Contents

I Introduction 1

1 Context and Motivation 3

2 Targeted Services 9

3 Design Goals 13

4 Solution Concept 17

5 Structure of this Thesis 21

II Foundations 23

6 The Clojure Programming Language 27
- 6.1 Evaluation Cycle . . . 28
- 6.2 Syntax . . . 29
- 6.3 Basic Types . . . 29
- 6.4 Collection Types . . . 30
- 6.5 Special Forms . . . 33
- 6.6 Abstractions . . . 39
- 6.7 Destructuring . . . 48
- 6.8 Higher-Order Functions . . . 50
- 6.9 Metadata . . . 53
- 6.10 Namespaces . . . 55
- 6.11 Java Interop . . . 57
- 6.12 Macros . . . 60
- 6.13 Utility macros . . . 64
- 6.14 Protocols . . . 66

7 Modeling Frameworks 69
- 7.1 JGraLab . . . 69
- 7.2 The Eclipse Modeling Framework . . . 73

III The FunnyQT Approach 79

8 On Genericity 83
- 8.1 Viable Options for Genericity . . . 83

8.2 Protocol-Based Genericity for Model Querying and Transformation . 88

9 Terminology and Conventions 91

10 The FunnyQT Architecture 95

11 Related Work 99

IV On Model Management and Querying 103

12 Generic Model Management and Querying 107
12.1 Creating and Deleting Elements and Relationships 107
12.2 Iterating Elements and Relationships 109
12.3 Attribute Value Access 111
12.4 Role Name Navigation 112
12.5 Type Checks 113
12.6 Containers and Contents 114
12.7 Neighboring Elements 115
12.8 Model Equality 115
12.9 Copying Models 116
12.10 Metamodel Access 116
12.11 Generating a Metamodel-Specific API .120

13 Managing and Querying TGraphs 123
13.1 Loading and Saving Graphs and Schemas 123
13.2 Creating Graphs, Vertices, and Edges .124
13.3 Accessing Graph Elements by ID 125
13.4 Accessing Vertices and Edges by their Order 125
13.5 Querying and Manipulating Element Order 126
13.6 Lazy Vertex and Edge Sequences 127
13.7 Attribute Value Access 128
13.8 Auxiliary Graph Functions 129
13.9 Auxiliary Vertex Functions 129
13.10 Auxiliary Edge Functions 130
13.11 Type Predicates 131
13.12 Schema Access 131
13.13 Generating a Schema-Specific API 132
13.14 Traversal Contexts 133

14 Managing and Querying EMF Models 137
14.1 Managing Resources 137
14.2 Creating and Deleting EObjects 138
14.3 Lazy Content Sequences 139
14.4 EObject Structural Feature Access 140
14.5 Access to Referenced Objects and Conceptual Edges 141
14.6 Type Predicates 143
14.7 Metamodel Access 143
14.8 Generating an Ecore-Model-Specific API 144

15 Generic Querying and Model Management Features 147
15.1 Regular Path Expressions 147
15.2 Polymorphic Functions 154
15.3 Model Visualization 158
15.4 XML Processing 159
15.5 Persistence of Model-Related Data 162
15.6 Miscellaneous 168

16 Related Work 173

V On Pattern Matching 179

17 Introduction 183
17.1 Pattern Matching in Functional Languages 184
17.2 Pattern Matching on Graphs 188

18 Defining Patterns 193
18.1 Pattern Syntax and Semantics 195
18.2 Framework-Specific Patterns 212

19 Related Work 217

VI On In-Place Transformations 219

20 Introduction 223

21 Defining In-Place Rules 227

22 Defining Control Flow and Modifying Behavior 231
22.1 Higher-Order Rule Combinators 231
22.2 Rule Application Modifiers 235

23 Exploring State Space 237

24 Related Work 249

VII On Out-Place Transformations 257

25 Introduction 261

26 Defining Rule-Based Transformations 265
26.1 Transformation Definitions 265
26.2 Example 276

27 Defining Extensional Transformations 281
27.1 Metamodel Constituents and Their Extensions 282
27.2 Indirect Specification of Extensions 284
27.3 Traceability 285

27.4 Extensional Transformation Constructs 287
27.5 Example 294

28 Related Work **301**

VIII On Relational Model Querying **309**

29 Introduction **313**
29.1 Relational Programming with core.logic 314

30 Defining Relational Model Queries **323**
30.1 Generic Relations 323
30.2 Metamodel-Specific Relations 328
30.3 Utilities 329

31 Example **331**

32 Related Work **335**

IX On Bidirectional Transformations **337**

33 Introduction **341**

34 Defining Bidirectional Transformations **345**
34.1 Transformation Definitions 346
34.2 Transformation Relations 347
34.3 Traceability 347
34.4 Transformation Direction and Modes 349
34.5 Preconditions 353
34.6 Target Clauses 354
34.7 Postconditions 355
34.8 Transformation Relation Inheritance 356
34.9 Transformation Inheritance 357
34.10 Plain Relations 359

35 Characteristics **361**

36 Example **363**

37 Related Work **369**

X On Co-Evolution Transformations **373**

38 Introduction **377**

39 Defining Co-Evolution Transformations **379**
39.1 Conformance of Schemas and Graphs 381
39.2 Co-Evolution Operations 383

40 Example **393**

41 Related Work **397**

XI Finale **401**

42 Evaluation **405**
42.1 The TTC 2013 Flowgraphs Case 406
42.2 The TTC 2013 Class Diagram Restructuring Case 407
42.3 The TTC 2013 Petri-Nets to Statechart Case 407
42.4 The TTC 2014 Movie Database Case 408
42.5 The TTC 2014 FIXML to Java, C#, and C++ Case 409
42.6 The TTC 2015 Model Execution Case .410
42.7 The TTC 2015 Java Refactoring Case .411
42.8 The TTC 2015 Train Benchmark Case .412
42.9 TTC Summary 413

43 Conclusion **415**
43.1 Summary and Contributions 415
43.2 Outlook 418

XII Appendix **421**

A Extensibility **423**
A.1 Extending Regular Path Expressions 423
A.2 Extending Polymorphic Functions 423
A.3 Extending the Pattern Matching Facilities 424
A.4 Extending the Rule-Based Out-Place Transformation DSL 426
A.5 Extending the Extensional Out-Place Transformation API 426
A.6 Extending the Relational Querying API 427
A.7 Extending the Bidirectional Transformation DSL 427

Bibliography **429**

Index **445**

Curriculum Vitae **455**

Part I

Introduction