CS 710: Complexity Theory 12/13/2011

Lecture 27: Interactive Proof Systems

Instructor: Dieter van Melkebeek Scribe: Xi Wu and Prathmesh Prabhu

In this lecture we first discuss bounded-round interactive proof systems and its relationship with other complexity classes. Along the way we prove an interesting result that coNP is not in AM unless the polynomial time hierarchy collapses to its second level. Given GNI has an AM protocol, this gives strong evidence that GI is not NP-complete. As it is unlikely that coNP has bounded round interactive proof, it was open for quite a while whether coNP has IPS at all. Actually complexity theorists come up with a relativized world in which coNP indeed does not have IPS. However, as it turns out and somewhat surprisingly, not only coNP has IPS but also every language in PSPACE does. Given the other direction that IP $\subseteq$ PSPACE, we now know that the power of IP equals the power of PSPACE. We prove both of these results in the final part of this lecture.

1 Bounded-Round Interactive Proof Systems

An interactive proof system is a bounded-round proof system if the total number of rounds of communication between V and P is bounded by a constant for all inputs. Here, a round is completed when both TMs have received some information from the other once.

We saw that an IPS can be public/private-coin depending on whether P has access to the random bits used by V. The following theorem, which we state without proof, shows that all proof systems can indeed be transformed to public proof systems with perfect completeness without blowing up the number of rounds in the proof system by too much.

Theorem 1. *Given an* IPS *for L with r rounds. There exists*

- *An* IPS *for L with r rounds and $c = 1$.*
- *A public coin* IPS *for L with $r + 2$ rounds.*

Public coin bounded-round interactive proof systems with perfect completeness are called Arthur-Merlin proof systems. Thus theorem 1 indicates that any bounded-round interactive proof system can be reduced to an Arthur-Merlin system.

We now define some classes of Arthur-Merlin proof systems and explore their relationship with classical complexity classes. The simplest classes are the ones where there is no interaction between the V (Arthur in the following) and P (Merlin in the following).

Definition 1 (A)**.** A *is the class of languages which have public coin systems where Arthur works on his own. There is no role of Merlin in this class.*

Clearly in this case A coincides with BPP.

Definition 2 (M)**.** M *is the class of languages which have public coin systems where Arthur is not allowed to use randomness. Thus he is only allowed to run a deterministic verifier which runs in time polynomial in the length of the common input on the purported proof sent from Merlin.*

Clearly M coincides with NP. The more interesting Arthur-Merlin games are those with a single round of communication.

Definition 3 (MA)**.** MA *is a public coin system in which Merlin moves first, and then Arthur flips some coins and runs a deterministic predicate on the coins, Merlin's message and the input. Formally, a language* $L \in \mathrm{MA}$ *if there exists* $V \in \mathrm{P}$*:*

$$x \in L \implies (\exists y) \Pr_z[V(x, y, z) = 1] \geq 2/3$$
$$x \notin L \implies (\forall y) \Pr_z[V(x, y, z) = 1] \leq 1/3$$

Definition 4 (AM)**.** AM *is a public coin system in which Arthur moves first. He flips some coins to determine what to ask Merlin. Arthur then runs a deterministic predicate on the coins, Merlin's reply and the input. Formally, a language* $L \in \mathrm{AM}$ *if there exists* $V \in \mathrm{P}$*:*

$$x \in L \implies \Pr_z[(\exists y) V(x, y, z) = 1] \geq 2/3$$
$$x \notin L \implies \Pr_z[(\exists y) V(x, y, z) = 1] \leq 1/3$$

Two remarks. First, there is a clear "asymmetry" between the soundness definition of MA and AM. In MA the quantifier over proofs is $\forall$ while in AM it is $\exists$. While intuitively this is right definition we should use, the syntactic difference still worth to remember. Especially for AM, the definition indicates that AM is like $\mathrm{BP} \cdot \mathrm{NP}$.

