COSC 544 Probabilistic Proof Systems 10/11/17

A First Succinct Argument

Lecturer: Justin Thaler

1 Overview of This Lecture

Last lecture we saw an efficient way to turn any computer program into an equivalent instance of the arithmetic circuit satisfiability problem. Roughly, we showed that the problem of checking whether a random access machine M taking at most T steps on an input of size x produces output y can be reduced to a circuit satisfiability instance $(\mathcal{C}, x, y)$, where $\mathcal{C}$ has size close to T and depth close to $O(\log T)$. That is, M outputs y on x if and only if there exists a w such that $\mathcal{C}(x, w) = y$.

This transformation is only useful in the context of interactive proofs and arguments if we can design efficient proof systems for solving instances of circuit satisfiability. Today, we will see our first example of such an argument system, by combining the GKR protocol with a cryptographic primitive called a *polynomial commitment scheme*.

2 A trivial approach to argument systems for circuit satisfiability

A trivial way to use the GKR protocol to solve circuit satisfiability is to have the prover explicitly send to the verifier the witness w satisfying $\mathcal{C}(x, w) = y$, and then running the GKR protocol to check that indeed $\mathcal{C}(x, w) = y$. The problem with this simple approach is that in many settings w can be very large. For example, in the transformation from last lecture, the witness w is supposed to be a transcript of M's entire execution on input x, and hence w has size at least T. This means that in the time that the verifier would take to read the whole witness, the verifier could have run M on x without any help from the prover.

3 Succinct Arguments

If an argument system avoids the above bottleneck of sending the entire witness to the verifier, then it is called *succinct*. Formally, an argument system for circuit satisfiability is succinct if the total communication is sublinear in the size of the witness $|w|$.[1]. Succinctness is important for a variety of reasons:

- Shorter proofs are always better. For example, in applications to block-chains that we will see later in the course, proofs must be stored in the block chain permanently. If proofs are long, it drastically increases the global storage requirements of the block chain.
- In some applications, witnesses are naturally large. For example, consider a hospital that publishes cryptographic hash of a massive database w of patient records, and later wants to prove that it ran a specific analysis on w.
- As mentioned above, known transformations from computer programs to circuit satisfiability produce circuits with very large witnesses.

[1]Some works use succinctness more informally to broadly refer to argument systems with short proofs

- The ability to use very large witnesses also makes it easier to generate circuits with “regular” wiring patterns, which is essential for obtaining argument systems that do not involve an expensive pre-processing phase for the verifier. For example, the ability to use large witnesses makes it easier to generate circuits such that $\widetilde{\text{add}}_i$ and $\widetilde{\text{mult}}_i$ can be evaluated in logarithmic time at any point. When applying to GKR protocol to such circuits, one avoids a pre-processing phase (see Lecture 8 notes).

The next several lectures will describe a variety of approaches to obtaining succinct arguments. This lecture will cover one specific approach.

4 A First Succinct Argument

4.1 The Approach

The approach of this lecture is to “simulate” the trivial application of the GKR protocol to circuit satisfiability described in Section 2, but *without* requiring the prover to explicitly send w to the verifier. We will accomplish this by using a cryptographic primitive called a *polynomial commitment scheme*. The idea of combining the GKR protocol with polynomial commitment schemes to obtain succinct arguments was first put forth in very recent work of Zhang et al. [ZGK+17].

Polynomial Commitment Schemes Roughly speaking, a polynomial commitment scheme allows a prover to commit to a low-degree polynomial $\tilde{w}$ and later reveal $\tilde{w}(r)$ for a point r of the verifier’s choosing. Even though in the commitment phase the prover does *not* send all of w to the verifier, the commitment still effectively binds the prover to a specific w. That is, at a later time, the verifier can ask the prover to reveal $\tilde{w}(r)$ for any desired r of the verifier’s choosing, and the prover is effectively forced to reveal $\tilde{w}(r)$ for a *fixed polynomial $\tilde{w}$ determined at the time of the original commitment*.

Combining Polynomial Commitment Schemes and the GKR Protocol When applying the GKR protocol to check that $\mathcal{C}(x,w) = y$, the verifier does not need to know any information whatsoever about w until the very end of the protocol, when (as explained in Section 4.2.1 below) the verifier merely needs to know $\tilde{w}(r)$ for a randomly chosen input r.

