A Computational Model of Lakatos-style Reasoning

Alison Pease

Doctor of Philosophy

School of Informatics
University of Edinburgh

2007

Abstract

Lakatos outlined a theory of mathematical discovery and justification, which suggests ways in which concepts, conjectures and proofs gradually evolve via interaction between mathematicians. Different mathematicians may have different interpretations of a conjecture, examples or counterexamples of it, and beliefs regarding its value or theoremhood. Through discussion, concepts are refined and conjectures and proofs modified. We hypothesise that *(i)* it is possible to computationally represent Lakatos's theory, and *(ii)* it is useful to do so. In order to test our hypotheses we have developed a computational model of his theory.

Our model is a multiagent dialogue system. Each agent has a copy of a pre-existing theory formation system, which can form concepts and make conjectures which empirically hold for the objects of interest supplied. Distributing the objects of interest between agents means that they form different theories, which they communicate to each other. Agents then find counterexamples and use methods identified by Lakatos to suggest modifications to conjectures, concept definitions and proofs.

Our main aim is to provide a computational reading of Lakatos's theory, by interpreting it as a series of algorithms and implementing these algorithms as a computer program.

This is the first systematic automated realisation of Lakatos's theory. We contribute to the computational philosophy of science by interpreting, clarifying and extending his theory. We also contribute by evaluating his theory, using our model to test hypotheses about it, and evaluating our extended computational theory on the basis of criteria proposed by several theorists. A further contribution is to automated theory formation and automated theorem proving. The process of refining conjectures, proofs and concept definitions requires a flexibility which is inherently useful in fields which handle ill-specified problems, such as theory formation. Similarly, the ability to automatically modify an open conjecture into one which can be proved, is a valuable contribution to automated theorem proving.

Acknowledgements

Thank you to my supervisors Alan Smaill, John Lee and Simon Colton, who have constantly inspired and helped me. In particular, thanks to Alan, who is the cleverest and kindest man that I know, to John for all his patience and encouragement, and to Simon for his endless enthusiasm, energy and ideas. I've really enjoyed working with you.

I am hugely indebted to Simon for his HR program. This has formed the backbone of the project and Simon has untiringly answered questions and supported me in learning about the program. I think it is unusual to get this level of support, and Simon has been fantastic.

I would also like to thank my examiners Aaron Sloman and Paul Schweizer, who made my viva an enjoyable, stimulating and challenging afternoon. They have made many valuable contributions to the thesis and have been particularly inspiring as to future directions that this project may now take.

Thanks also to all of the DReaMers, especially Fiona McNeill, Laura Meikle, Alex Heneveld, Lucas Dixon, Graham Steel, Ewen Maclean and Manuel Contreras Maya. Thanks to all the boys – Colin Fraser, Seb Mhatre, Dan Winterstein, Joe Halliwell and Paul Crook. Paul deserves special thanks; we have worked together from the very first to the very last day, and this has made these last few years really enjoyable. Thanks to everyone in my family for their encouragement. Lastly, I'd like to thank Marcus Pearce for all his love and support.

This work has been supported by EPSRC grant GR/M45030, for which I am very grateful.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Alison Pease*)

Publications

Parts of chapter 2 and 11 have appeared in the Journal of Computing and Philosophy (ECAP) [Pease et al., 2004].

Parts of chapter 6 have appeared in the Proceedings of Edilog 2002, the 6th Workshop on the Semantics and Pragmatics of Dialogue, Pease et al. [2002].

Parts of chapter 12 have appeared in the Proceedings of the Automated Reasoning Workshop, 2004 [Pease and Colton, 2004], and Workshop on Disproving, Proceedings of IJCAR'04, [Colton and Pease, 2004].

Table of Contents

1 Introduction **1**

1.1 Motivation 3

1.2 Aims of project 5

1.3 Contributions 6

1.4 Organisation of thesis 6

1.5 Summary 7

2 Computational Philosophy of Science **9**

2.1 Computational Philosophy of Science 9

2.2 Lakatos's logic of discovery and justification in mathematics 11

2.3 Lakatos's three main methods 13

2.3.1 The method of monster-barring 13

2.3.2 The method of exception-barring 13

2.3.3 The method of proofs and refutations 14

2.4 Reaction to Lakatos's theory 15

2.5 A computational reading of Lakatos's theory 16

2.6 Possible objections to a model of Lakatos's ideas 16

2.7 Conclusion 17

3 Background **19**

3.1 Automated theory formation and HR 19

3.1.1 Representation of objects of interest and concepts in HR 20

3.1.2 Generation of concepts and conjectures in HR 24

3.1.3 Evaluation of concepts and conjectures in HR 29

3.1.4 Representing a theory in HR 32

3.1.5 Using HR in a multiagent system 33

3.2 Multiagent Systems 33

3.2.1 Environment 34

3.2.2 Agent design 34

3.2.3 Society design 36

3.2.4 Argumentation-based negotiation 37

3.3 Methodology 38

3.3.1 Evaluating theories within the philosophy of science 39

3.3.2 Thagard's criteria for evaluating theories 39

3.3.3 Sloman's criteria for evaluating theories 40

3.3.4 Popper's criteria for evaluating theories 41

3.3.5 Lakatos's criteria for evaluating theories 42

3.3.6 Summary of criteria 43

3.4 Summary 44

4 Overview of HRL **45**

4.1 Environment 45

4.2 Agent design 46

4.3 Society design 47

4.4 Extensions to HR 48

4.5 Parameter values and their selection 51

4.6 Further details of HRL 53

4.7 HRL and machine learning 53

4.8 Summary 53

5 The method of surrender **55**

5.1 Lakatos's method of surrender 55

5.1.1 Two types of surrender 56

5.1.2 When should we give up on a conjecture? 56

5.2 Algorithm for the method of surrender 57

5.3 Illustrative sessions 57

5.3.1 Session one 58

5.3.2 Session two 59

5.3.3 Session three 59

5.4 Discussion 60

5.5 Summary 60

6 The method of monster-barring 63

6.1 Lakatos's method of monster-barring 64

6.2 Monster-barring in other mathematical domains 67

6.3 Key points in monster-barring 71

6.4 Implementing monster-barring 71

6.4.1 Lakatos's distinction between concepts and definitions 72

6.4.2 Making a concept less ambiguous 72

6.4.3 Concept stretching 73

6.4.4 When to perform monster-barring 73

6.4.5 Two decision points 73

6.4.6 Social dialogue 74

6.4.7 Further design considerations 76

6.5 The monster-barring algorithms 77

6.6 Illustrative sessions 81

6.6.1 Session one: barring the number one 81

6.6.2 Session two: adding the number one 82

6.6.3 Session three: barring the number zero 82

6.7 Discussion 83

6.8 Related work 86

6.8.1 Work on ambiguity and argumentation 86

6.8.2 Different interpretations of terms 88

6.9 Summary 89

7 The method of exception-barring **91**

7.1 Exceptions in Mathematics 91

7.2 Using exceptions to further knowledge: their uses and limits 92

7.3 Lakatos's two methods of exception-barring 93

7.4 Exception-barring in Number Theory 94

7.5 Exception-barring and other types of conjecture 95

7.6 Piecemeal Exclusion 98

7.6.1 Illustrative session 99

7.7 Counterexample barring 100

7.7.1 The counterexample-barring algorithm 102

7.7.2 Illustrative sessions 102

7.8 Strategic Withdrawal 103

7.8.1 Illustrative sessions 104

7.9 Related work 105

7.9.1 The boosting technique 105

7.9.2 Work in diagrammatic reasoning 106

7.9.3 Hayes-Roth's heuristics for repairing flawed plans 106

7.10 Summary 109

8 A computational representation of Cauchy's proof **111**

8.1 Cauchy's proof 112

8.2 Using counterexamples to improve a faulty conjecture and a faulty proof 114

8.3 Representing Cauchy's proof in HRL 115

8.3.1 Using Haggith's argument structures to represent Cauchy's proof . . . 115

8.3.2 Writing the proof as a series of Java methods 117

8.3.3 Using HR's production rules and conjecture making mechanisms to represent Cauchy's proof 119

8.4 A hybrid representation of Cauchy's proof 120

8.5 Summary 122

9 The method of lemma-incorporation 123

9.1 Three types of counterexample 124

9.1.1 Local, but not global counterexamples 124

9.1.2 Global and local counterexamples 125

9.1.3 Global, but not local counterexamples 125

9.2 Discussion of the method of lemma-incorporation 126

9.2.1 Combining the methods 127

9.2.2 Identifying a problem lemma in hidden lemma-incorporation 127

9.2.3 Controversy over whether a global counterexample is local or not . . . 128

9.2.4 The *type* of counterexamples in hidden lemma-incorporation 129

9.2.5 The number of applications of lemma-incorporation 130

9.3 Lemma-incorporation applied to other examples 130

9.3.1 Cauchy's Principle of Continuity 130

9.3.2 Hilbert's theorem 131

9.3.3 An example in group theory 134

9.4 Determining which type of lemma-incorporation to perform 136

9.5 Given a counterexample which is local but not global 138

9.5.1 Illustrative session 139

9.5.2 Discussion . . . 141

9.6 Given a counterexample which is both global and local . . . 141

9.6.1 Illustrative session . . . 141

9.6.2 Discussion . . . 143

9.6.3 An example in groupoid theory . . . 143

9.6.4 Illustrative session . . . 144

9.6.5 Discussion . . . 145

9.7 Given a global but not local counterexample . . . 145

9.7.1 Modelling surprise . . . 146

9.7.2 Illustrative session . . . 148

9.7.3 Algorithm for hidden lemma-incorporation . . . 151

9.7.4 Illustrative session . . . 151

9.7.5 Discussion . . . 153

9.8 The method of proofs and refutations . . . 154

9.8.1 Illustrative session . . . 154

9.9 Related work . . . 154

9.10 Summary . . . 156

10 Testing hypotheses in HRL **157**

10.1 Introduction . . . 157

10.2 Experimental setup . . . 158

10.2.1 Variables in HRL . . . 158

10.2.2 Variables at the student level . . . 158

10.2.3 Variables at the agency level . . . 160

10.2.4 Variable settings 160
10.2.5 Domains 160
10.3 It is possible to fine-tune the method of surrender 163
10.3.1 Experimental setup 163
10.3.2 Evaluation criteria 164
10.3.3 Results 165
10.3.4 Discussion 165
10.4 It is possible to fine-tune the method of monster-barring 168
10.4.1 Experimental setup 168
10.4.2 When to propose to bar an object 169
10.4.3 How to perform monster-barring 169
10.4.4 How to evaluate a proposal to bar an object 170
10.4.5 Evaluation criteria 172
10.4.6 When to propose to bar an object: results and discussion 174
10.4.7 How to perform monster-barring: results and discussion 175
10.4.8 How to evaluate a proposal to bar an object: results and discussion . . . 178
10.5 It is possible to fine-tune the method of exception-barring 181
10.5.1 Experimental setup 181
10.5.2 Evaluation criteria 184
10.5.3 Communal piecemeal exclusion results 184
10.5.4 Piecemeal exclusion and counterexample-barring results 185
10.5.5 Discussion 186
10.5.6 Strategic withdrawal results 188
10.5.7 Discussion 189
10.6 Conclusions 190

11 Philosophical evaluation **191**

11.1 Recap of criteria of a good theory 192

11.2 The criteria in terms of sets 192

11.3 Evaluating the method of surrender 194

11.4 Evaluating the method of monster-barring 196

11.4.1 Ambiguity in theory formation 196

11.4.2 When to perform monster-barring 196

11.4.3 Proposing a new definition 199

11.5 Evaluating the method of exception-barring 200

11.6 Evaluating the method of lemma-incorporation 201

11.7 Further criteria of a good theory 202

11.7.1 A theory should explain a range of possibilities 202

11.7.2 A good theory should be definite 202

11.7.3 A good theory should be rigorous 203

11.7.4 A good theory should be economical 203

11.7.5 A good theory should be extendable 204

11.8 Answers suggested by the computational approach 204

11.8.1 The scope of the methods 204

11.8.2 Applying Lakatos's methods to other types of conjecture 206

11.8.3 Comparing the methods 207

11.9 Applying Lakatos's MSRP to this project 207

11.9.1 Falsification 210

11.10 Summary 211

12 Application to automated theorem proving **213**

12.1 The Theorem Modifier System 213

12.2 Experiments and results 215

12.3 Conclusion 216

13 Further work **217**

13.1 Improving our implementation of Lakatos's methods 217

13.1.1 Extending the method of surrender 217

13.1.2 Extending the method of monster-barring 218

13.1.3 Implementing the method of monster-adjusting 219

13.2 Generating an initial problem and an initial proof scheme 220

13.2.1 Generating an initial problem 220

13.2.2 Generating an initial proof scheme 223

13.3 A more sophisticated agency 224

13.4 Flexible parameter settings 226

13.5 A cognitively plausible notion of mathematical concepts 227

13.6 Applications of our system 229

13.7 Conclusion 231

14 Conclusions **233**

14.1 Have we achieved our aims? 233

14.2 Contributions 235

14.2.1 Computational philosophy of science 235

14.2.2 Automated theory formation 235

14.2.3 Automated theorem proving 235

14.3 Conclusions 236

A Polyhedra and their Euler characteristic **237**

B Mathematical proofs **239**

B.1 Proof of Euler's conjecture 239

B.2 The Diagonalisation proof 240

B.3 Hilbert's proof 240

C Output from HRL **243**

C.1 Lemma-incorporation 243

D Further details of HR **245**

D.1 Production rules 245

Glossary of Philosophical terms **251**

Glossary of Mathematical terms **253**

Bibliography **255**

List of Figures

3.1 The construction path of the concepts in the conjecture *all primes are non-square*, where $b|a$ means that *b divides a*, $a*b$ means *a multiplied by b*, "&" is the logical connective *and*, and $|\{b : b|a\}|$ means *the size of the set of elements b such that b divides a*. The arrows between nodes represent the derivation which takes place when the given production rule and parameterisations are applied to an "old" concept, to get a new concept, with the direction of the arrow denoting going from old to new. 33

4.1 The interaction protocol of HRL 49

4.2 A Venn Diagram representation of the conjecture that for all *x*, $prime(x) \leftrightsquigarrow odd(x)$, where x is an integer between 1 and 10 50

7.1 Counterexamples to Euler's conjecture - for all polyhedra, V-E+F = 2 92

7.2 Supporting examples for Euler's conjecture; these are all convex 93

7.3 A Venn-diagram representation of the conjecture: for all polyhedra, V-E+F = 2 97

7.4 primes $\leftrightsquigarrow$ odds 97

8.1 Given the cube, after removing a face and stretching it flat, we are left with the network in part 1. After triangulating, we get part 2. When removing a triangle, we either remove one edge and one face, or two edges, one vertex and a face – shown in parts 3(a) and (b) respectively. 113

8.2 The picture frame, for which $V-E+F = 16-32+16 = 0$ 114

8.3 The original proof of Euler's conjecture, represented in Haggith's terms. The arrows represent the justification relation. 117

8.4 The first counter-argument, represented in Haggith's terms. Unmarked arrows represent the justification relation. 118

9.1 Given the network which results from taking the cube, removing a face and stretching it flat, and triangulating, we can remove a triangle which results in removing one face, no edges and no vertices. 125

9.2 If we remove a face from the cylinder and stretch it flat, then we either get case 1, if we remove an end face, or case 2, if we remove the jacket. Either way, we have satisfied the first lemma. 127

9.3 Diagram of Hilbert's proof of the theorem that for two points A and C, there always exists at least one point D on the line AC that lies between A and C. In the first figure, A and C are different points, and in the second, they are the same. 132

9.4 Given this graph, the concept 'remove triangle from graph' returns graphs resulting from removing triangle efg, or from removing triangle abf. 139

9.5 The ancestor tree for the right hand concept in lemma (1) 150

10.1 Results from testing the surrender variables. The numbers denote the sessions, and student one is represented as S1, etc.. The bottom right quadrant contains true positives; the top right quadrant contains false positives; the bottom left quadrant contains false negatives; and the top left quadrant contains true negatives. 166

10.2 The number of conjectures for which each of the five surrender variables give us desirable (above the *x*-axis) or undesirable (below the *x*-axis) results 167

10.3 The percentage of correct classifications, given a proposed monster, for each domain: sessions 49 - 68 180

10.4 The average percentage of correct classifications, given a proposed monster, over the three domains 181

10.5 The exception-barring results for sessions 2 - 16. In those sessions which do not appear, no modifications were performed. Session numbers appear in the figure. 187

11.1 A simplified Venn diagram representation of Lakatos's inspiring set (I), the mathematical conjectures which his theory explains (E), our inspiring set ($I1$), and the mathematical conjectures which our theory explains ($E1$) 194

13.1 The star polyhedron 220

13.2 Building an interface between a HRL, a schematic proof generator and a model generator would make a more complete cycle of mathematical discovery and justification. Automating step two would be a further project. 224

A.1 Regular polyhedra 237

A.2 The hollow cube: V - E + F = 16 - 24 + 12 = 4 237

A.3 The twin tetrahedra: V - E + F = 6 - 11 + 8 = 3 238

A.4 The picture frame: V - E + F = 16 - 32 + 16 = 0 238

A.5 The cylinder: V - E + F = 0 - 2 + 3 = 1 238

A.6 The star polyhedron: V - E + F = 12 - 30 + 12 = -6 (if each face is a pentagon); or 32 - 90 + 60 = 2 (if each face is a triangle) 238

B.1 The two possible connected plane graphs consisting of one edge, and their associated Euler characteristics 239

B.2 Hilbert's proof of the theorem that for two points A and C there always exists at least one point D on the line AC that lies between A and C. 241

Chapter 1

Introduction

Historically there are strong relationships between mathematics, the philosophy of mathematics, artificial intelligence, logic and computer science. In many ways these relationships are symbiotic. For instance, Frege developed predicate calculus as part of his program to reduce arithmetic to logic [Frege, 1879]. Robinson later developed the clausal form of predicate calculus, which is used in logic programming languages, such as PROLOG, and, along with his resolution principle, in automated theorem proving systems. Similarly, Russell and Whitehead's attempts to reduce mathematics to logic [Whitehead and Russell, 1913], in which they developed type theory, had important, albeit unintended, implications for computer science. Church developed another theory of types, and invented λ-calculus, intending it to provide a new foundation for logic in the style of Russell and Whitehead [Church, 1932]. This calculus, which focused on functions rather than classes, became the basis of LISP and other functional programming languages, and is strongly typed. Thus, philosophically and mathematically motivated work turned out to have important implications for computer science (these connections are shown in [Gillies, 2002]). Conversely, Gillies argues, ideas developed in computer science, such as non-monotonic logic, and more specifically the programming language PROLOG, have filtered back into logical theory. *Negation as failure*, in which $\neg p$ is derived from failure to derive p, is one example of a new non-monotonic inference rule which was developed in a computer science context.

Gillies [1996] examines the interaction between theories of scientific method and developments in AI. He outlines the philosophical debate between inductivists, such as Bacon, and anti-inductivists, such as Popper. Bacon argued that his definition of induction, generalising from a large number of independently observed patterns to a universal law, is the proper methodology of science. Additionally, he thought that it is a mechanical procedure which is not dependent on the intelligence or imagination of a scientist. Popper argued that it is impossible

to observe without a pre-existing theory which enables selection of observation, a language and a context. He also attempted to avoid using induction in science, developing his falsification methodology, whereby a scientist starts with a conjecture (which may require great leaps of intuition and imagination to devise), rather than an observation, and then attempts to refute it. Gillies claims that machine-learning algorithms have brought into existence inductive rules of inference, which is a mechanical method for generating hypotheses from a data set. He argues that machine learning systems such as ID3 [Quinlan, 1979, 1986] and GOLAM [Muggleton and Feng, 1992] which make inductive inferences based on many observations, can be seen as new tools for human scientists to use. Thus, many of Bacon's ideas about how induction has been carried out have been incorporated into successful machine-learning programs. This provides new support for Bacon's position in philosophy of science. However, there is also support for Popper's position, in that the rules used in machine learning do use background knowledge. Additionally, falsification plays a role in the process of testing hypotheses which have been induced (iterating a basic inductive rule of inference to produce a final rule). Therefore work in machine learning has affected the inductivist/anti-inductivist debate in the philosophy of science.

More generally, since Turing [1950] famously devised his computational experiment in order to answer the question "can machines think?", barriers between philosophy and AI have begun to blur. Sloman [1978] argues that philosophers of the future will learn via computational modelling about possible mechanisms which underlie the processes of inference and discovery and will therefore be in a better position to discuss the nature of mathematical discovery and other forms of *a priori* learning. He focuses on the power of computers and programmes to help us to develop new concepts and metaphors for thinking about complex systems. Seeing complex processes as computational processes, where information flows between sub-processes and construction and manipulation of symbolic structures is a new paradigm which can transform subjects including psychology, social and natural sciences. Philosophical questions about processes such as "How can concepts be acquired through experience, and what other methods of concept formation are there?", "Are there rational procedures for generating theories or hypotheses?" and "How can non-empirical knowledge, such as logical or mathematical knowledge, be acquired?" can be discussed in a new way using computational models, thus opening up avenues of enquiry. Just as AI can provide tools to clarify, reformulate, re-evaluate and give new answers to old philosophical problems, it will also generate new problems. Sloman argues that the importance of AI for philosophy is so great that: "within a few years, if there remain any philosophers who are not familiar with some of the main developments in artificial intelligence, it will be fair to accuse them of professional incompetence, and that to teach courses in philosophy of mind, epistemology, aesthetics, philosophy of science, philosophy of language, ethics, metaphysics, and other main areas of philosophy, without discussing the

relevant aspects of artificial intelligence will be as irresponsible as giving a degree course in physics which includes no quantum theory"[Sloman, 1978, chapter 1].

In this thesis we aim to extend the relationship between philosophy of mathematics and artificial intelligence, in particular automated theory formation and automated theorem proving. Lakatos [1976] was a philosopher of mathematics who argued against the deductivist style of mathematics, in which axioms, lemmas and concepts are first developed, then theorems, and finally proofs of the theorems are produced. Instead, he argued that these stages are interdependent and concepts, theorem and proofs evolve gradually together via contact with counterexamples. Additionally, Lakatos stressed the dialectical aspect of mathematics – experts have different beliefs and it is through the interaction of these beliefs that mathematics progresses. Lakatos characterises this interaction via methods, or heuristics. Our central thesis is that it is possible to give a computational reading of these methods. Our secondary thesis is that it is useful to do so, both from the philosophy of mathematics perspective, and the AI perspective.

We have constructed an agent architecture which enables dialogue, where the problem of the agency is to model the social process of concept and conjecture refinement described by Lakatos, with the task being to develop interesting concepts and conjectures to add to a mathematical theory. The knowledge the agency starts with consists of objects of interest, such as integers, and concepts such as multiplication and addition, which is provided at the start of a session. A general motivation is to develop an interesting theory about mathematical entities and concepts, to find patterns in the data and to invent and explore properties within the theory. Specifically, agents use Lakatos's methods to accept, modify, or reject conjectures. Agents communicate by sending conjectures, concepts, counterexamples, and requests, and discussion is directed by a coordinator agent. In order to build such a model we use an automated theory formation program HR [Colton, 2002]. This provides us with a context within which to develop ideas about communication between mathematicians, as identified by Lakatos.

Although aspects of the methods which Lakatos identifies in [Lakatos, 1976] have been explicitly modelled, such as [Hayes-Roth, 1983], or implicitly modelled, such as [Skalak and Rissland, 1991] (which are both described in §7.9), this is the first attempt at a systematic implementation and subsequent evaluation of the methods.

1.1 Motivation

Our motivation for implementing Lakatos's methods is twofold:

• to clarify, extend and evaluate the methods (philosophy of mathematics);

• to develop new and useful techniques in artificial intelligence: in particular in the fields of automated theory formation and automated theorem proving.

These twin motivations overlap to a certain extent, in particular in the area of testing the methods. However we state them separately as each discipline has its own methodology and evaluation criteria.

Clarify, extend and evaluate Lakatos's theory

Lakatos's work has been praised as a "masterpiece" [Kadvany, 2001, p.1]; "the first major bridge between historical philosophy and serious mathematics" [Kadvany, 2001, p.14]; a "brilliantly sustained tour de force" [Feferman, 1978, p.311]; and "a philosophical and literary achievement of the stature of Hume on natural religion or Berkeley's Hylas and Philonous" [Hacking, 1981, p.135]. However, it has also been widely criticised. For instance, although Lakatos claimed that his methods applied to many areas of mathematics, as well as empirical sciences, Lakatos only considered two case studies, in geometry and real analysis. Feferman [Feferman, 1978] argues that these case studies are not sufficient to claim that these methods have a general application. Implementing a model allows us to test the scope of Lakatos's methods in terms of what domains they can usefully be applied to. Additionally, Feferman points out that the conjecture in both case studies is an implication conjecture, i.e. of the form $\forall x[A(x) \rightarrow B(x)]$. In our implementation we have extended the methods to apply to two further types of conjecture: equivalence conjectures and non-existence conjectures, of the form $\forall x[A(x) \leftrightarrow B(x)]$ and $\neg\exists x[A(x)]$.

We believe that since Lakatos's work [Lakatos, 1976] was the first attempt to characterise informal mathematics (see [Corfield, 1997] and [Feferman, 1978]), it is likely to be incomplete, and hence be open to criticism and extension. Lakatos himself neither considered the methods complete nor definitive, arguing only that they provide a more realistic and helpful portrayal of mathematical discovery than the Euclidean methodology. We argue that, in accordance with the computational philosophy paradigm, implementing Lakatos's theory provides a new way of understanding the methods.

The dialogue format of Lakatos's methods enables us to model social processes, and this interaction between agents constitutes an important part of our work. Our implementation contrasts programs such as BACON [Langley et al., 1987] and PI [Thagard, 1993] which are more concerned with the thought processes of an individual.

New techniques in automated theory formation and automated theorem proving

Automated theory formation is the field in which the processes of inventing new concepts, finding examples, and making new conjectures are automated and interlinked. We extend the capabilities of an automated theory formation program HR [Colton, 2002], and argue that our extensions improve its theory formation abilities. In particular it can now generate interesting conjectures and concepts which it was previously unable to, discuss and reject objects which are controversial, and represent and improve proof plans.

A final motivation is in the field of automated theorem proving. Current automated theorem provers are inflexible in that if they are passed a conjecture which is either false or too difficult to prove within the allocated time, they simply fail. Using Lakatos's methods, Colton and Pease have jointly implemented a system which is able to take in a conjecture, try to prove it and, if unsuccessful, produce modified versions of the conjecture which it *can* prove.

1.2 Aims of project

We aim to:

• *provide a computational reading of Lakatos's theory*, by interpreting his methods as algorithms and implementing these algorithms as a computer program;

• *clarify and extend the methods*, by providing various interpretations and further divisions of the methods, and extending them to apply to equivalence and non-existence conjectures;

• *test the domains to which the methods apply*, both further mathematical domains which were not looked at by Lakatos, namely number theory and group theory, and non-mathematical domains, for instance to theories about animal taxonomies;

• *evaluate the methods*, by testing two hypotheses: *(i)* the techniques are of general use (rather than working only in a contrived domain), and *(ii)* the techniques can be applied to other types of conjecture ; and

• *evaluate our implementation*, in particular testing our hypothesis that it is possible to fine-tune the methods.

1.3 Contributions

This is the first systematic automated realisation of Lakatos's methods. We contribute to the philosophy of mathematics and computational philosophy of science by providing a clarification of the methods. We also contribute to automated theory formation by developing and exploring the methods and demonstrating their usefulness. A further contribution is demonstrating the usefulness of the methods within the field of automated theorem proving. This thesis also provides a new evaluation of the work of Lakatos on the basis of criteria proposed by several theorists, including Lakatos himself in his other well known work on evaluating scientific research programmes [Lakatos, 1970].

1.4 Organisation of thesis

We organise the rest of the thesis as follows:

In chapter 2 we introduce the field of computational philosophy of science. We also briefly outline the background to Lakatos's work, describe his methods of mathematical discovery and justification, and discuss various reactions to his work. This chapter provides the motivation to model Lakatos's methods, and places them within a philosophical context.

In chapter 3 we describe HR, the automated theory formation system which we build upon. This system is able to automatically generate and evaluate conjectures and concepts in a variety of domains (where concepts are defined largely extensionally), using only a few heuristics and basic background knowledge. We also outline work in agent architectures and argumentation which we later draw upon. This chapter provides the technical details which are necessary in order to understand our system. Finally, we introduce our methodology.

In chapter 4 we describe the design specification of our system and give an overview of the system. This includes the agent architecture which we have developed, and details of the language which the agents use to communicate.

The next five chapters, 5, 6, 7, 8 and 9, contain the main body of work, and discuss our work with Lakatos's methods of surrender, monster-barring, exception-barring and lemma-incorporation. Each chapter, except chapter 8, considers one method and our implementation of it. Firstly we describe the method and discuss how it may be applied to other domains. We use a simplified representation of a subset of number theory as a running domain within which to develop our ideas (this simplification already existed in Colton's work [Colton, 2002]). Since

the methods differ quite widely from each other, it is sometimes necessary to introduce further background material which is relevant to the particular method at this stage. We then describe our implementation of the method and the algorithms which we have devised. We demonstrate our system working by giving several illustrative examples, and then discuss our implementation. Again, because of the differences in the methods, there is work which is related only to a particular method, and so we outline relevant related work at the end of each of these chapters. Chapter 8 is a prelude to chapter 9, in which we discuss our computational representation of Cauchy's proof, which is necessary for the following chapter.

Chapters 10, 11 and 12 contain the evaluation of our project. In chapter 10 we outline our three main hypotheses: that it is possible to fine-tune the methods of surrender, monster-barring and exception-barring, and we present and discuss our experiments and results. In chapter 11 we evaluate our extended theory of mathematical discovery and justification and consider whether the computational approach has enabled us to improve upon Lakatos's theory. Where appropriate, we refer to illustrative sessions as evidence for our claims. In chapter 12 we evaluate our claim that Lakatos's methods can be applied to automated theorem proving, by presenting a spin-off system, TM, developed by Colton and Pease, and demonstrating its application to this field.

Chapter 13 contains ideas for further ways to develop and apply Lakatos's work. We conclude in chapter 14.

Appendix A contains figures of polyhedra and their Euler characteristic. Appendix B contains a proof of Euler's conjecture, the diagonalisation proof of the theorem that the set of real numbers is uncountable, and Hilbert's proof of the theorem that for two points A and C there always exists at least one point D on the line AC that lies between A and C. In appendix C we present further output from our system, in addition to that which can be found in the main text. In appendix D we present additional details of HR. We include these for completeness sake; they may be omitted without loss. Finally, we provide a glossary of philosophical and mathematical terms.

1.5 Summary

Lakatos outlined various methods by which mathematical discovery and justification can occur. These methods suggest ways in which concepts, conjectures and proofs gradually evolve via interaction between mathematicians. We hold that *(i)* it is possible to provide a computational reading of these methods, and *(ii)* it is useful to do so. To evaluate our hypotheses we develop

a system within which we implement Lakatos's theory. We show that we improve upon the state of the art in both automated theory formation and automated theorem proving, and that we have provided a computational model of Lakatos's theory which both extends and clarifies his thoughts on the philosophy of mathematics.

Chapter 2

Computational Philosophy of Science

In this chapter, we motivate the project by introducing the field of computational philosophy of science, and outlining Lakatos's theory of discovery and justification in mathematics. In §2.1, we define computational philosophy of science, briefly describe two models of discovery, and outline three open problems in the field, identified by Thagard [1998]. In §2.2, we outline the background to Lakatos's work; in §2.3, we describe his methods of mathematical discovery and justification; and in §2.4, we discuss reaction to his work. We then consider why a computational reading of Lakatos's ideas is a worthwhile endeavour, in §2.5, and consider possible objections to such a reading in §2.6. We conclude the chapter in §2.7.

2.1 Computational Philosophy of Science

Computational philosophy of science uses computational modelling to understand the growth and structure of scientific knowledge, in particular how hypotheses are constructed and evaluated. This complements and extends philosophy of science, in which logical analysis and historical case studies are used. Sloman [1978] argues that the computational paradigm provides new tools for understanding the processes which philosophers study, including the philosophy of mathematics. Thagard [1993] also emphasises that philosophy of science and artificial intelligence have much to learn from each other. Computational data structures are seen as analogous to human mental representations, and algorithms analogous to mental procedures. Artificial intelligence brings new conceptual resources to philosophy of science, as well as a new methodology, which involves constructing and testing computer models, to the philosophy of science. Implementing a theory as a computer program requires precision, provides a test of its assumptions (both implicit and explicit) about the structure and processes of scientific development, and enables us to test the underlying theory by investigating how well the

program works on examples of different kinds. Ways of evaluating computational models of scientific discovery and evaluation include examining whether the model is a genuine instantiation of the theoretical ideas, and the program a genuine implementation of the model; testing to see whether the model applies to examples other than those which were used in the model's development; whether the model scales up to examples which are larger and more complex than the ones to which it has been applied; its qualitative fit – whether it performs similar tasks to humans in similar ways; its quantitative fit – whether it simulates quantitative aspects of psychological experiments; and whether it is compatible with representations and processes found in theoretical accounts. The goal of the computational modelling might be to model human performance (the cognitive modelling approach), or to develop machine intelligence (the engineering approach).

Simon [1997] points out that the question of whether philosophy of science should be concerned with how scientists discover hypotheses has in part been answered by researchers in computational philosophy of science, who have developed models of scientific discovery based on heuristic search. Programs which employ heuristic search and show that it is much more efficient than random search suggest that discovery can be explained by laws, although these laws may well not be unique or complete. For instance, Langley et al. [1987] developed BACON, which takes empirical data and conducts a heuristic search over the instance space and the hypothesis space, for a law that fits the data. Using this limited data, and with no theoretical knowledge, BACON has rediscovered many of the most important laws of physics and chemistry that were discovered in the 18th and 19th centuries, including Kepler's third law of planetary motion. Other models, such as PI [Thagard, 1993], are able to generate and combine new theoretical concepts, and perform abduction, inventing new hypotheses to explain puzzling phenomena. Another idea, which Simon [1997] argues supports the view that there can be a theory of scientific discovery, is the rejection of the view that discovery and verification are disjoint processes. If instead, they are seen as closely intermingled, then theories about how to evaluate a hypothesis should also account for its generation. Lakatos held and expanded this view, as described below.

Thagard [1998] highlights three open problems which he considers to be amenable to computational, or philosophical, investigation. The first concerns how new questions, such as "how might species evolve?", are generated. The second concerns the role of visual imagery in the structure and growth of scientific knowledge. The third concerns social processes such as how consensus is formed in science. Computational models of scientific discovery and evaluation, such as BACON [Langley et al., 1987] and PI [Thagard, 1993] usually concern the thought processes of individual scientists. However, Thagard [1998] argues that ..."it might be possible to develop models of social processes such as consensus formation along the lines of the field

known as distributed artificial intelligence which considers the potential interactions of multiple intelligent agents" [Thagard, 1998, p. 7]. Simon [1997] also stresses that science is a social enterprise.

