CSC 2/486

Lane A. Hemaspaandra

Welcome to the course!

Do lion packs (prides) and tennis tournaments have a role in our understanding of the nature of semi-feasible algorithms?

What functions exist? Do one-way functions exist? What is feasibly computable?

In complexity theory, what are NP-complete problems, and can we identify NP-complete sets? Are NP-complete sets NP-hard?

Complete and reducible functions. What is the relationship between these properties?

These slides are not for redistribution.

© Lane A. Hemaspaandra.
COMPUTATIONAL COMPLEXITY!

Yeah!!
Informal Def.

\[ \text{DTIME}[f(n)] \] is the class of languages accepted by a deterministic TM that always runs within \( f(n) \) steps on inputs of length \( N \).

\[ \text{NTIME}[f(n)] \] .... nondeterministic ......

Example

\[ \text{Time}[2^N] \]

Chug... at most \( 2 \times 1 \) steps

\[ x \rightarrow \text{TM} \rightarrow \text{acc. or rej.} \]

Claim \[ \text{Time}[2^N] \subseteq \Xi_0 = \text{recursive sets} \]

Notes:

1. For time, \( f(N) \) is shorthand for \( \max(N+1, \lceil f(N) \rceil) \).
2. "Space, " " " " " " " " " " " " " " " " " " " " " max \( (1, \lceil f(N) \rceil) \)"
$P = \bigcup_{k \in \mathbb{N}^+} \text{Time}[N^k]$

$NP = \bigcup_{k \in \mathbb{N}^+} \text{NTime}[N^k]$  \text{“nondeterministic polynomial time.”}

\text{Nerdy notes: (1) $N^+$ rather than $N$ to avoid 0.}
(2) For $N=0$, recall last page’s Nerdy Notes... or...
\[ P = \text{"polynomial time"} = \bigcup_{i \geq 1} \text{Time}[N^i + i]. \]

\[ NP = \text{"nondeterministic polynomial time"} = \bigcup_{i \geq 1} \text{NTime}[N^i + i]. \]

Note: This alternate definition works even without the "f(N) means \(\max(N+1, \lceil f(N) \rceil)\)" assumed tweak.

- By convention, we say that problems in \(P\) are tractable.

- Gar-Joh's book (6 decades of study) show that many useful problems are in \(NP\). This is the holy grail of theoretical computer science!

\[ \textbf{GOAL} \ P = NP. \]
Claim \( P \subseteq \text{NP} \).

Pf. Immed. \( \square \)

What's to come?

- Parallels between \( \text{NP} \) (PH) and \( \Sigma_1 \) (Kleene hierarchy).
- Reductions, \( \text{NP} \)-complete sets for \( \text{NP} \).
- The polynomial hierarchy
- Cook's (C-Karp-Levin) Theorem
- \( \text{NP} \), \( \text{coNP} \) and the Borodin-Demers Theorem
- Lots of looking into the nature/structure of \( \text{NP} \); for example, can \( \text{NP} \)-complete sets:
  - be sparse?
  - have “small circuits” (equiv., be in \( P^{\text{sparse}} \))?
  - have semi-feasible algorithms (be in \( P^{\text{sel}} \))?
  - ....
Why/how is $\Sigma^p_1$ like $\Sigma^p_1$?

$\Sigma^p_1 : \exists p \text{ rec. predicate}$

$\Sigma^p_1 \equiv \Sigma^p_1$

$\Sigma^p_1 : \exists p \text{ poly-time predicate}$

$\exists p \text{ poly-time predicate}$

$\Sigma^p_1 : \exists p \text{ poly-time predicate}$

Secret siblings! This is like the plot twists on a soap opera! Is there more in this family's closet? Yes....
The Polynomial Hierarchy

\[ \Sigma_0^p = \Pi_0^p = \Delta_0^p = \Delta_1^p = \mathsf{P} \]

\[ \Sigma_i^p = \mathsf{NP} \Sigma_{i-1}^p, \quad i \geq 0. \quad \text{E.g., } \Sigma_1^p = \mathsf{NP} = \mathsf{NP} \quad \text{So } \Sigma_1^p = \mathsf{NP} \]

\[ \Pi_i^p = \text{co} \Sigma_{i+1}^p = \{ \mathcal{L} \mid \mathcal{L} \in \Sigma_{i+1}^p \}, \quad i \geq 0. \]

Examples:

\[ \Sigma_2^p = \mathsf{NP} \Sigma_1^p = \mathsf{NP} \Sigma_1^p = \mathsf{NP} \mathsf{NP} \]

\[ \Pi_1^p = \{ \mathcal{L} \mid \exists \mathcal{L} \in \Pi_1^p \} = \text{coNP} \]

\[ \text{notation (coNP = coNP)} \]

Soooo not a Venn-Euler diagram? Almost Hasse...

