The Power of Interactive Proof Systems

Notes by Yael Kalai

MIT - 6.5610
Lecture 11 (March 11, 2024)

> ***Warning:*** This document is a rough draft, so it may contain bugs. Please feel free to email me with corrections.

Outline

- Sumcheck protocol

Recap

Last lecture we defined the model of interactive proofs, which were defined for the goal of constructing zero-knowledge proofs. Today, and in the next few lectures, we will learn about the power of interactive proofs, and their impact on how proofs are designed today.

Jumping ahead, we will show how to use interactive proofs, together with cryptographic magic, to construct *"succinct proofs"* of correctness. Namely, we will show how given a Turing machine M, an input x and a time-bound T, one can compute the output $y = M(x)$ together with a "succinct proof" π that certifies that indeed M on input x outputs y within T steps.

We start by demonstrating the power of interactive proofs via the Sumcheck protocol, which is an interactive proof for a statement that we do not know how to prove succinctly using a classical proof.

Sumcheck Protocol

Intuitively, the Sumcheck protocol proves the value of the sum of a multivariate polynomial on exponentially many values. Specifically, let $\mathbb{F}$ be a finite field. One can think of $\mathbb{F} = \mathsf{GF}[p]$ which consists of the elements $\{0, 1, \ldots, p-1\}$ where addition and multiplication are modulo p.

Definition 1 (Sumcheck Problem)**.** Given a polynomial $f : \mathbb{F}^m \to \mathbb{F}$ of degree $\leq d$ in each variable and a fixed set $H \subseteq \mathbb{F}$, compute

$$\beta = \sum_{h_1,\ldots,h_m \in H} f(h_1, \ldots, h_m).$$

Often we consider the special case where $H = \{0, 1\}$.

Assuming that the verifier has oracle access to f, we will exhibit an interactive proof for the Sumcheck problem. While this problem seems very specific (and possibly not interesting) at first, it turns out that this is an important building block in many of our succinct proofs. In particular, it is the main building block in the proof that IP = PSPACE [2] and is the main building block in the GKR protocol which we will learn in the next lecture.

The Sumcheck protocol proceeds as follows:

1. The prover computes and sends
$$g_1(x) = \sum_{h_2,\ldots,h_m \in H} f(x, h_2, \ldots, h_m).$$
This is a unvariate degree $\leq d$ polynomial where the first argument to f is a free variable.
2. The verifier checks that $g_1(x)$ is a univariate polynomial of degree $\leq d$ and that $\sum_{h_1 \in H} g_1(h_1) = \beta$. (Reject if either check fails.)
3. The verifier sends a uniformly sampled $t_1 \xleftarrow{R} \mathbb{F}$.
4. The prover sends
$$g_2(x) = \sum_{h_3,\ldots,h_m \in H} f(t_1, x, h_3, \ldots, h_m).$$
This is again a univariate degree $\leq d$ polynomial, where the first argument of f has been fixed and the second argument is a free variable.
5. The verifier checks that $g_2(x)$ is degree $\leq d$ and that $\sum_{h_2 \in H} g_2(h_2) = g_1(t_1)$.
6. The verifier sends a uniformly sampled $t_2 \xleftarrow{R} \mathbb{F}$.
7. The prover replies with
$$g_3(x) = \sum_{h_4,\ldots,h_m \in H} f(t_1, t_2, x, h_4, \ldots, h_m).$$
8. The verifier checks that $g_3(x)$ is degree $\leq d$ and that $\sum_{h_3 \in H} g_3(h_3) = g_2(t_2)$.
9. Repeat this procedure on all other variables. The final check will be as follows:
10. The prover sends $g_m(x) = f(t_1, t_2, \ldots, t_{m-1}, x)$.
11. The verifier samples a uniform $t_m \xleftarrow{R} \mathbb{F}$ and checks that $g_m(t_m) = f(t_1, t_2, \ldots, t_m)$ using its oracle access to f. It Accepts if and only if all the checks have passed.

Analysis of the Sumcheck protocol

The completeness of this protocol is straightforward so we will focus on soundness. We will not give a formal proof, rather will give a high-level idea for why this protocol is sound. The soundness analysis is "round-by-round". Suppose that the instance is false. Namely suppose the instance is $f : \mathbb{F}^m \to \mathbb{F}$ of degree $\leq d$ in each variable, a set $H \subset \mathbb{F}$ and an element $\beta \in \mathbb{F}$ such that

$$\sum_{h_1,\ldots,h_m \in H} f(h_1, \ldots, h_m) \neq \beta.$$

Fix any cheating prover P^* that tries convince the verifier to accept this false statement. We argue that for each round i, if we start with a false claim of the form

$$g^*_{i-1}(t_{i-1}) = \sum_{h_i,\ldots,h_m \in H} f(t_1, \ldots, t_{i-1}h_i, \ldots, h_m) \tag{1}$$

