Complexity Classes and Complexity in LaTeX Cheat Sheet

Antonis Antonopoulos

Various Symbols

$\mathcal{O}(\cdot)$	`\bO{}`	The big O asymptotic Notation
$\mathcal{C}$	`\C`	An arbitrary Complexity Class
$co\mathcal{C}$	`\cC`	The complement of an arbitrary Complexity Class: $co\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\}$
$\mathcal{F}$	`\F`	An arbitrary Function Complexity Class
$co\mathcal{F}$	`\cF`	The Complement of $\mathcal{F}$
$\{0,1\}$	`\zo`	
$\{0,1\}^n$	`\zon`	The set of strings of length n produced using $\{0,1\}$ as alphabet
$\{0,1\}^*$	`\zos`	The set of strings produced using $\{0,1\}$ as alphabet
Σ^*	`\Ss`	The set of strings produced using Σ as alphabet
$\{C_n\}_{n\in\mathbb{N}}$	`\Cirfam`	The arbitrary family of circuits

62 Essential Complexity Classes

Deterministic & Nondeterministic Classes

$\mathbf{DTIME}[t(n)]$	`\DTIME{t(n)}`	Languages accepted by DTMs in $\mathcal{O}(t(n))$ time
$\mathbf{NTIME}[t(n)]$	`\NTIME{t(n)}`	Languages accepted by NTMs in $\mathcal{O}(t(n))$ time
$\mathbf{DSPACE}[s(n)]$	`\DSPACE{t(n)}`	Languages accepted by DTMs using $\mathcal{O}(s(n))$ space
$\mathbf{NSPACE}[s(n)]$	`\NSPACE{t(n)}`	Languages accepted by NTMs using $\mathcal{O}(s(n))$ space
$\mathbf{P}$	`\cP`	Languages accepted by DTMs in poly time: $\mathbf{P} = \cup_{n\in\mathbb{N}}\mathbf{DTIME}[n^c]$
$\mathbf{NP}$	`\NP`	Languages accepted by NTMs in poly time: $\mathbf{NP} = \cup_{n\in\mathbb{N}}\mathbf{NTIME}[n^c]$
$co\mathbf{NP}$	`\cNP`	The complement of $\mathbf{NP}$
$\mathbf{EXP}$	`\EXP`	Languages accepted by DTMs in exponential time: $\mathbf{EXP} = \mathbf{DTIME}[2^{poly(n)}]$
$\mathbf{E}$	`\E`	Languages accepted by DTMs in exponential time with linear exponent: $\mathbf{E} = \mathbf{DTIME}[2^{\mathcal{O}(n)}]$
$\mathbf{NEXP}$	`\NEXP`	Languages accepted by NTMs in exponential time: $\mathbf{NEXP} = \mathbf{NTIME}[2^{poly(n)}]$
$\mathbf{L}$	`\cL`	Languages accepted by DTMs using log space: $\mathbf{L} = \mathbf{DSPACE}[\log n]$
$\mathbf{NL}$	`\NL`	Languages accepted by NTMs using log space: $\mathbf{L} = \mathbf{NSPACE}[\log n]$
$co\mathbf{NL}$	`\cNL`	The complement of $\mathbf{NL}$
$\mathbf{SL}$	`\SL`	Languages accepted by *symmetric* TMs using log space
$\mathbf{PSPACE}$	`\PSPACE`	Languages accepted by DTMs using poly space: $\mathbf{PSPACE} = \cup_{n\in\mathbb{N}}\mathbf{DSPACE}[n^c]$
$\mathbf{NPSPACE}$	`\NPSPACE`	Languages accepted by NTMs using poly space: $\mathbf{NPSPACE} = \cup_{n\in\mathbb{N}}\mathbf{NSPACE}[n^c]$
Σ_2^p	`\Stwo`	The second level of the Polynomial Hierarchy: $\Sigma_2^p = \mathbf{NP}^{\mathbf{NP}}$
Π_2^p	`\Ptwo`	The complement of the second level of the Polynomial Hierarchy: $\Pi_2^p = co\mathbf{NP}^{\mathbf{NP}}$
Θ_2^p	`\Ttwo`	Another notation for $\mathbf{P}^{\mathbf{NP}[\log n]}$.
$\mathbf{P}^{\mathbf{NP}[\log n]}$	`\Ttwod`	Languages accepted in determnistic polynomial-time using an $\mathbf{NP}$-oracle with at most $\mathcal{O}(\log n)$ queries allowed.
Σ_k^p	`\Sp{k}`	The k-th level of the Polynomial Hierarchy: $\Sigma_k^p = \mathbf{NP}^{\Sigma_{k-1}^p}$
Π_k^p	`\Pp{k}`	The complement of the k-th level of the Polynomial Hierarchy: $\Pi_k^p = co\mathbf{NP}^{\Sigma_{k-1}^p}$
Δ_k^p	`\Dp{k}`	$\mathbf{P}$ with an Σ_{k-1}^p oracle
$\mathbf{PH}$	`\PH`	The Polynomial Hierarchy: $\mathbf{PH} = \cup_{k\in\mathbb{N}}\Sigma_k^p$
$\mathbf{FP}$	`\FP`	The function analogue of $\mathbf{P}$
$\mathbf{FNP}$	`\FNP`	The function analogue of $\mathbf{NP}$
$\mathbf{TFNP}$	`\TFNP`	The total function analogue of $\mathbf{NP}$
$\mathbf{FP}^{\mathbf{NP}[\log n]}$	`\FTtwod`	The function analogue of $\mathbf{P}^{\mathbf{NP}[\log n]}$

