$\mathcal{P}$, $\mathcal{NP}$ and Completeness

We write $\mathcal{N}$ to mean the *natural numbers*, the numbers you learned in pre-school: the numbers people used before anyone invented negative numbers, rational numbers, real numbers, complex numbers, and even zero. In other words, the positive integers.

A *polynomial function* is a function $f : \mathcal{N} \to \mathcal{N}$ such that, for some positive integers k and N, $f(n) \leq n^k$ for all $n \geq N$.

We say that a language $L \subseteq \Sigma^*$ is a member of the class $\mathcal{P}$-TIME if there is a machine M which decides L in polynomial time. More specifically: there is a deterministic machine M which takes any string $w \in \Sigma^*$ as input, and which outputs 1 if $w \in L$ and 0 if $w \notin L$, and which runs in at most $f(n)$ steps, where f is a polynomial function and $n = |w|$, the length of w.

A language L is in the class $\mathcal{NP}$ if there is a non-deterministic machine which accepts L in polynomial time. A language L is in the class $\mathcal{NP}$–complete every $\mathcal{NP}$ language has a polynomial time reduction to L. Boolean satisfiability (SAT) is $\mathcal{NP}$–complete.

If $\mathcal{C}$ is any class of languages, co-$\mathcal{C}$ is the class of all languages whose complements are in $\mathcal{C}$. The factoring problem for binary integers is $\mathcal{NP}$ and also co-$\mathcal{NP}$, but is not known to be $\mathcal{P}$*–time*.

A language L is in the class $\mathcal{P}$–SPACE if there is a determinitic machine which decides L using only polynomial space. Every $\mathcal{P}$–TIME language is $\mathcal{P}$–SPACE. A language L is in the class $\mathcal{P}$–SPACE–complete every $\mathcal{P}$–SPACE language has a polynomial time reduction to L. The jigsaw puzzle problem, *i.e.* can a given set of shapes be placed inside a given area without overlap, is $\mathcal{P}$–SPACE–complete.

A language L is *decidable* (*recursive*) if there is a machine which decides L. A language is *acceptable* if there is a machine which accepts L. A language is acceptable if and only if it is $\mathcal{RE}$ (recursively enumerable). A language L is decidable if and only if it is both $\mathcal{RE}$ and co-$\mathcal{RE}$. The halting problem is $\mathcal{RE}$ but not co-$\mathcal{RE}$, hence undecidable, while the context-free grammar equivalence problem is co-$\mathcal{RE}$ but not $\mathcal{RE}$, hence undedidable.

A language is *context-sensitive* if it is generated by a context-sensitive grammar. Every context-sensitive language is decidable. A language is $\mathcal{RE}$ if and only if it is generated by some general grammar, also called unrestricted grammar.

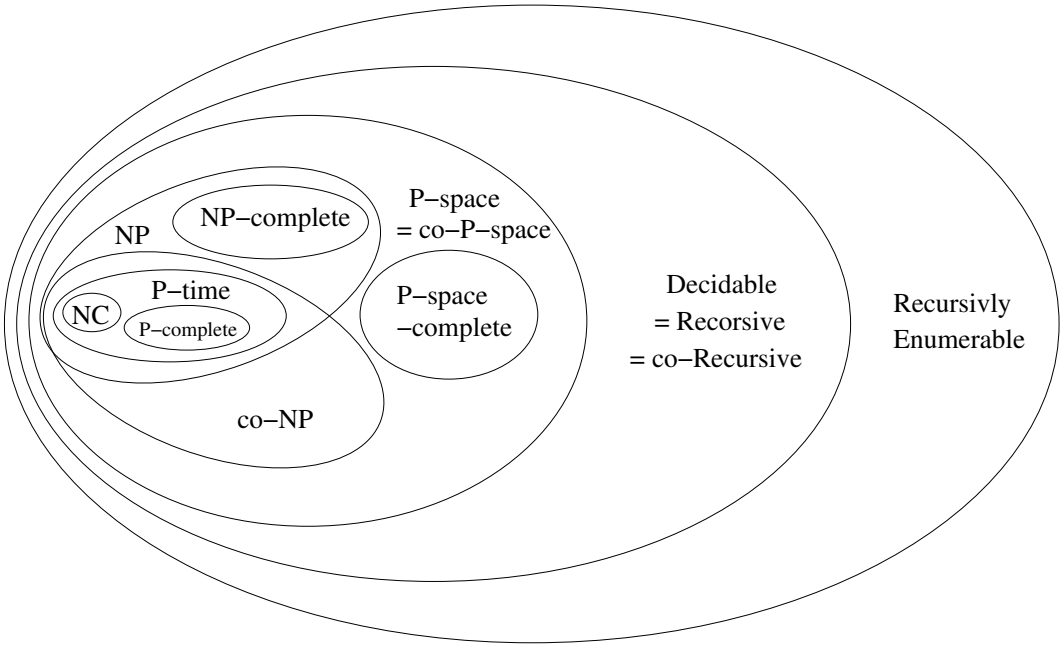
A language L is Nick's Class ($\mathcal{NC}$) if there is a parallel algorithm which decides L in polylogarithmic time using polynomially many processors. Every context-free language is $\mathcal{NC}$. Every regular language is contex-free. A function F is said to be $\mathcal{NC}$ if F can be computed in polylogarithmic time using polynomially many processors. Every $\mathcal{NC}$ language is $\mathcal{P}$–TIME. A language is $\mathcal{P}$–COMPLETE if every $\mathcal{NC}$ language has a $\mathcal{NC}$ reduction to L. The Boolean Circuit Problem is $\mathcal{P}$-complete.

