Complexity Theory (236313)

Eyal Kushilevitz

June 24, 1997

Lecture #1: March 3rd 1997

Introduction: The course will deal with several basic notions in complexity theory. In particular we intend to cover the following (not necessarily in this order): space complexity classes; gap theorems and hierarchy theorems; randomized computations; interactive proofs; hardness of approximation results; counting problems; the polynomial hierarchy; and circuit complexity.

We recall the definitions of Turing machines, and the classes DTIME($t(n)$) and NTIME($t(n)$). Then, we define the basic space complexity classes DSPACE($s(n)$) and NSPACE($s(n)$). We consider, for example, the language CON that consists of all triples (G, s, t) where G is a directed graph, and s, t are two vertices in G such that s is connected to t. We show that $CON \in$ NSPACE($\log n$).

We ask the natural question whether allowing more (space) resources allows recognizing more languages, and we proved the following counter-intuitive result that says that this is not always the case.

Borodin's Gap Theorem: Let $g : \mathcal{N} \to \mathcal{N}$ be a computable function such that $g(n) \geq n$ for all n. Then, there exists a computable function $s(n)$ such that DSPACE($s(n)$) = DSPACE($g(s(n))$).

Tutorial: In the tutorial we show, by counting the number of configurations of a space $s(n)$ machine, that for every "reasonable" function s, if a language can be recognized in space $s(n)$ it can be recognized in time $2^{O(s(n))}$. We also show that the definition of space complexity is insensitive to the number of tapes of the Turing machine.

Lecture #2: March 10th 1997

We prove the following theorems regarding space complexity.

Hierarchy Theorem: For every $s_1(n), s_2(n)$ such that s_2 is a space function, $s_1(n) \geq \log n$, and $s_1(n) = o(s_2(n))$ the class DSPACE($s_1(n)$) is strictly contained in the class DSPACE($s_2(n)$).

Savitch Theorem: Let $s(n)$ be a space function such that $s(n) \geq \log n$. Then NSPACE($s(n)$) $\subseteq$ DSPACE($s^2(n)$).

Immerman Theorem: Let $s(n)$ be a space function such that $s(n) \geq \log n$. Then $L \in$ NSPACE($s(n)$) implies $\bar{L} \in$ NSPACE($s(n)$).

Tutorial: In the tutorial we discuss other hierarchy theorems as well as the pumping principle.

Lecture #3: March 17th 1997

We discuss the notion of reductions with respect to various complexity classes. In particular PSPACE completeness is defined (using the standard polynomial-time reductions) and the language TQBF is shown to be PSPACE-complete.

Tutorial: More hierarchy theorems as well as the relations between PSPACE and games.

Lecture #4: March 24th 1997

We define log-space reductions, and completeness in the class NL with respect to these reductions. We proved that CON is NL-complete.
We define Oracle Turing machines (and Turing reductions) and use them to define the polynomial hierarchy. We prove some basic facts about the polynomial hierarchy.

Tutorial: More on the connection between PSPACE and games as well as a pair of oracles A and B such that $\mathrm{P}^A = \mathrm{NP}^A$ but $\mathrm{P}^B \neq \mathrm{NP}^B$.

Lecture #5: March 31st 1997

We concluded our discussion of the polynomial hierarchy by introducing few more simple properties of it.
We started a discussion of randomized computations. We demonstrated the power of randomized algorithms by presenting Schwartz algorithm for deciding whether a multivariate polynomial is identically 0.

Tutorial: We introduced the class operators $\exists$ and $\forall$ and some of their basic properties, and gave the (equivalent) definition of the polynomial hierarchy in terms of these operators.

Lecture #6: April 7th 1997

We define the classes RP (one-sided error), BPP (bounded two-sided error) and PP (unbounded two-sided error) corresponding to polynomial-time, randomized computations. We show for both RP and BPP that the error probability can be made smaller than 2^{-k} while increasing the running time by a factor of $\Theta(k)$. The main result that we proved is:

Theorem: $\mathrm{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$.

Tutorial: Complete languages in the various classes of the polynomial Hierarchy were shown. The class RL was introduced and undirected connectivity was shown to be in this class.

Lecture #7: April 14th 1997

We discussed *counting problems* and introduced the class $\#P$ of all functions f for which there exists a nondeterministic Turing machine that runs in polynomial-time and for every input x the number of accepting paths is exactly $f(x)$. We defined completeness in this class, with respect to Turing reductions. We proved that the problem of counting the number of satisfying assignments of a CNF formula is $\#P$-complete. We also gave (part of) the proof that the problem of computing the permanent of a 0-1 matrix is $\#P$-complete.

Tutorial: It was shown that there exists a *randomized* reduction from SAT to Unique-SAT (deciding whether a CNF formula admits exactly one satisfying assignment), and that there exists a Turing-reduction from Unique-SAT to SAT. Thus, NP is "as hard" as recognizing whether there exists a unique solution.

