Resources in Interactive Proofs

CS6230: Topics in Information Security
Lecture 3 (Aug 24, 2021)
Lecture and notes: Prashant Nalini Vasudevan

Having asked how powerful interactive proofs are, we now move on to understanding their definition in greater detail. There were a number of choices we made while defining them that we will now look into more carefully, and understand to what extent these choices were arbitrary. Recall that the following is the original definition we saw earlier.

Definition 0.1. An interactive protocol (P, V), where V runs in polynomial time and is possibly randomised, is said to be and *interactive proof (IP) for a language L* if:

- **Completeness:** For every $x \in L$, when (P, V) is run with x as input, V accepts at the end with probability at least $2/3$.
- **Soundness:** For every $x \notin L$, and any prover strategy P^*, when (P^*, V) is run with x as input, V accepts at the end with probability at most $1/3$.

1 Errors

Let us start with the choice of the errors we allow in the completeness and soundness of an interactive proof. We set these to $1/3$, but these are somewhat arbitrary.

Theorem 1.1. *If a language L has an IP with completeness and soundness errors $1/3$, then it also has an IP with completeness and soundness errors $1/2^n$.*

The idea is to simply repeat the protocol several times, though proving this works can be tricky. We will start by proving the following more precise theorem in the special case of IPs with perfect completeness.

Theorem 1.2. *Suppose a language L has an IP with perfect completeness, soundness error $1/3$, and c bits of communication over r rounds. Then, for any $t \in \mathbb{N}$, L also has an IP with perfect completeness, soundness error $1/3^t$, and $t \cdot c$ bits of commmunication over $t \cdot r$ rounds.*

Proof. Given a protocol $\Pi = (P, V)$ with perfect completeness and soundness error $1/3$, the new protocol $\Pi' = (P', V')$ is obtained by repeating Π t times in sequence, and the accepting if and only if all the repetitions accept.

Efficiency. It is clear that the communication complexity and the number of rounds in Π' is t times that of Π. The same is roughly true of the prover and verifier's running time as well.

Completeness. For any $x \in L$, each repetition of Π will always accept. Thus, Π' also always accepts, and is perfectly complete.

Soundness. Given any $x \notin L$, we wish to prove that the probability that all t repetitions of the base protocol Π accept is at most $1/3^t$. Suppose this is not the case. Fix such an x and any (possibly cheating) prover strategy P'^* for Π' such that when run on x with the prover P'^*, the verifier V' accepts with probability more than $1/3^t$.

Denote by E_i the event that, in an execution of Π' with P'^*, the verifier V in the i^{th} repetition of Π accepts. In order for the verifier V' of Π' to accept, all of the events E_i have to happen. The probability of this happening is as follows:

$$\Pr[E_1 \wedge \cdots \wedge E_t] = \Pr[E_1] \cdot \Pr[E_2 \mid E_1] \cdot \cdots \cdot \Pr\left[E_t \mid \wedge_{i=1}^{t-1} E_i\right]$$

In order for the above probability to be more than $1/3^t$, at least one of the t terms in the right-hand side above has to be more than $1/3$. This gives the following claim.

Claim 1.1. *There is some $k \in [t]$ such that $\Pr\left[E_k \mid \wedge_{i=1}^{k-1} E_i\right] > 1/3$.*

Given such a k, we now construct a cheating prover P^* that contradicts the soundness of Π. This prover, before it starts interacting with the verifier in Π, samples for itself a random execution of the first $(k-1)$ repetitions of Π in the interaction between P'^* and V' such that all the events $E_1, \ldots, E_{k-1}$ happen. Then, it continues executing P'^* as though for the k^{th} repetition, but this time actually interacts with the verifier V instead of simulating it.