Randomized Classes

$\mathbf{BPTIME}[t(n)]$	`\BPTIME{t(n)}`	Languages accepted by PTMs with two-sided bounded error in $\mathcal{O}(t(n))$ time
$\mathbf{RTIME}[t(n)]$	`\RTIME{t(n)}`	Languages accepted by PTMs with one-sided error in $\mathcal{O}(t(n))$ time
$\mathbf{BPP}$	`\BPP`	Languages accepted by PTMs with two-sided bounded error in poly time: $\mathbf{BPP} = \cup_{n\in\mathbb{N}}\mathbf{BPTIME}[n^c]$
$\mathbf{RP}$	`\RP`	Languages accepted by PTMs with one-sided error in poly time: $\mathbf{RP} = \cup_{n\in\mathbb{N}}\mathbf{RTIME}[n^c]$
$co\mathbf{RP}$	`\cRP`	The complement of $\mathbf{RP}$
$\mathbf{RL}$	`\RL`	Languages accepted by PTMs with one-sided error using log space
$\mathbf{ZPP}$	`\ZPP`	Languages accepted by PTMs without error in the answer, in *expected* polynomial time
$\mathbf{PP}$	`\PP`	Languages accepted by PTMS with (unbounded) two-sided error in polynomial time

Non-Uniform Classes

$\mathbf{SIZE}[f(n)]$	`\SIZE{f(n)}`	Languages accepted by non-uniform circuit families of size $f(n)$
$\mathbf{P}_{/\mathbf{poly}}$	`\Ppoly`	Languages accepted by non-uniform circuit families of poly size
$\mathbf{NC}^i$	`\NC{i}`	
$\mathbf{NC}$	`\NC{}`	
$\mathbf{AC}^i$	`\AC{i}`	
$\mathbf{AC}$	`\AC{}`	
$\mathbf{TC}^i$	`\TC{i}`	
$\mathbf{SC}^i$	`\SC{i}`	
$\mathbf{AC}^0[m]$	`\ACz{m}`	
$\mathbf{ACC}^0$	`\ACCz`	
$\mathbf{RNC}$	`\RNC`	

Interactive Proof Classes

$\mathbf{IP}$	`\IP`	The class of languages having an Interactive Proof System, with a probabilistic verifier.
$\mathbf{AM}$	`\AM`	The class of Arthur-Merlin games (Interactive Proof Systems using public coins)
$co\mathbf{AM}$	`\cAM`	The complement of $\mathbf{AM}$
$\mathbf{MA}$	`\MA`	The class of Merlin-Arthur games (similar to $\mathbf{AM}$, but now Merlin plays first)
$co\mathbf{MA}$	`\cMA`	The complement of $\mathbf{MA}$
$\mathbf{PCP}[r(n), q(n)]$	`\PCP{r(n)}{q(n)}`	The class of languages with *probabilistic checkable proofs* using $\mathcal{O}(r(n))$ random bits, and querying $\mathcal{O}(q(n))$ bits of the proof

