7 Interactive Proofs

A proof is a way of convincing a party of a certain claim. When talking about proofs, we consider two parties: a *prover* and a *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 all 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, which leads 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.

Consider, for example, the SAT problem. Given a satisfiable Boolean expression ϕ, it is (believed to be) hard to *find* an assignment that satisfies ϕ, but it is easy to *verify* that a given assignment satisfies ϕ. A non-deterministic Turing machine M only has to worry about the verification part because it can "guess" a satisfying assignment (if it exists) and then verify whether it is indeed a satisfying assignment for ϕ or not. If so, M accepts, and otherwise M rejects. According to the definition of non-deterministic Turing machines, such a Turing machine would correctly decide L.

We can view M's actions above as a proof system where the prover is the part guessing the satisfying assignment and the verifier is the part checking whether the assignment is indeed satisfying ϕ. In general, a good proof system should 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 proof y exists for x).
   - **Soundness:** For a false assertion, no convincing proof strategy exists (in the case of NP, if $x \notin L$ then no proof y exists that can convince the verifier that $x \in L$).

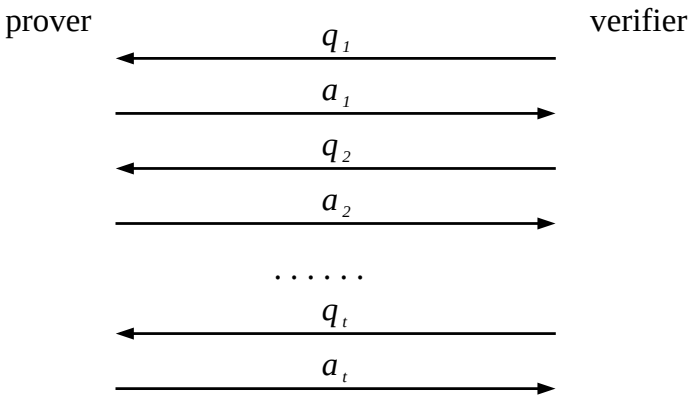
Notice that the prover can be infinitely powerful. We only have restrictions on the verification process. From our discussion above, the conditions can be satisfied for SAT when using the following verifier strategy:

The verifier asks the prover for a satisfying assignment for ϕ and expects the prover to return some assignment T. If not, the verifier rejects. If the prover delivers a T, the verifier checks whether $T \models \phi$. If so, the verifier accepts, and otherwise it rejects.

It is easy to verify that this strategy indeed satisfies the conditions for a good proof system above. Since SAT is NP-complete, this means that every problem in NP has a good proof system.

In the following, 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 q_i and the prover answers with a message a_i. At the end of the interaction, the verifier decides based on the knowledge he acquired in the process whether the claim is true or false.



7.1 The definition of IP

Following the above discussion, we define

Definition 7.1 *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.*
2. *Correctness requirements:*
   - **Completeness:** *For every $x \in L$, there exists a prover strategy P such that when interacting on the common input x, the prover P convinces the verifier with probability at least $2/3$.*
   - **Soundness:** *For every $x \notin L$, when interacting on the common input x, every prover strategy P^* can convince the verifier with probability at most $1/3$.*

The prover strategy is computationally unbounded.

For our proofs later, it will be important to specify a bit more clearly what we mean by "prover strategy". For any fixed input, any prover strategy can be seen as a partial mapping $f : \{0,1\}^* \to \{0,1\}^*$ that specifies what reply to give back to the verifier under a given history

of queries and replies. Assume, for example, that the current history of queries and replies is $q_1, a_1, q_2, a_2, \ldots, q_k$. Then the answer a_k can only depend on these queries and replies because (apart from the input) this is the only information the prover has. Given that queries and replies are given as binary strings, any $q_1, a_1, q_2, a_2, \ldots, q_k$ can be encoded as a string $q_1\#a_1\#q_2\#a_2 \ldots q_k$ which can be converted into a binary string using the following transformation rules: $0 \rightarrow 00$, $1 \rightarrow 10$, and $\# \rightarrow 11$. Hence, it indeed suffices to consider a mapping f on the set of binary strings. f only needs to be a partial mapping since the verifier is required to be a probabilistic polynomial time procedure. Hence, there is a polynomial $p(n)$ so that the runtime of the verifier is bounded by $p(n)$, and therefore the maximum length of strings relevant for f is $2p(n)$.

