TECHNISCHE UNIVERSITÄT DRESDEN

COMPLEXITY THEORY

Lecture 18: Questions and Answers

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 20th Dec 2017

Polynomial Hierarchy: Mostly Questions

The Polynomial Hierarchy Three Ways

We discovered a hierarchy of complexity classes between P and PSpace, with NP and coNP on the first level, and infinitely many further levels above:

Definition by ATM: Classes $\Sigma_i^{\mathrm{P}}/\Pi_i^{\mathrm{P}}$ are defined by polytime ATMs with bounded types of alternation, starting computation with existential/universal states.

Definition by Verifier: Classes $\Sigma_i^{\mathrm{P}}/\Pi_i^{\mathrm{P}}$ are given as projections of certain verifier languages in P, requiring existence/universality of polynomial witnesses.

Definition by Oracle: Classes $\Sigma_i^{\mathrm{P}}/\Pi_i^{\mathrm{P}}$ are defined as languages of NP/coNP oracle TMs with $\Sigma_{i-1}^{\mathrm{P}}$ (or, equivalently, Π_{i-1}^{P}) oracle.

Using such oracles with determinstiic TMs, we can also define classes Δ_i^{P}.

Is the Polynomial Hierarchy Real?

Questions:

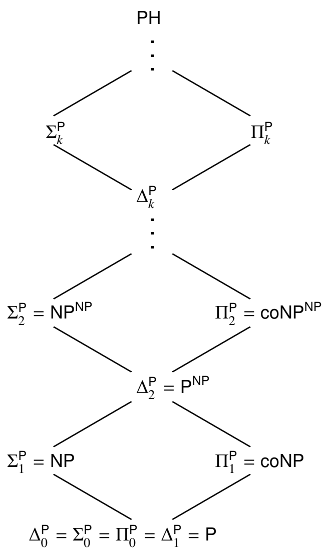
Are all of these classes really distinct?
Nobody knows.

Are any of these classes really distinct?
Nobody knows.

Are any of these classes distinct from P?
Nobody knows.

Are any of these classes distinct from PSpace?
Nobody knows.

What do we know then?



What We Know (Excerpt)

Theorem 18.1: If there is any k such that $\Sigma_k^{\mathsf{P}} = \Sigma_{k+1}^{\mathsf{P}}$ then $\Sigma_j^{\mathsf{P}} = \Pi_j^{\mathsf{P}} = \Sigma_k^{\mathsf{P}}$ for all $j > k$, and therefore $\mathsf{PH} = \Sigma_k^{\mathsf{P}}$.
In this case, we say that the polynomial hierarchy collapses at level k.

Proof: Left as exercise (not too hard to get from definitions). □

Corollary 18.2: If PH ≠ P then NP ≠ P.

Intuitively speaking: "The polynomial hierarchy is built upon the assumption that NP has some additional power over P. If this is not the case, the whole hierarchy collapses."

What We Know (Excerpt)

Theorem 18.3: PH ⊆ PSpace.

Proof: Left as exercise (induction over PH levels, using that $\mathsf{PSpace}^{\mathsf{PSpace}} = \mathsf{PSpace}$). □

Theorem 18.4: If PH = PSpace then there is some k with $\mathsf{PH} = \Sigma_k^{\mathsf{P}}$.

Proof: If PH = PSpace then **True QBF** ∈ PH. Hence **True QBF** $\in \Sigma_k^{\mathsf{P}}$ for some k. Since **True QBF** is PSpace-hard, this implies $\Sigma_k^{\mathsf{P}} = \mathsf{PSpace}$. □

What We Believe (Excerpt)

"Most experts" think that:

- The polynomial hierarchy does not collapse completely (same as P ≠ NP)
- The polynomial hierarchy does not collapse on any level
  (in particular PH ≠ PSpace and there is no PH-complete problem)

But there can always be surprises …

Question 1: The Logarithmic Hierarchy

Q1: The Logarithmic Hierarchy

The Polynomial Hierarchy is based on polynomially time-bounded TMs

It would also be interesting to study the Logarithmic Hierarchy obtained by considering logarithmically space-bounded TMs instead, wouldn't it?