So rather than having the prover send w in full to the verifier as in Section 2, we can have the prover merely send a *commitment* to $\tilde{w}$ at the start of the protocol. The prover and verifier can then happily apply the GKR protocol to the claim that $\mathcal{C}(x,w) = y$, ignoring the commitment entirely until the very end of the protocol. At this point, the verifier needs to know $\tilde{w}(r)$. The verifier can force the prover to reveal this quantity using the commitment protocol.

Because the polynomial commitment scheme *bound* the prover to a fixed low-degree polynomial $\tilde{w}$, the soundness analysis of the argument system is essentially the same as if the prover had sent all of w explicitly to the verifier at the start of the protocol as in Section 2.

4.2 Details

4.2.1 What The GKR Verifier Needs to Know About The Witness

In this Subsection, we justify the assertion from Section 4.1 that the only information the verifier needs about w in order to apply the GKR protocol to check that $\mathcal{C}(x,w) = y$ is $\tilde{w}(r_1,\dots,r_{\log n})$.

Let z denote the concatenation of x and w. Let us assume for simplicity throughout this section that x and w are both of length n, so that each entry of z can be assigned a unique label in $\{0,1\}^{1+\log n}$, with the ith entry of x assigned label $(0,i)$, and the ith entry of w assigned label $(1,i)$.

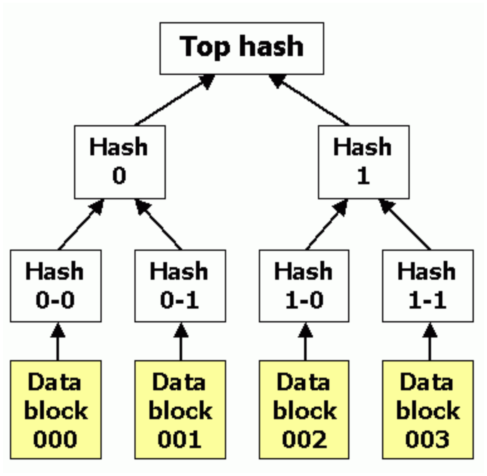

Figure 1: A hash tree. Image source: `http://commons.wikimedia.org/wiki/File:Hash_tree.png`

A key observation is that when applying the GKR protocol to check that $\mathcal{C}(z) = y$, the verifier doesn't need to know the exact value of z. Rather, the verifier only needs to know $\tilde{z}(r_0, \ldots, r_{\log n})$ at a single, randomly chosen input $(r_0, \ldots, r_{\log n})$. Moreover, the verifier doesn't even need to know $\tilde{z}(r)$ until *the very end of the protocol*, after the interactive with the prover has finished. We now explain that in order to calculate $\tilde{z}(r)$, it suffices for the verifier to know $\tilde{w}(r_1, \ldots, r_{\log n})$.

It is straightforward to check that

$$\tilde{z}(r_0, r_1, \ldots, r_{\log n}) = r_0 \cdot \tilde{x}(r_1, \ldots, r_{\log n}) + (1 - r_0) \cdot \tilde{w}(r_1, \ldots, r_{\log n}). \tag{1}$$

Indeed, the left hand side and right hand side are both multilinear polynomials, and they agree whenever $(r_0 \ldots r_{\log n}) \in \{0,1\}^{1+\log n}$; hence, they must be the same polynomial.

Equation (1) implies that, given $\tilde{w}(r_1, \ldots, r_{\log n})$, the verifier can evaluate $\tilde{z}(r_0, \ldots, r_{\log n})$ in $O(n)$ time, since the verifier can evaluate $\tilde{x}(r_1, \ldots, r_{\log n})$ in $O(n)$ time (cf. Lemma 1.8 of Lecture 4).

4.2.2 A First Polynomial Commitment Scheme

There are a number of ways to design polynomial commitment schemes. In this lecture, we describe a simple commitment scheme that is likely folklore, and was recently explicitly proposed by Yael Kalai [Kal17]. This scheme is impractical owing to a large prover runtime, but it provides a clean and simple introduction to cryptographic commitment schemes. We will see more efficient examples of polynomial commitment schemes later in the course.

The scheme makes essential use of two important concepts: Merkle Trees and low-degree tests.