The reason why we only need to consider fixed partial mappings, i.e., deterministic provers, is that if the completeness and soundness conditions hold for randomized provers (i.e., a prover that, given a history $x \in \{0,1\}^*$, has a probability distribution on the next reply), they also hold for deterministic provers, and therefore randomization on the prover side does not add anything to the complexity of our interactive proofs.

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 $\mathrm{IP}(r(\cdot))$ 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

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

It is not difficult to check that $\mathrm{BPP} = \mathrm{IP}(0)$ and $\mathrm{NP} \subseteq \mathrm{IP}(1)$. Furthermore, the number of rounds in IP can never 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 $\mathrm{IP} = \mathrm{IP}(poly)$. Also, we can assume that the length of the messages exchanged between the prover and the verifier is a polynomial in the length of the common input, since in a polynomial number of steps the verifier can only read and write a polynomial number of symbols.

The following claim shows that the probabilities $2/3$ and $1/3$ 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 $p(\cdot)$.

Claim 7.2 *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 the majority vote. Let the random variable X denote the number of accepting votes. If the assertion holds, then X is the sum of k independent Bernoulli trials with probability of success at least $2/3$. An error in the new protocol only happens if $X \leq k/2$. Hence,

$$\begin{aligned} \Pr[\text{error}] &\leq \Pr[X \leq k/2] \leq \sum_{i=0}^{k/2} \binom{k}{i} \left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{k-i} \\ &\leq \sum_{i=0}^{k/2} \binom{k}{i} \left(\frac{2}{3}\right)^i \left(\frac{1}{3}\right)^{k-i} \left(\frac{2/3}{1/3}\right)^{k/2-i} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{k/2} \binom{k}{i} \left(\frac{2}{3}\right)^{k/2} \left(\frac{1}{3}\right)^{k/2} \\
&= \left(\frac{2}{9}\right)^{k/2} \sum_{i=0}^{k/2} \binom{k}{i} \\
&\leq \left(\frac{2}{9}\right)^{k/2} \cdot 2^k \leq \left(\frac{8}{9}\right)^{k/2} \leq \frac{1}{2^{k/12}} .
\end{aligned}
$$

Since k can be an arbitrary polynomial, the claim follows. □

We note that introducing both interaction and randomness in the IP class is essential.

By adding interaction only, the interactive proof systems collapse to NP-proof systems, because for a deterministic verifier, the prover can predict the verifier's part on the interaction and send the full transcript as an NP-witness. The verifier then just has to check that the witness is a valid and accepting transcript of the original proof system.

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 the class IP(1), which seems to be simply a randomized (and perhaps stronger) version of NP.

7.2 Graph non-isomorphism

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 one-to-one 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 $G_1 \ncong G_2$).

The *graph non-isomorphism problem* is defined as

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

In the following we describe 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 $V_1 = V_2 = \{1, \ldots, n\}$.
- The verifier chooses $i \in \{1, 2\}$ and $\pi \in S_n$ (S_n is the group of all permutations on $\{1, \ldots, n\}$) uniformly at random. He applies the mapping π on the graph G_i to obtain a graph $H = (\{1, \ldots, n\}, E_H)$ where $E_H = \{(\pi(u), \pi(v)) \mid (u, v) \in E_i\}$ and sends the graph H to the prover.
- The prover sends $j \in \{1, 2\}$ to the verifier.
- The verifier accepts if and only if $j = i$.

Claim 7.3 *The protocol is an interactive proof system for GNI.*

Proof. We show that the above protocol is an interactive proof system with soundness probability at most 1/2 rather than 1/3 as in the formal definition. However, this is equivalent due to Claim 7.2.

First we note that the verifier's strategy can be easily implemented in probabilistic polynomial time. Hence, it remains to check the completeness and soundness of the proof system.

- Completeness: In case $G_1 \ncong G_2$, every graph can be isomorphic to at most one of G_1 or G_2 (otherwise, $G_1 \cong G_2$). It follows that the prover can always send the correct j and therefore Pr[verifier accepts] = 1.

