Table of Contents

1. **Introduction** 1

Part I. Planning

2. **State-Based Planning** 13
   - 2.1 Standard Strips 13
     - 2.1.1 A Blocks-World Example 14
     - 2.1.2 Basic Definitions 14
     - 2.1.3 Backward Operator Application 18
   - 2.2 Extensions and Alternatives to Strips 20
     - 2.2.1 The Planning Domain Definition Language 20
     - 2.2.2 Situation Calculus 24
   - 2.3 Basic Planning Algorithms 27
     - 2.3.1 Informal Introduction of Basic Concepts 28
     - 2.3.2 Forward Planning 29
     - 2.3.3 Formal Properties of Planning 32
     - 2.3.4 Backward Planning 35
   - 2.4 Planning Systems 40
     - 2.4.1 Classical Approaches 40
     - 2.4.2 Current Approaches 42
     - 2.4.3 Complex Domains and Uncertain Environments 44
     - 2.4.4 Universal Planning 45
     - 2.4.5 Planning and Related Fields 48
     - 2.4.6 Planning Literature 50
   - 2.5 Automatic Knowledge Acquisition for Planning 51
     - 2.5.1 Pre-planning Analysis 51
     - 2.5.2 Planning and Learning 51

3. **Constructing Complete Sets of Optimal Plans** 55
   - 3.1 Introduction to DPlan 55
     - 3.1.1 DPlan Planning Language 56
     - 3.1.2 DPlan Algorithm 57

3.1.3 Efficiency Concerns ... 58
3.1.4 Example Problems ... 59
3.2 Optimal Full Universal Plans ... 64
3.3 Termination, Soundness, Completeness ... 66
3.3.1 Termination of DPlan ... 66
3.3.2 Operator Restrictions ... 67
3.3.3 Soundness and Completeness of DPlan ... 70

4. Integrating Function Application in Planning ... 71
4.1 Motivation ... 71
4.2 Extending Strips to Function Applications ... 74
4.3 Extensions of FPlan ... 79
4.3.1 Backward Operator Application ... 79
4.3.2 Introducing User-Defined Functions ... 81
4.4 Examples ... 82
4.4.1 Planning with Resource Variables ... 83
4.4.2 Planning for Numerical Problems ... 85
4.4.3 Functional Planning for Standard Problems ... 87
4.4.4 Mixing ADD/DEL Effects and Updates ... 88
4.4.5 Planning for Programming Problems ... 88
4.4.6 Constraint Satisfaction and Planning ... 90

5. Conclusions and Further Research ... 93
5.1 Comparing DPlan with the State of the Art ... 93
5.2 Extensions of DPlan ... 94
5.3 Universal Planning versus Incremental Exploration ... 95

Part II. Inductive Program Synthesis

---

6. Automatic Programming ... 99
6.1 Overview of Automatic Programming Research ... 100
6.1.1 AI and Software Engineering ... 100
6.1.2 Approaches to Program Synthesis ... 102
6.1.3 Pointers to Literature ... 109
6.2 Deductive Approaches ... 110
6.2.1 Constructive Theorem Proving ... 110
6.2.2 Program Transformation ... 115
6.3 Inductive Approaches ... 124
6.3.1 Foundations of Induction ... 124
6.3.2 Genetic Programming ... 134
6.3.3 Inductive Logic Programming ... 140
6.3.4 Inductive Functional Programming ... 150

6.4 Final Comments 164
    6.4.1 Inductive versus Deductive Synthesis 164
    6.4.2 Inductive Functional versus Logic Programming 165

7. Folding of Finite Program Terms 167
7.1 Terminology and Basic Concepts 168
    7.1.1 Terms and Term Rewriting 168
    7.1.2 Patterns and Anti-unification 171
    7.1.3 Recursive Program Schemes 172
7.2 Synthesis of RPSs from Initial Programs 182
    7.2.1 Folding and Fixpoint Semantics 182
    7.2.2 Characteristics of RPSs 182
    7.2.3 The Synthesis Problem 185
7.3 Solving the Synthesis Problem 185
    7.3.1 Constructing Segmentations 186
    7.3.2 Constructing a Program Body 195
    7.3.3 Dealing with Further Subprograms 198
    7.3.4 Finding Parameter Substitutions 206
    7.3.5 Constructing an RPS 215
7.4 Example Problems 220
    7.4.1 Time Effort of Folding 220
    7.4.2 Recursive Control Rules 222

