Outline (Umans slides)

- Oracle Turing Machines
- The Polynomial-Time Hierarchy (**PH**)
- Quantified SAT
- Complete problems for classes in **PH**, **PSPACE**

Oracle Turing Machines

- Oracle Turing Machine (OTM):
  - multitape TM M with special "query" tape
  - special states $q_?$, q_{yes}, q_{no}
  - on input x, with oracle language A
  - M^A runs as usual, except…
  - when M^A enters state $q_?$:
    - y = contents of query tape
    - $y \in A \Rightarrow$ transition to q_{yes}
    - $y \notin A \Rightarrow$ transition to q_{no}

Oracle Turing Machines

Shorthand #1:

- applying oracles to entire complexity classes:
  - complexity class **C**
  - language A

  $\mathbf{C}^A$ = {L decided by OTM M with oracle A with M "in" **C**}

  - example: $\mathbf{P}^{SAT}$

Oracle Turing Machines

- Nondeterministic OTM
  - defined in the same way
  - (transition relation, rather than function)
- oracle is like a subroutine, or function in your favorite PL
  - but each call counts as single step

  e.g.: given $\varphi_1, \varphi_2, \ldots, \varphi_n$ are even # satisfiable?
  - poly-time OTM solves with SAT oracle

The Polynomial-Time Hierarchy

$$\Sigma_0 = \Pi_0 = P$$

$$\Delta_1 = P^P \qquad \Sigma_1 = NP \qquad \Pi_1 = coNP$$
$$\Delta_2 = P^{NP} \qquad \Sigma_2 = NP^{NP} \qquad \Pi_2 = coNP^{NP}$$

$$\Delta_{i+1} = P^{\Sigma_i} \qquad \Sigma_{i+i} = NP^{\Sigma_i} \qquad \Pi_{i+1} = coNP^{\Sigma_i}$$

Polynomial Hierarchy $PH = \cup_i \Sigma_i$

Oracle Turing Machines

Shorthand #2:

- using complexity classes as oracles:
  - OTM M
  - complexity class **C**
  - $M^{\mathbf{C}}$ decides language L if for some language $A \in \mathbf{C}$, M^A decides L

Both together: $\mathbf{C}^{\mathbf{D}}$ = languages decided by OTM "in" **C** with oracle language from **D**

exercise: show $\mathbf{P}^{SAT} = \mathbf{P}^{NP}$

The Polynomial-Time Hierarchy

$$\Sigma_0 = \Pi_0 = P$$
$$\Delta_{i+1} = P^{\Sigma_i} \qquad \Sigma_{i+i} = NP^{\Sigma_i} \qquad \Pi_{i+1} = coNP^{\Sigma_i}$$

- Example:
  - MIN CIRCUIT: given Boolean circuit C, integer k; is there a circuit C' of size at most k that computes the same function C does?
  - MIN CIRCUIT $\in \Sigma_2$

The Polynomial-Time Hierarchy

- can define lots of complexity classes using oracles
- the following classes stand out
  - they have natural complete problems
  - they have a natural interpretation in terms of alternating quantifiers
  - they help us state certain consequences and containments (more later)

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

where $R \in \mathbf{\Pi_{i-1}}$.

- Corollary: $L \in \mathbf{\Pi_i}$ iff expressible as

$$L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \}$$

where $R \in \mathbf{\Sigma_{i-1}}$.

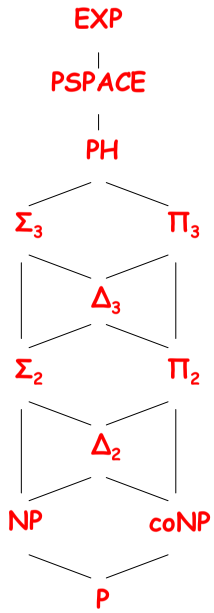
The PH

PSPACE: generalized geography, 2-person games…

3rd level: V-C dimension…

2nd level: MIN CIRCUIT, Succinct Set Cover, **BPP…**

1st level: SAT, UNSAT, factoring, etc…



Alternating quantifiers

Nicer, more usable version:

- $L \in \mathbf{\Sigma_i}$ iff expressible as

$$L = \{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q y_i (x, y_1, y_2, \ldots, y_i) \in R \}$$

where $Q = \forall/\exists$ if i even/odd, and $R \in \mathbf{P}$

- $L \in \mathbf{\Pi_i}$ iff expressible as

$$L = \{ x \mid \forall y_1 \exists y_2 \forall y_3 \ldots Q y_i (x, y_1, y_2, \ldots, y_i) \in R \}$$

where $Q = \exists/\forall$ if i even/odd, and $R \in \mathbf{P}$

Useful characterization

- Recall: $L \in \mathbf{NP}$ iff expressible as

$$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

where $R \in \mathbf{P}$.