- Soundness: In case $G_1 \cong G_2$, every graph H sent to the prover has an equal probability to come from G_1 or G_2. To prove this, let

$$S_{G_k} = \{\sigma \in S_n : \ \sigma(G_k) = H\}$$

for $k = 1, 2$ and 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 $\sigma\tau \in S_{G_2}$, and for every $\sigma \in S_{G_2}$ it follows that $\sigma\tau^{-1} \in S_{G_1}$. Hence, $|S_{G_1}| = |S_{G_2}|$, and therefore, given that H was sent, the probability that the verifier chose $i = 1$ is equal to the probability of the choice $i = 2$. Since the prover does not know the random choices of the verifier, for every decision the prover makes he has success probability 1/2, and therefore his total probability of success is 1/2.

□

The above protocol is implemented with only 2 rounds. Therefore,

Corollary 7.4 $GNI \in \text{IP}(2)$.

Note that for the protocol for the GNI problem to work it is essential that the random choices of the verifier are kept *private*. If we require the random choices to be public, interactive proof systems are usually called *Arthur Merlin games*. It was shown by Goldwasser and Sipser that the power of interactive proofs and Arthur Merlin games is equivalent. Hence, it is also possible to construct a protocol for the GNI problem in which the verifier can reveal his random choices to the prover.

7.3 #SAT

Next we look at a more complex problem that will help us determining the power of the complexity class IP. Let

$$\#\text{SAT} = \{\langle \phi, k \rangle \mid \phi \text{ is a CNF-formula with exactly } k \text{ satisfying assignments}\}$$

We show:

Theorem 7.5 #SAT $\in$ IP.

Proof. Consider any instance $\langle \phi, k \rangle$, and suppose ϕ has variables $x_1, \ldots, x_m \in \{0, 1\}$ where 0 represents "false" and 1 represents "true". For any $0 \leq i \leq m$ and $a_1, \ldots, a_i \in \{0, 1\}$ let

$$f_i(a_1, \ldots, a_i) = \text{number of satisfying assignments given that } x_1 = a_1, \ldots, x_i = a_i$$

Then it holds:

- $f_0()$ gives the total number of satisfying assignments for ϕ

- $f_m(a_1, \ldots, a_m) \in \{0, 1\}$
- $f_i(a_1, \ldots, a_i) = f_{i+1}(a_1, \ldots, a_i, 0) + f_{i+1}(a_1, \ldots, a_i, 1)$ for every $i < m$.

In order to specify f_i in terms of ϕ we need to arithmetize ϕ. For this, we transform ϕ into a polynomial $p(x_1, \ldots, x_m)$ by using the following rules:

- $\alpha \wedge \beta \rightarrow \alpha \cdot \beta$
- $\neg\alpha \rightarrow (1 - \alpha)$
- $\alpha \vee \beta \rightarrow \alpha * \beta = 1 - (1 - \alpha)(1 - \beta)$ (recall that $\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$)

Using these rules, the following properties easily follow:

- $p(a_1, \ldots, a_m) \in \{0, 1\}$ for all $a_1, \ldots, a_m \in \{0, 1\}$
- $\phi(a_1, \ldots, a_m)$ satisfied $\Leftrightarrow p(a_1, \ldots, a_m) = 1$
- $\deg(p) \leq n$, i.e., p can only have terms of the form x_1^n or $x_2^{n-3} \cdot x_3^3$ but not x_3^{n+1} or $x_4^4 \cdot x_5^{n-3}$

Hence, for all i and all $a_1, \ldots, a_i \in \{0, 1\}$,

$$f_i(a_1, \ldots, a_i) = \sum_{a_{i+1}, \ldots, a_m \in \{0,1\}} p(a_1, \ldots, a_m)$$

Thus, every f_i is a polynomial in $x_1, \ldots, x_i$, and $\deg(f) \leq n$.

Now we are ready to give the interactive proof for #SAT. Consider any field $\mathcal{F}$ of size q where $q \geq n^4$ is a prime. In the following, we assume that all the arithmetic is done in $\mathcal{F}$. Let P denote the prover and V denote the verifier. The verifier works as follows:

