6.966 Algebra and Computation	November 23, 1998
Lecture 21	
Lecturer: Madhu Sudan	*Scribe: Venkatesan Guruswami*

Our goal in this lecture is to prove the following theorem:

Theorem 1 *For any two algebraically closed fields K, L of characteristic zero, we have*

$$\mathrm{NP}_K = \mathrm{P}_K \Longleftrightarrow \mathrm{NP}_L = \mathrm{P}_L.$$

Note that the above theorem is important as it lends a certain robustness to the model of algebraic computation we have been considering. Indeed, the model would not be very attractive at all if the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question had one answer for one field, but the opposite one for another (similar) field.

We will also observe, as an aside, in course of the proof that

Theorem 2 $0/1-\mathrm{P}_{\mathcal{C}} \subseteq \mathrm{BPP}$.

Theorem 1 will be proved through the following two lemmas

Lemma 3 *If K, L are algebraically closed fields with $K \subset L$, then $\mathrm{NP}_K = \mathrm{P}_K$ implies $\mathrm{NP}_L = \mathrm{NP}_L$.*

Lemma 4 *If K is any algebraically closed field of characteristic zero, then $\mathrm{NP}_K = \mathrm{P}_K$ implies $\mathrm{NP}_{\bar{\mathrm{Q}}} = \mathrm{P}_{\bar{\mathrm{Q}}}$ where $\bar{\mathrm{Q}}$ is the algebraic closure of the field of rational numbers.*

First observe that given the above two lemmas, the proof of Theorem 1 is in fact straightforward.

Proof of Theorem 1: Let K, L be any two algebraically closed fields of characteristic zero. Suppose $\mathrm{NP}_K = \mathrm{P}_K$. Then, by Lemma 4, $\mathrm{NP}_{\bar{\mathrm{Q}}} = \mathrm{P}_{\bar{\mathrm{Q}}}$. Now, since L is of characteristic zero, $\mathrm{Q} \subseteq L$, and this means $\bar{\mathrm{Q}} \subseteq L$ as L is algebraically closed. Using Lemma 3 this implies $\mathrm{NP}_L = \mathrm{P}_L$. Similarly, we can prove that $\mathrm{NP}_L = \mathrm{P}_L$ implies $\mathrm{NP}_K = \mathrm{P}_K$. Hence $\mathrm{NP}_K = \mathrm{P}_K \Leftrightarrow \mathrm{NP}_L = \mathrm{P}_L$. $\square$

It therefore suffices to prove Lemmas 3 and 4.

Proof of Lemma 3

Assume $\mathrm{NP}_K = \mathrm{P}_K$. As seen in the previous lecture, HN/K is NP_K-complete, thus this condition is *equivalent* to $\mathrm{HN}/K \in \mathrm{P}_K$. In particular, this implies that there is a polynomial-time K-Turing Machine M that decides HN/K. Our goal is to prove that $\mathrm{HN}/L \in \mathrm{P}_L$, which will imply that $\mathrm{NP}_L = \mathrm{P}_L$. We do this by claiming that the Turing Machine M without any change by itself decides HN/L when fed inputs over the field L. (Note that since $K \subset L$, M is a legal L-Turing Machine as the *constants* it uses will all lie in L.)

Suppose not, so that there is an instance of HN/L on which on which M errs. Note that an instance of HN/L is given by polynomials $f_1, f_2, \ldots, f_m \in L[X_1, X_2, \ldots, X_N]$; we denote by N the number of coefficients in all the f_i's and denote by $\vec{C} \in L^N$ the *explicit* representation of the input in terms of the vector of coefficients. Thus there is an input $\vec{C} \in L^N$ on which the machine errs with respect to its decision on whether or not $\vec{C} \in \mathrm{HN}/L$. Assume, for definiteness, that $\vec{C}$ is an YES instance of HN/L but that the machine M rejects $\vec{C}$ (the other possibility can be handled similarly and will therefore not be discussed).