2.2 Lakatos's logic of discovery and justification in mathematics

Lakatos attacked the view that mathematical knowledge is timeless, certain and *a priori* [Lakatos, 1976]. Lakatos's work in the philosophy of mathematics is a controversial mathematical analogy of Hume's problem of induction combined with Popper's theory of falsification. That is, Lakatos both identified the problem of the impossibility of mathematical knowledge, and suggested a solution. His solution consisted of heuristic methods which guide the development of mathematical conjectures, concepts and proofs. These evolve through dialectic and analysis triggered by counterexamples. Counterexamples, therefore, play a vital role in [Lakatos, 1976], though they are a starting, rather than finishing point: criticism has to be constructive if it is to be valuable.

Lakatos's work on philosophy of mathematics had three major influences. Firstly, Hegel's dialectic, in which the *thesis* corresponds to a naïve mathematical conjecture and proof; the *antithesis* to a mathematical counterexample; and the *synthesis* to a refined theorem and proof (described in these terms in [Lakatos, 1976, pp.144-145]). Lakatos emphasises the dialectical aspect in the style of his book, which takes the form of a dialogue in a classroom. Thus he is able to represent different mathematical and philosophical positions by using the voices of different students. The role of the teacher in the book is to ensure that the discussion keeps moving and they do not get caught up in petty asides or dead end avenues. Secondly, Lakatos used Popper's ideas on the impossibility of certainty in science and the importance of finding anomalies. Lakatos argued that Hegel and Popper "represent the only fallibilist traditions in modern philosophy, but even they both made the mistake of preserving a privileged infallible status for mathematics" [Lakatos, 1976, p.139]. Thirdly, Polya [1954] and his work on mathematical heuristic, the study of the methods and rules of discovery and invention, was also a major influence; in particular his work on defining an initial problem and finding a conjecture to develop. Lakatos claims that his own work starts where Polya's leaves off.

Rather than being concerned with whether mathematical knowledge is possible (the argument between dogmatists, who claim that we can know, and sceptics, who claim that we cannot know, or at least that we cannot know that we know), or what type of knowledge it might be, Lakatos emphasised the importance of guessing. In [Lakatos, 1978a], he argued that the important question is not *how do we know?*, but rather *how can we improve our guesses?* He

presented a fallibilist approach to mathematics, in which proofs, conjectures and concepts are fluid and open to negotiation. Lakatos strongly criticised the deductivist approach in mathematics, in which definitions, axioms and theorem statements are presented with no explanation about their development, and considered to be eternal, immutable truths, and he also rejected the Hilbertian notion of proof. Instead, Lakatos saw mathematics as an adventure in which, via patterns of analysis, conjectures and proofs are gradually refined but never certain. He warned that hiding this process makes the subject impenetrable to students and prevents experts from developing concepts or conjectures which may arise out of earlier versions of a theorem statement. Lakatos demonstrated his argument by presenting case studies of the development of Euler's conjecture that for any polyhedron, the number of vertices (V) minus the number of edges (E) plus the number of faces (F) is equal to two; and Cauchy's proof of the conjecture that the limit of any convergent series of continuous functions is itself continuous. [Lakatos, 1976] is a rational reconstruction of the history of philosophy of mathematics as well as these two mathematical conjectures, which traces psychologism, intuitionism, rationalism, historicism, pragmatism, dogmatism, Kant's idea of infallible mathematics, refutationism, inductivism and deductivism. As one of the characters puts it [Lakatos, 1976, p.55], they discuss the packaging – the philosophical framework, as well as what is in the packet – the mathematical content.

Lakatos held an essentially optimistic view of mathematics. He saw the process of mathematical discovery, traditionally thought of as impenetrable and inexplicable by rational laws, and considered to be lucky guess work or intuition, in a rationalist light – thereby opening up new arenas of rational thought. He challenged Popper's view [Popper, 1972] that philosophers can form theories about how to evaluate conjectures, but not how to generate them, which should be left to psychologists and sociologists. Rather, Lakatos believed that philosophers could theorise about both of these aspects of the scientific and mathematical process. He challenged Popper's view in two ways - arguing that *(i)* there *is* a logic of discovery, the process of generating conjectures and proof ideas or sketches *is* subject to rational laws; and *(ii)* the distinction between discovery and justification is misleading as each affects the other; i.e., the way in which we discover a conjecture affects our proof (justification) of it, and proof ideas affect what it is that we are trying to prove (see [Larvor, 1998]). This happens to such an extent that the boundaries of each are blurred.

The first chapter of [Lakatos, 1976] was originally published as [Lakatos, 1963a], [Lakatos, 1963b], [Lakatos, 1963c] and [Lakatos, 1964]. The second chapter and the appendices of [Lakatos, 1976], however, were not published during Lakatos's lifetime, as he saw this work as an unfinished project (perhaps analogous to his view of mathematics). One drawback of this failure to publish is that he could not answer criticisms of the book. The fact that Lakatos never pronounced himself completely happy with his theory does, however, strengthen our argument

that this work is worth implementing, as it implies that there may be gaps in the theory which we can hopefully identify and fill.

2.3 Lakatos's three main methods

Lakatos [1976] explicitly outlines six methods for modifying mathematical ideas and guiding communication: surrender, monster-barring, exception-barring, monster-adjusting, lemma-incorporation, and proofs and refutations. Of these, the three main methods of theorem formation are monster-barring, exception-barring, and the method of proofs and refutations [Lakatos, 1976, p.83]. Crudely speaking, monster-barring is concerned with concept development, excep tion-barring with conjecture development, and the method of proofs and refutations with proof development. However, these are not independent processes; much of Lakatos's work stressed the interdependence of these three aspects of theory formation. In the following sections we give a brief outline of the main methods, which are discussed in more detail in chapters 6, 7, and 9.

2.3.1 The method of monster-barring

Monster-barring is a way of excluding an unwanted counterexample. This method starts with the argument that a 'counterexample' can be ignored because it is *not* a counterexample, as it is not within the claimed concept definition. Rather, the object is seen as a monster which should not be allowed to disrupt a harmonious theorem. For instance, one of the students suggests that the hollow cube (a cube with a cube-shaped hole in it) is a counterexample to Euler's conjecture, since $V - E + F = 16 - 24 + 12 = 4$ (see appendix A). Another student uses monster-barring to argue that the hollow cube does not threaten the conjecture as it is not in fact a polyhedron. The concept polyhedron then becomes the focus of the discussion, with the definition being formulated explicitly for the first time; as 'a solid whose surface consists of polygonal faces' (according to which, the hollow cube *is* a polyhedron), and 'a surface consisting of a system of polygons' (according to which, the hollow cube is *not* a polyhedron) [Lakatos, 1976, p.14]. Using this method, the original conjecture is unchanged, but the meaning of the terms in it may change.

2.3.2 The method of exception-barring

Lakatos's treatment of exceptions is noteworthy for two reasons. Firstly, he highlights their role in mathematics — traditionally thought of as an exact subject in which the occurrence of

exceptions would force a mathematician to abandon a conjecture. Secondly, Lakatos showed how exceptions, rather than simply being annoying problem cases, which we may be able to dismiss as monsters, can be used to further knowledge. Lakatos discusses two forms of exception-barring: *piecemeal exclusion* and *strategic withdrawal*. *Piecemeal exclusion* deals with exceptions by excluding a whole class of counterexamples. This is done by generalising from a counterexample to a class of counterexamples which have certain properties. For instance, the students generalise from the hollow cube to *polyhedra with cavities*, and then modify Euler's conjecture to 'for any polyhedra without cavities, $V-E+F=2$'. Thus exceptions are seen as objects which are valid (in this case, as polyhedra), as opposed to monsters, and force us to modify a faulty conjecture by changing the domain to which it refers. *Strategic withdrawal* is the only one of the methods which does not directly use counterexamples. Instead, it uses positive examples of a conjecture and generalises from these to a class of object, and then limits the domain of the conjecture to this class. For instance, the students generalise from the regular polyhedra to *convex polyhedra*, and then modify Euler's conjecture to 'for any convex polyhedra, $V-E+F=2$'.

2.3.3 The method of proofs and refutations

As the title of his book suggests, this is considered by Lakatos to be the most important method. Commentators and critics, for instance [Feferman, 1978], usually share this view, often seeing the rest of the book as a prelude to this method. It starts off as the method of *lemma incorporation*, and is developed via the dialectic into the method of *proofs and refutations*. This method works on a putative proof of a conjecture.

Lemma incorporation works by distinguishing global and local counterexamples. The former is one which is a counterexample to the main conjecture, and the latter is a counterexample to one of the proof steps (or lemmas). A counterexample may be both global and local, or one and not the other. When faced with a counterexample, the first step is to determine which type it is. If it is both global and local, i.e., there is a problem both with the argument and the conclusion, then one should modify the conjecture by incorporating the problematic proof step as a condition. If it is local but not global, i.e., the conclusion may still be correct but the reasons for believing it are flawed, then one should modify the problematic proof step but leave the conjecture unchanged. If it is global but not local, i.e., there is a problem with the conclusion but no obvious flaw in the reasoning which led to the conclusion, then one should look for a hidden assumption in the proof step, then modify the proof and the conjecture by making the assumption an explicit condition.

Proofs and refutations consists of using the proof steps to suggest counterexamples (by look-

ing for objects which would violate them). For any counterexamples found, it is determined whether they are local or global counterexamples, and then lemma incorporation is performed.

2.4 Reaction to Lakatos's theory

Since [Lakatos, 1976] was the first attempt to characterise informal mathematics (see [Corfield, 1997] and [Feferman, 1978]), it is likely to be incomplete, and hence be open to criticism and extension. For instance, [Feferman, 1978, pp. 316-320] has identified ten criticisms of Lakatos [1976], consisting mainly of highlighting gaps in his theory. We list them below.

(i) What happened before 1847?[1]

(ii) Is the method most appropriate to describe transitional foundational periods?

(iii) How does this "logic of mathematical discovery" relate to working experience?

(iv) Is there no end to guessing?

(v) What constitutes improvement in a proof?

(vi) What constitutes an initial proof? Where does it come from?

(vii) What is the form of the conjectures?

(viii) Can ordinary logical analysis account for the same examples as the method of proofs and refutations?

(ix) Are there no crystal-clear concepts?

(x) What is distinctive about mathematics?

We argue that the process of providing a computational representation enables us to answer some of these questions and also raises more questions which are then answered. We examine this claim in chapter 11.

Despite any potential gaps in Lakatos's theory, his work is still relevant today. This is evidenced by its inclusion in a recent work on new directions in the philosophy of mathematics [Tymoczko, 1998], in which [Lakatos, 1978b] attacks foundationalism and argues for a renaissance of empiricism in mathematics. Tymoczko includes Lakatos as a representative of a "major perspective on the philosophy of mathematics" [Tymoczko, 1998, p. 1].

[1] Feferman [1978] points out that Lakatos claimed that the method of proofs and refutations was discovered in 1847 by Seidel. Clearly mathematical progress was made prior to this date and yet, Feferman states, Lakatos offers no ideas as to how.

2.5 A computational reading of Lakatos's theory

We hold that Lakatos's theory of discovery and justification in mathematics is an important theory to model within the computational philosophy paradigm. This is because it is open to improvement, it is relevant today, and furthermore its emphasis on dialogue enables us to model social processes – thus our work addresses the third open problem in computational philosophy of science identified by Thagard [1998].

In accordance with the computational philosophy paradigm, implementing Lakatos's theory should enable us to improve upon it. Modelling Lakatos's theory has two main benefits:

• the process of having to write an algorithm for the methods forces us to identify areas in which Lakatos was vague, and aspects he omitted;

• running the model allows us to test hypotheses about the methods, for instance that they generalise to scientific thinking, or that one method is more useful than another. These hypotheses may be claims that Lakatos or other commentators have made, or new ones.

A further benefit of implementing Lakatos's ideas is that they suggest ways of improving the fields of automated theory formation and theorem proving (see chapters 10 and 12).

2.6 Possible objections to a model of Lakatos's ideas

Larvor [1998] refers to Lakatos as "the most celebrated critic of formalism in the philosophy of mathematics", and Lakatos certainly emphasised the informal nature of mathematics. Therefore, our attempt to implement and thus formalise it could be seen as being as objectionable as the editors' controversial addition of the 'final' chapter (chapter 2) in the history of Euler's conjecture, which presented Lakatos's work as a finished philosophy rather than a step on the path of Hegel's dialectic. In this chapter, Poincaré's vector-algebraic proof of Euler's conjecture is presented, as an example of a proof which satisfies Euclidean standards, and can be known beyond any possibility of doubt. This contrasts sharply with Lakatos's expansion in the remainder of [Lakatos, 1976]. Our justification for our approach is threefold. Firstly, we by no means finalise the work, nor even aim to. We investigate which parts can be formalised and whether and how that adds to Lakatos's work. We do not see modelling informal mathematics as paradoxical, but rather as a contribution both to philosophy in terms in investigating what can be formalised and how that affects a theory, and to automated theory formation and mathematical reasoning by modelling mathematics as it is actually done by humans.

Secondly, even if it were the case that Lakatos would disapprove of an implementation of his work as a computer program, it may still prove profitable to develop his ideas in this new direction. It often happens that a person's work is developed in a direction which they did not foresee and may not have approved of. This does not necessarily detract from the value of the new work, which should be judged solely on its own merit.

Our third justification lies in questioning whether Lakatos would actually object to a formal representation of his work. It may be that what Lakatos objected to were the current rules of formalism, not the doctrine itself. In fact, the heuristics which he set out in [Lakatos, 1976] can be seen as new rules within game formalism (where we use the term "game formalism" in the way defined in philosophy of mathematics, rather than game theory: see glossary for details). We have seen (§2.2) that Lakatos *did* believe that there was a rationality of discovery. Therefore, we can see him as a game formalist who suggested different rules to the game. Clearly, as with many games, there is no claim that a given set of rules is complete or unique, rather that they make the game better in some way.

2.7 Conclusion

As Simon [1997] argues, computer modelling, performed in conjunction with historical studies and laboratory experiments, has proved to be a powerful tool for building a computational theory of the processes of scientific discovery and evaluation. The resulting theory is then useful for understanding processes which human scientists employ, as well being a valuable discovery aid to scientists, either in interactive or fully automated mode.

We hold that it is possible to implement Lakatos's theory, in which different agents discuss and modify conjectures, concepts and proofs via the methods described above. Additionally, we hold that it is useful to do so.

Chapter 3

Background

This chapter provides the technical details which are necessary in order to understand our system, and the methodological details necessary to understand our evaluation of the project. Firstly, in §3.1 we describe HR [Colton, 2002], the automated theory formation system which we build upon. HR is able to automatically generate and evaluate conjectures and concepts in a variety of domains, using only a few heuristics and basic background knowledge, where the concepts are largely defined in an extensional way. Since our system uses and extends HR, it is necessary to understand how it represents, generates and evaluates concepts and conjectures, in order to understand the theory formation aspect of our system. Then, in §3.2, we outline work in agent architectures and argumentation, which we draw upon for the dialogue aspect of our system. Finally, in §3.3, we introduce our methodology and evaluation criteria.

3.1 Automated theory formation and HR

Automated theory formation, as advocated by Colton [2002], aims to combine various reasoning techniques (inductive, deductive, and abductive) in order to build theories containing concepts, hypotheses, examples, proofs, etc. In [Colton, 2002] the theories were constructed in mathematical domains, and built from first principles. Constructing a theory includes finding examples of objects of interest, inventing new concepts, making plausible statements relating those concepts, and proving and disproving conjectures.

Below we discuss some technical details of HR, with reference to a running example.

3.1.1 Representation of objects of interest and concepts in HR

Objects of interest are the entities which a theory discusses. For instance, in number theory the objects of interest are integers, in group theory they are groups, etc. These are represented by unary predicates, where the argument is the object of interest, for instance, $integer(1)$. We use the terms entities and objects of interest interchangeably.

In mathematics, the word 'concept' has many interpretations. These include:

• a description of a set of objects of interest. For instance, the concept of a natural number, or a prime number;

• a relation between two or more objects. For instance, the relation *larger than*, *equals*, and *divides*, shown in the examples $5 > 2$, $4 = 2 + 2$, and 3 divides 6;

• a function, which is a mapping from one set of objects (the domain) to another set of objects (the co-domain), associating a unique output in the co-domain for each input in the domain. For instance, the number of divisors of a number (the *tau function*) maps from the natural numbers to the natural numbers.

The representation of concepts in HR comes from an analysis of the different interpretation of the word 'concept', and the way in which new mathematical concepts are presented in textbooks. Colton [2002] references [Kerber, 1992] as arguing that mathematical concepts are introduced with a definition, examples, and often lemmas about some property of the concept. For instance, in [Barnard and Neill, 1996], the concept of a mathematical group is first introduced with two examples of groups (and some of their characteristics highlighted); the definition of a group is then given in terms of the group axioms (see D.1); and this is followed by theorems about groups, the first being that the identity element of a group is unique. Colton tries to capture these aspects of a concept with a tripartite representation of concept: a *definition*, a *data table* (or table of examples), and a *categorisation*. The data table is a function from an object of interest, such as the number 1, or the prime 3, to a truth value or a set of objects. Any objects of interest with a the same value under this mapping are grouped together in a *categorisation*. Other concepts may have resulted in the same grouping, or categorisation of the entities, in which case the concepts are said to share a categorisation. Conversely, a particular categorisation may be a new grouping of the objects within a theory. Categorisations are used by HR in a part of its judgement as to whether a new concept is interesting, where it considered to be more surprising if it results in a new categorisation. For instance, the concept *tau function*, in a theory where the objects of interest are the integers $1-10$, is represented as shown below. Note that in the definition we show, $S = \{x : P(x)\}$ denotes the set of all elements

x such that some property $P(x)$ holds, $|S|$ denotes the size (number of members) of the set S, $x|y$ denotes *x divides y*), and $\wedge$ denotes the logical connective *and*. The tau function then, maps an integer a to the number of divisors of a.

definition: $[a, b] : a$ is an integer $\wedge$ b is an integer & $b = |\{c: c \text{ is an integer} \wedge c|a\}|$
data table: $f(1) = [[1]]$
$f(2) = [[2]]$
$f(3) = [[2]]$
$f(4) = [[3]]$
$f(5) = [[2]]$
$f(6) = [[4]]$
$f(7) = [[2]]$
$f(8) = [[4]]$
$f(9) = [[3]]$
$f(10) = [[4]]$
categorisation: $[[1],[2,3,5,7],[4,9],[6,8,10]]$

Another example is the concept *prime number*. This is defined in terms of the tau function, i.e., a is a prime if and only if the size of the set of integers which divide the integer a is equal to 2, or a is a prime if and only if a is an integer with exactly two divisors. We show its representation in HR, in a theory where the objects of interest are the integers $1-10$, below.

definition: a is an integer $\wedge$ $|\{b : b \text{ is an integer} \wedge b|a\}| = 2$
data table: $f(1) = false$
$f(2) = true$
$f(3) = true$
$f(4) = false$
$f(5) = true$
$f(6) = false$
$f(7) = true$
$f(8) = false$
$f(9) = false$
$f(10) = false$
categorisation: $[[2,3,5,7],[1,4,6,8,9,10]]$

All concepts in HR are represented by some or all of these aspects. There are three different types of concept in HR, which we present in order of increasing sophistication.

1) The simplest type of concept is given solely in terms of an enumerative definition. That is, at the beginning of a session, the user gives an explicit list of all objects which fall under the definition. This list is considered to be exhaustive, and no other definition is given. Other objects in the theory, or new objects which may arise during theory formation are not considered

to be examples of the concept. An example might be the concept *animal*, about which the only information given is a finite list of animals, for example {dog, dolphin, platypus, bat, trout, herring, shark, eel, lizard, crocodile, t-rex, turtle, snake, eagle, ostrich, penguin, cat, dragon}. Thus, *animal* is represented purely as the predicates: animal(dog), animal(dolphin), etc, with no other meaning of the concept animal provided either as hard code or in a way that the system can generate. Another concept in this domain is *homeothermic*. In the input domain file, the user might specify that if an object of interest is homeothermic then it is an animal, and then give the extension of homeothermic in terms of the objects of interest in the theory, for example, {dog, dolphin, platypus, bat, eagle, ostrich, penguin, cat}. Again, these would be listed as predicates; homeothermic(dog), homeothermic(dolphin), homeothermic(platypus), etc. Given a new object, for instance, hippopotamus, HR would have no principled way of determining whether it was homeothermic, and therefore would treat it as not.

Another example is the concept *integer*, which is hard-coded as a finite list of integers, for instance, integer(1), integer(2), integer(3), etc., where "1", "2", "3" are the objects of interest. Given a new object, for instance, "40", or "cat", HR has no way to determine whether it is an integer or not, other than its closed world assumption, which would mean that it would evaluate the new object as a non-integer. Concepts of this type are represented in HR by a data table, which is a function from object to truth value for every object of interest in the theory. This is the extension of the concept; for instance, the concept *integer* would have the data table: $1 = true$; $2 = true$; $3 = true$; $4 = true$; $5 = true$ for entities 1 - 5, and the concept *homeothermic* would have the data table: $dog = true$, $dolphin = true$, $platypus = true$, $bat = true$, $trout = false$, $herring = false$, $shark = false$, $eel = false$, $lizard = false$, $crocodile = false$,$t-rex = false$, $turtle = false$, $snake = false$, $eagle = true$, $ostrich = true$, $penguin = true$,$cat = true$, $dragon = false$ for the entities listed above. Note that part of the representation of the concept *homeothermic* is that $homeothermic(A) \rightarrow animal(A)$, and so HR would not evaluate integers with respect to this concept, and the data table would not contain $1 = false$, etc.

In terms of categorisation, the concept *integer* would have a single categorisation [[1,2,3,4,5]], whereas the concept *homeothermic* would have the categorisation [[dog, dolphin, platypus, bat, eagle, ostrich, penguin, cat], [trout, herring, shark, eel, lizard, crocodile, t-rex, turtle, snake, dragon]].

2) A concept might also be given to the system in terms of both its extension and intension. That is, a concept is given in a domain file at the start of a session and HR has the functionality to enable it to generate examples of the concept for its objects of interest. An example of this is the concept of addition. This is represented as a tuple $[x,y,z]$ where $x = y + z$. The code for calculating the extension of the concept of addition, for a given, finite set of integers, is

in a class file of HR called *UserFunctions*, which contains code for generating data tables for concepts in number, graph and group theory (we have also extended this functionality). In this example, given a new entity x, the algorithm for the concept addition would convert it to an example of the Java class Integer, get its "int" value (which is a basic type in Java), and then cycle through the integers, with an integer i which takes each integer value from 0 to x, each time adding the current value of i and $x-i$ to the tuple. For instance, given the entity 4, the code for addition would return the following tuples: $[4,0,4]$; $[4,1,3]$; $[4,2,2]$; $[4,3,1]$; $[4,4,0]$. This set of tuples comprises the extensional definition of addition, and the code in UserFunctions provides an intensional definition. Given this code, HR could calculate a table of addition for any new positive integer. Another example of this type of definition is in group theory, where, given a group[1], the concept of the inverse function is intensional as it cycles through members of the group and calculates the inverse for each. Concepts of this type are represented by a data table and categorisation, as above, as well as the intensional definition in *UserFunctions*. We call concepts of type (1) or type (2) *core-concepts*. These are user-given concepts which are used to generate new concepts.

3) The third type of concept in HR is that which HR generates automatically using its production rules and the user-input concepts (see §3.1.2.1). These concepts are represented both intensionally and extensionally, where the extension is considered to be complete if the concept is derived from a concept of the first type, and only relatively complete for a given set of objects of interest if it is derived from concepts of the second type. In the latter case, HR can calculate whether a new entity should be included in the extension of the concept, and, if relevant, calculate the tuple for the new entity. For instance, given concepts *integer* and *divisor*, HR can use its production rules to generate the new concept of *prime number*, where this generated concept is defined intensionally, as the rule: a is a prime if and only if a is an integer with exactly two divisors (shown above). Thus, HR explicitly gives the necessary and sufficient conditions for an integer to be considered prime (in order to generate the concept, HR removes any repeated divisors, and so the number of divisors is calculated as the exact number). HR would also define the concept extensionally by producing a data table for the integers in its theory. Given a new entity x, and told explicitly that it was an integer (which HR would not be able to determine), it could calculate the divisors of the new entity and then the *number* of divisors, and add the entry $x = true$ to the data table for the concept "prime" if x had exactly two divisors, and $x = false$ otherwise. Concepts of this type are represented by a data table, categorisation and definition. We call concepts of this type *developed concepts*, which contrasts the core concepts above.

We adopt this tripartite representation of concepts as it is integral to HR and provides us with

[1] A set which is closed under an associative binary operation with respect to which there exists a unique identity element within the set and every element has an inverse within the set.

a concept generation tool which is adequate for our purposes of modelling Lakatos-style reasoning. However, this leaves us open to the objection that these representations of concept, especially the first case which is purely extensional, are cognitively implausible. If we were to attribute any kind of understanding of the concept "integer" to a human mathematician, or even to a child, then given that they knew the integers $1-20$, if they were then given the entity 21, they would have some intuition that it was also an integer. To this objection we answer that it is not our intention in this project to produce a convincing model of an individual human mathematician, but to model the style of cooperative reasoning that Lakatos described. In chapter 13 we discuss how our model could be improved in this regard.

Concepts such as that of being a prime number, which is either positive or negative for each entity, have arity one. Note that in this thesis we denote the situation where an entity exhibits the characteristics specified by a concept of arity one, by the term 'covered by'. For instance, we would say that the number 3 is *covered by* the concept of a prime number. We may also consider these concepts as sets, and say that the number 3 is *in* the set of primes.

3.1.2 Generation of concepts and conjectures in HR

3.1.2.1 Using production rules to develop concepts

The methodology of using production rules to build up a series of ever more complex concepts comes from Colton's observation that it is possible to gain an understanding of a complex concepts by decomposing it via small steps into simpler concepts. For instance, Colton notes that ring theory is the study of rings, which are themselves groups with an addition operation, and groups are themselves sets, and so on. Colton reverses this idea and has constructed fourteen production rules, which take in one or two concepts and output a new (developed) concept.

Concepts which the production rules take in are represented as any of the three types we described in section 3.1.1, and so are defined in terms of a finite data table and categorisation, and possibly also by a definition. Initially, the concepts used will be core concepts, and so the sorts of concepts developed via the production rules will depend on which core concepts have been input at the start of the session. Starting from different initial concepts and objects of interest, such as the integers $1-20$, or the integers $20-40$, will lead to different concepts and conjectures. This is contrast to standard number theory, which always deals with the same infinite set of natural numbers.

Each production rule works by taking one or two input data tables, and a set of parameterisations, and performs operations on their contents to produce a new output data table. HR then

generates a new definition and categorisation to go with the new data table, thus forming a new concept. The production rules are applied repeatedly, thus the restriction of taking in only one or two old concepts is not limiting, as, for example, the compose production rule could be applied once to two concepts, and then again on the result of this application and a third concept, and so on. The set of concepts, production rules and parameters which have been used in the generation of each concept is recorded as its construction history. The idea behind recording this information is not motivated by human understanding of mathematical concepts, rather is a computational consideration. The information in the construction history of a concept is used to construct different definitions which are readable by a user, the theorem prover Otter, and the model finder MACE.

The production rules are *Arithmeticb*, *Compose*, *Disjunct*, *EmbedAlgebra*, *EmbedGraph*, *Equal*, *Examples*, *Exists*, *Forall*, *Match*, *Negate*, *Record*, *Size*, and *Split*. These are described fully in [Colton, 2002, Chapter 6]. Of these, we only use *Exists*, *Match*, *Negate*, *Size*, *Split* and *Compose*. In this chapter we motivate and explain the *Exists* production rule. In appendix D we also motivate and explain the other production rules to which we refer in this thesis.

The *Exists* production rule is motivated by concepts with existential quantifiers. For instance, a cyclic group is a group for which *there exists* an element of order equal to the size of the group. Another example might be the concept of a continuous function, which is a function $f(x)$ such that for every $\varepsilon > 0$ *there exists* a $\delta > 0$ such that $|f(x) - f(a)| < \varepsilon$ for all x such that $|x - a| < \delta$. The rule produces a summary of the concept it takes in, changing a statement about a property to a statement that such a property exists. For instance, suppose that the old concept is *integers, a, b, c where a = b+c and b and c are both prime*. Note that b and c may not be unique, in which case there will be more than one tuple in the set of examples. Conversely, there may not be two primes which sum to a given integer, in which case the set of examples would be empty.

For integers $30-35$, the examples for this concept are all of the tuples where $30 \leq a \leq 35$ and $a = b + c$ and b and c are both prime. So (omitting repetitions) the examples for 30 would be $b = 7$, $c = 23$, also $b = 11$, $c = 19$, and $b = 13$, $c = 17$. Examples for 31 would be $b = 2, c = 29$. There are no primes which sum to 35, and so the example set for this number would be empty. Applying the *exists* production rule to this concept would result in the concept of integers for which there exist (at least) two primes which sum to the integer. This would result in a list of positives, i.e., integers such as 30 and 31, for which there are examples, and negatives, such as 35 whose example sets are empty. In HR representation, the definition for the old concept would look like:

• $[a,b,c]$: $c + b = a \wedge b + c = a \wedge 2 = |\{d : d|b\}| \wedge 2 = |\{e : e|c\}|$,

and the definition for the new concept would be:

• $[a]$: $\exists bc(c+b=a \wedge b+c=a \wedge 2=|\{d:d|b\}| \wedge 2=|\{e:e|c\}|)$

We show the examples for these two concepts, and an intermediate table, for the integers $1-10$, in table 3.1.

The *exists* production rule is parameterised by a set of column numbers, which denote those columns which are to be kept in the output data table. All other columns are removed. It then also removes any repetitions in the rows. Note that in HR terminology, the set of parameters is represented by angle brackets $<>$. For instance, the *exists* rule might have parameters $<1>$, which would mean that the first column only is to be kept (in PROLOG notation this would be $[X]$); $<2>$ would mean that the second column only is to be kept; $<3,4>$ would mean that the third and fourth columns only are to be kept, etc. Angle brackets are used for computational purposes and denote a list (this is not a set since the order of the parameters is important). This list may contain only numbers, as in the example of the exists production rule, or values such as $<2=1>$, as in the *split* production rule, which means that HR extracts rows from a data table where the second column has the value 1 (in PROLOG notation this would be $[X,1,Y,..]$). Further details, and PROLOG notation are given in appendix D.

The production rules and parameterisations are usually applied automatically according to a search strategy which has been entered by the user. However, it is also possible for the user to force the application of a production rule at any given time, in order to produce a desired concept. This is done by selecting one (or two) concepts in the theory, the production rule and the parameters which determine how the rule applies, and putting this step to be carried out at the top of the agenda. Forcing is a way of fast tracking: finding concepts which may eventually have been found automatically, to be found sooner. This can be done interactively (on-screen) or via a file of instructions. If the guidance is undertaken at the very start of a theory formation session, we can see forcing steps simply as a different way of providing background information to the system. This is because forcing concepts guarantees their existence at the start of the theory in the same way as explicitly providing the concepts guarantees their existence in the theory. We use this functionality in our illustrative examples in later chapters.

An example of using production rules to develop a concept

Starting with the core concept 'divisors of an integer', we apply the production rule *size* and parameterisation $<1>$, which refers to a column for which we should count the number of rows. For instance, the parameterisation $<1>$ means that the number of tuples for each entry in the first column are counted, and this number is then recorded for each entry. This would result in the τ function 'number of divisors of an integer'. We could then apply the *split* production rule and parameterisation $<2=2>$. In this context the parameterisation means

input	
a	*b, c*
1	[]
2	[]
3	[]
4	[2, 2]
5	[2, 3]
5	[3, 2]
6	[3, 3]
7	[2, 5]
7	[5, 2]
8	[3, 5]
8	[5, 3]
9	[2, 7]
9	[7, 2]
10	[3, 7]
10	[5, 5]
10	[7, 3]

→

intermediate
integer
4
5
5
6
7
7
8
8
9
9
10
10
10

→

output
integer
4
5
6
7
8
9
10

Table 3.1: In this example, we start with the concept of *integers a, b, c where a = b+c and b and c are prime*, and apply the *exists* production rule to it with the parameter $< 1 >$. This means that only the first column (of the two columns) is kept, and only the rows with a positive entry. The intermediate table shows the same table as the input column, with only the first column and relevant rows. The output table is the intermediate table with the repetitions omitted, and shows example from 1 - 10 for the concept *integers which can be written as the sum of two primes*.

that we record all rows whose entry in the *second* column is *2*. This would result in the concept 'number of divisors of an integer = 2', i.e., the concept of a prime number. The data tables which correspond to these three concepts are shown in table 3.2. The construction history of the concept *primes* is $\{[$*divisors*, *size* $< 1 >]$, $[\tau$ *function*, *split* $< 2 = 2 >]\}$. Note that the labels *divisors*, τ *function* and *primes* are given by us, to explain the concepts in familiar terms, rather than by HR. HR would label each concept uniquely, in order to store and retrieve it, but this label would have no meaning for a mathematician, reflecting only the order in which it was found.

3.1.2.2 Further parameterisations

We have explained above the parameterisations for the *exists*, *size* and *split* production rules. We further explain the parameterisation for the *match* and *negate* production rules, in order to be able to understand examples to which we refer throughout the thesis. Further details and

divisors	
integer	divisor
1	1
2	1
2	2
3	1
3	3
.	.
.	.
.	.
10	1
10	2
10	5
10	10

$size < 1 > \rightarrow$

τ function	
integer	number
1	1
2	2
3	2
4	3
5	2
6	4
7	2
8	4
9	3
10	4

$split < 2 = 2 > \rightarrow$

primes
integer
2
3
5
7

Table 3.2: In order to generate the concept of a prime number, HR would take in the concept of divisors, represented by a data table for a subset of integers (partially shown here for $1-10$. It would apply the *size* production rule with the parameterisation $< 1 >$ which means that the number of tuples for each entry in the *first* column are counted, and this number is then recorded for each entry. For instance, 1 appears only once in the first column of this data table, 2 and 3 appear twice each, and 10 appears four times. This number is recorded next to the entries in a new data table (the table for the concept τ function). HR then takes in this new concept τ function and applies the *split* production rule with the parameterisation $< 2 = 2 >$, which means that it produces a new data table consisting of those entries in the previous data table whose value in the *second* column was *2*. This is the concept of a prime number.

example applications of these production rules are given in appendix D. The *match* production rule takes parameters $< c_1, c_2, ...c_n >$ where *n* is the arity of the input concept, and where the *nth* entry takes the value of the specified column c_i. For instance the parameters $< 1,2,2 >$ would take a data table with arity three (i.e. there are three columns in the data table) and keep those rows of the data table where the entry in the first column is equal to itself, the entry in the second column is equal to itself, and the entry in the third column is equal to the entry in the second column. Thus this production rule extracts those rows from the data table where the entries in given columns are equal. The *negate* production rule does not require a parameterisation. We denote this by $<>$. *Negate* finds the complement of a concept, i.e. those entities with a certain property, such as *integer*, which do not satisfy the predicate of an input concept, such as *square* (which would result in the new concept *non-square*).

