1 Interactive Verification of co-NP Statements

19 March 2003.

In this lecture we shall see the LFKN [6] protocol for verifying co-NP statements. The algebraic techniques used by [6] play a central role in the proof of the PCP theorem. The setting for LFKN's protocol is somewhat different from that of the PCP theorem. This is the setting of *interactive proofs* that predated the notion of PCP's, and were viewed as more natural and useful at the time (partly due to the notion of *zero knowledge*, which is beyond the scope of this course).

1.1 Interactive proofs

In an *interactive proof* a computationally unlimited prover P needs to convince a random polynomial time verifier V that a common input x is in some language L. The setting is similar to a PCP, except that P is *adaptive*, and at every point in the protocol may base his next message on the complete history of the protocol, rather than only on the last message sent by V. Likewise, V too is adaptive. Interactive proofs were introduced independently in [4, 3] (there are some differences between the definitions of these two papers), and play an important role in cryptography (beyond the scope of this course).

Definition 1 *An* interactive proof *for a language L is a two player protocol involving a computationally unlimited prover P and a random polynomial time verifier V. Both share a common input x, and can exchange messages of polynomial length. The protocol needs to satisfy two properties:*

- Completeness. *If $x \in L$, then the prover has a strategy (a function from the history of messages so far to the next message that the prover sends) that causes the verifier to accept.*
- Soundness. *If $x \notin L$, then regardless of the prover's strategy, the verifier accepts with probability at most 1/2 (probability taken over the coin tosses of V).*

We shall describe the LFKN interactive proof for every language in co-NP. The original version of the protocol involved special properties of the *permanent* function (see [10] regarding the definition and the complexity of this important function), and we present here a later version of the protocol that more easily extends to later PCPs that we will present.

We remark that Shamir [8] extended the results of LFKN and designed interactive proofs for every language in PSPACE. However, we choose not to present the Shamir protocol as its additional complication are not needed for our later PCP constructions.

An informative and entertaining (though possibly biased) presentation of the exciting research activities in the end of 1989, beginning of 1990, appears in [2].

1.2 Soundness not bounded away from 1

It suffices to design an interactive proof system for a co-NP-complete language, and then by polynomial time reductions we have interactive proofs for all co-NP languages. The complement of any NP-complete language is complete for co-NP, and we shall construct an interactive proof for co-3SAT.

How does P convince V that a 3CNF formula ϕ has no satisfying assignment? As is always the case with interactive proofs and PCPs, the problem is with the soundness condition: getting low acceptance probability when $x \notin L$. If we just want the probability of acceptance to be smaller than 1 in this case, then a simple protocol in which V selects a random assignment and accepts unless it is satisfying would do.

One key ingredient in the LFKN protocol is a generalization of the question. Rather than prove that ϕ has 0 satisfying assignments, the protocol allows P to announce the number of satisfying assignments that ϕ has, and then convince V that this is indeed the case.

Remark: Interactive proofs trivially have the power of NP. To use NP machinery (the prover) in order to also get co-NP, the problem of checking whether there are 0 satisfying assignments is generalized to its counting version: determining the exact number of satisfying assignments even when it is not 0. This is somewhat analogous to the proof that nondeterministic space is closed under complement (e.g., that nondeterministic log-space machines can recognize languages in co-NLOGSPACE), which was proved by the method of inductive counting [9, 5] (which counts the number of paths between two vertices in a directed graph rather than just determine that there is no path). However, the original LFKN protocol was not designed with this analogy in mind, but rather was inspired by work on the permanent function.

As an introduction to the LFKN protocol, let us present a simple protocol by which a truthful P always convinces V to accept, whereas for every cheating P (initially claiming a wrong number of satisfying assignments), V rejects with some nonzero probability (at least 2^{-n}).

Input to the protocol: a 3CNF formula ϕ with n variables $x_1, \ldots, x_n$ and m clauses, and and a number s (the claimed number of satisfying assignments). The protocol eliminates variable by variable from the formula as explained below.

1. Let j be the index of the variable with the lowest index remaining in the formula, let ϕ_j be the formula, and let s_j be the claimed number of satisfying assignments for ϕ_j. (Initially, $j = 1$, $\phi_j = \phi$ and $s_j = s$.) If $j = n$ then count the number of satisfying assignments for ϕ_2 (which is either 0,1 or 2), accept if equal to s_n, and reject otherwise.

