COSC 544 Probabilistic Proof Systems 10/17/17

A State of the Art MIP For Circuit Satisfiability

Lecturer: Justin Thaler

1 A 2-Prover MIP for Low-Depth Arithmetic Circuit Satisfiability

The succinct argument from the previous lecture can be directly adapted to yield a 2-prover MIP. The idea is to use the second prover to function as the polynomial commitment scheme.

In more detail, the verifier uses the first prover to apply the GKR protocol to the claim $\mathcal{C}(x,w) = y$. As explained last lecture, at the end of this protocol, the prover makes a claim about $\tilde{w}(r)$.

Last lecture, this claim was checked by forcing the prover to reveal $\tilde{w}(r)$ via the polynomial commitment protocol (which itself involved a set-commitment combined with a low-degree test).

In the MIP, the verifier simply uses the second prover to play the role of the polynomial commitment scheme (i.e., to provide a claimed value for $\tilde{w}(r)$ that does not depend on the questions the verifier asked to the first prover, and to execute a low-degree test).

For example, if the low-degree test used is the point-versus-line test, then the verifier picks a random line λ in $\mathbb{F}^{\log n}$ containing r, and sends λ to the second prover, who is asked to respond with a univariate polynomial of degree $\log n$ claimed to equal $\tilde{w}$ restricted to λ. Since r is on the line λ, this univariate polynomial implicitly specifies $\tilde{w}(r)$, and the verifier checks that this value matches the first prover's claim about $\tilde{w}(r)$.

2 A 2-Prover MIP for General Arithmetic Circuit Satisfiability

2.1 Motivation

A downside of the 2-prover MIP for arithmetic circuit satisfiability from the previous section is that the communication cost and the verifier's runtime grow linearly with the circuit depth. So the protocol does not save the verifier time for deep, narrow circuits.

In general, this is not a major downside, because Lecture 11 explained that *any* computer program running in time T can be turned into an equivalent instance of arithmetic circuit satisfiability where the circuit is short and wide rather than long and narrow (specifically, the circuit has depth roughly $O(\log T)$).

Nonetheless, it is interesting to give a 2-prover MIP that directly handles deep, narrow circuits as efficiently as it handles short and wide ones. In so doing, we will see some ideas that will recur in later lectures when we study argument systems based on PCPs and linear PCPs.

The 2-Prover MIP of this section is from Blumberg et al. [BTVW14]. It combines several new ideas with techniques from the original **MIP** = **NEXP** proof of [BFL91], as well as the GKR protocol [GKR08] and its refinements by Cormode, Mitzenmacher, and Thaler [CMT12].

2.2 Protocol Summary

2.2.1 Terminology

Let $\mathcal{C}$ be an arithmetic circuit over a field $\mathbb{F}$ taking an explicit input x and a non-deterministic input w. Let $S = 2^s$ denote the number of gates in $\mathcal{C}$, and assign each gate in $\mathcal{C}$ a binary label in $\{0,1\}^s$. Refer to an assignment of values to each gate of $\mathcal{C}$ as a *transcript* of $\mathcal{C}$, and view the transcript as a function $W : \{0,1\}^s \to \mathbb{F}$ mapping gate labels to their values.

Given a claim that $\mathcal{C}(x,w) = y$, a *correct transcript* for is a transcript in which the values assigned to the input gates are those of x, the intermediate values correspond to the correct operation of each gate in $\mathcal{C}$, and the values assigned to the output gates are y. The arithmetic circuit satisfiability problem on instance Given a triple $\{\mathcal{C},x,y\}$ is equivalent to determining whether there is a correct transcript for $\{\mathcal{C},x,y\}$.

2.2.2 The MIP

The MIP works by having $\mathcal{P}_1$ claim find and "hold" an extension Z of a correct transcript W for $\{\mathcal{C},x,y\}$. If the prover is honest, then Z will equal $\widetilde{W}$, the multilinear extension of W. The protocol then identifies a polynomial $g_{x,y,Z} \colon \mathbb{F}^{3s} \to \mathbb{F}$ (which depends on x, y, and Z) satisfying the following property: $g_{x,y,Z}(\mathbf{a},\mathbf{b},\mathbf{c}) = 0$ for all Boolean inputs $(\mathbf{a},\mathbf{b},\mathbf{c}) \in \{0,1\}^{3s} \iff Z$ is indeed an extension of a correct transcript W.