3.1.2.3 HR's conjecture making mechanism

Each time a new concept is generated, HR checks to see whether it can make a conjecture with it. This could be an *equivalence* conjecture, if the new concept has the same data table as a previous concept; an *implication* conjecture, if the data table of the new concept either subsumes or is subsumed by that of another concept, or a *non-existence* conjecture, if the data table for the new concept is empty.

An example of HR's conjecture making mechanism

We show how HR would make the concepts non-square and prime number, and the conjecture that all prime numbers are non-square. Note that some of the steps are shown in more detail in appendix D).

1. supply HR with concepts C_i = [a]: a is an integer and C_{ii} = [a, b, c] : a is an integer & b is an integer & $b|a$ & c is an integer & $c|a$ & $c*b=a$ & $b*c=a$ (i.e. the concepts of integers and multiplication);
2. apply the production rule *match* $<0,1,1>$ to C_{ii}, to get $C1$ = [a, b]: a is an integer & b is an integer & $b|a$ & $b*b=a$;
3. apply the production rule *exists* $<1>$ to $C1$, to get $C2$ = [a]: a is an integer & exists b (b is an integer & $b|a$ & $b*b=a$) (i.e. the concept of a square number);
4. apply the production rule *negate* $<>$ to $C2$ and C_i to get $C3$ = [a]: a is an integer & -(exists b (b is an integer & $b|a$ & $b*b=a$)) (i.e. the concept of a non-square);
5. generate the concept of prime numbers (shown in table 3.2). As part of the theory formation step, HR will check to see whether the data table is equivalent to, subsumed by, or subsumes another data table, or whether it is empty. In this case, it will see that all of its prime numbers are also non-squares, and so conjecture that this is true for all prime numbers. That is, it will make the implication conjecture $(2=|\{b:b|a\}|) \rightarrow$ (-(exists b ($b|a$ & $b*b=a$))).

We show the construction history of this conjecture in table 3.1.

3.1.3 Evaluation of concepts and conjectures in HR

HR evaluates its conjectures and concepts using various ways of measuring interestingness [Colton et al., 2000b], and this drives the heuristic search.

3.1.3.1 Evaluating concepts

[Colton, 2002] suggests that desirable properties of a single concept include being novel, or surprising, having a simple definition (overly complex definitions are difficult to understand), and applying to many entities.

These are formalised as follows:

• The *applicability* of a concept is the number of entities the concept covers divided by number of entities in the theory. This lies between 0 and 1.

• The *complexity* of a concept is the number of concepts in its construction path (including the concept itself).

• The *novelty* of a concept is one divided by the number of concepts (including the concept) which share its categorisation.

• The *comprehensibility* of a concept is one divided by its complexity.

• The *parsimony* of a concept is one divided by the size of its data table, where the size of data table is rows * columns.

Colton argues that a desirable property of the set of concepts in a theory is that between them they develop all of the user-given concepts, in order to avoid redundancy, and that they achieve a high number of categorisations, in order to generate a diverse theory which describes different ways of grouping the objects of interest.

Example of evaluating a concept

Given objects of interest $1 - 10$, HR would evaluate the concept of prime where the derivation is shown above, as follows:

• *applicability* = $4/10$

• *complexity* = 3

• *comprehensibility* = $1/3$

• The *parsimony* = $4 * 1 = 1$

The *novelty* would depend on the rest of the theory.

Note that the motivation behind these criteria is to capture aspects of mathematical concepts which human mathematician would find interesting. For instance, Colton argues that a concept which is very specialised and hence describes a small set of entities is less interesting than one which describes a larger, possibly infinite set. For example, Colton argues, the concept of *groups of order 3* is not very interesting, partly because there is only one such group. Conversely, the concept of *Abelian groups*, for which there are infinitely many, is very interesting. Examples like these inspired the *applicability* measure. However, this is clearly a simplification and in particular, would fail to capture the difference between a *group of order 3* and an *Abelian group* in an infinite domain of groups, as the relevant calculation would be $\frac{1}{\infty}$, which is undefined. Many mathematical domains do describe an infinite number of entities, and therefore this measure would not be one used by human mathematicians. However, the criterion is useful for the purposes of evaluating how interesting a concept is, in a finite domain. A measure which were applicable to infinite domains such as real numbers, might simply measure the size of the set which a concept describes, which would be a subset of the real numbers, rather than being a ratio. The larger the set, the more interesting the concept. Clearly this value could itself be infinite, for instance working within the domain of real numbers, the concept of a *natural number* (describing an infinite set) is highly interesting. The applicability measure is only one measure of interestingness. Likewise, the *parsimony* measure of a concept is motivated by mathematical examples. Colton argues that more parsimonious concepts give more succinct descriptions of a set of entities. To describe the number 10 with the concept of prime numbers, the description is "false", whereas to describe it in terms of its divisors, requires the list $[1,2,5,10]$. The parsimony measure therefore was introduced to estimate how succinct descriptions would be if that concept is used to describe the entities in a theory. Again, this would not be a measure that a human mathematician would use, since a human mathematician would not represent examples in a data table. However the measure goes some way towards capturing the value of succinctness. Finding measures of interestingness which could apply to infinite domains, measured cognitively plausible representations of mathematical concepts and could be calculated by human mathematics, would be a different, and challenging project.

3.1.3.2 Evaluating conjectures

Desirable properties of a conjecture include involving concepts with (a) high applicability and (b) high comprehensibility, and that it be surprising. These are formalised as follows:

• The *applicability* of a conjecture is the proportion of entities that the conjecture discusses, where a conjecture "discusses" an entity if the left hand concept of an implication or equivalence conjecture covers the entity. For instance, an equivalence or implication conjecture about

primes "discusses" the number 7, but not the number 4, and a non-existence conjecture does not discuss any entity. The applicability of an equivalence conjecture is the applicability of the concepts which are hypothesised to be equivalent. The applicability of an implication conjecture is the applicability of the antecedent concept). The applicability of a non-existence conjecture is zero.

• The *comprehensibility* of a conjecture is the reciprocal of the number of distinct concepts which appear in the construction path of the concepts discussed in the conjecture. Comprehensibility is not defined for non-existence conjectures.

• The *surprisingness* of an equivalence or an implication conjecture is the number of concepts which appear in the construction path of one concept, but not both. The surprisingness of a non-existence conjecture is measured by the applicability of its parent concept.

An example of evaluating a conjecture

We show the construction history of the conjecture 'all primes are non-square' in figure 3.1, which we described above in §3.1.2.3. HR would evaluate this conjecture as follows.

• *applicability* = $4/10$

• *comprehensibility* = $1/8$

• *surprisingness* = 8

Note that all measures are normalised in HR.

HR also uses third party automated reasoning software in order to prove or disprove its conjectures. We currently make no use of this.

3.1.4 Representing a theory in HR

Theories are represented in HR by a Java class containing fields which are initialised at the start of a session, such as whether to extract implication conjectures from equivalence conjectures, and which productions rules to use. The theory class also contains lists of concepts, categorisations, conjectures and entities, which is added to during the theory formation. It also includes methods for measuring the interestingness of a concept, and the method for performing a theory formation step. When we refer to "theory" throughout this thesis, we refer to this class; in particular to the set of concepts, conjectures and entities.

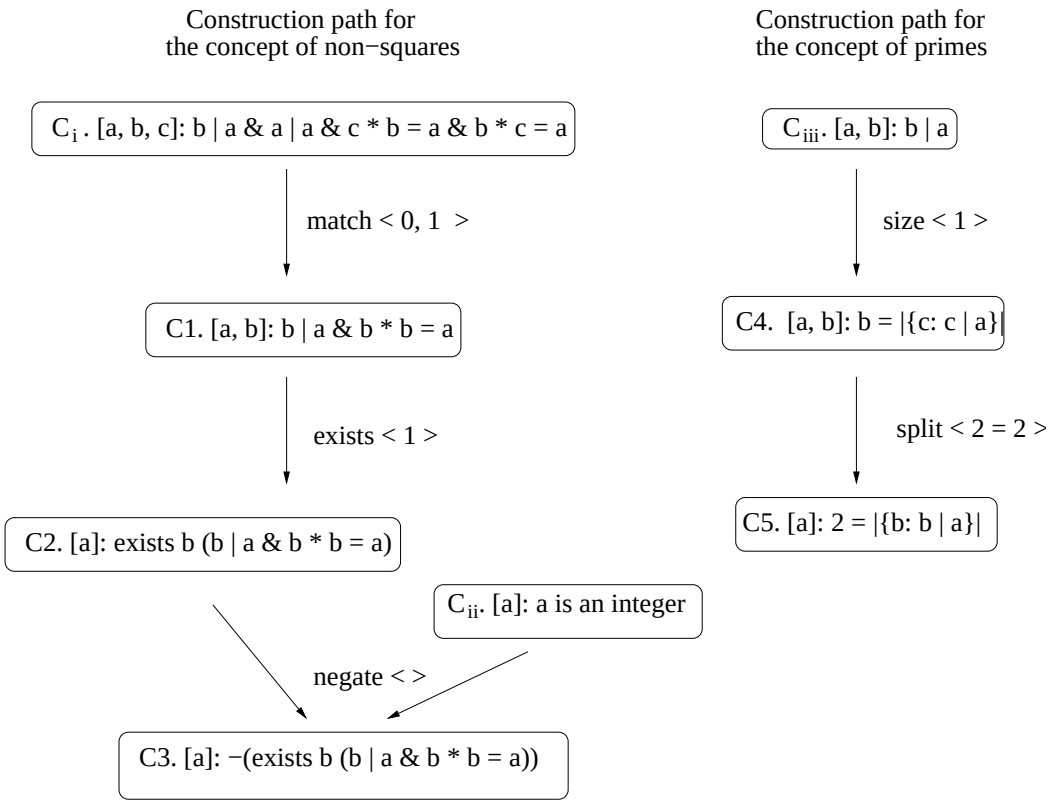


Figure 3.1: The construction path of the concepts in the conjecture *all primes are non-square*, where $b|a$ means that *b divides a*, $a * b$ means *a multiplied by b*, "&" is the logical connective *and*, and $|\{b : b|a\}|$ means *the size of the set of elements b such that b divides a*. The arrows between nodes represent the derivation which takes place when the given production rule and parameterisations are applied to an "old" concept, to get a new concept, with the direction of the arrow denoting going from old to new.

3.1.5 Using HR in a multiagent system

In Colton et al. [2000a] Colton describes a multiagent approach to concept formation in pure mathematics, using the HR program. Four copies of HR were employed as agents in an architecture where they communicated the most interesting concepts they invented.

3.2 Multiagent Systems

Since Lakatos presented his work as a dialogue where different students had different beliefs and motivations, and mathematical theories were formed via disagreement and negotiation, agents are an obvious metaphor for a computational representation of his theory. Additionally,

the dynamic nature of the environment of conjectures and entities which is open and continually changing, suggests that an agent-based solution is appropriate. In this section we consider characteristics of multiagent systems.

3.2.1 Environment

Russell and Norvig [1995] provide the following classification of environments, which are discussed in [Wooldridge, 2002]:

• accessible versus inaccessible – the extent to which it is possible to gain complete, accurate, up to date information about the environment;

• deterministic versus non-deterministic – the extent to which any action has a single guaranteed effect;

• static versus dynamic – the extent to which an environment changes in ways which are beyond an agent's control; and

• discrete versus continuous – whether there are a fixed, finite number of actions and percepts in an environment or not.

Environments may be a hybrid of these, for instance, an environment might consist of a finite number of discrete actions and percepts, and an infinite number of continuous actions. One example of this would be a human environment, in which positions, temperature, distances etc. vary continuously, while topological relations, such as whether objects are convex or not, connected or not, touching or not, etc., vary discretely. A further difference concerns whether objects in an environment can be described by vectors or lists of uniform complexity, or whether structural descriptions are necessary, as would be the case for representing sentences, town maps, family trees, etc..

3.2.2 Agent design

Wooldridge [2002] makes the micro/macro distinction, which distinguishes the design of the agents from that of the society. He addresses two problems:

(i) building an agent which is capable of independent, autonomous action, in order to carry out tasks which we assign to it, and

(ii) building agents which are capable of interacting with other agents, who may not share the same goals, in order to carry out tasks which we assign to them.

These problems are known respectively as agent design, which we discuss in this section, and society design, which we discuss in the following section.

There is some controversy over what constitutes an agent, but a widely accepted criterion is that it be autonomous. Wooldridge defines an agent as "a computer system that is *situated* in some *environment*, and [that] is capable of *autonomous action* in this environment in order to meet its design objectives" [Wooldridge, 2002, p. 15]. This is a very general definition, which, as Wooldridge points out, includes control systems, such as thermostats, and software demons, such as background processes in the Unix operating system. He goes on to highlight capabilities that we might expect an *intelligent* agent to have: reactive, proactive and social capabilities. That is, the ability to perceive and react to one's environment, exhibit goal-directed behaviour, and interact with other agents or humans, in order to satisfy its design objectives. Wooldridge suggests that the ability to influence and be influenced by one's environment is also important. Further attributes which agents may have include learning, and an ability to deliberate between different courses of action.

Dennett [1971] examines the concept of a system whose behaviour can be explained and predicted by ascribing to the system mentalistic concepts such as belief, desire, intention and expectation. He develops the intentional stance, which is used to predict behaviour by ascribing to a system the possession of certain information (sometimes called beliefs), and supposing it to be directed by certain goals (sometimes called desires), and then working out the most reasonable course of action. This stance is contrasted with the *design stance*, where one predicts the behaviour of a system, based on the purpose it is intended to fulfil, assuming that everything functions as it was intended to. The intentional stance also contrasts with the *physical stance*, where behaviour is predicted based on knowledge of the laws of nature and the original configuration of a system. Dennett argues that some systems, for instance chess-playing computers, are inaccessible to prediction from the design stance or the physical stance. He takes the pragmatic approach that if it is useful to endow agents with mental states, in terms of predicting and explaining, then we should do so.

Wooldridge [2001] discusses belief-desire-intention architectures. These are rooted in the field of practical reasoning, which involves the processes of deciding what to achieve and how to achieve it. Deciding what to achieve involves generating the options available, choosing between them and committing to one. The chosen course of action then becomes an *intention*. Characteristics of intentions include believing that there is a good chance of achieving them (we do not intend to do something which we know that we cannot do), acting upon them (mak-

ing a reasonable attempt to achieve them), and the intentions persisting (rather than giving up at the first hurdle). This latter characteristic, however, is to be balanced with dropping an intention if it becomes unfeasible. Achieving the right balance between reconsidering intentions sufficiently often to drop them when necessary, and not wasting resources on constantly reconsidering, rather than achieving intentions, is called the problem of balancing pro-active, goal directed against reactive, event driven behaviour.

Beaudoin [1995] elucidates goal processing in autonomous agents from a design stance, and identifies this type of problem as *meta-management*. He refers to Hayes-Roth [1985] who worked on the "control problem", i.e., how a system decides which of a set of currently possible computational actions it should perform next. Beaudoin suggests a system he calls "heuristic meta-management", where an agent which is capable of meta-management engages its meta capabilities at timely junctures where a shift in processing is required. Such timely junctures include: oscillation between decisions (when one keeps changing one's mind); ongoing disruption (where an insistent goal has been postponed or rejected but nevertheless keeps reappearing); high busyness (in which case it is necessary to prioritise goals); digressions (when a goal has been scheduled for deliberation, but while deliberating, the agent loses sight of the original goal), and maundering (managing a goal for some length of time without ever having made the meta-management decision to do so). Beaudoin goes on to ask empirical questions such as "What species are capable of managing their own management processes?", "What are the mechanisms that a given class of organisms has for meta-management?", and "What formalisms best match their language?"[Beaudoin, 1995, p 72].

3.2.3 Society design

Wooldridge [2002] argues that in multiagent systems we are interested in how cooperation can emerge within societies of agents which are self-interested; what sorts of common languages agents can use to communicate to each other and to humans; how self-interested agents can recognise when their interests conflict, and how they can resolve this, and how agents can coordinate their activities so as to cooperatively achieve their goals.

Huhns and Stephens [2001] describe environments in which agents interact productively. They discuss communication protocols which enable agents to exchange and understand messages. For instance, an agent may propose a course of action, accept a course of action, reject a course of action, retract a course of action, disagree with a proposed course of action, or counter propose a course of action.

Huhns and Stephens [2001] also discuss interaction protocols, which enable agents to have conversations. For instance, one agent may propose a course of action to a second agent. The

second one then might evaluate the proposal and either send acceptance of it, rejection of it, disagreement with it, or a counterproposal to it. The first then evaluates what it receives and reacts accordingly, and so on.

Further work which is relevant to communication and interaction protocols, on ambiguity, argumentation and negotiation is outlined in §6.8.

3.2.4 Argumentation-based negotiation

Jennings et al. [1998] emphasise the importance of negotiation in multiagent research, and outline an informal framework describing its key features. They divide negotiation issues into *protocols* (rules which govern interaction), *objects* (the range of issues over which agreement must be reached) and *Agents' Decision Making Models* (the way in which an agent follows the protocol to achieve its objectives).

Agents move through the space of possible agreements (in our case all the candidate definitions for a concept), defining their own spaces of acceptable points. Negotiation works by agents suggesting points in the space which are acceptable, and evaluating each point suggested. This ranking may change during negotiation, as agents are persuaded that a point is valid. The way in which they rate points may also change. A minimal requirement is that agents be able to propose some part of the agreement space as acceptable, and can respond to other agents' suggestions. A more efficient model would give agents the capability to explain *why* they are rejecting/proposing a certain point. This might include rejecting a proposal but stating which aspects were considered good, a *critique*, or making a *counter-proposal* in response to a proposal. Such a model might include *justifications* – in which the agent states its reasons for making a proposal, or *persuasion* – in which an agent tries to change another's agreement space or rating over the space. These arguments help to support an agent's stance.

Jennings et al. [1998] state that an agent capable of argumentation-based negotiation must have a mechanism for:

- communicating proposals and supporting arguments;
- generating proposals;
- assessing proposals and arguments; and
- responding to proposals.

Kind of AI	Main Activity	Result
Applied AI	Apply AI techniques to real life problems	Product
Cognitive Science	Formulate theories Build models	Theory of intelligence supported by a computational model
Basic AI	Explore AI techniques	New/improved techniques More knowledge of techniques

Table 3.3: The three kinds of AI, outlined in Bundy [1990]

3.3 Methodology

Our task is to provide a computational representation of Lakatos's methods and then use our representation to evaluate the methods, as well as evaluating our representation itself. This evaluation has two main forms, which correspond to our two motivations. Our first motivation is the cognitive science motivation of modelling, evaluating and improving Lakatos's theory of discovery and justification. Our secondary motivation can be seen in terms of Bundy's 'basic AI' [Bundy, 1990]. The aim of basic AI is to develop, explore or improve useful AI techniques, where 'technique' includes algorithms, ways of representing knowledge, architectures and methods of eliciting knowledge. The process of exploring AI techniques may lead us to discover new techniques, and their properties and interrelations. Our two motivations can be seen as the second and third kinds of AI shown in table 3.3.

In order to evaluate our work from the cognitive science perspective, we use criteria suggested by Thagard, Sloman, and Popper, described below. These consist of consilience, simplicity and analogy (§3.3.2); being definite, general (but not too general), able to explain fine structure, non-circular, rigorous, plausible, economical, rich in heuristic power, and extendable (§3.3.3); not being *ad hoc*, and being rich in content (§3.3.4). These criteria overlap to a certain extent, and are used by humans. We furthermore evaluate our work using Lakatos's criteria for evaluating scientific research programmes (§3.3.5).

In order to evaluate our work from the basic AI perspective, we use the criteria for automatic evaluation from [Colton, 2002]. These consist of applicability, complexity, novelty, comprehensibility, and parsimony for concepts, and applicability, comprehensibility, and surprisingness for conjectures (§3.1.3). We furthermore test the application of Lakatos's methods to automated theorem proving by testing to see if they can be used to find meaningful modifications to non-theorems, where 'meaningful' is defined as a theorem for which a proof can be found in the required time.

3.3.1 Evaluating theories within the philosophy of science

The method of evaluating models in computational philosophy of science is not well established. In the philosophy of science, ideas on evaluating a theory or hypothesis came before ideas on how the hypothesis is discovered. In computational philosophy of science it appears to be the other way around. That is, ideas on how to generate, or discover programs which model scientific progress have come before ideas on how we should evaluate these programs. Clearly much work in the philosophy of science examines what makes a scientific theory good, for example [Popper, 1972]. Yet, although much of philosophy concerns evaluating different arguments, there seems to have been little explicitly written about how to evaluate meta-theories, i.e., philosophical theories about scientific theories. If the field of computational philosophy of science is to progress, there has to be discussion and agreement on the criteria by which we judge computational theories (philosophical theories which are at least partially derived using computational techniques). Additionally, these criteria must be formal enough that comparative claims can be supported and progress measured. Unsurprisingly, this mirrors the situation in the machine creativity field, in which attempts are being made to find a framework which is both practically useful and theoretically feasible, i.e., formal but not oversimplified (for example [Pease et al., 2001], [Ritchie, 2001], [Ritchie, 2005]). Given this situation, we have followed [Colton, 2002] and adopted a shotgun approach, using criteria from different sources to evaluate our computational interpretation of Lakatos's theory. These criteria are taken from computational philosophy of science (§'s 3.3.2 and 3.3.3) as well as traditional philosophy of science (§3.3.4).

Part of our motivation behind implementing Lakatos's theory is to provide a way of evaluating, as well as improving upon it. In chapter 11 we discuss how our model helps us to evaluate Lakatos's theory according to criteria suggested by Thagard, Sloman Popper, and Lakatos (§'s 3.3.2, 3.3.3, 3.3.4 and 3.3.5 respectively) and argue that by the same criteria we have improved Lakatos's theory.

3.3.2 Thagard's criteria for evaluating theories

Thagard [1993] suggests criteria for evaluating explanatory theories. These are intended for evaluating scientific, rather than philosophical theories, and have been extracted from studying examples of scientific theories. However, Thagard also claims that they can be used to determine the best explanation in metaphysical theories [Thagard, 1993, p.99]. The criteria are consilience, simplicity and analogy. Consilience is a measure of how many observables a theory explains, and the variety and importance of the facts explained. The notion of simplicity

is a way of constraining consilience by ensuring that the theory is not *ad hoc*. This means that the theory explains more than just the data which it was introduced to explain, i.e., it is not fine-tuned. Hence the first and second criteria need to be taken in conjunction with each other.

3.3.3 Sloman's criteria for evaluating theories

Sloman [Sloman, 1978, p. 50 - 53] sets out criteria which he claims can be used to judge 'theories which purport to explain possibilities', including scientific possibilities as well as theories in human sciences. We set out his criteria below. They are divided into a necessary criterion for something to be considered a theory at all, and a much longer set of criteria for something to be considered a *good* theory.

T is a theory if:

1. it explains a range of possibilities, i.e., the possibilities are validly derivable from T, according to criteria for validity generated by the semantics of the language used for T.

T is a *good* theory if (1) is satisfied, and:

2. it is definite (there is a clear demarcation between what it does and what it does not explain);
3. it is general (it should explain many significantly different possibilities, preferably some which were not known about before the theory was invented. However it should not explain too many possibilities which have not been shown to exist, i.e., it should not be *too* general);
4. it accounts for fine structure (the descriptions or representations of possibilities generated by T should be rich and detailed);
5. it is non-circular (it should not assume that which it purports to explain);
6. derivations from it are rigorous (it should be clear how the possibilities which T can generate are generated, or derived, from T);
7. it is plausible (the assumptions made in T should not contradict known *facts* – although they may contradict widely held *beliefs*);
8. it is economical (it does not include assumptions or concepts which are not necessary to explain the possibilities which it explains);

9. it is rich in heuristic power (the components of the theory, i.e., the assumptions, concepts, representation language and way in which possibilities are generated, should be such that the detection of errors and gaps, design of problem solving strategies etc. is easily manageable); or

10. it is extendable (it should be possible to embed the theory in an improved enlarged theory which explains further possibilities or has a higher degree of fine structure).

These criteria provide us with ways by which we can compare two philosophical theories. Sloman also considers the further criteria that a theory enables us to control or predict phenomena. He argues that these criteria are often over-emphasised, for instance the theory of evolution is arguably one of the most important scientific theories, but its power lies mainly in explaining possibilities, rather than controlling or predicting biological developments.

3.3.4 Popper's criteria for evaluating theories

Following Tarski, Popper [1972] suggests that we divide the universal class of all statements into true and false, T and F. He claims that the aim of science is to discover theories (explanations) whose content covers as much of T and as little of F as possible, where the content of a theory is the set of all statements logically entailed by it. This set may also be divided into true and false statements (the theory's truth and falsity content). A good theory should suggest where to look, i.e., new observations which we had not thought of making before.

This is comparable to the situation described in [Ritchie, 2001], in which we divide the universal class of all basic items in a domain into good and bad, V and V'. If we describe the content of a program as its output set O which may be divided into good and bad artefacts, then we can claim that one aim within AI is for a program to generate as much of V (and as little of V') as possible. A good system should suggest new areas of the search space, i.e., find artefacts which we had not thought of generating before. If we accept this analogy then Popper's criteria for evaluating theories sheds light on our criteria for evaluating programs.

Popper sets out two criteria for a satisfactory theory (in addition to it logically entailing what it explains). Firstly it must not be *ad hoc*. By this is meant that the theory (explicans) cannot itself be evidence for the phenomena to be explained (explicandum), or vice versa. For example if the explicandum is 'this rat is dead', then it is not enough to suggest that 'this rat ate poison' if the evidence for it having done so is that it is now dead. There must be independent evidence, such as 'the rat's stomach contains rat poison'. The opposite of an *ad hoc* explanation then, is one which is independently testable. Secondly, a theory must be rich in content. For example a

theory which explains phenomena other than the specific phenomena it was designed to explain has a much richer content, and is therefore of greater value, than one which is less general (the principle of universality).

Applying these criteria to our programs, if we see a program (K) as the theory and the set of artefacts we wish to generate (I) as the phenomena to be explained, then we are interested in the independent testability of K and the richness of its content. A program which has been carefully tailored in order to produce very specific artefacts cannot be claimed to be a good program on the grounds that it produces those artefacts. There must be independent grounds for its value, such as also generating other valuable artefacts. Within the programming analogy, this is clearly connected to the richness of content criterion; the more valuable artefacts outside of I and fewer worthless artefacts a program generates, the better that program is.

3.3.5 Lakatos's criteria for evaluating theories

Lakatos [1970] attempted to salvage some of Popper's falsification methodology, which had suffered in the light of Kuhn's analysis of paradigm change. However, he thought that Popper's focus on the relationship between theory and observation in his falsification methodology was too simplistic, arguing instead that a methodology must take into account the *structure* of a theory. In his account he developed the notion of a scientific research programme, which comprises a *hard core* and a *protective belt*. The hard core consists of the defining characteristics of a programme: these are very general theoretical hypotheses which form the basis of the programme. If a scientist were to reject or modify these hypotheses, then essentially he or she would be abandoning the research programme (this is similar to Kuhn's notion of a paradigm shift). The protective belt consists of explicit auxiliary hypotheses and assumptions which are less central to a research programme; these could be rejected or modified without serious repercussions to the research programme. If a hypothesis from the hard core appears to have been falsified, then in order to remain in the same research programme, appropriate changes or additions would be made to the protective belt, rather than to the hypothesis directly. The *positive heuristic* of a research programme indicates *how* the protective belt can be altered in order to protect and extend the predictive and explanatory power of the hard core. The *negative heuristic* states that the hard core must remain unchanged. Another constraint is that modifications made to hypotheses in the protective belt must be independently testable.

Lakatos uses these ideas to show how we can evaluate work done within a research programme, and to evaluate competing research programmes. He also used them as demarcation criteria between science and non-science. Research programmes can be evaluated according to whether they are *progressive* or *degenerative*. A programme is progressive if it satisfies two criteria:

firstly if it comprises a coherent hardcore which involves a definite mapping out of predictions and future research, and secondly if it, at least occasionally, leads to the discovery of novel phenomena. These also serve as demarcation criteria. A programme is degenerative if it is gradually coming undone, and there are no recent novel predictions to its name. It is difficult to evaluate a research programme except in retrospect, as we can never be sure that a new discovery is not just around the corner.

One way of evaluating two rival theories is to look at Lakatos's falsification criteria, outlined in [Lakatos, 1970, p. 116]. A theory T is falsified if and only if another theory T' has been proposed with the following three criteria:

i) T' predicts novel facts, i.e., phenomena which was not predicted by T (this is a sign of a theoretically progressive research programme);

ii) T' explains all of the confirming instances the T explained; and

iii) some of the excess content of T' is corroborated (this is a sign of an empirically progressive research programme).

Note that T and T' may share the same hard core.

Lakatos differed from Sloman, Popper and Thagard in that the criteria he identified as a good scientific theory were intended to be used for demarcation criteria, i.e., they could help to distinguish science from non-science. Conversely, while Sloman, Popper and Thagard set out criteria of a good scientific theory, they also commented that the same criteria could be used to evaluate non-scientific theories.

3.3.6 Summary of criteria

We have considered criteria suggested by Thagard [1993], Sloman [1978], Popper [1972] and Lakatos [1970]. The first three consist of Thagard's notions of consilience, simplicity and analogy (§3.3.2); Sloman's criteria that a theory *(i)* explain a range of possibilities, and a good theory is *(ii)* definite, *(iii)* general, *(iv)* able to explain fine structure, *(v)* non-circular, *(vi)* rigorous, *(vii)* plausible, *(viii)* economical, *(ix)* rich in heuristic power, and *(x)* extendable (§3.3.3); and Popper's criteria of being independently testable, and rich in content (§3.3.4). These three sets of criteria overlap in the criteria shown below. A good theory should:

- be as general as possible (Thagard's notion of consilience, and Sloman's third generality criterion);

- explain more than it set out to explain (Thagard's notion of simplicity, and Popper's richness in content criterion);
- not assume what it sets out to explain (Sloman's fifth non-circularity criterion, and Popper's independently testable criterion).

Since there is no ready made integrated, coherent, well-motivated and widely accepted set of criteria relevant to evaluating our work, we focus on these criteria in our philosophical evaluation of our project, in chapter 11. In particular, we focus on the first and second of these criteria. In a different section of the philosophical evaluation chapter, section 11.9, we discuss our project with reference to Lakatos's philosophy of science.

3.4 Summary

In this chapter we have introduced the HR system [Colton, 2002], and described how it represents, generates and evaluates objects of interest, concepts and conjectures. This is necessary in order to understand the theory formation aspect of HRL. We are aiming to work within the context of multiagent systems, and have discussed when an agent-based solution is appropriate, environment, agent design and society design in this context. Finally, we have outlined our methodology and described the criteria which we use to evaluate our project.

Chapter 4

Overview of HRL

In this chapter we outline our system, HRL, drawing on the background material given in chapter 3. The HRL system incorporates HR, which is named after mathematicians Godfrey Harold Hardy (1877 - 1947) and Srinivasa Aiyangar Ramanujan (1887 - 1920), and extends it by modelling Lakatos's methods and enabling dialogue between multiple copies of HR. Thus, the name HRL reflects our debt to HR, and highlights the importance of the philosopher Imre Lakatos (1922-1974) and the deep influence of his work on our system.

HRL has two main strands: the methods that we have implemented, and the agent architecture which enables dialogue. Chapters 5 to 9 contain details of the former, and in this chapter we discuss the latter. We also detail some extensions to HR which we have made.

Following the discussion in [Lakatos, 1976], HRL models a number of student agents and a teacher agent, where each agent has a suitably configured copy of HR, and starts with a different database of objects of interest to work with, and different interestingness measures. Our system is written in Java, which was a natural language to use since HR is in Java, and object oriented languages are an obvious metaphor for agent architectures. The agents in HRL use sockets to communicate. The number of agents is flexible, determined by the user, but in each case, one agent is meant to represent a teacher, with the other agents being students.

4.1 Environment

There are two stages in HRL: an independent work phase in which students form theories without interacting, and a discussion phase in which they discuss and refine their most interesting conjectures. The independent work phase gives the students time to generate conjectures

and concepts, as described in §3.1, which are then communicated and used during the discussion phase. Each agent, therefore, has its own individual theory. In addition there is the group's theory, which is generated from the interaction of the students and the teacher, consisting of every message which has been sent, including conjectures, concepts, proof schemes, entities, and proposed modifications. Each agent records the group's theory in a list called *group discussion*. The combination of all of the individual theories and the group's theory is the environment of the agency, where by *agency* we mean the collection of students and the teacher which have been set to run at the start of a session.

The agents only have direct access to their own individual theories and the group theory. Therefore each agent has partial access to the environment, with the other students' theories hidden. It is a non-deterministic environment since the order of the agents' messages cannot be predicted and the order in which they are received influences the discussion. However, in all other aspects it is a deterministic environment. A single agent has full control over its own theory. At the end of each discussion phase, a student will look through what has been said and take what it considers to be interesting into its own theory to further develop. Clearly, one student has no control over another student's theory, although it can influence the theory by making a contribution to the group theory which the second student evaluates as interesting enough to develop further (i.e. the interestingness evaluation is over a user given threshold). The agents therefore are able both to influence and be influenced by their environment.

4.2 Agent design

Each agent has a copy of HR, and starts with a different database of objects of interest and different interestingness measures. Making the evaluation subjective agrees with Larvor's point[1] that Lakatos considered mathematics to be a matter of taste: "Why not have mathematical critics just as you have literary critics, to develop mathematical taste by public criticism?" *Gamma* — [Lakatos, 1976, p. 98]. Each agent, therefore, is capable of independent, autonomous action, in order to carry out the task which we assign to it, which is to form an interesting theory about the data with which we provide it. Agents are reactive in the sense that they can react to their environment, since they can perceive the group theory and react to it. For instance, an agent might react by modifying a faulty conjecture or agreeing to reject a certain entity as a monster. They exhibit goal-directed behaviour by achieving sub-goals of modifying faulty conjectures or proof schemes in order to form an interesting theory, and are therefore proactive. They also interact with other students in the agency and the teacher, again, with the aim of forming

[1] Personal communication.

an interesting theory. Therefore the agents in HRL are intelligent according to Wooldridge's account (§3.2.2).

Agents in HRL are currently unable to deliberate between different courses of action (see §13.3 for ideas on how the latter attribute may be implemented).