- Phase 0: P sends $f_0()$ to V. V rejects if $k \neq f_0()$ (because P failed to convince V that $\langle \phi, k \rangle \in$ #SAT). Otherwise, V continues.
- Phase 1: P sends $f_1(z)$ to V. V checks that $\deg(f_1) \leq n$ and $f_0() = f_1(0) + f_1(1)$. If any of the conditions is violated, V rejects. Otherwise, V sends a random $r_1 \in \mathcal{F}$ to P.
- Phase 2: P sends $f_2(r_1, z)$ to V. V checks that $\deg(f_2) \leq n$ and $f_1(r_1) = f_2(r_1, 0) + f_2(r_1, 1)$. If any of the conditions is violated, V rejects. Otherwise, V sends a random $r_2 \in \mathcal{F}$ to P.
- Phase 3: P sends $f_3(r_1, r_2, z)$ to V. V checks that $\deg(f_3) \leq n$ and $f_2(r_1, r_2) = f_3(r_1, r_2, 0) + f_3(r_1, r_2, 1)$. If any of the conditions is violated, V rejects. Otherwise, V sends a random $r_3 \in \mathcal{F}$ to P.
- $\ldots$
- Phase m: P sends $f_m(r_1, \ldots, r_{m-1}, z)$ to V. V checks that $\deg(f_m) \leq n$ and $f_{m-1}(r_1, \ldots, r_{m-1}) = f_m(r_1, \ldots, r_{m-1}, 0) + f_m(r_1, \ldots, r_{m-1}, 1)$. If any of the conditions is violated, V rejects.
- Phase $m+1$: V evaluates $p(r_1, \ldots, r_m)$ and compares this with $f_m(r_1, \ldots, r_m)$. If they are equal, V accepts, and otherwise V rejects.

Next we show that this interactive proof is in IP. If $\langle \phi, k \rangle \in \#\text{SAT}$, then P just gives the correct polynomials (which it can do because it has infinite power) and V will always accept.

So suppose that $\langle \phi, k \rangle \notin \#\text{SAT}$. Then we will show that no prover is able to fool the verifier with more than probability 1/3. For this we need the following number-theoretic result.

Lemma 7.6 *Let $\mathcal{F}$ be a finite field with q elements and p be a non-zero polynomial on the variables $x_1, \ldots, x_m$, where each variable has a degree of at most d. If $a_1, \ldots, a_m$ are selected randomly in $\mathcal{F}$, then*

$$\Pr[p(a_1, \ldots, a_m) = 0] \leq \frac{m \cdot d}{q}$$

For a proof see, for example, the book by Sipser. This immediately implies the following result.

Lemma 7.7 *Let $\mathcal{F}$ be a finite field with q elements and p and p' be any two distinct polynomials on the variables $x_1, \ldots, x_m$, where each variable has a degree of at most d. If $a_1, \ldots, a_m$ are selected randomly in $\mathcal{F}$, then*

$$\Pr[p(a_1, \ldots, a_m) = p'(a_1, \ldots, a_m)] \leq \frac{m \cdot d}{q}$$

This lemma will be useful for our proof. Suppose that the function provided by the prover in phase i is $\tilde{f}_i$. Because $\langle \phi, k \rangle \notin \#\text{SAT}$, $f_0() \neq k$, and therefore the prover must provide a $\tilde{f}_0()$ that is different from $f_0()$ in order to prevent the verify from rejecting the input right away. But if $\tilde{f}_0() \neq f_0()$, then also $\tilde{f}_1(z) \neq f_1(z)$ because otherwise V would discover that the condition $\tilde{f}_0() = \tilde{f}_1(0) + \tilde{f}_1(1)$ is violated. Since V picks a random $r_1 \in \mathcal{F}$ it follows from Lemma 7.7 that

$$\Pr[\tilde{f}_1(r_1) = f_1(r_1)] \leq \frac{m \cdot n}{n^4} \leq \frac{1}{n^2}$$

If an r_1 is picked so that $\tilde{f}_1(r_1) = f_1(r_1)$, then P was lucky and does not have to lie any more about the functions f_i afterwards. Thus, V will accept the input in this case. But if $\tilde{f}_1 \neq f_1(r_1)$, then the prover must provide a $\tilde{f}_2(r_1, z)$ with $\tilde{t}_2(r_1, z) \neq f_2(r_1, z)$ because otherwise V would discover that the condition $\tilde{f}_1(r_1) = \tilde{f}_2(r_1, 0) + \tilde{f}_2(r_1, 1)$ is violated. In general, it follows that as long as an r_i is picked so that $\tilde{f}_i(r_1, \ldots, r_i) \neq f_i(r_1, \ldots, r_i)$, P must provide a $\tilde{f}_{i+1}(r_1, \ldots, r_i, z)$ with $\tilde{f}_{i+1}(r_1, \ldots, r_i, z) \neq f_{i+1}(r_1, \ldots, r_i, z)$. Furthermore,

$$\Pr[\tilde{f}_i(r_1, \ldots, r_i) \neq f_i(r_1, \ldots, r_i)] \leq \frac{1}{n^2}$$

for all i. If P has to continue to lie about f_i till the end, then V will discover this lie because in phase $m+1$ it can directly check whether $\tilde{f}_m(r_1, \ldots, r_m) = p(r_1, \ldots, r_m)$. Hence, the probability that V accepts the input is at most $m \cdot 1/n^2 \leq 1/n$. □

