$	$\longrightarrow$	$\cdot$
$\phi(x_1, ..., x_n)$	$\longrightarrow$	$\Phi(x_1, ..., x_n)$

Every boolean $3CNF$ formula ϕ is transformed into a multi-variable polynomial Φ. It is easy to see that for every assignment $x_1, ..., x_n$, we have

$$\phi(x_1, ..., x_n) = F \Longleftrightarrow \Phi(x_1, ..., x_n) = 0$$

Summing over all possible assignments, we obtain an equation for the non-satisfiability of ϕ:

$$\phi \text{ is unsatisfiable } \Longleftrightarrow \sum_{x_1=0,1} ... \sum_{x_n=0,1} \Phi(x_1, ..., x_n) = 0$$

Suppose ϕ has m clauses of length three each, thus any 0-1 assignment to $x_1, ..., x_n$ gives $\Phi(x_1, ..., x_n) \leq 3^m$. Since there are 2^n different assignments, the sum above is bounded by $2^n \cdot 3^m$. This fact allows us to move our calculations to a finite field, by choosing a prime q such that $q > 2^n \cdot 3^m$, and working modulo this prime. Thus proving that ϕ is unsatisfiable reduces to proving that

$$\sum_{x_1=0,1} ... \sum_{x_n=0,1} \Phi(x_1, ..., x_n) \equiv 0 \pmod{q}$$

We choose q to be not much larger than $2^n \cdot 3^m$ (this is possible due to the density of the prime numbers). Thus, we obtain that all calculations over the field $GF(q)$ can be done in polynomial-time (in the input length). Working over a finite field will later help us in the task of uniformly selecting an element in the field.

The interactive proof system for $\overline{3SAT}$:

- Both sides receive the common boolean formula ϕ. They perform the arithmetization procedure and obtain Φ.
- The prover picks a prime q such that $q > 2^n \cdot 3^m$, and sends q to the verifier. The verifier checks that q is indeed a prime. If not he rejects.
- The verifier initializes $v_0 = 0$.
- The following is performed n times (i runs from 1 to n):
  - The prover sends a polynomial $P_i(\cdot)$ of degree at most m to the verifier.

- The verifier checks whether $P_i(0) + P_i(1) \equiv v_{i-1} \pmod q$ and that the polynomial's degree is at most m.

  If not, the verifier rejects.

  Otherwise, he uniformly picks $r_i \in_R GF(q)$, calculates $v_i = P_i(r_i)$ and sends r_i to the prover.

- The verifier accepts if $\Phi(r_1, ...r_n) \equiv v_n \pmod q$ and rejects otherwise.

Motivation: The prover has to find a sequence of polynomials that satisfies a number of restrictions. The restrictions are imposed by the verifier in the following interactive manner: after receiving a polynomial from the prover, the verifier sets a new restriction for the next polynomial in the sequence. These restrictions guarantee that if the claim holds (ϕ is unsatisfiable), such a sequence can always be found (we call it the "Honest prover strategy"). However, if the claim is false, any prover strategy has only a small probability of finding such a sequence (the probability is taken over the verifier's coin tosses). This yields the completeness and soundness of the suggested proof system. The nature of these restrictions is fully clarified in the proof of soundness, but we will first show that the verifier strategy is efficient.

The verifier strategy is efficient: Most steps in the protocol are calculations of polynomials of degree m in n variables, these are easily calculated in polynomial-time. The transformation to an arithmetic field is linear in the formula's length.

Checking primality is known to be in BPP and therefore can be done by the verifier. Furthermore, it can be shown that primality testing is in NP, so the prover can send the verifier an NP-witness to the fact that q is a prime, and the verifier checks this witness in polynomial-time.

Finally, picking an element $r \in_R GF(q)$ can be done in $O(\log q)$ coin tosses, that is polynomial in the formula's length.

The honest prover strategy: For every i define the polynomial:

$$P_i^*(z) = \sum_{x_{i+1}=0,1} ... \sum_{x_n=0,1} \Phi(r_1, ..., r_{i-1}, z, x_{i+1}, ..., x_n)$$

Note that $r_1, ..., r_{i-1}$ are constants set by the verifier in the previous stages and known to the prover at the i^{th} stage. z is the polynomial's variable.
The following facts are evident about P_i^*:

- Calculating P_i^* may take exponential-time, but this is no obstacle for a computationally unbounded prover.