Consider the operation of M on input $\vec{C}$. By assumption it reaches a rejecting configuration σ, and looking at the computation path leading from the start configuration to σ, we obtain polynomials $r_1, r_2, \dots, r_l \in K[X_1, \dots, X_N]$ and polynomials $s_1, s_2, \dots, s_k \in K[X_1, X_2, \dots, X_N]$ such that any input $\vec{C}$ to M *leads* to the reject configuration σ if

$$r_1(\vec{C}) = r_2(\vec{C}) = \cdots = r_l(\vec{C}) = 0 \text{ and } s_1(\vec{C}) \neq 0, s_2(\vec{C}) \neq 0, \cdots, s_k(\vec{C}) \neq 0. \tag{1}$$

We assumed that $\vec{C}$ was a YES instance of HN$/L$, and we now proceed to characterize YES instances of HN$/L$. Let $f_1, f_2, \dots, f_m \in L[X_1, X_2, \dots, X_n]$ be such that $f_1, f_2, \dots, f_m$ have a common zero; which, by the Weak Nullstellensatz, is equivalent to $\sum_i f_i q_i \neq 1 \ \forall q_1, q_2, \dots, q_n \in L[X_1, \dots, X_N]$. We can express this as a linear system (with unknowns z being the coefficients of the q_i's and the entries of A being polynomials in the coefficients of $f_1, f_2, \dots, f_m$, and therefore polynomials in $\vec{C}$), and our condition is equivalent to this linear system not having a solution. Expressing this as a condition in terms of the ranks of certain sub-matrices of A (as was done in Lecture 16 on Quantifier Elimination) we get polynomials $\phi_1^j, \dots, \phi_m^j, \psi_1^j, \dots, \psi_m^j \in \mathcal{Z}[X_1, \dots, X_N]$ (for a certain number of j's) such that $\vec{C} \in \text{HN}/L$ if and only if

$$\exists j \text{ s.t } \phi_1^j(\vec{C}) = \cdots \phi_m^j(\vec{C}) = 0 \text{ and } \psi_1^j(\vec{C}) \neq 0, \dots, \psi_m^j(\vec{C}) \neq 0. \tag{2}$$

Combining Conditions 1 and 2, and renaming the polynomials for convenience, we obtain that there exist polynomials $g_1, g_2, \dots, g_a, h \in K[X_1, X_2, \dots, X_N]$ such that

$$g_1(\vec{C}) = g_2(\vec{C}) = \cdots = g_a(\vec{C}) = 0 \text{ and } h(\vec{C}) \neq 0. \tag{3}$$

(You might wonder whether there should be multiple polynomials which should all be non-zero at $\vec{C}$ in the above condition, but clearly this condition is equivalent to the single condition that their product is non-zero at $\vec{C}$.)

We are now going to claim the existence of a $\vec{C'} \in K^N$ which also satisfies the above Condition 3. This claim clearly follows from Lemma 5, and implies that $\vec{C'}$ is an YES instance of HN$/K$ but that M ends up in a reject configuration (namely σ) on input $\vec{C'}$, contradicting the hypothesis that M correctly decides HN$/K$. This contradiction proves that M in fact decides HN$/L$ as well.

Lemma 5 *Let $S \subset K[X_1, X_2, \dots, X_N]$, $h \in K[X_1, X_2, \dots, X_N]$ and $V_K = \{\vec{x} \in K^N : f(\vec{x}) = 0 \ \forall f \in S\}$ and $V_L = \{\vec{x} \in L^N : f(\vec{x}) = 0 \ \forall f \in S\}$. Suppose that $\exists \vec{y} \in V_L$ such that $h(\vec{y}) \neq 0$. Then $\exists \vec{z} \in V_K$ such that $h(\vec{z}) \neq 0$.*

Proof: Since $h(\vec{y}) \neq 0$, we have using the strong form of the Nullstellensatz that $h \notin \text{Rad}(\text{Ideal}_L(S))$ (the subscript L indicates that this ideal is considered in the ring $L[X_1, \dots, X_N]$). Hence $h \notin \text{Rad}(\text{Ideal}_K(S))$ as well, which implies, once again by the Nullstellensatz, that $V_K \cap \{\vec{x} : h(\vec{x}) \neq 0\} \neq \emptyset$, as desired. $\square$

The proof of Lemma 3 is thus complete; in fact we have also proved the following:

Corollary 6 *If K, L are algebraically closed fields with $K \subset L$, then* $\text{HN}/K = K^* \cap \text{HN}/L$.

