

# Advanced Topics in Theoretical Computer Science

## Part 5: Complexity (Part 3)

2.02.2022

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: [sofronie@uni-koblenz.de](mailto:sofronie@uni-koblenz.de)

# Contents

---

- Recall: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- **Complexity**

# Until now

---

- P, NP, PSPACE; Relationships between these classes

**Open problem:** Is  $P = NP$ ? (Millenium Problem)

Closure of complexity classes:

P, PSPACE closed under complement

Open problem: NP closed under complement? *mapsto* co-NP

- Complexity classes for functions
- Polynomial time reducibility
- NP-complete and NP-hard languages; SAT is NP-complete  
Examples of NP-complete problems

# Until now

---

- PSPACE-complete and PSPACE-hard languages  
Quantified Boolean Formulae

**Theorem** QBF is PSPACE complete

Proof (Idea only)

(1) QBF is in PSPACE: we can try all possible assignments of truth values one at a time and reusing the space ( $2^n$  time but polynomial space).

(2) QBF is PSPACE complete. We can show that every language  $L'$  in PSPACE can be polynomially reduced to QBF using an idea similar to that used in Cook's theorem (we simulate a polynomial space bounded computation and not a polynomial time bounded computation).

# The structure of PSPACE

---

# NP vs. Co-NP

---

co-NP is the class of all languages for which the complement is in NP

**Example:**

$L_{\text{tautologies}} = \{w \mid w \text{ is a tautology in propositional logic}\}$  is in co-NP.

# NP vs. Co-NP

---

co-NP is the class of all languages for which the complement is in NP

## Example:

$L_{\text{tautologies}} = \{w \mid w \text{ is a tautology in propositional logic}\}$  is in co-NP.

## Informally

$L \in NP$  iff there exists a language  $L' \in P$  and a  $k \geq 0$  s.t. for all  $w \in \Sigma^*$ :

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  and s.t.  $\langle w, c \rangle \in L'$   
(can use  $c$  to check in PTIME that  $w \in L$ )

# NP vs. Co-NP

---

co-NP is the class of all languages for which the complement is in NP

## Example:

$L_{\text{tautologies}} = \{w \mid w \text{ is a tautology in propositional logic}\}$  is in co-NP.

## Informally

$L \in \text{NP}$  iff there exists a language  $L' \in \text{P}$  and a  $k \geq 0$  s.t. for all  $w \in \Sigma^*$ :

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  and s.t.  $\langle w, c \rangle \in L'$   
(can use  $c$  to check in PTIME that  $w \in L$ )

$L \in \text{co-NP}$  iff the complement of  $L$  is in NP (with test language  $L'$ )

$w \in L$  iff  $\forall c$  of length polynomial in  $|w|$ ,  $\langle w, c \rangle \notin L'$   
(can use  $c$  to check in PTIME that  $w \in L$ )



# NP vs. Co-NP

---

co-NP is the class of all languages for which the complement is in NP

## Example:

$L_{\text{tautologies}} = \{w \mid w \text{ is a tautology in propositional logic}\}$  is in co-NP.

## Informally

$L \in \text{NP}$  iff there exists a **PTIME deterministic verifier**  $M$  s.t. for all  $w \in \Sigma^*$ :

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  and s.t.  $M(w, c) = 1$

$L \in \text{co-NP}$  iff the complement of  $L$  is in  $\text{NP}$  (with test language  $L'$ )

$w \in L$  iff  $\forall c$  of length polynomial in  $|w|$ ,  $M(w, c) = 1$ .

# The structure of PSPACE

---

... Beyond NP

# The structure of PSPACE

---

**Idea:** ( $M$  PTIME deterministic verifier)

**NP**

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  s.t.  $M(w, c) = 1$ .

**co-NP**

$w \in L$  iff  $\forall c$  of length polynomial in  $|w|$ , s.t.  $M(w, c) = 1$ .

$\Sigma_2^P$

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  s.t.

$\forall d$  of length polynomial in  $|w|$ ,  $M(w, c, d) = 1$

# The structure of PSPACE

---

**Idea:** ( $M$  PTIME deterministic verifier)

**NP**

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  s.t.  $M(w, c) = 1$ .

**co-NP**

$w \in L$  iff  $\forall c$  of length polynomial in  $|w|$ , s.t.  $M(w, c) = 1$ .

$\Sigma_2^P$

$w \in L$  iff  $\exists c$  (witness) of length polynomial in  $|w|$  s.t.

