Computational Classes

The Euler diagram below shows the widely believed relationship between the indicated complexity classes. Although “everyone” believes that $P$-time is not equal to $NP$-time, and that $NC$ (Nick’s Class) is not equal to $P$-time, for example Some inequalities are known: for example, it is known that $P$-time is not equal to $EXP$-time, and that $P$-space is not equal to $EXP$-space.

Polynomially and Exponentially Bounded Functions

An increasing function $f$ on integers is *polynomially bounded* if there is some $k > 0$ such that $f(n) = O(n^k)$. Write $P$ for the class of polynomial functions. We say that a function $f$ is *exponentially bounded* if $f(n) = O(2^{g(n)})$ for some $g \in P$. Let $EXP$ be the class of exponentially bounded functions. Clearly, $P \subseteq EXP$.

We say that a function $f$ is *polylogarithmically bounded* if there is some constant $k$ such that $f(n) = O(\log^k n)$.

Is there Anything in Between Polynomial and Exponential?

Does there exist a function which grows faster than any polynomial function, but slower than any strictly exponential function? The answer is yes; for example, if $F$ is any solution to the recurrence $F(n) = F(n-1) + F(n/2) + 1$, then $F \notin P$, but $F$ grows more slowly than $2^k$ for any positive constant $k$.

$P$-Time and EXP-Time

$P$-time and EXP-time (typically written as simply $P$ and EXP) are the classes of languages (problems) that can be decided (solved) by a deterministic machine in time which is polynomially...
bounded, or exponentially bounded.

\( \mathcal{NP} \)-Time, NEXP-Time, \( \mathcal{P} \)-Space, and EXP-Space

These are the classes of languages which can be accepted in polynomial time, or exponential time, by a non-deterministic machine, and the classes of languages which can be decided by a deterministic machine which uses polynomially bounded space or exponentially bounded space.

Nick’s Class

Nick’s class (\( \mathcal{NC} \)) is defined to be the class of languages which can be decided (or problems that can be solved) in polylogarithmic time using polynomially many processors. Many processors running in parallel can be emulated by a single processor emulating the many processors in round-robin fashion. Thus \( \mathcal{NC} \subseteq \mathcal{P} \)-time.

“–Complete” Subclasses

Recall that an \( \mathcal{NP} \)-complete problem is the hardest problem in the class \( \mathcal{NP} \)-time in some sense. In what sense? From a practical perspective, the partition problem appears far easier than SAT, for example. The perspective is that polynomial time computation is considered “trivial,” and if you view all \( \mathcal{P} \) computations to have zero hardness, what’s left for any two \( \mathcal{NP} \) problems is equally hard. The same concept guides the definitions of all “–complete” subclasses in our diagram.

A language \( L \) in \( \mathcal{NP} \) is \( \mathcal{NP} \)-complete if every language in \( \mathcal{NP} \) reduces to \( L \) in polynomial time.

A language \( L \) in \( \mathcal{P} \)-space is \( \mathcal{P} \)-space-complete if every language in \( \mathcal{P} \)-space reduces to \( L \) in polynomial time. Sliding block puzzles, such as Rush Hour, are \( \mathcal{P} \)-space-complete.

A language \( L \) in EXP-time is EXP-time-complete if every language in EXP-time reduces to \( L \) in polynomial time. 2-person board games such as generalized checkers and generalized chess are EXP-time complete. That is, the set of all generalized chess configurations from which White can force a win is EXP-time-complete.

A language \( L \) in NEXP-time is NEXP-time-complete if every language in NEXP-time reduces to \( L \) in polynomial time.

A language \( L \) in EXP-space is EXP-space-complete if every language in EXP-space reduces to \( L \) in polynomial time.

\( \mathcal{P} \)-completeness is defined differently. A \( \mathcal{P} \)-time language is \( \mathcal{P} \)-complete if every \( \mathcal{P} \)-time language can be reduced to \( L \) by an \( \mathcal{NC} \) function. CVP is \( \mathcal{P} \)-complete.