CS 6743 **Lecture 19** [1] **Fall 2007**

1 $\#SAT \in \mathsf{IP}$

Define #3SAT to be a function that maps a 3-CNF $\phi(x_1,\dots,x_n)$ into a number s that is equal to the number of satisfying assignments of ϕ. We'll show the following

Theorem 1 (Lund, Fortnow, Karloff, Nisan). *#3SAT is in* IP.

Remark Here, by IP we mean IP[poly], i.e., an interactive protocol with a polynomial number of rounds.

The key ingredient in the proof of both results mentioned above is *arithmetization* of propositional formulas. Namely, the conversion of a given 3-CNF $\phi(x_1,\dots,x_n)$ into a an arithmetic formula computing a multivariate polynomial $f(x_1,\dots,x_n)$ satisfying the following property. For any truth assignment $a=(a_1,\dots,a_n)$ (which we view as a 0-1 vector), if $\phi(a)$ is True, then $f(a)=1$; and if $\phi(a)$ is False, then $f(a)=0$.

Such an arithmetization of a formula ϕ is carried out inductively. A variable x becomes the function x, and the literal $\bar{x}$ becomes the function $1-x$. A formula $\phi_1\wedge\phi_2$ becomes the function $f_1 * f_2$, where f_i is an arithmetization of ϕ_i, $i=1,2$. Finally, a formula $\phi_1\vee\phi_2$ becomes the function $1-(1-f_1)(1-f_2)$, where f_i is the arithmetization of ϕ_i, $i=1,2$. (To make sense of the last rule, recall that by de Morgan's rule, $\phi\vee\psi\equiv\neg(\neg\phi\wedge\neg\psi.)$

Example: Let $\phi(x_1,x_2,x_3,x_4)=(x_1\vee x_2\vee\bar{x_3})\wedge(\bar{x_1}\vee\bar{x_3}\vee x_4)$. Then the corresponding arithmetic formula will be $f(x_1,x_2,x_3,x_4)=[1-(1-x_1)(1-x_2)x_3][1-x_1x_3(1-x_4)]$, which is a polynomial of degree 6.

Note that the degree of the constructed polynomial is 3 times the number of clauses in ϕ. This is not a coincidence. It is easy to see that any 3-CNF ϕ with m clauses is transformed by the arithmetization procedure described above into a polynomial of total degree at most $3m$.

Lemma 2. *Let $\phi(x_1,\dots,x_n)$ be a 3-CNF with m clauses, and let $f(x_1,\dots,x_n)$ is the arithmetic formula obtained by arithmetizing ϕ. Then (1) the total degree of f is at most $3m$, (2) on any 0-1 vector a, $f(a)\in\{0,1\}$, and (3) on any 0-1 vector a, $f(a)=1$ iff $\phi(a)$ is True.*

Proof. Exercise. (Hint: use induction.) □

For a 3-CNF ϕ and the corresponding arithmetization f, let's define $\#\phi=$ (# satisfying assignments of ϕ), and $\#f=\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n=0}^{1}f(x_1,x_2,\dots,x_n)$. Then, by the lemma above, we get that $\#\phi=\#f$. So, proving that $\#\phi=s$ is equivalent to proving that $\#f=s$. The latter is what our IP protocol is going to do.

Recall that we take a 3-CNF ϕ and arithmetize it, obtaining a polynomial f. We get $\#\phi=$ (# satisfying assignments of ϕ) is the same as $\#f=\sum_{x_1=0}^{1}\sum_{x_2=0}^{1}\cdots\sum_{x_n=0}^{1}f(x_1,x_2,\dots,x_n)$. So we need to design an IP protocol for proving that $\#f=s$.

[1]This lecture is based on the notes by Valentine Kabanets

Note: $0 \leq \#\phi(x_1, \ldots, x_n) \leq 2^n$. So, if we take a prime $p > 2^n$, then $\#\phi(\mod p) = \#\phi$. It will be convenient for the Verifier to get such a prime from the Prover, and do all the checking $\mod p$, i.e., over the finite field $\mathbb{Z}_p$.

The Prover will claim that

$$\sum_{x_1=0}^{1} \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(x_1, x_2, \ldots, x_n) = s, \tag{1}$$

for some number $0 \leq s \leq 2^n$. The verifier's strategy will depend on the following way of "removing" the summation signs from the above arithmetic expression.

