Computational Complexity Classes

Stathis Zachos
zachos@cs.ntua.gr
C.S. – E.C.E.
National Technical University of Athens

Abstract

Computational complexity theory deals with the classification of problems into classes of hardness called complexity classes. We define complexity classes using general structural properties, such as the model of computation (Turing machine, RAM, finite automaton, PDA, LBA, PRAM, monotone circuits), the mode of computation (deterministic, nondeterministic, probabilistic, alternating, uniform parallel), the resources (time, space, number of processors, circuit size and depth) and also randomness, oracles, interactivity, etc. Inclusions and separations between complexity classes constitute central research goals and form some of the most important open questions in theoretical computer science. Inclusions among some classes can be viewed as complexity hierarchies. We will present some of these: the arithmetical hierarchy, the Chomsky hierarchy, the polynomial-time hierarchy, a counting hierarchy, an approximability hierarchy and a search hierarchy.

SECTION 1: DECIDABILITY. During the 30's, researchers such as Gödel, Church, Kleene, Post and Turing tried to give a definition of what algorithmically solvable is, and investigated, within that model, what is computable. Their cumulative efforts gave rise to the following hierarchy:

$$\mathsf{Elementary} \subsetneq \mathsf{PrimRec} \subsetneq \mathsf{Rec} \subsetneq \mathsf{RE} \subsetneq \mathsf{Arithmetical} \subsetneq \mathsf{Analytical},$$

where a relation $R \subseteq \mathbb{N}^k$ (or set of strings (a language), or set of natural numbers, or problem) is:

- **Elementary** iff it is loop-computable with number of nested for-loops at most 2;
- **Primitive Recursive** iff it is loop-computable;
- **Recursive** iff it is TM-computable;
- **Recursively Enumerable** iff it is TM-listable;
- **Arithmetical** iff it is definable by first-order quantified formula;
- **Analytical** iff it is definable by a second-order quantified formula.

SECTION 2: FORMAL LANGUAGES AND AUTOMATA. In the next two decades, the theory of automata was developed (Chomsky, Rabin, Scott, Kleene). In 1956, Noam Chomsky described his hierarchy:

$$\mathsf{FIN} \subsetneq \mathsf{REG} \subsetneq \mathsf{CF} \subsetneq \mathsf{CS} \subsetneq \mathsf{RE},$$

where a language (i.e., a set of strings) is:

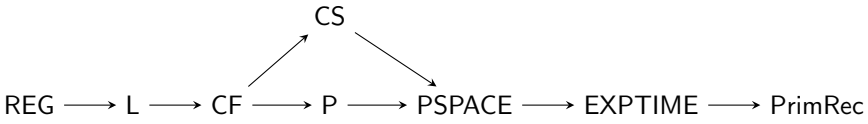
- **REGular** iff it is decidable (acceptable) by a (deterministic or nondeterministic) finite automaton, iff it is definable by a regular expression, iff it is generatable by a right-linear Grammar;
- **Context Free** iff it is decidable (acceptable) by a (nondeterministic) push-down automaton, iff it is generatable by a context-free grammar;
- **Context Sensitive** iff it is decidable (acceptable) by a (nondeterministic) linearly-bounded automaton, iff it is generatable by a context-sensitive grammar.
- **RE**: Listable (or acceptable) by a Deterministic or Nondeterministic TM, equivalently generetable by a General Grammar.

SECTION 3: COMPUTATIONAL COMPLEXITY. In the 60's, computational complexity makes its appearance (Hartmanis etc.). Questions like "Can we solve a problem?" get now a quantitative counterpart: "How hard is it to solve a problem?". Define TIME($f(n)$) (SPACE($f(n)$)) as the set of problems which can be decided by a Turing machine in time (space respectively) $O(f(n))$. Define L, P, PSPACE, $\mathsf{EXPTIME}$ as the set of problems which can be decided by a deterministic Turing machine in space $O(\log n)$, time $O(n^c)$, space $O(n^c)$, time 2^{n^c}, for some c respectively. The tight hierarchy theorems show that with more resources, one can decide more problems:

Theorem 1 (Hartmanis, Lewis, Stearns, 1965)**.** *For a space-constructible $s(n)$*

$$SPACE(o(s(n))) \subsetneq SPACE(s(n))$$

Theorem 2 (Fürer, 1982)**.** *For a time-constructible $t(n)$, $TIME(o(t(n))) \subsetneq TIME(t(n))$.*

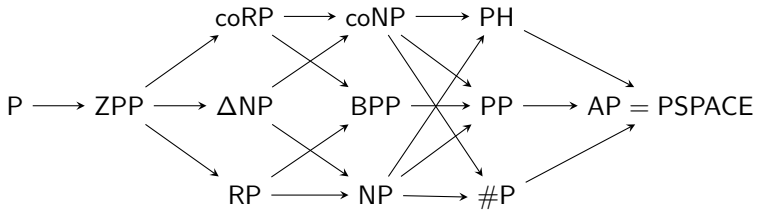


SECTION 4: NONDETERMINISM. A burst of research activity in Computational Complexity follows in the 70's (Cook, Karp, Savitch). Non-determinism re-enters the picture, the theory of NP-completeness is developed.

