Lecture 9

Interactive Proofs and Arthur-Merlin Games

April 27, 2004
Lecturer: Paul Beame
Notes: Chris Ré

9.1 Background and Motivation

In the 1980's two notions interactive computation were developed. One, due to Babai, originated in generalizations of NP to allow more powerful verifiers that include probabilistic verification. The other, due to Goldwasser, Micali, and Rackoff, originated in cryptography and was a means to the end of defining zero-knowledge proofs, protocols that allow a party in a cryptographic protocol to convince another party of some property without revealing additional information. (In this course we will not discuss the zero-knowledge aspects, however.) Today's lecture will focus on showing the relationship between these two definitions of interactive computation. (Combined, these two papers won the Gödel Prize.) The definitions here began research on the path to the PCP theorem.

9.2 Interactive Proof Preliminaries

We can view a typical NP algorithm as an interaction between an all-powerful *prover* P and a deterministic polynomial-time bounded *verifier* V. On a shared input x that the verifier wishes to prove is in L, the prover produces y, a certificate depending on x, sends the certificate y to the verifier and the verifier accepts if and only if $V(x, y) = 1$. This can be thought of as one round of interaction. The exact power of the prover is not important here but everything still must be verifiable by the deterministic polynomial time machine. So perhaps the amount of interaction is important? What if we were to allow more rounds of interaction as in the following figure?

Formally, we say that a *round* of an interactive protocol corresponds an uninterrupted sequence of communications of a single party. In this picture $y_1, y_2, \ldots, y_\ell$ denote the messages sent by the prover and $z_1, z_2, \ldots, z_{\ell-1}$ denote the messages sent in response by the verifier. We can formally define the verifier as an algorithm V and the actions of the verifier are given by by $z_1 = V(x_1, y_1)$, ..., $z_\ell = V(x, y_1, z_1, y_2, z_2, \ldots, y_\ell)$. The prover's actions can be defined similarly. The (still deterministic) polynomial time verifer accepts iff $V(x, y_1, z_1, y_2, z_2, \ldots, y_\ell) = 1$ at the end of the computation. (This is $2\ell - 1$ round computation.)

Definition 9.1. Given a Prover P and a Verifier V let $(P, V)(x)$ denote the output of the verifier on input x when the protocol (P, V) is executed.

Definition 9.2. (P, V) is an interactive proof for L if and only if

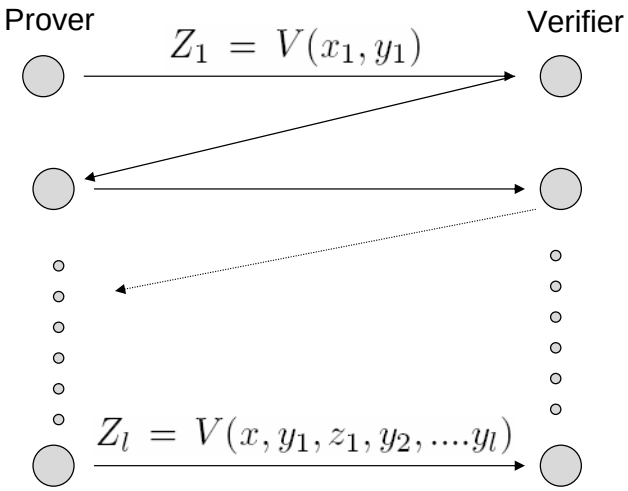

Figure 9.1: Prover Verifier Interaction

$$\begin{cases} (P,V)(x) = 1 & \text{for all } x \in L \\ (P^*,V)(x) = 1 & \text{for all } x \notin L \text{ for all provers } P^*. \end{cases}$$

We claim this is the same power as NP.

Lemma 9.1. *Interactive proofs with deterministic polynomial-time verifiers yields proofs for precisely the languages in* NP.

Proof. To see this, notice that for a multi-round proof the prover can determine in advance what the verifier will say at each response and simply send the entire sequence $(y_1, z_1, \ldots, y_\ell)$ to the verifier which yields a single round of interaction and thus NP. □

In each of the protocols it sufficed for the prover to have the power of NP in order to execute the protocol. However, our limits on the power of the verifier were what restricted the set of languages for which we had interactive proofs.