(where $g^*_0 = \beta$), then the next round claim, which is of the form

$$g^*_i(t_i) = \sum_{h_{i+1},\ldots,h_m \in H} f(t_1, \ldots, t_i, h_{i+1}, \ldots, h_m), \tag{2}$$

is also false with with probability $\geq 1 - \frac{d}{|\mathbb{F}|}$ (assuming the verifier does not reject $g^*_i(\cdot)$). Thus, by a union bound, at the end of the Sumcheck protocol the verifier will reject P^* with probability $\geq 1 - \frac{dm}{|\mathbb{F}|}$. So we get "good" soundness if $|\mathbb{F}| >> dm$. To see why the round-by-round soundness holds note that if $g^*_{i-1}(t_{i-1})$ is false then g^*_i must also be false or else the verifier will reject it. This is the case since the verifier checks that

$$\sum_{h \in H} g^*_i(h) = g^*_{i-1}(t_{i-1}).$$

If g^*_i is false and is of degree d then it agrees with the true polynomial on at most d points, and thus $g^*_i(t)$ on a random $t \xleftarrow{R} \mathbb{F}$ remains incorrect with probability $1 - \frac{d}{|\mathbb{F}|}$.

Communication complexity. The protocol has m rounds of communication, one for each variable of f. In each round, the prover sends one degree-d polynomial, which is represented by d field elements; and the verifier sends one field element t_i. Therefore the communication complexity is $O(dm \log |\mathbb{F}|)$.

Runtime. In each of the m rounds, the verifier evaluates a degree-d polynomial on $|H|$ points; so the verifier runtime is $O(m \cdot |H| \cdot d \cdot \text{polylog}\, |\mathbb{F}|)$. The prover runs in time $O(m \cdot |H|^m \cdot T_f)$, where T_f denotes the time to compute f.

Remark. The Sum-Check protocol has the desirable property that the verifier only sends uniformly sampled field elements in each

round (each field element constitutes $\log |\mathbb{F}|$ random bits). Such a protocol is called a **public-coin** protocol. Public-coin protocols are of great interest because as we will see, we can later use cryptography to eliminate interaction from such protocols using the Fiat-Shamir transform (coming up, stay tuned!).

Why do we care about the Sumcheck protocol?

Beyond being a proof of concept that interactive proofs are powerful, the Sumcheck protocol is extremely important in the design of succinct proof systems. Indeed, the Sumcheck protocol was used by Shamir [2] to construct an interactive proof for any language in PSPACE. We will not show Shamir's protocol, rather we will show an alternative protocol (the GKR protocol [1]) that has efficiency advantages and is conceptually simpler. The main drawback of Shamir's protocol is that to prove the correctness of a time T space-S computation, the runtime of the prover is $\geq 2^{S \cdot logS}$, which may be exponential in T. The runtime of the verifier is proportional to S. This raises the following fundamental question:

Is proving necessarily harder than computing?

Doubly Efficient Interactive Proofs

So far we placed no restriction on the prover's runtime, and restricted only the verifier's runtime. Indeed, when interactive proofs were original defined they referred to the prover as Merlin (an all powerful wizard). In reality, however, we do care about the computational power of the prover. Of course, we still need to allow the prover more computational power than the verifier, as otherwise the prover is not helpful.

Definition 2 (Doubly-Efficient Interactive Proof (DE-IP))**.** A doubly-efficient interactive proof for a language $L \in \mathsf{DTIME}(T(n))$ is an interactive proof such that:

1. The honest prover's runtime is $\text{poly}(T)$.
2. The verifier's runtime is much less, ideally $\tilde{O}(n)$, where $\tilde{O}$ omits $\text{polylog}(n)$ factors.

In practice it is desirable that the prover's runtime is $O(T)$.

We will show how to use the Sumcheck protocol to construct a doubly efficient interactive proof for every bounded depth computation.

Theorem 3. *For any circuit C of depth D and size S (that is log-space uniform) there exists a doubly efficient interactive proof such that*

We will explain what the log-space uniformity condition is when we describe the GKR protocol

- *The number of rounds is* $D \cdot \text{polylog}(S)$.
- *The communication complexity is* $D \cdot \text{polylog}(S)$.
- *The verifier's runtime is* $O(n) + D \cdot \text{polylog}(S)$ *where n is the input length (assuming the circuit is log-space uniform)*
- *The prover's runtime is* $\text{poly}(S)$.

The doubly efficient interactive proof that achieves this theorem is called the GKR protocol [1]. The main ingredient used in the GKR protocol is the Sumcheck protocol!

References

[1] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 113–122. ACM, 2008.

[2] Adi Shamir. Ip=pspace. In *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15. IEEE Computer Society, 1990.