CS3470 Compilers and code generation
Lecture Diary 2018/19

Lecture 1 Introduction, high and low level languages, why we need compilers. Aims of the course. Vocabulary and semantics. Course outline. Personal development. ACM/IEEE Code of ethics. BCS Code of good practice. Professional responsibility to be trustworthy and to be competent. Relating professional codes to compiler design.

Lecture 2 Compiler front and back ends, portability, interpreters. Stages of compilation. Overview of input buffering, lexical, syntax and semantic analysis, code improvement and generation. Multi-pass compilers. Syntax analyser as the front end manager. The need for an input buffer. Two-buffer input system.

Lecture 3 Tokens, lexemes and patterns. Regular expressions. Regular definitions. Finite state automata.

Lecture 4 Regular expressions and NFAs. Transition tables. Formal description of Thompson's construction. Example of building an NFA using Thompson's construction. Introduction to the subset construction.

Lecture 5 Subset construction and dead state minimisation. Examples.

Lecture 6 Building a complete lexer. LEX. The role of a symbol table. Structure of a symbol table.

Lecture 7 Hash coded symbol tables. Introduction to parsing. Context free grammars. Derivations and derivation trees.

Lecture 8 Top down and bottom up parsing. Ambiguity. LL parsing. Guiding the parser. First sets.

Lecture 9 First sets. Examples. Termination. Left recursion and immediate left recursion removal. Examples.

Lecture 10 Left recursion removal continued. Examples. Left factored grammars. Avoiding backtracking. Follow sets. Follow set calculation - Examples. LL(1) grammars. Recursive descent parsers.

Lecture 11 Recursive descent parsers - examples and LL(1) grammar example. EBNF. **rdp** briefly. First match and longest match in rd parsers. Resolving if-then-else.

Lecture 12 Lab1 – getting and running rdp. Building a parser for small grammars. Using rdp's IBNF. Running vcg. Looking at rdparser.c. Left recursion and left factoring. Expression grammar. Attributes and semantic actions in rdp, return values and input parameters.

Lecture 13 Introduction to bottom-up parsing. NFAs from grammars. Parsing with NFAs. Generating DFAs. LR(0)-items and LR(0) DFA direct construction. (All with Examples.)

Lecture 14 LR(0) DFA direct construction continued. Parsing with DFAs. Parse tables and LR(0) grammars. Using a stack to parse.

Lecture 15 LR(0) example, non-LR(0) example. Using lookahead with follow sets. SLR(1) tables. Examples of direct DFA construction. LR(0) versus SLR(1) on incorrect strings. Introduction to generalised parsing.

Lecture 16 Worked solutions to Assignment 1.

Lecture 17 Lab sheet 1 completed. Conflicts in SLR(1) tables. LR(1) and LALR tables. YACC. Resolving ambiguity in an LR parser. Longest match conflict resolution in an LR parser.

Lecture 18 Explicit stack handling in RD parsers. Using elementary descriptors with non-LL(1) grammars.

Lecture 19 Combining stacks into a GSS. Full descriptors. Popping nodes in a GSS. The create() function and the set $\mathcal{P}$. Example of the full GLL algorithm. GLL constructing templates.

Lecture 20 Syntax Directed Translation. Translation schemes. Top down translation. Recursive descent parsers as top down translators. Semantics in rdp. Symbol tables in rdp. If statements with guards on actions. Why top down translation is not enough.

Lecture 21 Lab 2 – Inherited attributes and conditionals. Symbol tables in rdp. Limitations of the parser as a compiler and the need for intermediate form. MVM assembley language.

Lecture 22 Semantic evaluation. Attributes and annotated parse trees. Inherited and synthesised attributes. Attribute Grammars. S-attribute grammars. L-attribute grammars.

Lecture 23 Worked solutions to Assignment 2.

Lecture 24 Further worked solutions to Assignment 2. ASTs and Reduced derivation trees. Promotion operators in rdp. Derivation trees from IBNF – EPSILON_TREE.

Lecture 25 Three address code. Three address code using syntax directed definitions. Flow of control in three address code. If statements. While loops.

Lecture 26 Do-while, Do-until. Generating three address code, example. Array indexing in three address code. Basic Blocks. Code improvement within basic blocks. Common subexpression elimination.

Lecture 27 Lab3 – MVM assembley language and miniloop.

Lecture 28 Dead code elimination, variable renaming, algebraic simplificaiton, copy propogation. Control flow graphs. Dominators and natural loops. Loop invariants. Code motion, code hoisting. Loop fusion.

Lecture 29 Code improvement using DAGs. Reconstructing code from DAGs - example. Error dectection, reporting and recovery in compilers. Types of errors.

Lecture 30 Worked solutions to Assignment 3.

Lecture 31 Panic mode recovery, symchronising sets. Overview of loading, linking, scheduling, code selection, register allocation.

Lecture 32 Discussion and worked solution to Assignment 3, Question 4.