

Theory of Computer Science

D6. Beyond NP

Gabriele Röger

University of Basel

May 17, 2021

Theory of Computer Science

May 17, 2021 — D6. Beyond NP

D6.1 coNP

D6.2 Time and Space Complexity

D6.3 Polynomial Hierarchy

D6.4 Counting

Complexity Theory: What we already have seen

- ▶ **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.

D6.1 coNP

Complexity Class coNP

Definition (coNP)

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

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

- $\text{NP} \stackrel{?}{=} \text{coNP}$

Known

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

D6.2 Time and Space Complexity

Reminder: Time Complexity Classes

Definition (Time Complexity Classes TIME and NTIME)

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

The **time complexity class $\text{TIME}(t(n))$** 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 .
- ▶ $P = \bigcup_{k \in \mathbb{N}} \text{TIME}(n^k)$
- ▶ $NP = \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$

Space

- ▶ **Analogously**: A TM decides a language L in **space f** if the computation on every input visits at most $f(|w|)$ tape cells besides its 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

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

Some known results:

- ▶ $\text{PSPACE} = \text{NPSPACE}$ (from Savitch's theorem)
- ▶ $\text{PSPACE} \subseteq \text{EXPTIME} \subseteq \text{EXPSPACE}$
(at least one relationship strict)
- ▶ $P \neq \text{EXPTIME}$, $\text{PSPACE} \neq \text{EXPSPACE}$
- ▶ $P \subseteq \text{NP} \subseteq \text{PSPACE}$

D6.3 Polynomial Hierarchy

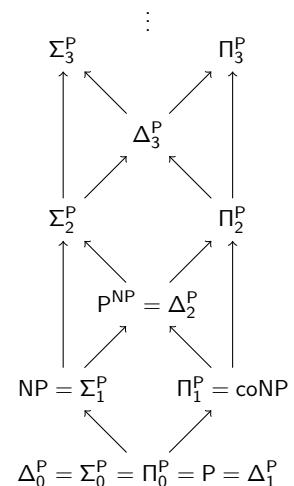
Oracle Machines

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:

- ▶ $P^{NP} = \{L \mid L \text{ can get decided in polynomial time by a DTM with an oracle that decides some problem in NP}\}$
- ▶ $NP^{NP} = \{L \mid L \text{ can get decided in pol. time by a NTM with an oracle deciding some problem in NP}\}$

Polynomial Hierarchy



Inductively defined:

- ▶ $\Delta_0^P := \Sigma_0^P := \Pi_0^P := P$
- ▶ $\Delta_{i+1}^P := P^{\Sigma_i^P}$
- ▶ $\Sigma_{i+1}^P := NP^{\Sigma_i^P}$
- ▶ $\Pi_{i+1}^P := coNP^{\Sigma_i^P}$
- ▶ $PH := \bigcup_k \Sigma_k^P$

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.

D6.4 Counting

#P

Complexity class #P

- ▶ Set of functions $f : \{0, 1\}^* \rightarrow \mathbb{N}_0$, 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** ✓
▷ What is a computation?
- C. **Turing computability** ✓
▷ What can be computed at all?
- D. **complexity theory** ✓
▷ What can be computed efficiently?
- E. **more computability theory**
▷ Other models of computability