The task of the agents in our architecture is to develop interesting concepts, conjectures and examples, and to react accordingly to the introduction of counterexamples to a false conjecture. In terms of the beliefs, desires and intentions approach outlined in § 3.2.2, the agents' *beliefs* consist of the data input to their copy of HR. Their *desires* include building an interesting theory, accepting, modifying or rejecting concept definitions, conjectures and proof schemes (depending on the proportion of their examples they hold for, and how interesting they are according to the interestingness measures of the agent). The *intentions* of the agents are to perform specific Lakatos methods or parts of the methods, e.g., to find a concept which covers specific examples in order to perform piecemeal exclusion.

4.3 Society design

HRL is an architecture of a group of equal-status agents called students. The architecture also contains a teacher agent, which has special status. The problem, or design objective, is to model the social process of concept, conjecture and proof refinement in the face of conjectured general properties and counterexamples to them, as expounded by Lakatos [1976]. The role of the teacher is to evaluate the messages it receives in the discussion and to set an agenda for discussion. Thus if one conjecture modification is becoming uninteresting (according to the interestingness criteria of the teacher), the teacher can tell the students to focus on another conjecture. Note that it would be possible to use Lakatos's methods in a different context, for instance by changing the flow of control. We describe one way in which we do this, in chapter 12.1.

Communication protocol:

The agents communicate by sending concepts, counterexamples and conjectures when the teacher requests them, and negotiating about concept definitions. Specifically, agents communicate by sending a request or a response.

Requests can be requests to do something, such as "work independently for twenty theory formation steps", or "modify faulty conjecture *C*"; requests to send something, such as "send counterexamples to conjecture *C*", or "send a concept to cover counterexamples $[x,y,z]$"; or

requests to negotiate a concept definition, such as "entity *x* should not be considered an example of concept *C*". Responses include conjectures, concepts, counterexamples, proof schemes, modifications and negotiations.

Interaction protocol:

Interaction usually works alternately between the teacher sending a request and the students all sending a response to the request. The exception to this is if a student responds to another student's response. For instance, one student may send an entity as a counterexample to a conjecture, and another student may respond by requesting that the entity be barred as a monster. The teacher will then make a further request, and the cycle continues. An example interaction is below.

1. The teacher requests that the students work independently for twenty theory formation steps and then send an interesting conjecture.

2. The students comply and all send a conjecture.

3. The teacher sorts the conjectures into an agenda for discussion. It sends a request for modifications to the first conjecture on the agenda.

4. Each agent looks at the examples and counterexamples it has for the conjecture. If it has any counterexamples then it attempts to modify the conjecture and sends its modification.

5. The teacher sorts the modified conjectures into the agenda and sends a request for modifications to the next conjecture on the agenda.

The cycle of interaction is shown in figure 4.1.

4.4 Extensions to HR

Two new types of conjecture

Every Lakatos method except strategic withdrawal requires counterexamples to a conjecture. In order to get these, HR has to be able to make conjectures which are known to be false. We have addressed this in two ways. Firstly, since we distribute the data amongst multiple agents, each agent is more likely to make a false conjecture than if we ran the system with a single agent, as each conjecture is based on a smaller subset of the data available. Secondly, we have enabled HR to generate *near-equivalence* and *near-implication* conjectures. These are equivalence and implication conjectures which hold for *x%* of its entities, where *x* is less than 100%, and is specified by the user.

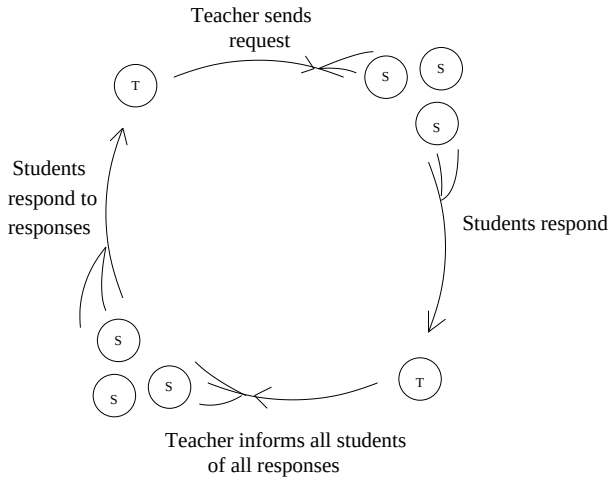


Figure 4.1: The interaction protocol of HRL

Implementing near-conjectures raises the question of whether negative examples should be considered supporting examples for a near-conjecture. That is, should the numbers 4, 6, 8 and 10 in figure 4.2 be considered supporting examples of the conjecture that almost all prime numbers are odd, or should they be considered irrelevant? Hempel's paradox in [Hempel, 1945], in which he showed how the principle of scientific induction can be counter-intuitive, is relevant here. Hempel demonstrated his point with the example hypothesis that "all ravens are black". According to scientific induction, the likelihood of this being true rises slightly with every new black raven that we see. However, the statement "all non-black-things are non-ravens" is logically equivalent. The likelihood of this being true rises slightly with every new non-black-thing which is not a raven. This suggests that if we see, for instance, a red apple, then this is evidence for the hypothesis that all ravens are black.

In our implementation, the calculation for near-equivalence conjectures between concept P and concept Q includes entities which are not positive for either concept (but are not counterexamples to the conjecture). Therefore, supposing that HR is given the integers $1-10$ as its objects of interest, then, in the example in figure 4.2, $prime \leftrightarrow odd$, the near-equivalence conjecture would score 70%, as there are three counterexamples to this conjecture; 1, 2 and 9, and ten entities in total. This is in the spirit of machine learning algorithms, which consider the negatives as well as positives, as a measure of confidence in the conjecture. However, this could include the case of two concepts $P(x)$ and $Q(x)$, each with very few examples, none of which are shared and the conjecture $P(x) \rightsquigarrow Q(x)$, if there are sufficient examples in the theory, which is clearly undesirable.

For near-implication conjectures therefore, we do not consider negative examples, i.e., entities in neither $P(x)$ nor $Q(x)$). That is, the conjecture $prime \rightarrow odd$ would not score 90% (which

we might expect as there is only one counterexample out of ten entities: 2). Similarly, the near-implication conjecture *odd* $\rightarrow$ *prime*, with counterexamples 1 and 9, would not score 80%. This is because when calculating the proportion of entities for which a near-implication conjecture holds, we consider only entities which are positive for the left hand concept, so the near-implication scores would be $3/4$ and $3/5$, i.e., 75% and 60% respectively.

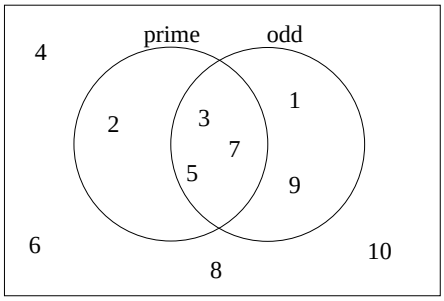


Figure 4.2: A Venn Diagram representation of the conjecture that for all x, $prime(x) \leftrightsquigarrow odd(x)$, where x is an integer between 1 and 10

A new production rule

As part of our implementation of the method of exception-barring, we have implemented a new production rule, *entity-disjunct*. This takes in an object of interest concept, for instance, *number*, and a set of entities as a parameter list, where the entities are all the same type as the object of interest concept, for instance, $[1,2]$. It returns a concept which is a disjunction of the entities, for instance, the concept of being *either the number 1 or the number 2*. We describe it in more detail, and give our motivation behind it, in §7.7.

A new measure of interestingness

Since some conjectures are false, we have reflected this in a new interestingness measure. This is intended to capture degree of belief, or confidence in a conjecture, and is measured by comparing the number of supporting examples of conjecture with the number of counterexamples. The *plausibility* of a conjecture is:

$(e-c)/e$, where e is the number of entities the conjecture discusses, from a database, and c is the number of counterexamples, in a database, to the conjecture.

For instance, the plausibility of the conjecture that all prime numbers are odd, given a database of numbers $1-20$, is $(8-1)/8 = 0.875$.

A new type of theory constituent

In HR, elements of a theory are represented as *theory constituents*, which is a Java class. Subclasses of theory constituent are concept, conjecture and entity. We have extended this by

adding another subclass: *proof scheme*. We need this class for our implementation of the method of lemma-incorporation, which modifies a faulty proof. Although HR was already interfaced with the theorem prover Otter [McCune, 1990], we could not use output from Otter, as we needed HR to work with informal, flawed proofs. Each proof scheme has a global conjecture (the conjecture which is 'proved'), a vector of conjectures in the proof, and a proof tree which specifies how the conjectures fit together to prove the global conjecture. Details of the proof scheme class are given in §8.3.

Pseudo theory-constituents

As a conjecture is surrendered or refined, the question arises as to whether to continue to store the previous versions. Likewise, when an entity is rejected as an example of a concept, via the method of monster-barring, the question arises as to whether to continue to store this entity. We have resolved this issue by creating *pseudo-conjectures* and *pseudo-entities*. These are identical to conjectures and entities in every way, except that they are stored in separate lists of *pseudo-conjectures* and *pseudo-entities* in the theory, rather than in the usual lists of conjectures and entities. Thus, they remain in the theory but cannot influence it in any way. Removing an entity from the list of entities, for instance, ensures that it will no longer count as a positive or negative example of conjectures. This affects the search space as well as the interestingness evaluation of new conjectures.

Our justification is that theory-constituents are not forgotten if consensus is arrived at within the agency that they do not count in some way. These pseudo theory-constituents do not currently affect a theory once they have been allocated "pseudo" status. In future versions, however, we may use elements in a pseudo theory-constituents list. For instance, a student agent could re-evaluate a pseudo-conjecture if it was later agreed that some, or all, of the counterexamples to it did not count. Likewise, a student agent might re-evaluate a pseudo-entity which had been considered to be a monster, but which is found to be useful once the theory is more developed.

4.5 Parameter values and their selection

Within HRL we have continued Colton's methodology of allowing for empirical experimentation rather than making design decisions, wherever possible. This allows us to test hypotheses and explore the model by differing the variables. In the illustrative sessions described in chapters 5, 6, 7, and 9 we have hand picked parameter values in order to demonstrate the methods at work. Selection of parameter values in this case was mainly by trial and error, or by working out in theory which parameters would illustrate our ideas. Below we list the settings which we have varied in the illustrative sessions and give common values that they take. Before each

illustrative session we detail relevant parameters. Unless otherwise specified, parameters in HR take default values. In chapter 10 we present results from a more systematic selection of parameter values.

- The domain varies, and is usually number theory. The core concepts given vary slightly, and are usually integer, addition, and multiplication. In some cases we forced concepts at the start of a run. These are concepts which are in an agent's search space: the forcing process simply ensures that the concepts are formed straightaway. The objects of interest given also vary, and usually comprise some subset of numbers {0, 1, 2, ..., 59, 60} for each student.

- The number of students varies between one, two and three.

- The distribution of entities between students in the same agency varies. For example, some students might have the number 0 while others do not, or sometimes the numbers 1 - 60 are divided into consecutive sets.

- Which of Lakatos's methods a student uses, and the way in which to use it varies, according to parameters specified in the relevant chapters. For instance, to demonstrate how piecemeal-exclusion works, we would set students in the agency to perform piecemeal-exclusion.

- The evaluation of interestingness varies. For instance, we may set a student to prefer conjectures which are plausible, surprising, etc.

- The type of conjectures which a student makes, for instance implication or equivalence conjectures varies. In the case of near-conjectures we also vary the percentage of entities for which the conjecture should hold.

- The type of conjectures that the teacher requests varies; for instance it might request a near-implication conjecture.

- The number of theory formation steps that the students work for in the individual work phase varies. This is usually set to 10 or 20 steps.

The inflexibility of our parameters is a limitation of our system and we consider future, more flexible ways of parameter selection in 13.4.

4.6 Further details of HRL

Further details about the HRL system, including how to download HRL, how to generate a run of the program, and how to read the results can be found at the two addresses below. We describe what it is like to use the HRL system and how one might create new agents. Example domain files can also be downloaded.

- http://homepages.inf.ed.ac.uk/s9904767/hrl/
- ttp://dream.inf.ed.ac.uk/projects/hrl/

4.7 HRL and machine learning

Mitchell [1997] states that the field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience. He gives a specific definition below:

> A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [Mitchell, 1997, p. 2].

HRL is not a typical machine learning program in that its task is to output a theory, as opposed to well known machine learning problems such as finding a classification, or navigating through a space. However, if we define the class of tasks T as: finding examples of objects of interest, inventing new concepts, making plausible statements relating those concepts, and proving and disproving conjectures, as we did in §3.1, and the performance measure P as the interestingness measures in HRL, then we can argue that it learns according to Mitchell's definition. Assuming that a modified conjecture is more interesting than the initial flawed conjecture, then the performance of HRL at making plausible statements relating concepts, has improved as a result of the discussion. Given that it is also possible for an individual agent to generate and then modify near-equivalence or near-implication conjectures (see §4.4), we can similarly argue that agents, as well as the system, may learn.

4.8 Summary

We have outlined our system, and in particular described the agent architecture which enables dialogue. We have related these aspects to the concepts of environment, agent design and society design. The *problem* of the agency is to model the social process of concept and conjecture

refinement described by Lakatos, with the *task* being to develop interesting concepts and conjectures to add to the theory. The *knowledge* the agency starts with is the input to each copy of HR, which consists of objects of interest, core concepts and possibly concepts which have been forced by the user. The *motivation* of the students is to accept, modify, or reject a conjecture, and this is done by *actions* which are Lakatos's methods. The students *communicate* by sending conjectures, concepts, and counterexamples, and the teacher by sending requests such as: work independently; send a concept to cover counterexamples $[x, y, z]$; or send a modification to faulty conjecture C. *Discussion* is directed by the teacher who keeps a group agenda and adds responses to it, either depth, breadth or best-first according to user given commands and the teacher's interestingness criteria. Items which have been discussed are recorded in a discussion vector.

Chapter 5

The method of surrender

> "Sir, your composure baffles me. A single counterexample refutes a conjecture as effectively as ten. The conjecture and its proof have completely misfired. Hands up! You have to surrender." *Gamma*, student in [Lakatos, 1976, p.13].

5.1 Lakatos's method of surrender

The method of surrender is the first method in [Lakatos, 1976], and Lakatos devotes the least amount of space to it. It consists of using a counterexample to refute a conjecture. When this happens, the conjecture is surrendered. There is only one example in [Lakatos, 1976]; when the hollow cube is found, *gamma* rejects the conjecture and suggests trying a 'radically new approach'. What this approach might be is not investigated at this stage.

In this chapter we describe Lakatos's method of surrender and distinguish between two types. We consider the question of when we should give up on a conjecture, and suggest five ways: when it has been already been modified, when it is not interesting, when it is less interesting than the average conjecture in the theory, when it is not plausible, and when its domain of application, relative to the number of entities in the database, is very small. We give our algorithm for when to perform the method of surrender, and give three illustrative sessions. We conclude by discussing the method.

It is difficult to find historical examples of this method, as conjectures which have been surrendered do not appear in text books. However, in history of mathematics books we see some examples. For instance, Burton [1985] discusses the history of perfect numbers – a number whose divisors sum to itself. The first four perfect numbers, 6, 28, 496 and 8128 were known to the ancient Greeks, and later writers conjectured that the *nth* perfect number P_n contains

exactly *n* digits. This conjecture is wrong; the fifth perfect number is 33,550,336. Given this (and other) counterexamples, the conjecture was later surrendered.

5.1.1 Two types of surrender

Although surrender is presented as the first and least constructive reaction to a criticism of, or counterexample to, a conjecture, the characters in [Lakatos, 1976] do later on come back to it and perform it in a more sophisticated way. The simpler way is called *naive surrender*, and this is strongly criticised by Lakatos as an unproductive method. The latter way is not referred to as the method of surrender, but consists of distinguishing between an initial problem and an initial conjecture. In the Euler case study, the initial problem is to find out if there is a relationship between the number of edges, vertices and faces on a polyhedron, analogous to the relation which holds for polygons, that the number of vertices is equal to the number of edges. The initial conjecture is $V - E + F = 2$. The two questions concerned with the method of surrender are:

1) *when* should we give up on a conjecture?, and

2) what should we do next?

Lakatos did not explicitly answer the first question, but criteria implicit in [Lakatos, 1976] include: *(i)* if we believe the counterexample(s) to be valid (we have no reason to monster-bar it); and *(ii)* if the conjecture is too specialised, i.e. we have performed other methods on it to such an extent that its domain of application is now severely reduced. The answer to the second question concerns this latter point *(ii)*, and is referred to in [Lakatos, 1976] as the problem of content.

In this chapter we focus on the first of these questions.

5.1.2 When should we give up on a conjecture?

In the HRL system, the first criterion mentioned above, of believing the counterexamples to be valid, is satisfied when the monster-barring flag is set to false, or when the number of examples which the counterexample breaks is below a user-set threshold (the optimal value for this threshold is explored in chapter 10).

Clearly, conjectures which have been over-modified will become dull or too specific (for instance after repeated application of piecemeal exclusion). This would also prevent the system

from investigating more interesting paths. The question of what exactly over-modified means now arises. We have implemented and extended the second criterion in §5.1.1, which suggests that a conjecture should be surrendered if it is too specialised, or overly modified, in the ways listed below.

- We record the number of times each conjecture has been modified. If this number is over a user-set threshold then a student will surrender it rather than further modify it.

If the student finds the conjecture uninteresting then it will surrender it rather than further modify it. Uninteresting is defined in the following ways:

- the user-set weighting of measures of interestingness which the agent uses to evaluate conjectures is below a user-set threshold;
- compared to its other conjectures, the conjecture under discussion is uninteresting (calculate the average interestingness value of its own conjectures and if the interestingness of the conjecture under discussion is lower than the average then surrender, otherwise try to modify);
- the conjecture is considered implausible by the agent, i.e. it fails to hold for the user-set x% of its examples;
- the domain of application of the conjecture divided by the number of entities in the student's database, is below a user-set threshold.

5.2 Algorithm for the method of surrender

We describe our algorithm for the method of surrender in algorithm 1. When surrender is performed, this consists of removing the conjecture from the list of conjectures in the theory and adding it to a list of pseudo-conjectures. Thus, the conjecture remains in the theory but cannot influence it in any way (we justify this in §4.4).

5.3 Illustrative sessions

In the following sessions we present selected interactions, which are of particular interest in that they show the method of surrender at work. Parameter values such as number of students,

Algorithm 1 Algorithm for when to perform the method of surrender

Require: The parameter "use surrender" is set to true, and given a conjecture and a counterexample to it, and asked for modifications, then:

1: **if** ("test modifications" is true **and** the conjecture has been modified more than t_1 times),
or
("test interestingness" is true **and** the interestingness of the conjecture is below t_2),
or
("test average interestingness" is true **and** the interestingness of the conjecture is below the average interestingness of all of the conjectures in the theory),
or
("test plausibility" is true **and** the conjecture holds for less than t_3% of the objects of interest),
or
("test domain application" is true **and** the domain of application of the conjecture is less than t_4% of the objects of interest)
then

2: perform method of surrender

3: **end if**

distribution of the objects of interest, core concepts, and the way in which to apply a given method are specified at the start of a session (see §4.5 for further details). In this, and the following illustrative sessions we have hand picked parameter values in order to demonstrate particular aspects of Lakatos's methods in our model.

5.3.1 Session one

Initial information

We ran the agency with two students. Student 1 started with integers $1-10$, and core concepts integer, divisor, addition and multiplication. It was set to use surrender if the number of modifications of the conjecture was higher than 3, the interestingness of the conjecture under discussion lower than the average interestingness of conjectures in the theory, the interestingness less than 0.5, the plausibility less than 0.7 or the domain of application less than 0.5. It was also set to make near equivalences which hold for 60% or more of its entities.

Student 2 started with integers $11-20$ and the same core concepts as student 1. It was set to use surrender if the interestingness of the conjecture was less than 0.6, the plausibility less than 0.8 or the domain of application less than 0.6. It was not set to consider the number of modifications or to compare the average interestingness.

The teacher requested the students to work independently for 10 theory formation steps and then send their best near equivalence conjecture for discussion.

Session details

Student 1 made and sent the conjecture that for every integer c, there exist two other integers a and b such that $a+b=b+a=c$. This has counterexample 1, since 0 was not given as an integer. The teacher then asked for modifications to this conjecture. Student 1 calculated that the interestingness of the conjecture was both less than 0.5, its interestingness threshold, and less than the average interestingness of the conjectures in its theory. Therefore it surrendered the conjecture. Student 2 however, reconstructed the conjecture as an equivalence, since it did not find the counterexample 1. Since it had no counterexamples, it did not perform surrender.

5.3.2 Session two

Initial information

We ran the agency again with two students. We gave the students the same input as above, except that they were not set to make near equivalences. The teacher requested that the students work independently for 10 theory formation steps and then to send their best equivalence conjecture for discussion.

Session details

Student 2 (with integers $11-20$) made the equivalence conjecture shown above. Student 1 found the counterexample 1, and since the interestingness of the conjecture was lower than the interestingness threshold set, the student surrendered the conjecture. Student 2 however, again found no counterexamples and so neither surrendered nor modified the conjecture.

5.3.3 Session three

Initial information

We ran the agency again with two students. Both students started with integers 1-10 and core concepts integer, divisor and multiplication. Student 1 was set to perform surrender when the number of modifications is higher than 3, the interestingness of the conjecture under discussion is lower than the average interestingness of conjectures in the theory, the interestingness is less than 0.5, the plausibility is less than 0.7 or the domain of application is less than 0.5. Student 2 was only set to perform surrender when the number of modifications is higher than 3 or the plausibility less than 0.3. Both students were set to make near equivalences which hold for 60% or more of their entities.

The teacher requested the students to work independently for 10 theory formation steps and then to send their best near equivalence conjecture for discussion.

Session details
Both students formed the conjecture that integer a is equal to the number of its divisors if and only if $a|a \wedge a * a = a$. This has counterexample 2, which both students found. Student 1 found that the interestingness was lower than 0.5, the plausibility lower than 0.7 and the domain of application was only 0.2 which was lower than 0.5. Therefore it rejected the conjecture. Student 2 however, calculated that the number of modifications so far was 0, which was not higher than 3, and the plausibility was 0.5, which was not less than 0.3. Therefore student 2 attempted to modify the conjecture.

5.4 Discussion

The first and second sessions show that students have no loyalty to the conjectures which they have themselves proposed. Each conjecture is evaluated in ways which are specified in the input file, regardless of who proposed the conjecture. Thus, we saw that the student who proposed the conjecture in example one later surrendered it on the grounds that it was not interesting enough to attempt to modify it, while the student who had not proposed it evaluated it as acceptable in the state suggested. It may be worth modelling loyalty to one's own conjecture in further versions of the system.

The sessions described above include cases where the conjecture was surrendered and where a modified version was attempted. We do not present sessions for all of the conditions of application here: in chapter 10 we present experiments in which further settings are used.

5.5 Summary

In this chapter we have introduced Lakatos's first method, the method of surrender. This consists of rejecting a conjecture once a counterexample to it has been found, and is given only very brief attention by Lakatos. Although most of our work in implementing Lakatos's methods focuses on *how* the methods should be performed, we have focused instead on the question of *when* the method of surrender should be performed, since it is straightforward to implement. The question of *when* consists of considering how much a conjecture has already been modified, if it is uninteresting with respect to a given threshold, if it is uninteresting with respect to the other conjectures in an agent's theory, if it is considered implausible, i.e. if there are too

many counterexamples to it, or if it applies to too few entities, i.e. is too specialised. We have described our algorithm for determining these aspects, and given three illustrative examples. Finally, we have discussed our implementation. In section 5.1.1 we introduced the further question of what we should do after having surrendered a conjecture. This concerns the question of specialisation, and is called the *problem of content* by Lakatos. It concerns the situation where a conjecture has been specialised to such an extent that its domain of application is severely reduced. In §13.1.1 we consider how this might be implemented.

Chapter 6

The method of monster-barring

> "But why accept the counterexample? ... Why should the theorem give way...? It is the 'criticism' that should retreat.... It is a *monster*, a pathological case, not a counterexample." *Delta*, student in [Lakatos, 1976, p.14].

One reaction to an unwanted counterexample to a conjecture is to claim that the counterexample is not valid. The person making that claim then needs to argue *why* it is not valid, for example it is not the sort of thing that the conjecture covers. Plato [1993] provides a famous example of this type of reasoning. Simonides proposes that "it is right to give back what is owed" [Plato, 1993, pp. 8-9]. This initial statement is questioned by Socrates with the counterexample of someone borrowing weapons from a friend who subsequently goes insane, in which case it would not be right to return the weapons. The discussion in [Plato, 1993] then turns to what it means to give back what is owed, with Polemarchus suggesting that people owe their friends good deeds, and their enemies bad ones. The dialogue later turns to what the concept of *doing right* means, and leads into Plato's treatment of justice. This is an example of monster-barring. Once the validity of a counterexample has been questioned, the focus of the argument switches from the *truth* of the conjecture to the *meaning* of its terms.

In this chapter we firstly describe Lakatos's method of monster-barring (§6.1), and then look at how it is relevant to other mathematical domains (§6.2). In this section we also introduce monster-accepting, and describe some historical examples in which a field has progressed by accepting, rather than barring a monster. From the discussion we then identify six key points in monster-barring (§6.3), and in §6.4 we describe how we have modelled these points, as well as discussing further design considerations. In §6.5 we outline our algorithms for performing the method of monster-barring, and in §6.6 we present three illustrative sessions. We discuss our approach in §6.7, and in §6.8 we outline related work in the fields of ambiguity, argumentation, and law, and state how our work differs. In §6.9 we summarise the chapter.

6.1 Lakatos's method of monster-barring

Lakatos's method of monster-barring is a way of excluding an unwanted counterexample to a conjecture by claiming that the object in question is *not* a counterexample as it is not within the intended concept definition in the conjecture. In this context, problematic objects are seen as monsters advocated by anarchists, who should not be allowed to upstage a theorem which brings order and harmony to a field. The concept definition in question is then modified to explicitly exclude the unwanted object or counterexample.

The example in [Lakatos, 1976] concerns the concept of polyhedron. Although regular polyhedra are known, this concept initially has no explicit definition (which raises the question of how a theorem about polyhedra could have any meaning: see §13.5 for a discussion of this question). It is only when the hollow cube is proposed as a counterexample to Euler's conjecture that debate over the meaning of the terms in the conjecture breaks out. In an effort to justify calling the hollow cube a polyhedron, the student *Gamma* in [Lakatos, 1976] defines a polyhedron as *a solid whose surface consists of polygonal faces*. In table 6.1 we show the evolving definition, as described by Lakatos [1976], and subsequent counterexamples, with their Euler characteristic shown in brackets, where italics are used to highlight the part of the definition which is changed in the next step. Diagrams of each shape are provided in appendix A.

This method exploits any ambiguity in concepts, in order to defend a conjecture. In the first statement of Euler's conjecture, that for all polyhedra, $V - E + F = 2$, it is assumed that the extension of polyhedron is known, i.e., we can distinguish between objects which are and are not polyhedra, even if the definition is not explicitly agreed. Once an object of ambiguous status arises, students do explicitly define polyhedra, i.e., they start with a vague definition and make it more specific; although the new definition may include ambiguous sub-concepts such as *polygon*, *area*, and *edge*, whose definitions are also open to debate. The only criterion for a candidate definition is that it distinguish the agreed polyhedra from agreed non-polyhedra. Some characteristics, such as being a solid whose surface consists of polygonal faces, are seen as *sufficient* by one student (for instance *Alpha* who then suggests the twin polyhedra: two tetrahedra with either an edge or a vertex in common), and *necessary* by others (such as *Delta*, who, after admiring *Alpha*'s "perverted imagination", adds the condition that the system of polygons must be arranged in a particular way). With each definition there is some discussion as to how to resolve the ambiguity, and arguments for competing definitions are made. These centre around whether the problematic object should be considered a polyhedron or a monster. The arguments include being able to calculate the same properties for it as can be calculated for other polyhedra (such as area), and drawing an analogy between two universes and objects in them in which the object in the analogue domain which corresponds to the problematic object

Definition → Counterexample/Monster
1) initially undefined → hollow cube (16-24+12=4)
2) a polyhedron is a surface consisting of a *system* of polygons [Lakatos, 1976, p. 14] → twin tetrahedra with *(i)* an edge in common, *(ii)* a vertex in common (for which the Euler characteristic is *(i)* 6-11+8=3, *(ii)* 7-12+8=3 respectively)
3) a polyhedron is a system of *polygons* arranged in such a way that (1) exactly two polygons meet at every edge and (2) it is possible to get from the inside of any polygon to the inside of any other polygon by a route which never crosses any edge at a vertex [Lakatos, 1976, p. 15] → star polyhedron (12-30+12=-6)
4) a polygon is a system of edges arranged in such a way that (1) exactly two edges meet at every vertex, and (2) the edges have no points in common except the vertices [Lakatos, 1976, p. 17] → picture frame (16-32+16=0)
5) a necessary characteristic of a polyhedron is that through any arbitrary point in space there will be at least one plane whose cross-section with the polyhedron will consist of one single polygon [Lakatos, 1976, p. 21] → cylinder (0-2+3=1)
6) a necessary characteristic of an edge is that it has two vertices (therefore a cylinder does not have any edges) [Lakatos, 1976, p. 22]

Table 6.1: The evolving definition of polyhedron, described by Lakatos [1976], and subsequent counterexamples to Euler's Conjecture

in the target domain *is* considered valid. The idea is that if a good argument can be made for accepting or rejecting the object as a polyhedron, then the wider or more restrictive definition will be agreed. In the end, temporary resolution is insisted on by the teacher, whose strategy is to take the strictest definition in each case: "can anybody offer something which even the most restrictive definition would allow as a counterexample?" [Lakatos, 1976, p. 16] (this strategy excludes the definition *a polyhedron is a system of polygons for which the equation V-E+F=2 holds*).

The section on monster-barring in [Lakatos, 1976] is interspersed with heated debate on methodology: whether mathematicians should study typical, ordinary examples and generate interesting and useful theorems about these, or focus on boundary cases, studying mathematics in its "critical state, in fever, in passion" [Lakatos, 1976, p. 23]. The teacher concludes that monster-barring is *not* a valid method; indeed, it is presented as the least sophisticated method after the method of surrender. The main criticisms are that monster-barrers are anti-falsificationists

who defend a conjecture at any cost, which makes the conjecture deteriorate into meaningless dogma, and that the method is *ad hoc*, since the border between monsters and counterexamples is done in fits and starts. "Using this method one can eliminate any counterexample to the original conjecture by a sometimes deft but always *ad hoc* redefinition of the polyhedron, of its defining terms, or the defining terms of its defining terms. We should somehow treat counterexamples with more respect, and not stubbornly exorcise them by dubbing them monsters" (*Teacher*, in [Lakatos, 1976, p. 23]). The Duhem-Quine thesis, described by Bird [1998], that a scientific theory cannot be tested in isolation, since a test of one theory always depends on other assumption or hypotheses, is also relevant to this discussion. We cannot falsify a conjecture, rather we can show that a collection of assumptions, concepts, counterexample and conjecture is internally inconsistent. The choice then arises as to which of the collection we reject. Monster-barrers would argue that we should reject the counterexample and certain concept definitions, and retain the conjecture, whereas critics of monster-barring might argue that we should reject the conjecture under discussion.

Criticism of monster-barring is later questioned by *Pi* [Lakatos, 1976, p. 83], who claims that those who defend a conjecture by barring counterexamples as monsters are not guilty of *contracting* concepts, of changing the definition of key terms in an *ad hoc* manner in order to preserve the truth of the conjecture. Rather, refutationists who think up counterexamples are guilty of concept *stretching*: it is they who keep changing the definition of polyhedron. The original definition may be ill-formulated but was never intended to include monsters such as the hollow cube and twin tetrahedra; that it included these was "unrealised and unintended" [Lakatos, 1976, p. 85]. *Pi* distinguishes between concepts and definitions; where a concept is an intuitive, informal idea, a collection of objects, and a definition is a formal list of characteristics which may include/exclude objects originally intended to be covered by the concept. Therefore a definition may be expressed which includes a monster, but it is the definition, rather than the conjecture or methodology which is at fault: it has failed to express what was meant by the concept, it is sloppy terminology. If the definition is adequately stated then monsters, and new definitions which include them, reveal "the falsehood of a *new* conjecture which nobody had stated or thought of before" (*ibid.*, p. 85). It is the refutationists, rather than the monster-barrers, who have shifted their position. According to this view, all six definitions in table 6.1 are formalisations of the same concept. The students also distinguish between logical and heuristic counterexamples: a logical counterexample is one which is inconsistent with a conjecture *in its intended interpretation*, and a heuristic counterexample is one which spurs the growth of knowledge. The hollow cube was therefore a heuristic rather than logical counterexample, as it expanded the concept of polyhedron in a new and imaginative way. While in [Lakatos, 1976] it is not clear whether Lakatos agreed with the teacher or with *Pi*, in [Lakatos, 1981, p. 117] he strongly criticises monster-barring as an *ad hoc* method as opposed to a "co-

herent, pre-planned positive heuristic". Rather than excluding counterexamples as monsters, Lakatos argues, we should perform lemma-incorporation, which is a more productive method (see chapter 9).

6.2 Monster-barring in other mathematical domains

Further examples of Lakatos's method of monster-barring are useful as they *(i)* motivate the method by showing its generality, and *(ii)* aid implementation by adding detail and suggesting extensions to the method, as well as providing further inspiring examples, which were not considered by Lakatos himself. In this section we also see the importance of the converse of monster-barring, monster-accepting.

The concept 'number'

The process of suggesting that a new sort of object *is* a number, initial denial and later acceptance when it proves its worth can be seen repeatedly in number theory. For instance, Burton [1985] asserts that the number zero was not held to be a valid counting number for centuries, only being commonly used in practical calculations in the sixteenth century[1]. This was partially due to the Greek reluctance to accept it - they branded it a monster for various reasons, including its violation of the conjecture that if you add a number and a second number, then the result is always larger than the second number. (When zero was eventually accepted as a number this conjecture was modified to exclude zero, i.e., if you add a *non-zero* number and a second number, then the result is always larger than the second number.) Even today, it is still ambiguous as to what type of number zero is. It is usually considered that the set of natural numbers, $\mathbb{N}$, is the set of *positive* integers, $\mathbb{Z}^+$, i.e., $\{1,2,3,...\}$, but it can also mean a *non-negative* integer, i.e., the set $\{0,1,2,3,...\}$. The Collins Dictionary of Mathematics [Borowski and Borwein, 1989], defines a *natural number* as follows:

> "One of the counting numbers; a number that can represent the cardinality of a finite set of objects, usually identified with the positive integers 1,2,3,4, ... There is some discretion about whether 0 is included, as is usual for whole numbers." [Borowski and Borwein, 1989, p.397].

