DD2445 COMPLEXITY THEORY: LECTURE 8

Polynomial hierarchy – beyond P, NP, and coNP

For $i \in \mathbb{N}^+$, a language L is in Σ_i^p if
$\exists$ deterministic poly-time TM M
$\exists$ polynomial q
such that

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

where all $u_j \in \{0,1\}^{q(|x|)}$
$Q_i = \exists$ for i odd, $\forall$ for i even

Polynomial hierarchy

$$PH = \bigcup_{i=1}^{\infty} \Sigma_i^p$$

$$\Pi_i^p = co\Sigma_i^p = \{ L \mid \overline{L} \in \Sigma_i^p \}$$

<u>Hypothesis</u>

"The polynomial hierarchy doesn't collapse"

$$\Sigma_1^p \subsetneq \Sigma_2^p \subsetneq \Sigma_3^p \subsetneq \cdots$$

Theorems on the form
"Unless ⟨statement S⟩ holds, the polynomial hierarchy collapses"
give circumstantial evidence that statement S is true ("If pigs can fly, then horses can whistle")

PHc I

DEF Language $\mathcal{L}$ is Σ_i^P-COMPLETE if

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

Π_i^P-complete and PH-complete languages defined anologously

But: We believe PH is a complexity class *without* complete problems/languages.

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

Proof Suppose $\exists$ PH-complete language $\mathcal{L}$

$$PH = \bigcup_{i=1}^{\infty} \Sigma_i^P$$

So there is some i^* such that $\mathcal{L} \in \Sigma_{i^*}^P$

By completeness, every language $\mathcal{L}'$ in PH can be reduced to $\mathcal{L} \in \Sigma_{i^*}^P$, so $PH = \Sigma_i^P$ ☺

More details: $x \in \mathcal{L}' \Leftrightarrow f(x) \in \mathcal{L}$

$$\Updownarrow$$

$$\exists u_1 \forall u_2 \ldots Q_{i^*} u_{i^*} \quad M_{\mathcal{L}}(f(x), u_1, \ldots, u_{i^*}) = 1$$

$f(x)$ computable in poly-time by TM M_f

Let M_{L}' be TM that computes $x \mapsto f(x)$ and then runs $M_L(f(x), u_1, \ldots, u_{i*})$

Then

$$x \in L' \\ \Updownarrow \\ \exists u_1 \forall u_2 \exists u_3 \ldots Q_i u_i \; M_L'(x, u_1, u_2, \ldots, u_{i*}) = 1$$

So $PH \subseteq \Sigma_i^P$ ▨

<u>COROLLARY</u> It holds that $PH \subseteq PSPACE$ but unless the polynomial hierarchy collapses we have $PH \neq PSPACE$.

<u>Proof</u> By definition, if $L \in PH$, then exists i and poly-time TM M s.t.
$x \in L$ iff $\exists u_1 \forall u_2 \exists u_3 \cdots Q_i u_i \; M(x, u_1, u_2, u_3, \ldots, u_i) = 1$

Look at computation $M(x, u_1, u_2, u_3, \ldots, u_i)$ given fixed x

Run Cook-Levin reduction to get CNF φ which is satisfied iff $M(x, u_1, u_2, u_3, \ldots, u_i) = 1$

Now get QBF

$$\exists u_1 \forall u_2 \exists u_3 \ldots Q_i u_i \; \varphi$$

Can check in PSPACE whether this QBF is true. Hence $PH \subseteq PSPACE$.

But PSPACE has complete problems (TQBF, for instance). So if $PH = PSPACE$, then the polynomial hierarchy collapses since PH has complete problems.

But Σ_i^P has complete problems

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

u_j – sets of variables
φ – Boolean formula
Say φ CNF if i odd
φ DNF if i even (Why?)

$\Pi_i SAT$ defined in the same way (and φ CNF if i even, DNF if i odd)

Proof of completeness? Exercise (Maybe on problem set)

ALTERNATING TURING MACHINES

A NDTM M is "on its way" to complete an accepting computation if $\exists$ 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 MIN ~~CNF~~ SIZE, could use TM that combines both approaches:

- Start in "existential mode" to guess small formula
- Then switch to "universal mode" to check formula 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 $\exists$ or $\forall$.

- Accepting computation from $\exists$-state if exists one path to accepting state
- Accepting computation from $\forall$-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

(Also a bit informal)

DEF

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

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

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

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

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 $\mathcal{C}$

and (other or same) cplx class $\mathcal{D}$ possessing complete problems

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

Why don't we care which language?