Define

$$f_1(z) = \sum_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots \sum_{x_n=0}^{1} f(z, x_2, x_3, \ldots, x_n).$$

Note that $f_1(z)$ is a univariate polynomial in variable z of degree at most that of f, i.e., at most $3m$. Clearly, equality (1) is true iff

$$f_1(0) + f_1(1) = s. \tag{2}$$

But how can we compute $f_1(z)$? It is very hard. The polytime verifier will not be able to do it (unless $\mathsf{NP} = \mathsf{P}$ or something even more dramatic happens). However, the Prover can compute $f_1(z)$. So, the Verifier can just ask the Prover to send the coefficients of $f_1(z)$. Since f_1 has degree at most $3m$, the Prover needs to send at most $3m + 1$ coefficients over $\mathbb{Z}_p$, which is small enough amount of information that the polytime Verifier can handle.

If the Prover is honest, then he'll send a correct s and f_1 so that equality (2) holds. What about a dishonest Prover? Suppose that the Prover sends a wrong s, i.e., equality (1) is false. Then the prover sends $g_1(z)$ (of degree at most $3m$), and claims that $g_1(z) \equiv f_1(z)$. Suppose that the Verifier checks first that $g_1(0) + g_1(1) = s$. If the check does not pass, the Verifier rejects. Either the polynomial g_1 sent by the Prover does not pass this check and the Verifier rejects, or $g_1(0) + g_1(1) = s$. In the latter case, since s is a wrong number, it must be the case that $g_1(z) \not\equiv f_1(z)$ (because $f_1(0) + f_1(1) \neq s$).

So, if the cheating Prover passes the Verifier's check, then it must be the case that $g_1(z)$ and $f_1(z)$ are different polynomials. Since both polynomials are of degree at most $3m$, if the Verifier picks a random value $r_1 \in \mathbb{Z}_p$, then with high probability (at least $1 - (3m)/p$), $g_1(r_1) \neq f_1(r_1)$.

What is $f_1(r)$? Note that $f_1(r) = \sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(r, x_2, \ldots, x_n)$. The expression on the right hand side is of the same type as the initial equality in (1), except with one less summation sign. By setting $s_1 = g_1(r_1)$, the Verifier reduces the original question about equality (1) to the new question about the equality with fewer summations:

$$\sum_{x_2=0}^{1} \cdots \sum_{x_n=0}^{1} f(r, x_2, \ldots, x_n) = s_1. \tag{3}$$

By what we argued above, the cheating Prover will have to give a wrong polynomial $g_1(z)$ so that, with high probability, equality (3) is wrong.

Now the Prover and the Verifier can engage in the same protocol as before, but for a smaller instance - equality (3). After n rounds of communication, the Prover has sent to the Verifier a last polynomial $g_n(z)$ which is supposed to be equal to $f(r_1, r_2, \ldots, r_{n-1}, z)$, for $r_1, \ldots, r_{n-1}$ chosen by the Verifier in the preceding rounds of the protocol. The Verifier again checks if $g_n(0) + g_n(1) = s_{n-1}$, for the value s_{n-1} determined by the Verifier in the previous round. If the check does not pass, the Verifier rejects. Otherwise, the Verifier checks if $g_n(z) \equiv f(r_1, \ldots, r_{n-1}, z)$ by testing if the two polynomials agree on $3m+1$ distinct elements of $\mathbb{Z}_p$. If they disagree, the Verifier rejects. Otherwise, the Verifier accepts.

Analysis First, note that an honest Prover, by answering truthfully to all challenges, will convince the Verifier to accept with probability 1. Now, suppose that a Prover is dishonest, and gives a wrong value of s to the Verifier. Then, either the Prover does not pass one of the Verifier's checks of the form $g_i(0) + g_i(1) = s_{i-1}$ in round i, $1 \le i \le n$, or all such checks are passed by the Prover. In the former case, the Verifier will reject, correctly. In the latter case, if the Prover cheated in round $i-1$, he'll be forced to cheat in round i, with probability at least $1 - (3m)/p$. So the probability that equality of type (1) holds in any of the n rounds is at most $n(3m)/p$, exponentially small. Thus, with high probability, in the last round of the protocol, $g_n(z) \not\equiv f(r_1, \ldots, r_{n-1}, z)$, and this will be discovered by the Verifier. So, if the Prover cheats, then with high probability, the Verifier will reject.

Remark Note that the Verifier in the described protocol just sends random strings to the Prover. So, the described protocol is actually of the Arthur-Merlin type.

