

Learning Effective Search Control Knowledge: An Explanation-Based Approach

Steven Minton

March 1988

CMU-CS-88-133

**Department of Computer Science
Carnegie-Mellon University
Pittsburgh, PA. 15213**

**Submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science at Carnegie-Mellon University.**

This research was supported in part by an AT&T Bell Laboratories Ph.D. Scholarship, in part by the Office of Naval Research under Contract N00014-84-K-0415 and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under contract F33615-87-C-1499, monitored by the Avionics Laboratory, Air Force Wright Aeronautics Laboratories, Aeronautical Systems Division (AFSC), Wright Patterson AFB, OHIO 45433-6543. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of Bell Laboratories, DARPA, the Air Force Office of Scientific Research or the US government.



Table of Contents

1. Introduction	5
1.1. Overview	5
1.2. What is Explanation-based Learning?	6
1.2.1. The Terminology of Explanation-based Learning	6
1.3. EBL in the PRODIGY System	9
1.4. Perspectives on the Learning Process	10
1.5. Scientific Contributions	11
1.6. A Reader's Guide to the thesis	12
2. Analyzing the Utility Problem	15
2.1. Using EBL to Improve Performance	15
2.2. A Simple Model of EBL	16
2.3. Removing Inefficiencies	20
2.4. Extending the Simple Model	23
2.5. Will the Utility Problem Go Away?	26
3. Overview of the PRODIGY Problem Solver	29
3.1. System Overview	29
3.2. Using the Problem Solver	29
3.3. An Example Problem	32
3.4. Control Rules	34
3.5. PRODIGY's Description Language (PDL)	36
3.5.1. Syntax	36
3.6. Semantics	37
3.6.1. The Role of Generators in PDL	37
3.6.2. Extensions to PDL	39
3.7. The Problem Solver's Control Structure in Detail	40
3.7.1. Choosing a Node	40
3.7.2. Choosing a Goal	41
3.7.3. Choosing an Operator	41
3.7.4. Choosing a Set of Bindings for the Operator	41
3.7.5. Creating a New Node	43
3.7.6. Summary	44
3.8. Domains	45
3.9. Advanced Features	46
3.9.1. Complex Descriptions	46
3.9.2. Negated Goals and Goals with Variables	46
3.9.3. Reason Maintenance	47
3.9.4. Negated Effects in Inference Rules	47
3.9.5. Meta Predicates	47
3.9.6. Conditional Effects	48
3.9.7. Interleaving Goals	49
3.9.8. User Interface	49

4. Specialization	51
4.1. Target Concepts	52
4.2. Scanning the Search Tree for Examples	53
4.3. Constructing an Explanation: The EBS Method	54
4.3.1. EBS and Bias	57
4.4. EBS: An Example	57
4.5. Utility Issues	59
4.6. Summary	60
5. Compression	61
5.1. How Compression is Used	61
5.2. Implementing Compression	63
5.2.1. Partial Evaluation	63
5.2.2. Taking Advantage of Logical Equivalences	67
5.2.3. Conjunct Reordering	67
5.2.4. Domain-Specific Transformations	68
5.3. The Cost of Compression	70
5.4. The Benefits of Compression	71
5.5. Summary	74
6. Utility Evaluation	75
6.1. Defining Utility	75
6.2. Measuring Utility	76
6.2.1. Estimating Utility	77
6.2.2. Empirical Utility Validation	78
6.3. Interactions between Rules	79
6.4. The Costs and Benefits of Utility Analysis	79
6.5. Summary	79
7. Learning from Success	81
7.1. Methods for Learning from Success	81
7.2. How PRODIGY Learns from Success	82
7.3. Heuristics for Selecting Examples of Success	83
7.4. Examples of Learning from Success	84
7.4.1. Scheduling Domain Example	84
7.4.2. GridWorld Example	87
7.5. Problems	90
7.6. Factors Influencing Utility	91
8. Learning from Failure	95
8.1. How PRODIGY learns from Failure	95
8.2. Heuristics for Selecting Examples	97
8.3. Examples of Learning from Failure	98
8.3.1. An In-depth Look at Explaining a Failure	100
8.3.2. The Utility of Learning From Failure: An Example	102
8.4. Factors Influencing Utility	105
9. Learning from Goal Interactions	107
9.1. How PRODIGY learns from Goal Interference	107
9.2. Heuristics for Selecting Examples	110
9.3. A Blocksworld Example	110
9.4. A Scheduling Example	112
9.5. Factors Influencing Utility	114
10. Performance Results	115