- Corollary: $L \in \mathbf{coNP}$ iff expressible as

$$L = \{ x \mid \forall y, |y| \leq |x|^k, (x, y) \in R \}$$

where $R \in \mathbf{P}$.

Complete problems

- Recall:

MIN CIRCUIT: given Boolean circuit C, integer k; is there a circuit C' of size at most k that computes the same function C does?

$\{ (C, k) \mid \exists C' \; \forall x \; (|C'| \le k \text{ and } C'(x) = C(x)) \}$

– Conclude: in Σ_2

– (open whether it is complete for Σ_2)

Alternating quantifiers

- Proof:
  - ($\Rightarrow$) induction on i
  - base case: true for Σ_1=NP and Π_1=coNP
  - consider $L \in \Sigma_i$:

$$L = \{x \mid \exists y_1 \; (x, y_1) \in R' \}, \text{ for } R' \in \Pi_{i-1}$$

$$L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q y_i \; (x, y_1, y_2, \ldots, y_i) \in R\}$$

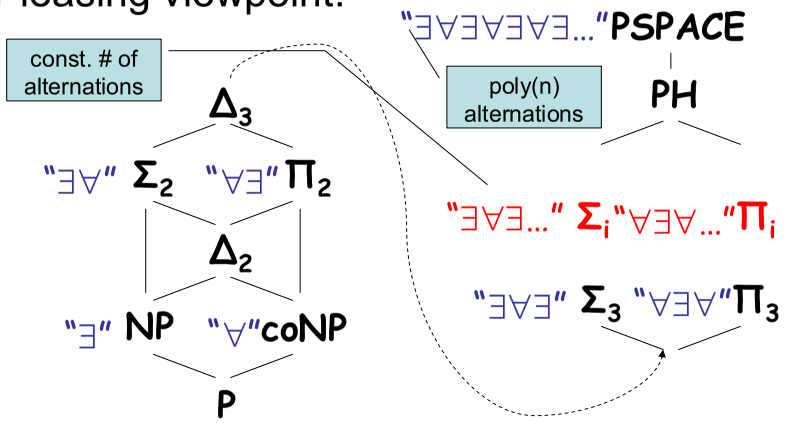
  - same argument for $L \in \Pi_i$
  - ($\Leftarrow$) exercise.

Complete problems

- three variants of SAT:
  - $QSAT_i$ (i odd) =
    {3-CNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i \; \varphi(x_1, x_2, \ldots, x_i) = 1\}$
  - $QSAT_i$ (i even) =
    {3-DNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_i \; \varphi(x_1, x_2, \ldots, x_i) = 1 \}$
  - QSAT = {3-CNFs φ for which
    $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n \; \varphi(x_1, x_2, \ldots, x_n) = 1\}$

Alternating quantifiers

Pleasing viewpoint:



PH collapse

– recall: $L \in \mathbf{\Sigma_{i+1}}$ iff expressible as

$$L = \{ x \mid \exists\, y\ (x, y) \in R \}$$

where $R \in \mathbf{\Pi_i}$

– since $\mathbf{\Pi_i = \Sigma_i}$, R expressible as

$$R = \{ (x,y) \mid \exists\, z\ ((x, y), z) \in R' \}$$

where $R' \in \mathbf{\Pi_{i-1}}$

– together: $L = \{ x \mid \exists\, (y,z)\ \ (x, (y,z)) \in R' \}$

– conclude $L \in \mathbf{\Sigma_i}$

QSAT is **PSPACE**-complete

Theorem: QSAT is **PSPACE**-complete.

- Proof:
  - in **PSPACE:** $\exists x_1 \forall x_2 \exists x_3 \ldots Q x_n\, \varphi(x_1, x_2, \ldots, x_n)?$
  - "$\exists x_1$": for each x_1, recursively solve
    $$\forall x_2 \exists x_3 \ldots Q x_n\, \varphi(x_1, x_2, \ldots, x_n)?$$
    - if encounter "yes", return "yes"
  - "$\forall x_1$": for each x_1, recursively solve
    $$\exists x_2 \forall x_3 \ldots Q x_n\, \varphi(x_1, x_2, \ldots, x_n)?$$
    - if encounter "no", return "no"
  - base case: evaluating a 3-CNF expression
  - poly(n) recursion depth
  - poly(n) bits of state at each level