To check that $g_{x,y,Z}$ vanishes at all Boolean inputs, the protocol identifies a related polynomial $h_{x,y,Z}$ such that $g_{x,y,Z}$ vanishes at all Boolean inputs $\iff$ the following equation holds:

$$\sum_{(\mathbf{a},\mathbf{b},\mathbf{c}) \in \{0,1\}^{3s}} h_{x,y,Z}(\mathbf{a},\mathbf{b},\mathbf{c}) = 0. \tag{1}$$

(Strictly speaking, the polynomial $h_{x,y,Z}$ is randomly generated, and there is a small chance over the random choice of $h_{x,y,Z}$ that Equation (1) holds even though $g_{x,y,Z}$ does not vanish at all Boolean inputs). The MIP applies the sum-check protocol to the polynomial $h_{x,y,Z}$ to compute this sum (note that if Z is a low-degree polynomial, then so is $h_{x,y,Z}$, as is required both to control costs and guarantee soundness in the sum-check protocol).

At the end of the sum-check protocol, $\mathcal{V}$ needs to evaluate $h_{x,y,Z}$ at a random point, which in turn requires evaluating Z at a random point $\mathbf{r} \in \mathbb{F}^s$. Unfortunately, $\mathcal{V}$ cannot compute $Z(\mathbf{r})$, since $\mathcal{V}$ does not have access to the polynomial Z (as Z only "exists" in $\mathcal{P}_1$'s head). Instead, $\mathcal{V}$ asks $\mathcal{P}_2$ to send her $Z(\mathbf{r})$, using a primitive called a *low-degree test*. Specifically, $\mathcal{P}_2$ is asked to send Z restricted to a plane Q, where Q is chosen to be a random plane in $\mathbb{F}^s$ containing $\mathbf{r}$. This forces $\mathcal{P}_2$ to implicitly make a claim about $Z(\mathbf{r})$ (note that $\mathcal{P}_2$ does not know *which* point in Q is $\mathbf{r}$); $\mathcal{V}$ rejects if $\mathcal{P}_1$ and $\mathcal{P}_2$'s claims about $Z(\mathbf{r})$ are inconsistent, and accepts otherwise.

The low-degree test cannot guarantee that Z itself is a low-degree polynomial, since $\mathcal{V}$ only ever inspects Z at a small number of points. Hence it is impossible to argue that $h_{x,y,Z}$ itself satisfies Equation (1): the soundness analysis for the sum-check protocol breaks down if the polynomial to which it is applied has large degree. However, the low-degree test *does* guarantee that if $\mathcal{P}_1$ and $\mathcal{P}_2$'s claims about $Z(\mathbf{r})$ are consistent with non-negligible probability over the random choice of $\mathbf{r}$, then Z is close to a low-degree polynomial Y, in the sense that $Y(\mathbf{r}') = Z(\mathbf{r}')$ for a large fraction of points $\mathbf{r}' \in \mathbb{F}^{3s}$. Since $h_{x,y,Y}$ *is* low-degree, it is straightforward to tweak the soundness analysis of the sum-check protocol to argue that $h_{x,y,Y}$ satisfies Equation (1), and hence that Y extends a correct transcript for $\{\mathcal{C},x,y\}$ (cf. Theorem 2.2).

Preview: The importance of checking that a polynomial vanishes on designated a subspace. The problem of checking that a certain polynomial $g_{x,y,Z}$ vanishes on a designated subspace plays a central role

in many MIPs and PCPs. The problem is sometimes referred to as checking a *Vanishing Reed-Solomon code* [BS08].