Other examples of ambiguity in the concept of number include initial barring of 1 (barred by the Pythagoreans as it challenged their belief that all numbers increase other numbers by multiplication); $\sqrt{2}$ (it violated the Greek belief that all numbers could be written as a fraction);

[1]Despite the fact that the number zero, or *omicron* (*o* was the first letter of the Greek word for "nothing") first appeared as a place holder in about 150 *A.D.* (in Babylonian positional notation the number 1, for instance, would have been ambiguous as it could equally represent 10, 100, etc.).

and $x = \sqrt{-1}$ (violating the law of trichotomy, for any numbers x, y, either $x = y$ or $x < y$ or $x > y$). Now of course 0, 1, irrational and imaginary numbers are accepted without question, and the concept of number has been generalised to complex numbers and beyond (for example, quarternions). Clearly number theory (and other areas of mathematics) have been greatly enhanced by these additions.

Another example can be seen in the late 19th century, when Cantor's research into trigonometric series led him to develop a theory of infinity based on set theory, in which a set is infinite if it can be put into a one-to-one onto correspondence with a proper subset of itself. Transfinite cardinals denote the size of an infinite set, with two sets having the same cardinality if there exists a bijective mapping between them. This led to the claim that there are different sizes of infinity, since there does not exist such a mapping between, for example, the set of real numbers and the set of natural numbers. This claim arose from Cantor's diagonalisation argument (see appendix B) in which he proved that the size of the set of natural numbers is different to the size of the set of reals. An order was defined on the cardinals, and the question of whether there were any cardinals in between those denoting the size of the naturals and the reals was labelled the continuum hypothesis. Proving or disproving the continuum hypothesis was a major focus in mathematics until Godel, in 1938 showed that it is consistent with ZF set theory plus the axiom of choice (ZFC)[2] (so cannot be disproved within the ZFC formalisation), and Cohen showed, in 1963, that it is independent of ZFC (so cannot be proved within ZFC). Additionally, Cantor proved that not only are there at least two sizes of infinity; there are an infinite number of transfinite cardinal numbers since no set can be put into a one-to-one onto correspondence with its power set.

Transfinite numbers were considered by many mathematicians at the time to all be monsters. For instance, the 'number' $\aleph_0$, which is the size of the set of all integers (the first transfinite number) is a counterexample to the conjecture above, that if you add a *non-zero* number and a second number, then the result is always larger than the second number, since $\aleph_0 + \aleph_0 = \aleph_0$. Similarly it violates the conjecture that any positive number multiplied by integer $n > 1$ is bigger than the number, as $\aleph_0.n = \aleph_0$. The law of monotonicity, that for all numbers a, b and c, if $b < c$, then $a+b < a+c$ fails if $a = \aleph_0$ (for any finite b and c). For these and other reasons, initial reaction to Cantor's work was hostile, and $\aleph_0$ was branded a monster and barred from the concept of number[3]. However Cantor was developing a whole area of mathematics which

[2] Zermelo-Fraenkel set theory, with the axiom of choice, commonly abbreviated to ZFC, is the standard form of axiomatic set theory.

[3] These reasons include the belief that infinity belonged to the domains of philosophy and theology rather than mathematics. Aristotle, Galileo and Bolzano considered it mathematically, but mainly in the context of a series of puzzling 'paradoxes' such as the idea that the set of naturals is the same size as the set of squares - which tended to confirm the idea that infinity and mathematics should not mix. Another reason was that Cantor's methods

he considered to be interesting and worthwhile, which included the number $\aleph_0$. Therefore he continued his research and tried to convince the mathematical community (eventually with success) that transfinite numbers *are* a kind of number and a valid area of mathematics.

As Kadvany [2001] points out, however, it is unlikely that Cantor included an implicit notion of transfinite number in his concept of number before his development of them. Cantor had to refine the notions of finite and infinite set, defining them in a precise, mathematical way. One of the first attempt to describe cardinal numbers was: "two sets have the same cardinal number or have the same power if they are equivalent" [Burton, 1985, p. 593]. Cantor later tried to give a more specific definition by abstracting away the elements in the set: "if we abstract both from the nature of the elements and from the order in which they are given, we get the cardinal number or power of the set" [Burton, 1985, p. 593]. Frege and Russell then independently tried to formalise this concept as: "the cardinal number of a set *A* is the set of all sets equivalent to *A*" [Burton, 1985, p. 593].

Note that the definitions for cardinal number change, but not definitions for number itself, although the inclusion of transfinite numbers sparked a row about the nature of number and even mathematics. Ultimately, a more general definition of number was given, as a cardinal number, and this was split into finite and transfinite cardinals. This is an example where it was eventually considered to be more productive to accept, rather than bar the monster under discussion.

All of these examples are examples of concept *stretching*, where a concept definition is widened to include an object which was previously excluded. In this example concept stretching has aided the development of the theory. The example also shows how an object which is controversial is judged on whether it breaks many conjectures, or theorems which are held. Additionally, it shows how it is possible for an object to force other objects, which were previously positive examples of a conjecture, to become counterexamples to conjectures in a theory.

We can also see monster-barring, where a concept definition is *narrowed* to exclude a problem

depended on non-constructivist reasoning - asserting the existence of something without providing a method for its construction. (Kroenecker, Cantor's former professor, in particular objected to this, considering it completely illegitimate. Partly as a result of this, Cantor was never offered a professorship in any of the important German universities, living instead at Halle university.) This led to strange results such as the size of the set of transcendental numbers, $\mathbb{T}$, is bigger than that of the integers, $\mathbb{Z}$. This is because the cardinality of the set of real numbers, $\mathbb{R}$, is bigger than that of $\mathbb{Z}$ (i.e., there does not exist a one-to-one function from $\mathbb{R}$ into $\mathbb{Z}$) - which means that $\mathbb{R}$ is uncountable (cannot be put into a one-to-one onto relationship with $\mathbb{Z}$); $\mathbb{R}$ is the union of the set of algebraic ($\mathbb{A}$) and transcendental numbers ($\mathbb{T}$); and $\mathbb{A}$ is countable which means that $\mathbb{T}$ must be uncountable. However, at the time of proving this result only one member of $\mathbb{T}$, *e*, was known (the theorem was proved in 1874, and *e* was established as a transcendental number in 1873 by Charles Hermite, with the second transcendental number, π established in 1882 by Ferdinand Lindemann [Burton, 1985]). The idea that a set containing only one known member is significantly larger than a set in which any number of members are known ($\mathbb{Z}$) and is clearly infinite, was disconcerting and for mathematicians who consider maths to be clear and definite, the assertion of the existence of a number which is not actually known is very unsatisfactory.

entity, on the concept of a set, or *menge* (which is also related to Cantor's work). Cantor originally defined a set as follows:

> "By a set we are to understand any collection into a whole M of definite and distinguishable objects of our intuition or our thought. These objects are called the elements of M." [Burton, 1985, p. 591]

However, this definition was not precise enough to prevent the problem sets *the set of all sets*, and *the set of sets which are not members of themselves*. This lead to paradoxes including Russell's paradox: *if S is the set of all those sets that are not members of themselves, then S is a member of itself, and it is not a member of itself*, which necessitated the refinement of his concept of set. But this was unsuccessful, meaning that today the concepts set and element are undefined, primitive terms[4].

The concept 'prime'

Dunmore [1992] considers the relationship between concept and conjecture (a conjecture is clearly dependent on the concepts within it, but the relationship also works the other way around). She cites an example in number theory, of the definition of prime number. A prime was initially defined to be 'a natural number which is only divisible by itself and 1'. However this definition includes the number 1, which was found to be a counterexample to many theorems and conjectures about primes. In particular, the Fundamental Theorem of Arithmetic (FTA) states that every natural number is either prime or can be expressed uniquely as a product of primes. If 1 is also considered prime, then this violates the uniqueness claim, since, for example, $6 = 2*3 = 2*3*1 = 2*3*1*1 = 2*3*1*1*1 =$ etc.. Rather than explicitly exclude the prime 1 from this (and other) theorems, it might be preferable to exclude it from the concept definition. Today it is often defined as 'a natural number with exactly two divisors', thus enabling many theorems about primes, including the FTA, to be neatly stated. This is an example where monster-barring has been beneficial.

[4] In this example it was axioms, rather than a concept definition, which were modified. [Frege, 1903] suggested a definition of number based on set theory, in his attempt to derive arithmetic from logic. This work contained *the comprehension principle*, which is the axiom that *given any condition expressible by a formula $f(x)$, it is possible to form the set of all sets x meeting that condition, denoted $\{x : f(x)\}$*. The comprehension principle was modified in order to prevent paradoxes such as Russell's paradox from occurring. This was done by limiting the type of sets which it is possible to build. The axiom became known as the *axiom of subsets*, or *separation* and was stated as axiom 3 in Zermelo-Fraenkel Set Theory as *for any property $P(x)$ of objects and any set A, there is a set $\{x \in A : P(x)\}$ which contains all the elements of A with the property P*. This states that given the set S, and any meaningful (definite) property P there exists a subset consisting of just those members of S which satisfy P. That is, in order to construct a new set, an existing set as well as a property is necessary. This prevents the construction of the set of all sets.

6.3 Key points in monster-barring

In summary, we have seen in sections 6.1 and 6.2 that the main points about monster-barring are:

1. it raises the distinction between concepts (ill-defined, intuitive, possibly with unexpressed meaning) and definitions (specific interpretations of a concept);
2. the process starts with an ambiguous concept, and makes it less ambiguous (although there may still be some grey areas);
3. monster-barring involves taking the narrow definition and excluding a monster. However, in contrast, concept stretching can also aid development of a theory;
4. arguments are made for rival definitions based on the desirability of admitting a certain problem object into a concept domain. This desirability may consist of wanting to:
   (a) exclude the object, in order to defend a certain conjecture;
   (b) include the object, in order to attack a certain conjecture;
   (c) exclude the object, since it breaks many conjectures in the theory;
   (d) exclude the object, since it forces other objects in the theory to become counterexamples to conjectures in the theory;
5. there are two decision points during the process of monster-barring; deciding whether to suggest that an object be barred, and deciding how to react to such a suggestion;
6. the dialogical aspect of Lakatos's work is clearly seen in this method, where different definitions are negotiated and debated.

6.4 Implementing monster-barring

In this section we describe our implementation of monster-barring, in terms of the six points made above. As with all methods, it is implemented within the architecture described in chapter 4 and each agent uses a copy of HR [Colton, 2002] (see chapter 3) to generate concepts and the initial conjectures. To recap, all communication is via the teacher who decides on a group agenda and sends requests to the students (who respond or send requests of their own). Students work in two phases: independently, in which they develop their own theories, and a discussion phase in which they communicate and discuss their conjectures, concepts and

counterexamples. The teacher initially asks each student for their most interesting conjecture where the interestingness function is a weighted sum of measures (see §3.1.3) which is input by the user. The user can also set the teacher to specify a certain type of conjecture, e.g., an implication.

6.4.1 Lakatos's distinction between concepts and definitions

We have seen that Lakatos distinguished in monster-barring between concepts, which are ill-defined, and definitions, which are specific interpretations of a concept. In order to implement this aspect, we exploit the distinction in HR between core concepts and developed concepts, described in §3.1. To recap, a core concept is one which is input by the user, and consists of a list of objects and possibly a way of generating them, but no specific definition. A developed concept, on the other hand, has been generated from other concepts using production rules, and does have a specific definition. This is analogous to Lakatos's distinction between concepts and definitions, although clearly, the concepts are generated in a very different way.

6.4.2 Making a concept less ambiguous

In order to find a specific definition for a core concept, i.e., to start with an ambiguous concept, and make it less ambiguous, we have implemented a type of monster-barring which we call "vague-to-specific". Given an undefined, core concept, a conjecture, and an entity which may be a 'monster' or may be a valid counterexample to the conjecture, if a student is set to perform "vague-to-specific" monster-barring, it will firstly generate a list L of developed concepts in its theory which are conjectured to be equivalent to the core concept. If the student wants to exclude the entity as a monster, it will remove from L any concept which includes the entity in question, and if it wants to include the entity as a valid counterexample, it will remove from L any concept which excludes the entity in question. The student then removes from L any concept which is in the conjecture under discussion, and then, if L is not empty, it will find the most interesting concept X in L, according to the weighted interestingness criteria. The student then proposes X as an alternative concept definition.

We have also implemented a type of monster-barring which we call "vague-to-vague", in which case the concept definition is left unchanged, and the debate is over whether the concept should cover a given entity or not.

6.4.3 Concept stretching

We have seen (§6.2) that concept-stretching – where a concept definition is widened to include an object which was previously excluded – can aid the development of a theory. Therefore we have implemented what we call *monster-accepting*. Given a conjecture, and a proposal to bar a potential counterexample from a concept in the conjecture, a student will perform monster-accepting if the entity does not break the conjecture under discussion, or if it breaks less than a user-defined proportion of conjectures in the student's theory. If this is the case, then the student will reject the proposal, and, if the flag "vague-to-specific" is set, the student will find a wider concept definition which does cover the entity definition and communicate this alternative definition to the teacher. The inconsistency in the conjecture under discussion is then dealt with by other methods.

6.4.4 When to perform monster-barring

Given an entity which is new to a student, and has been suggested as a counterexample to a conjecture, the student will propose to monster-bar the entity in the following cases:

- if the conjecture under discussion has no counterexamples in the student's theory; or
- if the number of conjectures which are broken by the entity is over a user-set monster-barring minimum; or
- if the number of counterexamples to the conjecture is more than half the total number of entities, and either the entity is responsible for all of the entities in the theory being counterexamples to the conjecture, or the entity in question forces other entities into being counterexamples for a higher number of conjectures than the proportion specified.

Flags are set at the start of each session to determine which of these cases is followed.

6.4.5 Two decision points

We have implemented two decision points during the process of monster-barring. The first occurs when a student receives an entity which is new to it, and the student decides whether to propose to monster-bar the entity. The second decision point occurs when a student receives such a proposal. The student then evaluates the proposal and votes on whether to accept or reject it.

6.4.6 Social dialogue

Recall that in §3.2.4 we described work by Jennings et al. [1998], who argue that an agent capable of argumentation-based negotiation must have a mechanism for communicating proposals and supporting arguments, generating proposals, assessing proposals and arguments, and responding to proposals. Agents move through the space of possible agreements, and negotiate points in the space which are acceptable. Negotiation may include agents explaining why they are rejecting/proposing a certain point. Jennings et al. [1998] do not suggest that the meaning of a concept could be a possible object over which agents negotiate (other people in the research community, however, have investigated this; see §6.8.2 for an outline). Instead they give examples of issues relating to negotiation over services or products. However, their framework is general enough to include this type of negotiation. In this section we describe our implementation in terms of their work.

Generating a proposal

The teacher prioritises the conjectures it receives and then, if the flag "communal piecemeal exclusion" is set, asks the students for counterexamples to the first conjecture on the agenda. (If this flag is not set then the teacher asks for modifications to the first conjecture in the group agenda.) If the teacher asks for counterexamples, the students look in their theories and send back any counterexamples they find. Before adding these to the agenda, the teacher sends them to the students, to check that they consider the counterexamples to be valid, or whether they want to monster-bar any. If all of the students accept the counterexamples as valid, then they will add any new entities to their theories. The teacher then requests a concept which covers all of the counterexamples, selects one concept from those concepts which are suggested, and modifies the conjecture by excepting that concept (see piecemeal exclusion in §7.6). Students can only decide to monster-bar counterexamples if they are set to do so by the user. If so, and a student is sent an entity which is new to it, then the student will propose to monster-bar in the following cases:

(i) if the object breaks the conjecture under discussion, which otherwise has no counterexamples in the student's theory (this captures *4a* and *4b*, in §6.3);

(ii) if the object breaks more than a user-set proportion of conjectures in the student's theory (this captures *4c*, in § 6.3);

(iii) if the object forces other objects to break the conjecture under discussion, or further conjectures in the student's theory. That is, if the student includes the object in its theory further objects become counterexamples to a conjecture, and if it does not, then these other objects are not counterexamples (this captures *4d*, in § 6.3).

See algorithm 2 on page 78 for the algorithm for the student to determine *when* to propose to bar an entity.

Communicating proposals and supporting arguments
If a student has generated a proposal in the previous stage, then it sends the proposal to the teacher, who puts the proposal into the group agenda. When it reaches the top of the agenda, the teacher sends the proposal to all of the students.

Assessing proposals and arguments
If a student receives a proposal to bar an object, then it will evaluate the proposal. The student will vote to accept the proposal if: *(i)* the object breaks the conjecture under discussion, which otherwise has no counterexamples in the student's theory; *(ii)* the student is set to accept the strictest definition, or *(iii)* the entity breaks more than a user-defined proportion of the student's conjectures. The student will vote to reject the proposal if: *(i)* it already has counterexamples to the conjecture under discussion in its theory, or *(ii)* the entity does not break more than a user-defined proportion of its conjectures.

Note that if the object is not new to a student, i.e., is already in their theory, then they can only vote to accept a proposal to bar the object, rather than suggesting the initial proposal that it be barred. This reflects a loyalty to one's own data set, although in future versions of the system, it might be worthwhile investigating whether allowing agents to propose to bar their own objects would result in more interesting theories (this would also depend on the distribution of the objects of interest at the start of a session). A student may wish to bar one of its own entities if the entity is a counterexample to many of the conjectures in its theory.

See algorithm 3 for the algorithm for the student to *evaluate a proposal* to bar an entity.

Responding to proposals
The students respond to a proposal to bar an object by sending the teacher their vote on whether to bar the object or not. The teacher counts the votes and makes a democratic decision. If the votes to bar the object outweigh those to accept it, then the teacher tells the students with the monster in their theories to remove it. Alternatively, if the votes to accept the object outweigh those to bar it, or if there is an equal number of votes for each position, then the teacher will tell the students who do not yet have the object (which is now agreed as a valid counterexample) to add it to their theories.

Jennings et al. [1998] defines *negotiation* as the problem of reaching mutually acceptable agreements. We define *semantic negotiation* – the problem of reaching mutually acceptable definitions – as a specialised type of negotiation. This is a part of the method of monster-barring. In Pease et al. [2002], we argue that semantic negotiation is an important part of human reasoning.

6.4.7 Further design considerations

Simplifying assumptions

Discussion of the validity of monster-barring concerns concepts which are difficult to simulate, such as *intended* definitions, as seen in § 6.1. In our implementation, we have ignored such subtleties and simplified the method. The question of whether a new definition was previously intended or not is beyond the scope of this thesis. Instead, we bypass the issue of whether a concept is being narrowed to exclude a monster, or stretched to include a counterexample, by implementing both monster-*barring* and monster-*accepting*.

Removing input entities from the database

In order to simulate examples such as the ambiguity of the number 1, we needed to set some agents running without the entity 1 in their list of numbers. When removing integers from the database, we had to consider the way in which the core concepts are generated. For instance, it seems odd for an agent not to have the entity 1 in its core concept of integer, yet still have it in the table of divisors, multiplication, etc. If a student had the integers $2-5$ and the concepts divisor and multiplication, then its data table would be be as shown in table 6.4.7.

integer	**divisor**	**multiplication**
2	f(2) = [[1],[2]]	f(2) = [[1,2],[2,1]]
3	f(3) = [[1],[3]]	f(3) = [[1,3],[3,1]]
4	f(4) = [[1],[2],[4]]	f(4) = [[1,4],[2,2],[4,1]]
5	f(5) = [[1],[5]]	f(5) = [[1,5],[5,1]]

This seemed counterintuitive, and we initially resolved it by adding new ways of forming the concepts divisor and multiplication (for instance, starting to calculate the divisors from 2 rather than 1), which excluded the number 1 from anywhere in the table. However, this also has disadvantages, since it means that the entity 1 is not known in any capacity, which was not the case in the historical example in which the status of the entity 1 was controversial (§6.2). The attitude of the Greeks, where the example came from, is that while it is possible to divide a number by itself and get 1, and multiply a number by 1 to get the same number, the entity 1 is not itself a number (but is rather the generatrix for all numbers). Euclid believed that a number is an aggregate composed of units - and 1 is not an aggregate of itself [Wells, 1997, p. 13]. Therefore, for the sessions we describe in §6.6, we do not use the new ways, but leave the tables as they are above, on the grounds that there is no reason why all of the entities in all of the tables be considered integers.

Removing entities from the database during the discussion

When a group has decided to monster-bar an object, it would be time consuming to go through

its theory and change all of the data tables of concepts involving the entity, and all those data tables of concepts deriving from these concepts, and all conjectures involving these concepts. This is also unrealistic conceptually: experts in a community would not immediately entirely clear their mind or theory of a problem entity once they have agreed not to use it. This would also mean much more work if it is ever decided, either communally or individually, to re-introduce the problem entity. Therefore we remove the entity from the list of entities (not the data tables), and put it into a list called *pseudo_entities*. Anything in this list does not count as a counterexample when generating or evaluating conjectures.

6.5 The monster-barring algorithms

Whenever a student sends any entities to the teacher for discussion, the teacher sends these to all other students and awaits their responses. The students then cycle through the received list of entities, and for each entity, if their monster-barring field is set (by the user) to "true", performs the tests below to determine whether to vote to monster-bar the entity, or to add the entity to their theory. If monster-barring is set to "false" then the receiving student checks to see whether it already knows the entity. If it does, then the student does nothing, and if not the student adds the entity to its theory. In algorithm 2 we describe how a student determines *when* to propose to monster-bar an entity, given an entity which is new to the student, and has been suggested as a counterexample to a conjecture. In algorithm 3 we describe how a student evaluates a proposal to monster-bar an entity, given a request from a student to bar the entity. Finally, in algorithm 4 we describe how a student performs monster-barring or monster-accepting, given that either the student is proposing to monster-bar an entity, or is voting on whether to accept or reject a proposal to monster-bar an entity.

If the teacher receives a request to monster-bar an entity, then it sends a request for all students to evaluate the proposal. Each student then sends a vote, to either accept the proposal (monster-bar the entity) or reject the proposal (monster-accept the entity). The teacher then counts the votes in the agenda which are for or against monster-barring, calculates the overall consensus (if there are an equal number of votes then the teacher takes the narrowest definition, which excludes the entity as a monster). The teacher then sends the appropriate request to the students to either downgrade the entity to a pseudo-entity, if the consensus is to monster-bar the entity, or to add the entity to their theories, if the consensus is to monster-accept the entity.

Algorithm 2 Algorithm for the student to determine *when* to propose to bar an entity

Require: An entity which is new to the student, and has been suggested as a counterexample to a conjecture.

if the monster-barring flag is set **then**
  if the flag "use breaks conj under discussion" is set, and the conjecture under discussion has no counterexamples in the student's theory **then**
    propose to bar the entity
  else if the flag "use percentage conjectures broken" is set, and the number of conjectures which are broken by the entity is over the monster-barring minimum **then**
    propose to bar the entity
  else if the flag "use culprit breaker" is set and the number of counterexamples to the conjecture is more than half the total number of entities **then**
    if the flag "use culprit breaker on this conjecture" is set **then**
      if a single entity is responsible for all of the entities in the theory being counterexamples to a given conjecture **then**
        propose to bar the culprit entity
      else
        add the entity to the list of entities
      end if
    else if the flag "use culprit breaker on all conjectures" is set **then**
      if the entity in question forces other entities into being counterexamples for a higher number of conjectures than the proportion specified **then**
        propose to bar the culprit entity
      else
        add the entity to the list of entities
      end if
    end if
  end if
else
  add the entity to the list of entities
end if

Algorithm 3 Monster-barring algorithm for the student to *evaluate a proposal* to bar an entity

Require: receive a request from a student to bar an entity

1: **if** the flag "use breaks conj under discussion" is set **then**
2: **if** the conjecture under discussion has no counterexamples already in the theory **then**
3: vote to accept the proposal to bar the entity, and send the vote to the teacher
4: **else**
5: vote to reject the proposal to bar the entity, and send the vote to the teacher (monster-accepting)
6: **end if**
7: **else**
8: **if** "accept strictest definition" flag is set **then**
9: vote to accept the proposal to bar the entity, and send the vote to the teacher
10: **else**
11: calculate the percentage x of conjectures in the theory which are broken by the entity
12: **if** $x >$the monster barring minimum **then**
13: vote to accept the proposal to bar entity, and send the vote to teacher
14: **else**
15: vote to reject the proposal to bar the entity (monster-accepting)
16: **end if**
17: **end if**
18: **end if**

Algorithm 4 Monster-barring algorithm for the student to decide *how* to perform monster-barring or monster-accepting

Require: the student is either proposing to monster-bar an entity, or is voting on whether to accept or reject a proposal to monster-bar an entity

if the flag "monster-barring-type" is set to "vague-to-specific" **then**
  generate a list L of concepts in the theory which are conjectured to be equivalent to the undefined concept
  if the student is either proposing to bar the entity, or voting to accept the proposal to bar the entity **then**
    remove from L any concept which includes the entity in question
  else
    if the student is voting to reject the proposal to bar the entity **then**
      remove from L any concept which excludes the entity in question
    end if
  end if
  remove from L any concept which is one of the concepts in the conjecture under discussion
  if L is not empty **then**
    find the most interesting concept X in L, according to the weighted interestingness criteria. Vote to accept/reject the proposal to bar the entity, as appropriate, and suggest X as an alternative concept definition. Send the vote to the teacher
  end if
else
  if the flag "monster-barring-type" is set to "vague-to-vague" **then**
    vote to reject the proposal to bar the entity, leave the definition as it is, and send the vote to teacher
  end if
end if

6.6 Illustrative sessions

In the following sessions, we present selected interactions, which show the method of monster-barring at work.

6.6.1 Session one: barring the number one

Input information:

We ran the agency with three students and a teacher. The first student started with the integers $1-30$ and background concepts of integers, divisors, and multiplication. It was set to propose or agree to monster-barring if the entity in question broke more than 10% of its conjectures. The second student started with the integers $2-30$ and background concepts of integers, divisors, addition and multiplication. It was set to propose monster-barring if the entity in question forced all of the other entities in its theory to become counterexamples, and to agree to monster-barring if the entity in question broke more than 20% of its conjectures. The third student started with the integers $2-30$ and background concepts of integers, divisors, and multiplication. It was set to propose or agree to monster-barring if the entity in question broke more than 10% of its conjectures. All three students were set to make near-equivalence conjectures which held for 80% of their entities. The teacher requested non-existence conjectures, i.e., conjectures stating that a particular definition has no examples.

Session details:

The third student made the conjecture that there does not exist a number such that when multiplied by itself gives the same number, i.e., not exists n such that $n*n=n$), and sent it to the teacher, who put it on the agenda for discussion. The teacher then asked for counterexamples to the conjecture, and the first student sent back the number 1. The teacher sent this to all the students to see if they were happy for it to be added to the group discussion agenda. The third student found that 1 was a counterexample to 30% of its conjectures, and proposed to bar it.

All students then evaluated the proposal. The first student found that 1 was a counterexample to 11% of its conjectures, and the second student found that it broke 44% of its conjectures. Therefore all students agreed that it should be barred, and the teacher requested them to downgrade 1 to a pseudo entity.

6.6.2 Session two: adding the number one

Input information:

We ran the agency again with three students and a teacher, where each student had the same background information as in the example above (§6.6.1), but different monster-barring settings. The first student was set to propose or agree to monster-barring if the entity in question broke more than 20% of its conjectures. The second student was set to propose monster-barring if the entity in question forced all of the other entities in its theory to become counterexamples, and to agree to monster-barring if the entity in question broke more than 50% of its conjectures. The third student was set to propose or agree to monster-barring if the entity in question broke more than 20% of its conjectures. The students were not set to make near-equivalence conjectures. As in §6.6.1, the teacher requested non-existence conjectures.

Results of session:

The third student made the conjecture that there does not exist a number such that when multiplied by itself gives the same number, i.e., not exists n such that $n*n=n$), and sent it to the teacher, who put it in the agenda for discussion. The teacher then asked for counterexamples to the conjecture, and the first student sent back the number 1. The teacher sent this to all the students to see if they were happy for it to be added to the group discussion agenda. The third student found that 1 was a counterexample to 30% of its conjectures, and proposed to bar it.

All students then evaluated the proposal. The first student found that 1 was a counterexample to 12% of its conjectures, and the second student found that it broke 40% of its conjectures, and both students voted to reject the proposal to bar the entity. As the votes to reject the proposal outnumbered those to accept the proposal, the teacher requested all students to add the number 1 to their theories.

6.6.3 Session three: barring the number zero

Input information:

We ran the agency with two students and a teacher. The first student started with the integers $0-10$ and the second student started with the integers $1-10$ (i.e., did not know the number 0). Both students started with the background concepts of integers, divisors, and multiplication. The teacher requested non-existence conjectures, i.e., conjectures about a concept which has no known examples. The students were set to work individually for 20 steps and then enter into discussion. The students were both set to use monster-barring, and specifically to test

whether an entity was a culprit breaker when deciding whether to propose monster-barring or not. The monster-barring minimum was set to 15%, i.e., if a proposal to monster-bar an entity was made, both students would evaluate the proposal by testing to see whether the entity was a counterexample to more than 15% of its conjectures (in which case the student would agree to bar it).

Session details:

The second student made the conjecture that there do not exist integers a, b such that $b+a=a$ and $a+b=a$, and sent it to the teacher, who put it in the agenda for discussion. The teacher then asked for counterexamples to the conjecture, and the first student sent back all its integers, since having 0 in its theory meant that *every* number appears in a counterexample (which consists of a pair of numbers). For instance, 1 appears in a counterexample since $1+0=1$, similarly 2 also appears in a counterexample since $2+0=2$, etc.. The teacher then asked for responses to the counterexamples, and the student with 0 tested to see whether there was a single 'culprit' entity which was forcing all of its entities to be counterexamples, and concluded that 0 was a culprit entity. As a consequence it then sent the request to the teacher to monster-bar 0. The teacher put this request into the agenda, and sent it to the second student, who tested to see how many of its conjectures the number 0 broke. It found that 0 broke 63% of its conjectures, and as that was more than 15%, voted to monster-bar 0. The teacher counted the votes and told both of the students to down-grade 0 to a pseudo-entity. Both students then added 0 to their pseudo-entities list, which meant that it was now in both of their theories but did not count as a valid integer.

6.7 Discussion

The illustrative results above are the beginning of an attempt to simulate the discovery and resolution of ambiguity in mathematics. For instance, the session described in §6.6.1, is a very simple model of the historical process of barring the number 1, when it was first considered, because it broke conjectures that people wanted to keep. It is an example of vague to vague semantic negotiation, as the concept of 'number' is no more concrete after the discussion and the barring of 1 than before. The session described in §6.6.2 shows an example of monster-accepting.

The failure at the beginning of the last century of the quest for a perfect language in which neither ambiguities, nor paradoxes, nor redundancies exist, showed the difficulties involved in writing a formal language which can be used to describe a reasonably large domain. Different

types of ambiguity include *lexical* (where a word has two different meanings); *syntactic* (where a sentence has two syntactically correct derivation trees which indicate different meanings); *semantic* (where a sentence has two meanings, only one of which makes sense), and *pragmatic* (where the meaning of a word is relative to the speaker). Much work on AI and ambiguity concerns methods to automatically determine a writer's intended meaning (for example Romacker and Hahn [2001] who focus on representing and managing ambiguity in natural language text understanding). In contrast, we are concerned with ways in which ambiguity may be exploited (or introduced into a previously unambiguous concept), in order to support an argument or set of beliefs.

Ambiguity is widespread in human argument, and can be exploited in order to support goals. However, this phenomenon is rarely used in AI research. Economic agents may well disagree on the price of a potato, even haggle over it in a reasonably sophisticated way, but they do not usually start arguing about what a potato is. The fact that although participants in a discussion use a shared language, while some of the terms are ambiguous, raises questions such as: what sorts of things can be ambiguous? How might ambiguity arise? How can it be resolved? Can it be used to produce richer theories? The method of monster-barring helps us to answer these questions[5].

What sorts of things can be ambiguous?

In mathematical theories, at least two types of component may be ambiguous - objects and concepts. Two ways in which an object can be ambiguous are: *(i)* there may be two different objects with the same name, or *(ii)* there may be a single object which is represented in different mutually inconsistent ways, similar to Rubin's vase (this is the method of monster-adjusting). If we see mathematicians as working within a domain, or 'universe', such as polyhedra, or numbers, then both sub-concepts within the universe, such as the concept of a prime number, as well as the universe itself, can be ambiguous.

How might ambiguity arise?

Some concepts are initially specifically defined (everyone agrees on the definition), and the definition changed (someone argues that a second definition would be more useful). Others are initially vague (it is not known whether some objects are examples of the concept or not) and

[5]There are also questions as to the extent and type of ambiguity. We have already seen deep disagreement over whether the concept of infinity is a mathematical one or not, with some mathematicians arguing that it simply cannot be represented mathematically (§6.2). Ryle [1949] describes a notion of *logical geography*, which is a conceptual scheme, or context within which concepts and propositions may legitimately be expressed. If, for instance, a philosopher represents a concept in a different logical geography from another, then misunderstanding and disagreement will ensue. Ryle demonstrates his point with reference to the traditional mind-body dualism which, he argues, is absurd since we use physical representations and concepts such as location in space to talk about mental concepts. He argues that the function of philosophy is to map the logical geography, which will consist in relocating, rather than denying, ideas. In this view an important aspect of ambiguity would be the distance on the logical map between different interpretations.

only when disagreement arises are different concept definitions made explicit.

How can ambiguity be resolved?

Experts evaluate the worth of each of the rival definitions (often with different results). The existing definition is assumed by default, with the onus on proponents of a new definition to convince the other experts of its value. Grounds for accepting the new definition include showing that it produces interesting new theories or results (including maintaining an interesting conjecture). There is usually a period during which it is unclear whether the object in question belongs to a concept or not. It passes through a period of indefinite status with some people accepting its status, others not, others unsure, until it either proves its worth and is generally accepted, or it fails to convince enough people and gradually disappears. Lakatos [1976] does not explore reasons for choosing one concept definition over another. Instead, the teacher in the dialogue simply asks everyone to accept the strictest, i.e., most limited definition suggested so far (at least for the duration of the discussion). However, the only clear end to this process is a tautology (hence a student's sarcastic suggestion that a polyhedron be defined as 'a system of polygons for which the equation $V - E + F = 2$ holds' – [Lakatos, 1976, p. 16]. Dunmore [1992] suggests that concept definitions are chosen and developed according to their use. For instance, a concept which is not associated with any interesting conjectures is unlikely to become well known and accepted within the mathematical community.

We leave the question of whether ambiguity can be used to produce richer theories, i.e., whether a more interesting theory can be produced using the method of monster-barring than without it, for chapter 10.

Can we apply monster-barring to other domains?

We have seen examples of monster-barring applied to number theory, which is a new mathematical domain for this technique. Furthermore, it is a technique often used in everyday arguments about other issues. There are many situations in which participants realise partway through an argument that they are interpreting one of the key terms differently. The meaning of the key term is then called into question, and the focus of the argument switches from the truth or acceptability of a claim or offer to the *meaning* of the term, i.e., they perform monster-barring. This is particularly common in subjects such as philosophy, law and politics, in which persuasive reasoning is all important and concept definitions are modified according to the proponents' goals.

