

Contents

1	Introduction	1
1.1	What is a compiler?	1
1.2	The phases of a compiler	2
1.3	Interpreters	3
1.4	Why learn about compilers?	4
1.5	The structure of this book	5
1.6	To the lecturer	6
1.7	Acknowledgements	7
1.8	Permission to use	7
2	Lexical Analysis	9
2.1	Introduction	9
2.2	Regular expressions	10
2.2.1	Shorthands	12
2.2.2	Examples	13
2.3	Nondeterministic finite automata	15
2.4	Converting a regular expression to an NFA	18
2.4.1	Optimisations	20
2.5	Deterministic finite automata	22
2.6	Converting an NFA to a DFA	22
2.6.1	Solving set equations	23
2.6.2	The subset construction	26
2.7	Size versus speed	29
2.8	Minimisation of DFAs	30
2.8.1	Example	32
2.8.2	Dead states	34
2.9	Lexers and lexer generators	36
2.9.1	Lexer generators	41
2.10	Properties of regular languages	43
2.10.1	Relative expressive power	43

2.10.2	Limits to expressive power	45
2.10.3	Closure properties	45
2.11	Further reading	46
	Exercises	47
3	Syntax Analysis	53
3.1	Introduction	53
3.2	Context-free grammars	54
3.2.1	How to write context free grammars	56
3.3	Derivation	58
3.3.1	Syntax trees and ambiguity	60
3.4	Operator precedence	63
3.4.1	Rewriting ambiguous expression grammars	65
3.5	Other sources of ambiguity	68
3.6	Syntax analysis	69
3.7	Predictive parsing	69
3.8	<i>Nullable</i> and <i>FIRST</i>	70
3.9	Predictive parsing revisited	73
3.10	<i>FOLLOW</i>	75
3.11	LL(1) parsing	77
3.11.1	Recursive descent	78
3.11.2	Table-driven LL(1) parsing	79
3.11.3	Conflicts	80
3.12	Rewriting a grammar for LL(1) parsing	82
3.12.1	Eliminating left-recursion	82
3.12.2	Left-factorisation	84
3.12.3	Construction of LL(1) parsers summarized	85
3.13	SLR parsing	86
3.14	Constructing SLR parse tables	90
3.14.1	Conflicts in SLR parse-tables	93
3.15	Using precedence rules in LR parse tables	94
3.16	Using LR-parser generators	97
3.16.1	Declarations and actions	97
3.16.2	Abstract syntax	98
3.16.3	Conflict handling in parser generators	101
3.17	Properties of context-free languages	103
3.18	Further reading	104
	Exercises	104

4	Symbol Tables	111
4.1	Introduction	111
4.2	Symbol tables	112
4.2.1	Implementation of symbol tables	112
4.2.2	Simple persistent symbol tables	113
4.2.3	A simple imperative symbol table	114
4.2.4	Efficiency issues	115
4.2.5	Shared or separate name spaces	115
4.3	Further reading	116
	Exercises	116
5	Interpretation	117
5.1	Introduction	117
5.2	The structure of an interpreter	118
5.3	A small example language	118
5.4	An interpreter for the example language	120
5.4.1	Evaluating expressions	120
5.4.2	Interpreting function calls	123
5.4.3	Interpreting a program	123
5.5	Advantages and disadvantages of interpretation	123
5.6	Further reading	127
	Exercises	127
6	Type Checking	129
6.1	Introduction	129
6.2	The design space of types	129
6.3	Attributes	131
6.4	Environments for type checking	132
6.5	Type checking expressions	132
6.6	Type checking of function declarations	135
6.7	Type checking a program	135
6.8	Advanced type checking	138
6.9	Further reading	140
	Exercises	140
7	Intermediate-Code Generation	143
7.1	Introduction	143
7.2	Choosing an intermediate language	144
7.3	The intermediate language	146
7.4	Generating code from expressions	147
7.4.1	Examples of translation	151

7.5	Translating statements	152
7.6	Logical operators	155
7.6.1	Sequential logical operators	156
7.7	Advanced control statements	159
7.8	Translating structured data	161
7.8.1	Floating-point values	161
7.8.2	Arrays	161
7.8.3	Strings	167
7.8.4	Records/structs and unions	167
7.9	Translating declarations	168
7.9.1	Example: Simple local declarations	168
7.10	Further reading	169
	Exercises	170
8	Machine-Code Generation	175
8.1	Introduction	175
8.2	Conditional jumps	176
8.3	Constants	177
8.4	Exploiting complex instructions	177
8.4.1	Two-address instructions	182
8.5	Optimisations	182
8.6	Further reading	184
	Exercises	185
9	Register Allocation	187
9.1	Introduction	187
9.2	Liveness	188
9.3	Liveness analysis	189
9.4	Interference	193
9.5	Register allocation by graph colouring	194
9.6	Spilling	197
9.7	Heuristics	198
9.7.1	Removing redundant moves	201
9.8	Further reading	202
	Exercises	202
10	Function calls	205
10.1	Introduction	205
10.1.1	The call stack	205
10.2	Activation records	206
10.3	Prologues, epilogues and call-sequences	207