This problem will arise several more times in this course, including in state of the art PCPs and linear PCPs described in several lectures. One difference is that in the PCP and linear PCPs of later lectures, the polynomial $g_{x,y,Z}$ is *univariate*, instead of $(3s)$-variate as in the MIP considered here. □

Comparison to the GKR Protocol. While the GKR protocol verifies the claim $\mathcal{C}(x,w) = y$ layer by layer, with a a different instance of the sum-check protocol required for each layer of $\mathcal{C}$, the MIP of this section verifies the whole circuit in one shot, using a single invocation of the sum-check protocol. The reason the GKR protocol must work layer-by-layer is that the verifier must force the prover to make a claim about (the multilinear extension of) *the input alone*, since the verifier never materializes the intermediate gates of the circuit. This is not necessary in the multi-prover setting: in the MIP, $\mathcal{P}_1$ makes a claim about an extension Z of *the entire transcript*. $\mathcal{V}$ cannot check this claim independently, but that is okay because there is a second prover to ask for help.

2.3 Protocol Details

Notation. Let $\text{add}, \text{mult}\colon \{0,1\}^{3s} \to \{0,1\}$ denote the functions that take as input three gate labels $(\mathbf{a},\mathbf{b},\mathbf{c})$ from $\mathcal{C}$ and outputs 1 if and only if gate $\mathbf{a}$ adds (respectively, multiplies) the outputs of gates $\mathbf{b}$ and $\mathbf{c}$. While the GKR protocol had separate functions add_i and mult_i for each layer of $\mathcal{C}$, the MIP of this section arithmetizes all of $\mathcal{C}$ at once. We also add a third wiring predicate, which has no analog within the GKR protocol: let $\text{io}\colon \{0,1\}^{3s} \to \{0,1\}$ denote the function that returns 1 when gate $\mathbf{a}$ is either a gate from the explicit input x or one of the output gates, and gates $\mathbf{b}$ and $\mathbf{c}$ are the in-neighbors of $\mathbf{a}$ (input gates have in-neighbors $\mathbf{b} = \mathbf{c} = \mathbf{0}$).

Notice that add, mult, and io are independent of the inputs x and purported outputs y. The final function that plays a role in the MIP does depend on x and y. Define $I_{x,y}\colon \{0,1\}^s \to \mathbb{F}$ such that $I_{x,y}(\mathbf{a}) = x_{\mathbf{a}}$ if $\mathbf{a}$ is the label of an input gate, $I_{x,y}(\mathbf{a}) = y_{\mathbf{a}}$ if $\mathbf{a}$ is the label of an output gate, and $I_{x,y}(\mathbf{a}) = 0$ otherwise.

Lemma 2.1. *For $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c})\colon \{0,1\}^{3s} \to \mathbb{F}$ defined as below, $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = 0$ for all $(\mathbf{a},\mathbf{b},\mathbf{c}) \in \{0,1\}^{3s}$ if and only if W is a correct transcript for $\{C,x,y\}$:*

$$G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = \text{io}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (I_{x,y}(\mathbf{a}) - W(\mathbf{a})) + \text{add}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (W(\mathbf{a}) - (W(\mathbf{b}) + W(\mathbf{c}))) + \text{mult}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (W(\mathbf{a}) - W(\mathbf{b}) \cdot W(\mathbf{c})).$$

Proof. If W is not a correct transcript, there are five cases:

1. Suppose $\mathbf{a} \in \{0,1\}^s$ is the label of an input gate. If $W(\mathbf{a}) \neq x_{\mathbf{a}}$, then $G_{x,y,W}(\mathbf{a},0,0) = I_{x,y}(\mathbf{a}) - W(\mathbf{a}) = x_{\mathbf{a}} - W(\mathbf{a}) \neq 0$.

2. Suppose $\mathbf{a} \in \{0,1\}^s$ is the label of a non-output addition gate with in-neighbors $\mathbf{b}$ and $\mathbf{c}$. If $W(\mathbf{a}) \neq W(\mathbf{b}) + W(\mathbf{c})$, then $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = W(\mathbf{a}) - (W(\mathbf{b}) + W(\mathbf{c})) \neq 0$.