Examples in which ambiguity is used to support an argument include an insurance company which argues that a house damaged in a hurricane is not covered by the owner's accident policy, as a hurricane is an 'act of God' rather than an accident. Similarly, a lawyer may argue that a client who jumped a red light while rushing his wife to a maternity ward is not guilty of

reckless driving. In the 1990's the European Union Food Standards discussed the definition of chocolate. The minimum cocoa content which a substance must contain in order to be called chocolate was debated, as countries which produce it with a higher cocoa content did not want their chocolate to be confused with products with a lower cocoa content. Finally, consider the recent controversy over the meaning of 'prisoners of war'. When challenged that their treatment of the Taliban prisoners violated the Geneva Convention, that all prisoners of war should be treated humanely, the American government argued that the prisoners were not 'prisoners of war' but 'battlefield detainees'[6]. In these examples, the terms *accident*, *reckless driving*, *chocolate* and *prisoners of war* are defined by each party in such a way as to aid their argument. Here we see that examples in non-mathematical domains might be theoretical, definition or ethical. The number of examples in non-mathematical domains suggests that it may be fruitful to test this technique in non-mathematical domains. We test this in chapter 10.

6.8 Related work

We have already seen that research in negotiation is relevant to this method. In this section we briefly outline work in ambiguity and argumentation, and show how our work differs. We also describe work by Skalak and Rissland [1991] on different interpretations of terms.

6.8.1 Work on ambiguity and argumentation

Aristotle [1957, 1955, 1976] identified three motivations behind arguments; *apodiactic*, *dialectic* and *rhetoric*, in which certainty, a general acceptance and convincing an audience are respectively sought. Although many would claim that motivation behind mathematical argument falls into the first category, the second (or even third) is more appropriate to mathematics as Lakatos [1976] describes it. Aristotle treated dialectical argument as a game between a defender and attacker, and suggested guidelines such as forcing the defender to contradict herself, state an untruth or paradox, or to defend a circular argument, for conducting the debate. Certain moves – called fallacies – were disallowed, including the fallacy of ambiguity. The relevant fallacy in our case is the fallacy of the ambiguous middle, which is where an argument consists of two premises and a conclusion, which each contains an occurrence of a term but the term is defined differently in the two premises, and thus the argument is invalid [Hamblin, 1970].

[6]The definition of *humane treatment* was also disputed, in particular whether it could ever include interrogation, as the American government felt it important to interrogate the prisoners while not wanting to be open to the charge of inhumane treatment.

In his work on controversy, Crawshay-Williams [1957] emphasised the need for clarification of concepts prior to discussion. He claimed that if participants in a discussion agree upon the criteria under which a statement will be tested, then agreement regarding its absolute/probable/indeterminate truth will soon be reached. He calls one such criterion *conventional*, to mean the condition that participants agree on the meaning of terms (the others are *logical*, in which inference rules must be agreed, and *empirical*, in which facts and their contextual description should also be agreed).

Naess [1953] also stated that criteria for the verification or falsification of a statement are essential (to the extent that if no such criteria are found then discussion should be abandoned). However, he includes agreeing on terms as a stage in the discussion, rather than a pre-requisite to it. The three stages in resolving a discussion, he suggests, are interpretation, clarification and argumentation. For any statement T there is a set of possible interpretations of T, and participants must agree on which interpretation they wish to discuss. He claims that *precizating* statements, (being more precise) helps to eliminate misunderstandings, where U is more precise than T if any interpretations of U are also interpretations of T, but there are interpretations of T which are not interpretations of U. This is useful only if the disagreement has occurred through different interpretations, and he does not advocate continual precization of statements since discussion would be practically impossible. Naess [1953] calls disagreements which are rooted in misunderstandings, *verbal* disagreements. If, after precization, there is still disagreement, then it is considered *real* disagreement. In the case of a real disagreement, the evidence is weighed up to see which of the two statements is more acceptable.

Cohen [1962] discusses the role that meaning change plays in the development of science, claiming that meaning is not timeless and unchanging. In order to analyse the history of a concept, he argues, the modern historian of science must analyse the history of the meaning of the concept in the context of the culture at different times. Conversely, concept meaning is not all temporal (as suggested by Engels's dialectic); instead, "the philosophy of meaning has to steer a course between these two doctrines of oversimplification"[Cohen, 1962, p. 23].

Carbogim et al. [2000] present a survey of issues which can be handled by automated argumentation systems and suggest directions for future research. They consider the generation and evaluation of arguments, including issues such as drawing conclusions from an incomplete or inconsistent knowledge base, decision making under uncertainty and multiagent negotiation systems. Argument about the meaning of terms used is not considered either explicitly in the text, nor in any of the examples, although it may turn out that semantic negotiation fits into one of the frameworks outlined.

Our view is that ambiguity plays an important role *within* a discussion. That is, questioning

the meaning of terms in a discussion *is* a valid strategy (with restrictions on which words may be questioned). We differ from Crawshay-Williams' approach in that we see debate of terms as an important part of discussion rather than a pre-requisite of it. Participants may not realise initially that they have different interpretations of a word, indeed they may not themselves have a clear interpretation. We also differ from Naess's linear approach: disagreement over terms could arise at any point in a discussion. Of particular interest in Naess's work is the *evidence* used to resolve 'real' disagreements, and we hope by implementing semantic negotiation to elucidate the sort of evidence required. While the survey by Carbogim et al. [2000] is not intended to be exhaustive, it does cover the central issues. The omission of semantic negotiation suggests that it is either irrelevant to argumentation or is a new direction which has received little attention. We hold that it is the latter.

6.8.2 Different interpretations of terms

Rissland has carried out much work on the role of examples in understanding a domain. In particular Skalak and Rissland [1991] discuss a theory of heuristics for making arguments in domains where "A rule may use terms that are not clearly defined, or not defined at all, or the rule may have unspoken exceptions or prerequisites" [Skalak and Rissland, 1991, p 1]. Clearly Lakatos's ideas are relevant to this sort of domain, where Skalak and Rissland's term *rule* corresponds to Lakatos's term *conjecture*, *term* to *concept*, *case* to *entity*, and *argument* to *proof*. In particular, Skalak and Rissland [1991] are interested in cases where terms within a rule are open to interpretation, and different parties will define the term differently according to their point of view. This corresponds very closely to Lakatos's method of monster-barring. As an example, Skalak argues that the term 'satisfactory progress' might be interpreted differently by a student than by a university, in the rule 'a student must make satisfactory progress towards a degree', with the student more likely to define it widely, thus including her own progress as satisfactory. Skalak and Rissland [1991] discuss argument moves which use cases to determine which interpretation of an ambiguous term in a rule is to be adopted. These moves are implemented within CABARET [Rissland and Skalak, 1991] which is a domain-independent architecture for combining a production system with a case-based reasoner.

McNeill et al. [2004] describe a system that can dynamically discover some ontological mismatches between agents during communication and then refine them, in order to enable communication between these agents. For instance, an agent may attempt to buy a ticket by locating and communicating with a ticket selling agent. Mismatches in representation between agents may arise during communication, such as the main agent representing money as a unary predicate $money(?Amount)$, and the ticket-selling agent representing it with the binary predicate

money(?*Amount*, ?*Currency*). McNeill et al. suggest techniques to diagnose and repair some such mismatches, and describe their system within which they have implemented these techniques.

Our work differs in motivation and application. Neither Skalak and Rissland [1991] nor McNeill et al. [2004] mention Lakatos; rather the motivation of the former is to model legal reasoning, and the latter to model representational change in the planning domain. Although there is an analogy between the method of monster-barring and their work, Lakatos's other methods are not modelled. With regard to application, Skalak and Rissland [1991] specifically contrast domains which include rules with terms which are open to interpretation, with mathematics: "Statutory interpretation can be contrasted with fields like mathematics, where the application of a rule to a set of facts is a straightforward application of an inference rule like *modus ponens*" [Skalak and Rissland, 1991, p 1]. One important different between our work and that of [McNeill et al., 2004] is the role that counterexamples play. Representational mismatch is not seen in terms of counterexamples by McNeill et al. [2004], whereas counterexamples are vital to our project.

6.9 Summary

In this chapter we have developed our computational reading of Lakatos's method of monster-barring, and introduced the converse: monster-accepting. We started by outlining Lakatos's work, and his views of the method. We then looked for other examples of the method, in the new domain of number theory, and argued that it can be seen at play in mathematician's attitudes to the concept of *number*, and 'monsters' including the number one, zero, irrational, imaginary and transfinite numbers. In §6.3, we outlined six key points in monster-barring and then described our implementation of each of these in §6.4 and our algorithms in §6.5. We presented illustrative sessions in §6.6 and discussed our approach in §6.7. We concluded the chapter with a discussion of related work in the fields of ambiguity, argumentation and law.

Chapter 7

The method of exception-barring

> "... I am ... against the term '*counterexample*'; it rightly admits them on a par with supporting examples, but somehow paints them in war colours, so that, ..., one panics when facing them, and is tempted to abandon beautiful and ingenious proofs altogether. No: they are just *exceptions*." *Beta*, student in [Lakatos, 1976, p.24].

This chapter discusses our computational reading of Lakatos's treatment of exceptions. This is noteworthy for two reasons. Firstly, their role in mathematics, traditionally thought of as an exact subject in which the occurrence of exceptions would force a mathematician to abandon a conjecture (§7.1). Secondly, Lakatos showed how exceptions, rather than simply being annoying problem cases, can be used to further knowledge (§7.2). In §7.3 we outline the method of exception-barring. In order to implement the method, we consider how exception-barring applies to another area of mathematics: number theory, in §7.4, and to different types of conjecture, in §7.5. This exploration suggests further distinctions in the method. In sections 7.6, 7.7 and 7.8 we describe our algorithms and example sessions for each of the exception barring methods: piecemeal exclusion, counterexample barring (which is a type of piecemeal exclusion) and strategic withdrawal. Note that unless otherwise stated, we use the default settings in HR during the example sessions. In particular, the interestingness measures we use are the default weightings. We conclude by considering the work from the fields of machine learning, diagrammatic reasoning, and planning, which is related to this method (§7.9).

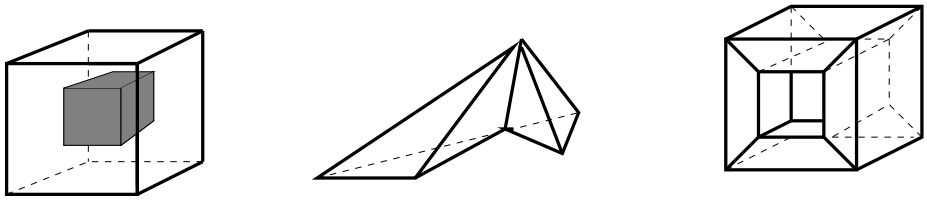
7.1 Exceptions in Mathematics

Lakatos was one of the first to make explicit the role of exceptions within mathematics. Previously (and often still today), it was thought that while in other subjects statements may be

susceptible to exceptions, mathematics was ‘pure’ in the sense that if a conjecture was found to have a counterexample then that was sufficient to prove it false. A mathematician would then move on to other, potentially provable conjectures. The idea that exceptions in mathematics could be used, as in other domains, to modify, as opposed to reject, faulty conjectures was greeted with horror, hence *Delta’s* argument that:

> “To agree to a peaceful coexistence of theorems and exceptions means to yield to confusion and chaos in mathematics.” *Delta*, student in [Lakatos, 1976, p.25].

However, Lakatos, by documenting a theoretical development of Euler’s conjecture, that *for all polyhedra, the number of faces (V) minus the number of edges (E) plus the number of faces (F) is 2*, showed just this: that mathematicians found the counterexamples of nested cubes, picture frame and twin tetrahedra and used them to modify the conjecture (see figure 7.1). This was done by generalising from the examples to the general case of polyhedra with cavities, tunnels and multiple structure, and then modifying the conjecture to *for all polyhedra except those with cavities, tunnels or multiple structure, V - E + F = 2*. Similarly, by examining the supporting examples, such as the cube, tetrahedron and octahedron (see figure 7.2), mathematicians re-treated to the ‘safe’ domain of convex polyhedra (topologically equivalent to a sphere). In a footnote, [Lakatos, 1976, p.28], Lakatos cites Abel restricting the domain of suspect theorems about functions to power series as another mathematical example of retreating to a safe domain when faced with counterexamples.



HOLLOW CUBE
16 - 24 + 12 = 4

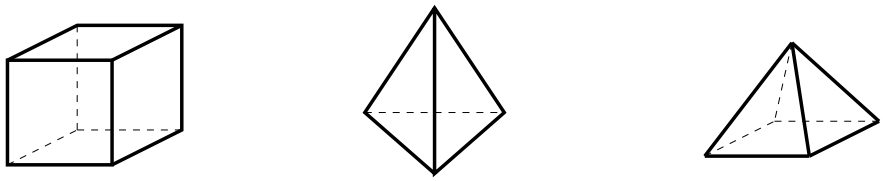
TWIN TETRAHEDRA
6 - 11 + 8 = 3

PICTURE FRAME
16 - 32 + 16 = 0

Figure 7.1: Counterexamples to Euler’s conjecture - for all polyhedra, V-E+F = 2

7.2 Using exceptions to further knowledge: their uses and limits

Lakatos showed how exceptions, or counterexamples, rather than simply being problem cases to be grudgingly dealt with, can be used to enhance understanding of a domain. Studying counterexamples prompts questions such as ‘what is it about the counterexamples which makes



CUBE
8 - 12 + 6 = 2

TETRAHEDRON
4 - 6 + 4 = 2

OCTAHEDRON
5 - 8 + 5 = 2

Figure 7.2: Supporting examples for Euler's conjecture; these are all convex

them false?', and 'what is it about the positive examples which makes them true?' Lakatos captured both these questions with his two methods of exception-barring. Lakatos [1976] showed how counterexamples can assist in understanding and improving a conjecture, and he suggested ways in which this can be done.

> "One should not confuse false theorems with theorems subject to some restriction." *Sigma*, student, quoting *Bérard*, [Lakatos, 1976, p.24].

> "I shall use [exception-barring] to determine precisely the domain in which the Euler conjecture holds." *Beta*, student in [Lakatos, 1976, p.26].

Exception-barring, however, is one of the less sophisticated methods in [Lakatos, 1976]. It is naïve in the sense that it doesn't use the 'proof' of a faulty conjecture to improve upon the conjecture, rather depending on examination of examples and counterexamples. Thus it is presented as an *ad hoc* method, which can never lead to a conjecture being accepted as a theorem as one cannot be sure that more counterexamples will not be found.

> "You *improved* the original conjecture, but you cannot claim to have *perfected* the conjecture, to have achieved perfect rigour in your proof." *Teacher*, in [Lakatos, 1976, p.30].

7.3 Lakatos's two methods of exception-barring

Generalising from the two approaches above, whereby counterexamples are examined and then excluded, and supporting examples examined and the domain of the conjecture limited to these, Lakatos identified two types of exception-barring: *(i)* piecemeal exclusion, and *(ii)* strategic withdrawal. Piecemeal exclusion works by considering the counterexamples and finding a concept which covers only these, then modifying the conjecture by excluding entities covered

by the concept; strategic withdrawal by considering the positive examples of a conjecture, finding a concept which covers only these, and limiting the domain of the conjecture to that of the concept. That is, given conjecture $\forall x.P(x) \rightarrow Q(x)$, a set of counter (or negative) examples *Neg* such that $\forall n \in Neg$, $P(n) \wedge \neg Q(n)$, and a set of positive examples *Pos* such that $\forall p \in Pos$, $P(p) \wedge Q(p)$:

(i) find a concept C such that for all n, $C(n)$, and for all p, $\neg C(p)$, and modify the conjecture to $\forall x\ ((\neg C(x) \wedge P(x)) \rightarrow Q(x))$, and

(ii) find a concept C such that for all p, $C(p)$, and for all n, $\neg C(n)$, and modify the conjecture to $\forall x\ ((C(x) \wedge P(x)) \rightarrow Q(x))$.

7.4 Exception-barring in Number Theory

Many conjectures and theorems in number theory have stated exceptions. For instance:

- all even numbers *except 2* are the sum of two primes (Goldbach's conjecture);
- all primes *except 2* are odd;
- all integers *except squares* have an even number of divisors, and
- all integers *except those of the form* 2^k can be written as the sum of 2 or more consecutive integers

are examples where piecemeal exclusion could feasibly have been used. Similarly, many are limited to a certain domain, for instance:

- for every number $n \geq 3$, $x^n + y^n = z^n$ has no integer solutions (Fermat's Last Theorem);
- every *odd* number > 5 is the sum of three primes (the Odd Goldbach Problem), and
- every *even* number is the difference of two primes

can be seen where examples of strategic withdrawal could feasibly have been used. Note that, as with Lakatos, we do not claim that these conjectures/theorems *were* found by using exception-barring, but that they could be.

Applying exception-barring to number theory highlights the fact that sometimes both methods result in the same conjecture refinement. For instance, we could present Fermat's Last Theorem

as an example of piecemeal exclusion, if we phrase it as '$x^n + y^n = z^n$ has no integer solutions, *except for* $n \leq 2$', or the conjecture that 'all integers *except squares* have an even number of divisors' as 'all non-squares have an even number of divisors'. To formalise a simple notion of this, let us see concepts as sets, and conjectures as statements which link the sets. Suppose that we have sets P and Q, and conjecture $\forall x, x \in P \rightarrow x \in Q$, and two further sets, S_1 and S_2, such that $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = P$. Then the conjecture refinement:

$\forall x, x \in (P \cap S_1) \rightarrow x \in Q$ is the same as

$\forall x, x \in (P \cap \neg S_2) \rightarrow x \in Q$,

since the sets $P \cap S_1$ and $P \cap \neg S_2$ are equivalent. This can be clearly seen in the conjecture above that "every even number is the difference of two primes", where P is the set of numbers, Q the set of numbers which are the difference of two primes, S_1 the set of even numbers, and S_2 the set of odd numbers.

We also notice that sometimes counterexamples are *listed* in the conjecture, as in "all even numbers *except 2* can be expressed uniquely as the sum of two primes", and "all primes *except 2* are odd". Therefore, a special case of piecemeal exception barring, is to bar a specific example (or examples). We call this counterexample-barring.

7.5 Exception-barring and other types of conjecture

Exception-barring in Lakatos [1976] is applied only to one conjecture, that one concept (polyhedra) almost implies another (shapes which satisfy the Euler equation). We represent this as:

$$poly(x) \rightsquigarrow euler(x).$$

There are many other types of conjecture in mathematics. Some of these types have been categorised and an ability to make them has been added to HR by [Colton, 2001a, chapter 7] as:

- Equivalence - the definitions of two concepts are logically equivalent ($P \leftrightarrow Q$)
- Implication - one concept is a specialisation of another ($P \rightarrow Q$)
- NonExists - no examples satisfy the definition of a given concept ($\nexists x$ st $P(x)$)

- Applicability - examples satisfying a definition are restricted to a particular finite set

This raises two questions:
1) How can we apply exception-barring to these types of conjecture, and what sort of conjectures would result?
2) Should we implement other types of conjecture in HR, in order to fully exploit our exception-barring method, and if so, which ones?

These are useful questions to consider as they provide a way of analysing, understanding, and extending usage of the methods, which accords with our goal of clarifying and extending the methods. Additionally, answering these questions will hopefully enable us to extend HR's theory formation abilities, thereby supporting our hypothesis that Lakatos's methods are useful in automated theory formation.

Applying exception-barring to HR's conjectures

We cannot directly apply exception-barring to any of HR's conjectures as they are only made if they hold empirically for all of the objects of interest in HR's theory, and Lakatos's methods require known counterexamples. HRL partially gets around this, as the agents have access to different databases, and therefore one agent makes a conjecture which holds for all the examples in its database and communicates it to others who do have counterexamples, and can then apply Lakatos's methods. This is only partially the case as, following the approach of Colton et al. [2000a], when an agent sends a conjecture to a second, the second has to reconstruct the conjecture (and its concepts). This is so that the receiving agent has the relevant data table and categorisation for its entities, and can evaluate the interestingness independently of the first agent (see §3.1 for more details). If the second agent does have counterexamples, it will not reconstruct the conjecture as any of the above types (which can only be constructed if the conjecture holds empirically for all the examples in the agent's database). However, if we ignore this for a moment, it is worth noting some general points about exception-barring and these types of conjecture. We then consider which new types of conjecture HR does need in order to represent conjectures with known counterexamples.

Equivalence

Recall that in §3.1.1, we stated that one way of representing concepts in Colton [2001a] is as a data table, i.e. examples with corresponding values (e.g., *prime(5) = [true]*, or $\tau(5) = 2$). We can represent an equivalence or implication conjecture as two sets of examples (corresponding to the two concepts) where the intersection contains those examples which share the same values. The example in Lakatos [1976] is represented as such in figure 7.3:

As this example is an implication, we only have counterexamples on one side, i.e., in one set. However, for equivalences, we may get counterexamples on both sides, in which case

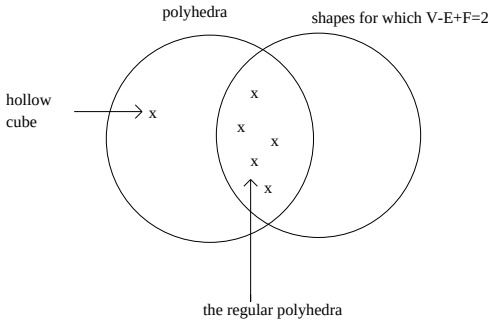


Figure 7.3: A Venn-diagram representation of the conjecture: for all polyhedra, V-E+F = 2

we might want to apply exception-barring twice, resulting in two implication conjectures. For instance, from the faulty conjecture in figure 7.4, "$\forall x.prime(x) \leftrightarrow odd(x)$" (shown in figure 7.4 for integers 1-10), we could use piecemeal exclusion to get the concepts *primes except 2* and *odd non-square numbers*, and the conjectures "all primes except 2 are odd" (which is true) and "all odd non-square numbers are prime" (which is false, the first counterexample is 15).

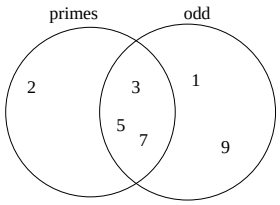


Figure 7.4: primes $\leftrightsquigarrow$ odds

Implication

All three methods: piecemeal exclusion, counterexample-barring and strategic withdrawal would result in another implication conjecture. Given an implication conjecture $P \rightarrow Q$, and concept $C1$ which covers the counterexamples, (possibly being a conjunction of the individual counterexamples), or concept $C2$ which covers the supporting examples, then the modified conjecture would take the form $P \wedge \neg C1 \rightarrow Q$ (in the case of piecemeal exclusion or counterexample-barring), or $P \wedge C2 \rightarrow Q$ (in the case of strategic withdrawal. That is, the modification would take the form of another implication conjecture.

NonExists:

Both strategic withdrawal and piecemeal exclusion would result in finding a more specialised concept than P, say P', and another nonexists conjecture, $\nexists x.P'(x)$. Counterexample barring would result in a conjecture which could be represented as an applicability conjecture, i.e. $P(x) \rightarrow x \in S$, where S is the set of counterexamples to the nonexists conjecture. This is the strategic withdrawal version of counterexample barring, where we retreat to a safe domain

which is so small (or we know of no concept which covers the positives), that we name each positive example in the conjecture itself.

Applicability
All three methods would result in another applicability conjecture. If the flawed conjecture is that examples satisfying a definition D are restricted to a particular finite set S, then counterexample barring would result in adding the counterexamples to the set S. Both piecemeal exclusion and strategic withdrawal would result in changing the definition: to $D \wedge \neg C1$ in the case of piecemeal exclusion, and $D \wedge C2$ in the case of strategic withdrawal.

Implementing new types of conjecture in HR
Recall that we have introduced two new types of conjecture into HR: near-equivalences and near-implications, to enable an agent to represent conjectures with known counterexamples. This was described in §4.4. These conjectures are like their empirically true counterparts, but allow a certain proportion of counterexamples. These can be used as a way of generating conjectures initially, in which case the user inputs the maximum proportion of counterexamples allowed. Their main function, however, is to represent conjectures which other agents send which do not hold for the entities in the receiving agent's theory. In this case, the proportion of counterexamples, and the set of counterexamples, is recorded along with the conjecture but there is no limit on the number of counterexamples acceptable.

The running example in [Lakatos, 1976] of Euler's conjecture relates to a concept which has, in the terminology of HR, arity one. That is, either a shape is a polyhedron or not, and its Euler characteristic is either two or not. So far, we limit ourselves to the same kind of concept, i.e. we stipulate that any conjecture suggested for modification must involve only concepts with arity one. Hence, near-equivalence and near-implication conjectures can currently only be made if the two concepts in the conjecture have arity one, i.e. they have positive or negative examples, as opposed to functions which have corresponding values for each entity.

7.6 Piecemeal Exclusion

Suppose that an agent has received a request from the teacher to modify a given conjecture which the agent has reconstructed to make either a near equivalence or a near implication, for instance, the near-equivalence conjecture $P \nleftrightarrow Q$. Suppose also that the agent is set to perform piecemeal exclusion. It will first retrieve the set of counterexamples associated with the conjecture, and then test to see whether the counterexamples are covered by P or Q, i.e., whether they exhibit the characteristics specified by the concept. Supposing that the counterexamples

are covered by P, the agent will look in its theory to see whether it already has a concept X which exactly covers those counterexamples covered by P, i.e., has a positive value for all entities covered by P and negative for all other entities. If so, the agent then uses the production rule *negate*, with concepts X and P and no parameters, to form the new concept $P \wedge \neg X$. It then makes the appropriate conjecture modification by making a new equivalence or implication conjecture. As there may be more than one modification to a given faulty conjecture, it sends back a (possibly empty) vector of modifications to the teacher.

See algorithm 5 for our piecemeal exclusion algorithm.

Algorithm 5 The piecemeal exclusion algorithm

Require: given a near equivalence $P \nleftrightarrow Q$:

1: Get the list of counterexamples, and determine whether each counterexample is covered by concept P or concept Q.
2: **if** all counterexamples are covered by P **then**
3: find a concept X in the theory which exactly covers all of the counterexamples
4: form the new concept $P \wedge \neg X$
5: add the new concept to the agenda, which will then form the conjecture $P \wedge \neg X \leftrightarrow Q$
6: return the conjecture
7: **else if** all counterexamples are covered by Q **then**
8: as for case 1, but instead form the concept $Q \wedge \neg X$ and the conjecture $P \leftrightarrow Q \wedge \neg X$
9: **else if** there are counterexamples covered by both P and Q **then**
10: find a concept X in the theory which exactly covers the counterexamples in P
11: form the concept $P \wedge \neg X$
12: add the new concept to the agenda, which will then form the conjecture $P \wedge \neg X \rightarrow Q$
13: do the same for Q
14: return both conjectures
15: **end if**

7.6.1 Illustrative session

Using three student agents and a teacher, we demonstrate the formation of the conjecture that "an integer is non-square if and only if it has an even number of divisors".

Initial information

Student 1 started with entities $1-10$, and core concepts integer, divisor and multiplication. It was set to use piecemeal exclusion. Before running it independently we forced Student 1 to form the concepts *square number* and *integers which have an even number of divisors*, in

the following way: starting with the concept of divisors it uses the *split* production rule with parameters [[1], [2]] to reinvent the concept of an even number. Again, starting with the concept of divisors it uses *size* [1] to get the concept number of divisors of an integer. It then takes these two concepts and the production rule *compose*, with parameters [s, 0, 1], and then *exists* [1] to get the concept integers with an even number of divisors. Finally, from the core concept of multiplication it uses *match* with parameters [0, 1, 1] and then exists with parameters [1] to get the concept of squares.

Student 2 started with the same input as Student 1, except with integers $11-50$.

Student 3 started with integers $51-60$, core concepts integer, divisor and multiplication, and forced concept of integers which have an even number of divisors.

Session details
The teacher sent a request for all students to work independently for 20 steps, and then send back their best implication conjecture. Student 3 formed the conjecture that all integers have an even number of divisors, and sent it to the teacher, who puts it on the group agenda for modifications. The other students found counterexamples [1,4,9] and [16, 25, 36, 49] respectively. They then found the concept of squares and formed the new concept non-squares. They used this new concept to form the conjecture that all non-squares have an even number of divisors, which is a theorem in number theory.

7.7 Counterexample barring

This is a special case of piecemeal exclusion and can be used when an agent does not have a concept X to cover the counterexamples to a faulty conjecture, or if the counterexample set is small, e.g., containing just one example. In order to build a concept in which no more information is known than the positive entities, we have written another production rule, *entity-disjunct*. This takes as input the entities which the concept expresses as parameters, and the object of interest concept with the same type as the entities in the parameter list. *Entity-disjunct* partially applies the production rule *split* with parameter $[0]$ to the object of interest concept and each of the entities in turn, and then partially applies the rule *disjunct* to the resulting concepts. We have implemented this as a production rule, rather than using a combination of *split* and *disjunct*, for efficiency reasons (the partial implementation ensures that we get the right specifications and hence definition, without storing superfluous information about the concept before it has been fully formed). Additionally, it reflects the degree of complexity of the concept, which is used as a measure of interestingness. The concept of being the number 2 is

not complex, and therefore we want to reflect this in our measure, but had it taken 5 production rules to generate it, it would have been evaluated as fairly complex. To avoid a concept which is just a long list of entities, we limit the number of entities which form the concept, with the default maximum set to three. (This number may seem, and indeed is, an arbitrary number, but *entity-disjunct* is designed to generate a concept which consists of only a few specific entities. See §13.5 for ideas on how to develop a more cognitively plausible concept representation.)

As an example of how the entity-disjunct production rule works, suppose that an agent has the integers 0 - 20 in its database, and the core concept of integers, denoted by int001. If the agent performs the theory formation step: *int001 entity-disjunct* $< 1 >$, i.e., it applies the entity-disjunct production rule, with parameter $< 1 >$, to the concept int001, then the result is the concept "a is an integer $\wedge$ a=1". We show the input and output data table in table 7.1 below.

Input	
integer	
0	true
1	true
2	true
3	true
4	true
.	.
.	.
.	.
17	true
18	true
19	true
20	true

$\rightarrow$

Output	
integer	
0	false
1	true
2	false
3	false
4	false
.	.
.	.
.	.
17	false
18	false
19	false
20	false

Table 7.1: The entity-disjunct production rule, with parameter $< 1 >$

The agent applies *entity-disjunct* to the object of interest concept, with the vector of counterexamples as parameters, to get, for instance, the concept of being 2. If the outcome is an equivalence conjecture then the agent returns this. However, if the outcome is a concept, then the agent applies the production rule *negate* with no parameters to the concept and the object of interest. If the previous step had returned the concept of being 2, this would now return the concept of being *not* 2. The resulting concept is then combined with the problem concept to get, for instance, the concept of primes except 2. If the faulty conjecture was a near equivalence then a new equivalence conjecture is now constructed and returned, and if it was a near implication then the modified implication conjecture is constructed and returned.

7.7.1 The counterexample-barring algorithm

We describe our counterexample-barring algorithm for near-implication conjectures in algorithm 6. Given a list of entities, this algorithm returns a concept for which the entities in the list are positives and all other entities in the theory are negatives. It works by taking the object of interest concept, (and if there is more than one object of interest concept, by taking the one which includes the entities in the given list), and applying the production rule *entity-disjunct*. It returns the concept which follows from this application.

Algorithm 6 The counterexample-barring algorithm

Require: Given a near implication $P \rightsquigarrow Q$, with counterexamples $[x_1,...,x_n]$, where $n \leq 3$, and there is no concept in the theory to cover exactly these entities, then:

1: find the object of interest concept with the same type as $x_1,...,x_n$.
2: apply the production rule *entity-disjunct* to this concept, with parameters $[x_1,...,x_n]$ to get the concept of being x_1 or ... or x_n.
3: apply the production rule *negate* with no parameters to P and the object of interest concept, to get the concept of *not* being x_1 or ... or x_n. Call this concept X'.
4: combine X' and P using *compose*, to get $P \wedge \neg(x_1 \vee ... \vee x_n)$
5: make the conjecture $P \wedge \neg(x_1 \vee ... \vee x_n) \rightarrow Q$
6: return the conjecture

In the case of near-equivalence conjectures: if there are counterexamples on one side of the conjecture only, then steps 1 to 4 are the same as those shown in algorithm 6, and step 5 returns an equivalence conjecture. If there are counterexamples on both sides of the conjecture, e.g., x,y,z in P and x',y',z' in Q, then steps 1 to 4 are carried out twice (firstly on P, then on Q), and an equivalence $P \wedge \neg(x \vee y \vee z) \leftrightarrow Q \wedge \neg(x' \vee y' \vee z')$ is made and returned.

7.7.2 Illustrative sessions

Session one: Using two students and a teacher, we get the conjecture that *all primes except 2 are odd*.

Initial information

Student 1 started with integers 1-10, and core concepts integer and divisor. It was set to make implications from subsumptions and to use piecemeal exclusion. Before running it independently we forced the concepts prime and odd number in the following way: apply *split* with parameters [[1], [2]] to the concept of divisor, to get the concept of even numbers, then *negate* [] to the concepts even numbers and integers to get odd numbers. Then apply *size* [1] to the

concept of divisors, to produce the number of divisors of an integer, and *split* [[1], [2]] to this to produce the concept of prime numbers.

Student 2 had the same information, except integers $11 - 20$ instead of integers $1 - 10$.

Session details
The teacher sent a request for the students to work independently for 20 steps, and then send back their best implication conjecture. Student 2 made and sent the conjecture that all primes are odd, which the teacher put on the group agenda for discussion. Student 1 found the counterexample 2, looked for a concept to cover it and failed, so makes the concepts of being 2, integers except 2, then primes except 2, and finally constructs the conjecture that all primes except 2 are odd.

Session two:
Using two students and a teacher, we got the conjecture that *all even numbers except 2 are the sum of two primes* (Goldbach's conjecture).

Initial information
Student 1 started with integers $1 - 10$ and core concepts integers and divisors. It was set to make implications from subsumptions and to use piecemeal exclusion. It forced the concepts even numbers and integers which are the sum of two primes in the following way: get even numbers and primes as above, then *compose* [0, 1, 0] on addition and prime to get the concept of being the sum of two numbers, one of which is prime, and *compose* [0, 0, 1] on prime and this concept, then *exists* [1, 2] on the resulting concept to get the concept of being the sum of two primes.

Student 2 started with the same information except integers $11 - 20$ instead of integers $1 - 10$.

Session details
The teacher sent a request for the students to work independently for 20 steps, and then send back their best implication conjecture. Student 2 formed the conjecture that all even numbers can be expressed as the sum of two primes. The teacher sent a request for modifications, and Student 1 found the counterexample 2, and made the concept 'even numbers except 2' and the conjecture that 'all even numbers except 2 are the sum of two primes'.

7.8 Strategic Withdrawal

Strategic withdrawal is the easiest of the exception-barring methods to implement, because it does not involve making new concepts. However, it is perhaps the least useful, for the same

reason. It consists of considering the positive examples for which a conjecture holds, looking to see if there if a concept to cover them, and if so, forming a new conjecture with it.