9.3 Interactive Proofs

Straight interaction with a deterministic verifier did not buy us any extra power. So instead we allow the Verifier to have access to a private random string r. Thus we can define a protocol pair $(P, V(r)$ and its actions on an input x as before. From now on we will assume that the verifier $V(r)$ runs in polynomial time as a function of the length of its inputs and that all messages are polynomial-length as a function of $|x|$.

Definition 9.3. A Prover P and a randomized verifier V with access to a random string r accepts a language L if and only if for some $\epsilon < 1/2$,

$$\begin{cases} \Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[(P, V(r))(x) = 1] > 1 - \epsilon & \text{for } x \in L \\ \forall P^*.\ \Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[(P^*, V(r))(x) = 1] \leq \epsilon & \text{for } x \notin L. \end{cases}$$

Notice that this is a BPP-like acceptance. It is common to say that the verifier is convinced if it accepts the interactive computation. A parameter we will be concerned with is the number of rounds. Notice that these are not round-trips but sending in one direction. This method of counting rounds corresponds nicely to alternation in the polynomial hierarchy. We can now define a complexity class of these interactions.

Definition 9.4. Define $\mathsf{IP}[k(n)]$ to be the set of languages L such that given an $x \in L$ there is a (polynomial-time randomized) protocol that takes at most $k(n)$ rounds to convince the verifier of x's membership in L.

Definition 9.5. $\mathsf{IP} = IP[n^{O(1)}] = IP[Poly(n)]$.

The following is an easy exercise.

Exercise 9.1. $\mathsf{IP} \subseteq \mathsf{PSPACE}$.

We will later prove that the reverse inclusion also holds, i.e. $\mathsf{IP} = \mathsf{PSPACE}$.

9.3.1 Graph Non-isomorphism $\in \mathsf{IP}[2]$

Definition 9.6. GRAPH-NON-ISOMORPHISM =
$\{\langle G_0, G_1\rangle \mid G_0, G_1$ are encodings of graphs and $\forall \sigma \in S_{|V(G_0)|},\ \sigma(G_0) \neq G_1\}$.

Notice the problem GRAPH-ISOMORPHISM of graph isomorphism is in NP since we can guess which permutation to use. This problem will be particularly interesting. Later we will show that if GRAPH-ISOMORPHISM were NP-complete then the polynomial-time hierarchy collapses.

Protocol

We will now give a 2 round protocol to decide GRAPH-NON-ISOMORPHISM. Both prover and verifier have access to G_0 and G_1.

$V \to P$: Verifier chooses $c \in_R \{0,1\}$, chooses $\sigma \in_R S_n$ where $n = |V(G_0)|$.
Verifier sends $\sigma(G_c)$ to the prover.

$P \to V$: The prover, with all possible computation power, determines to which of the two graphs this one is supposed to be isomorphic, say G_b for $b \in \{0,1\}$. (If the input graphs are isomorphic to each other the prover can choose b randomly.) The prover send b to the verifer.

Verification: The Verifier accepts iff $b = c$.

Discussion of Protocol

The important thing to notice is that if $\langle G_0, G_1\rangle$ is not in L, that is they are isomorphic graphs then the prover has exactly a 50-50 chance of guessing which value the verifier chose for c. If $\langle G_0, G_1\rangle \in L$ then the prover should never get it wrong.

This acceptance condition yields a probability only of $1/2$. We can massage this into the correct form that we can execute many copies in parallel. That is in the first round the verifier ships $G^1, .., G^k$ to the prover where each G^i is an independent uniformly random selection according to the first round of the

original protocol. The verifier accepts only if the prover answers correctly for each of the graphs. This does not increase the number of rounds but the prover only fools the verifier with probability $\frac{1}{2^k}$. Notice that it is crucial that the verifier not have access to the coin flips.

BPP and RP have *Amplification Lemmas* and these essentially carry over with us now using parallel independent invocations of the verifier as an analog of multiple independent copies of the machine.