Let's turn to \( \mathsf{NP} \)-completeness!

Nerdy note: \( \text{co-NP}^p \cap \mathsf{NP}^p = (\text{coNP})^p \), so \( \Pi_2^p \) is often expressed as \( \text{NP}^p \).
Def. \( L_1 \) is \( \mathsf{NP} \)-complete exactly if:

1. \( L_1 \in \mathsf{NP} \) and
2. \( \forall \hat{L} \in \mathsf{NP} \left[ \hat{L} \leq_m \mathsf{P} \hat{L}_1 \right] \)

**Nondy note:** 2. define “\( \mathsf{NP} \)-hard.” \( \mathsf{P} \) was “\( \mathsf{P} \) in old days

Def. (many-one polynomial-time reduction’s) \( A \leq_{m}^1 B \) iff there exists a polynomial-time computable function \( f \) such that \( \left( \forall x \in \Sigma^* \right) \left[ x \in A \iff f(x) \in B \right] \).

Picture:

Claim Let \( N_1, N_2, N_3, \ldots \) be std. enumeration of \( \mathsf{NPTMs} \). Wlog., let \( N_i \) run within time \( n^i \) on inputs of length \( n \). Then

\[
U = \{ N_i \# x \# 1^k \mid N_i(x) \text{ accepts within } k \text{ steps} \}
\]

is \( \mathsf{NP} \)-complete.

Why? Let \( L \in \mathsf{NP} \). Say \( L = L(N_i) \).

Consider: \( x \in L(N_j) \) \( \implies \) poly-reduction \( N_j \# x \# 1^{1x_j} \in U \)? So \( L \leq_m^1 U \).

Does \( L \leq_m^1 U \) hold even without the \( 1^k \) being in \( U \)'s definition??
\[ SAT = \{ f | f \text{ is a satisfiable Boolean formula} \}. \]

- "true" \[ T \in SAT \]
- "false" \[ F \notin SAT \]
- \( x_i \in SAT \)
- \( \overline{x}_i \notin SAT \)
- \( x_i \land \overline{x}_i \notin SAT \)
- \( x_i \land x_j \in SAT \)

etc.

Looks easy...

.... but it is not (we believe - it is a $1M question!)
Cook's Theorem (a.k.a. the Cook-Karp-Levin Theorem) SAT is NP-complete.

Claim SAT ∈ NP

\[ f = x_1 \land x_2 \]

I remember how acceptance of NPTMs is defined: If any path accepts on the given input, the machine is said to have accepted that input. So this is an ACCEPT!
Cook's Theorem

Pf 1 Hop-Ull

Pf 2 Gar-Joh

Proof's core insight/achievement is:

Given $N_i$ and $x$ (wlog $N_i$ runs in time $n^{i+1}$), we can construct a formula $f(i, x)$ such that

$$x \in L(N_i) \iff f(i, x) \text{ is satisfiable;}$$

and we can perform the formula construction in time polynomial in $|N_i| + 1 \times i^i$. 
Galil (-Cook) observed that one can ensure that $F(i, x)$ displays $N_i$ and $x$ in an obvious fashion. We'll discuss (and use!) Galil's insight soon,... but before we do that, let's spend a good bit of time discussing $\mathbf{NP}$-complete sets, and how to prove sets to be $\mathbf{NP}$-complete.
Some NP-Complete Sets

Clique
\[ \{ (G, k) \mid \text{Graph } G \text{ has a clique of size } k \} \]

\[ \square \triangle \text{ 4-clique } \triangle \text{ 3-clique} \]

Graph Coloring:
\[ \{ (G, k) \mid G \text{ is } k\text{-colorable} \} \]

i.e., there is a way of associating each vertex with the colors 1, 2, ..., k, such that no edge has both its ends touching nodes of the same color.

Example:

\[ \square \text{ is 3-colorable} \]
\[ \triangle \text{ but is not 2-colorable} \]

\[ \text{is 2-colorable} \]
\[ \text{but is not 1-colorable} \]

Nerdy note: Testing 2-colorability (equivalently, bipartiteness, equiv., having no odd length cycle) is in P. But even 3-colorability testing is already NP-complete.
Some \( \text{NP-Complete Sets, continued} \)

The Traveling Salesperson Problem

![Map of the United States with distances marked between cities.]

\[
\{ \text{directed pairwise distance table for a collection of } \ k \text{ cities} \} \\
\text{There is a tour of length } \leq k
\]

Nerdy note: the "symmetric" TSP (\( \text{dist}(a,b) = \text{dist}(b,a) \)) remains NPC; thanks goes to Hamiltonian cycle.
I want to eat carrots and relax! I don't want to have to do a Cook-like breakthrough each time I want to prove NP-completeness. Isn't there a handy tool to help me?
Thanks!