3. Suppose $\mathbf{a} \in \{0,1\}^s$ is the label of a non-output multiplication gate with in-neighbors $\mathbf{b}$ and $\mathbf{c}$. If $W(\mathbf{a}) \neq W(\mathbf{b}) \cdot W(\mathbf{c})$, then $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = W(\mathbf{a}) - (W(\mathbf{b}) \cdot W(\mathbf{c})) \neq 0$.

4. Suppose $\mathbf{a} \in \{0,1\}^s$ is the label of an output addition gate with in-neighbors $\mathbf{b}$ and $\mathbf{c}$. If $y_{\mathbf{a}} \neq W(\mathbf{b}) + W(\mathbf{c})$, then $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = I_{x,y}(\mathbf{a}) - W(\mathbf{a}) + (W(\mathbf{a}) - (W(\mathbf{b}) + W(\mathbf{c}))) = y_{\mathbf{a}} - (W(\mathbf{b}) + W(\mathbf{c})) \neq 0$.

5. Suppose $\mathbf{a} \in \{0,1\}^s$ is the label of an output multiplication gate with in-neighbors $\mathbf{b}$ and $\mathbf{c}$. If $y_{\mathbf{a}} \neq W(\mathbf{b}) \cdot W(\mathbf{c})$, then $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = I_{x,y}(\mathbf{a}) - W(\mathbf{a}) + (W(\mathbf{a}) - (W(\mathbf{b}) \cdot W(\mathbf{c}))) = y_{\mathbf{a}} - (W(\mathbf{b}) \cdot W(\mathbf{c})) \neq 0$.

On the other hand, if W is a correct transcript then it is immediate from the definition of $G_{x,y,W}$ that $G_{x,y,W}(\mathbf{a},\mathbf{b},\mathbf{c}) = 0$ for all $(\mathbf{a},\mathbf{b},\mathbf{c}) \in \{0,1\}^{3s}$. □

For any polynomial $Z\colon \mathbb{F}^s \to \mathbb{F}$, define the associated polynomial:

$$g_{x,y,Z}(\mathbf{a},\mathbf{b},\mathbf{c}) = \widetilde{\text{io}}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (\widetilde{\text{I}}_{x,y}(\mathbf{a}) - Z(\mathbf{a})) + \widetilde{\text{add}}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (Z(\mathbf{a}) - (Z(\mathbf{b}) + Z(\mathbf{c}))) + \widetilde{\text{mult}}(\mathbf{a},\mathbf{b},\mathbf{c}) \cdot (Z(\mathbf{a}) - Z(\mathbf{b}) \cdot Z(\mathbf{c})).$$

It follows from Lemma 2.1 that Z extends a correct transcript W if and only if $g_{x,y,Z}$ vanishes on the Boolean hypercube. We now define a polynomial $h_{x,y,Z}$ such that $g_{x,y,Z}$ vanishes on the Boolean hypercube if and only if $\sum_{\mathbf{u} \in \{0,1\}^{3s}} h_{x,y,Z}(\mathbf{u}) = 0$.

Defining $h_{x,y,Z}$. Consider a polynomial $k_{x,y,Z}$ whose *coefficients* are given by the evaluations of $k_{x,y,Z}$ on $\{0,1\}^{3s}$. Specifically, define

$$k_{x,y,Z}(t) = \sum_{\mathbf{u} \in \{0,1\}^{3s}} g_{x,y,Z}(\mathbf{u}) \cdot t^{\mathbf{u}}.$$

Here, the bit string $\mathbf{u} \in \{0,1\}^{3s}$ in the exponent denotes the integer $\sum_{i=0}^{3s-1} u_i \cdot 2^i$ (i.e., the integer whose binary representation is $\mathbf{u}$). Notice that $k_{x,y,Z}$ is the zero polynomial if and only if $G_{x,y,Z}$ vanishes on $\{0,1\}^{3s}$. Moreover, $k_{x,y,Z}(t)$ is a univariate polynomial of degree at most S^3, and so it can have at most S^3 roots if it is nonzero. So if $\mathcal{V}$ picks a random point q from $[S^4]$ and determines that $k_{x,y}(q) = 0$, it is safe for $\mathcal{V}$ to believe that $k_{x,y,Z}$ is the zero polynomial.

