LECTURE 5

Last time

- Space-bounded computation
- Polynomial space PSPACE ($\supseteq$ NP $\cup$ coNP)
- QBF PSPACE-complete (proof using configuration graph $G_{M,x}$)
- PSPACE = NPSPACE
- Savitch's theorem: Nondeterminism can be simulated with quadratic blow-up in space

Today

- Sublinear regime: logarithmic space L & NL
- NL-complete problems [mentioned but not defined last time]
- coNL
- And then to the other side of NP: Polynomial hierarchy

Most abstract concepts in course? Will get more concrete from here.

When studying logarithmic space, polynomial-time reductions are no good — so powerful that the reduction can solve the problem

Reduction should also work in (at most) log space.
But if so, how can reduction produce poly-sized output?

Two solutions:

(1) Write-only output tape — space doesn't count.
Write once write or move right
Never read, never move left

(2) Compute reduction bit by bit

Get equivalent definitions — we go for (2)

DEF 1

$f: \{0,1\}^* \to \{0,1\}^*$ implicitly logspace computable if

a) f polynomially bounded ($\exists c \forall x \; |f(x)| \leq c \cdot |x|^c$)

b) $L_f = \{ \langle x, i \rangle \mid f(x)_i = 1 \}$
$L'_f = \{ \langle x, i \rangle \mid i \leq |f(x)| \}$ are both in L

Language B is logspace reducible to language C, denoted $B \leq_L C$ if $\exists$ implicitly logspace computable f s.t. $x \in B \iff f(x) \in C$

C is NL-complete if $C \in NL$ and $\forall B \in NL \quad B \leq_L C$.

PROPOSITION 2

1. $B \leq_L C$ and $C \leq_L D \Rightarrow B \leq_L D$
2. $B \leq_L C$ and $C \in L \Rightarrow B \in L$

Proof Not hard but needs a bit of care. See textbook.

THEOREM 3 PATH is NL-complete

Recall $PATH = \{ \langle G, s, t \rangle \mid \exists$ path $s \rightsquigarrow t$ in digraph $G \}$

Proof Argued $PATH \in NL$ last time.

Let B in NL decided by M in log space

Define $f(x)$ to be configuration graph $G_{M,x}$ together with $|C_{start}| = s$ and $t = C_{accept}$.

Representation adjacency matrix

1 in position (C, C') if C, C' legal transition.

size $G_{M,x}$ has $\leq 2^{space}$ vertices. $2^{(\log)} = poly$ — OK.

computation given C, C', look up current state and tape contents and check that C' is one of two possible configs to follow from C.

Certificate-style definition of NL? | DEC 5 : III

"For every $x \in B$ $\exists$ witness y
s.t. $M(x,y) = 1$ and M runs in log space"

Need to be careful!

Suppose x CNF-formula, y satisfying assignment
Let M look up clauses in x one by one
then look up assignments in y
check that every clause satisfied.

Proves that CNF-SAT $\in$ NL ☐

Hence NP = NL (and P=NP) - great!

FIX: Make certificate read-once

DEF 4 Certificate-style definition of NL

C is in NL if exists deterministic TM M (verifier)
with
- read-only input tape
- read-once certificate tape [read or move right each step]
- read-write tapes with $O(\log |x|)$ space bound

s.t. $x \in C \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$

(for some fixed poly p depending on C).

LEMMA 5 Definitions 1 and 4 give exactly the same class NL

Proof Exercise (not hard but useful).

Complements of space-bounded complexity classes

Lec 5 : IV

$\overline{\text{PATH}} = \{ \langle G, s, t \rangle \mid \text{No path } s \rightsquigarrow t \text{ in digraph } G \}$

$\overline{\text{PATH}}$ in coNL (since PATH $\in$ NL)

In fact, $\overline{\text{PATH}}$ coNL-complete (since PATH NL-complete).

Log space NDTM deciding PATH:

Just walk nondeterministically for $|V(G)|$ steps from s, reject if didn't reach t

Most computation ~~paths~~ branches might reject, but if $\exists$ path then one branch will find it.

Log space NDTM deciding $\overline{\text{PATH}}$

Walk nondet & accept if didn't reach t?

A non-starter...

How can you make sure all branches find path $s \rightsquigarrow t$ for a no instance of PATH?!

Obviously can't be done, right? Seems clear that NL $\neq$ coNL, right? Wrong.

THM 6 NL = coNL — Immerman '88, Szelepcsényi '87

Proof Show that $\overline{\text{PATH}} \in$ NL.