Natural complete problems

- We now have versions of SAT complete for levels in **PH**, **PSPACE**
- Natural complete problems?
  - **PSPACE**: games
  - **PH**: almost all natural problems lie in the second level

PH collapse

Theorem: if $\mathbf{\Sigma_i = \Pi_i}$ then for all j > i

$$\mathbf{\Sigma_j = \Pi_j = \Delta_j = \Sigma_i}$$

"the polynomial hierarchy collapses to the i-th level"

- Proof:
  - sufficient to show $\mathbf{\Sigma_i = \Sigma_{i+1}}$
  - then $\mathbf{\Sigma_{i+1} = \Sigma_i = \Pi_i = \Pi_{i+1}}$; apply theorem again

Natural complete problems

– MIN CIRCUIT
  • good candidate, still open

– MIN DNF: given DNF φ, integer k; is there a DNF φ' of size at most k computing same function φ does?

– example:

$x_1x_2x_3 \vee x_1x_2\neg x_3 \vee x_4$

Natural complete problems

– MIN CIRCUIT
  • good candidate, still open

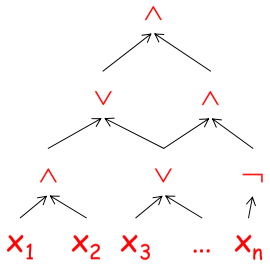
– MIN DNF: given DNF φ, integer k; is there a DNF φ' of size at most k computing same function φ does?

– example:

$x_1x_2x_3 \vee x_1x_2\neg x_3 \vee x_4 \equiv x_1x_2 \vee x_4$

Boolean circuits

- **circuit** C
  – directed acyclic graph
  – nodes: AND (∧); OR (∨); NOT (¬); variables x_i



- C computes function $f:\{0,1\}^n \to \{0,1\}$ in natural way
  – identify C with function f it computes

Simpler version of MIN DNF

Theorem (U): MIN DNF is Σ_2-complete.

- we'll consider a simpler variant:

  – IRREDUNDANT: given DNF φ, integer k; is there a DNF φ' consisting of at most k terms of φ computing same function φ does?

Connection to TMs

- TM M running in time t(n) decides language L
- can build circuit family $\{C_n\}$ that decides L
  - size of $C_n = O(t(n)^2)$
  - Proof: CVAL construction

- Conclude: $L \in \mathbf{P}$ implies family of polynomial-size circuits that decides L

Boolean circuits

- **size** = # gates
- **depth** = longest path from input to output
- **formula (or expression)**: graph is a tree

- every function $f:\{0,1\}^n \rightarrow \{0,1\}$ computable by a circuit of size at most $O(n2^n)$
  - AND of n literals for each x such that f(x) = 1
  - OR of up to 2^n such terms

Uniformity

- Strange aspect of circuit family:
  - can "encode" (potentially uncomputable) information in family specification
- solution: uniformity – require specification is simple to compute
  - Definition: circuit family $\{C_n\}$ is **logspace uniform** iff TM M outputs C_n on input 1^n and runs in O(log n) space

Circuit families

- circuit works for specific input length
- we're used to $f:\sum^* \rightarrow \{0,1\}$
- circuit **family** : a circuit for each input length $C_1, C_2, C_3, \ldots$ = "$\{C_n\}$"
- "$\{C_n\}$ computes f" iff for all x

$$C_{|x|}(x) = f(x)$$

- "$\{C_n\}$ decides L", where L is the language associated with f

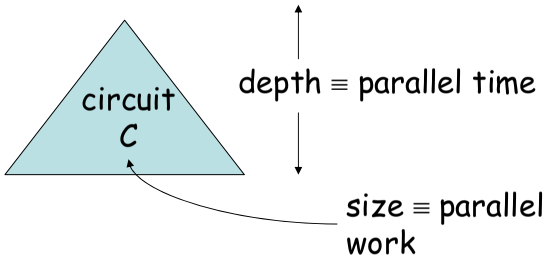
Uniformity

Theorem: **P** = languages decidable by logspace uniform, polynomial-size circuit families $\{C_n\}$.

- Proof:
  - already saw ($\Rightarrow$)
  - ($\Leftarrow$) on input x, generate $C_{|x|}$, evaluate it and accept iff output = 1

Parallelism

- uniform circuits allow refinement of polynomial time:



Parallelism

- the **NC** ("**N**ick's **C**lass") Hierarchy (of logspace uniform circuits):

  $\mathbf{NC_k}$ = $O(\log^k n)$ depth, poly(n) size

  $\mathbf{NC} = \cup_k \mathbf{NC_k}$
- captures "efficiently parallelizable problems"
- not realistic? overly generous
- OK for proving non-parallelizable