Thus, in the MIP, $\mathcal{V}$ chooses q uniformly at random from the set $[S^4]$, and is convinced that Z extends a correct transcript for $\{\mathcal{C}, x, y\}$ as long as $k_{x,y,Z}(q) = 0$. As explained below, $\mathcal{V}$ will outsource the computation of $k_{x,y,Z}(q)$ by writing it in a form that is amenable to checking via the sum-check protocol.

For any $q \in [S^4]$ and $\mathbf{u} \in \{0,1\}^{3s}$, $q^{\mathbf{u}}$ can be written as a multilinear polynomial $p_q(\mathbf{u})$ in the coordinates of $\mathbf{u}$ as follows. Define $q^{(i)} = q^{2^i}$. Then it holds that $q^{\mathbf{u}} = \prod_{i=0}^{3s-1} q^{(i)^{u_i}} = \prod_{i=0}^{3s-1} \left(1 + (q^{(i)} - 1)u_i\right) := K_q(\mathbf{u})$. Defining the polynomial $h_{x,y,Z}\colon \mathbb{F}^{3s} \to \mathbb{F}$ as $h_{x,y,Z}(\mathbf{u}) = g_{x,y,Z}(\mathbf{u}) \cdot K_q(\mathbf{u})$, it holds that:

$$k_{x,y,Z}(q) = \sum_{\mathbf{u} \in \{0,1\}^{3s}} g_{x,y,Z}(\mathbf{u}) \cdot K_q(\mathbf{u}) = \sum_{\mathbf{u} \in \{0,1\}^{3s}} h_{x,y,Z}(\mathbf{u}).$$

Hence, with probability $1 - 1/S$ over the random choice of q, $g_{x,y,Z}$ vanishes on the Boolean hypercube if and only if $\sum_{u \in \{0,1\}^{3s}} h_{x,y,Z}(\mathbf{u}) = 0$. For simplicity, the remainder of the presentation ignores the $1/S$ probability of error in this step (the $1/n$ can be folded into the soundness error of the entire MIP).

2.3.1 Applying the Sum-Check Protocol to $h_{x,y,Z}$

$\mathcal{V}$ applies the sum-check protocol to $h_{x,y,Z}$, with $\mathcal{P}_1$ playing the role of the prover in this protocol. To perform the final check in this protocol, $\mathcal{V}$ needs to evaluate $h_{x,y,Z}$ at a random point $\mathbf{r} \in \mathbb{F}^{3s}$. Let $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ denote the first, second, and third s entries of $\mathbf{r}$. Then evaluating $h_{x,y,Z}(\mathbf{r})$ requires evaluating $K_q(\mathbf{r})$, $\widetilde{\text{io}}(\mathbf{r})$, $\widetilde{\text{add}}(\mathbf{r})$, $\widetilde{\text{mult}}(\mathbf{r})$, $\widetilde{\text{I}}_{x,y}(\mathbf{r}_1)$, $Z(\mathbf{r}_1)$, $Z(\mathbf{r}_2)$, and $Z(\mathbf{r}_3)$. $\mathcal{V}$ can compute the first five evaluations without help in $O(\log(T))$ time, assuming that $\widetilde{\text{add}}$ and $\widetilde{\text{mult}}$ can be computed within this time bound. However, $\mathcal{V}$ cannot evaluate $Z(\mathbf{r}_1)$, $Z(\mathbf{r}_2)$, or $Z(\mathbf{r}_3)$ without help. To deal with this, the verifier first uses the "Reducing the Verification of a Single Point" technique from Lecture 8 (on the GKR protocol), to reduce the evaluation of Z at the three points $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$, to the evaluation of Z at a single point $\mathbf{r}_4 \in \mathbb{F}^s$. To obtain the evaluation $Z(\mathbf{r}_4)$, $\mathcal{V}$ turns to $\mathcal{P}_2$.

