# Review for the Final Exam

A. Regular Languages
- DFAs, NFAs, $\varepsilon$-NFAs   You should be able to convert any of the others to a DFA.
- Regular Expressions.  It is fairly easy to convert a regular expression to a DFA.  It is possible but harder to convert a DFA to a regular expression.
- The Pumping Lemma:  If $|w| > p$ then $w=xyz$ where $|xy|<=p$, y is not empty, and $xy^iz$ is in the language for all $i>= 0$.
- Properties of Regular Languages:  Unions, Intersections, Differences and Complements of regular languages are regular.

B. Context-Free Languages
- Grammars
- PDAs
- To show that grammars generate the same languages as PDAs we found algorithms to convert a grammar to a PDA (easy) and to convert a PDA to a grammar (hard). I won't ask you  to do the latter on the final exam.
- Chomsky Normal Form and the algorithm for finding a CNF grammar equivalent to a given grammar.
- The Pumping Lemma for Context-Free languages:  If $|z| > p$ then $z=uvwxy$ where $|vwx|<=p$, v and x aren't both empty, and $uv^iwx^iy$ is in the language for all $i>= 0$.
- Properties of CF Languages:  Unions and concatenations of CF languages are CF. Intersections and Complements of CF languages are not necessarily CF.

C. Turing Machines
- Simple TMs, multli-track, multi-tape and non-deterministic TMs
- Church's Thesis: TMs embody our notion of an algorithm

D. Decidability
- Recursive languages, Recursively enumerable languages, Decidable problems, Recognizable problems
- The diagonal language $\mathcal{L}_d$={M | M does not accept its own encoding} is not RE.
- The universal language $\mathcal{L}_u$ = {(M,w) | M accepts w} is RE but not Recursive. The complement of $\mathcal{L}_u$ is not RE.
- The halting language $\mathcal{L}_{halt}$ = {(M.w) | M halts on input w} is RE but not recursive.
- Rice's Theorem: Any nontrivial property of context-free languages is undecidable.

E. NP-Completeness

- $\mathscr{P}$ is the class of problems that can be solved deterministically in polynomial time
- $\mathscr{NP}$ is the class of problems that can be solved non-deterministically in polynomial time, which usually means that a solution can be verified deterministically in polynomial time.
- A problem is NP-hard if all NP problems reduce to it.
- A problem is NP-Complete if it is both in $\mathscr{NP}$ and NP-hard. If any NP-Complete problem was in $\mathscr{P}$ then $\mathscr{P}$ would equal $\mathscr{NP}$.
- Cook's (or Cook-Levin) Theorem: SAT is NP-Complete.
- CNF-SAT and 3CNF-SAT are both NP-Complete.
- You should know what all of this means, but I am unlikely to ask you to prove that a specific language is NP-Complete.