In detail, we can define:

- $\Sigma_0^{\mathsf{L}} = \Pi_0^{\mathsf{L}} = \mathsf{L}$
- $\Sigma_{i+1}^{\mathsf{L}} = \mathsf{NL}^{\Sigma_i^{\mathsf{L}}}$ alternatively: languages of log-space bounded Σ_{i+1} ATMs
- $\Pi_{i+1}^{\mathsf{L}} = \mathsf{coNL}^{\Sigma_i^{\mathsf{L}}}$ alternatively: languages of log-space bounded Π_{i+1} ATMs

Q1: What is the Logarithmic Hierarchy?

How do the levels of this hierarchy look?

- $\Sigma_0^{\mathsf{L}} = \Pi_0^{\mathsf{L}} = \mathsf{L}$
- $\Sigma_1^{\mathsf{L}} = \mathsf{NL}^{\mathsf{L}} = \mathsf{NL}$
- $\Pi_1^{\mathsf{L}} = \mathsf{coNL}^{\mathsf{L}} = \mathsf{coNL} = \mathsf{NL}$ (why?)
- $\Sigma_2^{\mathsf{L}} = \mathsf{NL}^{\Sigma_1^{\mathsf{L}}} = \mathsf{NL}^{\mathsf{NL}} = \mathsf{NL}$ (why?)
- $\Pi_2^{\mathsf{L}} = \mathsf{coNL}^{\Sigma_1^{\mathsf{L}}} = \mathsf{coNL}^{\mathsf{NL}} = \mathsf{NL}$ (why?)

Therefore $\Sigma_i^{\mathsf{L}} = \Pi_i^{\mathsf{L}} = \mathsf{NL}$ for all $i \geq 1$.

The Logarithmic Hierarchy collapses on the first level.

Historic note: In 1987, just before the Immerman-Szelepcsényi Theorem was published, Klaus-Jörn Lange, Birgit Jenner, and Bernd Kirsig showed that the Logarithmic Hierarchy collapses on the second level [ICALP 1987].

Question 2: The Hardest Problems in P

Q2: The hardest problems in P

What we know about P and NP:

- We don't know if any problem in NP is really harder than any problem in P.
- But we do know that NP is at least as challenging as P, i.e., $\mathsf{P} \subseteq \mathsf{NP}$.

So all problems that are hard for NP are also hard for P, aren't they?

Q2: Is NP-hard as hard as P-hard?

Let's first recall the definitions:

Definition: A problem **L** is NP-hard if, for all problems **M** $\in$ NP, there is a polynomial many-one reduction **M** $\leq_m$ **L**.

Definition: A problem **L** is P-hard if, for all problems **M** $\in$ P, there is a log-space reduction **M** $\leq_L$ **L**.

How to show "NP-hard implies P-hard"?

- Assume that **L** is NP-hard.
- Consider any language **M** $\in$ P.
- Then **M** $\in$ NP.
- So there is a polynomial many-one reduction f from **M** to **L**
- Hence, ... well..., nothing much, really.

Q2: Is NP-hard as hard as P-hard?

For all we know today, it is perfectly possible that there are NP-hard problems that are not P-hard.

Example 18.5: We know that L $\subseteq$ P $\subseteq$ NP but we do not know if any of these subsumptions are proper. Suppose that the truth actually looks like this: L $\subsetneq$ P = NP. Then all non-trivial problems in P are NP-hard (why?), but not every problem would be P-hard (why?).

Note: This is really about the different notions of reduction used to define hardness. If we used log-space reductions for P-hardness and NP-hardness, the claim would follow.

Question 3: Problems Harder than P

Q3: Problems harder than P

Polynomial time is an approximation of "practically tractable" problems:

- Many practical problems are in P, including many very simple ones (e.g., $\emptyset$)
- P-hard problems are as hard as any other problem in P, and P-complete problems therefore are the hardest problems in P
- However, there are even harder problems that are no longer in P

So, clearly, problems that are not even in P must be P-hard, right?