2.4 The Low-Degree Test

The paper [BTVW14] uses the *point vs. plane* low-degree test, as analyzed by Moshkovitz and Raz [MR08]. It could have used the point-versus-line test from the previous lecture, and this would shave a logarithmic factor off of $\mathcal{P}_2$'s asymptotic time cost, as well as the total communication cost of the MIP (see Remark 1). However, the constants appearing in Arora and Sudan's analysis of the point-versus-line test [AS03] are too large to yield a practical protocol, as the verifier would be forced to work over an enormous field to guarantee a suitable soundness error.

The point-versus-plane test involves two oracles Z, A', with Z modeling the first prover in our MIP, and A' modeling the second. The oracle Z takes as input a $\mathbf{r}_4 \in \mathbb{F}^s$ and returns some value $Z(\mathbf{r}_4)$. Let $\mathcal{Q}$ denote the set of all planes in $\mathbb{F}^s$. A' takes as input a plane $Q \in \mathcal{Q}$ returns some bivariate polynomial $A'(Q)$ of degree s in each variable (purported to be $Z \circ Q$ the restriction of Z to the plane specified by Q).

In the low-degree test, the verifier chooses a random point $\mathbf{r}_4 \in \mathbb{F}^s$, and a random plane Q containing $\mathbf{r}_4$, and queries $A(\mathbf{r}_4)$ and $A'(Q)$, accepting if and only if the claims of A and A' regarding $Z(\mathbf{r}_4)$ are consistent.

Moshkovitz and Raz show that if the oracles pass the low-degree test with probability γ, then there is a polynomial Y of total degree at most s^2 such that A agrees with Y on a fraction of at least $\gamma - \varepsilon$ of the points in $\mathbb{F}^s$, where $\varepsilon = 2^7 \cdot s\left(|\mathbb{F}|^{-1/8} + s^2|\mathbb{F}^{-1/4}|\right)$. Note that $\varepsilon = o(1)$ if $|\mathbb{F}|$ is sufficiently large, say $|\mathbb{F}| > s^{10}$.

In summary, if $|\mathbb{F}| > s^{10}$, and $\mathcal{P}_1$ and $\mathcal{P}_2$ pass the low-degree test with probability γ, then there is a polynomial Y of total degree at most s^2 that agrees with Z on a $\gamma - o(1)$ fraction of points in $\mathbb{F}^s$.

2.4.1 MIP Soundness Analysis

Theorem 2.2. *Suppose that $\mathcal{P}_1$ and $\mathcal{P}_2$ convince the MIP verifier to accept with probability $\gamma > .5 + \varepsilon$ for $\varepsilon = \Omega(1)$. Then there is some polynomial Y such that $h_{x,y,Y}$ satisfies Equation* (1).

Detailed Sketch. Since $\mathcal{P}_1$ and $\mathcal{P}_2$ pass the low-degree test with probability at least γ, the low-degree test guarantees that there is some probability Y of total degree at most s^2 such that Z and Y agree on a $p \geq \gamma - o(1)$ fraction of points. Since Y has total degree at most s^2, $h_{x,y,Y}$ has total degree at most $s^2 + 2$.

Suppose that $h_{x,y,Y}$ does not satisfy Equation (1). Let us say that $\mathcal{P}_1$ *cheats* at round i of the sum-check protocol if he does not send the message that is prescribed by the sum-check protocol in that round, when applied to the polynomial $h_{x,y,Y}$. The soundness analysis of the sum-check protocol (Lecture 5) implies that if $\mathcal{P}_1$ falsely claims that $h_{x,y,Y}$ does satisfy Equation (1), then with probability at least $1 - 3s \cdot (s^2+2)/|\mathbb{F}| = 1 - o(1)$, $\mathcal{P}_1$ will be forced to cheat at all rounds of the sum-check protocol including the last one.

