DD2445 COMPLEXITY THEORY LECTURE 10

WHAT WE DID LAST LECTURE

- TQBF PSPACE-complete
- PSPACE = NPSPACE
- Can simulate nondeterministic computation with quadratic blow-up in space
- Logarithmic space: L and NL
- PATH = $\{\langle G, s, t\rangle \mid \exists$ path $s \rightsquigarrow t$ in digraph $G\}$ NL-complete
- Some care needed with logarithmic space:
  - Reductions computed bit by bit
  - Witnesses only read-once in verifier-style definition
- NL = co-NL

... And somewhere here we ran out of ~~space~~ time ...

Reduction must not be stronger than complexity class reduced to.

Prove $NL = coNL$ by showing $\overline{PATH} \in NL$

Construct nondeterministic certificate (or show how NL-machine can guess successfully)

Yes-instance

$\langle G, s, t \rangle$ $\quad s \not\rightsquigarrow t \quad$ $n = |V(G)|$

Remember

$R(i) = \{$ vertices reachable from s in $\leq i$ steps $\}$

$s \not\rightsquigarrow t \iff t \notin R(\infty) \iff t \notin R(n)$

Idea

Compute $R(0) = \{s\}, R(1), R(2), \ldots, R(n-1), R(n)$

Show $t \notin R(n)$

Problem

We cannot remember (nondeterministically) $R(i)$ in log space

Only $|R(i)|$

Solution

Amazingly, this is enough!

Three subcertificates (that will be combined)

IsMember(v, i) = "$v \in R(i)$"

Just list path of length $i' = i$

MembershipExpansion(i, r, r') = "$|R(i-1)| = r' \Rightarrow |R(i)| = r$"

ListMembers(i, r) = List of r elements in $R(i)$ in increasing order, each with IsMember certificate

Full certificate:

MembershipExpansion $(1, r_1, 1)$
MembershipExpansion $(2, r_2, r_1)$
MembershipExpansion $(3, r_3, r_2)$
$\vdots$
MembershipExpansion (n, r_n, r_{n-1})
ListMembers (n, r_n)

Verification

Check that r_i is correct, keeping r_{i-1} in memory (log n space for counters) for $i = 1, 2, \ldots, n$

Finally check that t is not listed in ListMembers (n, r_n)

Done!

IsMEMBER and LISTMEMBER are clear.

MEMBERSHIP EXPANSION (i, r, r')

We already know $|R(i-1)| = r'$ (by assumption)

Give subcertificates for all vertices $j = 1, 2, \ldots, n$ in increasing order

(a) $j \in R(i)$

j: IsMEMBER (j, i) proves this increment r by one.

(b) $j \notin R(i)$

j: LISTMEMBERS $(i-1, r-1)$

Go over list

For every member u, check

(i) $u \neq j$

(ii) u does not have edge to j

After having verified all subcertificates, we know $r = |R(i)|$.

But note that for every single $j \in R(i)$, the same long certificate LISTMEMBERS $(i-1, r')$ is repeated over and over again...

Extremely wasteful.

Summing up:

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXP$$

Some inclusions must be strict

[since $L \subsetneq PSPACE$ (space hierarchy theorem)
$P \subsetneq EXP$ (time hierarchy theorem)]

But we don't know which

Probably most of them, or even all...

POLYNOMIAL HIERARCHY (AGAIN) PH I

What lies between P and PSPACE?

C circuit, α assignment

$$\text{CIRCUITEVAL} = \{\langle C, \alpha\rangle \mid C(\alpha) = 1\}$$

in $\boxed{P}$ (in fact, P-complete).

$$\text{CIRCUITSAT} = \{C \mid \exists \alpha\ C(\alpha) = 1\}$$

$\boxed{NP}$-complete

$$\text{MINSIZECIRCUIT} = \{\langle C, s\rangle \mid \exists \text{ circuit } C' \text{ of size} \le s \text{ s.t. } C' \equiv C\}$$

Two quantifiers:

$\exists$ circuit C' $\forall$ assignments α $\quad C'(\alpha) = C(\alpha)$

In gpbc class $\boxed{\Sigma_2^P}$

$L \in \Sigma_2^P$ if $\exists$ poly-time deterministic TM M
$\exists$ polynomial q
s.t. for binary strings $u_1, u_2 \in \{0,1\}^{q(|x|)}$
it holds

$$x \in L \Leftrightarrow \exists u_1\ \forall u_2\ M(x, u_1, u_2) = 1$$

More generally:

DEF POLYNOMIAL HIERARCHY

Fix $i \in \mathbb{N}^+$

$L \subseteq \{0,1\}^*$ is in Σ_i^P if

$\exists$ deterministic poly-time TM M

$\exists$ polynomial q

$x \in L \iff \exists u_1 \forall u_2 \exists u_3 \cdots Q_i u_i \; M(x, u_1, u_2, \ldots, u_i) = 1$

where all $u_i \in \{0,1\}^{q(|x|)}$

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

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

$$\Pi_i^P = \text{co}\Sigma_i^P = \{ \bar{L} : L \in \Sigma_i^P \}$$

$\Sigma_1^P = NP \qquad \Pi_1^P = \text{co}NP$

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

so $PH = \bigcup_{i \in \mathbb{N}^+} \Pi_i^P$

Is it true that

$$\Sigma_1^P \subsetneq \Sigma_2^P \subsetneq \Sigma_3^P \subsetneq \Sigma_4^P \subsetneq \cdots \; ?$$