Lemma 9.2 (IP Amplification Lemma). *Let $p(n)$ be a polynomial. Let V be a Verifier which on inputs of length n, a total of $g(n)$ messages each of length $m(n)$, using $\ell(n)$ random bits and error probability at most $1/2 - \delta(n)$. Then there is a V' such that $L(V) = L(V')$, using at most $g(n)$ messages, each of length $O(p(n)m(n)/\delta^2(n))$ using $O(p(n)\ell(n)/\delta^2(n))$ random bits and with an error probability at most $2^{-p(n)}$.*

Proof. (P', V') perform $O(p(n)/\delta^2(n))$ independent parallel simulations of (P, V) and V' takes the majority of the answers. Clearly this blows up the message by the corresponding orders of magnitude and the number of rounds is unchanged. The calculation for the probability of success is identitical that of Lemma 3.3 □

9.4 Arthur-Merlin Games

In this variant of interactive protocols at any point the in the protocol the Prover (now the all-powerful wizard Merlin) is allowed access to all the random coins used by the Verifier (now the not-so smart king Arthur) so far. With this definition it suffices to assume that the strings that are passed back and forth now simply contain the random guesses used by Arthur since it is clear that Merlin, who is all powerful, could easily have computed whatever string Arthur would have computed based on these random guesses.

The key is that Merlin is unaware of the outcomes of the coin before the Arthur sends them to him Notice that Merlin is powerful enough to simulate all possible flips of the coin ahead of time and therefore can play optimally. The game ends with a polynomial time deterministic verification based on the input and the message exchanged. We now use r_i to denote the i-th random string that Arthur sent. The acceptance condition is BPP-like as with ordinary interactive proofs. More precisely, there is a deterministic polynomial time verifier A and a prover M such that if $(r_1, y_1, \ldots, y_\ell)$ is the sequence of communicated values on input x, where each r_i is chosen uniformly at random, then $\Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[A(x, z_1, r_1, y_1, ..., y_k) = 1] > 1 - \epsilon$ if $x \in L$ and for all choices of $(y_1^*, \ldots, y_k^*)$, $\Pr_{r \in \{0,1\}^{|x|^{O(1)}}}[A(x, z_1, r_1, y_1, ..., y_k) = 1] < \epsilon$ if $x \notin L$. We still count rounds in the half round trips and either party is allowed to go first.

9.4.1 Arthur-Merlin Complexity Classes

We denote the complexity class of languages accepted by Arthur-Merlin protocols by an alternating sequence of letters A and M where the number of letters equal to the number of rounds. For example, AMA = { L | there is a 3 round Arthur-Merlin protocol for L in which Arthur starts}.

Notice that ending at an A is like having a BPP machine to do the verification. The class M is NP and the class A is BPP.

Consider the following view of AM:

$L \in$ AM if and only if there is a polynomial-time verifier V, and a polynomial p such that $\forall x$,

$$\Pr[\ \text{Я}^{p(|x|)} r \exists^{p(|x|)} y V(x, y, r) = 1] \text{ is } \begin{cases} > 1 - \epsilon & \text{if } x \in L \\ \leq \epsilon & \text{if } x \notin L \end{cases}$$

The first quantifier acts like a BPP machine that makes one call to an NP oracle accepting its answer; so $\mathsf{AM} = \mathsf{BP} \cdot \mathsf{NP}$. The *value* of the Arthur-Merlin game is defined to be the probability that Arthur (the verifier) is convinced given optimal play by Merlin. We can express the value of the Arthur-Merlin game on input x using a form of quantifier, namely A as an Averaging quantifier and M as a Maximization quantifier. The value of the above AM protocol is then $A^{p(|x|)}r\ M^{p(|x|)}y.\ V(x, y, r)$. This use of A and M as the quantifiers is the source of the names Arthur and Merlin.

Now consider MA. In this case after doing all the nondeterministic work, we get access to a BPP machine, so we get $\mathsf{MA} = \mathsf{N} \cdot \mathsf{BPP}$. In quantifiers with a similar verifier the game would have a value $M^{p(|x|)}r\ A^{p(x)}y.\ V(x, y, r)$.

Definition 9.7. Define
$\mathsf{AM}[k] = \{L \mid L$ has a k round Arthur-Merlin game with Arthur starting$\}$ and
$\mathsf{MA}[k] = \{L \mid L$ has a k round Arthur-Merlin game with Merlin starting$\}$.

Today we will describe the surprising result that Arthur-Merlin games and interactive proofs are essentially equivalent in power.

Theorem 9.3 (Goldwasser, Sipser). *IP[t(n)] $\subseteq$ AM[t(n)+2]*

Next class we will prove the following Theorem.

