Theory of Computer Science

E6. Beyond NP

Gabriele Röger

University of Basel

May 25, 2020

Theory of Computer Science

May 25, 2020 — E6. Beyond NP

E6.1 coNP

E6.2 Time and Space Complexity

E6.3 Polynomial Hierarchy

E6.4 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 $\mathsf{P} \subseteq \mathsf{NP}$, but we do not know whether $\mathsf{P} = \mathsf{NP}$.
- 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.

E6.1 coNP

Complexity Class coNP

Definition (coNP)

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

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 $X \leq_{\mathrm{p}} Y$ for all problems $X \in \mathrm{C}$.

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

Example (Tautology)

The following problem Tautology is coNP-complete:

Given: a propositional logic formula φ

Question: Is φ valid?

Known Results and Open Questions

Open

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

Known

- P $\subseteq$ coNP
- If X is NP-complete then $\bar{L}$ 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.

E6.2 Time and Space Complexity

Time

Definition (Reminder: Accepting a Language in Time f)

Let M be a DTM or NTM with input alphabet Σ, $L \subseteq \Sigma^*$ a language and $f : \mathbb{N}_0 \to \mathbb{N}_0$ a function.

M accepts L in time f if:

1. for all words $w \in L$: M accepts w in time $f(|w|)$
2. for all words $w \notin L$: M does not accept w

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

Space

- Analogously: A TM accepts a language L in space f if every word $w \in L$ gets accepted using at most of $f(|w|)$ space besides it input on the tape and no $w \notin L$ gets accepted.
- SPACE(f): all languages accepted by a DTM in space f.
- NSPACE(f): all languages accepted by a NTM in space f.

Important Complexity Classes Beyond NP

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

Some known results:

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

E6.3 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:

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

Polynomial Hierarchy

Inductively defined:

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

$$\begin{tikzcd}
& \vdots & \\
\Sigma_3^{\mathrm{P}} & & \Pi_3^{\mathrm{P}} \\
& \Delta_3^{\mathrm{P}} \arrow[ul] \arrow[ur] & \\
\Sigma_2^{\mathrm{P}} \arrow[uu] \arrow[ur] & & \Pi_2^{\mathrm{P}} \arrow[uu] \arrow[ul] \\
& \mathrm{P}^{\mathrm{NP}} = \Delta_2^{\mathrm{P}} \arrow[ul] \arrow[ur] & \\
\mathrm{NP} = \Sigma_1^{\mathrm{P}} \arrow[uu] \arrow[ur] & & \Pi_1^{\mathrm{P}} = \mathrm{coNP} \arrow[uu] \arrow[ul] \\
& \Delta_0^{\mathrm{P}} = \Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathrm{P} = \Delta_1^{\mathrm{P}} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

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.

E6.4 Counting

#P

Complexity class #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: How many models does φ have?

What's Next?

contents of this course:

A. background ✓
  - ▷ mathematical foundations and proof techniques

B. logic ✓
  - ▷ How can knowledge be represented?
    How can reasoning be automated?

C. automata theory and formal languages ✓
  - ▷ What is a computation?

D. Turing computability ✓
  - ▷ What can be computed at all?

E. complexity theory ✓
  - ▷ What can be computed efficiently?

F. more computability theory
  - ▷ Other models of computability