Same ideas yield stronger statement (which we will not prove)

COR 7 For every space constructible $s(n) \geq \log n$ it holds that NSPACE$(s(n))$ = coNSPACE$(s(n))$

Moral: Don't trust your intuition too much regarding "obvious" truths in computational complexity theory (P vs NP, anyone?)

Proof of Thm 6

Provide read-once certificate for NL-complete language PATH

Important Read-once access to certificate
But can scan graph G as many times as wanted
(but not store on work tape)

$R(i) := \{v \in V(G) \mid \exists \text{ path } s \rightsquigarrow v \text{ of length} \le i\}$

$n = |V(G)|$ denote $V(G) = \{1, 2, \ldots, n\}$
denote $r_i = |R(i)|$

Want to certify $t \notin R(n)$

Starting point: Certifying $v \in R(i)$ easy

Give vertices in path $u_0 = s, u_1, u_2, \ldots, u_{i'} = v$ for $i' \le i$

Verification
- read vertices one by one
- keep u_j and u_{j+1} in memory - log space
- keep j in memory - Log space
- at each step, check $(u_j, u_{j+1}) \in E(G)$ by scanning input tape.
- check that j never exceeds i.

Let such a certificate be denoted
IsMEMBER(v, i) - $v \in R(i)$

Use this to construct two other types of certificates

(A) MEMBERSHIPEXPANSION (i, r, r')

Assuming $|R(i-1)| = r'$,

prove that $|R(i)| = r$

(B) NOTMEMBER (v, i, r)

Assuming that $|R(i)| = r$

prove that $v \notin R(i)$.

Suppose we can build such read-once verifiable subcertificates. Then we're done!

We all know $R(0) = \{s\}$ and $|R(0)| = 1$

~~Suppose~~ Let $r_i = |R(i)|$.

Here is the certificate

```
MEMBERSHIPEXPANSION (1, r_2, 1);
MEMBERSHIPEXPANSION (2, r_2, r_1);
MEMBERSHIPEXPANSION (3, r_3, r_2);
   ⋮
MEMBERSHIPEXPANSION (n, r_n, r_{n-1});
NOTMEMBER (t, n, r_n)
```

- check each line in read-once fashion.
- keep counter i and neighbourhood size r_i
  → $\log n$ space
- finally verify nonmembership certificate
- each expansion certificate for step j is verified using stored r_{j-1} → $\log n$ space

(8) NOTMEMBER (v, i, r)

Suppose $R(i) = \{u_1, u_2, \ldots, u_r\}$ $u_1 < u_2 < \ldots < u_r$

Let certificate be sorted list of u_j's with membership certificates

u_1 :	ISMEMBER (u_1, i)	Denote this by
u_2 :	ISMEMBER (u_2, i)	=: LISTMEMBERS (i, r)
⋮		
u_r :	ISMEMBER (u_r, i)	

Verification
- r is known
- go over list and read u_j
- for each u_j, check certificate of membership
- check $u_j > u_{j-1}$
- check $u_j \neq v$
- check # u_j-vertices $= r$

(A) MEMBERSHIPEXPANSION (i, r, r')

Use auxiliary certificate NOTMEMBERORNEIGHBOUR (v, i, r')

Assuming $|R(i+1)| = r'$, prove that $v \notin R(i+1)$

Reuse ~~LISTMEMBERS (i, r)~~ LISTMEMBERS (i, r')

Verification Go over list and verify u_j's as above

For each u_j, check $u_j \neq v$

and that $(u_j, v) \notin E(G_i)$

Now we can write down

MEMBERSHIPEXPANSION (i, r, r')

as ordered list of vertices $1, 2, 3, \ldots, n$

If vertex $j \in R(i)$, the line for vertex j is

j : ISMEMBER (j, i) $\quad (j, i-1, r')$

If $j \notin R(i)$, the line for vertex j is

j : NOTMEMBERORNEIGHBOR $(j, i-1, r')$

$(=$ LISTMEMBERS $(i-1, r')$

Verification - For each j check correctness of membership or non-membership cert.

- Count total # members; check that sum is $= r$

This concludes the proof □

SUMMARY SO FAR DURING THE COURSE

$$\mathcal{L} \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$$

Some inclusions must be strict, since

- $\mathcal{L} \subsetneq PSPACE$ (space hierarchy theorem)
- $P \subsetneq EXP$ (time hierarchy theorem)

But we don't know which.

Probably most of them, or even all.

So far today looked at classes below P

Let's now zoom in on NP and PSPACE and what might be between them.