Theorem 9.4 (Babai, Babai-Moran). *For constant $k \geq 2$, $\mathsf{AM} = \mathsf{AM}[k] = \mathsf{MA}[k+1]$, moreover, for any $t(n)$. AM[2t(n)] $\subseteq$ AM[t(n)+1].*

From these two theorems and the protocol for GRAPH-NON-ISOMORPHISM we derive.

Corollary 9.5. *GRAPH-NON-ISOMORPHISM* $\in \mathsf{AM}$

This is a bit surprising since our protocol relied so heavily on the secret coins which have disappeared in the Arthur-Merlin model.

Proof of Theorem 9.3. Suppose there is an interactive proof (P, V) for L. Merlin will convince Arthur that P *would have convinced* V for a large fraction of random strings r. The key to doing this will be to derive a short protocol that will convince Arthur that certain sets are large. This will be done using universal hashing.

Definition 9.8. A *(pairwise-independent) universal hash function family* is a family of functions, H, from a set U to a set V satisfying the following properties:

- For all $h \in H$, $h : U \to V$.
- For all $x \in U$ and for all $y \in V$, $\Pr_{h \in_R H}[h(x) = y] = \frac{1}{|V|}$.
- For all $x_1, x_2 \in U$ with $x_1 \neq x_2$ and for all $y_1, y_2 \in V$, $\Pr_{h \in_R H}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = \frac{1}{|V|^2}$.

Notice that the third property is a pairwise independence property, saying that knowing where one item is hashed does not give any information about where another item is hashed. The following is an example of such a family of hash functions.

Example 9.1. Let $U = \{0,1\}^n = \mathbb{F}_2^n$, $V = \{0,1\}^m = \mathbb{F}_2^m$. Choose an $n \times m$ matrix $A \in \{0,1\}^{n\times m} = \mathbb{F}_2^{n\times m}$, and a vector $v \in \{0,1\}^m = \mathbb{F}_2^m$. Then define $h_{A,v}(x) = Ax + v$ over $\mathbb{F}_2$. Let H be the set of all such functions. Note the similarity to the functions used in the proof of the Valiant-Vazirani lemma. The addition of the random vector v allows us to get around the fact that linear transformations always map the vector $0 \in V$ to $0 \in V$ and ensure that for h chosen at random, the output is a random vector in V and thus the second condition holds. The third condition holds by the same reasoning as in the proof of the Valiant-Vazirani lemma.

The following lemma gives the basis for the set size lower bound protocol.

Lemma 9.6. *Let $S \subseteq U = \{0,1\}^n$, V and a universal hash function family H from U to V. Select $t = 2n$ hash functions $h_1, \ldots, h_t \in_R H$ and $s = 3n$ points $r_1, \ldots, r_s \in_R V$.*

1. *If $|S| \geq |V|/2$ then $\Pr[\exists i,\ j$ such that $r_j \in h_i(S)] \geq 1 - 2^{-n}$.*
2. *If $|S| < |V|/d$ then $\Pr[\exists i,\ j$ such that $r_j \in h_i(S)] < 6n^2/d$.*

Proof. Fix $r \in V$ and $i \in \{1, \ldots, t\}$. For any $z \in U$, $\Pr_{h\in_R H}[h(z) = r] = \frac{1}{|V|}$ by definition of H. If $|S| \geq |V|/2$, let S' consist of the first $|V|/2$ elements of S. By inclusion-exclusion, for $h \in_R H$,

$$\begin{aligned}
\Pr[\exists z \in S.\ h(z) = r] &\geq \Pr[\exists z \in S'.\ h(z) = r] \\
&\geq \sum_{z\in S'} \Pr[h(z) = r] - \sum_{z'\neq z\in S'} \Pr[h(z) = r \text{ and } h(z') = r] \\
&= \left(\frac{|S'|}{|V|} - \frac{|S'|(|S'|-1)}{2}\right) \cdot \frac{1}{|V|^2} \\
&\geq 1/2 - 1/8 = 3/8
\end{aligned}$$

This implies that $E[|h(S')|] = \sum_{r\in V} \Pr[r \in h(S')] \geq 3|V|/8$. Now consider the probability that $\Pr_{h\in_R H}[|h(S')|] \geq |V|/4$? Using the fact that for any h, $|h(S')| \leq |S'| \leq |V|/2$, a standard Markov's inequality argument shows that this probability is at least $1/2$: Suppose the probability is $< 1/2$. Then $E[|h(S')|] < \frac{1}{2} \cdot |V|/2 + \frac{1}{2}|V|/4 = 3|V|/8$, contradicting our lower bound on $E[|h(S')|]$.