- The degree of P_i^* is at most m. Since there are at most m clauses in ϕ, the highest degree of any one variable is m (if it appears in all clauses).

Completeness of the proof system: When the claim holds, the honest prover always succeeds in convincing the verifier. For $i > 1$:

$$(3.1) \quad P_i^*(0) + P_i^*(1) = \sum_{x_i=0,1} P_i^*(x_i) =_{(1)} \sum_{x_i=0,1} ... \sum_{x_n=0,1} \Phi(r_1, ..., r_{i-1}, x_i, ..., x_n)$$

$$=_{(2)} P_{i-1}^*(r_{i-1}) \equiv_{(3)} v_{i-1} \pmod q$$

Equality (1) is due to the definition of P_i^*. Equality (2) is due to the definition of P_{i-1}^*. Equality (3) is the definition of v_{i-1}.
Also for $i = 1$, since the claim holds we have:

$$P_1^*(0) + P_1^*(1) = \sum_{x_1=0,1} P_1^*(x_1) = \sum_{x_1=0,1} \ldots \sum_{x_n=0,1} \Phi(x_1, \ldots, x_n) \equiv v_0 \pmod q$$

And finally: $v_n = P_n^*(r_n) = \Phi(r_1, \ldots, r_n)$.

We showed that the polynomials of the honest prover pass all of the verifier's tests, obtaining perfect completeness of the proof system.

Soundness of the proof system: If the claim is false, an honest prover will definitely fail after sending P_1^*, thus a prover must be dishonest.

Roughly speaking, we will show that if a prover is dishonest in one round, then with high probability he must be dishonest in the next round as well. In the last round, his dishonesty is revealed. Formally:

Lemma 3.2 *If $P_i^*(0) + P_i^*(1) \not\equiv v_{i-1} \pmod q$ then either the verifier rejects in the i^{th} round, or $P_i^*(r_i) \not\equiv v_i \pmod q$ with probability at least $1 - \frac{m}{q}$, where the probability is taken over the verifier's choices of r_i.*

We stress that P_i^* is the polynomial of the honest prover strategy (as defined above), while P_i is the polynomial actually sent by the prover (v_i is set using P_i).
Proof: (of lemma) If the prover sends $P_i = P_i^*$, we get:

$$P_i(0) + P_i(1) \equiv P_i^*(0) + P_i^*(1) \not\equiv v_{i-1} \pmod q$$

and the verifier rejects immeadiately.

Otherwise the prover sends $P_i \neq P_i^*$. We assume P_i passed the verifier's test (if not the verifier rejects and we are done). Since P_i and P_i^* are of degree at most m, there are at most m choices of $r_i \in GF(q)$ such that

$$P_i^*(r_i) \equiv P_i(r_i) \pmod q$$

For all other choices:

$$P_i^*(r_i) \not\equiv P_i(r_i) \equiv v_i \pmod q$$

Since the verifier picks $r_i \in_R GF(q)$, we get $P_i^*(r_i) \equiv v_i \pmod q$ with probability at most $\frac{m}{q}$, ■

Suppose the verifier does not reject in any of the n rounds. Since the claim is false (ϕ is satisfiable), we have $P_1^*(0) + P_1^*(1) \not\equiv v_0 \pmod q$. Applying alternately the lemma and the following equality: for every $i \geq 2$ $P_{i-1}^*(r_{i-1}) = P_i^*(0) + P_i^*(1)$ (due to equation 3.1) , we get that $P_n^*(r_n) \not\equiv v_n \pmod q$ with probability at least $(1 - \frac{m}{q})^n$. But $P_n^*(r_n) = \Phi(r_1, \ldots, r_n)$ so the verifier's last test fails and he rejects. Altogether the verifier fails with probability $(1 - \frac{m}{q})^n > 1 - \frac{nm}{q} > \frac{2}{3}$ (by the choice of q). ■

4 Public-Coin Systems and the Number of Rounds

An interesting question is how the power of interactive proof systems is affected by the number of rounds allowed. We have already seen that GNI can be proved by an interactive proof with 2 rounds. Despite this example of a coNP language, we conjecture that coNP $\not\subseteq$ IP$(O(1))$. Together with the previous theorem we get:

Conjecture 7

$$\mathcal{IP}(O(1)) \subset \mathcal{IP}(poly) \qquad \textit{(strict containment)}$$

A useful tool in the study of interactive proofs, is the public coin variant, in which the verifier can only ask random questions.