Q3: Are problems harder than P also hard for P?

Can we find any problem that is surely harder than P? Yes, easily:

- The Halting Problem is undecidable and therefore not in P
- Any ExpTime-complete problem is not in P (Time Hierarchy Theorem); e.g., the Word Problem for exponentially time-bounded DTMs

These concrete examples both are hard for P:

- The Word Problem for polynomially time-bounded DTMs is hard for P
- This polytime Word Problem log-space reduces to the Word Problem for exponential TMs (reduction: the identity function)
- It also log-space reduces to the Halting problem: a reduction merely has to modify the TM so that every rejecting halting configuration leads into an infinite loop

Q3: Are problems harder than P also hard for P?

Rephrasing the question: Are there problems that are not in P, yet not hard for P?

Some observations:

- $\emptyset$ is not P-hard (why?)
- Any ExpTime-complete problem **L** is not in P (why?)
- We can enumerate DTMs for all languages in P (how?)
- We can enumerate DTMs for all P-hard languages in ExpTime (how?)

So, it's clear what we have to do now . . .

Q4: Are decidable problems harder than P also hard for P?

Schöning to the rescue (see Theorem 15.2):

Corollary 18.6: Consider the classes C_1 = ExpPHard (P-hard problems in ExpTime) and C_2 = P. Both are classes of decidable languages. We find that for either class C_k:

- We can effectively enumerate TMs $\mathcal{M}_0^k, \mathcal{M}_1^k, \ldots$ such that $C_k = \{\mathbf{L}(\mathcal{M}_i^k) \mid i \geq 0)\}$.
- If $\mathbf{L} \in C_k$ and $\mathbf{L'}$ differs from $\mathbf{L}$ on only a finite number of words, then $\mathbf{L'} \in C_k$

Let $\mathbf{L_1} = \emptyset$, and let $\mathbf{L_2}$ be some ExpTime-complete problem. Clearly, $\mathbf{L_1} \notin$ ExpPHard and $\mathbf{L_2} \notin$ P (Time Hierarchy), hence there is a decidable language $\mathbf{L_d} \notin \text{ExpPHard} \cup \text{P}$.
Moreover, as $\emptyset \in$ P and $\mathbf{L_2}$ is not trivial, $\mathbf{L_d} \leq_p \mathbf{L_2}$ and hence $\mathbf{L_d} \in$ ExpTime. Therefore $\mathbf{L_d} \notin$ ExpPHard implies that $\mathbf{L_d}$ is not P-hard.

This idea of using Schöning's Therorem has been put forward by Ryan Williams (link). Our version is a modification requiring $C_1 \subseteq$ ExpTime.

Q3: Are problems harder than P also hard for P?

No, there are problems in ExpTime that are neither in P nor hard for P.

(Other arguments can even show the existence of undecidable sets that are not P-hard)

Discussion:

- Considering Questions 3 and 4, the use of the word hard is misleading, since we interpret it as difficult
- However, the actual meaning difficult would be "not in a given class" (e.g., problems not in P are clearly more difficult than those in P)
- Our formal notion of hard also implies that a problem is difficult in some sense, but it also requires it to be universal in the sense that many other problems can be solved through it

What we have seen is that there are difficult problems that are not universal.

Your Questions

Summary and Outlook

We do not know if the Polynomial Hierarchy is real or collapses

Answer 1: The Logarithmic Hierarchy collapses.

Answer 2: We don't know that NP-hard imples P-hard.

Answer 3: Being outside of P does not make a problem P-hard.

What's next?

- Holidays
- Circuits as an alternative model of computation
- Randomness

Here's wishing you

a Merry Christmas, a Happy Hanukkah,

a Joyous Yalda, a Cheerful Dōngzhì,

a Great Feast of Juul,

and a Wonderful Winter Solstice,

respectively!