Therefore if we choose $h_1, \ldots, h_{n+1} \in_R H$,

$$\Pr_{h_1,\ldots h_{n+1}\in_R H}[\exists i.\ |h_i(S')| \geq |V|/4] \geq 1 - 2^{-2n}.$$

Suppose now that this holds. Thus the probability that every $r_j \in_R V$ is $\notin \bigcup_i h_i(S')$ is at most $3/4$ and the choices are independent. Therefore the probability that none of the r_j is in $\bigcup_i h_i(S)$ is at most $(3/4)^s < 2^{-n-1}$ for $s = 3n$. Thus the total failure probability is at most 2^{-n}.

Now suppose that $|S| \leq |V|/d$. Then $|\bigcup_{i=1} t h_i(S)| \leq t|S|$ and thus

$$\Pr[\exists i,\ j,\ \text{such that } r_j \in h_i(S)] \leq st\frac{|S|}{|V|} \leq 6n^2/d$$

as required. □

Thus, for Merlin to prove to Arthur that a set $S \in \{0,1\}^n$ is large relative to some bound 2^b, we have the following protocol:

Merlin sends to Arthur The value of the bound b.

Arthur sends to Merlin A sequence of random bits interpreted as $h_1, \ldots, h_{2n}$ independently chosen hash functions from $\{0,1\}^n$ to $V = \{0,1\}^b$ and $r_1, \ldots, r_{3n}$ independently chosen values from $V = \{0,1\}^b$.

Merlin send to Arthur A witness $z \in S$ such that $h_i(z) = r_j$ for some i and j.

If Merlin is unable to produce such a witness then the protocol fails. If $|S| \geq 2^b$ then Merlin is able to succeed with probability greater than $1 - 2^{-n}$; if $|S|$ is much smaller than 2^b then the protocol will only have a polynomially small success probability. It is important that Arthur can play this protocol even if the set S is implicit or known only to Merlin. The only difference is that if the set S is too small then Merlin is likely forced to produce a witness $z \in \{0,1\}^n - S$. Later Arthur may catch Merlin because of this. We will use this property in sequel.

Note. Note that we could have refined the above argument somewhat by taking advantage of the fact that all we need is $\bigcup_i h_i(S)$ to be large rather than any individual $h_i(S)$. A small modification of the above argument will show that

$$E[|h_i(S') - \bigcup_{i'<i} h_{i'}(S')|] \geq 3|V - \bigcup_{i'<i} h_{i'}(S')|/8.$$

Since the maximum value this quantity can take is at most $|V - \bigcup_{i'<i} h_{i'}(S')|$, the probability that it is at least $1/4$ of this maximum value is at least $1/6$ and these events are mutually independent. Thus a constant fraction of the time each additional hash function reduces the size of the uncovered portion of V by a constant factor. By choosing $O(n)$ independent hash functions (or even $O(b)$ if a failure probability only at most 2^{-b} is required) we can assure almost certainly that every point of V is covered by the image of S' under some hash function. Thus, only a single random query r is required.

Protocol Overview By the amplication lemma we can assume that that $(P, V(r))(x)$ accepts with very high probability if $x \in L$ and any $(P^*, V(r))(x)$ accepts with at most exponentially small probability.

Suppose that the set of all random strings used by the verifier V is chosen from $\{0,1\}^\ell$. Merlin contends that the set of random strings for which V would have been convinced is not very small, say at least $2^{\ell-1}$. Let $A_\epsilon = \{r \mid (P, V(r)) \text{ accepts}\}$ be this set of random strings. We need to decide whether or $|A_\epsilon|$ is nearly 2^ℓ or exponentially smaller than 2^ℓ.

We first will assume that IP protocol is a 2-round protocol of the style that was used for GRAPH-NON-ISOMORPHISM in which the verifier V first sends a string z and then the prover P sends a string y to the verifier who makes a final decision based also the random string.