EXAMPLES

For $G = (V, E)$, an INDEPENDENT SET is a subset $S \subseteq V$ s.t. $\forall u, v \in S \quad (u, v) \notin E$

$$\text{INDEPENDENTSET} = \{ \langle G, k \rangle \mid G \text{ has an independent set of size } k \}$$

NP-complete.

What about

$$\text{EXACTINDEPENDENTSET} = \{ \langle G, k \rangle \mid \text{largest independent set in } G \text{ has size exactly } k \}$$

"Exists certificate of size k that works and Forall certificates of size $k+1$ they have to fail"

$\text{EXACTINDEPENDENTSET} \in \Sigma_2^P$

$L \in \Sigma_2^P$ if $\exists$ poly-time Deterministic TM M and polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \; \forall v \in \{0,1\}^{q(|x|)} \; M(x, u, v) = 1$$

DEF 9 — LEC 5 X

POLYNOMIAL HIERARCHY

For $i \in \mathbb{N}^+$, $L \subseteq \{0,1\}^*$ is in Σ_i^P if exists (deterministic) TM M polynomial-time and polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \ldots Q_i u_i \in \{0,1\}^{q(|x|)} M(x, u_1, \ldots, u_i) = 1$$

$Q_i = \exists$ for i odd and $= \forall$ for i even

Polynomial hierarchy $PH = \bigcup_{i \in \mathbb{N}^+} \Sigma_i^P$

Define $\Pi_i^P = co\Sigma_i^P = \{ \overline{L} : L \in \Sigma_i^P \}$

$$\Sigma_1^P = NP$$

$$\Pi_2^P = co\,NP$$

$$\Sigma_i^P \subseteq \Pi_{i+1}^P \subseteq \Sigma_{i+2}^P$$

Is it true that $\Sigma_i^P \subsetneq \Sigma_{i+1}^P \subsetneq \Sigma_{i+2}^P \subsetneq \ldots$

Is it true that "the polynomial hierarchy doesn't collapse"?

THM 10

1. For every i, if $\Sigma_i^P = \Pi_i^P$ then $PH = \Sigma_i^P$; i.e. the polynomial hierarchy collapses to the ith level.
2. If $P = NP$ then $PH = P$, that is, the hierarchy collapses to P.

Many complexity theory results have form: "Unless ⟨sth we believe to be true⟩ then PH collapses to the ith level."

Smaller i ⇒ stronger claim ←

Proof of Thm 10.2

Prove by induction: If $P = NP$ then $\Sigma_i^P = \Pi_i^P = P$.

Base case ($i=1$) Nothing to prove. By assumption $NP = P$. Since P closed under complement, $coNP = P$.

Induction step Suppose $\Sigma_{i-1}^P = P = \Pi_{i-1}^P$.

By definition $\Pi_{i-1}^P \subseteq \Sigma_i^P$ so $P \subseteq \Sigma_i^P$.

Enough to prove $\Sigma_i^P \subseteq P$. Then $P = \Sigma_i^P$ and again $P = \Pi_i^P$ by taking complements.

Consider any $L \in \Sigma_i^P$. Want to prove $L \in P$.

By definition, $\exists$ poly-time TM M and polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \dots Q_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, u_2, \dots, u_i) = 1$$

Define L' by

$$\langle x, u_1 \rangle \in L' \iff \forall u_2 \in \{0,1\}^{q(|x|)} \dots Q_i u_i \in \{0,1\}^{q(|x|)} \; M(x, u_1, u_2, \dots, u_i)$$

By pattern matching $L' \in \Pi_{i-1}^P$.

By inductive hypothesis $\Pi_{i-1}^P = P$, i.e. $\exists$ poly-time TM M' deciding L'.

That is

$$\langle x, u_1 \rangle \in L' \iff M'(x, u_1) = 1$$

But then

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \; M'(x, u_1) = 1$$

Shows that $L \in NP$ with verifier M'.

But $NP = P$ by assumption, so $L \in P$.

Hence $\Sigma_i^P \subseteq P$ QED ∎

DEF 11

Language $L \subseteq \{0,1\}^*$ is Σ_i^P-complete if

- $L \in \Sigma_i^P$
- For every $L' \in \Sigma_i^P$ it holds that $L' \leq_P L$

Π_i^P-complete and PH-complete languages defined analogously.

LEMMA 12 PH does not have complete languages unless the hierarchy collapses.