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 u_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)
  - to classify problems harder than NP (e.g. is an NP-solution "optimal")
  - to obtain circumstantial evidence for other claims (if we believe the polynomial hierarchy doesn't collapse)

BOOLEAN CIRCUITS

- Studied since 1940s
- Shannon: Circuit minimization (Σ_2^P-problem)
- Connection to P vs NP soon made in 1970s
- Circuits are <u>explicit</u> combinatorial objects — easier to reason about than Turing machines?

<u>DEF</u>
- Directed acyclic graph (DAG)
- n inputs/sources labelled by variables $x_1, x_2, \ldots, x_n$
- one sink/output
- Internal (non-source) vertices labelled by

<u>Not</u> true for real-world circuits

GATES

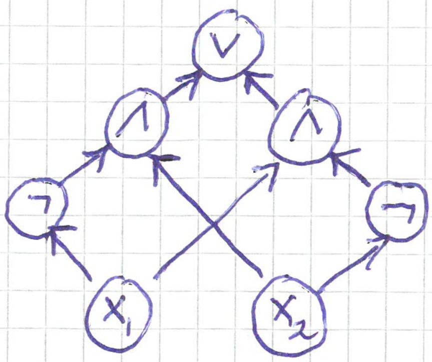
$\wedge$	AND	fan-in 2
$\vee$	OR	fan-in 2
$\neg$	NOT	fan-in 1

- size = # vertices
- depth = length of longest path in DAG

Value computed by circuit on $x \in \{0,1\}^n$
Defined bottom-up in natural way
Sort vertices in some topological order
Compute values of gates in this order
(values feeding into gate always defined)
Output of circuit = value of sink

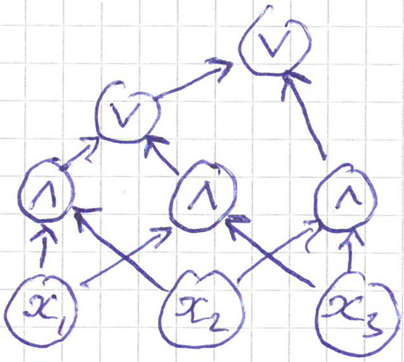
Can also define multi-output circuits by considering DAGs with several sinks

EXAMPLE A



x_1 XOR x_2 = exactly one of x_1 & x_2 true

EXAMPLE B



MAJ(x_1, x_2, x_3) = what is the most common bit value?

How do we decide problems/languages

A circuit is a fixed object with fixed number of inputs

Languages (typically) contain strings of different lengths

Consider <u>families</u> of circuits

DEF 4 $T : \mathbb{N} \to \mathbb{N}$ function

A $T(n)$-size circuit family is a sequence $\{C_n\}_{n \in \mathbb{N}}$ of Boolean circuits where $\forall n$ C_n has n inputs, one output, and size $|C_n| \leq \underbrace{T(n)}_{k\cdot}$

for some constant k

$L \in SIZE(T(n))$ if $\exists$ $T(n)$-size $\{C_n\}_{n \in \mathbb{N}}$ s.t. $L = \{x \mid C_n(x) = 1 \text{ for } n = |x|\}$

EX 5 $\{1^n : n \in \mathbb{N}\} \in SIZE(n)$

Build tree of AND-gates

Proved in lec 2: Any $f : \{0,1\}^n \to \{0,1\}$ computed by CNF of size $n 2^n$

In fact, can do $O(2^n/n)$.

More interesting: poly-size circuits

DEF 6 $P/poly = \bigcup_{c \in \mathbb{N}^+} SIZE(n^c)$ "P slash poly"

Why this name? Will get back to it later today.

THM 7 $P \subseteq P_{/poly}$

BTW Arora-Barak $P_{/poly}$ subscript
Also common $P/poly$

Proof sketch Similar to Cook-Levin.

Given poly-time TM M

Make oblivious TM $\tilde{M}$ with quadratic time blow-up

Given oblivious $T(n)$-time $\tilde{M}$, construct $\{C_n\}_{n \in \mathbb{N}}$

s.t. $C_n(x) = M(x)$ $\forall x \in \{0,1\}^n$

Transcript of M on x

Sequence of snapshots $z_1, z_2, \dots, z_{T(n)}$
- machine state
- symbols read

Snapshot z_i: constant # bits

depends on constant # bits
- previous snapshot z_{i-1}
- last times current tape positions visited $z_{i_1}, \dots, z_{i_k}$

$\Rightarrow \exists$ constant-size circuit computing z_i from $z_{i-1}, z_{i_1}, \dots, z_{i_k}$

Glue such circuits together and check $z_{T(n)}$ ~~acce~~ accepting. $\boxed{\text{LM}}$

