Theory of Computer Science

D6. Beyond NP

Gabriele Röger

University of Basel

May 17, 2023

- Complexity theory investigates which problems are "easy" to solve and which ones are "hard".
- two important problem classes:
 - P: problems that are solvable in polynomial time by "normal" computation mechanisms
 - NP: problems that are solvable in polynomial time with the help of nondeterminism
- We know that $P \subseteq NP$, but we do not know whether P = NP.
- Many practically relevant problems are NP-complete:
 - They belong to NP.
 - All problems in NP can be polynomially reduced to them.
- If there is an efficient algorithm for one NP-complete problem, then there are efficient algorithms for all problems in NP.



coNP

Complexity Class coNP

Definition (coNP)

coNP is the set of all languages L for which $\bar{L} \in NP$.

Example: The complement of SAT is in coNP.

Hardness and Completeness

Definition (Hardness and Completeness)

Let C be a complexity class.

A problem Y is called C-hard if $X \leq_p Y$ for all problems $X \in C$.

Y is called C-complete if $Y \in C$ and Y is C-hard.

Example (TAUTOLOGY)

The following problem **TAUTOLOGY** is coNP-complete:

Given: a propositional logic formula φ

Question: Is φ valid, i.e. is it true under all variable assignments?

Known Results and Open Questions

Open

■ NP $\stackrel{?}{=}$ coNP

Known

- $P \subseteq coNP$
- If X is NP-complete then \bar{X} is coNP-complete.
- If $NP \neq coNP$ then $P \neq NP$.
- If a coNP-complete problem is in NP, then NP = coNP.
- If a coNP-complete problem is in P, then P = coNP = NP.

Time and Space Complexity

Definition (Time Complexity Classes TIME and NTIME)

Let $t : \mathbb{N} \to \mathbb{R}^+$ be a function.

The time complexity class TIME(t(n)) is the collection of all languages that are decidable by an O(t) time Turing machine, and NTIME(t(n)) is the collection of all languages that are decidable by an O(t) time nondeterministic Turing machine.

- TIME(f): all languages accepted by a DTM in time f.
- NTIME(f): all languages accepted by a NTM in time f.
- $P = \bigcup_{k \in \mathbb{N}} \mathsf{TIME}(n^k)$
- NP = $\bigcup_{k \in \mathbb{N}} \mathsf{NTIME}(n^k)$

- Analogously: A TM decides a language L in space f if the computation on every input visits at most f(|w|) tape cells besides it input on the tape.
- **SPACE**(f): all languages decided by a DTM in space f.
- NSPACE(f): all languages decided by a NTM in space f.

Important Complexity Classes Beyond NP

- PSPACE = $\bigcup_{k \in \mathbb{N}} SPACE(n^k)$
- NPSPACE = $\bigcup_{k \in \mathbb{N}} NSPACE(n^k)$
- EXPTIME = $\bigcup_{k \in \mathbb{N}} \mathsf{TIME}(2^{n^k})$
- EXPSPACE = $\bigcup_{k \in \mathbb{N}} SPACE(2^{n^k})$

Some known results:

- PSPACE = NPSPACE (from Savitch's theorem)
- PSPACE ⊂ EXPTIME ⊂ EXPSPACE (at least one relationship strict)
- P \neq EXPTIME, PSPACE \neq EXPSPACE
- \blacksquare P \subset NP \subset PSPACE

Polynomial Hierarchy

Oracle Machines

An oracle machine is like a Turing machine that has access to an oracle which can solve some decision problem in constant time.

Example oracle classes:

- NP^{NP} = { $L \mid L$ can get decided in pol. time by a NTM with an oracle deciding some problem in NP}

Polynomial Hierarchy

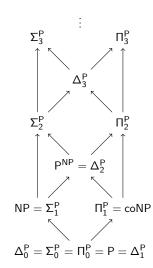
Inductively defined:

$$\Delta_0^P := \Sigma_0^P := \Pi_0^P := P$$

$$\quad \blacksquare \ \Sigma_{i+1}^P := \mathsf{NP}^{\Sigma_i^P}$$

$$\quad \blacksquare \ \Pi_{i+1}^{\mathsf{P}} := \mathsf{coNP}^{\Sigma_i^{\mathsf{P}}}$$

$$PH := \bigcup_{k} \Sigma_{k}^{P}$$



Polynomial Hierarchy: Results

- PH \subseteq PSPACE (PH $\stackrel{?}{=}$ PSPACE is open)
- There are complete problems for each level.
- If there is a PH-complete problem, then the polynomial hierarchy collapses to some finite level.
- If P = NP, the polynomial hierarchy collapses to the first level.

Counting



Complexity class #P (pronounced "Sharp P")

■ Set of functions $f: \{0,1\}^* \to \mathbb{N}_0$, where f(n) is the number of accepting paths of a polynomial-time NTM

Example (#SAT)

The following problem #SAT is #P-complete:

Given: a propositional logic formula φ

Question: Under how many variable assignments is φ true?

What's Next?

contents of this course:

- A. background √▷ mathematical foundations and proof techniques
- B. automata theory and formal languages √b What is a computation?
- C. Turing computability √b What can be computed at all?
- D. complexity theory▷ What can be computed efficiently?
- E. more computability theory

 ▷ Other models of computability

What's Next?

contents of this course:

- A. background √▷ mathematical foundations and proof techniques
- B. automata theory and formal languages √b What is a computation?
- C. Turing computability √b What can be computed at all?
- D. complexity theory √b What can be computed efficiently?
- E. more computability theory

 ▷ Other models of computability