7.4 The complexity of IP

We have already seen that NP $\subseteq$ IP. Next we show a much stronger result.

Theorem 7.8 IP = PSPACE.

Proof. We first show that IP $\subseteq$ PSPACE. Let L be a language in IP. Then there is a verifier V that fulfills the conditions of an interactive proof system. That is, for any input x it holds that if $x \in L$ then there is a prover P such that V accepts x with probability at least $2/3$, and if $x \notin L$ then there is no prover P such that V accepts x with probability at least $1/3$. Hence, it suffices to find out what the probability of acceptance for V is, given a best possible prover strategy.

Since the verifier is polynomial time bounded, there must be a polynomial $p(\cdot)$ such that for any input of size n the number of rounds, the message sizes, and the number of computational steps of the verifier is at most $p(n)$. Without loss of generality, we assume that all random choices of the verifier are known to the prover (see the comments about Arthur Merlin games above). We also assume that the verifier plays first. Let q_i be the ith message sent by the verifier and a_i be the ith message sent by the prover. Furthermore, let r_i be the random coin tosses of the verifier between the $(i-1)$th and ith round. Since any fixed sequence $r_1, a_1, r_2, a_2, \ldots, r_i, a_i$ uniquely determines the questions $q_1, q_2, \ldots, q_i$ of the verifier, we will only consider its random choices and the answers of the prover in the following.

Given some fixed input x, $|x| = n$, let $\hat{P}_x$ denote the probability of acceptance for V, given a best possible prover strategy. Obviously,

$$\hat{P}_x = \sum_{r_1} \Pr[r_1] \cdot \hat{P}_x(r_1) \ ,$$

where $\hat{P}_x(r_1)$ is the maximum probability of acceptance given the random choices r_1. Furthermore,

$$\hat{P}_x(r_1) = \max_{a_1} \hat{P}_x(r_1, a_1) \ ,$$

where $\hat{P}_x(r_1, a_1)$ is the maximum probability of acceptance given r_1 and answer a_1. In general we get for any $i \geq 1$ that

$$\hat{P}_x(r_1, a_1, \ldots, r_i, a_i) = \sum_{r_{i+1}} \Pr[r_{i+1} \mid r_1, a_1, \ldots, r_i, a_i] \cdot \hat{P}_x(r_1, a_1, \ldots, r_i, a_i, r_{i+1})$$

and

$$\hat{P}_x(r_1, a_1, \ldots, r_i, a_i, r_{i+1}) = \max_{a_{i+1}} \hat{P}_x(r_1, a_1, \ldots, r_i, a_i, r_{i+1}, a_{i+1}) \ .$$

Since V decides whether to accept or reject after at most $p(n)$ rounds, there is for every possible sequence of random choices and answers an $i \leq p(n)$ such that $\hat{P}_x(r_1, a_1, \ldots, r_i, a_i) \in \{0, 1\}$.

It remains to show that $\hat{P}_x$ can be computed in polynomial space. For this we use the following algorithm:

Compute $\hat{P}_x$ in a depth-first-search manner, going through all possible choices for $r_1, a_1, \ldots, r_i$, a_i. For each round j, we store the current choice for r_j and for a_j. Furthermore, we store the current value of $\hat{P}_x(r_1, a_1, \ldots, r_j)$ (which holds the maximum $\hat{P}_x(r_1, a_1, \ldots, r_j, a_j)$ over all

choices for a_j investigated so far) and of $\hat{P}_x(r_1, a_1, \ldots, r_j, a_j)$ (which holds the sum of $\Pr[r_{i+1} \mid r_1, a_1, \ldots, r_j, a_j] \cdot \hat{P}_x(r_1, a_1, \ldots, r_j, a_j, r_{j+1})$ over all r_{j+1} investigated so far). Since storing r_j, a_j, $\hat{P}_x(r_1, a_1, \ldots, r_j)$, and $\hat{P}_x(r_1, a_1, \ldots, r_j, a_j)$ each requires at most $O(p(n))$ space and the maximum number of rounds for which these items have to be stored is at most $p(n)$, the space requirement for the computation of $\hat{P}_x$ is bounded by $O(p(n)^2)$, which is a polynomial.

