COSC 544 Probabilistic Proof Systems 9/19/17

LFKN's Interactive Proof for #SAT

Lecturer: Justin Thaler

1 A First Application of Sum-Check: #SAT $\in$ IP

Let ϕ be any Boolean formula on n variables of size $S = \text{poly}(n)$. That is, ϕ is a directed tree of AND, OR, and NOT gates of fan-in 2, with leafs corresponding to inputs $x_1, \ldots, x_n$. Abusing notation, we will use ϕ to refer to both the formula itself and the function on $\{0,1\}^n$ that it computes. In the #SAT problem, the goal is to compute the number of satisfying assignments of ϕ, or equivalently to compute $\sum_{\mathbf{x} \in \{0,1\}^n} \phi(\mathbf{x})$. This is precisely the kind of function that Lund et al. [LFKN92] designed the sum-check protocol to compute.

However, in order to apply the sum-check protocol, we need to identify a polynomial extension g of ϕ of total degree $\text{poly}(S)$ over a suitable finite field $\mathbb{F}$, such that the verifier can evaluate g at a random point $\mathbf{r}$ in polynomial time. We define g as follows.

Let $\mathbb{F}$ be a finite field of size, say, S^4 (to guarantee soundness error $\delta_s \leq 1/3$, the field must be big enough so that $d \cdot v/|\mathbb{F}| \leq \delta_s$). We can turn ϕ into an *arithmetic* circuit ψ over $\mathbb{F}$ that computes the desired extension g of ϕ. An arithmetic circuit $\mathcal{C}$ has input gates, output gates, intermediate gates, and directed wires between them. Each gate computes addition or multiplication over a finite field $\mathbb{F}$.

For any gate in ϕ computing the AND of two inputs y, z, ψ replaces $\text{AND}(y,z)$ with multiplication of y and z over $\mathbb{F}$; it is easy to check that the bivariate polynomial $y \cdot z$ extends the Boolean function $\text{AND}(y,z)$; i.e., $\text{AND}(y,z) = y \cdot z$ for all $y, z \in \{0,1\}$. Likewise, ψ replaces a gate computing $\text{NOT}(y)$ by $1 - y$, and a gate computing $\text{OR}(y,z)$ by $y + z - y \cdot z$. This transformation is depicted in Figures 1 and 2. It is easy to check that $\psi(\mathbf{x}) = \phi(\mathbf{x})$ for all $\mathbf{x} \in \{0,1\}^n$.

For the polynomial g computed by ψ, $\sum_{i=1}^{v} \deg_i(g) \leq S$. Thus, the total communication cost of the sum-check protocol applied to g is $O(S \cdot n)$ field elements, and $\mathcal{V}$ requires $O(S)$ time to check the first $n-1$ messages from $\mathcal{P}$. To check $\mathcal{P}$'s final message, $\mathcal{V}$ must also evaluate $g(\mathbf{r})$ for the random point $\mathbf{r} \in \mathbb{F}^n$ chosen during the sum-check protocol. $\mathcal{V}$ can clearly evaluate $g(\mathbf{r})$ gate-by-gate in time $O(S)$.

As explained in the previous lecture, the prover can runs in time (at most) $2^n \cdot T$, where T is the cost of evaluating g at a point. Since g can be evaluated at g in time $O(S)$, the prover in the #SAT protocols run in time $O(S \cdot 2^n)$. The costs of this protocol are summarized in Table 1.

Proving IP = PSPACE. To establish that **IP** $\subseteq$ **PSPACE**, it suffices to show that, for any interactive proof protocol with communication cost $c(n)$, the verifier's acceptance probability for any given prover strategy $\mathcal{P}$ can be computed in space $\text{poly}(c(n))$, as $x \in \mathcal{L}$ if and only if this acceptance probability is larger $1/3$ for some $\mathcal{P}$. Eliding some details, this acceptance probability for any prover strategy $\mathcal{P}$ can be computed by enumerating over every possible setting of the verifier's random coins and computing the fraction of settings that lead the verifier to accept.

The more challenging direction is to show that **PSPACE** $\subseteq$ **IP**. The #SAT protocol of Lund et al. [LFKN92] described above already contains the main ideas necessary to prove this. Shamir [Sha92] extended the #SAT protocol to solve the **PSPACE**-complete language TQBF, and Shen [She92] gave a simpler proof (the cost of Shamir's and Shen's protocols are similar to those of the #SAT protocol described

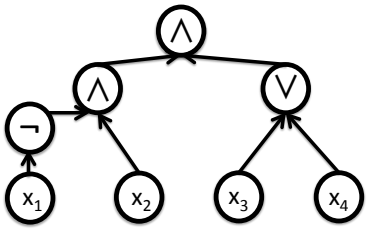

Figure 1: A Boolean formula ϕ.

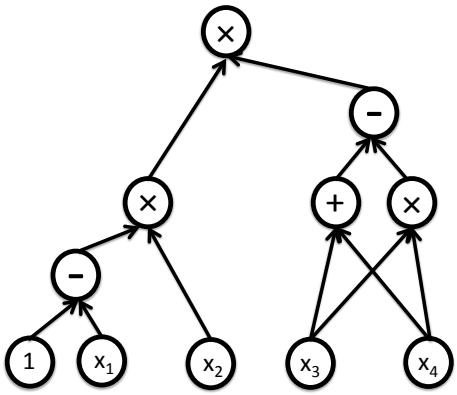

Figure 2: An arithmetic circuit ψ computing a polynomial extension g of ϕ over a finite field $\mathbb{F}$.

Communication	Rounds	$\mathcal{V}$ time	$\mathcal{P}$ time
$O(S)$ field elements	n	$O(S)$	$O(S \cdot 2^n)$

Table 1: Costs of the #SAT protocol of Section 1 when applied to a Boolean formula $\phi : \{0,1\}^n \to \{0,1\}$ of size S.

above). We do not cover Shamir or Shen's extensions here, since later lectures will provide a different and quantitatively stronger proof that **PSPACE** $\subseteq$ **IP** (via the *GKR protocol*).

Open Problem: On The Power of the Prover, or Are Sum-Check Techniques Really Necessary to Solve Languages in coNP? The prover in the protocol for the **PSPACE**-complete problem TQBF can itself be implemented in **PSPACE**. Similarly, the prover in the #**P**-complete problem #SAT protocol can itself be implemented via polynomially many calls to a function in #**P**. However, there is no known interactive for the **coNP**-complete language 3S$\bar{\text{A}}$T in which the prover need not solve #**P**-complete problems. Is there a protocol for 3S$\bar{\text{A}}$T with a prover that can be implemented in, say, $\mathbf{P}^{\mathbf{NP}}$? Under plausible complexity assumptions, $\mathbf{P}^{\mathbf{NP}}$ is powerful enough to approximate the number of satisfying assignments to a factor of $1 \pm 1/\text{poly}(n)$,[1], but is not believed to be powerful enough to exactly count them, as can be done in #**P**. □

References

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39:859–868, October 1992.

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39:869–877, October 1992.

[She92] A. Shen. IP = PSPACE: Simplified Proof. *J. ACM*, 39:878–880, October 1992.

[1] See e.g. `http://mathoverflow.net/questions/2218/characterize-pnp`