If $\mathcal{P}_1$ does cheat in the last round, the only possible rescue is for the verifier, in the final check of the protocol, to wind up choosing a point in $\mathbb{F}^{3s}$ at which $h_{x,y,Y}$ and $h_{x,y,Z}$ disagree. This only happens if $\mathcal{V}$ picks a point $\mathbf{r} \in \mathbb{F}^s$ for use in the low-degree test such that $Y(\mathbf{r}) \neq Z(\mathbf{r})$. But this occurs with probability only $1 - p = 1 - \gamma + o(1)$. In total, the probability that $\mathcal{P}_1$ passes all tests within the sum-check protocol is therefore at most $1 - \gamma + o(1)$. If $\gamma > \frac{1}{2}$, this contradicts the fact that $\mathcal{P}_1$ and $\mathcal{P}_2$ convince the MIP verifier to accept with probability at least γ. □

Recall that if $h_{x,y,Y}$ satisfies Equation (1), then $g_{x,y,Y}$ vanishes on the Boolean hypercube, and hence Y is an extension of a correct transcript for $\{\mathcal{C}, x, y\}$. So Theorem 2.2 implies that if the MIP verifier accepts with probability $\gamma > \frac{1}{2}$, then there is a correct transcript for $\{\mathcal{C}, x, y\}$.

Although the soundness error can reduced from $\frac{1}{2} + o(1)$ to an arbitrarily small constant with $O(1)$ independent repetitions of the MIP, this would be highly expensive in practice. Fortunately, [BTVW14] performs a more careful soundness analysis that establishes that the MIP itself, *without repetition*, has soundness error $o(1)$.

Communication	Rounds	$\mathcal{V}$ time	$\mathcal{P}_1$ and $\mathcal{P}_2$ time
$O\left(\log^2 S\right)$ field elements	$O(\log S)$	$O(n+\text{polylog}(S))$	$O(S\log S)$

Table 1: Costs of the MIP of Section 2.1. The stated costs assume that $\widetilde{\text{add}}$, $\widetilde{\text{mult}}$, and $\widetilde{\text{io}}$ can be evaluated in time $\text{polylog}(S)$. As explained in Lecture 11, this holds for the circuits generated by reductions from RAM simulation.

The bottleneck in the soundness analysis of Theorem 2.2 that prevents the establishment of soundness error less than $\frac{1}{2}$ is that, if the prover's pass the low-degree test with probability $\gamma < \frac{1}{2}$, then one can only guarantee that there is a polynomial Y that agrees with Z on a γ fraction of points. The verifier will choose a random point $\mathbf{r}$ in the sum-check protocol at which Y and Z disagree with probability $1-\gamma > \frac{1}{2}$, and in this case all bets are off.

The key to the stronger analysis is to use a stronger guarantee from the low-degree test, known as a *list-decoding guarantee*. Roughly speaking, the list-decoding guarantee ensures that if the oracles pass the low-degree test with probability γ, then there is a "small" number of low-degree polynomials $Q_1, Q_2, \ldots$ that "explain" essentially all of the tester's acceptance, in the sense that for almost all points $\mathbf{r}$ at which the low-degree test passes, $A(\mathbf{r})$ agrees with $Q_i(\mathbf{r})$ for at least one i. This allows one to argue that even if the provers pass the low-degree test with probability only $\gamma < \frac{1}{2}$, the sum-check protocol will still catch $\mathcal{P}_1$ in a lie with probability very close to 1.

2.4.2 Protocol Costs

Verifier's Costs. $\mathcal{V}$ and $\mathcal{P}_1$ exchanges two messages for each variable of $h_{x,y,Z}$, and where $\mathcal{P}_2$ exchanges two messages in total with $\mathcal{V}$. This is $O(\log S)$ messages in total. Each message from $\mathcal{P}_1$ is a polynomial of degree $O(1)$, while the message from $\mathcal{P}_2$ is a bivariate polynomial of total degree $O(\log S)$. In total, all messages can be specified using $O(\log^2 S)$ field elements (the bottleneck is $\mathcal{P}$'s message). As for $\mathcal{V}$'s runtime, the verifier has to process the provers' messages, and then to perform the last check in the sum-check protocol, she must evaluate $\widetilde{\text{add}}$, $\widetilde{\text{mult}}$, $\widetilde{\text{io}}$, and $\widetilde{\text{I}}$ at random points. The verifier requires $O(\log^2 S)$ time to process the provers' messages, and Lemma 1.8 of Lecture 4 implies that $\mathcal{V}$ can evaluate $\widetilde{\text{I}}$ at a random point in $O(n)$ time. We assume that $\widetilde{\text{add}}$, $\widetilde{\text{mult}}$, and $\widetilde{\text{io}}$ can be evaluated at a point in time $\text{polylog}(S)$ as well— as explained in Lecture 11, this (essentially) holds for the circuits generated by reductions from RAM simulation.