Define NP, NL, $\mathsf{NPSPACE}$ as the set of problems which can be decided by a non-deterministic Turing machine in time $O(n^c)$, in space $O(\log n)$, in space $O(n^c)$ for some c.

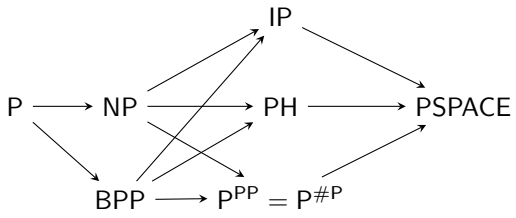
$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} = \mathsf{NPSPACE}.$$

SECTION 5: PROBABILITY, ORACLES, ALTERNATION. Concepts such as oracle computation, the polynomial hierarchy, alternation are brought into the complexity setting, and while complexity theory is in line with its parent discipline, computability theory, new ideas are emerging such as probabilistic computation and counting complexity classes (Stockmeyer, Valiant, Gill).



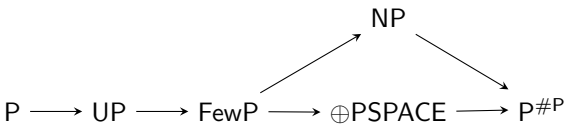
- $L \in \mathsf{BPP} \overset{\text{def}}{\iff} \exists R \in \mathsf{P} : \begin{cases} x \in L \implies \exists^{+} y\, R(x,y), \\ x \notin L \implies \exists^{+} y\, \neg R(x,y). \end{cases}$
- $L \in \mathsf{RP} \overset{\text{def}}{\iff} \exists R \in \mathsf{P} : \begin{cases} x \in L \implies \exists^{+} y\, R(x,y), \\ x \notin L \implies \forall y\, \neg R(x,y). \end{cases}$
- $\mathsf{ZPP} \overset{\text{def}}{=} \mathsf{RP} \cap \mathsf{coRP}$, $\Delta\mathsf{NP} \overset{\text{def}}{=} \mathsf{NP} \cap \mathsf{coNP}$, in general $\Delta\mathsf{C} \overset{\text{def}}{=} \mathsf{C} \cap \mathsf{coC}$.
- $L \in \mathsf{PP} \overset{\text{def}}{\iff} \exists R \in \mathsf{P} : \begin{cases} x \in L \implies \exists_{1/2} y\, R(x,y), \\ x \notin L \implies \exists_{1/2} y\, \neg R(x,y). \end{cases}$
- PH: Polynomial Hierarchy (the polynomial analogue of the arithmetical hierarchy).
- $\#\mathsf{P}$ is the class of functions f for which there is a polynomial time NDTM, whose computation tree has exactly $f(x)$ accepting computation paths (for input x).
- AP is the class of problems which can be decided by an Alternating Polynomial time Turing machine in polynomial time.

SECTIONS 6–10. The proliferation of research activity continues in the 80's. Motivated by neophyte fields such as **cryptography** and **parallel computation**, the concepts of **interactive proofs** and one-way functions are introduced, and researchers start to investigate classes below P (including classes of problems decided by **families of circuits**). At the same time, remaining faithful to the parent disciplines of theory of computation and mathematical logic, researchers develop the theory of **descriptive complexity**, which had started in 1974 with Fagin's theorem, and the theory of efficient provability. New complexity classes are introduced. The zoo of complexity classes is becoming more and more refined:



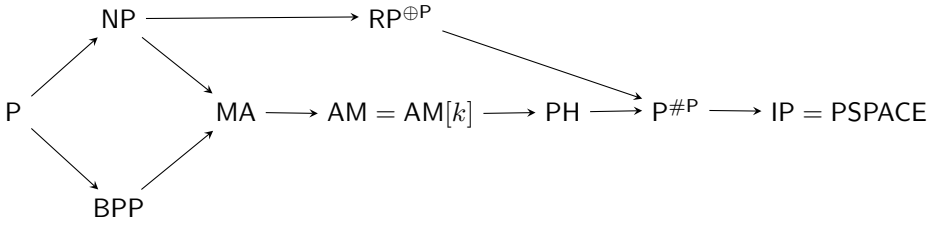
$L \in \mathsf{IP} \overset{\text{def}}{\iff}$ there is an **interactive proof** and a verifier V:

- $x \in L \implies \exists$prover P such that V accepts with overwhelming probability ($\exists^+$).
- $x \notin L \implies \forall$provers P, V does not accept with overwhelming probability ($\exists^+$).