The probability that V now accepts is clearly $\Pr\left[E_k \mid \wedge_{i=1}^{k-1} E_i\right]$, which by Claim 1.1 is more than $1/3$, contradicting the soundness of Π. Thus, the P'^* that we assumed cannot exist, and the soundness error of Π' is at most $1/3^t$. □

Note that in the proof above, we relied crucially on the ability of the prover P^* to sample an execution of P'^* such that the first $(k-1)$ iterations of Π accept. In general, this could be very inefficient even if P'^* itself was efficient. This inefficiency can be replaced with non-uniformity where P^* could simply be hard-coded with a specific transcript of the first $(k-1)$ iterations of Π such that conditioned on the transcript of Π' being this, E_k happens with probability more than $1/3$.

Exercise 1. *Show that the transcript described above exists if P'^* as in the proof of Theorem 1.2 exists.*

In order to extend the above theorem to the case where Π does not have perfect completeness, we will need the following bound.

Lemma 1.3 (The Chernoff Bound)**.** *Let the $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$. Let $X = \sum_{i=1}^n X_i$, and $\mathrm{E}[X] = \mu$. Then, for any $\delta \in [0,1]$,*

$$\Pr[|X-\mu| \geq \delta\mu] \leq e^{-\frac{\delta^2\mu}{3}}$$

Theorem 1.4. *Suppose a language L has an IP with completeness and soundness errors $1/3$, and c bits of communication over r rounds. Then, for any $t \in \mathbb{N}$, L also has an IP with perfect completeness and soundness errors $1/2^{\Omega(t)}$, and $t \cdot c$ bits of commmunication over $t \cdot r$ rounds.*

Proof. Given a protocol $\Pi = (P, V)$ with completeness and soundness errors $1/3$, the new protocol $\Pi' = (P', V')$ is obtained by repeating Π t times in sequence, and the accepting if and only if *a majority of* the repetitions accept.

Efficiency. It is again clear that the communication complexity and the number of rounds in Π' is t times that of Π. The same is roughly true of the prover and verifier's running time as well.

Completeness. For any $x \in L$, each repetition of Π will accept with probability at least $2/3$. Define indicator random variables $X_1, \ldots, X_t$ such that $X_i = 1$ if the i^{th} repetition of Π accepts, and is 0 otherwise. Let $X = \sum_{i=1}^t X_i$. Note that the probability that each repetition accepts is at least $2/3$, and so $\mathrm{E}[X] \geq 2t/3$. We can thus bound the probability that Π' does not accept as follows using the Chernoff bound:

$$\Pr[X \leq t/2] = \Pr[(2t/3 - X) \geq t/6] \leq \Pr[(\mathrm{E}[X] - X) \geq t/6] \leq e^{-\Omega(t)}$$

Soundness. In order to bound the soundness error of Π' we will make explicit an intuitive fact that was implicit in the proof of Theorem 1.2 – that even in the best approach a cheating prover could take, it can act independently in each of the t repetitions of Π. For the purpose of this proof, we say that a prover P' for Π' is *composite* if it consists of t separate prover algorithms $(P'_1, \ldots, P'_t)$ such that in the i^{th} repetition of Π, P' simply executes P'_i independently of whatever happened in all the other repetitions so far.

Claim 1.2. *For any prover strategy P'^* for Π' and any input x, there is a composite prover strategy $\hat{P}'^*$ such that:*

$$\Pr\left[\langle \hat{P}'^*, V' \rangle(x) \text{ accepts}\right] \geq \Pr\left[\langle P'^*, V' \rangle(x) \text{ accepts}\right]$$

Proof. Given P'^*, we construct $\hat{P}'^* = (\hat{P}_1'^*, \ldots, \hat{P}_t'^*)$ in much the same way as we constructed P^* in the proof of Theorem 1.2. $\hat{P}_i'^*$ finds a transcript for the interaction between P'^* and V' in the first $(i-1)$ repetitions of Π that maximises the probability that the i^{th} repetition accepts, and then it resumes executing P'^* in the i^{th} iteration when actually interacting with the verifier V.

