Complexity Theory

Jan Křetínský

Technical University of Munich

Summer 2019

April 29, 2019

Lecture 3

Basic Complexity Classes

Agenda

- decision vs. search
- basic complexity classes
  - time and space
  - deterministic and non-deterministic
- sample problems

Decision vs. Search

- often one is interested in functions $f : \{0,1\}^* \to \{0,1\}$
- f can be identified with the language $L_f = \{x \in \{0,1\}^* \mid f(x) = 1\}$
- TM that computes f is said to decide L_f (and vice versa)

Example (Indset)

Consider the independent set problem.

Search What is the largest independent set of a graph?

Decision $\text{Indset} = \{\langle G, k\rangle \mid G \text{ has independent set of size } k\}$

Often decision plus binary search can solve search problems.

Agenda

- decision vs. search ✓
- basic complexity classes
  - time and space
  - deterministic and non-deterministic
- sample problems

Time complexity

Definition (DTIME)

Let $T : \mathbb{N} \to \mathbb{N}$ be a function. $L \subseteq \{0,1\}^*$ is in $\mathbf{DTIME}(T)$ if there exists a TM deciding L in time T' for $T' \in \mathcal{O}(T)$.

- D refers to deterministic
- constants are ignored since TM can be sped up by arbitrary constants

Space complexity

Definition (SPACE)

Let $S : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. Define $L \in \mathbf{SPACE}(S)$ iff

- there exists a TM M deciding L
- no more than $S'(n)$ locations on M's work tapes ever visited during computations on every input of length n for $S' \in \mathcal{O}(S)$

Remarks

- more detailed definition (cf. exercises): count non-$\square$ symbols, where $\square$ must not be written
- constants do not matter
- as for time complexity, require space-constructible bounds
  - S is space-constructible: there is TM M computing $S(|x|)$ in $\mathcal{O}(S(|x|))$ space on input x
  - TM knows its bounds
- work tape restrictions: allows to store input
- space bounds $< n$ make sense (as opposed to time)
- require space $\log n$ to remember positions in input

Non-deterministic TMs

Definition (NDTM)

A non-deterministic TM (NDTM) is a triple (Γ, Q, δ) like a deterministic TM except

- Q contains a distinguished state q_{accept}
- δ is a pair (δ_0, δ_1) of transition functions

- in each step, NDTM non-deterministically chooses to apply either δ_0 or δ_1
- NDTM M accepts x, $M(x) = 1$ if there exists a sequence of choices s.t. M reaches q_{accept}
- $M(x) = 0$ if every sequence of choices makes M halt without reaching q_{accept}

On non-determinism

- not supposed to model realistic devices
- remember impact of non-determinism finite state machines, pushdown automata
- NDTM compute the same functions as DTM (why?)
- non-determinism $\sim$ guessing

Non-deterministic complexity
Define **NTIME**(T) and **NSPACE**(S) such that T and S are bounds regardless of non-deterministic choices.

Basic complexity classes

deterministic | non-deterministic

time

P	=	$\bigcup_{p\geq 1}$ **DTIME**(n^p)	**NP**	=	$\bigcup_{p\geq 1}$ **NTIME**(n^p)
EXP	=	$\bigcup_{p\geq 1}$ **DTIME**(2^{n^p})	**NEXP**	=	$\bigcup_{p\geq 1}$ **NTIME**(2^{n^p})

space

L	=	**SPACE**$(\log n)$	**NL**	=	**NSPACE**$(\log n)$
PSPACE	=	$\bigcup_{p>0}$ **SPACE**(n^p)	**NPSPACE**	=	$\bigcup_{p>0}$ **NSPACE**(n^p)

Agenda

- decision vs. search ✓
- basic complexity classes ✓
    - time and space
    - deterministic and non-deterministic
- sample problems

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

L: essentially constant number of pointers into input plus logarithmically many boolean flags

- UPath $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **undirected** graph $G\}$ [Reingold 2004]
- Even $= \{x \mid x$ has an even number of 1s$\}$

Interesting examples

Most examples are the hardest within a given complexity class. They are complete for the class (wrt suitable reductions).

L: essentially constant number of pointers into input plus logarithmically many boolean flags

- UPath $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **undirected** graph $G\}$ [Reingold 2004]
- Even $= \{x \mid x$ has an even number of 1s$\}$

NL: **L** plus guessing, read-once certificates

- Path $= \{\langle G, s, t\rangle \mid \exists$a path from s to t in **directed** graph $G\}$
- 2SAT $= \{\varphi \mid$ φ satisfiable Boolean formula in CNF with two literals per clause $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\text{Circuit} - \text{Eval} = \{\langle C, x\rangle \mid C \text{ is a } n\text{−in}/1\text{−out circuit, } x \text{satisfying signals}\}$
- $\text{Primes} = \{x \mid x \text{ prime}\}$ [AKS 2004]
- many graph problems like DFS and BFS