$\forall d$  of length polynomial in  $|w|$ ,  $M(w, c, d) = 1$

**Example:** QBF with one quantifier alternation

$\Sigma_2 SAT = \{F = \exists P_1 \dots P_n \forall Q_1 \dots Q_m \bar{F}(P_1, \dots, P_n, Q_1, \dots, Q_n) \mid F \text{ true}\}$

# The structure of PSPACE

---

## Remarks

- in fact,  $\Sigma_2 SAT$  is complete for  $\Sigma_2^P$
- more alternations lead to a whole hierarchy
- all of it is contained in PSPACE

# The structure of PSPACE

---

For  $i \geq 1$ , a language  $L$  is in  $\Sigma_i^P$  if there exists a PTIME deterministic verifier  $M$  such that:

$$\begin{aligned} w \in L \quad \text{iff} \quad & \exists u_1 \text{ of length polynomial in } |w| \\ & \forall u_2 \text{ of length polynomial in } |w| \\ & \dots \\ & Q_i u_i \text{ of length polynomial in } |w| \\ & \text{such that } M(w, u_1, \dots, u_i) = 1 \end{aligned}$$

where  $Q_i$  is  $\exists$  if  $i$  is odd and  $\forall$  otherwise.

The polynomial hierarchy is the set  $PH = \bigcup_{i \geq 1} \Sigma_i^P$

$$\Pi_i^P = \text{co-}\Sigma_i^P = \{\bar{L} \mid L \in \Sigma_i^P\}$$

# The structure of PSPACE

---

## **Formal definition** (main ideas)

Extend the notion of polynomial reducibility:

Nondeterministic Turing Machine with an oracle: NTM + oracle tape

- makes initial guess
- consult an oracle

Informally: NOTM for problem  $P$ : nondeterministic algorithm with a subroutine for  $P$ .

# The structure of PSPACE

---

Extend the notion of polynomial reducibility:

Nondeterministic Turing Machine with an oracle: NTM + oracle tape

- makes initial guess
- consult an oracle

Informally: NOTM for problem  $P$ : nondeterministic algorithm with a subroutine for  $P$ .



# The structure of PSPACE

---

## The polynomial hierarchy (Informally)

$P^Y$  : the class of languages decidable in polynomial time by a Turing machine augmented by an oracle for some complete problem in class  $Y$ .

$NP^Y$  : the class of languages decidable in polynomial time by a non-deterministic Turing machine augmented by an oracle for some complete problem in class  $Y$ .

$A^B$  : the class of languages decidable by an algorithm in class  $A$  with an oracle for some complete problem in class  $B$ .

# The structure of PSPACE

---

The polynomial hierarchy (Informally)

$A^B$  : the class of languages decidable by an algorithm in class  $A$  with an oracle for some complete problem in class  $B$ .

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

$$\Delta_{k+1}^P = P^{\Sigma_k^P}$$

$$\Sigma_{k+1}^P = NP^{\Sigma_k^P}$$

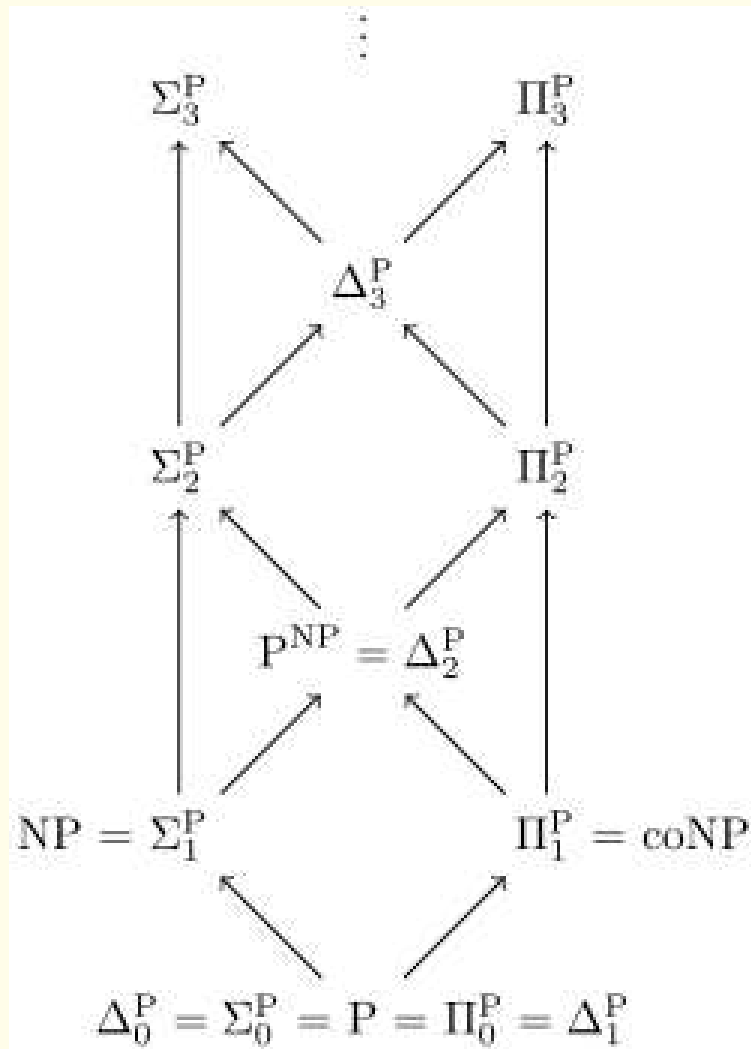
$$\Pi_{k+1}^P = \text{co-NP}^{\Sigma_k^P}$$

$$\Pi_1^P = \text{co-NP}^P = \text{co-NP}; \Sigma_1^P = NP^P = NP; \Delta_1^P = P^P = P.$$

$$\Delta_2^P = P^{NP}; \Sigma_2^P = NP^{NP}$$

# The structure of PSPACE

PSPACE



# The structure of PSPACE

---

It is an open problem whether there is an  $i$  such that  $\Sigma_i^P = \Sigma_{i+1}^P$ .

