

Contents

Preface	v
1. Introduction	1
1.1 Safe Code Generation	2
1.2 Homogeneous Code Generators	4
1.3 Heterogeneous Code Generators	6
1.3.1 Abstract Syntax Trees	6
1.3.2 Print Statements	7
1.3.3 Term Rewriting	7
1.3.4 Text-Templates	12
1.4 Conclusions	15
1.5 Improving the Quality of Code Generators	17
2. Preliminaries	19
2.1 Basic Definitions and Notations	19
2.2 Context-free Grammars	22
2.3 Regular Tree Grammars	25
2.4 Relations between CFL and RTL	26
2.5 Abstract Syntax Trees	28
2.6 Used Languages and Formalisms	29
2.6.1 The PICO Language	29
2.6.2 Syntax Definition Formalism	30
2.6.3 ATerms	33
3. The Unparser	35
3.1 Deriving Abstract Syntax Trees	36
3.2 The Unparser Generation Pattern	42
3.3 Unparser Completeness	46
3.4 Conclusions	49
4. The Metalanguage	51
4.1 Code Generators	52
4.2 The Unparser Complete Metalanguage	54
4.2.1 Template Evaluation	55
4.2.2 Block-Structured Symbol Table	57

4.2.3	Subtemplates	61
4.2.4	Match-Replace	66
4.2.5	Substitution	70
4.2.6	Conditional	71
4.2.7	Iteration	72
4.2.8	Unparser Completeness	73
4.3	Example: The PICO Unparser	75
4.4	Related Template Systems	77
4.4.1	ERb	77
4.4.2	Java Server Pages	81
4.4.3	Velocity	85
4.4.4	StringTemplate	88
4.4.5	Evaluation	90
4.5	Conclusions	92
5.	Syntax-Safe Templates	93
5.1	Syntax-Safe Templates	94
5.2	The Metalanguage Grammar	100
5.2.1	Shared Syntax	101
5.2.2	Subtemplates	102
5.2.3	Match-Replace	104
5.2.4	Substitution Placeholder	105
5.3	Grammar Merging	106
5.4	Similar Approaches	109
5.5	Conclusions	110
6.	Repleo: Syntax-Safe Template Evaluation	111
6.1	Syntax-Safe Evaluation	112
6.2	Substitution Placeholder	113
6.3	Match-replace Placeholder	114
6.4	Subtemplate Placeholder	115
6.5	Substitution Placeholder Revisited	116
6.6	Ambiguity Handling	118
6.7	Separator Handling	122
6.8	Repleo	123
6.9	Other Syntax-Safe Template Approaches	123
6.10	Case Studies	124
6.10.1	PICO Unparser	124
6.10.2	Java	126
6.10.3	XHTML	128
6.10.4	SQL	128
6.10.5	Multi-language Templates	130
6.10.6	Embedded Languages	135
6.11	Conclusions	135
7.	Case Studies	137
7.1	Code Generator Architectures	138
7.1.1	Single-Stage Generator	138
7.1.2	Two-Stage Generator	138

7.1.3	Model-View-Controller Architecture	140
7.2	Metrics	141
7.3	ApiGen	143
7.3.1	Introduction	143
7.3.2	Annotated Data Type	144
7.3.3	From ADT to an API	145
7.3.4	Original Code Generator	150
7.3.5	Reimplemented Code Generator	153
7.3.6	Difference Old and New Implementation	157
7.3.7	Evaluation	159
7.4	NunniFSMGen	160
7.4.1	Finite State Machines	160
7.4.2	NunniFSMGen Input Model	161
7.4.3	State Machine Implementation	165
7.4.4	Original Code Generator	167
7.4.5	Reimplemented Code Generator	169
7.4.6	Difference Old and New Implementation	175
7.4.7	Evaluation	176
7.5	Dynamic XHTML generation	177
7.5.1	Cross-site Scripting	177
7.5.2	Preventing Cross-site Scripting	180
7.5.3	Example Web Application: Shout Wall	181
7.5.4	Unhackability	186
7.5.5	Preventing Injections at the Door	188
7.5.6	Evaluation	188
7.6	Conclusions	189
8.	Conclusions	191
8.1	Unparser Completeness	192
8.2	A Template Metalanguage	192
8.3	Syntactical Correctness of Templates	193
8.4	Syntax-Safe Evaluation	193
8.5	Case Studies	194
	Bibliography	197
	Index	203
		205

Code Generation with Templates

Arnoldus, J.; van den Brand, M.; Serebrenik, A.;

Brunekreef, J.J.

2012, IX, 204 p. 100 illus., 2 illus. in color., Hardcover

ISBN: 978-94-91216-55-8

A product of Atlantis Press