The model is a nondeterministic polynomial time TM. The computation tree on input x is a full complete binary tree of height $p(|x|)$.

- $\oplus\mathsf{P}$: if answer is "yes" then the number of accepting paths is odd, if answer is "no" then the number of accepting paths is even.
- FewP: if answer is "yes" then the number accepting paths is bounded by a polynomial with respect to the size of input.
- UP: if answer is "yes" then exactly one accepting path.

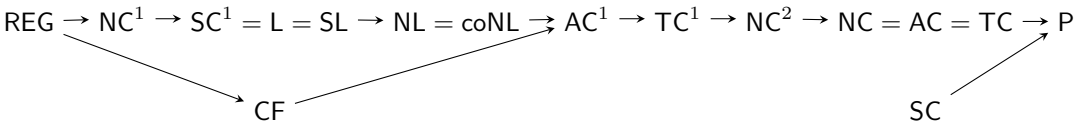


$L \in \mathsf{AM}[k] \overset{\text{def}}{\iff}$ there is a k-move game where Arthur plays first, Merlin follows, and:

- $x \in L \implies \exists$Merlin convincing Arthur with overwhelming probability that $x \in L$.
- $x \notin L \implies \forall$Merlin, Arthur is convinced with negligible probability that $x \in L$.

Theorem 3 (Immerman, Szelepcsényi, 1987)**.** *NSPACE*$(s(n)) =$ *co*$NSPACE(s(n))$.

CS = *co*CS follows from Theorem 3.



- NC^k is the class of languages acceptable by DLOGTIME-uniform circuit families of polynomial size and $O(\log^k n)$ depth, using bounded fan-in gates.
- AC^k is the class of languages acceptable by DLOGTIME-uniform circuit families of polynomial size and $O(\log^k n)$ depth, using unbounded fan-in gates.
- TC^k is the class of languages acceptable by DLOGTIME-uniform circuit families of polynomial size and $O(\log^k n)$ depth, using threshold gates.
- SC^k is the class of languages acceptable by a DTM in polynomial time and in $O(\log^k n)$ space.
- NC, AC, TC, SC is the union of all NC^k, AC^k, TC^k, SC^k respectively.
- SL is the class of all problems decidable by a symmetric logspace TM.

SECTION 11: PCP. In the 90's, the theory of probabilistically checkable proofs is developed, as well as the closely related theory of approximability.
$L \in \mathsf{PCP} \stackrel{\text{def}}{\iff}$ there is a verifier V:

- $x \in L \implies \exists$ proof Π such that V always accepts
- $x \in L \implies \forall$ proof Π, V does not accept with overwhelming probability.

$\mathsf{PCP}(r(n), q(n))$ consists of languages $L \in \mathsf{PCP}$ such that the polynomial time verifier V uses $O(r(n))$ random bits and queries $O(q(n))$ bits of the proof.

Theorem 4 (Arora, Lund, Motwani, Sudan, Szegedy)**.** $\mathsf{NP} = \mathsf{PCP}(\log n, 1)$.

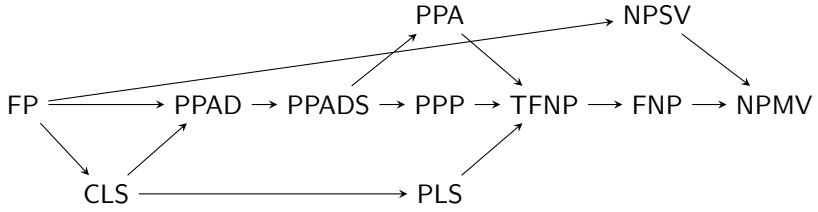
Approximation Classes

$$\mathsf{PO} \subseteq \mathsf{FPTAS} \subseteq \mathsf{PTAS} \subseteq \mathsf{APTAS} \subseteq \mathsf{APX} \subseteq \log \mathsf{APX} \subseteq \mathit{poly}\ \mathsf{APX} \subseteq \mathsf{NPO}.$$

- NPO is the class of optimization problems for which the underlying decision problem is in NP (with the condition that there are feasible solutions for every instance).
- PO is the class of optimization problems for which the underlying decision problems is in P.
- APX is the class of problems for which there exists a ρ-approximative algorithm for some constant $\rho > 0$.
- log APX is the class of problems for which there exists a $\log n$-approximative algorithm.
- *poly* APX is the class of problems for which there exists a $p(n)$-approximative algorithm for some polynomial p.
- PTAS is the class of problems for which there exists a polynomial time approximation scheme, i.e., a $(1+\epsilon)$-approximative algorithm for any constant $\epsilon > 0$.
- FPTAS: problems for which there exists a fully polynomial time approximation scheme, i.e., a $(1+\epsilon)$-approximative algorithm for any constant ϵ, where, the time needed is also polynomial with respect to $1/\epsilon$
- APTAS is the class of problems for which there exists an asymptotic polynomial time approximation scheme, i.e., a $(1 + \epsilon + c/OPT)$-approximative algorithm for any constant $\epsilon > 0$, for some constant c.

