The GKR Protocol and SNARGs for NP

Notes by Yael Kalai

MIT - 6.5610
Lecture 19 (April 14, 2025)

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

Outline

- The GKR Protocol
- Succinct Interactive Proofs for NP
- SNARGs for NP

The GKR protocol [1] is a doubly efficient interactive proof that achieves the following guarantees.

Theorem 1. *For any circuit C of depth D and size S (that is log-space uniform) there exists a doubly efficient interactive proof such that*

- *The number of rounds is $D \cdot \text{polylog}(S)$.*
- *The communication complexity is $D \cdot \text{polylog}(S)$.*
- *The verifier's runtime is $\tilde{O}(n) + D \cdot \text{polylog}(S)$ where n is the input length (assuming the circuit is log-space uniform). Moreover, if the verifier has access to the multilinear extension of the input then it runs in time $D \cdot \text{polylog}(S)$ and accesses the oracle in a single point!*
- *The prover's runtime is $\text{poly}(S)$.*

C is log-space uniform if there exists a log-space machine that takes as input the description of three wires and outputs 1 iff there wires are connected via an ADD gate or via a MULT gate.

The protocol uses the notion of a multilinear extension, defined last class. Recall that the multilinear extension (MLE) of any function $f : \{0,1\}^m \to \{0,1\}$, with respect to a finite field $\mathbb{F}$, is the (unique) multilinear function $\tilde{f} : \mathbb{F}^m \to \mathbb{F}$ that agrees with f on all inputs in the domain $\{0,1\}^m$; i.e., $\forall(b_1, \ldots, b_m) \in \{0,1\}^m$,

One can think of f as an arbitrary vector of bits of length 2^m.

$$\tilde{f}(b_1, \ldots, b_m) = f(b_1, \ldots, b_m),$$

or more concisely $\tilde{f}\,|_{\{0,1\}^m} \equiv f$. Notice that the domain of $\tilde{f}$ is $\mathbb{F}^m$, a superset of $\{0,1\}^m$, hence the name "extension".

Theorem 2. *Let $f : \{0,1\}^m \to \{0,1\}$ be any function (i.e., an arbitrary sequence of 2^m bits) and let $\mathbb{F}$ be a (finite) field. Then there exists a unique multilinear polynomial $\tilde{f} : \mathbb{F}^m \to \mathbb{F}$ s.t. $\tilde{f} \mid_{\{0,1\}^m} \equiv f$ Moreover,*

$$\tilde{f}(t_1, \ldots, t_m) = \sum_{b_1,\ldots,b_m} f(b_1, \ldots, b_m) \cdot \tilde{\beta}(t_1, \ldots, t_m, b_1, \ldots, b_m),$$

where $\tilde{\beta}$ is the unique multilinear function that for every $x, y \in \{0,1\}^m$,

$$\tilde{\beta}(x,y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

The GKR protocol

Fix boolean circuit $C : \{0,1\}^n \to \{0,1\}$ of size (number of gates) S and depth D. The GKR protocol is an interactive proof for the fact that $C(x) = 1$. We assume that the verifier has a succinct description of C. Formally, we assume that C is log-space uniform i.e. it can be generated by some log-space Turing Machine M, and we assume that the verifier has a description of M. Assume without loss of generality that C is layered which means that each gate belongs to a layer, and each gate in layer i is connected by neighbors only in layer $i + 1$. Let layer 0 denotes the output layer and D denotes the input layer.

One can always layer a circuit by adding dummy intermediate gates. This can be done while increasing the depth to depth at most D^2.

The GKR protocol consists of D-subprotocols, where the i'th subprotocol reduces a claim about the value of a point in the MLE of layer $i - 1$ to a claim about the value of a point in the MLE of layer i. It has the guarantee that if the value corresponding to layer $i - 1$ was incorrect then with high probability, the value corresponding to layer i is also incorrect. Eventually, it will be reduced to a claim about the value of a point in the MLE of the input layer (layer D), which the verifier can compute on its own.

Detailed description of the protocol

Step 1: Arithmetize C. Convert C to a layered arithmetic circuit (over GF[2]) with fan-in 2. Arithmetic circuit (over GF[2]) means that it consists only of gates of the form ADD and MULT (where addition and multiplication are done modulo 2). We can convert any Boolean circuit, with gates $\wedge$ and $\neg$, into an arithmetic circuit, by converting a gate $\wedge$ into a gate MULT, and converting a gate $\neg$ into a gate ADD where we add a constant 1 as an input to the gate.