Definition 8 (public-coin interactive proofs – $\mathcal{AM}$): *Public coin proof systems (known also as Arthur-Merlin games) are a special case of interactive proof systems, in which, at each round, the verifier can only toss coins, and send their outcome to the prover. In the last round, the verifier decides whether to accept or reject.*

For every integer function $r(\cdot)$, the complexity class $\mathcal{AM}(r(\cdot))$ consists of all the languages that have an Arthur-Merlin proof system in which, on common input x, at most $r(|x|)$ rounds are used.

Denote $\mathcal{AM} = \mathcal{AM}(2)$.

We note that the definition of $\mathcal{AM}$ as Arthur-Merlin games with two rounds is inconsistent with the notation IP= IP(*poly*)). (Unfortunately, that's what is found in the literature.)

The difference between the Arthur-Merlin games and the general interactive proof systems can be viewed as the difference between asking tricky questions, versus asking random questions. Surprisingly it was shown that these two versions are essentially equivalent:

Theorem 9 (Relating $\mathcal{IP}(\cdot)$ to $\mathcal{AM}(\cdot)$):

$$\forall r(\cdot) \quad \mathcal{IP}(r(\cdot)) \subseteq \mathcal{AM}(r(\cdot)+2)$$

The following theorem shows that power of $\mathcal{AM}(r(\cdot))$ is invariant under a linear change in the number of rounds:

Theorem 10 (Linear Speed-up Theorem):

$$\forall r(\cdot) \geq 2 \quad \mathcal{AM}(2r(\cdot)) = \mathcal{AM}(r(\cdot))$$

The above two theorems are quoted without proof. Combining them we get:

Corollary 11 $\forall r(\cdot) \geq 2 \quad \mathcal{IP}(2r(\cdot)) = \mathcal{IP}(r(\cdot))$.

Corollary 12 (Collapse of constant-round IP to two-round AM):

$$\mathcal{IP}(O(1)) = \mathcal{AM}(2)$$

5 Perfect Completeness and Soundness

In the definition of interactive proof systems we require the existence of a prover strategy that for $x \in L$ convinces the verifier with probability at least $\frac{2}{3}$ (analogous to the definition of the complexity class BPP). One can consider a definition requiring *perfect completeness*; i.e., convincing the verifier with probability 1 (analogous to coRP). We will now show that the definitions are equivalent.

Theorem 13 *If a language L has an interactive proof system then it has one with perfect completeness.*

We will show that given a public coin proof system we can construct a perfect completeness public coin proof system.

We use the fact that public coin proof systems and interactive proof systems are equivalent (see Theorem 9), so if L has an interactive proof system it also has a public coin proof system. We define:

$$AM^0(r(\cdot)) \quad = \quad \{L \mid \text{ L has perfect completeness } r(\cdot) \text{ round public coin proof system}\}$$

So given an interactive proof system we create a public coin proof system and using the following lemma convert it to one with perfect completeness. Thus, the above theorem which refers to arbitrary interactive proofs follows from the following lemma which refers only to public-coin interactive proofs.

Lemma 5.1 *If L has a public coin proof system then it has one with perfect completeness*

$$AM(r(\cdot)) \subseteq AM^0(r(\cdot)+1)$$

Proof: Given an Arthur-Merlin proof system, we construct an Arthur-Merlin proof system with perfect completeness and one more round. We use the same idea as in the proof of BPP $\subseteq$ PH.

Assume, without loss of generality, that the Arthur-Merlin proof system consists of $2t$ rounds, and that Arthur sends the same number of coins m in each round (otherwise, ignore the redundant coins). Also assume that the completeness and soundness error probabilities of the proof system are at most $\frac{1}{3tm}$. This is obtained using amplification (see Section 2.1).

We denote the messages sent by Arthur (the verifier) $r_1, ..., r_t$ and the messages sent by Merlin (the prover) $\alpha_1, ..., \alpha_t$. Denote by $\langle P, V\rangle_x(r_1, ..., r_t)$ the outcome of the game on common input x between the optimal prover P and the verifier V in which the verifier uses coins $r_1, ..., r_t$: $\langle P, V\rangle_x(r_1, ..., r_t) = 0$ if the verifier rejects and $\langle P, V\rangle_x(r_1, ..., r_t) = 1$ otherwise.

