

Table of Contents

Introduction	1
Section I	
What Are Problem-Solving Methods	
1 Making Assumptions for Efficiency Reasons	7
1.1 A Definition of a Task	10
1.2 A Non-efficient Problem Solver	11
1.3 An Efficient Problem Solver	13
1.4 Summary of the Case Study	17
1.5 The Twofold Role of Assumptions	19
1.6 How Deal Other Approaches with Assumptions and Efficiency	22
2 An Empirical Survey of Assumptions	26
2.1 Assumptions Necessary to Define the Task	27
2.1.1 Identifying Abnormalities	27
2.1.2 Identifying Causes	27
2.1.3 Defining Hypotheses	29
2.1.4 Defining Diagnoses	29
2.1.5 Summary	30
2.2 Assumptions Necessary to Define an Efficient Problem Solver	33
2.2.1 Reducing the Worst-Case Complexity	34
2.2.2 Reducing the Average-Case Behavior	35
2.2.3 Search Guidance	36
2.2.4 Summary	37
2.3 Assumptions in System-Environment Interaction	38
2.4 Summary	39
Section II	
How to Describe Problem-Solving Methods	
3 A Four Component Architecture for Knowledge-Based Systems	43
3.1 The Entire Framework	44
3.1.1 The Main Elements of a Specification	44
3.1.2 The Main Proof Obligations	48
3.2 Task	49
3.3 Problem-Solving Method	50
3.3.1 The Black Box Description: Competence and Requirements	51
3.3.2 The Operational Specification	51
3.4 Domain Model	54
3.5 Adapters	56
3.5.1 Connecting Task and Problem-Solving Method	57
3.5.2 Connecting with the Domain Model	58
3.6 Related Work	58
4 Logics for Knowledge-Based Systems: MLPM and MCL	61
4.1 Specification Languages for Knowledge-Based Systems	62
4.1.1 $(ML)^2$	62

4.1.2	KARL	66
4.1.3	Design Rationales for a Logic of Dynamics	67
4.2	Logics for the Dynamics of Knowledge-Based Systems	71
4.2.1	Modal Logic of Predicate Modification (MLPM)	71
4.2.2	Modal Change Logic (MCL)	73
4.2.3	Modeling MLPM with MCL	73
4.3	Formalizing Other Approaches	74
4.3.1	Formalizing KADS Languages	74
4.3.2	Using MCL to Formalize Abstract State Machines	75
4.3.3	Approaches Using Different Paradigms	76
5	A Verification Framework for Knowledge-Based Systems	78
5.1	The Architecture in KIV	79
5.2	Formalizing a Task	79
5.3	Formalizing a Problem-Solving Method	81
5.4	Proving Total Correctness of the Problem-Solving Method	84
5.5	Adapter: Connecting Task and Problem-Solving Method	87
5.6	A Specific Pattern in Specifying Architectures of Knowledge-Based Systems	88
5.7	Future Work	90

Section III

How to Develop and Reuse Problem-Solving Methods

6	Methods for Context Explication and Adaptation	95
6.1	Inverse Verification of Problem-Solving Methods	98
6.1.1	First Example: A Local Search Method	100
6.1.2	Second Example: Finding an Abductive Explanation	103
6.1.3	Heuristic Assumptions	105
6.1.4	Related Work	108
6.2	Stepwise Adaptation of Problem-Solving Methods	109
6.2.1	Local Search	110
6.2.2	Hill-Climbing	110
6.2.3	Set-Minimizer	111
6.2.4	Abductive Diagnosis	113
6.2.5	Generalization and Limitation of Refinement with Adapters	114
7	Organizing a Library of Problem-Solving Methods	116
7.1	The Three Dimensions in Method Organization	117
7.2	Deriving Task-Specific Problem-Solving Methods	119
7.2.1	Problem Type Design	121
7.2.2	Local Search	121
7.2.3	Local Search as Design Problem Solving	122
7.2.4	Problem Type Parametric Design	122
7.2.5	Local Search as Parametric Design Problem Solving	123
7.3	Varying the Problem-Solving Paradigm	123
7.3.1	Derive Successor Candidates	124
7.3.2	Select the Design Model that Is to Be Expanded Next	126
7.3.3	Update the Set of Future Candidates	126
7.4	Conclusions	127

Conclusions and Future Work

129

References

133