Next we sketch the proof of PSPACE $\subseteq$ IP. Recall that QSAT is PSPACE-complete. Hence, if we can show that there is an IP for QSAT then we also showed that there is an IP for every problem in PSPACE, which implies that PSPACE $\subseteq$ IP.

Let ψ be a quantified Boolean expression of the form

$$\psi = Q_1 x_1 Q_2 x_2 \ldots Q_m x_m \ \phi$$

where ϕ is a Boolean expression in CNF and Q_i is $\exists$ or $\forall$. We define function f_i as for the #SAT problem, except that now we take the quantifiers into account. For $0 \leq i \leq m$ and $a_1, \ldots, a_m \in \{0, 1\}$ we want functions f_i so that

$$f_i(a_1, \ldots, a_i) = \begin{cases} 1 & \text{if } Q_{i+1} x_{i+1} \ldots Q_m x_m \ \phi(a_1, \ldots, a_i) \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

Then it holds:

- $f_0()$ gives the truth value of ψ
- $f_m(a_1, \ldots, a_m) \in \{0, 1\}$
- $Q_i = \forall$: $f_i(a_1, \ldots, a_i) = f_{i+1}(a_1, \ldots, a_i, 0) \cdot f_{i+1}(a_1, \ldots, a_i, 1)$
- $Q_i = \exists$: $f_i(a_1, \ldots, a_i) = f_{i+1}(a_1, \ldots, a_i, 0) * f_{i+1}(a_1, \ldots, a_i, 1)$

Recall that we defined $x * y$ as $1 - (1 - x)(1 - y)$.

A natural variation of the protocol for #SAT suggests itself where we apply the f_i's to a finite field and use the identities for the quantifiers instead of the identities for the summation. The problem with this idea is that, when arithmetized, every quantifier may double the degree of the resulting polynomial. The degrees of the polynomials might then grow exponentially large, which would require the verifier to run for an exponential time to process the exponentially many coefficients that the prover would need to send to describe the polynomials.

To keep the degrees of the polynomials small, we introduce a reduction operation R that reduces the degrees of the polynomials without changing their behavior on Boolean inputs. For the arithmetization of *phi* we replace ϕ by

$$\psi' = Q_1 x_1 R x_1 Q_2 x_2 R x_1 R x_2 Q x_3 R x_1 R x_2 R x_3 \ldots Q x_m R x_1 \ldots R x_m \ \phi$$

We rewrite ψ' as

$$\psi' = S_1 y_1 S_2 y_2 \ldots S_k y_k \ \phi$$

where each $S_i \in \{\forall, \exists, R\}$ and $y_i \in \{x_1, \ldots, x_m\}$. We define $f_k(x_1, \ldots, x_m)$ to be the polynomial $p(x_1, \ldots, x_m)$ obtained by arithmetizing ϕ. For $i < k$ we define f_i as

- $S_i = \forall$: $f_i(\ldots) = f_{i+1}(\ldots, 0) \cdot f_{i+1}(\ldots, 1)$
- $S_i = \exists$: $f_i(\ldots) = f_{i+1}(\ldots, 0) * f_{i+1}(\ldots, 1)$
- $S_i = R$: $f_i(\ldots, a) = (1-a)f_{i+1}(\ldots, 0) + af_{i+1}(\ldots, 1)$

The last input on the right hand side of the equations always represents y_{i+1}. Notice that the Rx operation on polynomials does not change their values on Boolean inputs. Therefore, $f_0()$ is still the truth value of ψ. However, the Rx operation produces a result that is linear in x. Thus, by adding $Rx_1 \ldots Rx_i$ after $Q_i x_i$ in ψ', we reduce the degree of each variable to 1 prior to the squaring due to arithmetizing Q_i.