It is easy to see that for each i, the probability that the i^{th} repetition accepts with $\hat{P}_i'^*$ as the prover is at least as much as when P'^* is the prover. Thus, probability that a majority of the repetitions accept is at least as much with $\hat{P}'^*$ as with P'^*. □

Thus, without loss of generality, we may assume that a cheating prover P'^* for Π^* is composite. For any $x \notin L$, then, the events of the different repetitions of Π accepting are independent, as both the verifier and the prover's actions in each repetition is independent of that in other repetitions. Letting Y_i be a random variable indicating the i^{th} repetition accepting, and $Y = \sum_{i=1}^{t} Y_i$, note that due to the soundness of Π, we have $\mathrm{E}[Y] \leq t/3$. Thus, the probability that Π' accepts may be bounded as:

$$\Pr[Y \geq t/2] = \Pr[(Y - t/3) \geq t/6] \leq \Pr[(Y - \mathrm{E}[Y]) \geq t/6] \leq e^{-\Omega(t)}$$

□

A few other facts about errors that we will visit later:

1. Any language that has an IP also has an IP that is perfectly complete.
2. Only languages in NP have IP's that are perfectly sound.
3. Repeating an IP in parallel (as opposed to sequentially as above) also amplifies the completeness and soundness, and has the added benefit of not increasing the number of rounds in the protocol.

2 Randomness

Recall that in our modelling of an interactive proof, the verifier has a private source of randomness that is not visible to the prover. In the protocol for Graph Non-Isomophism that we saw, the fact that this randomness was private was critical – the prover's task in the protocol was to guess a specific bit in this random string. In general, however, it turns out that, while randomness is indeed necessary for non-trivial interactive proofs, the privacy of the verifier's randomness is not essential. Consider the following version of interactive proofs that has no private randomness.

Definition 2.1. An interactive protocol (P, V) is said to be *public-coin* if, in each round of communication, the verifier's action consists solely of sampling a number of uniformly random bits and sending them to the prover.

Note, in particular, that the sumcheck protocol satisfies the above conditions, and is thus a public-coin protocol. Given this, it is perhaps not surprising that such protocols are as powerful as private-coin protocols. This fact, however, was shown by Goldwasser and Sipser [GS86] years before the sumcheck protocol was discovered.

Theorem 2.1 ([GS86])**.** *Any language that has an IP with k messages also has a public-coin IP with $k+2$ messages.*

The proof of this statement employs a public-coin protocol for the task of lower-bounding the size of a set. This protocol is interesting on its own and involves some very useful ideas. We first look at this protocol, and then use it to prove a weaker version of the above theorem.

2.1 Set Lower Bounds

The input to the set lower-bound protocol is a set $S \subseteq \{0,1\}^m$ for some m, and a number $t \leq m$. This set may be arbitrarily large, and is specified by means of a "membership oracle" $M_S : \{0,1\}^m \to \{0,1\}$ that is such that $M_S(x) = 1$ if $x \in S$, and $M_S(x) = 0$ otherwise. The objective is for the prover to prove to the verifier that the set S is larger than 2^t. A hard cutoff at this size, however, is not really possible to achieve, and the protocol we will see has the following properties:

- **Completeness:** If $|S| \geq 2^k$, the verifier accepts with high probability.
- **Soundness:** If $|S| \leq 2^k/10$, the verifier rejects with high probability.

The gap of a factor of 10 above is somewhat arbitrary, and the protocol can be modified so that soundness holds when $|S| \leq (1-\delta)2^k$ for any constant $\delta > 0$, and even for some values $\delta = o(1)$.

The idea behind the lower-bound protocol is quite simple. The verifier randomly partitions the space $\{0,1\}^m$ into smaller sets (whose size is determined by t), specifies one of the partitions, and asks the prover to find an element of S that is contained in that partition. If the set S is large, then most partitions will contain at least one of its elements in it, and if it is small, most partitions will not.

2.1.1 Protocol using random functions

The simplest way to implement this intuition is the following protocol whose input is oracle access to M_S and a $t \leq m$:

1. V samples a random function $h : \{0,1\}^m \to \{0,1\}^t$, and sends it to P.
2. P finds an $x \in S$ such that $h(x) = 0$, and sends it to V.
3. V checks that $h(x) = 0$ and $M_S(x) = 1$, accepts if so, and rejects otherwise.