And I'll carefully remember ALL THREE aspects of the hypothesis.

Key Tool If
(1) $L$ is $NP$-complete, and
(2) $L' \in NP$, and
(3) $L \leq^p_m L'$, then $L'$ is $NP$-complete.

Proof $L' \in NP$ by (1).

We must show that every $NP$ problem reduces to $L'$.

Let $\hat{L} \in NP$. $\hat{L} \leq^p_m L$, since $L$ is $NP$-complete.

So $\hat{L} \leq^p_m L \leq^p_m L'$. Thus $\hat{L} \leq^p_m L'$, by the $p$-time reduction function $g(f(\cdot))$.

Nerdy note: Yes, that was an on-the-fly proof that $\leq^p_m$ is transitive... embedded in our proof of the Tool.
Taking Stock (Plus some facts)

- **SAT** = \( \{ f \mid \text{f is a satisfiable Boolean formula}\} \) is NP-complete (w.r.t. \( \leq_P \) reductions).
- **KEY TOOL**: \((L \text{ is } NP\text{-complete AND } L' \in NP \text{ AND } L \leq_P L') \Rightarrow L' \text{ is } NP\text{-complete} \) (aka NPC).

Fact (comes for free from Cook's Theorem's proof):
\[
CNF_{SAT} = \{ f \mid f \in SAT \land f \text{ is in conjunctive normal form} \}
\text{ is } NP.
\]

**Conjunctive normal form**: \((x_1 \lor x_17 \lor \overline{x_{27}}) \land (x_{17} \lor \overline{x_2} \lor \overline{x_1} \lor \overline{x_7}) \land \ldots\)

Fact: \(3CNF_{SAT}\) is NPC. (\(3CNF_{SAT}\) is the same as \(CNF_{SAT}\) except each clause can have at most three literal occurrences.)
\[
(x_1 \lor \overline{x_4} \lor \overline{x_{17}}) \land (\overline{x_2} \lor \overline{x_9} \lor \overline{x_{11}}) \land (x_2 \lor x_5 \lor x_8) \land \ldots
\]

**Nerdy note**: \(x_1\) is a literal. So is \(\overline{x_1}\). \{ holds for each variable \(x_i\).
\[ L_{\log} = \{ f | f \text{ is in 3CNF-form } \wedge f \in \text{SAT} \wedge \text{ the number of variables that appear complemented in } f \text{ is less than or equal to } \log_2(|f|) \} \]

\[ L_{\log} = \{ f_1 \ldots \ldots \ldots \ldots \ldots \ldots \log_2(|f|) \sqrt{1+|f|} \} \]

**Question:** How hard are \( L_{\log} \) and \( L_{\log} \)?

**Claim:** \( L_{\log} \in P \).

**Why?**

\[ (x_1 \lor \overline{x_1} \lor x_2) \land (x_3 \lor x_5 \lor \overline{x_1}) \land \ldots \]

**Algorithm:**

1. Check for 3CNF-form. Check that \# complemented vars. \( \leq \log_2(|f|) \).
   
   Reject if either is violated.

2. Instantiate the \( \leq \log_2(|f|) \) complemented variables every possible way.

\[ \# \text{ ways} \leq \log_2(|f|) \]

\[ \leq 2 \leq |f| \]

\[ \leq \log_2(|f|) \]

\[ \leq \log_2(|f|) \sqrt{1+|f|} \]

If any way satisfies then accept, else reject.
Claim \( L_{t_1t_2} \) is NPC.

**Part 1** \( L_{t_1t_2} \) is in \( NP \).

**Part 2** \( 3\text{CNF}_{\text{SAT}} \leq^p L_{t_1t_2} \). We'll build a \( \leq^m \) reduc. \( f \) so that

\[
(\forall x) [x \in 3\text{CNF}_{\text{SAT}} \iff f(x) \in L_{t_1t_2}].
\]

\[ g \rightarrow \begin{cases} 
F & \text{if } g \text{ is not in 3CNF-form.} \\
q \land \text{new}_1 \land \text{new}_2 \land \ldots \land \text{new}_{486} & \text{otherwise.} \\
\end{cases}
\]

So \( q \in 3\text{CNF}_{\text{SAT}} \iff f(q) \in L_{t_1t_2}. \) (Do you see why?) \[ \square \]
Thm \( L_\equiv = \{ f \mid \text{(each variable appearing in } f \text{ appears the same # of times)} \} \) is NPC.