The interactive proof and the proof of its correctness work now on ψ' as the interactive proof for ϕ in the #SAT proof. Formulating the interactive proof will be an assignment. □

7.5 Multiple-prover interactive proofs

In this section we consider an extension of the notion of an interactive proof system. Specifically, we consider the interaction of a verifier with several provers. The provers may share an a-priori selected strategy, but it is assumed that they cannot interact with each other during the time period in which they interact with the verifier.

Definition 7.9 *A* multiple-prover interactive proof system *for a language L is a game between a verifier and $k \geq 2$ provers that interact on a common input in a way satisfying the following properties:*

1. *The verifier strategy is a probabilistic polynomial time procedure.*
2. *The only interaction allowed is between the verifier and each of the provers.*
3. *A prover does not know the messages exchanged between the verifier and any other prover.*
4. *Correctness requirements:*
   - **Completeness:** *For every $x \in L$, there exists a strategy P for the provers such that when interacting on the common input x, P convinces the verifier with probability at least $2/3$.*
   - **Soundness:** *For every $x \notin L$, when interacting on the common input x, any strategy P^* of the provers convinces the verifier with probability at most $1/3$.*

The prover strategies are computationally unbounded.

The complexity class MIP consists of all the languages having a multiple-prover interactive proof system. For every $k \geq 2$, the complexity class k-IP consists of all the languages that have a k-prover interactive proof system. The following result was shown by Ben-Or, Goldwasser, Kilian, and Wigderson.

Theorem 7.10 2-IP = MIP

Proof. 2-IP $\subseteq$ MIP is obvious. So we only need to show that MIP $\subseteq$ 2-IP. Consider any k-prover interactive proof system between a verifier V and provers $P_1, \ldots, P_k$ that satisfies the conditions of a MIP. In order to simulate the interaction, we use a verifier $\tilde{V}$ and two provers $\tilde{P}_1$ and $\tilde{P}_2$. $\tilde{V}$ repeats the following protocol $T = \Theta(k)$ times in a row:

$\tilde{V}$ tosses all coins V ever uses and sends them to prover $\tilde{P}_1$. In return, $\tilde{P}_1$ sends $\tilde{V}$ the entire transcript of the interaction that would have occurred for these coin tosses between V and the provers $P_1, \ldots, P_k$. If this is an accepting conversation for V, $\tilde{V}$ now uses $\tilde{P}_2$ to check the validity of the conversation. This is done by $\tilde{V}$ selecting at random an original prover P_i and simulating with $\tilde{P}_2$ the conversation between V and P_i on these coin tosses. If the conversation does not match the conversation sent by $\tilde{P}_1$ then $\tilde{V}$ rejects the round, and otherwise $\tilde{V}$ accepts the round.

$\tilde{V}$ accepts only if it accepted *all* rounds and a majority of the conversations ends with V accepting it.

In order to show that $\tilde{V}$ satisfies the conditions of a 2-IP, we distinguish between two cases:

- $x \in L$: Then there is a prover strategy $(P_1, \ldots, P_k)$ so that V accepts with probability at least 2/3. Hence, if $\tilde{P}_1$ always sends the transcript of $(P_1, \ldots, P_k)$ to $\tilde{V}$, also $\tilde{V}$ will accept with probability at least 2/3.

- $x \notin L$: Then for all prover strategies $(P_1, \ldots, P_k)$, V accepts with probability at most 1/3. For any round t, let $(P_1^{(t)}, \ldots, P_k^{(t)})$ be the prover strategy that $\tilde{P}_1$ and $\tilde{P}_2$ agreed upon for round t of $\tilde{V}$. Once $\tilde{V}$ reveals its random coin tosses to $\tilde{P}_1$, $\tilde{P}_1$ can either choose to lie about the transcript of $(P_1^{(t)}, \ldots, P_k^{(t)})$ or to tell the truth about it. If $\tilde{P}_1$ lies, then there is at least one prover P_j for which its transcript deviates from the correct transcript for $(P_1^{(t)}, \ldots, P_k^{(t)})$. Hence, the probability that $\tilde{V}$ accepts the round in this case is at most $1 - 1/k$. Recall that $\tilde{V}$ only accepts x if it accepts *all* rounds. Suppose that $\tilde{P}_1$ lies in T' of the T rounds. Then the probability that $\tilde{V}$ accepts all rounds is

$$\left(1 - \frac{1}{k}\right)^{T'} \leq e^{-T'/k}$$

  If $T' \geq 2k$ then this is less than 1/3. Hence, if $\tilde{P}_1$ lies at least $2k$ times, then $\tilde{V}$ accepts x with probability at most 1/3, as desired. So suppose that $\tilde{P}_1$ lies at most $2k$ times. Then it holds that if $T = \Theta(k)$ is chosen large enough, the probability that $\tilde{V}$ accepts x can also be made to be at most 1/3. Determining this T is an assignment. Hence, also for $x \notin L$ the conditions of a 2-IP can be satisfied, which completes the proof.