Second, MA systems are sometimes referred to as *publishable proof systems*: P publishes a purported proof of the claim and V verifies that the proof is indeed correct. Unlike in NP, in this case, V can use randomness to verify the correctness of the proof.

Lemma 1. $\mathrm{MA} \subseteq \mathrm{AM}$

Proof. Let $L \in \mathrm{MA}$ and consider its MA protocol

$$x \in L \implies (\exists y) \Pr_z[V(x, y, z) = 1] \geq 2/3$$
$$x \notin L \implies (\forall y) \Pr_z[V(x, y, z) = 1] \leq 1/3$$

We want to swap the quantifier over y into the Pr (probabilistic for all) quantifier. Obviously the first is still true, that is, $\Pr_z[(\exists y) V(x, y, z) = 1] \geq 2/3$. For the second we want $x \notin L \implies \Pr_z[(\exists y) V(x, y, z) = 1] \leq 1/3$ Clearly this is stronger than requiring that $(\forall y) \Pr_z[V(x, y, z) = 1] \leq 1/3$. Note that however $\Pr[(\exists y) V(x, y, z)] \leq \sum_y \Pr_z[V(x, y, z) = 1]$, so the idea is to shrink the probability, for any fixed y, $\Pr_z[V(x, y, z)]$ to be so small that even we sum over all y still gives that $\sum_y \Pr_z[V(x, y, z) = 1] \leq 1/3$. Clearly this task can be done by standard error reduction method, as MA protocol guarantee that for any fixed y, $\Pr_z[V(x, y, z) = 1] \leq 1/3$. □

One can envision defining further classes with more rounds of interaction, for instance AMA, MAM and so on. Indeed the class $\mathrm{AM}(k)$ to capture all these. Note that we use AM to mean AM(2), while using IP to denote interactive proofs of arbitrary rounds. This is a little bit inconsistent, due to historical reasons. Now as the last lemma indicates, we can swap MA to AM to reduce the number of rounds.

$$\mathrm{AMA} \subseteq \mathrm{AAM} = \mathrm{AM}$$

Indeed with some additional nontrivial tweak, we have the following theorem (left as an exercise).

Theorem 2. *Let $r(n)$ be a polynomial in n with $r(n) > 2$. Then* $\mathrm{AM}(2r(n)) = \mathrm{AM}(r(n))$.

As an immediate corollary we have the following.

Corollary 1. $\mathrm{AM}(k) = \mathrm{AM}$ *for all constant k.*

An important caveat is that this does not imply that IP collapses to AM. In the reduction of (2) the running time of the prover and verifier is blown up by a polynomial factor. In the case of the corollary, applying the reduction constant number of times still gives polynomial time. However applying the reduction $f(n)$ times where f is any monotone and unbounded function of n blows up the running time of the IPS to super-polynomial. Indeed it is an important open question to find a nice characterization of $\mathrm{AM}(f(n))$ where $f(n)$ is slowly growing, for example $f(n) = \log n$ or $f(n) = \log \log n$.

2 Arthur-Merlin Games and the Polynomial Time Hierarchy

Lemma 2.

$$\mathrm{AM} \subseteq \Pi_2^p$$
$$\mathrm{MA} \subseteq \Sigma_2^p \cap \Pi_2^p$$

Proof. For the first we have noted that AM is like $\mathrm{BP} \cdot \mathrm{NP}$. Actually look at the proof of $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$, the only thing changes is that the inner predicate is an NP predicate. By picking the Π_2^p predicate outside and merging the $\exists$ quantifier, we have that $\mathrm{AM} \subseteq \Pi_2^p$.

