CS151 Complexity Theory

Lecture 12
May 11, 2023

1

RL

- Recall: probabilistic Turing Machine
  - deterministic TM with extra tape for "coin flips"
- **RL** (Random Logspace)
  - $L \in \mathbf{RL}$ if there is a probabilistic logspace TM M:

    $x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq ½$

    $x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$
  - important detail #1: only allow one-way access to coin-flip tape
  - important detail #2: explicitly require to run in polynomial time

2

RL

- $\mathbf{L} \subseteq \mathbf{RL} \subseteq \mathbf{NL} \subseteq \mathbf{SPACE}(\log^2 n)$
- Theorem (SZ) : $\mathbf{RL} \subseteq \mathbf{SPACE}(\log^{3/2} n)$
- Belief: L = RL (major open problem)

3

RL

$\mathbf{L} \subseteq \mathbf{RL} \subseteq \mathbf{NL}$

- Natural problem:

  Undirected STCONN: given an **undirected** graph G = (V, E), nodes s, t, is there a path from s → t?

Theorem: USTCONN ∈ **RL.**

(Recall: STCONN is **NL**-complete)

4

Undirected STCONN

- Proof sketch: (in Papadimitriou)
  - add self-loop to each vertex (technical reasons)
  - start at s, random walk 2|V||E| steps, accept if see t
  - Lemma: expected **return time** for any node i is $2|E|/d_i$
  - suppose $s=v_1, v_2, \ldots, v_n=t$ is a path
  - expected time from v_i to v_{i+1} is $(d_i/2)(2|E|/d_i) = |E|$
  - expected time to reach $v_n \leq |V||E|$
  - Pr[fail reach t in 2|V||E| steps] ≤ ½
- Reingold 2005: USTCONN ∈ **L**

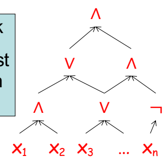
5

6

A motivating question

- Central problem in logic synthesis:

> • given Boolean circuit C, integer k
>
> • is there a circuit C' of size at most k that computes the same function C does?



- Complexity of this problem?
  - **NP**-hard? in **NP**? in **coNP**? in **PSPACE**?
  - complete for any of these classes?

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

- Nondeterministic OTM
  - defined in the same way
  - (transition relation, rather than function)
- oracle is like a subroutine, or function in your favorite programming language
  - 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

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

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 $\mathbf{P}^{SAT} = \mathbf{P}^{\mathbf{NP}}$

The Polynomial-Time Hierarchy

- can define lots of complexity classes using oracles
- the classes on the next slide 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)

The Polynomial-Time Hierarchy

$\Sigma_0 = \Pi_0 = \mathbf{P}$

$\Delta_1 = \mathbf{P}^{\mathbf{P}}$ $\Sigma_1 = \mathbf{NP}$ $\Pi_1 = \mathbf{coNP}$

$\Delta_2 = \mathbf{P}^{\mathbf{NP}}$ $\Sigma_2 = \mathbf{NP}^{\mathbf{NP}}$ $\Pi_2 = \mathbf{coNP}^{\mathbf{NP}}$

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

Polynomial Hierarchy $\mathbf{PH} = \cup_i \Sigma_i$

13

The Polynomial-Time Hierarchy

$\Sigma_0 = \Pi_0 = \mathbf{P}$

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

- 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 $\in \Sigma_2$

14

The Polynomial-Time Hierarchy

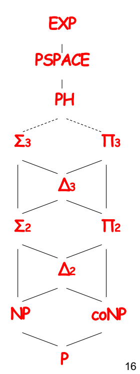
$\Sigma_0 = \Pi_0 = \mathbf{P}$

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

- Example:
  - EXACT TSP: given a weighted graph G, and an integer k; is the k-th bit of the length of the *shortest* TSP tour in G a 1?
  - EXACT TSP $\in \Delta_2$

15

The PH



PSPACE: generalized geography, 2-person games…

3rd level: V-C dimension…

2nd level: MIN CIRCUIT, **BPP…**

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

16

Useful characterization

- Recall: L $\in$ **NP** iff expressible as
  $L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$
  where R $\in$ **P**.
- Corollary: L $\in$ **coNP** iff expressible as
  $L = \{ x \mid \forall y, |y| \le |x|^k, (x, y) \in R \}$
  where R $\in$ **P**.