This would imply that  $\Sigma_i^P = PH$ : the hierarchy collapses to the  $i$ -th level.

Most researchers believe that the hierarchy does not collapse.

If  $NP = P$  then  $PH = P$ , i.e. the hierarchy collapses to  $P$ .

# The structure of PSPACE

---

A complete problem for  $\Sigma_k^P$  is satisfiability for quantified Boolean formulas with  $k$  alternations of quantifiers which start with an existential quantifier sequence (abbreviated  $QBF_k$  or  $QSAT_k$ ).

(The variant which starts with  $\forall$  is complete for  $\Pi_k^P$ ).

# Beyond PSPACE

---

EXPTIME, NEXPTIME

DEXPTIME, NDEXPTIME

EXSPACE, .....

# Discussion

---

- In practical applications, for having efficient algorithms polynomial solvability is very important; exponential complexity unacceptable.
- Better hardware is no solution for bad complexity

Question which have not been clarified yet:

- Does parallelism/non-determinism make problems tractable?
- Any relationship between space complexity and run time behaviour?

# Other directions in complexity

---

Parameterized complexity

Pseudopolynomial problems

Approximative and probabilistic algorithms



# Motivation

---

Many important problems are difficult (undecidable; NP-complete; PSPACE complete)

- **Undecidable:** validity of formulae in FOL; termination, correctness of programs
- **NP-complete:** SAT, Scheduling
- **PSPACE complete:** games, market analyzers

# Motivation

---

## Possible approaches:

- Identify which part of the input is cause of high complexity
- Heuristic solutions:
  - use knowledge about the structure of problems in a specific application area;
  - renounce to general solution in favor of a good “average case” in the specific area of applications.
- Approximation: approximative solution
  - Renounce to optimal solution in favor of shorter run times.
- Probabilistic approaches:
  - Find correct solution with high probability.
  - Renounce to sure correctness in favor of shorter run times.

# (I) Parameterized Complexity

---

Parameterized complexity is a branch of computational complexity theory that focuses on classifying computational problems according to their inherent difficulty with respect to **multiple parameters of the input**.

This allows the classification of NP-hard problems on a finer scale.

↳ Fixed parameter tractability.

## **Example:** SAT

Assume that the number of propositional variables is a parameter.

A given formula of size  $m$  with  $k$  variables can be checked by brute force in time  $O(2^k m)$

For a fixed number of variables, the complexity of the problem is linear in the length of the input formula.

# (I) Parameterized Complexity

---

**Fixed parameter tractability** parameter specified: Input of the form  $(w, k)$   
 $L$  is fixed-parameter tractable if the question  $(w, k) \in L?$  can be decided in running time  $f(k) \cdot p(|w|)$ , where  $f$  is an arbitrary function depending only on  $k$ , and  $p$  is a polynomial.

An example of a problem that is thought not to be fixed parameter tractable is graph coloring parameterised by the number of colors.

It is known that 3-coloring is NP-hard, and an algorithm for graph  $k$ -colouring in time  $f(k)p(n)$  for  $k = 3$  would run in polynomial time in the size of the input.

Thus, if graph coloring parameterised by the number of colors were fixed parameter tractable, then  $P = NP$ .

## (II) Approximation

---

Many NP-hard problems have optimization variants

- **Example:** Clique: Find a possible greatest clique in a graph

... but not all NP-difficult problems can be solved approximatively in polynomial time:

- **Example:** Clique: Not possible to find a good polynomial approximation (unless  $P = NP$ )

# (III) Probabilistic algorithms

---

## Idea

- Undeterministic, random computation
- Goal: false decision possible but not probable
- The probability of making a mistake reduced by repeating computations
- $2^{-100}$  below the probability of hardware errors.

# Probabilistic algorithms

---

**Example:** probabilistic algorithm for 3-Clique

NB: 3-Clique is polynomially solvable (unlike Clique)

**Given:** Graph  $G = (V, E)$

Repeat the following  $k$  times:

- Choose randomly  $v_1 \in V$  and  $\{v_2, v_3\} \in E$
- Test if  $v_1, v_2, v_3$  build a clique.

**Error probability:**

$k = (|E| \cdot |V|)/3$ : Error probability  $< 0.5$

$k = 100(|E| \cdot |V|)/3$ : Error probability  $< 2^{-100}$

# Overview

---

- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- **Complexity**
- Other computation models