For MA the argument is similar, except the $\exists$ quantifier comes on the left. By picking the Σ_2^p predicate in the proof of $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$ and merging the $\exists$ quantifier, we have that $\mathrm{MA} \subseteq \Sigma_2^p$. Finally because $\mathrm{MA} \subseteq \mathrm{AM} \subseteq \Pi_2^p$, it follows that $\mathrm{MA} \subseteq \Sigma_2^p \cap \Pi_2^p$. The proof is complete. □

Lemma 3. $\mathrm{AM} \subseteq \mathrm{NP}/\mathrm{poly}$

Proof. The same argument as in the proof of $\mathrm{BPP} \subseteq \mathrm{P}/\mathrm{poly}$ except now inside the Pr (probabilistic for all) quantifier it is an NP predicate. This gives that $\mathrm{AM} \subseteq \mathrm{NP}/\mathrm{poly}$. □

We showed that under reasonable hardness assumptions, BPP collapses to P. Similarly, AM collapses to NP under reasonable conditions, for example the existence of a language L in $\mathrm{NE} \cap \mathrm{coNE}$ which requires linear-exponential *nondeterministic* circuits. Further, like the BPP case, we have different levels of derandomization under different hardness assumptions. For example, under weaker hardness assumptions, AM collapses to classes between NP and NEXP.

We know that if $\mathrm{P} \neq \mathrm{NP}$, there is a class of NP-Intermediate problems, which are neither in P nor NP-complete. Graph isomorphism (GI) is a natural candidate of such a problem. Interestingly enough, while we have no evidence that GI is not in P, we do have strong evidence that it is not NP-Complete. Namely if it is, then polynomial time hierarchy collapses to its second level.

Theorem 3. GI *is not* NP*-complete unless* $\Sigma_2^p = \Pi_2^p$.

Proof. Suppose that GI is NP-complete. Then, GNI is coNP-complete. Let L be a Σ_2^{P} language. Then for some constant c and a polynomial-time predicate V we have that

$$x \in L \iff (\exists y \in \Sigma^{n^c})(\forall z \in \Sigma^{n^c})[\langle x, y, z\rangle \in V]$$

As GNI is coNP-complete and we know that GNI $\in$ AM, therefore $(\forall z \in \Sigma^{n^c})[< x, y, z > \in V]$ coNP can be reduced to an AM protocol. Now by expressing the existential quantifier as a Merlin phase, we have that

$$\Sigma_2^P \subseteq \text{MAM} \subseteq \text{AM} \subseteq \Pi_2^p$$

where the second inclusion is by theorem 2, and the third inclusion is by lemma 2. Therefore $\text{PH} = \Sigma_2^p$ and the proof is complete. □

Look into the proof we found that any coNP-complete problem has AM protocol will reduce Σ_2^p to MAM and thus AM, therefore the following corollary is immediate.

Corollary 2. coNP $\not\subseteq$ AM *unless* $\Sigma_2^{\text{P}} = \Pi_2^{\text{P}}$

While this indicates that coNP is unlikely to have IPS of bounded rounds, it turns out that they do have IPS of unbounded rounds. The proof system is known in literature as the sumcheck protocol and is shown in the next section.

3 Sumcheck Protocol

For quite some time it was unknown whether coNP has interactive proofs. Actually complexity theorists come up with a relativized world in which coNP indeed does not have IPS. However, as it turns out and somewhat surprisingly, not only coNP has IPS, but also every language in PSPACE does. Along the line to reach this result, an important algebraic method, namely arithmetization, is developed and turns out to be quite useful in computational complexity. Note that by our statement that coNP $\subseteq$ IP while $\text{coNP}^A \not\subseteq \text{IP}^A$ for some oracle A, it implies that the proof of coNP $\subseteq$ IP must not relativize. Indeed this is one of the few major results in computational complexity that does not relativize.

Our next goal is to show that coNP has IPS. Indeed a stronger result is shown: counting the number of satisfying assignment of a Boolean formula has IPS. Formally consider the following language.

$$L = \{(\varphi, \#\,\text{SAT}(\varphi)) \mid \varphi \text{ is a Boolean Formula. } \}$$

Theorem 4. $L \in \text{IP}$.