Of course, sampling a random function and writing down its description is not an efficient task, and the verifier in the above protocol is not going to run in polynomial time. But this we will fix later, and focus for now on ascertaining its properties.

Soundness. Suppose $|S| \leq 2^t/10$. Then, the probability that, for a random function h, there exists an $x \in S$ such that $h(x) = 0$ may be bounded as follows:

$$\Pr_h\left[\exists x \in S : h(x) = 0\right] \leq \sum_{x \in S} \Pr_h\left[h(x) = 0\right] = |S| \cdot \frac{1}{2^t} \leq \frac{1}{10}$$

where the first inequality is the union bound, the equality uses the fact that the value of $h(x)$ is equally likely to be anything in $\{0,1\}^t$, and the second inequality follows from the bound on the size of S.

Completeness. Suppose $|S| = 2^t$ (if S is larger, the probability of accepting only increases). Using the inclusion-exclusion, principle, the probability that we are interested in may be lower-bounded as follows:

$$\begin{aligned}
\Pr_h\left[\exists x \in S : h(x) = 0\right] &\geq \sum_{x \in S} \Pr_h\left[h(x) = 0\right] - \sum_{x < x' \in S} \Pr_h\left[h(x) = 0 \wedge h(x') = 0\right] \\
&= \sum_{x \in S} \Pr_h\left[h(x) = 0\right] - \sum_{x < x' \in S} \Pr_h\left[h(x) = 0\right] \cdot \Pr_h\left[h(x') = 0\right] \\
&= |S| \cdot \frac{1}{2^t} - \binom{|S|}{2} \left(\frac{1}{2^t}\right)^2 \\
&\geq \frac{|S|}{2^t} - \frac{1}{2} \cdot \left(\frac{|S|}{2^t}\right)^2 \\
&= \frac{1}{2}
\end{aligned}$$

where the first equality follows from the fact that the events $h(x) = 0$ and $h(x') = 0$ are independent if h is a uniformly random function. Thus, $\Pr_h\left[h(x) = 0 \wedge h(x') = 0\right] = \Pr_h\left[h(x) = 0\right] \cdot \Pr_h\left[h(x') = 0\right]$.

Thus, there is a gap between the probability of acceptance when $|S|$ is more than 2^t and when it is less than $2^t/10$. This gap can then be amplified by the techniques from the previous section.

2.1.2 Derandomising the protocol

Note that in the proof of completeness and soundness above, we only used the following two properties of the random function h:

1. For any $x \in \{0,1\}^m$, the probability that $h(x) = 0$ is $1/2^t$
2. For any distinct $x, x' \in \{0,1\}^m$, the events $h(x) = 0$ and $h(x') = 0$ are independent

This suggests that h need not be a completely random function – it just needs to be from a set of functions such that a random function from that set satisfies the above two properties. Then, the proofs above will work as is. Further, if such a set consists of functions that are efficiently samplable, then the above protocol can also be implemented efficiently.

For this purpose, we will make use of functions that satisfy a slightly stronger property called *pairwise independence.*

Definition 2.2. A family of functions $H = \{h : X \to Y\}$ is said to be pairwise independent if, for any distinct $x_1, x_2 \in X$ and any $y_1, y_2 \in Y$,

$$\Pr_{h \leftarrow H}\left[h(x_1) = y_1 \wedge h(x_2) = y_2\right] = \frac{1}{|Y|^2}$$

In other words, when the output of these functions on any two fixed input is completely random, though there may be correlations when three or more outputs are considered. Often in our applications, the set Y will be much smaller than X, and we refer to these families as pairwise independent hash families or hash functions. It may be verified that any pairwise independent family of functions satisfies the two properties above.

