

Contents

1 Introduction

1.1 Graph Rewriting Everywhere? — The Purpose of Graph Rewriting Systems	1
1.2 The Major Flaw of Graph Rewriting - Its Complexity	4
1.3 The Plan of Attack - Finding the Gap	5
1.4 Outline	7

2 Graph Rewriting Systems — The Basic Concepts

2.1 Vertices, Edges, and Labels make a Graph — Preliminary Definitions	10
2.2 Algorithmic Graph Rewriting Systems — The Basic Formalism	13
2.3 An Abstract Machine for Labelled Subgraph Matching	24
2.4 Summary and Related Work	31

3 UBS-Graph Rewriting Systems — Matching Subgraphs In Constant Time

3.1 Unique Labels — Singularities in Derived Graphs	38
3.1.1 Unique Vertex Labels	38
3.1.2 Unique Edge Labels	43
3.2 Label Triples — Detecting Dead Rules and Saving Memory	47
3.2.1 Definition and Properties of Label Triples	47
3.2.2 Approximation of the Set of Label Triples	49
3.3 Strong V-Structures — The Branching Points	56
3.3.1 Strong V-Structures and Matching in Constant Time	58
3.3.2 Determination of a Bypassing Connected Enumeration	63
3.3.3 Conditions for the Introduction of Strong V-Structures	65
3.3.4 Approximation of the Set of Strong V-Structures	81
3.4 Summary and Related Work	85

4 Programmed Attributed Graph Rewrite Systems — An Advanced Modelling Formalism

4.1	Attributed Graph Rewriting Systems — Extension I	92
4.1.1	The Formalism	93
4.1.2	Application of the Analyses	99
4.2	Programmed Graph Rewriting Systems — Extension II	101
4.2.1	The Syntax	105
4.2.2	The Failure Semantics	106
4.2.3	The Collection Semantics	110
4.2.4	Analysis of Further Control Structures	114
4.3	Summary and Related Works	117

5 The Abstract Machine for Graph Rewriting — Supporting a Fast Implementation

5.1	Optimization of the Application Test for Rule Sets	125
5.2	The Graph Rewriting Environment	136
5.2.1	The State of the Core Abstract Machine	138
5.2.2	Matching Instructions	141
5.2.3	Structural Graph Updates	145
5.2.4	Evaluation of Embedding Descriptions	146
5.2.5	Attribute Evaluation	147
5.2.6	Control Instructions	149
5.3	Code Generation for the Abstract Machine	149
5.3.1	The Syntax	150
5.3.2	The Code Generation Rules	152
5.4	Improvements by Rule Set Optimization	157
5.5	Summary and Related Work	160

6 A Graphical Implementation of Functional Languages — A Case Study in UBS-Graph Rewriting Systems

6.1	Aspects of Functional Programming	165
6.2	Translating Functions to Trees — Graph Reduction approaches Graph Rewriting	168
6.2.1	A Graphical Notation for Functional Expressions	168
6.2.2	Function Definitions as Rewriting Rules	170
6.2.3	Translating Expressions	171
6.2.4	Translating Function Definitions	176

6.3	Sharing — Graph Reduction Meets Graph Rewriting	181
6.4	Normal Order Reduction — Enforcing a Deterministic Evaluation Order	185
6.4.1	A Stack of Spine Pointers	187
6.4.2	Deconstruction of the Spine Stack	191
6.4.3	Unwinding the Spine	192
6.5	The Printing Mechanism	197
6.6	The Translation of Lazy Evaluation is UBS	201
6.6.1	A Mandatory Set of Prohibited Strong V-Structures	202
6.6.2	None of the Prohibited Strong V-Structures Occur	206
6.7	Summary and Related Work	213
7	Conclusions	217
Appendix A	List of Figures and Tables	223
Appendix B	Implementation of a Functional Program	225
Appendix C	References	255
Appendix D	Index	261