10.1. Randomly Generating Problems	115
10.2. Standards for Comparisons	116
10.3. Methodology	117
10.4. Performance in the Blocksworld Domain	118
10.5. Performance in the STRIPS Domain	123
10.6. Performance in the Scheduling Domain	126
10.7. Evaluating the Components of the Learning System	129
10.7.1. Target Concepts	129
10.7.2. Compression Analysis	130
10.7.3. Utility Evaluation	132
10.8. Comparison with Macro-Operator Learning	132
10.9. Discussion of Performance Results	137
11. Proofs, Explanations, and Correctness: Putting It All Together	139
11.1. The Missing Level of Description	139
11.2. Explanations, Proofs and Weakest Preconditions	140
11.2.1. Proofs as Explanations	141
11.2.2. Proof Systems	141
11.2.3. The EBL Process	142
11.2.4. Correctness of EBS	143
11.3. Generality	146
11.3.1. Transforming Explanations to Improve the Utility of EBL	151
11.4. Conclusion	152
12. Related Work	153
12.1. Explanation-Based approaches	153
12.1.1. The EBG Approach	154
12.1.2. Other EBG Related Research	157
12.1.3. Constraint-Based Generalization	158
12.1.4. DeJong's Paradigm: EBL for Schema Acquisition	160
12.1.5. STRIPS and MORRIS	162
12.2. Other Knowledge Intensive Learning Methods	164
12.2.1. Chunking in SOAR	164
12.2.2. Analogical and Case-based Reasoning	165
12.2.3. Other Macro-operator Systems	166
12.3. Summary	167
13. Conclusion	169
13.1. Summary of Central Principles and Contributions	170
13.2. Where is the Knowledge?	171
13.3. Looking Ahead	172
Appendix A. Domain Specifications	173
A.1. Scheduling World	173
A.1.1. Procedure for Generating Scheduling Problems	173
A.1.2. Domain Specification	174
A.2. 3D Gridworld	178
A.2.1. Domain Specification	178
A.3. Blocksworld	180
A.3.1. Procedure for Generating Problems	180
A.3.2. Domain Specification	180
A.4. Extended STRIPS Robot Domain	181
A.4.1. Procedure for Generating Problems	182
A.4.2. Domain Specification	183
A.4.3. Initial Control Rules for STRIPS Domain	186

Appendix B. Details of the Correctness Proof for EBS-WP	189
B.1. Terminology	189
B.2. Definitions	189
B.3. Description of Procedure EBS-WP	191
B.4. Lemmas	191
B.5. Correctness Proof for EBS-WP	193
B.5.1. PART ONE: Derivability	193
B.5.1.1. Derivability: Case One	194
B.5.1.2. Derivability: Case Two	194
B.5.1.3. Derivability: Case Three	195
B.5.2. PART TWO: Generality	195
B.5.2.1. Generality: Case One	196
B.5.2.2. Generality: Case Two	196
B.5.2.3. Generality: Case Three	197
Appendix C. Performance Data	199
C.1. Training Phase Data	199
C.2. Test Phase Data	203
C.3. Macro-Operator Comparison Data	208
Appendix D. Architectural-Level Proof Schemas	211
D.1. Architecture-Level Schemas for SUCCEEDS	211
D.2. Architecture-Level Axioms relevant to FAILS	212
D.3. Architecture-Level Schemas for SOLE-ALTERNATIVE	216
D.4. Architecture-Level Schemas for GOAL-INTERFERENCE	216
D.5. Architecture-Level Axioms for Computing Regressions	217

Abstract

In order to solve problems more effectively with accumulating experience, a problem solver must be able to learn and exploit search control knowledge. Although previous research has demonstrated that Explanation-Based Learning (EBL) is a viable approach for acquiring control knowledge, in practice the learned control knowledge may not be useful. For control knowledge to be effective, the cumulative benefits of applying the knowledge must outweigh the cumulative costs of testing whether the knowledge is applicable. Previous research in EBL has ignored this issue, which I refer to as the *utility* problem. Most researchers have simply demonstrated that EBL can improve performance on particular examples without analyzing exactly when performance improvement will occur. In practice, it is much more difficult improve performance over a population of examples than it is to improve performance on isolated examples.

One answer to the utility problem is to search for "good" explanations -- explanations that can be profitably employed to control problem solving. Instead of simply adding control knowledge haphazardly, a learning system must be sensitive to the problem solver's computational architecture and the potential costs and benefits of adding knowledge. This thesis analyzes the utility of EBL, and describes a method for searching for good explanations. The method, implemented in the PRODIGY/EBL system, consists of a three-stage heuristic search. Given a problem solving trace, PRODIGY first selects what to learn. The system chooses from a variety of target concepts, each representing a different strategy for optimizing performance. Secondly, after creating an initial explanation from the trace, PRODIGY searches for a representation of the explanation that is efficient to match. Finally the system empirically tests the effectiveness of the learned control knowledge to determine whether it is actually worth keeping.

The thesis includes a set of comprehensive experiments testing the performance of the PRODIGY/EBL system and its components in several domains. In addition, a formal description of EBL is presented, together with a correctness proof for PRODIGY's generalization method.