SECTIONS 12–15. Since 2000, new concepts are constantly entering computational complexity: approximation algorithms, parameterized complexity, exact exponential complexity etc.

Search Hierarchy



- FP is the function problem version of P.
- FNP is the class of partial multi-valued functions computed by an NPTM. The computation tree for input x has leaves which answer either ? or the signature of the path y satisfying $R(x, y)$.
- NPMV is the class of partial multi-valued functions computed by an NPTM. The computation tree for input x has leaves which answer either ? or one of the possible output strings.
- NPSV is the class of single-valued NPMV functions.
- TFNP is the class of all FNP functions for which $\forall x \exists y\ R(x, y)$.
- PLS: Polynomial Local Search, based on: every finite directed acyclic graph has a sink.
- CLS: Continuous Local Search, PLS analogue for continuous spaces and functions. CLS contains search problems of local optimum approximation of a continuous function, using an oracle for a continuous function f.
- PPP: Polynomial Pigeonhole Principle, based on pigeonhole principle.
- PPA: Polynomial Parity Argument, based on: all graphs of max degree 2 have an even number of leaves.
- PPAD: Polynomial Parity Argument Directed. Like PPA, but graph is directed: find a sink or a source.
- PPADS: Polynomial Parity Argument Directed Sink. Like PPAD, but find a sink.

Counting Complexity

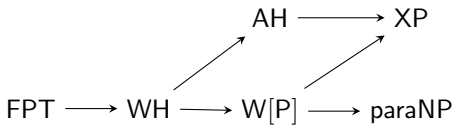
$$\mathsf{FP} \subseteq \mathsf{TotP} \subseteq \#\mathsf{PE} \subseteq \#\mathsf{P} \subseteq \#\mathsf{NP} \subseteq \#\mathsf{PH}.$$

- $\#\mathsf{NP} \stackrel{\text{def}}{=} (\#\mathsf{P})^{\mathsf{NP}}$
- $\#\mathsf{PH} \stackrel{\text{def}}{=} (\#\mathsf{P})^{\mathsf{PH}}$
- $\#\mathsf{PE}$ is the subclass of $\#\mathsf{P}$ that contains functions with easy decision version.
- TotP consists of all functions for which there exists a NPTM M such that the function value on x is equal to the total number of computation paths of M on input x.
- SpanL is the # distinct outputs of a logspace NTM transducer.

Theorem 5. $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{TotP}[1]}$

In the last decade new disciplines are gaining importance:

SECTION 16: Parameterized Complexity (Downey, Fellows) defines for a language, along with a parameter $k : \Sigma^* \to \mathbb{N}$, the notion of **Fixed Parameter Tractability**. A language (L, k) is Fixed Parameter Tractable (FPT) if it can be decided in $O(f(k(x)) \cdot p(|x|))$ time, for some computable function f. The advantage of this approach is that we can concentrate the hardness of a problem to a certain parameter. As in the theory of NP-completeness, there is the class $\mathsf{W[P]}$ of parameterized intractable problems, and the question FPT vs $\mathsf{W[P]}$ is the parameterized analogue of P vs NP.



SECTION 17: Nonuniform Circuit Complexity (Razborov, Williams) uses as a model of computation Non-Uniform families of circuits, i.e. sequences of circuits without an algorithm for constructing each member of the sequence. The most notable class is $\mathsf{P/poly}$, the class of languages decided by a circuit family, such that each circuit has polynomial size. $\mathsf{P/poly}$ properly contains P and BPP, and because the computational model is not uniform, $\mathsf{P/poly}$ contains also undecidable problems.

Theorem 6 (Karp-Lipton)**.** *If* $\mathsf{NP} \subseteq \mathsf{P/poly}$*, then* $\mathsf{PH} = \Sigma_2^p$.

Theorem 7 (Razborov-Andreev-Alon-Boppana)**.** *There exists an* $\varepsilon > 0$*, s.t.* $\forall k \leq n^{1/4}$*, the k-clique problem cannot be decided by monotone circuits of size less than* $2^{\varepsilon\sqrt{k}}$.

SECTION 18: Quantum Complexity Theory (Feynman, Shor, U. Vazirani) adopts the quantum models of computation, such as Quantum TMs and Quantum Circuits. The gates of such circuits are unitary transformations of their input. Quantum Complexity Classes like BQP (the quantum analogue of BPP), QMA (corresponding to MA), PQP (corresponding to PP) and QIP (corresponding to IP) appeared, and some relations with Classical Complexity classes were proved.