2 Generalization to the IP = PSPACE

Since TQBF is a complete problem for PSPACE, it suffices to give an IP protocol for TQBF. A QBF ϕ can be of the form $\forall x_1 \forall x_2 \exists x_3 \ldots \phi(x_1, x_2, \ldots, x_n)$. As before, we can arithmetize the formula ϕ, obtaining a multivariate polynomial $f(x_1, \ldots, x_n)$ that agrees with ϕ over any Boolean n-bit input. By induction, it's easy to show that the QBF ϕ is True iff $\prod_{x_1=0}^{1} \prod_{x_2=0}^{1} \sum_{x_3=0}^{1} \cdots f(x_1, x_2, \ldots, x_n) > 0$. That is, we replace each $\forall x_i$ quantifier with $\prod_{x_i=0}^{1}$, and each $\exists x_j$ with $\sum_{x_j=0}^{1}$. Let's call the resulting arithmetic expression A.

As in the #3SAT protocol in the previous section, we could try to remove the $\sum$'s and $\prod$'s, one by one, and doing the checks of the form $g_i(0) + g_i(1) = s_{i-1}$ or $g_i(0) * g_i(1) = s_{i-1}$, respectively.

There are several problems with this approach. First, the value of A can be as big as 2^{2^n}, since for the formula with n universal quantifiers, we need to multiply together $f(a_1, \ldots, a_n)$, for all 2^n possible binary vectors $(a_1, \ldots, a_n)$. To deal with this problem, we can use modular arithmetic (as in the case of Polynomial Identity Testing for Arithmetic Circuits). So, we pick a random prime p of about $\mathsf{poly}(n)$ bit-size, and do all our verification $\mod p$.

Another problem is the following. Suppose we "remove" the left-most $\sum x_1$ (or $\prod x_1$), and consider the univariate polynomial $f_1(z)$ equal to the original expression A where x_1 is replaced by z and the quantification over x_1 is dropped. This univariate polynomial in z can have degree as large as 2^{n-1}, since there may be $n-1$ $\prod$'s between the quantified x_1 and the occurrence of x_1 in the QBF formula ϕ, and each such $\prod$ doubles the degree of z in the polynomial $f_1(z)$. So, a polytime Verifier cannot ask the Prover for the coefficients of $f_1(z)$ — there are just too many of them!

This is a much more serious problem than the first. Upon closer study, one observes that this problem is due to the fact that a QBF may have an unbounded number of $\forall$-quantifiers between a quantifier for a variable x and the occurrence of x in the formula. If we could somehow transform any given QBF so that between any quantifier for x and the occurrence of x in the formula there is at most one universal quantifier, we would be able to argue that the degree of polynomial $f_1(z)$ is at most $2*$(the degree of $f(x_1, \ldots, x_n)$), which is small enough for the Verifier to be able to receive the coefficients of $f_1(z)$.

It turns out that such a transformation is indeed possible! Here's how. For every occurrence of the situation $Qx \ldots \forall y \ldots x$, we introduce a new "place-holder" variable x' for x, and write the equivalent QBF $Qx \ldots \exists x'((x' \leftrightarrow x) \wedge \forall y \ldots x')$. Note that after this transformation, there is one universal quantifier between $\exists x'$ and the occurrence of x' in the formula. At the same time, the number of universal quantifiers between Qx and the occurrence of x in the new formula is decreased by 1. So, continuing in the same way, we can convert any given QBF to a QBF of the desired form, where there is at most one universal quantifier between any Qx and the occurrence of x in the formula. Observe that this transformation results in a QBF which may not be in prenex form with all the quantifiers in the front of the formula, but it's OK for our purposes.

So, after doing the aforementioned transformation of a given QBF, and doing all verification modulo a large enough prime, we can design an IP protocol for TQBF which is very similar to the one for #3SAT described earlier. This proves that $\mathsf{PSPACE} \subseteq \mathsf{IP}$. The other inclusion $\mathsf{IP} \subseteq \mathsf{PSPACE}$ is fairly straightforward: in PSPACE we can compute the probability that a Verifier accepts a given input; the details are left as an exercise.

3 AM and MA

Now we will briefly look at two restricted versions of interactive protocols. Instead of allowing a polynomial number of rounds, there are just two rounds and the prover can see verifier's randomness. Call the class where the Prover starts the interaction MA, and where the Verifier starts AM. In these names "A" stands for "Arthur" and "M" for Merlin. Merlin is the prover: the all-powerful wizard, and Arthur is the king Arthur who has limited power, but does not want to be fooled by Merlin. The class MA can be viewed as a version of NP in which verification is randomized.

Theorem 3. *$MA \subseteq AM$.*

We skip the proof.