See algorithm 7 for our implementation of strategic withdrawal.

Algorithm 7 The strategic withdrawal algorithm

Require: Given a near-equivalence $P \leftrightsquigarrow Q$:

1: Get the list of counterexamples, and determine whether each counterexample is covered by concept P or Q.
2: **if** all counterexamples are covered by P **then**
3: find a concept X in the theory which exactly covers the positive examples (and is different from Q)
4: make the conjectures $X \leftrightarrow Q$; $X \rightarrow Q$; $X \rightarrow P$
5: return the conjectures
6: **else if** all counterexamples are covered by Q **then**
7: As for case 1, instead forming the conjectures $X \leftrightarrow P$; $X \rightarrow P$; $X \rightarrow Q$, where the concept X is different from P
8: **else if** there are counterexamples which are covered by both P and Q **then**
9: find a concept X in the theory which exactly covers the positives
10: make the conjectures $X \rightarrow P$, and $X \rightarrow Q$
11: return the conjectures
12: **end if**

7.8.1 Illustrative sessions

Using two students and a teacher, we get the conjecture that *all odd non-squares are prime*.

Initial information

Student 1 started with integers $1-10$, and core concepts integer, less than or equal, divisor, multiplication and addition. It was set to make near equivalences which hold for 60% of the objects of interest and to perform strategic withdrawal.

Before running it independently we forced the concepts prime and odd number in the way described above (§7.7.2, example 1)

Student 2 had the same information, with the forced concept odd non-squares, where the concepts odd number and square number are forced as described in §7.6.1, non-square number is forced by applying *negate* [] to square and integer, and the concept of odd non-square numbers produced by applying *compose* [1] to the concepts odd number and non-square number.

Session details

The teacher sent a request to the students to work independently for 20 steps, and then send back their best near equivalence conjecture. Student 1 formed the near equivalence prime $\leftrightsquigarrow$ odd, the teacher then asked for modifications and Student 2 found the counterexample [2], which is prime and not odd, and [1,9] which are odd but not prime. It looked for a concept to cover the positives, [3,5,7] and found the concept odd non-square, then makes the conjectures odd and non-square $\rightarrow$ odd (which is a tautology) and odd and non-square $\rightarrow$ prime (this is false; the first counterexample is 15).

7.9 Related work

7.9.1 The boosting technique

The methods of exception-barring, by looking at supporting examples and counterexamples, are clearly related to machine learning techniques. For instance, boosting, described in [Freund and Schapire, 1999] and [Schapire, 2002], is a technique in which there are are several learners, which use possibly different algorithms to learn a classification task. The boosting algorithm is a meta-algorithm which forms a classification rule by selecting a subset of a training set, a sample set, and giving it as input to the first learner. It then looks at the output of the learner, its 'weak' rule, and the examples which the learner misclassified (i.e., counterexamples). The boosting algorithm increases the weighting of these counterexamples in the training set, thus increasing the likelihood that these examples are selected for the sample given as input to the next learner. Based on this input, the next learner generates another weak rule for the same classification task. The boosting algorithm then further increases the weighting of any examples which both the first and the second learner misclassified. Another sample from the training set is then selected for a third learner, and so on. The cycle continues until it arrives at some point, for instance, all of the learners have been consulted. The boosting algorithm then combines the weak rules into a single weighted sum, where the weight of each rule is determined by its accuracy.

This process is similar to piecemeal exclusion and counterexample-barring in that it involves different 'agents' forming a rule (or conjecture) about the examples in their databases, and then attempting to improve the rule by focusing on counterexamples. However, it differs in the following ways. Firstly, the examples which a learner has may differ, over continued application of the cycle by the boosting algorithm. That is, the same learner may be asked for another weak rule for the same classification and given different examples as input for the rule. This contrasts with our system, in which the set of examples which an agent has may be added to, but

otherwise remains constant. A second difference is that in boosting there might not be a fixed number of learners: further learners may be added until a given accuracy is achieved. Thirdly, the boosting algorithm only raises the likelihood of a counterexample being selected as input, whereas piecemeal exclusion and counterexample-barring focus purely on counterexamples. The most important difference is that in boosting, the weak rule suggested by the learners is generated independently of previous weak rules. Although the final rule is a combination of these weak rules, it is not the case that the first is modified. Conversely, exception-barring explicitly builds upon a faulty conjecture by stating the exceptions and formally excluding them in a new expression of the conjecture. The criteria to which each technique refers differ: the final rule produced by the boosting algorithm is judged solely on accuracy, whereas other qualities are sought in the final conjecture produced by exception-barring, such as how simple it is, or how surprising. Finally, the goal for which the technique is used is different: the goal of the boosting algorithm is to accurately predict an example, whereas our goal is to build a theory. These are different intelligent tasks and therefore a detailed comparison would be inappropriate.

7.9.2 Work in diagrammatic reasoning

Winterstein [2004] looked at using diagrammatic reasoning to prove mathematical theorems in the continuous domain of analysis. He argued that one approach, his *generalisation method*, can be seen as a simple form of Lakatos's method of strategic withdrawal [Winterstein, 2004, p. 69]. This method is one of the methods which Winterstein has devised for representing and reasoning with diagrams. It works by firstly proving a theorem in a specific case, for instance by instantiating variables x and y in a theorem and then proving the theorem in this case. The second step is to analyse the proof to check which cases it can be applied to, i.e. to examine whether the proof depended on the values of x and y and to show that the values were arbitrary or at least arbitrary within certain constraints. The final step is to generalise the proof to all cases where the chain of reasoning is guaranteed to be valid. Winterstein argues that this is a form of strategic withdrawal since it analyses positive examples of a proof, abstracts the key features from these examples, and then restricts the domain of application of the theorem and proof accordingly. Our work differs in that it is a systematic computational reading of Lakatos's theory, rather than a single method.

7.9.3 Hayes-Roth's heuristics for repairing flawed plans

Hayes-Roth [1983] describes 5 heuristics for repairing flawed beliefs, which are based on Lakatos's methods. These constitute the only attempt that we have seen, to produce a com-

puter model which is inspired by Lakatos's methods. While two of Hayes-Roth's techniques correspond to surrender and monster-barring, we include his work here since the remaining three correspond to exception-barring.

Hayes-Roth explains his heuristics in terms of revising a flawed strategy in a card game, where the counterexample corresponds to a move which follows the strategy but which does not have the desired outcome. Below we express Hayes-Roth's terminology in Lakatos's terms, with examples where relevant.

Lakatos's terminology	**Hayes-Roth's terminology**
conjecture:	a plan – if conditions C hold and action A is executed then effects E will result i.e. $C \wedge A \rightarrow E$
proof:	justification of a plan (e.g. because this plan will minimise the number of my points)
entity:	an action which is executed according to a plan (e.g. play the 9 spades)
counterexample:	an action which is executed according to a plan but which fails to achieve the goal i.e. $C \wedge A \wedge \neg E$
concept:	a concept (e.g. a spade lower than the Queen)

Hayes-Roth demonstrates his heuristics with reference to the card game hearts (which is similar to whist). The pack is divided amongst players, then one player plays a card and the others must all put down a card in the same suit as the first if they have one, and otherwise play any card. The person who played the highest card in the specified suit wins that trick and starts the next. One point is awarded for each heart won in a trick, and 13 for the queen of spades (QS). The aim of the game is to get either as few points as possible ("go low") or all the points ("shoot the moon"). A strategy which beginners sometimes employ is to win a trick to take the lead, and then play a spade in order to flush out the QS and avoid the 13 points. Hayes-Roth represents this as shown (p.230):

Plan:	Flush the QS
Effects:	(1) I will force the player who has the QS to play that card
	(2) I will avoid taking 13 points
Conditions:	(1) I do not hold the QS
	(2) The QS has not yet been played
Actions:	First I win a trick to take the lead, and whenever I lead I play a spade

The plan (analogous to a faulty conjecture) may backfire if the beginner starts with the king of spades (KS) and then wins the trick and hence the unwanted points (this situation is a counterexample to the plan). The heuristics then provide various ways of revising the plan:

1) **Retraction** (like surrender) - retract the part of the plan which fails, in this case effect (2).

2) **Exclusion** (like monster-barring) - bar the theory from applying to the current situation, by excluding the situation. Add the condition *I do not play KS.*

3) **Avoidance** (like piecemeal exclusion) - rule out situations which can be predicted to fail the plan, by adding conditions to exclude them. For example by assessing why the plan failed add the condition *I do not win the trick in which the queen of spades is played*. A system can further improve its plan by negating the new condition - *I win the trick in which the queen of spades is played*, using this and its knowledge of the game to infer that it must play the highest card in the specified suit, and then negating the inference to get *I must not play the highest card in the specified suit*. This is then incorporated into the action which becomes *First I win a trick to take the lead and whenever I lead, I play a spade which is not the highest spade*.

4) **Assurance** (like strategic withdrawal) - change the plan so that it only applies to situations which it reliably predicts. In this case the faulty prediction is effect (2), and so the system looks for conditions which guarantee it. It does this by negating it, inferring consequents and then negating one of these and incorporating it into the action. For example negating effect (2) gives *I do take 13 points*, the game rules state that *the winner of the trick takes the points in the trick* so we can infer that *I win the trick*, then use this and the rule that *the person who plays the highest card in the suit led wins the trick* to infer that *I play the highest card in the suit led*. Given that *player X plays the QS* we can now infer that *I play a spade higher than the QS* and negate it to get *I play a spade lower than the QS*.

5) **Inclusion** (also like strategic withdrawal) - this differs from assurance in that the situations for which the plan is known to hold are listed rather than a new concept being devised. Therefore instead of adding *I play a spade lower than the QS* to the action, we add *I play a spade in the set* $\{2ofspades, 3ofspades, 4ofspades..., 10ofspades, jackofspades\}$.

Hayes-Roth argues that these heuristics can be implemented using existing techniques (although adding that this may take considerable effort). The primary capabilities, he claims, are symbolic deduction and heuristic search. He suggests ways in which one heuristic may be preferred over another, in order to avoid a combinatorial explosion. These include preferring general to specific theories, seeking canonical representations and experimentally evaluating alternative fixes to determine the most fruitful.

Our work differs from [Hayes-Roth, 1983] in several ways. Firstly, his motivation is different. Hayes-Roth uses Lakatos's ideas as inspiration for modifying plans, rather than attempting a faithful model of Lakatos's work. Secondly, Hayes-Roth differs in his application of Lakatos's methods, which is to machine learning tasks as opposed to theory formation. Although these two fields are linked, as we argued above (§7.9.1), they differ in that typically the output from a machine learning program would be a classification, or concept definition which has been induced from some positive and possibly some negative examples; whereas output from a theory formation program is a theory, consisting of examples, concepts, conjectures, and pos-

sibly proofs. In particular, there is usually a specific task in machine learning, as opposed to the rather more fluid goal in automated theory formation of developing interesting theories. Hayes-Roth applies the methods to situations in which there is a clear goal, for example to minimise the number of points scored in hearts. The third difference is that Hayes-Roth is proposing how the techniques *may be* automated and applied, rather than presenting a finished implemented system. It is this latter difference which our project most clearly contrasts, as we present an implemented system: HRL.

7.10 Summary

In this chapter we have described Lakatos's treatment of exceptions, in Lakatos [1976], and his methods of piecemeal exclusion and strategic withdrawal. By considering these methods in number theory, we have further distinguished between counterexample and piecemeal exclusion, and the need to apply a method twice to equivalence conjectures. We have described our implementation of these methods in HRL, and described some sessions which illustrate the types of concept and conjecture which can be developed using these methods. Finally, we have considered related work.

Chapter 8

A computational representation of Cauchy's proof

> ... "I propose to retain the time-honoured technical term 'proof' for a *thought-experiment – or 'quasi-experiment' – which suggests the decomposition of the original conjecture into subconjectures or lemmas*, thus *embedding it* in a quite possibly distant body of knowledge." Teacher, in [Lakatos, 1976, p. 9].

The method of lemma-incorporation is a precursor to the method of proofs and refutations, and, combined, these two methods are considered by Lakatos to be the most sophisticated methods. They are the only methods to consider the 'proof' of a conjecture; the other methods consider only the conjecture and the concepts in it. Given a counterexample to a conjecture, lemma-incorporation uses the proof to improve the conjecture, as well as improving the proof itself.

Euclid attempted, in *The Elements*, to present results in geometry in a formally precise way using the axiomatic method. Since then, mathematicians have traditionally ascribed great importance to proofs. Given this, it is surprising that only two of Lakatos's seven methods which detail the evolution of a conjecture, actually consider the proof of it. This is particularly surprising given the case study which Lakatos chose, since Euler proposed a proof at the same time as the conjecture itself (he published the conjecture in [Euler, 1758b] and the proof in [Euler, 1758a]). Cauchy [1813] also published a proof, and it is this proof which is discussed in [Lakatos, 1976]. Therefore, proofs of Euler's conjecture *were* known to those who found counterexamples and those who modified the conjecture. When counterexamples arose, it seemed, mathematicians performed exception-barring without considering how a proof of a faulty conjecture was possible, nor what to do with the old proof now that a problem had

arisen. The situation was even more strange in Lakatos's second case study, in which Fourier found a counterexample to Cauchy's principle of continuity, which Cauchy was aware of, *before* Cauchy proved it (although Fourier did not consider it to be a valid counterexample).

The 'proven' fallibility of proofs has led mathematicians to change their view of the principal role that proof plays, from a guarantee of truth, to an aid to understanding a theorem [Hardy, 1928], a way of evaluating a theorem by appealing to intuition [Wilder, 1944], and a memory aid [Polya, 1945] (described in [Lakatos, 1976, p 29]). However, even given the changing view of the role of proof (which may include a combination of the roles suggested) the idea that proof does indeed play an important role in mathematics is usually unquestioned. Possibly because of this, the methods which deal with proof are considered by both Lakatos and commentators (for instance, [Corfield, 1997]) to be the most important.

There were two stages to our implementation of lemma-incorporation. Firstly it was necessary to represent Cauchy's proof of Euler's conjecture, in order to be able to build and run a model of lemma-incorporation, which was our second task. In this chapter we describe the first stage. We have not intended to represent the proof in a cognitively plausible manner: this would have necessitated modelling an understanding of space, motion, stretchability etc. to be able to visualise the operations of removing parts of a polyhedron and transforming the remainder. The question of how humans are able to perceive and manipulate rigid and non-rigid spacial structures is outwith the scope of this thesis (although ideas in §13.5 may be a starting point for such a project). This chapter is simply a building block for the following chapter, in which we outline Lakatos's method of lemma-incorporation, and discuss our implementation of it.

We organise this chapter as follows: in §8.1 we outline Cauchy's proof of Euler's conjecture. In §8.2 we briefly describe how the proof is relevant to the method of lemma-incorporation, and we introduce Lakatos's distinction between global and local counterexamples. In §8.3 we consider three ways of representing a proof scheme in HRL: using Haggith's argument structures (§8.3.1), as a series of Java methods (§8.3.2), and as a series of production rules and associated conjectures (§8.3.3). Finally, in §8.4 we describe the representation we chose to use, which is a hybrid of those discussed. We summarise the chapter in §8.5.

8.1 Cauchy's proof

A proof idea for Euler's conjecture is presented by the teacher on the second page of [Lakatos, 1976]. This comes from Cauchy [1813], and consists of three steps. For a diagrammatic representation of these steps, carried out on the cube, see figure 8.1, taken from [Lakatos, 1976, p.8].

> *Step 1:* Let us imagine the polyhedron to be hollow, with a surface made of thin rubber. If we cut one of the faces, we can stretch the remaining surfaces flat on the blackboard, without tearing it. The faces and edges will be deformed, the edges may become curved, and V and E will not alter, so that if and only if $V-E+F=2$ for the original polyhedron, $V-E+F=1$ for this flat network - remember that we have removed one face. *Step 2:* Now we triangulate our map - it does indeed look like a geographical map. We draw (possibly curvilinear) diagonals in those (possibly curvilinear) polygons which are not already (possibly curvilinear) triangles. By drawing each diagonal we increase both E and F by one, so that the total $V-E+F$ will not be altered. *Step 3:* From the triangulated map we now remove the triangles one by one. To remove a triangle we either remove an edge - upon which one face and one edge disappear, or we remove two edges and a vertex - upon which one face, two edges and and one vertex disappear. Thus, if we had $V-E+F=1$ before a triangle is removed, it remains so after the triangle is removed. At the end of this procedure we get a single triangle. For this $V-E+F=1$ holds true. [Lakatos, 1976, p.7-8]

Lakatos argues that nineteenth century mathematicians viewed this proof as establishing the truth of the 'theorem' beyond doubt (in [Lakatos, 1976, p.8] he cites Crelle [1827], Matthiessen [1863] and Jonquières [1890] as examples).

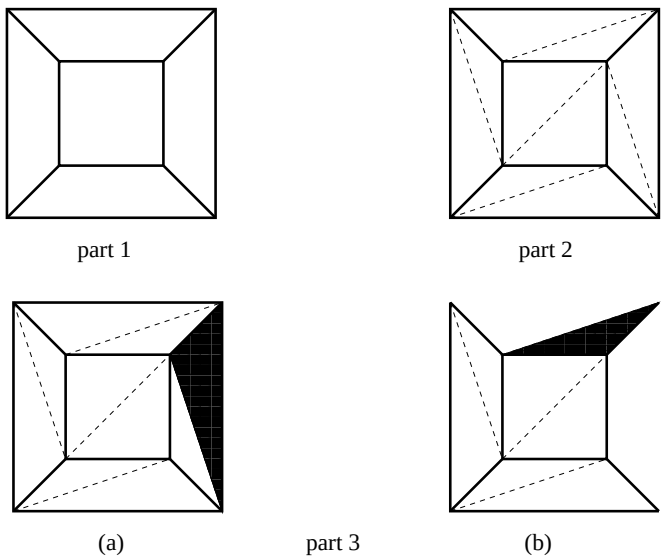


Figure 8.1: Given the cube, after removing a face and stretching it flat, we are left with the network in part 1. After triangulating, we get part 2. When removing a triangle, we either remove one edge and one face, or two edges, one vertex and a face – shown in parts 3(a) and (b) respectively.

8.2 Using counterexamples to improve a faulty conjecture and a faulty proof

Lemma-incorporation works by considering an object which is a counterexample either to a conjecture or to one of the proof steps, and using it to improve the conjecture, the faulty proof or both. Lakatos distinguishes between a *global* counterexample, which is a counterexample to the conjecture, and a *local* counterexample, which is a counterexample to one of the conjectures in the proof. For instance, the picture frame (in figure 8.2) is a global counterexample, since $V-E+F=16-32+16=0$. It is also a local counterexample as it violates the conjecture that if we remove a face from any polyhedron then we can stretch it flat on a board. Note that while the picture frame is both a local and global counterexample, it is possible for an entity to be a local but not global counterexample, if it is a counterexample to a conjecture in the proof, but not to the main conjecture, or to be a global but not local counterexample, if it is a counterexample to the main conjecture, but not to any of the conjectures in the proof. The first of these suggests that the conjecture may be true but the proof of it is flawed, and the second, less intuitive case, suggests that the proof is overly simplified and contains hidden assumptions.

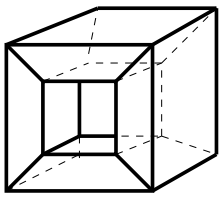


Figure 8.2: The picture frame, for which $V-E+F=16-32+16=0$

Note that Lakatos calls the conjectures in the proof 'lemmas': we adopt this terminology, despite the fact that the word 'lemma' has a recognised meaning in mathematics as a minor theorem, which is usually used in the proof of a more important result. We do this partly to be consistent with Lakatos, and partly to easily distinguish between the global conjecture and those in the proof. We can see the proof sketch as combining the following three lemmas, which we show for the cube in figure 8.1 (taken from [Lakatos, 1976, p.8]).

Conjecture: for any polyhedron, $V-E+F=2$
'Proof':
lemma 1: any polyhedron, after having a face removed, can be stretched flat on the blackboard.
lemma 2: in triangulating a map one will always get a new face for any new edge.
lemma 3: if we drop the triangles one by one from a triangulated map, there are only two alternatives – the disappearance of one edge or else of two edges and a vertex.

There is a fourth lemma: if we drop the triangles one by one from a triangulated map, we will end up with a single triangle. This lemma is inherent in Lakatos's early descriptions of Cauchy's proof, but omitted later on. We discuss this further in §9.2.3.

8.3 Representing Cauchy's proof in HRL

In order to modify a faulty proof, we first need to be able to represent it. Although HR is already interfaced with the theorem prover Otter [McCune, 1990], we cannot use Otter's output, as we need HR to work with informal, flawed proofs. Therefore we have extended HR to incorporate *proof schemes*. Our proof scheme class is a subclass of the pre-existing class 'theory constituent'; other subclasses of theory constituent are concept, conjecture and entity.

The method of lemma-incorporation starts *from* the proof, rather than looking at how it was generated (Lakatos claims that Polya [1962] suggested ways of doing this, and that [Lakatos, 1976] starts where [Polya, 1962] leaves off). That is, the interesting aspect of lemma-incorporation starts *given* a proof and counterexample to it.

We considered three ways in which to represent a proof scheme:
(i) using Haggith's argument structures [Haggith, 1996];
(ii) as a series of Java methods; and
(iii) as a series of production rules and associated conjectures.

8.3.1 Using Haggith's argument structures to represent Cauchy's proof

Haggith [1996] starts from the viewpoint that if a domain is controversial, then there may be more than one answer to a question and therefore disagreements may be useful, rather than an obstacle to be overcome. The primary goal of the system described in [Haggith, 1996] therefore, is to *explore* rather than resolve conflicts. In order to incorporate a high degree of flexibility, Haggith represented arguments at the meta-level which is independent of logic or any specific representation language or domain. Knowledge is represented as a set of propositions related by disagreement, equivalence, enlargement or justification. An argument A in which C is the conclusion, derived from P1 and P2, where P2 is itself derived from P3 is represented as $A = \{C <= \{P1, P2 <= \{P3\}\}\}$. Trees are constructed using the justification relation between nodes within the tree and the other three relationships between nodes in different trees. For example, given a proposition P and an argument A for P, possible argument moves which provide support for P include:

• *corroboration* - an argument for a proposition which is equivalent to (or is) P;

• *enlargement* - an argument for an elaboration of P, and

• and *consequence* - an argument in which P is a premise.

Argument moves which oppose A include:

• *rebuttal* - an argument for a proposition which disagrees with P;

• *undermining* - an argument for a proposition which disagrees with a proposition which is an elaboration of, or is equivalent to P;

• *undercutting* - an argument for a proposition which disagrees with a premise of P;

• *target* - an argument which contains a premise which disagrees with P, and

• *counter-consequence* - an argument which contains a premise which disagrees with the conclusion of another argument in which P is a premise.

We can express Cauchy's proof in Haggith's terms, if we write it as a series of propositions and the relationships between them. This is shown in figure 8.3.

The proof looks as follows: $A = \{C <= \{P0, P1 <= \{P2 <= \{P7\}, P3, P4 <= \{P8\}, P5, P6\}\}\}$. We can also represent the counter-arguments offered (we state them as propositions whereas in [Lakatos, 1976] they are questions, i.e., "are you sure that"... rather than "it is not possible"...). The counter-arguments are all examples of Haggith's target arguments, disagreeing with a premise of C.

Counter-argument 1:
$-P0$: Some polyhedra, after having a face removed, cannot be stretched flat on a board.

Counter-argument 2:
$-P8$: In triangulating the map, we will not always get a face for every new edge.

Counter-argument 3:
$-P7$: There are more than two alternatives, when we remove the triangles one by one, that either one edge and a face; or two edges, a face and a vertex disappear.

Counter-argument 4:
$-P3$: If we remove triangles one by one from a triangulated map, then we will not be left with a single triangle.

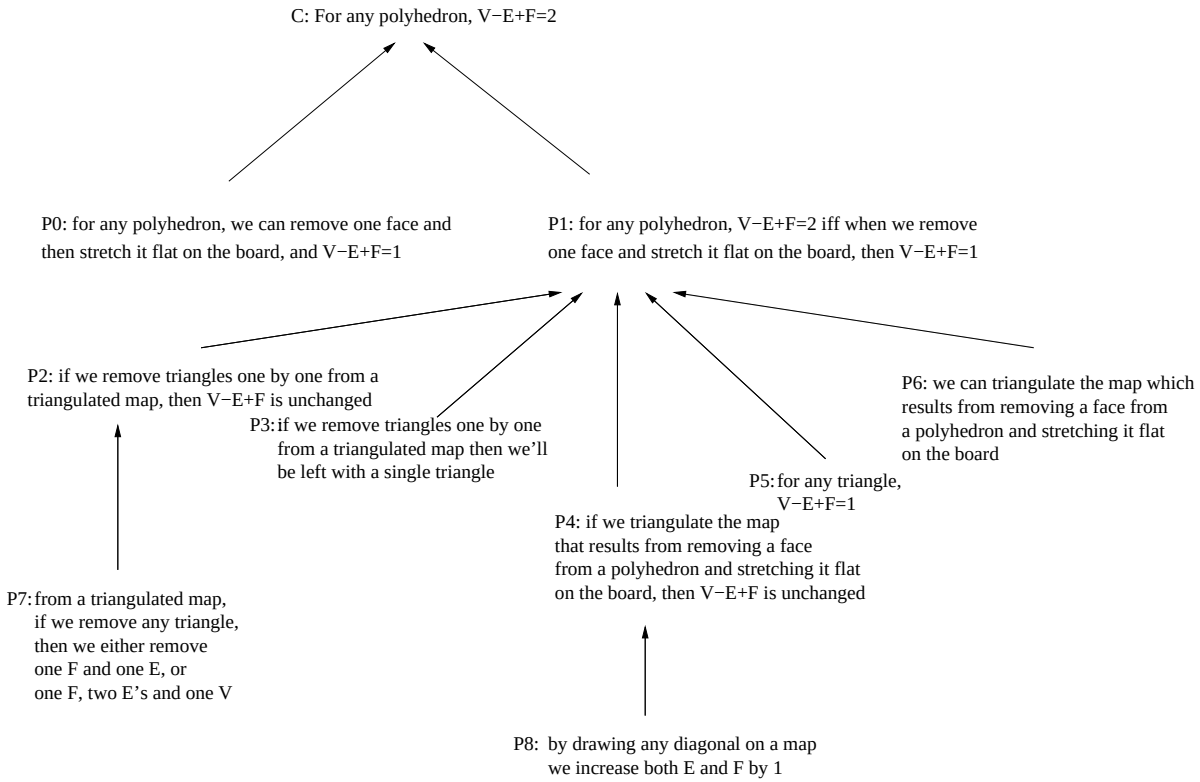


Figure 8.3: The original proof of Euler's conjecture, represented in Haggith's terms. The arrows represent the justification relation.

An advantage of using this representation is that it is a well thought out way of representing arguments and disagreement, based on argumentation research such as Elvang-Goransson et al. [1993], Toulmin [1958] and Sartor [1993], who inspired the rebuttal, under-cutting and counter-argument forms respectively. A minor consideration is that using this representation to represent mathematical 'proofs' and disagreements extends Haggith's work, as mathematics is a new domain which is not considered in [Haggith, 1996].

One disadvantage of using this representation is that the propositions are simply strings and therefore impenetrable. Strings such as these currently form no part of HR's theories, and would not be an obviously useful addition, and thus the representation sits rather poorly with the HR methodology.

8.3.2 Writing the proof as a series of Java methods

Lakatos's version of Cauchy's proof has a strongly procedural flavour, which, in the interests of staying faithful to Lakatos's book, would be desirable to incorporate into our representation. Work on procedural proofs has been done by, for example, [Winterstein, 2004], [Jamnik,

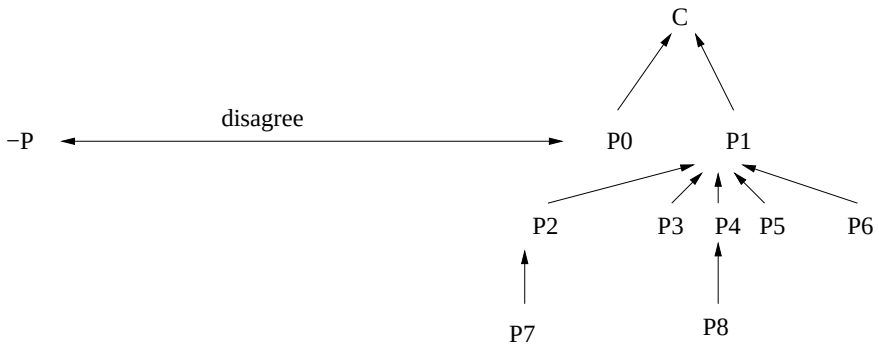


Figure 8.4: The first counter-argument, represented in Haggith's terms. Unmarked arrows represent the justification relation.

2001a]. One way of representing the proof procedurally is as a series of Java methods. For instance, we could write a Cauchy Proof Scheme class in the HR code, consisting of three main methods which constitute the three main steps in the proof. These might look like the following methods:

1) *removeOneFaceAndStretchFlatOnBoard* would take in a polyhedron, remove a face, stretch the resulting object flat, and return the network which results;

2) *triangulateNetwork* would take in a network, triangulate it and return the result;

3) *removeTriangle* would take in a triangulated network, remove a triangle, and return the result.

After each step, the Euler characteristic would be calculated, and claims about its value tested. Before performing any of the methods, the Euler characteristic of the polyhedron passed to the proof should be 2, and after each of the methods, the Euler characteristic of the resulting object should be 1. The third method would be repeated until a single triangle remains, which would be returned.

If a counterexample were passed to the proof, it could fail on two counts. Firstly, the method may fail. Java may throw an exception as the procedure cannot be performed, for instance in the case of the hollow cube being passed to the first method, where it is not possible to remove a face and stretch the resulting object flat. The other way in which a method could fail is if it loops infinitely, for instance in the third method if passed an infinite triangulated network (assuming that we can represent such an object), or if it is passed a triangulated network which has no triangles (such as a circle). In these two cases, a single triangle could never result. Secondly, the method may succeed, but the Euler characteristic of the resulting object have a different value to that which the proof claims it will have. For instance, in the third method,

given the triangulated network resulting from removing a face of a cube and stretching it flat, it is possible to remove an inner triangle. This would remove only a face and leave the number of vertices and edges unchanged, thus altering the Euler characteristic.

As the proof is procedural, and HR is written in Java, this would be a reasonable way of representing the proof. The main objection to this way is that we are adding code for a specific proof to the general HR code. This is very much against the HR methodology, and would provide no way of testing whether our lemma-incorporation algorithm works on other proofs, without having to write a separate proof class for each case we want to test.

8.3.3 Using HR's production rules and conjecture making mechanisms to represent Cauchy's proof

Lakatos (or the teacher) describes his interpretation of a proof as "a thought-experiment - or 'quasi-experiment' - which suggests a decomposition of the original conjecture into subconjectures" [Lakatos, 1976]. One way of representing the proof, therefore, is as a set of conjectures, and procedures for constructing the concepts in the conjectures. In §8.2 we suggested three conjectures, with a possible fourth one which together represent the proof. Here we break down the proof further, and below we state the procedures and conjectures and the step in Cauchy's proof to which they correspond.

The procedures in Cauchy's proof are:
- remove one of the faces (step 1)
- stretch the remaining surfaces flat (step 1)
- draw diagonals on any polygons which are not already triangles (step 2)
- recursively remove each triangle one at a time (step 3)

The conjectures are:

1. A polyhedron with a face removed can be stretched flat (step 1, P0)
2. Given a polyhedron, its Euler characteristic is 2 if and only if the Euler characteristic of its flat network is 1 (step 1, P1)
3. By drawing each diagonal we increase both E and F by one (step 2, P8)
4. The Euler characteristic of an untriangulated map is equal to that of the corresponding triangulated map (step 2, P4)
5. If we remove an edge from a triangulated network, then one face and one edge disappear (step 3, P7)

6. If we remove two edges and a vertex from a triangulated network, then one face, two edges and one vertex disappear (step 3, P7)
7. The Euler characteristic of a triangulated network remains constant when we remove a triangle (step 3, P2)
8. If we take a triangulated network and remove the triangles one by one, then we are left with a single triangle (step 3, P3)
9. The Euler characteristic of a triangle is 1 (step 3, P5)

This way of representing proof fits very well with HR's methodology of using production rules to change one concept into another, and forming conjectures which link a new concept to other concepts. Additionally, it clearly fits with Lakatos's way of viewing proofs.

A disadvantage of representing a proof as a series of conjectures is that it is not always clear how the conjectures fit together to form a proof. Given the above series of conjectures, we can represent the proof starting with the last conjecture, *9*, and working backwards; i.e., *9* $\rightarrow$ *8* $\rightarrow$ ... $\rightarrow$ *2* $\rightarrow$ *1*. However, the order of the conjectures and relationship between them is neither obvious nor unambiguous, and a straight series of implications may not be a rich enough representation to capture this, and other, proofs.

8.4 A hybrid representation of Cauchy's proof

We have attempted to combine the best of each of these approaches and avoid the disadvantages, mainly using ideas from sections 8.3.1 and 8.3.3.

We use Haggith's representation of arguments as trees, from [Haggith, 1996]. However, rather than each node being a string, it is a conjecture which HRL has formed. To avoid hard coding proofs in the HRL code, we force the concepts and conjectures in an input file. This is the usual way of inputting information to HR; the input file also contains information such as which production rules are to be used in the session. At the end of the file, we give the structure of the proof in the form of Haggith's argument trees. In order to get core concepts such as whether a network is planar or not, and the triangulated version of a network, we use ideas from §8.3.2, where the methods are written into the UserFunctions class in HRL. This is the usual place for calculating information about core concepts, containing code for calculating the divisors of an integer, etc..

We have written a new domain file for HRL, in which we include concepts from number theory, polyhedra and graph domains. From number theory, we have the concept of being an integer.

From the polyhedra domain, we have the concept of being a polyhedron, of removing a face from a polyhedron and stretching it out, and the Euler characteristic of a polyhedron. From graph theory, we have the concept of being a graph, the Euler characteristic of a graph, the concept of taking in a graph and triangulating it, of whether a given graph is planar and whether it is triangulated, the concept of removing a triangle from a graph, and the concepts of removing one edge and one face from a graph, and removing two edges, one vertex and one face from a graph. The entities are integers 1-8, the polyhedron cube – represented just as the string 'cube', and a series of graphs – represented by a string of vertices (denoted by letters) and edges (denoted by two vertices). The set of graphs include the graph in figure 8.1, and those created by recursively removing a triangle from this graph.