10.4	Caller-saves versus callee-saves	209
10.5	Using registers to pass parameters	212
10.6	Interaction with the register allocator	215
10.7	Accessing non-local variables	217
10.7.1	Global variables	217
10.7.2	Call-by-reference parameters	218
10.7.3	Nested scopes	219
10.8	Variants	222
10.8.1	Variable-sized frames	223
10.8.2	Variable number of parameters	223
10.8.3	Direction of stack-growth and position of FP	224
10.8.4	Register stacks	224
10.9	Further reading	224
	Exercises	225
11	Analysis and optimisation	227
11.1	Data-flow analysis	228
11.2	Common subexpression elimination	229
11.2.1	Available assignments	229
11.2.2	Example of available-assignments analysis	232
11.2.3	Using available assignments for common subexpres- sion elimination	233
11.3	Jump-to-jump elimination	236
11.4	Index-check elimination	237
11.5	Limitations of data-flow analyses	240
11.6	Loop optimisations	241
11.6.1	code hoisting	241
11.6.2	Memory prefetching	242
11.7	Optimisations for function calls	245
11.7.1	Inlining	245
11.7.2	Tail call optimisation	246
11.8	Specialisation	249
11.9	Further reading	251
	Exercises	251
12	Bootstrapping a compiler	253
12.1	Introduction	253
12.2	Notation	254
12.3	Compiling compilers	256
12.3.1	Full bootstrap	257
12.4	Further reading	260

Exercises 260

List of Figures

2.1	Regular expressions	11
2.2	Some algebraic properties of regular expressions	14
2.3	Example of an NFA	17
2.4	Constructing NFA fragments from regular expressions	19
2.5	NFA for the regular expression $(a b)^*ac$	20
2.6	Optimised NFA construction for regular expression shorthands	21
2.7	Optimised NFA for $[0-9]^+$	21
2.8	DFA constructed from the NFA in figure 2.5	29
2.9	Non-minimal DFA	32
2.10	Minimal DFA	34
2.11	Combined NFA for several tokens	38
2.12	Combined DFA for several tokens	39
3.1	From regular expressions to context free grammars	56
3.2	Simple expression grammar	57
3.3	Simple statement grammar	57
3.4	Example grammar	59
3.5	Derivation of the string aabbbcc using grammar 3.4	59
3.6	Leftmost derivation of the string aabbbcc using grammar 3.4	60
3.7	Syntax tree for the string aabbbcc using grammar 3.4	61
3.8	Alternative syntax tree for the string aabbbcc using grammar 3.4	61
3.9	Unambiguous version of grammar 3.4	62
3.10	Preferred syntax tree for $2+3*4$ using grammar 3.2	64
3.11	Unambiguous expression grammar	67
3.12	Syntax tree for $2+3*4$ using grammar 3.11	67
3.13	Unambiguous grammar for statements	68
3.14	Fixed-point iteration for calculation of <i>Nullable</i>	72
3.15	Fixed-point iteration for calculation of <i>FIRST</i>	73
3.16	Recursive descent parser for grammar 3.9	79
3.17	LL(1) table for grammar 3.9	80

3.18	Program for table-driven LL(1) parsing	80
3.19	Input and stack during table-driven LL(1) parsing	81
3.20	Removing left-recursion from grammar 3.11	83
3.21	Left-factorised grammar for conditionals	85
3.22	SLR table for grammar 3.9	89
3.23	Algorithm for SLR parsing	89
3.24	Example SLR parsing	90
3.25	Example grammar for SLR-table construction	90
3.26	NFAs for the productions in grammar 3.25	91
3.27	Epsilon-transitions added to figure 3.26	92
3.28	SLR DFA for grammar 3.9	92
3.29	Summary of SLR parse-table construction	94
3.30	Textual representation of NFA states	102
5.1	Example language for interpretation	119
5.2	Evaluating expressions	122
5.3	Evaluating a function call	124
5.4	Interpreting a program	125
6.1	The design space of types	130
6.2	Type checking of expressions	133
6.3	Type checking a function declaration	136
6.4	Type checking a program	137
7.1	The intermediate language	146
7.2	A simple expression language	148
7.3	Translating an expression	150
7.4	Statement language	152
7.5	Translation of statements	153
7.6	Translation of simple conditions	154
7.7	Example language with logical operators	157
7.8	Translation of sequential logical operators	158
7.9	Translation for one-dimensional arrays	162
7.10	A two-dimensional array	164
7.11	Translation of multi-dimensional arrays	165
7.12	Translation of simple declarations	169
8.1	Pattern/replacement pairs for a subset of the MIPS instruction set	181
9.1	Gen and kill sets	190
9.2	Example program for liveness analysis and register allocation .	191