LEMMA 8 $P \neq P_{/poly}$

Any unary language $L \subseteq \{1^n : n \in \mathbb{N}\}$ is in $P_{/poly}$ (use idea from Ex 5)

Let

$\text{UHALT} = \left\{ 1^n \;\middle|\; \begin{array}{l}\text{binary expansion of } n \text{ encodes } \langle M, x\rangle \\ \text{such that } M \text{ halts on input } x\end{array} \right\}$

Boolean circuits can be used to prove Cook–Levin

DEF 9 $\text{CIRCUITSAT} = \left\{ x \;\middle|\; \begin{array}{l} x \text{ describes a circuit } C \text{ and} \\ \exists u \text{ s.t. } C(u) = 1 \end{array} \right\}$

LEMMA 10 CIRCUITSAT is NP-complete

Proof Clearly in NP.

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

Run proof of Thm 7 to get C s.t. $M(x,u) = C(u)$ (Hard-wire value x)

BC IV

LEMMA 11 CIRCUITSAT $\leq_p$ 3-SAT

Proof idea For every circuit gate/node v_i introduce variable z_i

Write clauses enforcing that z_i is correctly computed given inputs

Say $v_i = v_j$ AND v_k. Get clauses:

$$(\overline{z}_j \vee \overline{z}_k \vee z_i)$$
$$\wedge (z_j \vee \overline{z}_i)$$
$$\wedge (z_k \vee \overline{z}_i)$$

Finally add unit clause z_i for output node v_i $\boxed{}$

UHALT $\in P_{/poly}$ is a bit weird. We have no way of constructing these circuits. Would be reasonable to require as well?

DEF 12 $\{C_n\}_{n \in \mathbb{N}}$ is P-uniform if $\exists$ poly-time TM M that on input 1^n outputs (description of) C_n.

More reasonable — but not so fun. Collapses $P_{/poly}$ to P

LEM 13 L is computable by P-uniform circuit family iff $L \in P$

Proof sketch

($\Rightarrow$) Given x, run M on $1^{|x|}$ to obtain $C_{|x|}$ and then evaluate $C_{|x|}$ on x.

($\Leftarrow$) Go over proof of Thm 7 carefully to check that construction is P-uniform.

Turing machines that take advice

Given x, look at $n = |x|$

Get advice string α_n which depends on n

(but is common for all x with $|x| = n$)

DEF 14 $T : \mathbb{N} \to \mathbb{N}$ $a : \mathbb{N} \to \mathbb{N}$ functions

$DTIME(T(n))/a(n)$ contains every L s.t.

$\exists \{\alpha_n\}_{n \in \mathbb{N}^+}$, $\alpha_n \in \{0,1\}^{a(n)}$,

$\exists$ TM M

s.t. $x \in L$ iff $M(x, \alpha_n) = 1$

and M runs in time $O(T(n))$.

THM 15 $P_{/poly} = \bigcup_{c,d} DTIME(n^c)/n^d$

($\Longrightarrow$) Suppose $L \in P_{/poly}$

Let advice string be description of $C_{|x|}$

($\Longleftarrow$) Use proof of Thm 7 to construct poly-sized circuit D_n s.t. $D_n(x, \alpha) = M(x, \alpha)$

Let $C_n = D_n$ with α_n hardwired as 2nd input.

Could it be that CNFSAT $\notin$ P but for all formulas φ of size s there is some really useful advice α_s so that we can solve φ given α_s ?!

That is, can it be that $NP \subseteq P_{/poly}$?

NO. Not unless the polynomial hierarchy collapses.

$\boxed{\begin{array}{l}\text{THEOREM [Karp-Lipton '80]} \\ \text{If } NP \subseteq P/poly, \text{ then } PH = \Sigma_2^P\end{array}}$

How to prove this?

By previous lecture, sufficient to show

$$\Sigma_2^P = \Pi_2^P$$

In fact, suffices to show $\Pi_2^P \subseteq \Sigma_2^P$ (why?)

Take Π_2^P-complete language L

Show that $L \in \Sigma_2^P$

$\Pi_2 SAT$ = All true formulas ψ of form

$$\psi = \forall u \in \{0,1\}^n \exists v \in \{0,1\}^n \varphi(u,v) = 1$$

$\Pi_2 SAT$ is in Σ_2^P if there is a poly-time TM M s.t.

$$\psi \in \Pi_2 SAT \iff \exists u^* \in \{0,1\}^{q(|\psi|)} \forall v^* \in \{0,1\}^{q(|\psi|)} \; M(\psi, u^*, v^*) = 1$$

I.e., flip $\forall$- and $\exists$-quantifier

How?!

Have to use assumption

$$NP \subseteq P/poly$$

Full proof will have to wait till ~~next~~ lecture