We construct a new proof system with perfect completeness, in which Arthur and Merlin play tm games simultaneously. Each game is like the original game except that the random coins are shifted by a fixed amount. The tm shifts (one for each game) are sent by Merlin in an additional round at the beginning. Arthur accepts if at least one of the games is accepting. Formally, we add an additional round at the beginning in which Merlin sends the shifts $S^1, ..., S^{tm}$ where $S^i = (S^i_1, ..., S^i_t), S^i_j \in \{0,1\}^m$ for every i between 1 and tm. Like in the original proof system Arthur sends messages $r_1, ..., r_t$, where $r_i \in_R \{0,1\}^m$. For game i and round j, Merlin considers the random coins to be $r_j \oplus S^i_j$ and sends as a message α^i_j where α^i_j is computed according to $(r_1 \oplus S^i_1, ..., r_j \oplus S^i_j)$. The entire message in round j is $\alpha^1_j, ..., \alpha^{tm}_j$. At the end of the protocol Arthur accepts if at least one out of the tm games is accepting.

In order to show perfect completeness we will show that for every $x \in L$ there exist $S^1, ..., S^{tm}$ such that for all $r_1, ...r_t$ at least one of the games is accepting. We use a probabilistic argument to show that the complementary event occurs with probability strictly smaller than 1.

$$\Pr_{S^1,...S^{tm}}[\exists r_1, ...r_t \bigwedge_{i=1}^{tm} (\langle P, V\rangle_x(r_1 \oplus S^i_1, ..., r_t \oplus S^i_t) = 0)]$$

$$\leq_{(1)} \sum_{r_1,...r_t \in \{0,1\}^m} \Pr_{S^1,...S^{tm}}[\bigwedge_{i=1}^{tm} (\langle P, V\rangle_x(r_1 \oplus S^i_1, ..., r_t \oplus S^i_t) = 0)]$$

$$\leq_{(2)} 2^{tm} \cdot (\frac{1}{3tm})^{tm} < 1$$

Inequality (1) is obtained using the union bound. Inequality (2) is due to the fact that the $r_j \oplus S^i_j$ are independent random variables so the results of the games are independent, and that the optimal prover fails to convince the verifier on a true assertion with probability at most $\frac{1}{3tm}$.

We still have to show that the proof system suggested satisfies the soundness requirement. We show that for every $x \notin L$ and for any prover strategy P^* and choices of shifts $S^1, ..., S^{tm}$ the probability that one or more of the tm games is accepting is at most $\frac{1}{3}$.

$$\mathrm{Pr}_{r_1,...r_t}[\bigvee_{i=1}^{tm}(\langle P^*, V\rangle_x(r_1 \oplus S^i_1, ..., r_t \oplus S^i_t) = 1)]$$

$$\leq_{(1)} \sum_{i=1}^{tm} \mathrm{Pr}_{r_1,...,r_t}[\langle P^*, V\rangle_x(r_1 \oplus S^i_1, ..., r_t \oplus S^i_t) = 1]$$

$$\leq_{(2)} \sum_{i=1}^{tm} \frac{1}{3tm} = \frac{1}{3}$$

Inequality (1) is obtained using the union bound. Inequality (2) is due to the fact that any prover has probability of at most $\frac{1}{3tm}$ of success for a single game (because any strategy that the prover can play in a copy of the parallel game can be played in a single game as well). ■

Unlike the last theorem, requiring *perfect soundness* (i.e. for every $x \notin L$ and every prover strategy P^*, the verifier always rejects after interacting with P^* on common input x) reduces the model to an NP-proof system , as seen in the following proposition:

Proposition 5.2 *If a language L has an interactive proof system with perfect soundness then $L \in \mathcal{NP}$.*

Proof: Given an interactive proof system with perfect soundness we construct an NP proof system. In case $x \in L$, by the completeness requirement, there exists an accepting transcript. The prover finds an outcome of the verifier's coin tosses that gives such a transcript and sends the full transcript along with the coin tosses. The verifier checks in polynomial time that the transcript is valid and accepting and if so - accepts. This serves as an NP-witness to the fact that $x \in L$. If $x \notin L$ then due to the perfect soundness requirement, no outcome of verifier's coin tosses yields an accepting transcript and therefore there are no NP-witnesses. ■