

Contents

1	Introduction	1
1.1	Language Composition in Practice	1
1.2	Outline	6
1.3	Origin of Chapters	8
2	Concrete Syntax for Objects	9
2.1	Introduction	9
2.2	Concrete Syntax for Objects	12
2.2.1	Code Generation	12
2.2.2	XML Document Generation	15
2.2.3	Graphical User Interface Construction	16
2.2.4	Other Applications	18
2.3	Realizing Concrete Syntax	19
2.3.1	Embedding and Assimilation	19
2.3.2	Java with Swul	21
2.3.3	Java with XML	25
2.3.4	Java with Java	27
2.4	Syntax Definition	30
2.4.1	SDF Overview	31
2.4.2	The Importance of Modularity	33
2.4.3	The Importance of Scannerless Parsing	34
2.5	Previous Work	37
2.6	Related Work	37
2.6.1	Extensible Syntax	37
2.6.2	Harmonia's Blender	38
2.6.3	Jakarta Tool Suite (JTS)	39
2.6.4	Syntax Macros	40
2.6.5	Lexical Macros	42
2.6.6	Metafront	42
2.7	Future Work	43
2.7.1	Application Domains	43
2.7.2	Open Compilers	44
2.8	Conclusions	45
3	Type-based Disambiguation of Concrete Object Syntax	47
3.1	Introduction	47
3.2	Metaprogramming with Concrete Object Syntax	50
3.2.1	Embedding	50
3.2.2	Assimilation	51
3.3	Ambiguity in Concrete Object Syntax	52
3.3.1	Causes of Ambiguity	52

3.3.2	Solutions	53
3.4	Generalized Type-Based Disambiguation	54
3.4.1	Syntax Definition and Parsing	55
3.4.2	Assimilation	55
3.4.3	Type-Based Disambiguation	56
3.4.4	Explicit Disambiguation	58
3.5	Experience	58
3.5.1	JavaJava	59
3.5.2	Meta-AspectJ	59
3.6	Discussion	60
3.6.1	Previous Work	60
3.6.2	Related Work	61
3.6.3	Future Work	62
3.7	Conclusion	63
4	Preventing Injection Attacks	65
4.1	Introduction	65
4.2	Approach	69
4.2.1	Overview	70
4.2.2	Syntax Embedding and Parsing	71
4.2.3	API Generation	73
4.2.4	Assimilation	78
4.2.5	Summary of Language Independence	80
4.3	Discussion	80
4.3.1	Static versus dynamic type-checking	81
4.3.2	Prevented classes of injection attacks	81
4.3.3	External queries	84
4.3.4	Generalized parsing techniques	84
4.3.5	Disambiguation design space	85
4.4	Related work	87
4.4.1	Explicit escaping and filtering	87
4.4.2	APIs	87
4.4.3	LINQ	88
4.4.4	Static analysis techniques	88
4.4.5	Runtime detection techniques	89
4.4.6	SQL-specific techniques	90
4.4.7	MetaBorg	90
4.5	Conclusion	91
5	Syntax Definition for AspectJ	93
5.1	Introduction	93
5.2	Scanning and Parsing	96
5.2.1	Tokenization or Scanning	96
5.2.2	Scanner and Parser Generators	97
5.2.3	Lexical Context	98
5.2.4	Programmatic Parsers	98

5.2.5	Scannerless Generalized LR Parsing	98
5.3	A Quick Introduction to AspectJ	99
5.4	Issues in Parsing AspectJ	101
5.5	The ajc Scanner and Parser	102
5.5.1	Parsing Pointcuts	102
5.5.2	Parameterized Types	105
5.5.3	Pseudo Keywords	106
5.6	The abc Scanner and Parser	108
5.6.1	Managing Lexical State	109
5.6.2	Parser	113
5.7	Summary and Discussion	114
5.8	A Declarative Syntax Definition for AspectJ	115
5.8.1	Integrating Lexical and Context-Free Syntax	116
5.8.2	Composing AspectJ	119
5.8.3	Disambiguation and Restrictions	121
5.8.4	Grammar Mixins	124
5.8.5	AspectJ in the Mix	125
5.8.6	ABC Compatibility Revised	127
5.9	AspectJ Syntax Extensions	128
5.9.1	Issues in Extensibility	128
5.9.2	Simple Extensions	129
5.9.3	Open Modules	130
5.9.4	Context-Aware Aspects	131
5.10	Performance	133
5.10.1	Benchmark Setup	136
5.10.2	Benchmark Results	136
5.10.3	Testing	137
5.11	Discussion	137
5.11.1	Previous Work	137
5.11.2	Related Work	139
5.11.3	Future Work	140
5.12	Conclusion	142
6	Parse Table Composition	145
6.1	Introduction	145
6.2	Motivation	147
6.2.1	Program Transformation and Generation	147
6.2.2	Language Extension and Embedding	148
6.2.3	Language Conglomerates	151
6.2.4	Requirements	152
6.3	Grammars and Parsing	153
6.3.1	Context-free Grammars	153
6.3.2	LR Parsing	154
6.3.3	Generating LR Parse Tables	154
6.4	LR Parser Generation: A Different Perspective	157
6.4.1	Generating LR(o) ϵ -NFA	157

6.4.2	Eliminating ϵ -Transitions	158
6.5	Composition of LR(o) Parse Tables	159
6.5.1	Generating LR(o) Parse Tables Components	162
6.5.2	Composing LR(o) Parse Table Components	163
6.5.3	Optimization	165
6.6	Extension to SLR	167
6.6.1	Nullable Nonterminals	170
6.6.2	First and Follow Sets	170
6.6.3	Algorithm	171
6.7	Extensions for Lexical Analysis	174
6.7.1	Scannerless Parsing	175
6.7.2	Context-Specific Layout	177
6.8	Evaluation	177
6.9	Related Work	178
7	Precedence Rule Recovery	183
7.1	Introduction	183
7.2	Grammars and Tree Patterns	186
7.2.1	Context-Free Grammars	186
7.2.2	Parse Trees	186
7.2.3	Parse Tree Patterns	186
7.3	Precedence Mechanisms	187
7.3.1	YACC	188
7.3.2	SDF	189
7.4	Precedence Rule Recovery	189
7.4.1	A Core Formalism for Precedence Rules	189
7.4.2	Tree Pattern Generation	191
7.4.3	Precedence Rule Recovery: YACC	191
7.4.4	Precedence Rule Recovery: SDF	194
7.5	Precedence Compatibility	195
7.5.1	Grammar Transformation	196
7.6	Evaluation	196
7.6.1	C99	197
7.6.2	PHP 5	198
7.7	Related Work	199
7.7.1	Grammar Engineering Vision	199
7.7.2	Grammar Engineering Tools	199
7.7.3	Grammar Testing	200
7.7.4	Pretty-Printing	200
7.8	Conclusion	200
8	Conclusion	203
8.1	Future Work	205
8.1.1	Scannerless Generalized LR Parser User Experience	205
8.1.2	Syntax Definition for Real Life Programming Languages	207
8.1.3	Module Systems for Grammar Formalisms	208

8.1.4	Grammar Engineering	209
8.1.5	Composition of Assimilations	209
	Bibliography	211
	Samenvatting	227
	Curriculum Vitae	231