|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| III Year I Semester | AUTOMATA THEORY & COMPILER DESIGN | L | T | P | C |
| 3 | 0 | 0 | 3 |

**Course Objectives:**

* To introduce the Fundamental concepts of formal languages, grammars and automata theory.
* To understand deterministic and non-deterministic machines and the differences between decidability and undecidability.
* Introduce the major concepts of language translation and compiler design and impart the knowledge of practical skills necessary for constructing a compiler.
* Topics include phases of compiler, parsing, syntax directed translation, type checking use of symbol tables, intermediate code generation.

**Course Outcomes:**

* Able to employ finite state machines for modeling and solving computing problems.
* Able to design context free grammars for formal languages.
* Able to distinguish between decidability and undecidability.
* Demonstrate the knowledge of patterns, tokens & regular expressions for lexical analysis.
* Acquire skills in using lex tool and design LR parsers.

**UNIT-I**

**Introduction to Finite Automata:** Structural Representations, Automata and Complexity, the Central Concepts of Automata Theory–Alphabets, Strings, Languages, Problems.

**Non deterministic Finite Automata:** Formal Definition, an application, Text Search, Finite Automata with Epsilon-Transitions.

**Deterministic Finite Automata:** Definition of DFA, How ADFA Process Strings, The language of DFA, Conversion of NFA with €-transitions to NFA without €-transitions. Conversion of NFA to DFA

**UNIT-II**

**Regular Expressions:** Finite Automata and Regular Expressions, Applications of Regular Expressions, Algebraic Laws for Regular Expressions, Conversion of Finite Automata to Regular Expressions.

**Pumping Lemma for Regular Languages:** Statement of the pumping lemma, Applications of the Pumping Lemma.

**Context-Free Grammars:** Definition of Context-Free Grammars, Derivations Using a Grammar, Left most and Right most Derivations, the Language of a Grammar, Parse Trees, Ambiguity in Grammars and Languages.

**UNIT-III**

**Push Down Automata:** Definition of the Pushdown Automaton, the Languages of a PDA, Equivalence of PDA's and CFG's, Acceptance by final state.

**Turing Machines:** Introduction to Turing Machine, Formal Description, Instantaneous description, The language of a Turing machine.

**Undecidability:** Undecidability, A Language that is Not Recursively Enumerable, An Undecidable Problem That is RE, Undecidable Problems about Turing Machines

**UNIT-IV**

**Introduction:** The structure of a compiler, Lexical Analysis: The Role of the Lexical Analyzer, Input Buffering, Recognition of Tokens, The Lexical-Analyzer Generator Lex. Syntax Analysis: Introduction, Context-Free Grammars, Writing a Grammar, Top- Down Parsing, Bottom-Up Parsing, Introduction to LR Parsing: Simple LR, More Powerful LR Parsers.

**UNIT-V**

**Syntax-Directed Translation:** Syntax-Directed Definitions, Evaluation Orders for SDD's, Syntax-Directed Translation Schemes, Implementing L-Attributed SDD's.

**Intermediate-Code Generation:** Variants of Syntax Trees, Three-Address Code. Run Time Environments: Stack Allocation of Space, Access to Non local Data on the Stack, Heap Management

**TEXT BOOKS:**

1. Introduction to Automata Theory, Languages, and Computation, 3ndEdition, John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Pearson Education.
2. Compilers: Principles, Techniques and Tools, Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffry D. Ullman,2ndEdition, Pearson.
3. Theory of Computer Science– Automata languages and computation, Mishraand Chandra shekaran, 2ndEdition, PHI.

**REFERENCE BOOKS:**

1. Introduction to Formal languages Automata Theory and Computation, Kamala Krithivasan, Rama R, Pearson.
2. Introduction to Languages and The Theory of Computation, John C Martin, TMH.
3. Lex & yacc–John R. Levine, Tony Mason, Doug Brown, O’reilly
4. Compiler Construction, Kenneth C. Louden, Thomson. Course Technology.