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

Don't know, but widely believed
Standard assumption in complexity theory

THM

1. For every $i \in \mathbb{N}^+$ it holds that if $\Sigma_i^P = \Pi_i^P$, then $PH = \Sigma_i^P$ ("the polynomial hierarchy collapses to the ith level")
2. If $P = NP$, then $PH = P$ ("the polynomial hierarchy collapses to P")

Many complexity theory results have form:

Unless ⟨ statement we believe to be true ⟩ holds,
PH collapses to the ith level

Smaller i $\Rightarrow$ stronger result

Ex NP has poly-size circuits $\Rightarrow$ PH collapses to 2nd level
(so we don't believe $NP \subseteq P/poly$)

Proof

1. Was on pset 1

2. Proof by induction:
If $P = NP$, then $\Sigma_i^P = \Pi_i^P = P$

Base case ($i = 1$): Nothing to prove
By assumption $P = NP$
$coNP = coP = P$ (P closed under complement)

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 we can take complements to get $P = \Pi_i^P$.

Consider $L \in \Sigma_i^P$. Want to show $L \in P$

By def, $\exists$ (poly-time TM M and poly q such that

$x \in L \Leftrightarrow \exists u_1 \forall u_2 \cdots Q_i u_i \; M(x, u_1, \ldots, u_i) = 1$

for $u_i \in \{0,1\}^{q(|x|)}$

Define L' by

$\langle x, u_1 \rangle \in L' \Leftrightarrow \forall u_2 \exists u_3 \cdots Q_i u_i \; M(x, u_1, u_2, \ldots, u_i)$

By syntactic 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,

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

But then

$$x \in L \iff \exists u_1 \; M'(x, u_1) = 1$$

so $L \in NP$

By induction hypothesis, $L \in NP = P$.

Since $L \in \Sigma_i^P$ was arbitrary, $\Sigma_i^P \subseteq P$, QED ▨

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

- $L \in \Sigma_i^P$
- $\forall L' \in \Sigma_i^P$ it holds that $L' \leq_p L$

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

But: We believe PH is a class *without* complete languages

Lemma PH does not have complete languages unless the hierarchy collapses

Proof Suppose $\exists$ PH-complete language L.

$PH = \bigcup_{i \in \mathbb{N}^+} \Sigma_i^P$, so $\exists i^*$ s.t. $L \in \Sigma_{i^*}^P$

But then every language in PH can be reduced to $L \in \Sigma_i^P$ ▨

COROLLARY: $PH \subseteq PSPACE$ but $PH \neq PSPACE$ unless the polynomial hierarchy collapses.

Proof: If $L \in PH$, then there exists a poly-time TM M s.t. $x \in L$ iff

$\exists u_1 \forall u_2 \exists u_3 \cdots Q_i u_i \; M(x, u_1, u_2, \ldots, u_i)$

Do CIRCUIT-SAT reduction for M

Obtain QBF. Verifiable in PSPACE

(Or argue from first principles)

PSPACE has complete problems (TQBF, for instance). So if $PSPACE = PH$, PH has complete problems and the hierarchy collapses.

Complete problems for Σ_i^P

Σ_i-SAT $= \{ \psi \mid \psi = \exists u_1 \forall u_2 \exists u_3 \cdots Q_i u_i \cdot \varphi(u_1, \ldots, u_i) \}$

where φ propositional formula

For Σ_{2i+1}-SAT can let φ be CNF formula.

For Σ_{2i}-SAT not (why? Good exercise.)

Π_i-SAT defined similarly

(and φ can be CNF for i even)

ALTERNATING TURING MACHINES

A NDTM M is "on its way" to complete an accepting computation if ∃ path from current configuration to accepting configuration.

Such a machine can solve SAT

We could also imagine a "co-NDTM" M' that accepts if all paths from current configuration accepts — would solve UNSAT

For MINSIZECIRCUIT, could use TM that combines both approaches:

- Start in "existential mode" to guess small circuit
- Then switch to "universal mode" to check circuit equivalence on all assignments

Such a machine could decide Σ_2^P - languages.

DEF (Informal; check Arora-Barak for full details)
An alternating TM has two transition function and every state labelled ∃ or ∀.

- Accepting computation from ∃-state if exists one path to accepting state
- Accepting computation from ∀-state if all paths 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 $\exists$- to $\forall$- or from $\forall$ to $\exists$-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 $\exists$
- along all computation paths, at most $i-1$ alternations.

Π_i TIME (T(n)).

The same, except starting state labelled $\forall$

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 L 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.

Not entirely trivial — see Arora-Barak for details.

SUMMING UP

- POLYNOMIAL HIERARCHY: Languages verifiable by "certificates" like

  There is a subwitness u_1
  s.t. for all challenges a_2
  there is a subwitness u_3 ...
  ... such that TM accepts $\langle x, u_1, \ldots, u_i \rangle$

- PH is (strictly?) between P and PSPACE
- PH 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 (state of the art for lower bounds that can be proven on SAT algorithms)
  - classify problems harder than NP (e.g. is an NP-solution "optimal")