□

Let NEXP $= \bigcup_{k \geq 1} NTIME(2^{n^k})$. The next theorem was shown by Babai, Fortnow, and Lund.

Theorem 7.11 MIP = NEXP

Proof. We will only show here that MIP $\subseteq$ NEXP. The proof for NEXP $\subseteq$ MIP is much more involved and therefore omitted here. The basic idea for that direction is similar to PSPACE $\subseteq$ IP: take a NEXP-complete problem and show that there is a MIP for it.

Let L be a language in MIP. Then there is a k-prover IP with verifier V that satisfies the conditions of a MIP (where k may depend on the input size but is at most polynomial in it because otherwise the verifier cannot be a probabilistic polynomial time procedure). That is, for any input x it holds that if $x \in L$ then there is a prover strategy $(P_1, \ldots, P_k)$ so that V accepts x with probability at least $2/3$, and if $x \notin L$ then there is no prover strategy $(P_1, \ldots, P_k)$ so that V accepts x with probability more than $1/3$. Our goal is to use this property in order to design a non-deterministic Turing machine M for L that has an accepting computation for x if and only if $x \in L$.

M does the following on a high level. Given any input x, M guesses a prover strategy $(P_1, \ldots, P_k)$ for x and then computes the probability that V accepts under $(P_1, \ldots, P_k)$. If this probability is at least $2/3$, then M accepts, and otherwise M rejects.

Suppose that the runtime of the verifier V is bounded by the polynomial $p(n)$. Then every prover strategy for V can be represented as a set of mappings $f_i : \{0,1\}^* \rightarrow \{0,1\}^*$, $i \in \{1, \ldots, k\}$, that map every binary string of length at most $2p(n)$ to a binary string of length at most $2p(n)$. The factor 2 is due to the encoding of a local transcript between V and a prover in binary form. For example, let 00 represent '0', 10 represent '1', and 11 represent '#'. Then any local transcript of the form $q_1 \# a_1 \# q_2 \# a_2 \ldots$ of length at most ℓ can be represented as a binary string of length at most 2ℓ.

Since fixing a mapping f_i requires at most $2p(n) \cdot 2^{2p(n)}$ bits, fixing a prover strategy requires at most $2p^2(n) \cdot 2^{2p(n)}$ bits. These bits can be guessed in exponential time by M.

In order to compute the probability of accepting x under a given prover strategy, M has to go through all possible random coin tosses for V. For any fixed set of coin tosses, M needs at most $O(p^3(n) \cdot 2^{2p(n)})$ time to compute the outcome of the interaction between V and the provers, and there are at most $2^{p(n)}$ possible outcomes for the coin tosses. Hence, M needs at most an exponential amount of time to determine the probability that V accepts x, which completes the proof. □

The theorem demonstrates that for the case of multiple provers, verifying a proof for a claim is significantly easier than proving the claim oneself.

7.6 References

- L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive proofs. *Computational Complexity* 1(1):3-40, 1991.
- M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson. Multi-prover interactive proofs: how to remove intractability assumptions. In *STOC 88*, pp. 113-131, 1988.
- O. Goldreich. Introduction to Complexity Theory. Lecture notes. Dept. of Computer Science and Applied Mathematics, Weizmann Institute.
- S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *STOC 85*, pp. 291-304, 1985. (See also *SIAM Journal on Computing* 18:186-208, 1989.)
- S. Goldwasser and M. Sipser. Private coins versus public coins in interactive proof systems. In *STOC 86*, pp. 59-68, 1986.

- A. Shamir. IP = PSPACE. *Journal of the ACM* 39(4):869-877, 1992.
- A. Shen. IP = PSPACE: simplified proof. *Journal of the ACM* 39(4):878-880, 1992.
- M. Sipser. *Introduction to the Theory of Computation*. PWS Publishing Company, 1997.