Proof. Given a pair of a Boolean formula $\varphi = \varphi(x_1, \ldots, x_n)$ and a, the idea is to check the following sum (so the name sumcheck)

$$\sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \varphi(x_1, \ldots, x_n) = a$$

Towards an IPS the first idea is to arithmetize φ to get a low-degree multivariate polynomial $\widetilde{\varphi}$ over $\mathbb{Z}$ which interpolates φ on the Boolean cube. In other words, for every $x \in \{0,1\}^n$, $\widetilde{\varphi}(x) = \varphi(x)$. Given a CNF formula φ with m clauses, the interpolation is easily divided into arithmetizing each clause and then times them together.

$$\widetilde{\varphi} = \prod_C \widetilde{C}$$

Consider a clause for example $x_1 \vee \overline{x_2} \vee x_3$, the arithmetization is $1-(1-x_1)x_2(1-x_3)$. Note in each clause we assume the underlying variables are different, otherwise they can be merged or the clause is trivially satisfied. Note that in this way, each variable in $\widetilde{\varphi}$ has degree at most m.

Now we can work on a multivariate low-degree polynomial and check whether

$$\sum_{x_1\in\{0,1\}}\sum_{x_2\in\{0,1\}}\cdots\sum_{x_n\in\{0,1\}}\widetilde{\varphi}(x_1,\ldots,x_n)=a$$

Working on polynomials clearly has more structure to exploit, at least our domain changes from Boolean domain to $\mathbb{Z}$. Indeed it turns out that this fact plus the fact that univariate nonzero low-degree polynomial has few roots is crucial.

The next idea is natural and critical. If we can reduce this problem to the problem of checking a polynomial of the same type but with one variable less then we are done by at most n rounds of interactions. In each round we hope that there is a small error not catching a cheating prover and finally a union bound is applied. To formalize this idea, we view the LHS as a univariate polynomial of x_1, namely

$$g_1(x_1)=\sum_{x_2\in\{0,1\}}\sum_{x_3\in\{0,1\}}\cdots\sum_{x_n\in\{0,1\}}\widetilde{\varphi}(x_1,\ldots,x_n)$$

Note that we are summing over $x_2,\ldots,x_n$ while x_1 is left as a variable. Arthur want to check whether $g_1(0)+g_1(1)=a$ but g_1 cannot be efficiently computed. So now Arthur asks Merlin to send the description of g_1 to him. Note that, as g_1 is a univariate polynomial, the description is short so Arthur can simply ask for all coefficients of g_1.

Upon receiving the purported description g_1' from Merlin, Arthur does the following

1. It checks whether $g_1'(0)+g_1'(1)=a$. This is natural as the *correct* $g'=g_1$ should at least satisfy this property. If this check fails then certainly Merlin is cheating and Arthur rejects.

2. If the first check passed, then Arthur he checks more about g_1'. For this, Arthur randomly picks a point ξ_1 over a sufficiently large range I, compute $g_1'(\xi_1)=a'$ and reduce the problem to check whether

$$\sum_{x_2\in\{0,1\}}\sum_{x_3\in\{0,1\}}\cdots\sum_{x_n\in\{0,1\}}\widetilde{\varphi}(\xi_1,x_2\ldots,x_n)=a'$$

   Now the crucial observation is that, as we want, this is the same type of checking. Namely write $g_2(x_2)=\sum_{x_3\in\{0,1\}}\cdots\sum_{x_n\in\{0,1\}}\widetilde{\varphi}(\xi_1,x_2,x_3\ldots,x_n)$ as a univariate polynomial in x_2 and ask Merlin to send the description of g_2 and the same procedure repeats.

3. Finally suppose that all variables $x_1,\ldots,x_n$ are fixed as $\xi_1,\ldots,\xi_n$, and we want to check whether $\widetilde{\varphi}(\xi_1,\ldots,\xi_n)$ equals some value κ. The key observation is that now Arthur no longer needs the help of prover as he knows the description of $\widetilde{\varphi}$ and all the variables are fixed! Therefore he directly checks whether this is correct.