Proof of Lemma 4

By virtue of Corollary 6, we know that if a K-Turing Machine M decides HN$/K$ then it also decides HN$/\bar{\mathrm{Q}}$ correctly. The problem, however, is that the machine M uses constants from K while it is only allowed constants from $\bar{\mathrm{Q}}$.

Recall (from the previous lecture) that M is defined by constants $\alpha_1, \alpha_2, \ldots, \alpha_k \in K$ and polynomials $g_1, g_2, \ldots, g_l \in \mathcal{Z}[Y_1, Y_2, \ldots, Y_k, X_1, X_2, \ldots, X_N]$ (N as before stands for the size of the input $\vec{C} \in \bar{\mathrm{Q}}^N$) that act on $\alpha_1, \alpha_2, \ldots, \alpha_k$ and the cell contents. Note we have assumed that the g_i's are polynomials over the integers, since we can alway achieve the functionality of a polynomial over K by a polynomial over $\mathcal{Z}$ together with some (small number of) additional constants from K. Further we may assume that each of these polynomials g_i is used by M only in a branch condition by collapsing computations between successive branch computations into a single one, so that reaching a particular accept/reject state at the end of M's computation is determined by some subset of the g_i's evaluating to 0 and some subset of the g_i's being non-zero at $\alpha_1, \alpha_2, \ldots, \alpha_k, \vec{C}$ (as in Equation 1).

We now want to simulate the behavior of the machine M by a $\bar{\mathrm{Q}}$-Turing Machine M' which uses constants $\beta_1, \beta_2, \ldots, \beta_k \in \bar{\mathrm{Q}}$ where the β_j's are a function only of $M, \alpha_1, \alpha_2, \ldots, \alpha_k$ and the input $\vec{C}$.

The following lemma says that constants which are algebraic are easy to handle.

Lemma 7 *If M uses constants from $K(\gamma)$ where $f(\gamma) = 0$ for some $f \in K[X]$ and decides $L \subseteq K^*$, then there exists a K-Turing Machine M' that decides L with the same asymptotic run-time.*

Proof: Constants in $K(\gamma)$ can be represented as $a_0 + a_1\gamma + \cdots + a_l\gamma_l$ where $l = \deg(f) - 1$. Now it is easy to see that operations on constants in $K(\gamma)$ can be simulated by operations on K with a poly(l) factor blow-up in run time. $\Box$

In light of the above Lemma, we can assume that every polynomial $P \in \bar{\mathrm{Q}}[Y_1, Y_2, \ldots, Y_k] \setminus \{0\}$ satisfies $P(\alpha_1, \alpha_2, \ldots, \alpha_k) \neq 0$. For $1 \leq i \leq l$, let $g_i^{(\vec{C})} \in \bar{\mathrm{Q}}[Y_1, Y_2, \ldots, Y_k]$ be the polynomial obtained from g_i by fixing the variables $X_1, X_2, \ldots, X_N$ to equal the input $\vec{C} \in \bar{\mathrm{Q}}^N$. Hence by our argument, if $g_i^{(\vec{C})} \not\equiv 0$, then $g_i^{(\vec{C})}(\alpha_1, \alpha_2, \ldots, \alpha_k) \neq 0$. Thus, the effect of using $\alpha_1, \alpha_2, \ldots, \alpha_k \in K$ can be achieved by using any constants $\beta_1, \beta_2, \ldots, \beta_k \in \bar{\mathrm{Q}}$ which satisfy $g_i^{(\vec{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for *every* i such that $g_i^{(\vec{C})} \not\equiv 0$. The goal of the remaining part of the proof, therefore, will be to indicate how such a choice of the β_i's can be made, and this will clearly complete the proof of Lemma 4.

