Lecture 28
ECE 664
Avi Kak

Definition of NP-Complete

- Polynomial transformability (see Lecture 27) between the problems in class NP provides us with a mechanism to partition the set NP into equivalence classes.

- We say that two problems Π_1 and Π_2 belong to the same equivalence class if we can show $\Pi_1 \propto \Pi_2$ and $\Pi_2 \propto \Pi_1$.

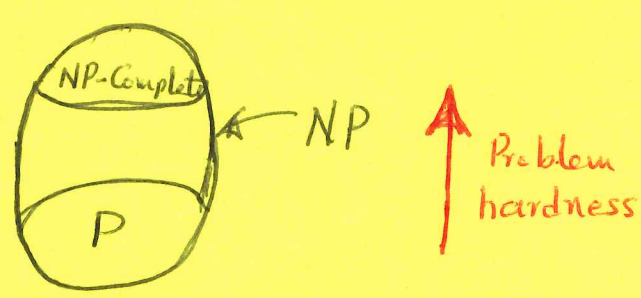
- Recall that when $\Pi_1 \propto \Pi_2$ we consider Π_2 to be at least as hard as Π_1. So when Π_1 and Π_2 are in the same equivalence class, we will associate the same level of hardness with both problems.

- In the partition of the class NP achieved in this manner, the subclass P forms the set of easiest-to-solve problems. [Try to convince yourself that all problems in P can be polynomially transformed to one-another. How about the following two problems: Π_1: Is it true that a given number is divisible by 4? and Π_2: Is it true that a given integer K is the product of two given integers M and N?]

- In the partitioning of the NP class as induced by polynomial transformability, we will refer to the class of the most difficult problems as NP-Complete problems.

- The above definition of NP-Completeness implies that a problem Π is NP-Complete iff $\Pi \in NP$ and that $\Pi' \propto \Pi$ for every problem $\Pi' \in NP$.

- We can therefore visualize class NP as: →



- Later we will show theoretically that there must exist problems that are neither solvable in polynomial time nor NP-complete.

- How do we figure out which problems can be placed in class NP-Complete? Let's assume for a moment that we know of a problem Π_1 that belongs in class NP-Complete. Now suppose we can construct a polynomial transformation from Π_1 to another problem Π_2, then it must be case that Π_2 is also NP-Complete. Recall that $\Pi_1 \propto \Pi_2$ implies Π_2 is at least as hard as Π_1.

- Therefore, in order to use polynomial transformability to populate the NP-Complete class of the most difficult problems in NP, we need a seed problem.

- The honor of being the first NP-complete problem goes to a decision problem in Boolean logic — the problem of SATISFIABILITY (SAT).
- To describe SAT, we need the following:
  1. Let $U = \{u_1, u_2, \ldots, u_m\}$ denote a set of Boolean variables. Obviously, each u_i is either True or False.
  2. A truth assignment for U is a function $t: U \to \{True, False\}$
  3. We will use the notation $\bar{u}_i$ to denote the negation of u_i. We will refer to u_i and $\bar{u}_i$ as the literals drawn from the set U.
  4. A clause over U is a set of literals. For example, the set $\{u_1, \bar{u}_2, u_3\}$ is a clause. A clause, more precisely speaking, is a disjunction of the literals that are in the set.
  5. We say a clause is satisfied by a truth assignment to U if at least one of the literals in the clause is true. (This is obviously a consequence of the fact that the literals in a clause are disjunctive.)
  6. A collection C of clauses over U is satisfiable iff there exists a truth assignment to U that simultaneously satisfies all clauses in C. Such a truth assignment is called the $\boxed{\text{satisfying truth assignment}}$ for C.

EXAMPLE: $U = \{u_1, u_2\}$ and $C = \{\{u_1, \bar{u}_2\}, \{\bar{u}_1, u_2\}\}$. For this C, we have a satisfying truth assignment: $t(u_1) = t(u_2) = True$. But if $C = \{\{u_1, u_2\}, \{u_1, \bar{u}_2\}, \{\bar{u}_1\}\}$, for this C there does <u>not</u> exist a satisfying truth assignment.

- We are now ready to present the definition of the SATISFIABILITY decision problem:

SATISFIABILITY (SAT)	
INSTANCE :	A set U of Boolean variables and a set C of clauses over U.
QUESTION :	Is there a satisfying truth assignment for C?

- Cook's Theorem: SAT is NP-Complete.

  Proof: Our proof must first demonstrate that $SAT \in NP$ and we must then show that for every $\Pi \in NP$ we have $\Pi \propto SAT$.
  - Showing $SAT \in NP$ is trivial since all we need to establish is that a guess for a truth assignment to U can be verified in polynomial time — polynomial in the size of the instance description (which is proportional to the number of clauses times the average number of literals in each clause).
  - However, showing that $\Pi \propto SAT$ for all $\Pi \in NP$ is more challenging. There are obviously infinity of $\Pi \in NP$. However, <u>every</u> $\Pi \in NP$ is solved by some polynomial time NDTM program M. So if can map a generic polynomial time NDTM program to SAT, we will have established $\Pi \propto SAT$ for every $\Pi \in NP$.

