CS294: Probabilistically Checkable and Interactive Proofs February 2, 2017

Doubly-Efficient Interactive Proofs

Instructor: Alessandro Chiesa & Igor Shinkar *Scribe: Lynn Chua*

1 Introduction

In Shamir's protocol, the running time of an honest prover on a QBF with n variables and m clauses is $2^{O(n)}$. Consider a machine M that runs in time T and space S. We would like an interactive proof to demonstrate to the verifier that $M(x) = 1$. The recursion for this computation has depth $\log T$, and at each level there are S possiblities. Hence an honest prover runs in time $\geq 2^{S \log T}$, which is inefficient.

Our meta-goal is to delegate tractable computations to untrusted parties, and for the verification that the computation was executed correctly to be considerably faster than performing the computation. For $x \in L \in \text{TimeSpace}(T, S)$, ideally in an interactive proof system we would like the prover to run in $\text{poly}(T, S)$ and verifier to run in $\text{poly}\log(T, S)$.

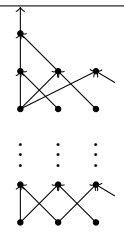
Theorem 1 (Goldwasser, Kalai, Rothblum, 08, [GKR15]) *Let L be a language that is decidable by a family of $O(\log S(n))$-space uniform circuits of size $S(n)$ and depth $D(n)$. Then L has an interactive proof such that the following properties are satisfied.*

- *Prover runs in time* $\text{poly}(S, D)$.
- *Verifier runs in time* $n\,\text{poly}(D, \log S)$.
- *Communication complexity (number of rounds) is* $\text{poly}(D, \log S)$.
- *Public coin.*

The proof of this theorem will take a few lectures. For today we remove the assumption of uniformity. Instead, we assume that the verifier has oracle access to information about the wires of the circuit. This is the "bare-bones protocol" in the paper [GKR15][Section 3].

2 Preliminaries

We assume that the circuit $C : \mathbb{F}^n \to \mathbb{F}$ is an arithmetic circuit over a field $\mathbb{F}$. We also assume that C is a *layered arithmetic circuit* with size S, depth D and fan-in 2, as illustrated below. We use the notation $[m] = \{0, 1, \ldots, m-1\}$ for all $m \in \mathbb{Z}_{>0}$.

[h] Layer		#Gates	Wire connections	Wire values
0 (output)		1		$V_0^* \in \mathbb{F}$
1		S	$\mathrm{add}_1, \mathrm{mul}_1$	$V_1^* : [S] \to \mathbb{F}$
2		S	$\mathrm{add}_2, \mathrm{mul}_2$	$V_2^* : [S] \to \mathbb{F}$
$\vdots$		$\vdots$	$\vdots$	
$D-1$		S	$\mathrm{add}_{D-1}, \mathrm{mul}_{D-1}$	$V_{D-1}^* : [S] \to \mathbb{F}$
D (input)		n	$\mathrm{add}_D, \mathrm{mul}_D$	$V_D^* : [n] \to \mathbb{F}$

In a layered arithmetic circuit of depth D, the gates are divided into $(D+1)$ layers, and the wires only connect gates in adjacent layers. Layer 0 is the output layer with 1 output gate, and layer D is the input layer with n input gates. For simplicity, we assume that layers 1 to $D-1$ all have S gates. The wire value functions V_i^*, for $0 \le i \le D$, map the gates at the i-th layer to their values.

For each layer i, $1 \le i \le D$, we define functions $\mathrm{add}_i, \mathrm{mul}_i : [S]^3 \to \{0,1\}$.

$$\mathrm{add}_i(a,b,c) = \begin{cases} 1 & \text{if } a \text{ is add gate in layer } i-1 \text{ and } b, c, \ (b \le c) \text{ are its inputs in layer } i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\mathrm{mul}_i(a,b,c) = \begin{cases} 1 & \text{if } a \text{ is multiply gate in layer } i-1 \text{ and } b, c, \ (b \le c) \text{ are its inputs in layer } i \\ 0 & \text{otherwise} \end{cases}. \tag{2}$$

We can identify $[S]$ with the boolean hypercube $\{0,1\}^{\log S}$, but this gives worse running times than in the theorem statement. A better choice would be to use Reed-Muller codes. Let $H \subset \mathbb{F}$ be a subfield such that $|H| = \log S$ and let $m = \frac{\log S}{\log |H|}$. We identify $[S]$ with H^m in the rest of this section.

We define the arithmetizations of $\mathrm{add}_i, \mathrm{mul}_i$ as $\hat{\mathrm{add}}_i, \hat{\mathrm{mul}}_i : \mathbb{F}^{3m} \to \mathbb{F}$. These are polynomials of degree at most δ in each variable, where $|H| - 1 \le \delta < |\mathbb{F}|$. We also define the functions $V_i : \mathbb{F}^m \to \mathbb{F}$ corresponding to the wire values V_i^*, where $1 \le i \le D-1$. The functions V_i have degree at most $|H|-1$ in each variable. Similarly, we define $V_0 \in \mathbb{F}$. As the last layer has n gates, we identify $[n]$ with $H^{m'}$, where $m' = \frac{\log n}{\log |H|} \le m$, and we define $V_D : \mathbb{F}^{m'} \to \mathbb{F}$.

