Today

- Diagonalization: Power & Problems
- Relativization
- Baker-Gill-Solovay
- Introduction to Alternation

Big picture in complexity

- E.g., Would like a complete map of complexity?
- Unfortunately: only one tool so far - Diagonalization.
- Diagonalization can prove:
  - Problems undecidable.
  - Space hieararchy, time hierarchy.
  - Ladner's theorem (between any two classes is an infinitely dense hierarchy).
  - But can it resolve NP $\stackrel{?}{=}$ P?

Preview of Ladner's theorem

- Suppose P $\neq$ NP.
- Let $L_1 \in$P and L_2 be NP-complete.
- Let $n_1 = 1$ and $n_i = 2^{n_{i-1}}$.
- Let $L = L_1$ for strings of length $[n_{i-1}, n_i)$ for odd i, and $L = L_2$ for strings of length $[n_{i-i}, n_i)$ for even i.
- $L \in$P? Probably not.
- Is L NP-complete? Probably not.
- Ladner's theorem picks a more careful choice of n_i's (by "lazy diagonalization"), to eliminate the "Probably's" above.

- Won't cover theorem in detail.

Power of diagonalization

- Can it resolve NP $\stackrel{?}{=}$ P?
- Question raised in the seventies.
- Baker-Gill-Solovay: No!
- Err.... some caveats ....

Relativization

Defn: Let C be a complexity class of languages decidable with machines having a certain resource bound. Let A be any language. Then C^A is the set of languages accepted by oracle machines, with the same (similar?) resource bound as machines in C, having access to oracle for A.

Warning: Not really a definition!

Defn: P^A is the set of all languages accepted by deterministic polynomial time oracle Turing machines with access to oracle for A.

Defn: NP^A is the set of all languages accepted by non-deterministic polynomial time oracle Turing machines with access to oracle for A.

B-G-S Proposition

Prop: If diagonalization shows $C_1 \not\subseteq C_2$, then for every A, $C_1^A \not\subseteq C_2^A$.

Jargon: $C_1 \not\subseteq C_2$ relativizes.

Proof (of Prop/Jargon):

- Exists machine in C_1 that can simulate any machine in C_2. (Since diagonalization works.)
- Augment this machine into an oracle machine.
- Machine now shows that C_1^A diagonalizes C_2^A.

BGS Lemmas

Lemma 1 There exists an oracle A such that NP^A =P^A.

Proof: Take some language that is sufficiently powerful. Example: Let A be any PSPACE-complete language. Then NP^A $=$ NPSPACE $=$ PSPACE $=$ P^A.

BGS Lemmas

Lemma 2 There exists an oracle B such that $\text{NP}^B \neq \text{P}^B$.

Proof:

- Insert proof here.

BGS Warnings

- Proof makes sense only when specialized (to say P vs. NP).
- Otherwise, it is pedagogy, not mathematics.
- Only rules out very specific proofs. Minor variations not accepted!
- Often misinterpreted, mispresented, misrepresent etc.

Constructive use of relativization

- What happens when A is an interesting problem, and C an interesting class? C^A must be interesting too?
- Example - we considered C = NP and A = PSPACE. What if A = NP? Is NP^{NP} = NP?
- No: actually get something new!

DNF Minimization

Defn: $\textsc{MinDNF}$ is the language consisting of pairs (ϕ, k), such that ϕ is a DNF formula such that no DNF formula with fewer than k literals is equivalent to ϕ.

Prop: $\textsc{MinDNF}$ is in NP^{NP}.

Proof: Below is an NP oracle machine M that accesses a SAT oracle:

- Guess a formula ψ with fewer than k literals.
- Ask SAT oracle if there exists an assignment x such that $\psi(x) \neq \phi(x)$.
- Accept if oracle says NO.

Note: we get the power to negate the oracles' response (or do any other polynomial time computation on it).

Introduction to Polynomial Hierarchy

Defn: $\Sigma_1^P = \text{NP}$. For $i > 1$, $\Sigma_i^P = \cup_{A \in \Sigma_{i-1}^P} NP^A$. $\Pi_i^P = \{\overline{L} | L \in \Sigma_i^P$. $\text{PH} = \cup_{i>0} \Sigma_i^P = \cup_{i>0} \Pi_i^P$.

Belief: For every $i > 0$ $\Sigma_i^P \neq \Sigma_{i+1}^P$.

Jargon: The Polynomial Hierarchy does not collapse.

More on the hierarchy later.

Alternation

- The hierarchy gains its power by complementing responses of oracles.
- DeMorgan's Law =¿ instead of existential guesses, it can now make universal guesses.
- Suppose we built this into a Turing machine.
- Machine has two special states: $\exists$ and $\forall$, both with two arcs leading out.
  - $\exists$ state accepts if one of the two paths leading out accepts.
  - $\forall$ state accepts if both paths accept.

- Alternation = Resource: write down computation tree: Count max. # times we alternate enter an $\exists$ node and then a $\forall$ node.
- This is a (valuable) resource!

Alternating complexity classes

- Three basic resources in ATM:
  - Time
  - Space
  - Alternations
- Classes:
  - ATIME$[t]$ = Languages accepted by ATMs running in time $t(n)$.
  - ASPACE$[s]$ = Languages accepted by ATMs using space $s(n)$.
  - (only of technical interest) ATISP$[a, t, s]$ = ... $a(n)$ alternations, $t(n)$ time, and $s(n)$ space.
- PH: Σ_i^P = languages accepted by polytime

bounded ATMs starting in existential state and making at most $i-1$ alternations.

Basic theorems about alternations

Thm 1: ATIME$(f) \subseteq$ SPACE$(f) \subseteq$ATIME(f^2).

Thm 1: ASPACE(f) = TIME$(2^{O(f)})$.