Advanced Analysis of Algorithms

K. Subramani, LCSEE, West Virginia University

November 14, 2013

1. Alphabet, strings, languages.
2. Language associated with a decision algorithm $\mathcal{A}$.
3. Deterministic and non-deterministic algorithms (Show picture).
4. Time complexity functions associated with both types of algorithms.
5. Deterministic algorithms and the class **P**.
6. Non-deterministic algorithms and the class **NP**. Guess and check algorithm. Easy verifiability. The tree of computations method to illustrate non-determinism. Show picture.
7. Without loss of generality, assume that degree of non-determinism is 2.
8. Time taken by a non-deterministic algorithm. Count the worst on any string of length n. Contrast with deterministic algorithm. You count the depth of the tree and not the total number of computations.
9. Main idea is from logic and theorem proving.
10. Simple problems. SAT, Circuit SAT, Vertex cover, clique, independent set, Independent set, hamilton path, hamilton circuit, TSP, Max cut, 2-Partitioning, scheduling on identical parallel machines, subset-sum, 0/1 knapsack.
11. Non-deterministic algorithms for simple problems.
12. Relationship between **P** and **NP**.
13. Notion of transformations. $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ A function $f : \Sigma_1^* \to \Sigma_2^*$, such that $\forall x \in \Sigma_1^*$, $x \in L_1 \leftrightarrow f(x) \in L_2$. Also called reductions. f is called a transducer. Denoted by $L_1 \leq L_2$.
14. HC. to TSP example.
15. Limits on f. Why needed? Polynomial time transformations. Mention log-space. Mention many-to-one.
16. If $L_1 \leq_p L_2$ and L_2 is in **P**, then L_1 is in **P**.
17. Transitivity of reductions.
18. Definition of NP complete. NP-hard. Optimization problems. TSP example.
19. If $L_1 \leq_p L_2$ and L_1 is **NPC**, then L_2 is in **NPC**.