Counting Classes

#P	`\sP`
⊕P	`\oddP`
UP	`\UP`
FewP	`\FewP`
$\mathbf{Mod}_k\mathbf{P}$	`\modP`

Subclasses of TFNP

PLS	`\PLS`	The class of function problems that are guaranteed to have a solution because of the lemma that "*every finite directed acyclic graph has a sink.*"
PPP	`\PPP`	The class of function problems that are guaranteed to have a solution because of the Pigeonhole Principle.
PPA	`\PPA`	The class of function problems that are guaranteed to have a solution because of the lemma that "*all graphs of maximum degree 2 have an even number of leaves.*"
PPAD	`\PPAD`	Same as **PPA**, except now the graph is directed, and we're asked to find either a source or a sink.

Relations & Reductions

$\subseteq$	`\cs`	The inclusion relation
$\subsetneq$	`\cps`	The *proper* inclusion relation
$\not\subseteq$	`\cns`	The non-inclusion relation
$\leq_m^p$	`\kred`	The Karp Reduction
$\leq_T^p$	`\tred`	The Cook Reduction
$\leq_{tt}^p$	`\ttred`	The Truth-Table Reduction
$\leq_l$	`\lred`	The Log-Space Reduction

Common Problems

`SAT`	`\SAT`	The Satisfiability Problem
`#SAT`	`\sSAT`	The counting version of `SAT`
`PERMANENT`	`\PREM`	The Permanent Problem
`TSP`	`\TSP`	The Travelling Salesman Problem
`GI`	`\GI`	The Graph Isomorphism Problem
`GNI`	`\GNI`	The Graph Non-Isomorphism Problem
`FACTORING`	`\FACT`	The Problem of Factoring
`HALTING PROBLEM`	`\HP`	The Halting Problem
`TQBF`	`\TQBF`	The True Quantified Boolean Formula Problem

Quantifier Characterizations

$\exists^+$	`\explus`	The overwhelming majority quantifier
(Q/Q')	`\qcc{Q}{Q'}`	The quantifier notation of complexity classes, where Q and Q' are (sequences of) quantifiers

Operators on Complexity Classes

$\mathcal{N}\cdot\mathcal{C}$	`\opN{\C}`	The nondeterministic operator
$\Delta\cdot\mathcal{C}$	`\opD{\C}`	The difference operator ($\Delta\cdot\mathcal{C} = \mathcal{C} \cap co\mathcal{C}$)
$\mathcal{BP}\cdot\mathcal{C}$	`\BP{\C}`	The (two-sided) bounded probabilistic operator
$\mathcal{P}\cdot\mathcal{C}$	`\opP{\C}`	The (two-sided) probabilistic operator
$\mathcal{R}\cdot\mathcal{C}$	`\opR{\C}`	The (one-sided) probabilistic operator
$\text{Almost}\mathcal{C}$	`\Alm{\C}`	The Almost operator: $\text{Almost}\mathcal{C} = \{L \mid \Pr_{A\subseteq\{0,1\}^*}\left[L \in \mathcal{C}^A\right] = 1\}$

Other (more rare) Classes

EE	`\EE`	
SUBEXP	`\SUBEXP`	
QuasiP	`\QuasiP`	
$\mathbf{EXP}_{/\mathbf{poly}}$	`\EXPpoly`	
$\mathtt{pseudo}_{\mathcal{C}}\mathcal{C}'$	`\pseudo{\C}{\C'}`	
S_2^p	`\Sytwo`	The second level of the Symmetric Hierarchy
S_k^p	`\Syp{k}`	The k-th level of the Symmetric Hierarchy

*To use the Complexity commands mentioned above, you have to include the CC.tex command definitions (http://www.corelab.ntua.gr/~aanton/CC.tex).
*Many of the definitions are taken from Scott Aaronson's Complexity Zoo!