Let $z = V(x, \epsilon, r)$, the string that V would have been sent to P given the random string $r \in \{0,1\}^\ell$. Associated with each message z that V sends in this round there is an optimal response string $y = y(z)$ that maximizes the probability that $V(x, (z, y), r)$ accepts. (Assume without loss of generality that $z \in \{0,1\}^m$.) Now define the set

$$A_z = \{r \mid z = V(x, \epsilon, r) \text{ and } V(x, (z, y(z)), r) \text{ accepts } \},$$

the set of all random strings r such that the verifier initially sent z, the prover can respond with $y(z)$ and the verifier will accept. Clearly $A_\epsilon = \bigcup_z A_z$ where the union is disjoint.

The goal is to give a protocol for an AM protocol that allows Arthur to be convinced that the Verifier would have accepted this input under the AM acceptance condition. Merlin will do this by proving to Arthur

that with large probability there is a large set of z such that A_z is large. To do this, Merlin will choose a set S_1 of z's each of which has roughly the same size.

The Set S_1 a: Command not found. where we have chosen b_1 to maximize $|\bigcup_{z \in S_1} A_z|$. That is, we put the A_z into buckets based on the order of magnitude of their size; after binning, we choose the bin that contributes the most to A_ϵ. By construction,

- $\sum z \in S_1 |A_z| \geq \frac{|A_\epsilon|}{\ell}$ and
- for each $z \in S_1$, $2^{b_1 - 1} < |A_z| \leq 2^{b_1}$.

Merlin will pick the set S_1 and send the value $s_1 = \lceil \log_2 |S_1| \rceil$ to Arthur and convince Arthur that S_1 is large relative to 2^{s_1}. In doing this using the set size lower bound protocol, Merlin will identify and element $z \in S_1$. He will then prove that A_z is large relative to 2^{b_1}. If $b_1 + s_1$ is large enough, Arthur will accept.

These will be done by applications of the hashing protocol above. As a result, Arthur can be convinced of the truth of the assertion if we show him that the set is at least a constant fraction $\frac{2^\ell}{\ell}$.

The Protocol

- $M \rightarrow A$: (Round 0)

  Merlin computes the set S_1, the value b_1 and sends $s_1 = \lceil \log_2 |S_1| \rceil$ and b_1 to Arthur.

- $A \rightarrow M$: (Round 1)

  Arthur sends 2ℓ random hash functions $h_1, \ldots, h_{2\ell}$ from $\{0,1\}^m$ to $\{0,1\}^{s_1}$ and 3ℓ random challenge strings $r_1, \ldots, r_{3\ell} \in_R \{0,1\}^{s_1}$ to Merlin.

  Arthur sends 2ℓ random hash functions $h'_1, \ldots, h'_{2\ell}$ from $\{0,1\}^\ell$ to $\{0,1\}^{b_1}$ and 3ℓ random challenge strings $r'_1, \ldots, r'_{3\ell} \in_R \{0,1\}^{b_1}$ to Merlin.

- $M \rightarrow A$: (Round 2)

  Merlin produces a string $z \in S_1$ such that $h_i(z) = r_j$ for some i and j if possible. (Otherwise Merlin chooses an arbitrary string z that maps to one of the r_j's if possible.)

  Merlin sends $(z, y(z))$ as well as (i, j) to Arthur.

  Merlin produces a string $r \in A_z$ such that $h'_{i'}(r) = r'_{j'}$ for some i' and j' if possible. (Otherwise, Merlin chooses an arbitrary string r that maps to one of the r_j's if possible.)

  Merlin sends r as well as (i', j') to Arthur.

- Verify: In the final deterministic verification Arthur checks that $h_i(z) = r_j$, that $h'_{i'}(r) = r'_{j'}$, that $V(x, \epsilon, r) = z$, that $V(x, (z, y), r)$ accepts (i.e., that $r \in A_z$), and that $s_1 + b_1 \geq \ell - \log_2 \ell - 1$.

Now let us verify that this yields a good Arthur-Merlin protocol.