We assume for simplicity, and without loss of generality, that the number of wires in each layer is the same, and we abuse notation and denote it by S.

Step 2: The prover computes the values of all wires in every layer of the circuit. Let $m = \log S$. For layer i, define the function $V_i : \{0,1\}^m \to \{0,1\}$, where $V_i(b_1, \ldots, b_m)$ is the value of the wire in layer i, whose binary representation is $(b_1, \ldots, b_m$. Let $\tilde{V}_i : \mathbb{F}^m \to \mathbb{F}$ be its multilinear extension (MLE).

The Protocol: The protocol consists with D "reduction" protocols, where each reduction protocol reduces a claim of the form $\tilde{V}_i(z_i) = v_i$ about layer i to a claim of the form $\tilde{V}_{i+1}(z_{i+1}) = v_{i+1}$ about about layer $i+1$. We start with the output layer where the prover claims that $\tilde{V}_0(z_0) = v_0 = 1$ where $z_0 \in \{0,1\}^m$ is the label of the only non-dummy gate in layer 0 that holds the output of the circuit. At the end of these D reduction protocols we will be left with a claim of the form $\tilde{V_D}(z_D) = v_D$. The verifier can check this on its own since it knows the input.

Actually, the reduction protocol will reduce checking two such claims about layer i to two such claims about layer $i+1$.

The reduction protocol

For every $i \in [D]$ we define two functions $\mathsf{ADD}_i, \mathsf{MULT}_i : (\{0,1\}^m)^3 \to \{0,1\}$ as follows:

$$\mathsf{ADD}_i(p, w_1, w_2) = \begin{cases} 1 & \text{gate } p \text{ in layer } i \text{ is an } \mathsf{ADD} \text{ gate connecting } w_1 \text{ and } w_2 \text{ in layer } i+1 \\ 0 & \text{Otherwise} \end{cases}$$

MULT_i is defined similarly with ADD replaced with MULT in the definition. Let

$$\widetilde{\mathsf{ADD}}_i, \widetilde{\mathsf{MULT}}_i : \mathbb{F}^{3m} \to \mathbb{F}$$

be the MLEs of ADD_i and MULT_i, respectively.

We can expand out the MLE definition and rewrite the claim $\tilde{V}_i(z_i) = v_i$ as

$$v_i = \tilde{V}_i(z_i) = \sum_{p \in \{0,1\}^m} V_i(p) \tilde{\beta}(p, z_i)$$

Then we can further express $V_i(p)$ as combination of $\widetilde{\mathsf{ADD}}_i, \widetilde{\mathsf{MULT}}_i$:

$$v_i = \sum_{p \in \{0,1\}^m} \sum_{w_1, w_2 \in \{0,1\}^m} \Big[\widetilde{\mathsf{ADD}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{\mathsf{MULT}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) \cdot \tilde{V}_{i+1}(w_2))\Big] \tilde{\beta}(p, z_i)$$

Denote the polynomial inside the sum by

$$f(p, w_1, w_2) = \Big[\widetilde{\mathsf{ADD}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) + \tilde{V}_{i+1}(w_2)) + \widetilde{\mathsf{MULT}}_i(p, w_1, w_2)(\tilde{V}_{i+1}(w_1) \cdot \tilde{V}_{i+1}(w_2))\Big] \tilde{\beta}(p, z_i)$$

In the i'th subprotocol, the prover and verifier run the Sumcheck protocol for

$$v_i = \sum_{p,w_1,w_2 \in H^m} f(p, w_1, w_2).$$

This reduces the problem of verifying that $v_i = \tilde{V}_i(z_i)$ to checking that $f(z_0, z_1, z_2) = t$ for some $t \in \mathbb{F}$ and for random $z_0, z_1, z_2 \in \mathbb{F}^m$ chosen by the verifier in the Sumcheck protocol. By definition of f, this reduces to checking that

$$t = \Big[\widetilde{\mathsf{ADD}}_i(z_0, z_1, z_2)(\tilde{V}_{i+1}(z_1) + \tilde{V}_{i+1}(z_2)) + \widetilde{\mathsf{MULT}}_i(z_0, z_1, z_2)(\tilde{V}_{i+1}(z_1) \cdot \tilde{V}_{i+1}(z_2))\Big]\widetilde{\beta}(z_0, z_i)$$

We assume for now that the verifier can compute on its own $\widetilde{\mathsf{ADD}}_i$ and $\widetilde{\mathsf{MULT}}_i$. This is precisely where we use the log-space uniformity condition of the underlying circuit C.