17

Useful characterization

Theorem: L $\in \Sigma_i$ iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$
where R $\in \Pi_{i-1}$.

- Corollary: L $\in \Pi_i$ iff expressible as
  $L = \{ x \mid \forall y, |y| \le |x|^k, (x, y) \in R \}$
  where R $\in \Sigma_{i-1}$.

18

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where $R \in \mathbf{\Pi_{i-1}}$.

- Proof of Theorem:
  - induction on i
  - base case (i =1) on previous slide
  - (⇐)
  - we know $\mathbf{\Sigma_i = NP^{\Sigma_{i-1}} = NP^{\Pi_{i-1}}}$
  - guess y, ask oracle if $(x, y) \in R$

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where $R \in \mathbf{\Pi_{i-1}}$.

(⇒)
- given $L \in \mathbf{\Sigma_i = NP^{\Sigma_{i-1}}}$ decided by ONTM M running in time n^k
- try: $R = \{ (x, y) :$ y describes valid path of M's computation leading to $q_{accept} \}$
- but how to recognize valid computation path when it depends on result of oracle queries?

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where $R \in \mathbf{\Pi_{i-1}}$.

- try: $R = \{ (x, y) :$ y describes valid path of M's computation leading to $q_{accept} \}$
- valid path = step-by-step description including **correct** yes/no answer for each A-oracle query z_j $(A \in \mathbf{\Sigma_{i-1}})$
- verify "no" queries in $\mathbf{\Pi_{i-1}}$:

  e.g: $z_1 \notin A \wedge z_3 \notin A \wedge \ldots \wedge z_8 \notin A$
- for each "yes" query z_j: $\exists w_j, |w_j| \le |z_j|^k$ with $(z_j, w_j) \in R'$ for some $R' \in \mathbf{\Pi_{i-2}}$ by induction.
- for each "yes" query z_j put w_j in description of path y

Useful characterization

Theorem: $L \in \mathbf{\Sigma_i}$ iff expressible as
$L = \{ x \mid \exists y, |y| \le |x|^k, (x, y) \in R \}$, where $R \in \mathbf{\Pi_{i-1}}$.

- single language R in $\mathbf{\Pi_{i-1}}$:

  $(x, y) \in R$

  ⇔

  all "no" z_j are not in A and
  all "yes" z_j have $(z_j, w_j) \in R'$ and
  y is a path leading to q_{accept}.
- Note: AND of polynomially-many $\mathbf{\Pi_{i-1}}$ predicates is in $\mathbf{\Pi_{i-1}}$.

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= ∀/∃ 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= ∃/∀ if i even/odd, and $R \in \mathbf{P}$

Alternating quantifiers

- Proof:
  - (⇒) induction on i
  - base case: true for $\mathbf{\Sigma_1 = NP}$ and $\mathbf{\Pi_1 = coNP}$
  - consider $L \in \mathbf{\Sigma_i}$:

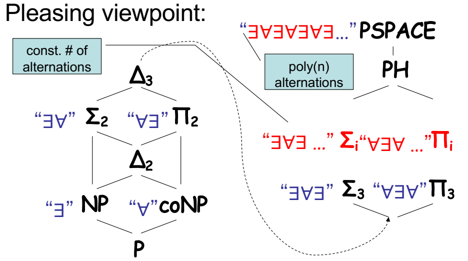
    $L = \{x \mid \exists y_1 (x, y_1) \in R'\}$, for $R' \in \mathbf{\Pi_{i-1}}$

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

    $L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q y_i (x, y_1, y_2, \ldots, y_i) \in R\}$
  - same argument for $L \in \mathbf{\Pi_i}$
  - (⇐) exercise.