So let M denote an arbitrary polynomial time NDTM program M specified by Γ (tape alphabet), Σ (input alphabet), b (for blank symbol, $b \in \Gamma - \Sigma$), Q (the states), q_0 (the start state), q_Y and q_N (halt states) and δ (the transition function). Let $p(n)$ be the polynomial that bounds the Time Complexity Function $T_M(n)$. Let L be the language accepted by M. We must now present a mapping f_L from the NDTM M to SAT so that $x \in L$ iff $f_L(x)$ has a satisfying truth assignment and we must show f_L can be computed in polynomial time. Central to this mapping is the fact that the operation of M can<u>not</u> involve tape squares except for those numbered $-p(n)$ through $p(n)+1$. Furthermore there can be no more than $p(n)$ steps in the checking stage. See the text for the main proof.

Supplemental Material For the Proof of Cook's Theorem:

- As mentioned earlier, crucial to the construction of a polynomial-time mapping from a generic NDTM (along with its operation that results in the acceptance of some language L_M) to SAT is the fact that the NDTM scanhead will visit no squares except those that are numbered $-p(n)$ to $p(n)+1$. Furthermore, there will be no more than $p(n)$ ~~steps~~ time steps involved in the Checking stage of the NDTM.

- In order to construct the mapping f_L, we must define a set U of variables and a set C of clauses over U so that the NDTM program M will accept $x \in L_M$ iff the mapped SAT instance $f_L(x)$ will have a satisfying truth assignment.

- To capture the operation of the ~~NDT~~ NDTM program M in a set of clauses, we define the following three kinds of Boolean variables:

Type 1 Boolean Variables :	$Q[i,k]$	$0 \le i \le p(n)$ $0 \le k \le r$
Meaning :	At time i, the Finite-state control of the NDTM is in state k	
Type 2 Boolean Variables :	$H[i,j]$	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$
Meaning :	At time i, the Read/Write head is looking at the square indexed j	
Type 3 Boolean Variables :	$S[i,j,k]$	$0 \le i \le p(n)$ $-p(n) \le j \le p(n)+1$ $0 \le k \le \lvert T \rvert$
Meaning :	At time i, the content of the tape square indexed j is s_k.	

- The operation of the NDTM induces a truth assignment on the above Boolean variables in an obvious way. We will also assume that if the NDTM halts by entering the states q_Y or q_N before time $p(n)$, the NDTM configuration would remain static until the end of the $p(n)$ time.

- We will next define clauses over the Boolean variables defined above. Different clauses will constrain the NDTM in different ways. For example, some of the clauses will ensure that the NDTM can be in only state at one time. Along the same lines, some other clauses will ensure that the Read/Write head of the finite-state control is looking at only square at one time. We will have other clauses that ensure that the content of a tape square changes only in accordance with the transition function for the underlying NDTM, etc.

- All of the clauses that are needed for the proof of Cook's theorem can be divided into six groups, with each group affecting a particular kind of constraint on the NDTM:

Clause Group G_1: $\{Q[i,0], Q[i,1], \ldots, Q[i,r]\}$ $\quad 0 \le i \le p(n)$

$\sim(A \wedge B) \equiv \tilde{A} \vee \tilde{B}$

$\{\overline{Q[i,j]}, \overline{Q[i,j']}\}$ $\quad 0 \le i \le p(n)$, $0 \le j < j' \le r$

Constraint on NDTM: If all these clauses are simultaneously true, then the NDTM will be allowed to be in <u>only one state</u> at time i.

Clause Group G_2: $\{H[i,-p(n)], H[i,-p(n)+1], \ldots, H[i,p(n)+1]\}$ $\quad 0 \le i \le p(n)$

$\{\overline{H[i,j]}, \overline{H[i,j']}\}$ $\quad 0 \le i \le p(n)$, $-p(n) \le j < j' \le p(n)+1$

Constraint on NDTM: If all these clauses are simultaneously true, the Read/Write head will be allowed to scan <u>only one</u> tape square at any one time.

Clause Group G_3: In the same manner as shown above, simultaneous satisfaction of all clauses in G_3 ensures that at any time i there will be only one symbol in each square.

Clause Group G_4: Simultaneous satisfaction of all G_4 clauses ensures that time $t=0$ we have: ① The tape content consists of the input string in squares indexed 1 through n; ② The R/W head is pointing at square 1; and ③ the finite-state control is in state q_0

Clause Group G_5: $\{Q[p(n),1]\}$

When this clause is true, at time $p(n)$, NDTM is in state 1 which is the index of YES state.

Clause Group G_6: places constraints on the values of the boolean variables at time $i+1$ vis-a-vis their values at time i. These constraints are based on the transition function of the NDTM.

- If you examine these six groups of clauses, it becomes obvious that if the underlying NDTM accepts a string $x \in L_M$, all of the clauses will evaluate to true. The converse is also true. That is, if ~~the clauses are~~ a truth assignment to the Boolean variables makes all the clauses simultaneously true, that would correspond to the underlying NDTM terminating in state q_y on input x.