Lecture #8: May 5th 1997

In this lecture we presented (and gave most of the proof for) Toda's theorem that demonstrates the power of the class $\#P$:

$$\mathrm{PH} \subseteq \mathrm{P}^{\#P}.$$

For the purpose of the proof we defined the operators $\oplus(\mathcal{C})$ and $BP(\mathcal{C})$ (for a class of languages $\mathcal{C}$) and proved some facts about them.

Tutorial: Some facts about the primality problem were given. Then the class ZPP (expected polynomial-time) was defined and it was shown that $\mathrm{ZPP} = \mathrm{RP} \cap co\mathrm{RP}$. The class $\#L$ was defined and it was shown that the function $\#CON$ is complete in $\#L$ with respect to log-space reductions and that $\#L \subseteq FP$.

Lecture #9: May 19th 1997

We defined the notion of interactive-proof systems and the corresponding class of languages IP. We proved that the language *Graph Non-Isomorphism* (GNI) is in IP. We also presented an interactive proof system for the permanent (i.e., the language $\{(A,s)|Per(A)=s\}$). This by Toda's theorem implies $\mathrm{PH} \subseteq \mathrm{IP}$.

Tutorial: We defined a special type of interactive-proof systems called Arthur-Merlin games, and in particular the classes AM and MA. We proved that the language GNI is in AM.

Lecture #10: May 26th 1997

We continued the discussion on the power of interactive proofs. We proved that $\mathrm{PSPACE} \subseteq \mathrm{IP}$. Together with the converse containment (that was given in the tutorial) we get Shamir's theorem that $\mathrm{IP} = \mathrm{PSPACE}$. For the purpose of the proof we presented the method of arithmetization for formula with and without quantifiers. We then presented an interactive proof system for the PSPACE-complete language TQBF which uses this arithmetization.

Tutorial: In addition to proving that $\mathrm{IP} \subseteq \mathrm{PSPACE}$, a direct proof for $co-\mathrm{NP} \subset \mathrm{IP}$ was presented.

Lecture #11: June 2nd 1997

We defined the model of *boolean circuits* and discussed various variants of it. We considered the circuit-value language CVAL and proved that it is P-complete with respect to log-space reductions.
We defined the class P/poly of languages which can be recognized by non-uniform families of polynomial size circuits. We proved that P/poly contains the classes P and BPP, as well as several non-recursive languages. On the other hand, we proved that if $\mathrm{NP} \subseteq \mathrm{P/poly}$ then the polynomial hierarchy collapses (more precisely, $\mathrm{PH} = \Sigma_2^p$). Finally, we defined the notion of *uniform* families of circuits.

Tutorial: We discussed formulae (which are a certain kind of circuits, where the fan-out of each gate is bounded by 1). We showed that the formula-value language FVAL is in DL. We also showed that the minimal depth of a fan-in 2 circuit for f equals (up to constants) to the logarithm of its minimal formula size. In particular, the functions with polynomial-size formulae are exactly those with $O(\log n)$ depth, fan-in 2 circuits (this class will be denoted NC^1).

Lecture #12: June 9th 1997

We defined the (uniform and non-uniform versions of the) classes $\mathrm{NC}^k, \mathrm{AC}^k, \mathrm{NC}, \mathrm{AC}$ and showed some trivial relations between them (e.g., NC = AC). More interestingly we proved that $\mathrm{NL} \subseteq$ log-space-uniform-AC^1 ($\subseteq$ log-space-uniform-NC^2). We mentioned (without a proof) the result that PARITY$\notin \mathrm{AC}^0$ (but is in NC^1).
We defined AC^0 reductions as an appropriate type of reductions for circuits. We discussed the definition and proved that MAJORITY$\leq_{\mathrm{AC}^0}$PARITY. This implies that MAJORITY$\notin \mathrm{AC}^0$.

Tutorial: Cancelled due to the teaching-assistants strike ...

Lecture #13: June 23rd 1997

We introduced the notion of *probabilistically checkable proofs*, and the class $\mathrm{PCP}(r(n), q(n))$ of all languages for which there is a verifier that uses $O(r(n))$ random bits and $O(q(n))$ queries in the verification. We stated the PCP Theorem saying that $\mathrm{NP} = \mathrm{PCP}(\log n, 1)$ and we proved the easy part of it ($\supseteq$).
Then, we showed how the PCP Theorem can be used to derive hardness of approximation results. In particular we proved that for a certain ϵ the MAX3SAT problem does not have a $1+\epsilon$ approximation algorithm (while it has a trivial 8/7 approximation algorithm), and that the MAX-CLIQUE problem does not have a n^ϵ approximation algorithm (for some ϵ).

Tutorial: We defined *Branching Programs* and proved that the languages in NC^1 are exactly those that can be recognized by width-5 polynomial-size Branching Programs.