Further, there are very simple and efficiently samplable pairwise independent families. For instance, consider the family of functions $H = \{h_{A,b}\{0,1\}^m \to \{0,1\}^t\}$ where each function is indexed by a matrix $A\{0,1\}^{t\times m}$ and a vector $b \in \{0,1\}^t$. On input an $x \in \{0,1\}^m$, such a function is computed as $h_{A,b} = A \cdot x + b$, where computations are over $\mathbb{F}_2$.

Exercise 2. *Prove that the above family of functions is pairwise independent.*

With this family of functions, we can derandomise the earlier protocol to be as follows:

1. V samples a random matrix $A \leftarrow \{0,1\}^{t\times m}$ and vector $b \leftarrow \{0,1\}^t$, and sends them to P.
2. P finds an $x \in S$ such that $A \cdot x + b = 0$, and sends it to V.
3. V checks that $A \cdot x + b = 0$ and $M_S(x) = 1$, accepts if so, and rejects otherwise.

The verifier in this protocol is clearly efficient, and the completeness and soundness follow from the same arguments as earlier.

2.2 Public-Coin IP for any IP

With the set lower-bound protocol, we can now show how to transform any IP into a public-coin IP. We do this for 2-message IPs here, and recommend reading the original paper by Goldwasser and Sipser [GS86] for the transformation for all IPs.

Any 2-message IP for some language L has the following form on input x:

1. V starts with a private random string $r \leftarrow \{0,1\}^\ell$
2. V computes the first message $\alpha \leftarrow V(x; r)$ and sends it to P, where $\alpha \in \{0,1\}^a$
3. P responds with message $\beta \in \{0,1\}^b$
4. V computes $V(x, \alpha, \beta; r)$, which outputs accept or reject

In order to simplify our discussions, we make the following assumptions regarding the above IP:

- It is perfectly complete
- It has soundness error less than 1/100
- Each possible verifier message $\alpha \in \{0,1\}^a$ is equally likely

In the public-coin protocol, the prover P' will try to prove to the verifier V' a lower bound on the number of strings r such that P can make V accept when V uses r as its random string. Indeed, when $x \in L$, all r's in $\{0,1\}^\ell$ satisfy this property, whereas when $x \notin L$, only 1/100 of them do. Note, however, that in order to employ our set lower-bound protocol, we need a membership oracle for such r's. This oracle is difficult to implement because in the protocol (P, V), r is private to V, and in the set lower-bound protocol, P' would need to know r.

Instead, we lower-bound the number of r's indirectly by bounding the number of messages α such that, if V's message was α, then prover can make it accept with high probability. For any $\alpha \in \{0,1\}^a$ and $\beta \in \{0,1\}^b$, define the following two sets:

$$S_{\alpha,\beta} = \{r \mid V(x, \alpha, \beta; r) \text{ accepts}\}$$
$$S = \left\{\alpha \mid \exists \beta : |S_{\alpha,\beta}| \geq 2^{\ell-a}\right\}$$

Note that, under our assumption that each α is equally likely, there are exactly $2^{\ell-a}$ values of r such that $V(x; r) = \alpha$. Due to perfect completeness and soundness error of less than 1/100, we have the following observations:

- If $x \in L$, for any α, there is a β such that $S_{\alpha,\beta}$ has size $2^{\ell-a}$.
- If $x \notin L$, the number of α's for which there is a β such that $S_{\alpha,\beta} \geq 2^{\ell-a}/10$ is at most $2^a/10$.

Thus, if $x \in L$, the set S contains all possible α's, and is of size 2^a. If $x \notin L$, the set S is of size at most $2^a/10$. In fact, in the latter case, it is much smaller (at most $2^a/100$), but the gap in the requirements on the size of $S_{\alpha,\beta}$ will be useful later.

The idea now is to do a set lower-bound protocol to prove that the size of S is at least 2^a. This needs a membership oracle that says whether a given α is contained in S. But note that this question again can be framed as a set lower-bound – an α is in S if there is a β such that the set $S_{\alpha,\beta}$ is large. Thus, given an α, P' and V' can run another set lower-bound protocol to check whether α is contained in S.