Here are some important open questions.

1. Is $\mathcal{NC} = \mathcal{P}$–TIME?
2. Is $\mathcal{P}$–TIME $= \mathcal{NP}$?
3. Is $\mathcal{P}$–TIME $= \mathcal{P}$–SPACE?

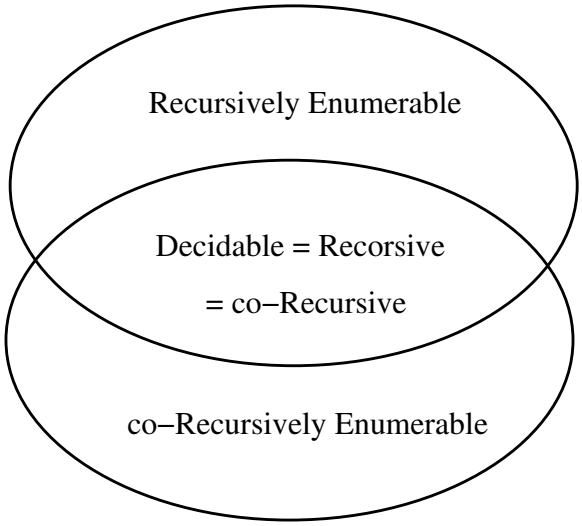
We assume that all these are false, although no one has proved any of them.

We could draw an Euler diagram which shows all the language classes mentioned above, but it would be too cluttered. Instead, we show several simpler diagreams. In the diagram below, we draw the ovals separately, assuming that no two of the classes shown are equal.



Recursive and Recursively Enumerable

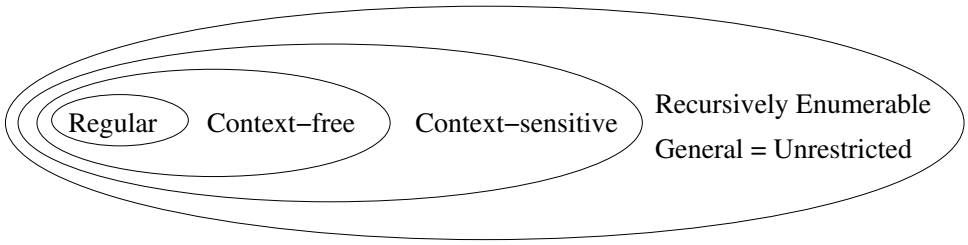
Here is the Euler diagram for the classes $\mathcal{RE}$, co-$\mathcal{RE}$, and Recursive These classes are known to be distinct.



Grammar Classes

The Chomsky Hierarchy contains four grammar classes, as shown below. How do the grammar classes relate to the complexity classes? The only complexity class equal to a grammar class is $\mathcal{RE}$, which is exactly the class of languages generated by general grammars, also known as unrestricted grammars, also known as phase-structure grammars.

All context-free languages are in Nick's class. All context-sensitive languages are $\mathcal{P}$–SPACE, and a language is in $\mathcal{RE}$ if and only if it is generated by a general grammar.



Grammar Classes