If $\Pr_r[(P, V(r))(x) = 1] > 1/2$: In this case, $|A_\epsilon| > 2^{\ell-1}$ and thus

$$\sum_{z \in S_1} |A_z| \geq |A_\epsilon|/\ell > 2^{\ell-1}/\ell \geq 2^{\ell - \log_2 \ell - 1}.$$

Since $|S_1| \geq 2^{s_1-1}$, by Lemma 9.6, Merlin will be able to find the required $z \in S_1$ with failure probability at most $2^{-\ell}$. Then for this z, $|A_z| \geq 2^{b_1-1}$ so again by Lemma 9.6 and the definition of A_z, Merlin will be able to find an r such that h' hashes r correctly, $V(x, \epsilon, r) = z$, and $V(x, (z, y), r)$ accepts. Finally, observe that

$$\sum_{z \in S_1} |A_z| \leq \sum_{z \in S_1} 2^{b_1} = |S_1| \cdot 2^{b_1} \leq 2^{b_1 + s_1}$$

and thus $b_1 + s_1 \geq \ell - \log_2 \ell - 1$.

If $\Pr_r[(P, V(r))(x) = 1] < 1/\ell^7$: In this case $|A_\epsilon| \leq 2^\ell/\ell^7$. Suppose that Merlin tries to cheat Arthur. What is the chance that Arthur will be convinced? In order for Arthur to be convinced Merlin must send b_1 and s_1 in Round 0 such that $b_1 + s_1 \geq \ell - \log_2 \ell - 1$, that is, $2^{b_1+s_1} \geq \frac{2^\ell}{2\ell}$. Choose $d = \ell^3$. Let S_1' be the set of all z such that $|A_z| \geq 2^{b_1}/d$. The size of S_1' is at most $d \cdot |A_\epsilon|/2^{b_1} \leq 2^{\ell - b_1}/\ell^4 \leq 2^{s_1 + \log_2 \ell + 1}/\ell^4 = 2^{s_1+1}/d$. The probability that Merlin could produce an element z of S_1' in response to the hash challenge for S_1 is $O(\ell^2/d)$ which is $O(1/\ell)$. If Merlin does not produce such an element then the probability that Merlin is able to produce an element r for A_z that will convince Arthur is also only $O(1/\ell)$. Therefore, the total probability that Arthur is convinced in this case is $O(1/\ell)$.

This proves correctness in the two round case.

More Rounds In general, for a procotol with more rounds, note that for any prefix of the computation, once r is fixed, whether or not V will accept beginning with that prefix is a deterministic property using the optimal play of Merlin. Therefore once Merlin fixes $(z_1, y_1(z_1))$ for his first response, we can define sets

$$A_{z_1,z} = \{r \mid V(x, \epsilon, r) = z_1 \text{ and } V(x, (z_1, y_1(z_1)), r) = z \text{ and } V(x, (z_1, y_1(z_1), z, y_2(z), \ldots), r) \text{ accepts } \},$$

$$A_{z_1,z_2,z} = \{r \mid s = (z_1, y_1(z_1), z_2, y_2(z_2), z, y_3(z)) \text{ is a valid execution on input } x \text{ and } V(x, (s \ldots), r) \text{ accepts } \},$$

etc. At each round, Merlin will choose an S_i consisting of those z in the highest weight bin for the sets $A_{z_1,\ldots,z_{i-1},z}$. At each round, Merlin will send Arthur $s_i = \lceil \log_2 |S_i| \rceil$ and convince Arthur that $|S_i| \geq 2^{s_i - 1}$ by sending z_i and $y_i(z_i)$. For the last round of the prover's communication, Merlin will send b_k in addition to s_k, and convince Arthur that $|A_{z_1,\ldots,z_k}| \geq 2^{b_k}$ by sending a choice of r. If there are $2k$ rounds beginning with then Arthur then there will be a loss of ℓ^k in the size of $|A_\epsilon|$ at most so Arthur will accept if all the hash conditions have been met, the final random string and the sequence of z_i and y_i will yield acceptance, and $b_k + \sum_{i=1}^k s_i \geq \ell - k \log_2 \ell - 1$. By a similar argument to the 2 round case, if the original acceptance probability is at most $1/\ell^{8k-1}$, then Arthur is convinced only an $O(k/\ell)$ fraction of the time.

Number of Rounds Notice how many rounds it took to prove the claim. In the new protocol we have added one communication from Merlin to Arthur of the set size s_1 (and b_1 in the two round case) at the beginning and one communication at the end for Merlin to send Arthur the random string r in response to the last challenge. If the protocol already began and ended with Arthur then this adds two rounds. If the protocol already began or ended with Merlin (as in the 2 round case) then we add fewer rounds.

□