The entire protocol now works as follows:

1. V' samples a random hash function h and sends it to P'
2. P' finds an $\alpha \in S$ such that $h(\alpha) = 0$, and a β such that $|S_{\alpha,\beta}| = 2^{\ell-a}$, and sends α and β to V'
3. V' checks that $h(\alpha) = 0$
4. V' sampels a random hash function h' and sends it to P'
5. P' find an $r \in S_{\alpha,\beta}$ such that $h'(r) = 0$, and sends it to V'
6. V' checks that $h'(r) = 0$ and $V(x, \alpha, \beta; r)$ accepts. If so, it accepts, else it rejects.

Exercise 3. *Prove that if (P, V) is an IP for L satisfying the assumptions at the beginning of Section 2.2, then the above (P', V') is a public-coin IP for L. (You might need to amplify the set lower-bound subprotocols first to make this work.)*

Note that, in general, this transformation of IPs to public-coin IPs does not preserve the complexity of the prover. That is, even if P had originally been quite efficient or of low complexity (say, running in quasipolynomial time or at a lower level of the polynomial hierarchy), P' may not be, as finding the α's and r's that it needs may be hard.

3 Rounds

The number of rounds – that is, the number of messages exchanged – in an interactive proof is another resource that is useful to keep track of. It is often useful to separate protocols into those that have $O(1)$ rounds, and those that have $\omega(1)$ rounds. The round complexity of various protocols will be a recurrent subject later in the course. For now, we will look at a few properties of public-coin protocols with $O(1)$ rounds.

Definition 3.1. For any constant k, an $\mathsf{AM}[k]$ protocol is a public-coin interactive protocol that has k messages where the verifier sends the first message. Further, at the end of the protocol, the verifier makes its decision to accept or reject deterministically based on the transcript. An $\mathsf{MA}[k]$ protocol is the same, except the prover sends the first message.

We use the notation AM to mean $\mathsf{AM}[2]$, and MA to mean $\mathsf{MA}[2]$ – recall that this is the class we encountered in the first lecture. When a public-coin protocol has a constant number of rounds, it turns out that the constant does not really matter. This was shown by Babai and Moran [BM88] early on in the history of interactive proofs.

Theorem 3.1 ([BM88])**.** *For any constant k, $\mathsf{AM}[k] = \mathsf{AM}[2]$.*

Put together with Theorem 2.1, this implies that any language that has an IP with constant rounds also has a *public-coin* IP with two rounds. We will show here a weaker version of Theorem 3.1 that has almost all of the components needed to prove the whole theorem.

Theorem 3.2. $\mathsf{MA} \subseteq \mathsf{AM}$

Proof Sketch. Suppose a language L has an MA protocol (P, V) where, given input x, the prover sends a proof $\beta \in \{0,1\}^b$, and the verifier samples a random string $r \leftarrow \{0,1\}^\ell$, and verifies it as $V(x, \beta; r)$. The following (P', V') is an AM protocol for L:

1. V' picks $r_1, \dots, r_{100b} \leftarrow \{0,1\}^\ell$, and sends them to P'
2. P' finds a $\beta \in \{0,1\}^b$ such that for a majority of the r_i's, $V(x, \beta; r)$ accepts, and sends β to V'
3. V' checks that for a majority of the r_i's, $V(x, \beta; r)$ accepts

Completeness and soundness are easy to prove using Chernoff bounds and the union bound. $\square$

Exercise 4. *Show how the above transformation can be used to prove Theorem 3.1. What happens to the complexity of the prover in this transformation?*

Note that this transformation increased the communication and computational complexity of the protocol by a factor of $\Omega(b)$. This is why it can only be applied to constant-round protocols – after $\omega(1)$ applications, the complexity of the protocol would become superpolynomial. Later in the course, we will see other ways to decrease the number of rounds in an interactive proof that make use of computational assumptions, but work for a broader class of proofs.

References

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68. ACM, 1986.