2. If $j < n$, then P sends s_j^0 and s_j^1, the claimed number of assignments that satisfy ϕ_j when $x_j = 0$ and $x_j = 1$ respectively.

3. V checks that $s_j^0 + s_j^1 = s_j$, and rejects otherwise.

4. V chooses at random a value $b \in \{0, 1\}$, sets $x_j = b$, simplifies ϕ_j (by removing satisfied clauses and unsatisfied literals), and obtains a new formula ϕ_{j+1}. V sets $s_{j+1} = s_j^b$.

5. V sends b to P, and P is now required to prove that ϕ_{j+1} has s_{j+1} satisfying assignments.

It is not hard to see that a truthful prover will always be accepted, and a cheating prover has probability at least 1/2 of being forced to continue cheating at every round, and hence is rejected with probability at least 2^{-n+1}.

1.3 Arithmetization

Let ϕ be a 3-SAT formula with n variables and m clauses. That is, $\phi = C_1 \wedge C_2 \wedge ... \wedge C_m$, and for each $1 \leq j \leq m$, $C_j = (\ell_{j1} \vee \ell_{j2} \vee \ell_{j3})$, where each ℓ_{jk} is a literal (either x_i, or $\bar{x}_i$, for some variable x_i).

We "arithmetize" each clause in the following way. Example: $C = (x_1 \wedge \bar{x}_2 \wedge x_3)$. Then

$$\tilde{C} = 1 - (1 - x_1)x_2(1 - x_3)$$

Observe that for every 0/1 assignment to the variables, $\tilde{C} = 1$ if the clause is satisfied, and $\tilde{C} = 0$ otherwise. We arithmetize ϕ as $\tilde{\phi} = \prod_{j=1}^{m} \tilde{C}_j$. Let A be a 0/1 assignment to the variables x_1, ... ,x_n. Then $\tilde{\phi}_A = 1$ if A is a satisfying assigment, and $\tilde{\phi}_A = 0$ if A is not a satisfying assigment. Thus the expression $\Phi = \sum_{A \in \{0,1\}^n} \tilde{\phi}_A$ evaluates to the total number of satisfying assignments to ϕ.

Observe that $\tilde{\phi}$ is a polynomial in the variables $\{x_i\}$. Its degree in every variable is at most m. In fact, over 0/1 assignments, this multinomial is equivalent to a multilinear function: the degree of a variable need never be larger than 1, because every term of the form x_i^2 can be replaced by x_i. However, explicitly writing $\tilde{\phi}$ as a multilinear expression involves exponentially many terms, and is beyond the capabilities of the verifier V.

1.4 Finite fields

In the interactive proof, all computations are performed modulo p, where p is a prime to be agreed upon by V and P. This is done so that the numbers involved in the protocol never grow beyond p, saving in computation and communication rsources.

The numbers $\{0, 1, 2, ..., p-1\}$ form a *finite field*, where the four operations +, -, *, /, can be performed (except division by 0). E.g., for computations mod 5, the finite field contains the elements $\{0, 1, 2, 3, 4\}$, and examples of computations are $4 + 2 = 1$, $1 - 2 = 4$, $4 * 2 = 3$, $3/2 = 4$. An important fact that we shall use is that the values of any two different polynomials of degree d agree on at most d arguments. (E.g., $x^2 - 2x = 5x^2 + 7 \pmod p$ has at most two solutions.)

1.5 The proof system

Common input ϕ. P claims that there are exactly k assignments satisfying ϕ. In the protocol, P tries to convince V that $\Phi = k$.

P selects a prime $2^n < p < 2^{n+1}$ and proves to V that p is prime (testing primality is in NP, see [7]). As the number of satisfying assignment to a formula is never more than 2^n, reducing this number modulo p does not change its value.

Eliminating a single variable:

Consider the polynomial $\Phi_1(x) = \sum_{A \in x \times \{0,1\}^{n-1}} \tilde{\phi}_A$. This is a polynomial in one variable x (that corresponds to the original x_1). It is of degree at most m.

P sends a polynomial $\Phi_1'(x)$. (Truthful prover sends $\Phi_1(x)$.)

V checks that $\Phi_1'(x)$ is of degree at most m, and that $\Phi_1'(0) + \Phi_1'(1) = k$.

V selects $r_1 \in Z_p$ at random and computes $k_1 = \Phi_1'(r_1)$.