3 Bare-bones protocol

We now describe the bare-bones protocol. The prover and verifier have as input $x \in \mathbb{F}^n$, and are given oracle access to the functions $\{\hat{\mathrm{add}}_i, \hat{\mathrm{mul}}_i\}_{1 \le i \le D}$, which specify the circuit C. The goal is for the prover to prove to the verifier that $C(x) = 0$. This is done in D phases.

We describe the protocol starting from the output layer, which is the first phase of the protocol. By the definitions above, we have

$$V_0 = \sum_{w_1, w_2 \in H^m} \hat{\mathrm{add}}_1(0, w_1, w_2) \cdot (V_1(w_1) + V_1(w_2)) + \hat{\mathrm{mul}}_1(0, w_1, w_2) \cdot (V_1(w_1) \cdot V_1(w_2)) \tag{3}$$

$$V_1(z) = \sum_{w_1, w_2 \in H^m} \hat{\mathrm{add}}_2(z, w_1, w_2) \cdot (V_2(w_1) + V_2(w_2)) + \hat{\mathrm{mul}}_2(z, w_1, w_2) \cdot (V_2(w_1) \cdot V_2(w_2)) \tag{4}$$

where $z \in H^m$. Each summand on the right hand side, when treated as a polynomial in w_1, w_2, has degree at most $\delta + |H| - 1 \le 2\delta$ in each variable.

The goal of the prover is to show that $V_0 = y_0$ for a claimed $y_0 \in \mathbb{F}$. This is done using the sum-check protocol. However, in the usual sum-check protocol the verifier would have to compute

$$\hat{\text{add}}_1(0, w_1^v, w_2^v) \cdot (V_1(w_1^v) + V_1(w_2^v)) + \hat{\text{mul}}_1(0, w_1^v, w_2^v) \cdot (V_1(w_1^v) \cdot V_1(w_2^v)) = y_1 \,, \tag{5}$$

for random $w_1^v, w_2^v \in \mathbb{F}^m$ (chosen by the verifier). The verifier has oracle access to $\hat{\text{add}}_1, \hat{\text{mul}}_1$, but the computation of $V_1(w_1^v), V_1(w_2^v)$ would require time poly(S), which breaks the assumption of the verifier's computational power in Theorem 1. Instead, the verifier sends w_1^v, w_2^v to the prover. The prover sends y_{11}, y_{12} to the verifier, with the claim that $V_1(w_1^v) = y_{11}$ and $V_1(w_2^v) = y_{12}$.

The next goal is to verify that $V_1(w_1^v) = y_{11}$ and $V_1(w_2^v) = y_{12}$. This reduces the task of proving $V_0 = y_0$ to proving these two claims. We further reduce these two claims to a single claim as follows.

- The verifier chooses $s, t \in \mathbb{F}$ and sends it to the prover.
- Let $\gamma : \mathbb{F} \to \mathbb{F}^m$ be the unique line such that $\gamma(s) = w_1^v$ and $\gamma(t) = w_2^v$. The prover sends the function $f = V_1 \circ \gamma : \mathbb{F} \to \mathbb{F}$ to the verifier. This is a polynomial of degree at most $m(|H| - 1)$.
- The verifier checks that $f(w_1^v) = y_{11}$ and $f(w_2^v) = y_{12}$. If so, the verifier sends a random $t \in \mathbb{F}$ to the prover.
- The prover and verifier proceed to the next phase, repeating the sum check protocol and the additional interactive protocol described above with the goal of proving that $V_1(\gamma(t)) = f(t)$.

When the prover and verifier reach the last layer, they proceed in a similar way with a few modifications as layer D has n gates whereas the previous layers have S gates. Thus in this phase the sum-check protocol is over $z, w_1, w_2 \in H^{m'}$.

After the last phase, the verifier has to verify on his own an equality of the form $V_d(z_d) = r_d$. We can write V_D as

$$V_D(z) = \sum_{w \in H^{m'}} \text{EQ}_{H,m'}(z, w) x_w \,, \tag{6}$$

where x is the input to the circuit and $\text{EQ}_{H,m'}$ denotes the equality polynomial

$$\text{EQ}_{H,m'}(z, w) = \prod_{i=1}^{m'} \sum_{a \in H} \prod_{\gamma \in H \setminus \{a\}} \frac{(z_i - \gamma)(w_i - \gamma)}{(\gamma - a)^2} \,. \tag{7}$$

We can check that if $z \in H^{m'}, w \in H^{m'}$, then $EQ_{H,m'}(z, w) = 1$ if $z = w$ and is 0 otherwise. $EQ_{H,m'}$ can take any values outside $H^{m'} \times H^{m'}$. We observe that EQ has degree $\leq |H|$ in each coordinate, and EQ can be evaluated efficiently at any point. Thus the verifier can perform this computation on his own to complete the final verification step.

References

[GKR15] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum, *Delegating computation: Interactive proofs for muggles*, Journal of the ACM **62** (2015), no. 4, 27:1–27:64.