CS 500: Introduction to Theory of Computation

Instructor: Deepak Kapur, kapur@cs.unm.edu, x7-1581, FEC 3140.

Course Contents

1. History and overview: origins of theory of computation from logic and mathematics; machines, languages, complexity.

   - influence of natural language processing – Chomsky
   - modeling of brain and neurons – finite state machines
   - hardware and programming languages – finite state machines, stack, push down automaton, logic, lambda calculus
   - various models of computations – lambda calculus (logic and programming languages), Turing machines (machines), recursive functions (logic, programming languages), Post systems (string rewriting and replacement), Thue systems (string rewriting and replacement)
   - determinism and nondeterminism

2. A brief introduction to propositional logic connectives, quantifiers, proofs by induction and proofs by contradiction.

3. Finite State Automata, Regular Languages and Regular Grammars.

   - Definitions of DFA and NDFA, regular languages and regular expressions, particularly distinction between acceptance of a string by a DFA vs NDFA.
   - Closure Properties Regular languages are closed under union, concatenation, intersection and complement.
   - Equivalences:
     (a) NDFA and DFA are equivalent: languages accepted by NDFAs are precisely regular languages.
     (b) For every regular expression, there is a NDFA accepting the language generated by the regular expression.
     (c) From a given NDFA, a regular expression that generates the regular language of NDFA can be constructed.
     NDFA, DFA and regular expressions are equivalent in expressive power. I.e, they all accept regular languages.

- Regular grammar and derivation of strings. Chomsky Hierarchy – Type 0, 1, 2 and 3 grammars.
- Constructing a NDFA from a regular grammar and vice versa. Arden's Lemma.
- Generalized DFAs to regular expressions.
- Minimization of DFAs. Equivalence classes of strings induced by a DFA. Statement of Myhill-Nerode Theorem.
- There exist nonregular languages, i.e. no NDFA can be constructed that accepts such a language. Pumping Lemma: Construction to show when a language is not regular.

4. Context free languages (CFLs), context free grammars (CFGs) and nondeterministic push down automata (NPDA)

   (a) Role of a stack in acceptance of languages.

   (b) Derivation/parse trees, a language geenrated by a grammar.

   (c) Conversion from a context free grammar to a NPDA and vice versa.

   (d) Ambiguous grammars and languages.

   (e) Chomsky Normal Form

   (f) Closure properties of context free languages.

   (g) Deterministic push down automata – a brief overview. A mention of deterministic context free grammars and context-sensitive languages and grammars.

   (h) Pumping Lemma for context free languages – generalization of pumping lemma for regular languages.

5. Computability, Church-Turing thesis, an overview of various formalisms for computability.

6. Turing Machines, Decidability and Undecidability.

   (a) Deterministic TMs, nondeterministic TMs, multi-head TMs and multi-tape TMs and their equivalence.

   (b) Definitions of a computable function, a decidable language, a recognizable language. Recursive languages, recursively enumerable (re) languages, co-re languages. Closure properties of Turing languages. Undecidability Hierarchy.

   (c) Cantor's theorem and Diagonalization arguments. Cardinality of a set. Countable sets and uncountable sets.

(d) Universal Turing Machine. Undecidability of the language consisting of pairs of a TM description and a string accepted by the TM. Showing the undecidability of the halting problem by reducing it to the acceptance problem.

(e) Informal statement of Rice's Theorem, namely every nontrivial property of languages is undecidable (a property is trivial if and only it is either true for no language or true for every language).

7. Post Systems. Statement of the Post correspondence problem. Informal statements of Hilbert's 10th problem and its undecidability, Tarski's decidability result about theory of real closed field, Decidability of Presburger arithmetic (formulas expressed using + and $\geq$ on natural numbers), Goedel-Herbrand's completeness theorem for first-order logic, Goedel's incompleteness theorem about Peano arithmetic.

8. Reductions – multilevel reduction, Turing reductions, Oracles, computable reductions, polynomial time reductions. To show decidability of a language, it suffices to reduce it by a total computable function to a decidable language. To show undecidability of a language, it is necessary to reduce a known undecidable language by a total computable function to it (the contrapositive of the previous statement).

9. Lambda calculus – α and β rules, Church-Rosser Theorem , Y operators, Turing numerals.

10. Goedel and Kleene's primitive recursive functions, μ-recursive functions. Partial functions and total functions. Existence of a total function that is not primitive recursive – Sudan and Ackermann's function.

11. Time Complexity, Reducibility, Class Complete Problems.

    (a) Small o and Big O notations. Polynomial Time Hierarchy. Polynomial Time (P), Nondeterministic Polynomia Time (NP)l, Exponential Time (EXPTIME), Nondeterministic Exponential Time (EXPTIME),

    (b) NP completeness, polynomial time reductions.

    (c) Cook-Levin Theorem and its proof: SAT is NP complete.

    (d) Reductions from SAT to 3-SAT. Reduction of 3-SAT to k-clique.

    (e) Proving NP-completeness: (i) show in NP (verifiability vs solvability) and (ii) prove NP-hardness.

    Reduction from k-clique to independence set and vertex cover. Graph subisomorphism, Hamiltonian path, Hamiltonian Circuit. Graph coloring. Longest Path.

12. David Putnam for Unsatisfiability (CO-NP complete) - chronological backtracking. Generalization by Davis Putnam Loveland and Longemann: unit propagation and pure literal.

13. Conflict driven Clause Learning.

14. Space Complexity: LogSpace, PSPACE, NPSPACE, EXPSPACE. Savitch's theorem and its proof: NPSPACE($s(n)$) = PSPACE($s^2(n)$). QBF is PSPACE complete.

15. Gap Theorems.