Count \( x_i \) and \( \overline{x_i} \) as same variable for the above purpose.

E.g., \( x_1 \lor \overline{x}_1 \lor x_2 \lor \overline{x}_2 \in L_\equiv \).
\( x_1 \lor x_2 \lor \overline{x}_3 \lor x_3 \notin L_\equiv \).

\( \implies \text{Can you see how to prove this Thm?} \)

\[ \begin{array}{c}
\hline
\end{array} \]
Recall: $\text{Cook}^P(N_i, x)$ ["Does $N_i(x)$ accept?"]

\[\text{whoosh... fast!}\]

$F_{i,x}$

1. $N_i(x)$ accepts $\iff F_{i,x}$ is satisfiable.

2. We can produce $F_{i,x}$, given $N_i$ and $x$, in time polynomial in $\ln |N_i| + |x|^i + i$.

We have $N_i, x \xrightarrow{\text{fast}} F_{i,x}$.

Let us seek to achieve (thank you, Zvi Galil!)

$N_i, x \xrightarrow{\text{fast}} \hat{F}_{i,x}$.

("I come from $N_i$ and $x" \wedge (F_{i,x})"
Let $z = \langle N_i, x \rangle$.
So say $z$ as a binary string is $b_1b_2\ldots b_k$.

Let $\gamma$ be a variable that does not occur in $F_{i, x}$.

\[
F_{i, x} = \left( \bigvee_{b_1} \bigvee_{b_2} \ldots \bigvee_{b_k} \right) \land (F_{i, x}), \text{ where by/in this, we take/write: } \Gamma_{b_i} \text{ as } \neg \gamma \text{ if } b_i = 1
\]

\[\Gamma_{b_i} \text{ as } \gamma \text{ if } b_i = 0.\]

E.g. If $z = 10011$, then $\hat{F}_{i, x}$ will be of the form:

\[
(z \lor \neg z \lor \neg z \lor z \lor z \lor z \lor \top) \land (F_{i, x})
\]

This has all the magic of Cook's reduction.... plus, the output formula openly reveals the machine and input that it is about.

Cool.... but can we put this to work to see something surprising?
Likely True or False: \( (\exists A) \left[ A \in P \land A \subseteq \text{SAT} \land \text{"no poly-time machine can find solutions to all of } A\" \right] \).

\( (H_1) \):

i.e. \( (\forall f \in \text{FP}) (\exists g \in A) \left[ f(g) \text{ is not a satisfying assignment of } g \right] \).

"Obviously satisfiable but not obvious how they are satisfied."

Likely True or False: \( \text{NP} \cap \text{coNP} \neq \text{P} \).

\( (H_2) \):

Which is "more likely" to hold?

Informal (icky)
Thm. (the Borodin-Demers Theorem) \( NP \cap \text{coNP} \neq P \)

\[ (\exists A)[A \in P \land A \in \text{SAT} \land \"\text{no poly-time machine can find solutions to all of } A\" ] \]

i.e. (A poly-time comp. fcn \( f \))

\[ (\exists q \in A) \left[ f(q) \text{ is not a satisfying assignment of } q \right] \]

Note: (\( \exists A \)[A_k \land A \in P \land A \in \text{SAT}]) \text{ hold unconditionally, e.g., } A = \{T, TVT, TVTVT, \ldots \}.

It is just due to also having the 3rd Conjecture that this theorem has (very) sharp teeth.
Proof of the Borodin-Demers Theorem:

We assumed $\text{NPC} \neq \text{P}$. So, let $L \in (\text{NPC}) \setminus \text{P}$.

Let $N_k$ be an NPTM (from the std. enum.) accepting $L$.

Let $N_k$ "..." accepting $L$.

Set: $S = \{f : (\exists x)[f \text{ is } (\hat{F}_{k,x}) \vee (\hat{F}_{k,x})]\}$.

This completes the construction!!—it took just a few lines!!!

Claim $S \subseteq \text{P}$.

Claim $S \subseteq \text{SAT}$.

Pf. $(\forall x)\exists \gamma \in L \vee x \notin L$.

So, $(\forall x)\exists (\hat{F}_{k,x}) \vee (\hat{F}_{k,x})$ is satisfiable.

Claim If you could efficiently (Ptime) find satisfying assignments for $S$, then you could determine membership in $L$ in poly time. (Why?) (But that yields a contradiction, as $L \notin \text{P}$ was assumed.)

Q.E.D.