The key observation now is that the degrees of the g_i's (and hence those of the $g_i^{(\vec{C})}$'s) are not very large. Indeed, if M runs for T units of time, then all polynomials it computes and uses are of degree at most 2^T. Now consider picking $\beta_1, \beta_2, \ldots, \beta_k \in_{\mathrm{R}} \{1, 2, \ldots, l \cdot 2^{T^2}\}$ – clearly with high probability this random choice will satisfy $g_i^{(\vec{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for all i such that $g_i^{(\vec{C})} \not\equiv 0$. Since T is polynomial in N, picking the β_i's only takes poly(N) random bits and poly(N) time.

The above argument already gives a BPP algorithm simulating M since the β_i's used above are only *small* integers. In fact, our proof so far yields the containment $0/1 - \mathrm{P}_{\mathcal{C}} \subseteq$ BPP which was claimed in Theorem 2.

It still remains to *derandomize* this simulation of M. The basic idea in doing this is to exploit the power we have of operating on very large integers in a single unit of time (since M' is a $\bar{\mathrm{Q}}$-Turing Machine) by choosing β_i's which are very large.

We first note that not only are the degrees of the g_i's all "small", but also the coefficients of each g_i are at most 2^{2^T}, which implies that the coefficients of $g_i^{(\vec{C})}$'s are at most $(\max\{2, \vec{C}_1, \ldots, \vec{C}_N\})^{4^T}$. We can now finish the argument using the following Lemma, whose proof we defer to the end.

Lemma 8 *If $h \in \mathcal{Z}[Y_1, Y_2, \ldots, Y_k]$ is a* non-zero *polynomial over the integers with total degree at most D and coefficients of absolute value at most A, then for $\beta_1 = D \cdot A^D$, $\beta_2 = D \cdot A^D \cdot \beta_1^{D^2}$, ..., $\beta_k = D \cdot A^D \cdot \beta_{k-1}^{D^2}$, we have $h(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$.*

Using the above lemma with $D = 2^T$ and $A = \max\{2, \vec{C}_1, \ldots, \vec{C}_N\}^{4^T}$, we find $\beta_1, \beta_2, \ldots, \beta_k$ such that $g_i^{(\vec{C})}(\beta_1, \beta_2, \ldots, \beta_k) \neq 0$ for every i such that $g_i^{(\vec{C})} \not\equiv 0$, as was desired.

It has to be still checked that these β_i's can be computed quickly. But this is clear: indeed β_1 can computed in $O(T)$ time, and each β_i can once again be computed in $O(T)$ time from β_{i-1}. Thus the overall time to compute $\beta_1, \beta_2, \ldots, \beta_k$ is $O(kT)$ which is polynomial in N, the size of the original input $\vec{C} \in \bar{\mathrm{Q}}^N$, as both T and k are definitely polynomial in N.

Thus the K-Turing Machine M can be simulated with just a polynomial slow down by the $\bar{\mathrm{Q}}$-Turing Machine M', and the proof of Lemma 4 is complete. □

Proof of Lemma 8: We claim, by induction, that the polynomial $h_i(Y_{i+1}, \ldots, Y_k)$ defined as $h(\beta_1, \beta_2, \ldots, \beta_i, Y_{i+1}, \ldots, Y_k)$ is non-zero and has coefficients at most $A \cdot \beta_i^D$. Clearly this will imply the Lemma, since the constant polynomial $h_k = h(\beta_1, \beta_2, \ldots, \beta_k)$ will be non-zero.

The base case of the induction ($i = 0$) is trivial. For $i \geq 1$, the bound on the coefficient size follows by simple inspection. The fact that h_i is non-zero given h_{i-1} is non-zero reduces to a univariate problem as follows: Express h_{i-1} as a polynomial in $Y_{i+1}, \ldots, Y_k$ with coefficients that are polynomials in Y_i. Since h_{i-1} is non-zero, one of these coefficients is a non-zero polynomial in Y_i, call this polynomial $p(Y_i)$. By induction hypothesis, the coefficients of p are at most $A \cdot \beta_{i-1}^D$. Thus our lemma reduces to proving the claim for the univariate case, namely that $\beta_i = D \cdot A^D \cdot \beta_{i-1}^{D^2}$ is *not* a root of $p(Y_i)$, and this follows from the bound on the size of roots of univariate polynomials over integers proved in Claim 1 of Lecture 10. □