Merkle Trees. The scheme makes essential use of a concept known as a Merkle tree or hash tree [Mer79]. A Merkle tree can be used to design a *string-commitment scheme*, which allows a sender to send a short commitment to a *string* $s \in \Sigma^n$ for any finite alphabet Σ. Later, the sender can efficiently reveal the value of any entries of s that are requested by the receiver.

Specifically, a Merkle tree makes use of a collision-resistant hash function h. The leaves of the treee are hashes of data blocks (i.e., each leaf is a symbol of a string s), and every internal node of the tree is assigned the hash of its two children. Figure 1 provides a visual depiction of a hash tree.

One obtains a string-commitment protocol from a Merkle tree as follows. In the commitment step, the sender commits to the string s by sending the root of the hash-tree (this need be just $O(\log n)$ bits assuming an exponentially hard collision-resistant hash function).

If the sender is later asked to reveal the ith symbol in s, the sender sends the value of the ith leaf in the tree (i.e., s_i), as well as the value of every node v along the root-to-leaf path for s_i, and the sibling of each such node v. We call all of this information the *authentication information* for s_i. The receive checks that the hash of every two siblings sent equals the claimed value of their parent.

Since the tree has depth $O(\log n)$, this is just $O(\log^2 n)$ bits of communication per symbol of s that is revealed.

The scheme is binding in the following sense. For each index i, there is at most one value s_i that the sender can successfully reveal without finding a collision under the hash function h. This is because, if the sender is abel to send valid authentication information for two different values s_i and s_i', then there must be at least one collision under h along the root-to-leaf path connection the root to the ith leaf, since the authentication information for both s_i and s_i' result in the same root hash value, but differ in at least one leaf hash value.

Low-Degree Tests. Suppose a receiver is given oracle access to a giant string s, which is claimed to contain all evaluations of an m-variate function over a finite field $\mathbb{F}$ (note that there are $|\mathbb{F}|^m$ such inputs). A low-degree test allows one determine whether or not the string is consistent with a low-degree polynomial, by looking at only a tiny fraction of symbols within the string.

Unfortunately, because the low-degree test only looks at a tiny fraction of s, it cannot determine whether s is *exactly* consistent with a low-degree polynomial (imagine if s were obtained from a low-degree polynomial p by changing its value on only one input. Then unless the test gets lucky and chooses the input on which s and p disagree, the test has no hope of distinguishing between s and p itself).

What the low-degree test can guarantee, however, is that s is *close* (in Hamming distance) to a low-degree polynomial. That is, if the test passes with probability γ, then there is a low-degree polynomial that agrees with s on close to a γ fraction of points.

Typically, low-degree tests are extremely simple procedures, but they are often very complicated to analyze (and existing analyses often involve very large constants that result in weak guarantees unless the field size is very large). An example of such a low-degree test is the point-versus-line test of Rubinfeld and Sudan, with a tighter analysis subsequently given by by Arora and Sudan [AS03]. In this test, one evaluates s along a randomly chosen line in $\mathbb{F}^m$, and confirms that s restricted to this line is consistent with a univariate polynomial of degree at most m. Clearly, if the string s agrees perfectly with a multilinear then this test will always pass. The works [RS96, AS03] roughly show that if the test passes with probability γ, then there is a low-degree polynomial that agrees with s at close to a γ fraction of points.[2]

A Polynomial Commitment Scheme by Combining Merkle Trees and Low-Degree Tests. Let $\tilde{w}\colon \mathbb{F}^{\log n} \to \mathbb{F}$ be a $(\log n)$-variate multilinear polynomial over $\mathbb{F}$. Let s be the string consisting of $|\mathbb{F}|^{\log n}$ the string consisting of all evaluations of $\tilde{w}$. One obtains a polynomial commitment scheme by applying the Merkle-tree based string commitment scheme of Section 4.2.2, and then applying a low-degree test to s (i.e., if the point-versus-line low-degree test is used, then the receiver picks a random line in $\mathbb{F}^{\log n}$, asks the sender to provide authentication information for all points along the line, and checks that the revealed values are consistent with a univariate polynomial of degree at most $\log n$).