Alternating quantifiers

Pleasing viewpoint:



25

Complete problems

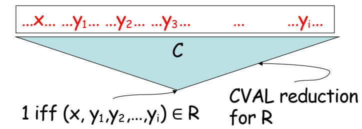
- three variants of SAT:
  - $QSAT_i$ (i odd) = {3-CNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which $\exists x_1 \forall x_2 \exists x_3 \ldots \exists x_i\ \varphi(x_1, x_2, \ldots, x_i) = 1$}
  - $QSAT_i$ (i even) = {3-DNFs $\varphi(x_1, x_2, \ldots, x_i)$ for which $\exists x_1 \forall x_2 \exists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

26

$QSAT_i$ is Σ_i-complete

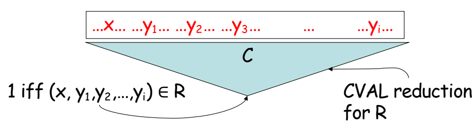
Theorem: $QSAT_i$ is Σ_i-complete.

- Proof: (clearly in Σ_i)
  - assume i odd; given $L \in \Sigma_i$ in form
    $\{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \exists y_i (x, y_1, y_2, \ldots, y_i) \in R \}$



27

$QSAT_i$ is Σ_i-complete

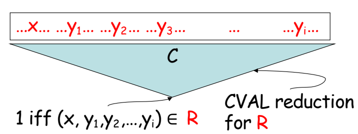


- Problem set: can construct 3-CNF φ from C:
  $\exists z\ \varphi(x, y_1, \ldots, y_i, z) = 1 \Leftrightarrow C(x, y_1, \ldots, y_i) = 1$
- we get:
  $\exists y_1 \forall y_2 \ldots \exists y_i \exists z\ \varphi(x, y_1, \ldots, y_i, z) = 1$
  $\Leftrightarrow \exists y_1 \forall y_2 \ldots \exists y_i\ C(x, y_1, \ldots, y_i) = 1 \Leftrightarrow x \in L$

28

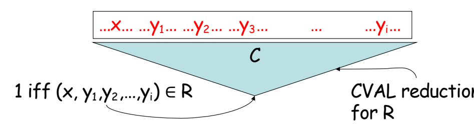
$QSAT_i$ is Σ_i-complete

- Proof (continued)
  - assume i even; given $L \in \Sigma_i$ in form
    $\{ x \mid \exists y_1 \forall y_2 \exists y_3 \ldots \forall y_i (x, y_1, y_2, \ldots, y_i) \in R \}$



29

$QSAT_i$ is Σ_i-complete



- Problem set: can construct 3-DNF φ from C:
  $\forall z\ \varphi(x, y_1, \ldots, y_i, z) = 1 \Leftrightarrow C(x, y_1, \ldots, y_i) = 1$
- we get:
  $\exists y_1 \forall y_2 \ldots \forall y_i \forall z\ \varphi(x, y_1, y_2, \ldots, y_i, z) = 1$
  $\Leftrightarrow \exists y_1 \forall y_2 \ldots \forall y_i\ C(x, y_1, y_2, \ldots, y_i) = 1 \Leftrightarrow x \in L$

30

QSAT is **PSPACE**-complete

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

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

31

QSAT is **PSPACE**-complete

- given TM M deciding L ∈ **PSPACE**; input x
- 2^{n^k} possible configurations
- single START configuration
- assume single ACCEPT configuration

- define:
  REACH(X, Y, i) ⇔ configuration Y reachable from configuration X in at most 2^i steps.

32

QSAT is **PSPACE**-complete

REACH(X, Y, i) ⇔ configuration Y reachable from configuration X in at most 2^i steps.