So we reduced checking the value v_i in layer i to checking the value of two elements (z_1, z_2) in round $i + 1$. It seems like if we continue in this way the number of elements we will need to check will grow exponentially! However, surprisingly this is not the case! We can reduce checking two elements in round $i + 1$ to checking two elements in round $i + 2$.

The idea is to run two Sumcheck protocols (one for each element) but where the verifier uses the same randomness in both these Sumcheck protocols! This will convert checking two elements in round i to checking two elements in round $i + 1$.

Overcoming the Low-Depth Restriction

We can use the GKR blueprint to construct doubly efficient protocols for any computation where the verifier's runtime and the number of rounds *do not grow* with the depth of the computation! Rather they grow only poly-logarithmically with the size. The basic idea is to "flatten" the circuit. Namely, suppose the prover wishes to prove to the verifier that $C(x) = y$ where C is of size S. The idea is for the prover and verifier to consider a new *shallow* circuit C' that takes as input an S-bit string, which corresponds to the values of all the wires of C, and checks that all the gates of C are satisfied. Importantly, note that C' is very shallow and is of depth $O(\log S)$. So, the idea is to run the GKR protocol on the shallow circuit C'.

Assume that S is the number of wires in C.

The problem is that the verifier does not know the input to C' since he cannot compute all the gates of C on its own – this is precisely the work we are trying to offload to the prover! As a result, the verifier cannot verify the GKR protocol, since to verify it the verifier needs to compute on its own a random point in the low-degree extension of the input to C'.

Cryptography to the rescue

To overcome this problem we have the prover "commit" to the multilinear extension $\tilde{V}_0$ of the input to C', using a special commitment which is referred to as a "polynomial commitment." Then the prover and verifier run the GKR protocol on the flattened circuit C', at the end at which the verifier needs to check the validity of $\tilde{V}_0(z_0) = v_0$. This is done by the prover "opening" the commitment at the point z_0.

A more detailed description follows.

1. The prover does the following:
   (a) Compute the string V_0 which corresponds to all the wires in $C(x)$.
   (b) Compute $\tilde{V}_0$ which is the multilinear extension of V_0.
   (c) Send to the verifier a succinct "polynomial commitment"of $\tilde{V}_0$

A polynomial commitment is a succinct and binding commitment of $\tilde{V}_0$ that allows the prover to "open" to any evaluation of $\tilde{V}_0$ succinctly.

2. The prover and verifier run the interactive GKR protocol w.r.t. C' on input V_0.

   To verify this protocol the verifier needs to check a claim of the form $\tilde{V}_0(z) = t$ for given $z \in \mathbb{F}^m$ and $t \in \mathbb{F}$.

3. The prover will "open" the polynomial commitment of $\tilde{V}_0$ at the point z.

We will not elaborate on how polynomial commitments are constructed.

Succinct Interactive Proofs for NP

Note that the above protocol gives a succinct interactive proof for NP.

1. First the prover sends a polynomial commitment to the multilinear extension of the NP witness w, which is denoted by $\widetilde{W} : \mathbb{F}^m \to \mathbb{F}$.

2. Run the GKR protocol with respect to the verification circuit that has the input x hardwired, and on input w outputs 1 if and only if w is a valid witness corresponding to x.

   To verify the GKR interactive proof the verifier needs to check the value of $\widetilde{W}$ at a single point $z \in \mathbb{F}^m$.

3. The prover will send an opening to the polynomial commitment at point z.

Succinct Non-Interactive Proofs (SNARGs) for NP

Finally, we note that since the GKR protocol is a public-coin protocol (i.e., a protocol where all the verifier's messages are random bits (corresponding to random field elements), we can eliminate interaction from these protocols by using the Fiat-Shamir paradigm! As a result we are able to convert any NP witness w into a succinct cryptographic witness π, where as opposed to w, a succinct witness π exists even for instances x that are not in the language, but are hard to find (under some hardness assumption on the Fiat-Shamir hash function and assuming the polynomial commitments are indeed binding).

Recall that this paradigm replaces the random messages of the verifier with the hash value of the transcript so far.

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.