Let's analyze this protocol. Clearly if $a=\#\,\mathrm{SAT}(\varphi)$ the an honest prover can send all polynomials required to make the verifier accept. This indicates that the protocol has perfect completeness. Now suppose $a\neq\#\,\mathrm{SAT}(\varphi)$ and Merlin wants to trick. Then in step 1, instead of sending g_1 where $g_1(0)+g_1(1)=\#\,\mathrm{SAT}(\varphi)\neq a$ he must send some $g_1'\neq g_1$ to get that $g_1'(0)+g_1'(1)=a$ in order

to pass the checked posed at step 1. Given $g' \neq g_1$, the probability that $a' = g'(\xi_1) = g_1(\xi_1)$ is bounded by $\frac{\deg(g)}{|I|} = \frac{m}{|I|}$.

On one hand, consider the event that $a' = g'_1(\xi_1) \neq g_1(\xi_1)$, which happens w.p. at least $1 - \frac{m}{|I|}$. Because $g_2(0) + g_2(1) = g_1(\xi_1)$ by definition, therefore Merlin cannot honestly send the description of g_2, otherwise he cannot pass step 1 of the second round. This case is good as Merlin must continue to cheat. On the other hand consider the event that $g'(\xi_1) = g_1(\xi_1)$ which happens w.p. at most $\frac{m}{|I|}$. Then in the second round Merlin can still send the description of g_2. In this case a cheating claim has be transformed into a correct claim and we cannot catch Merlin any more.

There are two analyses. First is the union bound analysis. Note that if cheating claim is never transformed into a correct claim, then we can finally catch Merlin. There are n rounds in total, and therefore

$$\Pr[(\exists i \in [n]) \text{ in round } i \text{ incorrect claim is transformed into a correct claim}] \leq \frac{nm}{|I|}$$

Therefore it suffices to pick $|I| = O(nm)$.

In the second analysis note that the verifier is independent in different round. Further it suffices to lower bound the probability that we catch Merlin if he tricks. Therefore it is safe to restrict our eyes to the first case. The probability that case 1 happens in all n rounds is at least $(1 - \frac{m}{|I|})^n \geq 1 - \frac{nm}{|I|}$. Note that in the final round this is still the case as Arthur can directly compute $\widetilde{\varphi}(\xi_1, \ldots, \xi_n)$. Therefore again it suffices to pick $|I| = O(nm)$.

Now everything seems fine except one issue, given a sufficiently large range I, we may end up computing a number of size exponential in n. For example let's pick $I = \{1, 2, \ldots, 4nm\}$, then the maximum number is $N = O((4nm)^n)$. One idea to solve this issue is to do the same as what we did in the randomized algorithm for the identity testing of arithmetic circuits. In each step, we randomly pick a number p, use the deterministic primality testing to ensure that it is a prime, and modulo the result by p. This clearly does not affect the completeness. For soundness, first note that by prime number theorem, the number of primes less than N is roughly $\frac{N}{\ln N}$. On the other hand the bad case is when $g'(\xi_1) \neq g_1(\xi_1)$ but $g'(\xi_1) = g_1(\xi_1) \pmod p$. However in this case $g'(\xi_1) - g_1(\xi_1)$ has at most $\log N$ prime factors. Therefore the probability that the remainder modulo p is zero is at most $\frac{\log N}{N/\ln N} \approx \frac{\log^2 N}{N} < 1/4$ for sufficiently large N. This only increases the error probability by a constant factor so the argument is not affected. □

Our final result of this lecture is to show that, actually, IP = PSPACE. The proof still uses sumcheck protocol, but some interesting tweak is needed.

Theorem 5. IP = PSPACE.

