# Advanced Analysis of Algorithms

K. Subramani, LCSEE, West Virginia University

November 19, 2013

1. **NP-complete** and **P** refer to set of languages. Languages are sets.
2. Non-determinism. Main idea is from logic and theorem proving.
3. Reductions or transformations.
4. Limits on $f$. Why needed? Polynomial time transformations.
5. If $L_1 \leq_p L_2$ and $L_2$ is in **P**, then $L_1$ is in **P**.
6. Transitivity.
7. If $L_1$ and $L_2 \in$ **NP**, $L_1 \leq_p L_2$ and $L_1$ is **NPC**, then $L_2$ is in **NPC**.
8. Another way of thinking: How can I use an algorithm for problem $L_2$ to solve problem $L_1$?
9. Reductions order problems just like $\leq$ orders numbers.
10. Relation between **P** and **NP**. Use SAT and knapsack as example.
11. **NP-completeness** is for decision problems only. For optimization problems, use a target.
12. Some common reductions. **HC** to TSP. Graph-coloring to SAT.
13. We need the first NPC problem. Cook's theorem.
14. Steps to show a problem is **NPC**:
    (a) Show that it is in **NP**.
    (b) Start with a good, **NP-complete** problem, say $\mathbf{P_1}$,
    (c) Find a suitable, polynomial-time transducer function $f$.
    (d) Reduce $P_1$ to our problem, using $f$.
15. 3SAT, 0/1 Integer Programming, Circuit-SAT.