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\mathrm{Circuit-Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in$/1-$out circuit, $x$satisfying signals$\}$
- $\mathrm{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\mathrm{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\mathrm{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\mathrm{Circuit-Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in$/1-$out circuit, $x$satisfying signals$\}$
- $\mathrm{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\mathrm{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\mathrm{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\mathrm{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

PSPACE: polynomial space, games, for instance
$\mathrm{TQBF} = \{Q_1 x_1 \dots Q_k x_k \varphi \mid k \geq 0, Q_i \in \{\forall, \exists\}, \varphi$ Boolean formula over x_i such that whole formula is true $\}$

Interesting examples

P: polynomial time, tractable, low-level choices of TM definitions are immaterial to **P**

- $\text{Circuit}-\text{Eval} = \{\langle C, x\rangle \mid C$ is a $n-$in/$1-$out circuit, $x$satisfying signals$\}$
- $\text{Primes} = \{x \mid x$ prime$\}$ [AKS 2004]
- many graph problems like DFS and BFS

NP: polynomially verifiable certificates, puzzles

- $\text{Indset} = \{\langle G, k\rangle \mid G$ has an independent set of size $k\}$
- $3-\text{Coloring} = \{G \mid G$ is 3-colorable$\}$
- $\text{3SAT} = \{\varphi \mid \varphi$ satisfiable Boolean formula in CNF with three literals per clause $\}$

PSPACE: polynomial space, games, for instance
$\text{TQBF} = \{Q_1 x_1 \dots Q_k x_k \varphi \mid k \geq 0, Q_i \in \{\forall, \exists\}, \varphi$ Boolean formula over x_i such that whole formula is true $\}$

EXP: exponential-time, for instance the language
$\text{Halt}_\text{k} = \{\langle M, x, k\rangle \mid$ DTM M stops on input x within k steps $\}$

Complements

Definition (Complement classes)

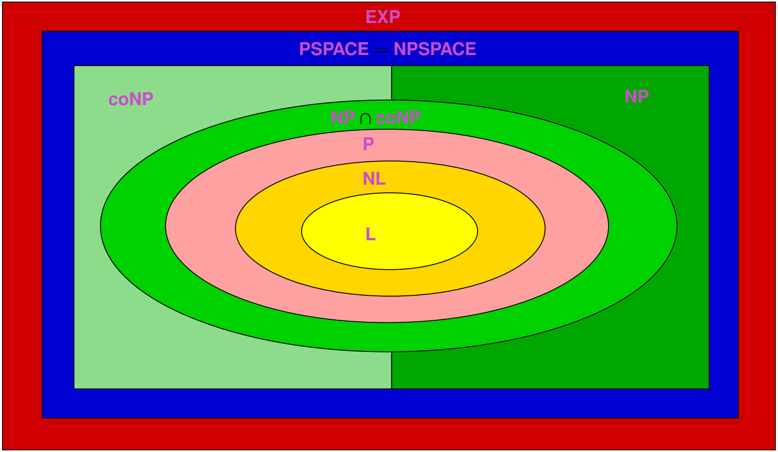
Let $\mathcal{C} \subseteq \mathcal{P}(\{0,1\}^*)$ be a complexity class. We define $co\mathcal{C} = \{\overline{L} \mid L \in \mathcal{C}\}$ to be the complement class of $\mathcal{C}$, where $\overline{L} = \{0,1\}^* \setminus L$ is the complement of L.

- important class **coNP**
- **coNP** is not the complement of **NP**
- example: Tautology $\in$ **coNP**, where a tautology is Boolean formula that is true for every assignment
- reminder: closure under complement wrt expressiveness and conciseness
  - finite state machines
  - pushdown automata
  - DTM, NDTM
- note: **P** $\subseteq$ **NP** $\cap$ **coNP**

Agenda

- universal Turing machine ✓
- decision vs. search ✓
- computability, halting problem ✓
- basic complexity classes ✓

Relation between classes



Teaser

A regular expression over $\{0, 1\}$ is defined by

$$r ::= 0 \mid 1 \mid rr \mid r|r \mid r^*$$

The language defined by r is written $\mathcal{L}(r)$.

What is the computational complexity of

- deciding whether two regular expressions are equivalent, that is $\mathcal{L}(r_1) = \mathcal{L}(r_2)$?
- deciding whether a regular expression is universal, that is $\mathcal{L}(r) = \{0, 1\}^*$?
- deciding the same for star-free regular expressions?

What have we learnt?

- TM can be represented as strings; universal TM can simulate any TM given its representations with polynomial overhead only
- uncomputable functions do exist (halting problem): diagonalization and reductions
- non-deterministic TMs
- space, time, deterministic, non-deterministic, complement complexity classes
- **L**, **NL**, **P**, **NP**, **EXP**, **PSPACE**
- 2SAT, 3SAT, Path, UPath, TQBF, Primes, Indset, 3−Coloring
- big picture
- up next: justify and explore the big picture