The direction that IP ⊆ PSPACE is not difficult and is left as an exercise. Consider the direction that PSPACE ⊆ IP, which is more interesting. One idea is to extend the proof of IP for coSAT to the case of TQBF. Suppose n is odd and let the TQBF formula be $\exists x_1 \forall x_2 \cdots \exists x_n \varphi(x_1, \ldots, x_n)$, where x_i is a group of k_i Boolean variables. Now it is clear that, instead of checking a sum, we need to check an alternating sum and products, as follows:

$$\sum_{x_1 \in \{0,1\}^{k_1}} \prod_{x_2 \in \{0,1\}^{k_2}} \cdots \sum_{x_n \in \{0,1\}^{k_n}} \varphi(x_1, \ldots, x_n) = a$$

The arithmetization remains the same. However the issue is that the product may increase the degree of a variable in the polynomial. As an example let $\varphi(x,y) = (x \vee \overline{y}) \wedge (x \vee y)$ where x, y are Boolean variables. $\widetilde{\varphi}(x,y) = (1-(1-x)y)(1-(1-x)(1-y))$. Then $\widetilde{\varphi}(x,0) = \widetilde{\varphi}(x,1) = x$. Therefore $\sum_{y\in\{0,1\}} \widetilde{\varphi}(x,y) = 2x$ while $\prod_{y\in\{0,1\}} \widetilde{\varphi}(x,y) = x^2$. Because of this, we cannot bound the degree of each variable by the number of clauses which is crucial in our analysis of soundness.

Nevertheless this approach could lead to a proof, where very roughly after each product phase, a delicate trick is applied to keep the degree small while maintaining the soundness. Rather than explaining this somewhat unnatural idea we will follow a different route and employ the idea of divide and conquer used in the proof of NSPACE($s(n)$) $\subseteq$ DSPACE($s^2(n)$).

Proof. (Of Theorem 5) Consider the direction PSPACE $\subseteq$ IP. Let $\Sigma = \{0,1\}$ be the binary alphabet, M be a DSPACE($s(n)$) machine, C_0 be the initial configuration and C_1 be the final accepting configuration. On input w, we want to check whether $C_0 \vdash_{2^s,M} C_1$ where this represents C_0 can reach C_1 in 2^s steps. Define $Q_\ell(C_0, C_1)$ such that

$$Q_\ell(C_0,C_1) = \begin{cases} 1 & \text{if } C_0 \vdash_{2^\ell,M} C_1 \\ 0 & \text{otherwise} \end{cases}$$

There are two observations. First, there is a natural recursion between these predicates, namely

$$Q_{\ell+1}(C_0,C_1) = \sum_{C_{1/2}\in\{0,1\}^s} Q_\ell(C_0,C_{1/2})Q_\ell(C_{1/2},C_1)$$

Second, as what we do in the previous theorem, we view $Q_s(C_0,C_1)$ as the value of a function evaluated at point C_0, C_1. Precisely, let $x = (x_1,\ldots,x_s)$ and $y = (y_1,\ldots,y_s)$ be $2s$ variables then $Q_s(C_0,C_1)$ can be viewed as $Q_s(x,y)$ at point $x = C_0$, $y = C_1$.

The next step is to arithmetize these predicates. First, there exists a polynomial $\widetilde{Q}_0(x,y)$ of degree $O(1)$ in each variable that agrees with $Q_0(C_i,C_j)$ on the Boolean cube where C_i, C_j represent two arbitrary configurations of s bits. The proof of this exploits the high locality of Turing machine and is left as an exercise. Now with the recursion mentioned above, $\widetilde{Q}_1,\ldots,\widetilde{Q}_s$ are all well defined. By induction on i each of the $2s$ variables in Q_s has degree $O(1)$.