Proof

Suppose L is PH-complete

$PH = \bigcup_{i \in \mathbb{N}^+} \Sigma_i^P$ so there is some i such that $L \in \Sigma_i^P$. But then every language in PH can be reduced to $L \in \Sigma_i^P$. ☐

COR 13 $PH \subseteq PSPACE$ but $PH \neq PSPACE$ unless the polynomial hierarchy collapses.

Proof Checking that $\forall u_1 \exists u_2 \forall u_3 \exists u_4 \ldots M(x, u_1, u_2, u_3, u_4 \ldots) = 1$ is essentially verifying a QBF formula.

Hence $PH \subseteq PSPACE$.

If $PSPACE = PH$ then TQBF is complete for PH.

By Lem 12, the hierarchy collapses ☐

EXAMPLE 14

The classes Σ_i^P, $i \in \mathbb{N}^+$, all have complete problems

$$\Sigma_i\text{SAT} = \{ \varphi \mid \varphi = \exists u_1 \forall u_2 \exists u_3 \ldots Q_i u_i \, \varphi(u_1, \ldots, u_i) \}$$

u_j bit vectors, φ CNF formula

ALTERNATING TURING MACHINE

A NDTM M is on its way to complete an accepting computation if $\exists$ path from current state to accepting state. Such a machine can solve SAT.

We could imagine a "co-NDTM" M' that accepts if <u>all paths</u> from current state accepts — would solve UNSAT.

For EXACT INDEPENDENT SET, could use TM that combines both approaches:

- starts out in "existential mode" to check $\exists$ independent set of size k
- then switches to "universal mode" to check that all vertex sets of size $k+1$ fail to be independent.

Such a machine could decide Σ_2^P - languages.

<u>DEF 15</u> (Informal)

An <u>alternating TM</u> has two transition functions and every state labelled $\exists$ or $\forall$

- accepting computation from $\exists$-state if $\exists$ <u>one</u> ~~accepting~~ path to accepting state
- accepting computation from $\forall$-state if <u>all paths</u> lead to accepting states.

L is in $ATIME(T(n))$ if decided by alternating TM in time $O(T(n))$.

Alternation: change along computation path from ∃ to ∀ or from ∀ to ∃ - labelled states

DEF 15½ (Also a bit informal)

Σ_i TIME(T(n)) : languages L decided by ATM that

- runs in time O(T(n))
- decides the language L
- start state labelled ∃
- along all computation paths, at most i-1 alternations

Π_i TIME (T(n)).

The same, except starting state labelled ∀

LEMMA 16

$$\Sigma_i^P = \bigcup_{c \in \mathbb{N}^+} \Sigma_i \text{TIME}(n^c)$$

$$\Pi_i^P = \bigcup_{c \in \mathbb{N}^+} \Pi_i \text{TIME}(n^c)$$

$$\text{PSPACE} = AP = \bigcup_{c \in \mathbb{N}^+} \text{ATIME}(n^c)$$

What on earth is this good for ?!

Can be used to show e.g. that SAT cannot be solved by machine running in nearly linear time and using much less than linear space

(But we'll skip this...)

Yet another definition of PH (the third!)
via... oracle TMs

Recall

$M^O(x)$ Output of TM M on input x when M can ask "oracle queries" $y \in O$?
(and get answers in one time-step)

Fix cplx class C

and (other or same) cplx class D possessing complete problems

$C^D = \{$cplx class C with addition of any complete language in D as oracle$\}$

THEOREM 17

For every $i \geq 2$ it holds that

$$\Sigma_i^P = NP^{\Sigma_{i-1}^P}$$

That is, any $L \in \Sigma_i^P$ can be decided by an NDTM M that runs in poly time, makes nondeterministic choices, and asks oracle questions about the satisfiability of Σ_{i-1}SAT - formulas.

SUMMING UP

LEC 5 XVI

- Today looked at log space bounded computations
- Care needed when defining reductions and completeness [reduction must not be stronger than cplx class studied]
- NL = coNL, and in general space cplx classes closed under complement!
- Polynomial hierarchy (PH): languages verifiable by "certificates" like

  There is a u_1,
  such that for all u_2
  There is a u_3 ...,
  such that TM accepts $\langle x, u_1, u_2, \ldots \rangle$

- PH is (strictly?) between P and PSPACE
- Can also be defined by
  - alternating TMs
  - oracle TMs
- Doesn't correspond to any computational device we know of, but can be useful:
  - as technical tool to study concrete questions
  - classify problems harder than NP (e.g. is an NP-solution "optimal")

(State of the art for the lower bounds we can prove on SAT)