Prover's Costs. Using the techniques developed to implement the prover in the GKR protocol, specifically Method 2 described there, $\mathcal{P}_1$ can be implemented in $O(S\log S)$ time. If the circuit is data parallel, then $\mathcal{P}_1$ can be implemented in $O(S)$ time using the same techniques described in Lecture 8. $\mathcal{P}_2$ needs to specify $\widetilde{W} \circ Q$, where Q is a random plane in $\mathbb{F}^s$. It suffices for $\mathcal{P}_2$ to evaluate $\widetilde{W}$ at $O(\log^2 S)$ many points—using Lemma 1.8 of Lecture 4, this can be done in $O(S)$ time per point, resulting in a total runtime of $O(S\log^2 S)$.

Remark 1. If the point vs. line test were used in place of the point vs. plane test, $\mathcal{P}_2$ would only need to evaluate $\widetilde{W}$ at $O(\log S)$ points, which can be done in $O(S\log S)$ time, and the total communication cost would drop from $O(\log^2 S)$ to $O(\log S)$.

3 A Succinct Argument for Deep Circuits

Using any polynomial commitment scheme, one can turn the MIP of the previous section into a succinct argument for deep and narrow arithmetic circuits. Specifically, one gets rid of the second prover, and instead just had the first prover commit to $\tilde{W}$ at the start of the protocol. At the end of the verifier's interaction with the first prover in the MIP above, the first prover makes a claim about $\tilde{W}(\mathbf{r}_4)$, which the verifier checks directly by having the prover reveal it via the polynomial commitment protocol.

This succinct argument has an advantage over the approach to succinct arguments from Lecture 12 that was based directly on the GKR protocol: namely, the argument system based on the MIP of the previous section is succinct with a nearly-linear time verifier *even for deep and narrow circuits*. The *disadvantage* of the argument system from the previous section is that it applies the polynomial commitment scheme to the entire *transcript extension* $\tilde{W}: \mathbb{F}^{\log|\mathcal{C}|} \to \mathbb{F}$, whereas the argument system of Lecture 12 applied the polynomial commitment scheme only to the multilinear extension of the *witness* $\tilde{w}$.

Existing polynomial commitment schemes are the concrete bottlenecks in argument systems that use them. Since the witness w can be much smaller than circuit $\mathcal{C}$, applying the polynomial commitment scheme to $\tilde{w}$ can be significantly less expensive than applying it to $\tilde{W}$.

Besides, we've seen that short, wide circuits are "universal" in the context of succinct arguments, since any RAM running in time T can be turned into an instance of arithmetic circuit satisfiability of size close to T and depth close to $O(\log T)$. Accordingly, the instructor anticipates that the approach that Lecture 12 takes to building succinct arguments is typically preferable in practice to the approach of this section.

References

[AS03] Sanjeev Arora and Madhu Sudan. Improved low-degree testing and its applications. *Combinatorica*, 23(3):365–426, 2003.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991.

[BS08] Eli Ben-Sasson and Madhu Sudan. Short pcps with polylog query complexity. *SIAM J. Comput.*, 38(2):551–607, 2008.

[BTVW14] Andrew J. Blumberg, Justin Thaler, Victor Vu, and Michael Walfish. Verifiable computation using multiple provers. *IACR Cryptology ePrint Archive*, 2014:846, 2014.

[CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Practical verified computation with streaming interactive proofs. In Shafi Goldwasser, editor, *ITCS*, pages 90–112. ACM, 2012.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, STOC '08, pages 113–122, New York, NY, USA, 2008. ACM.

[MR08] Dana Moshkovitz and Ran Raz. Sub-constant error low degree test of almost-linear size. *SIAM J. Comput.*, 38(1):140–180, 2008.