

Theory of Computer Science

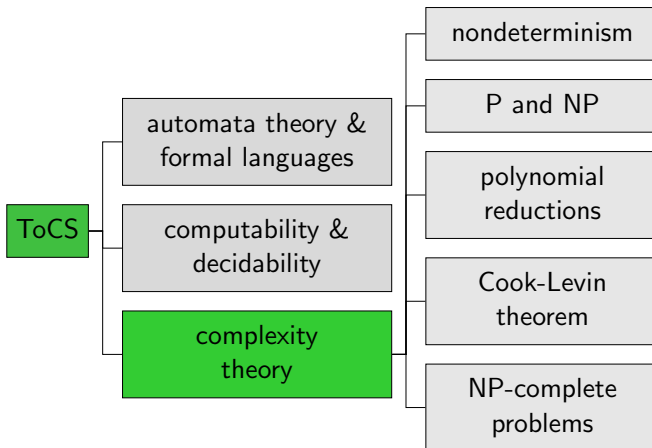
D6. Beyond NP

Gabriele Röger

University of Basel

May 21, 2025

Content of the Course



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.

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

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

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

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 $\text{NTIME}(t(n))$ is the collection of all languages that are decidable by an $O(t)$ **time nondeterministic Turing machine**.

- $\text{TIME}(f)$: all languages accepted by a **DTM** in time f .
- $\text{NTIME}(f)$: 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)
- $\text{P} \neq \text{EXPTIME}$, $\text{PSPACE} \neq \text{EXPSPACE}$
- $\text{P} \subseteq \text{NP} \subseteq \text{PSPACE}$

Counting

#P

Complexity class **#P** (pronounced “Sharp 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?

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

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

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