CSE 1002C TH: Theory of Computation

Credit:4

Course Outcomes (CO):

i. Understand the requirements and applications of each phase of compiler and various compilation techniques needed to obtain high performance on modern computer architectures.
ii. Understand different optimization techniques and difference between them.
iii. Understand formal language theory and its application to computer science.
iv. Apply mathematical preliminaries to develop the basic components of language design.
v. Design simple computational machines using the concepts of language theory Correlate computability with formal computational machines.

Syllabus:

Optimization and decision problems, Reductions, Turing Machine as an acceptor and as an enumerator— **Techniques of Turing Machine construction** – parallel tracks and storage in control, subroutine Turing Machine, Church-Turing thesis, **Variants of Turing Machine** –multitape, **nondeterministic**—their equivalences with other models. Properties of recursively enumerable and recursive sets. Relations between unrestricted grammars and Turing Machines. **Linear Bounded Automata** —relation with Context Sensitive Languages Enumeration of Turing Machines, existence of undecidable problems, Undecidable problems involving Turing Machines and CFG's. Universal Turing Machine as a model of general-purpose computer, **Post Correspondence Problem** – Applications, valid and invalid computations of Turing Machines. Time and Space complexity of Turing Machines, NP- completeness.

References:

1. John C. Martin: Introduction to languages and the theory of computation, 2nd Ed., McGraw Hill.
2. D.P. Bovet & P. Gescenzi: Introduction to Theory of Complexity, PH.
3. Rozenberg&Salomaa: Handbook of Formal languages, Vol. I&II.

Introduction: Introduction to language theory, tokens. Alphabets, definition of grammar Production rules, sentences, sentential forms, language definitions, derivations. **Regular languages:** Pumping Lemma of regular sets, Minimization of finite automata. Chomsky Hierarchy of languages. **Finite Automata:** Finite automaton, Deterministic, Non-Deterministic and equivalence. Transition diagrams, epsilon transitions, Equivalence of regular expressions and FA. Moore and Mealy machines. **Context Free Language:**

Relations between classes of languages, Context Free Grammar, Derivation trees, ambiguity simplification, Normal forms, applications. **Lexical Analysis:** Interface with input, parser and symbol table, token, lexeme and patterns, difficulties in lexical analysis, error reporting, and implementation. Regular definition, Transition diagrams, LEX. **Syntax analysis:** context free grammars, ambiguity, associativity, precedence, top down parsing, recursive descent parsing, transformation on the grammars, predictive parsing, Bottom up parsing, operator precedence grammars, LR parsers (SLR, LALR, LR), YACC. **Pushdown Automata:** Pushdown automata, definitions, context free languages, construction of PDA for simple CFLs, Linear bounded automata. **Turing machines:** Turing machines, Introduction to computability, Universal Turing Machines, Types of Turing Machines, Techniques for construction of Turing machines, Halting problem. Assembler, Loader, Linker: basic concept; absolute and Relocatable, assemblers and macroprocessors Linkers- concept and design; loaders, different types. Editors and debuggers. Interpreters. Compilers: - Various phases; lexical analyzers- design. Parsing top down (L.L. (1) and recursive descent), bottom- up, (Shift – reduce concept to L.R. (1) symbol tables, error handling. Syntax – directed Translation – attributes and intermediate codes. Optimization concepts and machine code. Generation Use of LEX and YACC.

Reference Books:

1. John C. Martin: Introduction to languages and the theory of computation, 2nd Ed., McGraw Hill.
2. D.P. Bovet & P. Gescenzi: Introduction to Theory of Complexity, PH.
3. Rozenberg & Salomaa: Handbook of Formal languages, Vol. I&II.