19IT212 FORMAL LANGUAGES & AUTOMATA THEORY AND COMPILER DESIGN

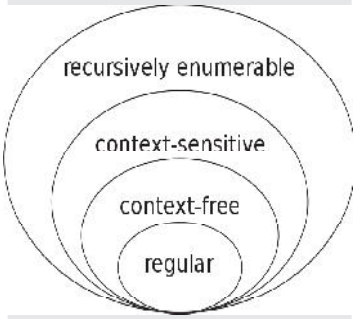
Hours Per Week :

L	T	P	C
3	-	-	3

Total Hours :

L	T	P
45	-	-

CS	WA/RA	SSH	SA	S	BS
5	5	30	20	5	5



source:
https://en.wikipedia.org/wiki Chomsky_hierarchy

PREREQUISITE COURSES: Data Structures; Discrete Mathematical Structures

COURSE DESCRIPTION AND OBJECTIVES:

This course introduces the concepts of design Deterministic finite Automata, Non-Deterministic finite Automata, lexical analyzer, parser, code generation and code optimization techniques. The objective of this course is to enable the student to acquire the knowledge of various automata's and various phases of compiler.

COURSE OUTCOMES:

Upon completion of the course, the student will be able to achieve the following outcomes:

COs	Course Outcomes	POs
1	Apply different finite state machines for solving a given language.	1
2	Understand the different phases of compiler with various examples.	1
3	Apply different Parsing and optimization techniques in the design of compiler.	1,5
4	Analyze and compare various translators to select optimum.	1,2
5	Design a compiler for simple programming languages.	2,5,8,9,10

SKILLS:

- ✓ *Design automata, regular expressions and context free grammars for accepting or generating a certain language.*
- ✓ *Transform between equivalent deterministic and non-deterministic finite automata, and regular expressions.*
- ✓ *Design various parsers using top-down and bottom-up approaches.*
- ✓ *Construction of automata for regular expressions.*
- ✓ *Analyze recognizer for programming language.*
- ✓ *Usage of generators like LEX and YACC.*

UNIT– I **L-9**

INTRODUCTION: Alphabets, Strings and languages, Automata and grammars, Regular languages, Deterministic finite automata (DFA)-formal definition, simplified notation, state transition graph, transition table, language of DFA; Nondeterministic finite Automata (NFA), NFA with epsilon transition, Language of NFA, Equivalence of NFA and DFA, Minimization of Finite Automata.

REGULAR EXPRESSION: Definition, Algebraic laws for regular expressions, Operators of regular expression and their precedence,

UNIT– II **L-9**

INTRODUCTION TO COMPILING: Compilers, Analysis of the source program, Cousins of the compiler, Phases of compiler, Grouping of phases, compiler construction tools, Role of Lexical analysis, Specifications of Tokens, Lexical analyzer generator –LEX tool.

UNIT– III **L-9**

SYNTAX ANALYSIS: The Role of the parser and Context-free grammars. Top-Down parsing- Recursive Descent parsing, Predictive parsing, Bottom-Up parsing, Shift-Reduce parsing and Operator-precedence parsing; LR Parsers, SLR parser, CLR parser, and LALR Parser; YACC automatic parser generator.

UNIT– IV **L-9**

SEMANTIC ANALYSIS: Intermediate forms of source programs – abstract syntax tree, Polish notation and three address codes; Attributed grammars, Syntax directed translation, Conversion of popular Programming languages, Constructs into Intermediate code forms, Declarations, Assignment, Statements, Boolean expressions.

UNIT– V **L-9**

CODE OPTIMIZATION: Basic blocks and flow graphs, The Principal sources of optimization, Optimization of basic blocks, Introduction to global data flow analysis, Global optimization.

Code Generation: Issues in the design of code generator, The target machine, A simple Code generator, DAG representation of basic blocks, Peephole optimization.

TEXT BOOKS:

1. Hopcroft and Ullman, “Introduction to Automata Theory, Languages and Computation”, 2nd edition, Pearson/Prentice Hall India, 2007.
2. Aho, Ullman and Ravisethi, “Compilers Principles, Techniques and Tools”, 2nd edition, Pearson Education, 2014.
3. Sipser, “Introduction to Theory of computation”, 2nd edition, Thomson, 2016.

REFERENCE BOOKS:

1. Andrew W. Appel, “Modern Compiler Construction in C” , 1st edition, Cambridge University Press, 2004.
2. Anand Sharma, “Theory of automata and formal languages”, 2nd edition,Laxmi Publications, 2015.