8. Transforming Plans into Finite Programs 227
8.1 Overview of Plan Transformation 228
    8.1.1 Universal Plans 228
    8.1.2 Introducing Data Types and Situation Variables 228
    8.1.3 Components of Plan Transformation 229
    8.1.4 Plans as Programs 229
    8.1.5 Completeness and Correctness 231
8.2 Transformation and Type Inference 231
    8.2.1 Plan Decomposition 231
    8.2.2 Data Type Inference 233
    8.2.3 Introducing Situation Variables 234
8.3 Plans over Sequences of Objects 235
8.4 Plans over Sets of Objects 240
8.5 Plans over Lists of Objects 246
    8.5.1 Structural and Semantic List Problems 246
    8.5.2 Synthesizing 'Selection-Sort' 248
    8.5.3 Concluding Remarks on List Problems 257
8.6 Plans over Complex Data Types 259
    8.6.1 Variants of Complex Finite Programs 259
    8.6.2 The 'Tower' Domain 260
    8.6.3 Tower of Hanoi 267

9. Conclusions and Further Research 271
- 9.1 Combining Planning and Program Synthesis 271
- 9.2 Acquisition of Problem Solving Strategies 272
  - 9.2.1 Learning in Problem Solving and Planning 272
  - 9.2.2 Three Levels of Learning 273

Part III. Schema Abstraction

10. Analogical Reasoning and Generalization 279
- 10.1 Analogical and Case-Based Reasoning 279
  - 10.1.1 Characteristics of Analogy 279
  - 10.1.2 Sub-processes of Analogical Reasoning 281
  - 10.1.3 Transformational versus Derivational Analogy 282
  - 10.1.4 Quantitive and Qualitative Similarity 283
- 10.2 Mapping Simple Relations or Complex Structures 284
  - 10.2.1 Proportional Analogies 284
  - 10.2.2 Causal Analogies 286
  - 10.2.3 Problem Solving and Planning by Analogy 286
- 10.3 Programming by Analogy 288
- 10.4 Pointers to Literature 290

11. Structural Similarity in Analogical Transfer 291
- 11.1 Analogical Problem Solving 291
  - 11.1.1 Mapping and Transfer 292
  - 11.1.2 Transfer of Non-isomorphic Source Problems 293
  - 11.1.3 Structural Representation of Problems 294
  - 11.1.4 Non-isomorphic Variants in a Water Redistribution Domain 296
  - 11.1.5 Measurement of Structural Overlap 300
- 11.2 Experiment 1 300
  - 11.2.1 Method 302
  - 11.2.2 Results and Discussion 303
- 11.3 Experiment 2 305
  - 11.3.1 Method 306
  - 11.3.2 Results and Discussion 307
- 11.4 General Discussion 309

12. Programming by Analogy 311
- 12.1 Program Reuse and Program Schemes 311
- 12.2 Restricted 2nd–order Anti–unification 312
  - 12.2.1 Recursive Program Schemes Revisited 312
  - 12.2.2 Anti-unification of Program Terms 314

12.3 Retrieval Using Term Subsumption 316
12.3.1 Term Subsumption 316
12.3.2 Empirical Evaluation 317
12.3.3 Retrieval from Hierarchical Memory 318
12.4 Generalizing Program Schemes 319
12.5 Adaptation of Program Schemes 320

13. Conclusions and Further Research 323
13.1 Learning and Applying Abstract Schemes 323
13.2 A Framework for Learning from Problem Solving 324
13.3 Application Perspective 325

Bibliography 327

Appendices

A. Implementation Details 343
A.1 Short History of DPlan 343
A.2 Modules of DPlan 345
A.3 DPlan Specifications 345
A.4 Development of Folding Algorithms 347
A.5 Modules of TFold 348
A.6 Time Effort of Folding 349
A.7 Main Components of Plan-Transformation 349
A.8 Plan Decomposition 350
A.9 Introduction of Situation Variables 351
A.10 Number of MSTs in a DAG 351
A.11 Extracting Minimal Spanning Trees from a DAG 352
A.12 Regularizing a Tree 353
A.13 Programming by Analogy Algorithms 355

B. Concepts and Proofs 357
B.1 Fixpoint Semantics 357
B.2 Proof: Maximal Subprogram Body 360
B.3 Proof: Uniqueness of Substitutions 365

C. Sample Programs and Problems 369
C.1 Fibonacci with Sequence Referencing Function 369
C.2 Inducing ‘Reverse’ with Golem 370
C.3 Finite Program for ‘Unstack’ 373
C.4 Recursive Control Rules for the ‘Rocket’ Domain 375
C.5 The ‘Selection Sort’ Domain 376
C.6 Recursive Control Rules for the ‘Tower’ Domain 377

C.7 Water Jug Problems 383
C.8 Example RPSs 388

Index 391

http://www.springer.com/978-3-540-40174-2

Inductive Synthesis of Functional Programs
Universal Planning, Folding of Finite Programs, and
Schema Abstraction by Analogical Reasoning
Schmid, U.
2003, XXII, 402 p., Softcover
ISBN: 978-3-540-40174-2