- Goal: produce 3-CNF $\varphi(w_1,w_2,w_3,\ldots,w_m)$ such that

$$\exists w_1\ \forall w_2 \ldots Qw_m\ \varphi(w_1,\ldots,w_m)$$
$$\text{REACH(START, ACCEPT, } n^k)$$

33

QSAT is **PSPACE**-complete

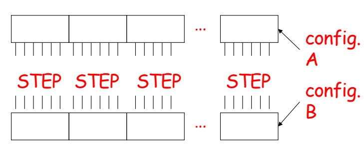
- for i = 0, 1, … n^k produce quantified Boolean **expressions** $\psi_i(A, B, W)$
  $\exists w_1\ \forall w_2 \ldots \psi_i(A, B, W) \Leftrightarrow \text{REACH}(A, B, i)$
- convert ψ_{n^k} to 3-CNF φ
  - add variables V
  - $\exists w_1\ \forall w_2 \ldots \exists V\ \varphi(A, B, W, V) \Leftrightarrow \text{REACH}(A, B, n^k)$
- hardwire A = START, B = ACCEPT
  $\exists w_1\ \forall w_2 \ldots \exists V\ \varphi(W, V)\ \Leftrightarrow\ x \in L$

34

QSAT is **PSPACE**-complete

- $\psi_o(A, B)$ = true iff
  - A = B or
  - A yields B in one step of M

  } Boolean expression of size $O(n^k)$



35

QSAT is **PSPACE**-complete

- Key observation #1:
  $$\text{REACH}(A, B, i+1)$$
  $$\Leftrightarrow$$
  $$\exists Z\ [\text{REACH}(A, Z, i) \wedge \text{REACH}(Z, B, i)]$$

- cannot define $\psi_{i+1}(A, B)$ to be
  $$\exists Z\ [\psi_i(A, Z) \wedge \psi_i(Z, B)]$$
  (why?)

36

QSAT is **PSPACE**-complete

– Key idea #2: use quantifiers

– couldn't do $\psi_{i+1}(A, B) = \exists Z\ [\psi_i(A, Z) \wedge \psi_i(Z, B)]$

– define $\psi_{i+1}(A, B)$ to be

$\exists Z \forall X \forall Y\ [((X=A \wedge Y=Z) \vee (X=Z \wedge Y=B)) \Rightarrow \psi_i(X, Y)]$

– $\psi_i(X, Y)$ is preceded by quantifiers

– move to front (they don't involve X,Y,Z,A,B)

37

QSAT is **PSPACE**-complete

$\psi_0(A, B)$ = true iff A = B or A yields B in 1 step

$\psi_{i+1}(A, B) =$

$\exists Z \forall X \forall Y[((X=A \wedge Y=Z) \vee (X=Z \wedge Y=B)) \Rightarrow \psi_i(X, Y)]$

– $|\psi_0| = O(n^k)$

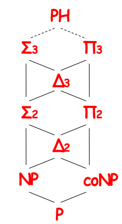
– $|\psi_{i+1}| = O(n^k) + |\psi_i|$

– total size of ψ_{n^k} is $O(n^k)^2$ = poly(n)

– logspace reduction

38

PH collapse



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

$\Sigma_j = \Pi_j = \Delta_j = \Sigma_i$

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

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

39

PH collapse

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

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

where $R \in \Pi_i$

– since $\Pi_i = \Sigma_i$, R expressible as

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

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

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

– conclude $L \in \Sigma_i$

40

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 and third level

41

Natural complete problems in PH

– MIN CIRCUIT
  - good candidate to be Σ_2-complete, still open

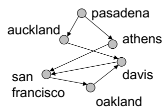
– MIN DNF: given DNF φ, integer k; is there a DNF φ' of size at most k computing same function φ does?

Theorem (U): MIN DNF is Σ_2-complete.

42

Natural complete problems in PSPACE

- General phenomenon: many 2-player games are PSPACE-complete.
  - 2 players I, II
  - alternate pick-ing edges
  - lose when no unvisited choice



- GEOGRAPHY = {(G, s) : G is a directed graph and player I can win from node s}

Natural complete problems in PSPACE

Theorem: GEOGRAPHY is PSPACE-complete.

Proof:
- in PSPACE
  - easily expressed with alternating quantifiers
- PSPACE-hard
  - reduction from QSAT

Natural complete problems in PSPACE

$\exists x_1 \forall x_2 \exists x_3 \ldots (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_3 \vee x_1) \wedge \ldots \wedge (x_1 \vee \neg x_2)$