Now it suffices to work with the arithmetized polynomials. Let I be a sufficiently large range to work with, we want to check whether $\widetilde{Q}_s(C_0,C_1) = 1$, by recursion it is equivalent to check whether $\sum_{C_{1/2}} \widetilde{Q}_{s-1}(C_0,C_{1/2})\widetilde{Q}_{s-1}(C_{1/2},C_1) = 1$. Let $C_{1/2} = (z_1,\ldots,z_s)$ then the RHS can be rewritten as a series of sums and thus the check is equivalent to the following.

$$\sum_{z_1\in\{0,1\}} \cdots \sum_{z_s\in\{0,1\}} \widetilde{Q}_{s-1}(C_0,z_1,\ldots,z_s)\widetilde{Q}_{s-1}(z_1,\ldots,z_s,C_1) = 1$$

Now we can apply sumcheck protocol to this problem and reduce it to check

$$\widetilde{Q}_{s-1}(C_0,\xi_1,\ldots,\xi_s)\widetilde{Q}_{s-1}(\xi_1,\ldots,\xi_s,C_1) = \kappa_s$$

For $\xi_1,\ldots,\xi_s \in I$ and some value κ_s. Note that problem of checking $\widetilde{Q}_s(C_0,C_1) = 1$ has been successfully reduced to the problem of checking $\widetilde{Q}_{s-1}(C_0,\xi_1,\ldots,\xi_s)\widetilde{Q}_{s-1}(\xi_1,\ldots,\xi_s,C_1) = \kappa_s$. An

important issue arising is that the latter check is a product, and a direct recursion gives exponentially terms to check. The essential new tweak is to construct a polynomial $\delta(x)$ such that

$$\delta(x) = \begin{cases} (C_0, r) & \text{if } x = 0 \\ (r, C_1) & \text{if } x = 1 \end{cases}$$

$\delta(x)$ has degree at most 1. Let $g_{s-1}(x) = \widetilde{Q}_{s-1}(\delta(x))$. Observe that g_{s-1} is a univariate polynomial of degree $O(s)$. Therefore it suffices to check $g_{s-1}(0)g_{s-1}(1) = \kappa_s$. Now the important thing is that we can apply the idea of sumcheck again, though it's not completely the same. Arthur asks the prover to send description of g_{s-1}, say the proof is g'_{s-1}. Arthur first checks whether $g'_{s-1}(0)g'_{s-1}(1) = \kappa_s$. If so then Arthur randomly picks a value ξ (yes, in order to force Merlin to continue to cheat), evaluate $g'_{s-1}(\xi) = \kappa_{s-1}$, and recurse to check whether $g_{s-1}(\xi) = \kappa_{s-1}$. In this way the problem is reduced to check whether $\widetilde{Q}_{s-1}(\delta(\xi)) = \kappa_{s-1}$, which is of the same type as $\widetilde{Q}_s(C_0, C_1) = 1$. Now the procedure can be iterated by applying sumcheck again. In the end it is reduced to check whether

$$\widetilde{Q}_0(\gamma) = \kappa_0$$

As in the case of theorem 4, $\widetilde{Q}_0$ is explicit and thus can be checked directly. If the equality holds then we accept and this is the only place where we are willing to accept.

Let's analyze this protocol. There are s stages (from $\widetilde{Q}_s$ to $\widetilde{Q}_0$), In each stage there are two phases. The first phase we go over all $C_{1/2}$, there the degree of each variable of the polynomial is $O(1)$, therefore the probability that the false claim is transformed into a correct one is bounded by $O(\frac{1}{|I|})$. The second phase, we construct $\delta(x)$, and transformed into check $\widetilde{Q}_i(\delta(\xi)) = \kappa_i$. In this case the error probability is bounded by $\frac{O(s)}{|I|}$. So in each stage the error probability is $s \cdot O(\frac{1}{|I|}) + \frac{O(s)}{|I|} = O(\frac{s}{|I|})$. Therefore the total error probability is $O(\frac{s^2}{|I|})$ and picking sufficiently large I works.

Finally, like theorem 4, we must ensure evaluating a polynomial in time polynomial in the input length. This can be done similarly by modulo a random prime in each computation. The same analysis goes through and the proof is complete. □