P is requested to convince V that $\Phi_1(r_1) = k_1$.

The procedure of eliminating single variables continues, until at the end V is left with checking that $\tilde{\phi}_{r_1, r_2, \ldots, r_n} = k_n$, which V can do by direct substitution.

1.6 Correctness

A truthful prover need never lie (k is the true number of satisfying assignments, and $\Phi_i'(x) \neq \Phi_i(x)$ for every i), and hence V evetually accept.

We sketch only the case of a cheating prover. If $\Phi \neq k$, then necessarily $\Phi_1'(x) \neq \Phi_1(x)$ (as polynomials). Since both polynomials are of degree at most m, it follows that $\Phi_1'(r_1) = \Phi_1(r_1)$ for at most m possible choices of r_1. Thus if P lies about the value of Φ, then with probability at least $1 - m/p$ he is forced to lie about the value of $\Phi_1(r_1)$. A similar argument shows that in each round of the protocol P has probability at most m/p of escaping from the chain of lies. Since V checks $\Phi_n(r_n)$ $(= \tilde{\phi}_{r_1,r_2,\ldots,r_n})$ by himself, P cannot lie about its value being k_n. Hence V accepts with probability at most $nm/p < nm/2^n$.

1.7 Improved efficiency

All computations in the LFKN protocol are done modulo a large prime $p > 2^n$. However, it suffices in fact to perform the protocol modulo a much smaller prime q, and save both in the computational effort and in the communication complexity. Specifically, let s be the true number of satisfying assignments, and let k be the claimed number. If q does not divide $s - k$, then the acceptance probability is at most nm/q.

How does one choose a prime q that does not divide $s - k$, especially as only k is known but k is unknown. Certainly, the prover P cannot be trusted with this task. Hence the idea is to let V choose q, and do so at random from among the first $4n$ primes that are larger than $4mn$. (This exact range is given for concreteness, though once he principles are understood, other ranges can be chosen instead.) The point is that unless $s = k$, then regardless of the value of $s \leq 2^n$, at most n primes divide $s - k$. As V chooses at random one of $4n$ primes, the probability of choosing a bad prime is at most $1/4$. Given that q is not bad, P is accepteded with probability at most $nm/q < 1/4$, so altogether the acceptance probability is at most $1/2$.

Note that the first $4n$ primes larger than $4mn$ can be listed in polynomial time, and hence choosing a random prime in the desired range can be done in random polynomial time by V.

Remark: More generally, one can choose a random prime in the range $[N, 2N]$ in expected time polynomial in $\log N$. One samples at random numbers in this range, and test them for primality. The bound on the expected running time follows from the density of primes (behaves like $O(1/\log N)$, and the fact that testing primality is in P [1].

1.8 Research questions

Power of the prover: Is there a proof system for co-NP in which the truthful prover does not need to do computations which are more difficult than deciding satisfiablity?

Constant round interactive proofs: Is there an interactive proof for co-NP with a constant number of rounds?

References

[1] M. Agrawal, N. Kayal, N. Saxena. "PRIMES is in P". *Manuscript*, 2002.

[2] L. Babai. "E-mail and the unexpected power of interaction". *Proc. 5th Structure in Complexity Theory Conference*, 30–44, 1990.

[3] L. Babai, S. Moran. "Arthur-Merlin games: a randomized proof system and a hierarchy of complexity classes". *JCSS, Vol. 36, No. 2, pp. 254-276, 1988.*

[4] S. Goldwasser, S. Micali, C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems". *SIAM J. Comput. Vol. 18, No. 1, pp. 186-208, February 1989.*

[5] N. Immerman. "Nondeterministic space is closed under complementation". *SIAM J. Computing,* 17, 935–938, 1988.

[6] C. Lund, L. Fortnow, H. Karloff, N. Nisan. "Algebraic methods for interactive proof systems". *JACM, 39 (1992), 859-868.*

[7] V. Pratt. "Every prime has a succinct certificate". *SIAM J. Comput. 4 (1975), 214-220.*

[8] A. Shamir. "IP=PSPACE". *JACM, 39 (1992), 869-877.*

[9] R. Szelepcsenyi. "The method of forcing for nondeterministic automata". *Bull. of the EATCS*, 33, 96–100, 1987.

[10] L. Valiant. "The complexity of computing the permanent". *Theoretical Computer Science 8, 1979, 189-201.*