The guarantee of this polynomial commitment scheme is the same as in the string-commitment scheme

[2]More precisely, there is a low-degree polynomial that agrees with s on at least a $\gamma - m^{O(1)}/|\mathbb{F}|^{\Omega(1)}$ fraction of points. This is $\gamma - o(1)$ as long as $|\mathbb{F}|$ is super-polynomially large in m.

of Section 4.2.2, except that the use of the low-degree test ensures that if the sender passes all of the receivers checks with probability γ, then not only is the sender bound to a fixed string s, but there is some low-degree polynomial that agrees with s at close to a γ fraction of points.

This guarantee is enough to use the polynomial commitment scheme in conjunction with the GKR protocol applied to the claim $\mathcal{C}(x,w) = y$, as outlined in Section 4.1. Specifically, if the verifier's checks in the polynomial commitment scheme pass with probability at least (say) $1/2$, then the prover is bound to a string s such that there is a multilinear polynomial p that agrees with s on close to a $1/2$ fraction of points. As long as the point $(r_1,\dots,r_{\log n})$ at which the verifier in the GKR protocol evaluates s is not one of the "bad" points on which s and p disagree, then the soundness analysis of the GKR protocol applies exactly as if the prover were bound to the multilinear polynomial p itself.

This is enough to argue that if the prover passes all of the verifier's checks with probability significantly larger than $1/2$, then indeed there exists a w (namely, the restriction of p to the domain $\{0,1\}^{\log n}$) such that $\mathcal{C}(x,w) = y$. The soundness error can be reduced from roughly $1/2$ to arbitrarily close to 0 by repeating the protocol many times and rejecting if any of the executions ever results in a rejection.

Costs of this succinct argument system. In addition to the communication involved in applying the GKR protocol to check that $\mathcal{C}(x,w) = y$, the argument system above requires additional communication for the prover to commit to $\tilde{w}$ and execute the point-versus-line low-degree test. Assuming an exponentially hard collision-resistant hash function, the total communication cost due to the polynomial commit scheme is $O(|\mathbb{F}| \cdot \log^2 n)$ bits (the cost is dominated by the cost of the prover revealing the value of $\tilde{w}$ on all $|\mathbb{F}|$ points along a line chosen by the verifier). This is $O(n)$ as long as $|\mathbb{F}| \leq n/\log^2 n$ (note that, while in practice we prefer to work over large fields, the soundness error of the GKR protocol is $O(d \log |\mathcal{C}|)$, so working over a field of size $O(n/\log^2 n)$ is enough to ensure soundness in the GKR protocol as long as $d \log |\mathcal{C}| \ll n/\log^2 n$.)

The verifier's runtime is the same as in the GKR protocol, plus the time required to play its part of the polynomial commit scheme. Assuming the collision-resistant hash function h can be evaluated in constant time, and the field size is $O(n/\log^2 n)$, the verifier spends $O(n)$ to execute its part of the polynomial commitment scheme.

The prover's runtime in the above argument system is dominated by the time required to commit to $\tilde{w}$. This requires building a Merkle tree over all possible evaluations of $\tilde{w}$, of which there are $|\mathbb{F}|^{\log n}$. If we work over a field of size (say) $O(n)$, then this runtime is $n^{O(\log n)})$, which is superpolynomial. So, as described, this polynomial commitment scheme is asymptotically efficient for the verifier, but not the prover.

Remark 1. It is possible to reduce the prover's runtime to $O(n^c)$ for some constant c in the above argument system. The way to do this is to tweak the parameters within the GKR protocol to enable working over a much smaller field, of size $O(\log n/\log\log n)$. This will be explained in more detail in a couple of lectures when we talk about designing succinct arguments from multi-prover interactive proofs.

References

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

[Kal17] Yael Kalai. A new perspective on delegating computation. Talk at Workshop on Probabilistically Checkable and Interactive Proofs @ STOC 2017 Theory Fest, 2017.

[Mer79] Ralph Merkle. *Secrecy, authentication, and public key systems*. PhD thesis, Electrical Engineering, Stanford, 1979.

[RS96] Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

[ZGK$^+$17] Yupeng Zhang, Daniel Genkin, Jonathan Katz, Dimitrios Papadopoulos, and Charalampos Papamanthou. vsql: Verifying arbitrary SQL queries over dynamic outsourced databases. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 863–880, 2017.