We have extended HR by adding a *proof scheme* class, which is a subclass of *theory-constituent*. Each proof scheme has a global conjecture, a vector of conjectures in the proof, and a proof tree which specifies how the conjectures fit together to prove the global conjecture. We continue to use the term 'lemma' to mean a conjecture which is in the proof: these objects are not different in any way from the usual conjecture object in HR. Below, we show an HRL proof scheme for Euler's conjecture, where the conjectures are shown as HR represents them in ascii format. This involves some repetition of concepts, which are independent of the conjecture that links them. HR is able to represent conjectures in a variety of other formats, but this one makes the concepts involved very clear. In the fifth lemma, a specific graph $egh[gh;he;eg]$ is shown: this is the concept in HR of a triangle graph, where the graph has vertices *e*, *g* and *h*, and edges *gh*, *he* and *eg*.

Global conjecture:
for all a: a is a polyhedron → a is a polyhedron & the euler characteristic of a is 2

Proof scheme:

1. for all *a*: *a* is a polyhedron → *a* is a polyhedron & exists *b* (*b* is a graph & remove a face from *a* and stretch it flat to get *b* & *b* is planar)

2. for all *a*: *a* is a polyhedron & the euler char of *a* is 2 ↔ *a* is a polyhedron & exists *b* (*b* is a graph & remove a face from *a* and stretch it flat to get *b* & *b* is planar & the euler char of graph *b* is 1)

3. for all *a b*: *b* is an integer & *a* is a graph & the euler char of the graph *a* is *b* & *a* is planar ↔ *b* is an integer & *a* is a graph & *a* is planar & exists *c* (*c* is a graph & *c* is the triangulated version of *a* & the euler char of the graph *c* is *b*)

4. for all *a*: *a* is a graph & *a* is triangulated → a is a graph & ((*b* is a graph & *b* is *a* with one triangle removed) → (*b* is a graph & ((*a* is a graph & *b* is *a* with one edge and one face removed) or (*a* is a graph & *b* is *a* with two edges, one vertex and one face removed))))

5. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a is triangulated & $\neg(a = egh[gh;he;eg]) \leftrightarrow b$ is an integer & a is a graph & exists c (c is a graph & c is a with one triangle removed & the euler char of the graph c is b)

6. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & $a = egh[gh;he;eg] \leftrightarrow b$ is an integer & a is a graph & the euler char of the graph a is b & $a = egh[gh;he;eg]$ & $b = 1$

We use Haggith's representation of proof structure.

8.5 Summary

We have laid the groundwork for our implementation of Lakatos's method of lemma-incorporation. Because this method uses a possibly flawed proof of a conjecture, we have had to enable HRL to represent a proof scheme which may be faulty. We have discussed different ways of doing this, and outlined the way we selected. Thus we are now able to describe our implementation of lemma-incorporation.

Chapter 9

The method of lemma-incorporation

> Not only do refutations act as fermenting agents for proof-analysis, but proof-analysis may act as a fermenting agent for refutations! What an unholy alliance between seeming enemies! *Sigma*, in [Lakatos, 1976, p. 48].

In the previous chapter, we outlined our representation of Cauchy's proof in HRL, and introduced Lakatos's method of lemma-incorporation. We now build on that work by discussing the method further, and describing our implementation of it in HRL. We organise the chapter in the following way. Firstly, we distinguish three types of counterexample: an entity which breaks a conjecture, one which breaks one of the proof steps, or one which breaks both. Each of these types of counterexample corresponds to a different type of lemma-incorporation, which we describe in §9.1. We discuss aspects of the method in §9.2 with a view to how we might implement it. In §9.3 we discuss how lemma-incorporation might be applied to examples other than Euler's conjecture and Cauchy's proof, giving examples from geometry, and group theory, as well as Lakatos's second case study in the field of real analysis. In order to implement lemma incorporation, we need *(i)* a way of representing the initial proof, and *(ii)* a way to modify it, given a counterexample. In §9.4 we show how HRL determines which kind of lemma-incorporation to perform, based on the type of counterexample being considered. In sections 9.5, 9.6, and 9.7, we describe our implementation of each type of lemma-incorporation, outlining our algorithm and presenting an illustrative session in each case. In §9.8 we briefly consider the method of proofs and refutations, which is a combination of the method of lemma-incorporation and counterexample generation. We discuss related work in §9.9, and conclude by summarising the chapter in §9.10.

9.1 Three types of counterexample

In the previous chapter, we described how lemma-incorporation works by considering an object which is a counterexample either to a conjecture or to one of the proof steps, and using it to improve either the conjecture, the faulty proof or both. We also introduced Lakatos's distinction between a *global* counterexample, which is a counterexample to the conjecture, and a *local* counterexample, which is a counterexample to one of the conjectures in the proof. In this section, we discuss the three different types of lemma-incorporation, which correspond to different types of counterexample.

We refer to Lakatos's description of Cauchy's proof sketch as the following three lemmas, taken from [Lakatos, 1976, p.8].

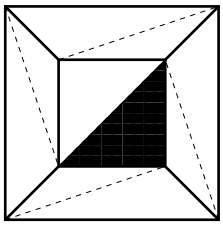
Conjecture: for any polyhedron, $V-E+F=2$
'Proof':
lemma 1: any polyhedron, after having a face removed, can be stretched flat on the blackboard.
lemma 2: in triangulating a map, one will always get a new face for any new edge.
lemma 3: if we drop the triangles one by one from a triangulated map, there are only two alternatives – the disappearance of one edge or else of two edges and a vertex.

As we observed in the previous chapter, a further fourth lemma, inherent in Lakatos's early descriptions of Cauchy's proof, but omitted later on is: if we drop the triangles one by one from a triangulated map, we will be left with a single triangle.

9.1.1 Local, but not global counterexamples

In the lemma-incorporation method, given a counterexample, the first step is to determine which type of counterexample it is. The first counterexample to arise in [Lakatos, 1976] is the cube. This is a supporting example of the conjecture, since $V-E+F=8-12+6=2$. When we have performed the steps in lemmas 1 and 2, we are left with the triangulated network in part 2, figure 8.1. However, when we remove the triangles one by one, there is a third possibility to the possibilities (a) and (b) which are shown in figure 8.1, on page 113. This is to remove the triangle shown in figure 9.1. This is a counterexample to lemma 3, as we have dropped a triangle, but no edges or vertices have disappeared. Therefore the cube is a local, but not global counterexample.

The cube is the least threatening type of counterexample, as it does not show that the conjecture is false, only that the reason for believing it is flawed. The reaction to this type of counterexample is to use it to modify the lemma in question, by generalising from the counterexample



part 3 (c)

Figure 9.1: Given the network which results from taking the cube, removing a face and stretching it flat, and triangulating, we can remove a triangle which results in removing one face, no edges and no vertices.

to a class of counterexamples, and excluding that class from the lemma. This is piecemeal exclusion, as we discuss further, in §9.2.1. In this example, we generalise from the triangle highlighted in figure 9.1, to *inner triangles*, and then form the concept of triangles which are not inner triangles, i.e., *boundary triangles*. We then limit the lemma to this type of triangle. The modified version of *lemma 3* is: 'there are only two alternatives – the disappearance of one edge or else of two edges and a vertex – when one drops the *boundary* triangles one by one'. We leave the global conjecture unchanged. We call this *local-only lemma-incorporation*.

9.1.2 Global and local counterexamples

The second type of counterexample the students discover is the hollow cube. This is both global, as $V - E + F = 16 - 24 + 12 = 4$, and local, as it violates the conjecture that if we remove a face from any polyhedron then we can stretch it flat on a board. In this case, we should find which step of a faulty proof a counterexample violates, i.e. which is the problem lemma, and then make that step a condition of the conjecture. The proof is left unchanged. Therefore, given the hollow cube, we should incorporate the first lemma into the global conjecture. The global conjecture becomes 'for any polyhedron which, by removing one face, can be stretched flat onto a blackboard, $V - E + F = 2$'. We call this *global and local lemma-incorporation*.

9.1.3 Global, but not local counterexamples

The last type of counterexample which the students discover is one which is global, but not local. This is an odd type of counterexample, as it suggests that the conjecture is flawed and yet the proof of it is upheld. This situation occurs because of 'hidden assumptions', or lemmas, in

the proof, which the counterexample *does* violate. The students only find one such counterexample, the cylinder. This is a global counterexample, as $V-E+F=0-2+3=1$. However, as the student *gamma* argues in a heated exchange about truth and meaning, [Lakatos, 1976, 43-45], it does not break any of the lemmas. We can remove a face and stretch it flat, as shown in figure 9.2 on page 127. In order to falsify the second lemma, we would have to draw an edge which joins two non-adjacent vertices, but does not create a new face. Clearly we cannot do this as there are no vertices on the map. Similarly, in order to falsify the third lemma, we would have to be able to remove a triangle and not remove either one edge and one face, or two edges, a vertex and a face, and since there are no triangles on the map, we cannot fail at this stage either.

Faced with such a counterexample, we have to retrace our progress through the proof, until we come to a step which is in some way surprising. One reason a step may be surprising is if it violates some hidden assumption in the mathematician's mind. Once this has been identified, we should make it explicit in the proof. The counterexample then becomes a global *and* local counterexample, which should be dealt with as outlined above.

In the case of the cylinder, there are two hidden assumptions. Firstly, we expect that having performed lemma 1, we are left with a connected network. Therefore we should add this into the proof explicitly, and modify lemma 1 to 'any polyhedron, after having a face removed, can be stretched flat on the blackboard, and the result is a connected network'. The cylinder clearly violates this, so we incorporate the new, explicit lemma, into the conjecture statement, which then becomes 'for any polyhedra which, after having a face removed, can be stretched flat on the blackboard *leaving a connected network*, $V-E+F=2$'. The second hidden assumption is that once we have a triangulated network, there is at least one triangle on the board. Again we can make this assumption explicit in the proof and incorporate it into the statement of the global conjecture.

Lakatos calls the principle of turning a global but not local counterexample into one which is both, the *principle of retransmission of falsity*. This requires that falsehood should be retransmitted from a global conjecture to the local lemmas. Thus, any entity which is a counterexample to the conjecture must also be a counterexample to one of the lemmas. Lakatos calls this *hidden lemma-incorporation*.

9.2 Discussion of the method of lemma-incorporation

In the previous section, we outlined Lakatos's three different types of lemma-incorporation which correspond to the three types of counterexample. In this section, we discuss these

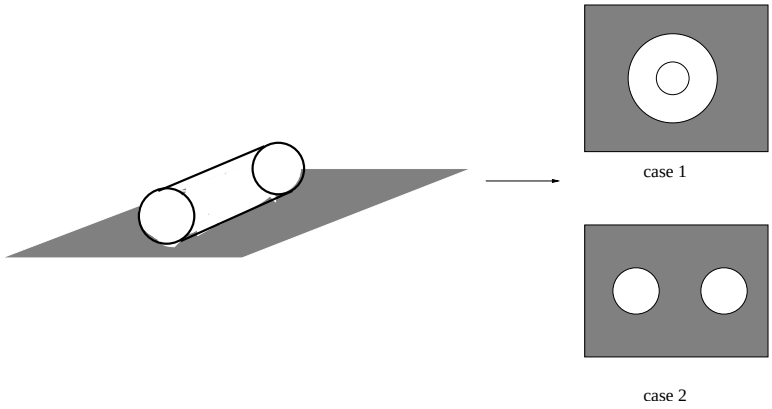


Figure 9.2: If we remove a face from the cylinder and stretch it flat, then we either get case 1, if we remove an end face, or case 2, if we remove the jacket. Either way, we have satisfied the first lemma.

lemma-incorporation methods, highlighting aspects which have influenced our implementation.

9.2.1 Combining the methods

This is the first time in [Lakatos, 1976] that a combination of methods is suggested: exception-barring to get a 'very fine delineation of the prohibited area' [Lakatos, 1976, p 37], and then lemma-incorporation.

> The best exception-barrers do a careful analysis of the prohibited area... in fact your method [the method of lemma-incorporation] is, in this respect, a limiting case of the exception-barring method... [Lakatos, 1976, p 37].

For implementation purposes, this means that within the method of lemma-incorporation, we can reuse our exception-barring methods.

9.2.2 Identifying a problem lemma in hidden lemma-incorporation

The problem of identifying lemmas which involve hidden assumptions which may be unjustified is not difficult for humans. This is because of the element of surprise which people feel when an entity does not behave in the expected way, where the 'expected way' has been learned from previous examples. Simulating this feeling of surprise, however, is a difficult task. To help

us, we consider here what exactly caused the students' surprise. Hidden assumptions are found in two lemmas and cause surprise in different ways.

Surprise caused by unexpected behaviour
Lemma one states that any polyhedron, after having a face removed, can be stretched flat onto a blackboard. Although the cylinder is a supporting example of this conjecture, it is surprising: when we remove the jacket from the cylinder, it falls into two parts, leaving two disconnected circles. This is surprising since all previous examples left connected networks. Therefore, we want to capture the idea of an entity being surprising with respect to a given conjecture.

Surprise caused by non-meaningful terms
The discussion of the second lemma, that *any face dissected by a diagonal falls into two pieces*, with respect to the two disconnected circles, is related to work on meaning and denotation, for instance, [Russell, 1971]. The problem is that although there are no diagonals on a circle, we are making a claim about the properties that they have. *Gamma* argues that it is correct to say that 'every new diagonal we draw on two disconnected circles results in a new face' (P), since the negation, that 'there is a diagonal of the two circles which does *not* create a new face' ($\neg P$), is false. This argument uses the law of excluded middle in classical logic, $P \vee \neg P$, i.e. $\neg(\neg P) \rightarrow P$. According to this argument, the cylinder is a global but not local counterexample. *Alpha*, who disagrees, argues that if we say that P is true then we must be able to construct at least one instance of it, i.e. there must be an existential clause in the lemma. The statement 'a face is simply connected' means 'for all x, if x is diagonal then x cuts the face into two; *and there is at least one x that is a diagonal*' [Lakatos, 1976, p. 45]. Under *Alpha*'s interpretation, the cylinder *is* a counterexample to this lemma, as there are no diagonals on the circle. Therefore the cylinder is a local as well as global counterexample, and the problem is no longer a case of hidden lemma-incorporation.

Although it would be difficult to model the surprise that a human feels when they attempt to triangulate a circle, the emphasis on vacuously true statements does give us an insight into how to automate this method.

9.2.3 Controversy over whether a global counterexample is local or not

Even if we disregard the different interpretations of the second lemma, and hence disagreement about whether the cylinder is a local as well as global counterexample, Gamma's argument that it is only global is not convincing. In the initial proof given (see §8.1) it *does* say explicitly that at the end of the process there is a single triangle. We represented this as a fourth lemma; *if we drop the triangles one by one from a triangulated map, we will end up with a single*

triangle, in §8.2, on page 115. This lemma is violated by the cylinder, making it both a global and local counterexample. This would allow for the usual modification of making the lemma a precondition, i.e. the conjecture would become:

'for any polyhedra which, after having a face removed, and then stretched flat, triangulated and the triangles removed one by one, *leaves a single remaining triangle*, $V-E+F=2$'.

Gamma is able to make his argument because the students get distracted by his claim that the cylinder can be triangulated, and the discussion turns to the meaning of statements which are vacuously true. If they had not disputed this point, Gamma would not have been able to uphold his argument. However, the cylinder is still an important example, in that it highlights hidden assumptions in the proof, which should be explicit.

9.2.4 The *type* of counterexamples in hidden lemma-incorporation

The *type* of entity that we are dealing with is important for computational purposes. This may change as we step through a proof. For instance, in Cauchy's proof, we begin by talking about *polyhedra* and, having removed a face and stretched it onto the board, we talk about *graphs*. Therefore it may appear that we have a counterexample which is global and not local, since a polyhedron can only be a counterexample to conjectures about polyhedra, not to conjectures about graphs. We have to look for the corresponding graph and determine whether this entity is a counterexample to those conjectures about graphs. In this case, the disconnected circles in lemma 2 correspond to the cylinder. This is glossed over in [Lakatos, 1976], as the following quote shows:

> *Gamma:* The cylinder *can* be pumped into a ball – so according to *your* interpretation it does comply with the first lemma.
>
> *Alpha:* Well... But you have to agree that it does *not* satisfy the *second* lemma, namely that '*any face dissected by a diagonal fall into two pieces*'. How will you triangulate the circle or the jacket? Are these faces simply connected? [Lakatos, 1976, p. 44].

When Alpha uses the word 'it', he refers to the cylinder. However, he then moves onto talking about the associated graph. While for humans this leap may be acceptable, when implementing this in a program we need to be explicit about which types of entity we are referring to. Suppose, for instance, that we hold that the graph which results from removing a face from the cylinder and stretching it flat, *is* a counterexample to a proof lemma. In this case, the cylinder is considered to be a local as well as a global counterexample, despite the fact that the cylinder itself is not a local counterexample. We use this observation in our algorithm for determining which kind of lemma-incorporation to perform.

9.2.5 The number of applications of lemma-incorporation

It is worth noting that students apply lemma-incorporation more than once. In the example in [Lakatos, 1976], it is applied at least three times, thus enabling the discussion on different types of lemma-incorporation. This is like the previous methods; one counterexample is found and dealt with and then more counterexamples to the modified conjecture and proof are sought.

9.3 Lemma-incorporation applied to other examples

It is useful to see how the method of lemma-incorporation is used in other examples, in order to help us to extract the important aspects, and to provide test cases for our implementation. In this section, we consider Lakatos's application of the method to Cauchy's Principle of Continuity, as well as our own descriptions of how it can be applied to two of Hilbert's theorems, and an imaginary conjecture in the theory of groupoids.

9.3.1 Cauchy's Principle of Continuity

In appendix 1 of [Lakatos, 1976], Lakatos shows how the method of hidden lemma-incorporation was used to fix Cauchy's faulty conjecture that 'the limit of any convergent series of continuous functions is itself continuous'. The counterexample, found by Fourier is:

cos x - 1/3 cos 3x + 1/5 cos 5x -...

which converges to the step function. The most interesting aspect of this example is the timing of various discoveries. Fourier discovered the above series in 1808 (see [Fourier, 1808]), and it was *after* this that Cauchy discovered the conjecture and proof, in 1821 (see [Cauchy, 1821]). One solution to this awkward situation was that the limit function was actually continuous, and therefore it was not a counterexample (Fourier held that it was continuous). However, Cauchy had provided a new interpretation of continuity, according to which the limit was *not* continuous (the existence of Fourier's example was considered by some to be evidence that the new interpretation should be rejected). Another possible solution was the argument that the series did not converge, although this view was not accepted by most mathematicians, including Cauchy, who later proved that it did converge. There was then a long gap until 1847 when Seidel found the hidden assumption of uniform convergence in the proof. Indeed, it was Seidel who invented the method of proofs and refutations. Lakatos thought that the main reason for such a long gap, and the willingness of mathematicians to ignore the contradiction, was a commitment on the part of mathematicians to Euclidean methodology. Deductive argument was considered infallible and therefore there was no place for proof analysis.

9.3.2 Hilbert's theorem

This example was highlighted by Meikle and Fleuriot's formalisation of Hilbert's *Grundlagen der Geometrie*, described in [Meikle and Fleuriot, 2003]. Hilbert's programme aimed to give a more precise and rigorous system of axioms for geometry than Euclid had given, by abstracting from any intuitive meaning of the terms. Thus, theorems should hold under any interpretation in which the axioms are satisfied. Bernays describes Hilbert's view thus: "it is insisted that in reasoning we should rely only on those properties of a figure that either are explicitly assumed or follow logically from the assumptions or axioms" [Bernays, 1967, p. 497], cited in [Shapiro, 2000, p. 152]. However, Meikle et al claim that by formalizing Hilbert's work in the theorem provers Isabelle/Isar, they have shown that "Hilbert's work glossed over subtle points of reasoning and relied heavily, in some cases, on diagrams which allowed implicit assumptions to be made" [Meikle and Fleuriot, 2003, p. 16]. Thus, Meikle et al's work not only suggests examples of where the method of lemma-incorporation may apply, but also demonstrates the widespread application of the method within mathematics. It does this by studying the work of a mathematician who explicitly set out to remove all hidden assumptions, and showing how even he failed to do so completely.

The example below can be seen as an instance of hidden lemma-incorporation (global and not local). Since Lakatos described his method in terms of a procedural proof, we paraphrase Hilbert's proof as a procedural proof, from the deductive proof which Hilbert gave. However, it is worth noting that the method of lemma-incorporation also applies to declarative proofs, which we demonstrate with respect to the first step of Hilbert's proof, and our group theory example below. The axioms which are referred to are in appendix B, as is Hilbert's declarative proof.

Theorem: For two points A and C there always exists at least one point D on the line AC that lies between A and C.

Proof: (paraphrased from [Hilbert, 1901, p. 6] as a procedural proof)

lemma 1: draw a line AC between the two points
lemma 2: mark a point E outside the line AC (axiom (I,3))
lemma 3: mark another point F such that F lies on AE and E is a point of the segment AF (axiom (II,2))
lemma 4: mark on FC a point G, that does not lie on the segment FC (axiom (II,2) and axiom (II,3))
lemma 5: the line EG must then intersect the segment AC at a point D (axiom (II,4)) □

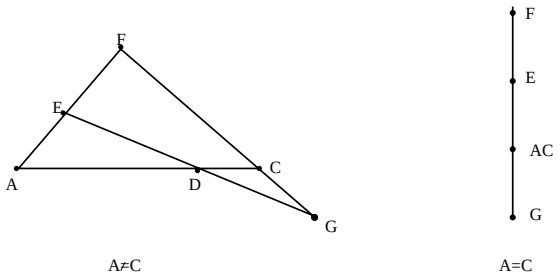


Figure 9.3: Diagram of Hilbert's proof of the theorem that for two points A and C, there always exists at least one point D on the line AC that lies between A and C. In the first figure, A and C are different points, and in the second, they are the same.

The counterexample comprises any two points which are identical, i.e., (a,a). With the proof phrased as above, (a,a) is a global, but not local counterexample. This example is interesting, since, in Hilbert's original German edition of the Grundlagen in 1899, reprinted in [Hallett and Majer, 2004], he does not exclude this example: there is nothing in the axioms to say that a line must join two *different* points,[1] nor that a line which intersects a segment AC must be strictly between the points A and C, etc. However, in later editions, Hilbert has amended this: for instance, at the beginning of [Hilbert, 1901], which is a later, English translation, Hilbert states that in all of the theorems, he assumes that where he says two points, they will be considered to be two distinct points (the relevant omission in his first work is [Hallett and Majer, 2004, pp. 437-438, Chap 5]). A diagram which illustrates the proof steps for both cases, where $A \neq C$ and $A = C$, is shown in figure 9.3.

The hidden assumption that the two points are different becomes obvious to humans when they try to draw the line which joins A and C. The resulting line is theoretically acceptable, but not what we were expecting. The first step of the proof then, contains the hidden assumption. Following the method of lemma-incorporation, this should be made explicit and the first step modified to:

lemma 1′: Draw a line AC between the two points, *which has positive length.*

Modified theorem: For two points A and C, such that when we draw a line AC between them, AC has positive length, there always exists at least one point D on AC that lies between A and C.

This can be simplified to: 'For two *distinct* points A and C there always exists at least one point D on the line AC that lies between A and C'.

[1] The relevant axiom is (I,1), which states that for every two points, A,B there exists a line that contains each of the points. The axiom does not specify that the points must be different.

Note that, although we have paraphrased Hilbert's proof as a procedural proof (to fit Lakatos's terminology), it was a declarative proof, and the method of lemma-incorporation applies equally well in this form. The first step of Hilbert's proof is:

By Axiom (I,3) there exists a point E outside the line AC, and by Axiom (II,2) there exists on AE a point F such that E is a point of the segment AF.

This would be modified via lemma-incorporation to:

By Axiom (I,3) there exists a point E outside the line AC, where the line AC has positive length, and by Axiom (II,2) there exists on AE a point F such that E is a point of the segment AF.

This example highlights the following points.

1) We were expecting a line of positive length, and being told to draw a line between two identical points was unexpected because it was only vacuously true that such a line existed. This again illustrates the role that surprise plays;

2) The method is not limited to procedural proofs, but also applies to declarative proofs in exactly the same way. This suggests that the method could be a powerful technique: not one which works only on a contrived example, but one of general use;

3) Exception-barring can be used to delineate the problem area. When the problem lemma is identified, it is simple to find a concept which covers the positive examples in the lemma and excludes the counterexample, i.e., to find the concept of positive length, and then limit the domain of the theorem to those points which satisfy this concept, thereby performing strategic withdrawal;

4) The *type* of counterexample changes: in the global theorem the counterexample is a *point*, whereas in lemma 1 the counterexample is a *line*;

5) The teacher and students in [Lakatos, 1976] consider the possibility of an entity being a local only, global only, or a both local and global counterexample. They do not consider that a problem entity might be neither global nor local, assuming that such entities are positive examples of the theorem and proof, and therefore support rather than attack it. However, the admission that lemmas in the proof may contain hidden assumptions which mean that an entity satisfies the lemma, albeit it in a surprising way, raises the question of whether there could be an entity which satisfies the global conjecture in the same way, thus uncovering a hidden assumption in the global conjecture itself.

9.3.3 An example in group theory

This example is an invented example of hidden lemma-incorporation. It is the left hand cancellation law, which is a theorem in group theory. Here we hypothesise that it holds for all groupoids, where a groupoid is a closed set with a binary operation. Note that all groups are groupoids, but not vice versa (see glossary of group theory terms on page 253). In the following, g^{-1} denotes the unique inverse of element g, and e denotes the identity element of G.

Global conjecture: let G be a groupoid and g, h, k be in G. Then $gh = gk \rightarrow h = k$

Proof:

lemma 1: $gh = gk \rightarrow g^{-1}(gh) = g^{-1}(gk)$

lemma 2: $g^{-1}(gh) = g^{-1}(gk) \rightarrow eh = ek$

lemma 3: $eh = ek \rightarrow h = k$

A counterexample is the groupoid $G = <\{0,1\},*>$, where the * operator is defined as follows: $0*0 = 0$; $1*0 = 0$; $0*1 = 1$; and $1*1 = 0$. We represent G in table 9.1. To see that G is a global counterexample, let $g = 1$, $h = 0$, and $k = 1$. Then $gh = 1*0 = gk = 1*1$, but $0 \neq 1$.

	0	**1**
0	0	0
1	1	0

Table 9.1: A tabular representation of the groupoid $G = <\{0,1\},*>$

This conjecture, proof and counterexample is a good analogy to Euler's conjecture, Cauchy's proof and the cylinder counterexamples discussed in [Lakatos, 1976]. The counterexample is global, and is actually local as well, since it is a counterexample to the final lemma. However, when going through the proof, we never get to the final step, because of the surprisingness of the intermediate lemmas. The reason they are surprising is because they are vacuously true (or meaningless, depending on your notion of truth). This follows exactly the same pattern as in [Lakatos, 1976].

The counterexample $G = <\{0,1\},*>$, vacuously satisfies both lemmas 1 and 2. This sounds very odd at first sight, as G does not contain an identity element, and therefore elements do not have a unique inverse. It follows the same sort of reasoning as *gamma* performed in [Lakatos,

1976, p. 44], in which he argues that because the statement 'there is a diagonal of the circle that does not create a new face' is false, its negation, 'all diagonals of the circle create a new face', must be true, despite the fact that there are no diagonals on a circle. In our case, because the statement 'there exists $g,h,k \in G$ such that $g^{-1}(gh) \neq g^{-1}(gk)$' is false, then its negation, that 'for all $g^{-1},g,h,\in G, g^{-1}(gh) = g^{-1}(gk)$' must be true.[2] Similarly, in lemma 2, because the statement 'there exists $h,k,e \in G$ such that $eh \neq ek$' is false, then its negation, 'for all $h,k,e \in G, eh = ek$', must be true. Therefore, for lemma 1 the consequent will always be true (hence the lemma is satisfied), and for lemma 2 both antecedent and consequent will always be true (hence again, the lemma will be satisfied). The final lemma however could be false, as the consequent could be false, whereas the antecedent must always be true. However, by studying the vacuously true lemmas, it is possible to uncover the hidden assumption being made, and thus improve the proof by making it explicit. The hidden assumption in this example is that G is associative, in which case we would have $g^{-1}(gh) = g^{-1}(gk) \rightarrow (g^{-1}g)h = (g^{-1}g)k$. To modify the proof then, we make this explicit by adding it in as an extra lemma, which our counterexample breaks. The modified proof is then:

lemma 1: $gh = gk \rightarrow g^{-1}(gh) = g^{-1}(gk)$

lemma 2: $g^{-1}(gh) = g^{-1}(gk) \rightarrow (g^{-1}g)h = (g^{-1}g)k$

lemma 3: $(g^{-1}g)h = (g^{-1}g)k \rightarrow eh = ek$

lemma 4: $eh = ek \rightarrow h = k$

G is now considered a local and global counterexample, so we incorporate the new lemma into the conjecture in the usual way:

Modified global conjecture: let G be a groupoid such that $g^{-1}(gh) = g^{-1}(gk) \rightarrow (g^{-1}g)h = (g^{-1}g)k$ holds. Let g, h, k be in G. Then $gh = gk \rightarrow h = k$

If we strengthen the extra condition into associativity, then we could simplify the modified global conjecture to: 'let G be an associative groupoid, and let g, h, k be in G. Then $gh = gk \rightarrow h = k$'.

This example highlights the following points.

1) Vacuously true lemmas can play a role in alerting us to the fact that a hidden assumption is being made. In particular, this is of interest *even when the counterexample is actually a local*

[2] Our analogy differs in that the diagonals of a circle are not assumed to be unique, whereas the inverse element of g *is* unique. Russell's work on definite description, and his example of the 'present king of France' [Russell, 1971], as a denoting phrase which denotes a non-existent object, is relevant to our example.

counterexample as well. If an earlier lemma is vacuously true, then the assumption should be made explicit.

2) There may be more than one hidden assumption. Here we assumed that G had an identity, inverse and was associative.

3) It may be preferable to add in a clause which is slightly stronger than that needed, in order to get a simpler conjecture statement.

9.4 Determining which type of lemma-incorporation to perform

When a student receives a proof scheme from the teacher, it reconstructs the global conjecture and all of the local lemmas in the proof. When it does this, it looks to see whether it has any counterexamples, and if so, reconstructs an implication conjecture which the teacher sent as a near-implication, or an equivalence as a near-equivalence. If the student does have a counterexample, it needs to determine which type of lemma-incorporation to perform, by determining which type of counterexample it has. If the counterexample is global, then the student first tests to see if the same counterexample violates one of the local lemmas. If it does, then the student performs *global and local lemma-incorporation* by incorporating this lemma into the global conjecture. If not, then in order to avoid the problem of confusing the *type* of counterexamples and mistakenly performing hidden lemma-incorporation (expounded in §9.2.4), the student performs another test. Firstly, it determines whether any of the lemmas have any counterexamples at all. If so, then taking the first lemma with a counterexample (known as a local counterexample), HRL looks to see if there is a concept in the theory which links the global counterexample and the local counterexample. If there is, then it finds the first lemma in the proof scheme in which this concept appears, and performs *global and local lemma-incorporation* by incorporation this lemma into the conjecture statement. We say that a concept links two entities if it is a function which takes one of them as input and outputs the other entity. If there is a global counterexample, but neither of these two tests are satisfied, then the student performs *hidden lemma-incorporation*. If there is no global counterexample, but there is a local counterexample, then the student will perform *local-only lemma-incorporation*. If there is no counterexample, then the student returns the conjecture and proof unchanged. We present the method to determine which kind of lemma-incorporation to perform as algorithm 8, below.

Algorithm 8 Algorithm to determine which kind of lemma-incorporation to perform

Require: Given a proof scheme consisting of a global conjecture and local lemmas, then:

if there are any counterexamples to the global conjecture **then**
  if these counterexamples are also counterexamples to any of the lemmas in the proof **then**
    perform *global and local lemma-incorporation*
  else
    if there is a counterexample to a local lemma **then**
      if there is a concept which links the local counterexample to the global counterexample **then**
        if this concept appears in one of the lemmas in the proof scheme **then**
          perform *global and local lemma-incorporation*
        end if
      end if
    end if
    perform *hidden lemma-incorporation*
  end if
else if there are counterexamples to any of the lemmas in the proof **then**
  perform *local-only lemma-incorporation*
end if
return the proof scheme

9.5 Given a counterexample which is local but not global

Our method for local but not global lemma-incorporation, shown as algorithm 9, uses our exception-barring techniques. This uses the idea discussed in §9.2.1, in which a combination of methods is suggested. Stages 1-6 below are an attempt to perform piecemeal exclusion on the faulty lemma; stages 9 - 10 are an attempt to perform counterexample-barring; and stages 13 - 14 are an attempt to perform strategic withdrawal. These methods are explained in more detail in sections 7.6, 7.7 and 7.8 respectively.

Algorithm 9 Algorithm for local not global lemma-incorporation

Require: Given a conjecture to which there are no known counterexamples, and a proof tree which contains a faulty lemma, $P \rightsquigarrow Q$, to which there is at least one counterexample, then:

1: **loop**
2: **if** there is a concept C in the theory which exactly covers the counterexamples **then**
3: make the concept $P \wedge \neg C$, add it to the agenda, and perform one theory formation step.
4: **if** this theory formation step results in the new conjecture $P \wedge \neg C \rightarrow Q$ **then**
5: replace the faulty lemma with $P \wedge \neg C \rightarrow Q$, and return the improved proof scheme.
6: **else**
7: force the conjecture $P \wedge \neg C \rightarrow Q$ explicitly (this situation may occur if HR is not set to make implications in its input file). Replace the faulty lemma with $P \wedge \neg C \rightarrow Q$, and return the improved proof scheme.
8: **end if**
9: **end if**
10: **if** we have up to three counterexamples; x, y and z **then**
11: find the object of interest concept with the same type as x, y and z, and apply the production rule *entity_disjunct* to this concept, with parameters [x,y,z], to get the concept C of being x or y or z.
12: **else if** there is a concept C in the theory which exactly covers the positives in P, and is different from Q **then**
13: make the new conjecture $C \leftrightarrow Q$. Replace the faulty lemma with $C \leftrightarrow Q$, and return the improved proof scheme.
14: **else**
15: return the old proof scheme.
16: **end if**
17: **end loop**

9.5.1 Illustrative session

Initial information

We ran the agency with one student and the teacher. Both teacher and student were given the concepts described in §8.4. Additionally, given the graph in figure 9.4, the student's data table for the concept of removing a triangle from a graph has two possibilities: to remove the triangle efg (shaded dark grey), or to remove the triangle abf (shaded light grey). Since graphs are represented in terms of vertices and edges only, removing the face highlighted returns the same graph. For instance, suppose that we are given the graph
`abcdefgh[ab;bc;cd;da;ae;bf;cg;dh;ef;fg;gh;he;af;bg;ch;de;eg]`, where `abcdefgh` is a list of all of the vertices and `[ab;bc;etc. ]` is a list of all the edges which are represented by the vertices which they join. Then, the concept 'remove triangle from graph' returns both the same graph, and the graph
`abcdefgh[bc;cd;da;ae;bf;cg;dh;ef;fg;gh;he;af;bg;ch;de;eg]`, which has the edge *ab* removed.

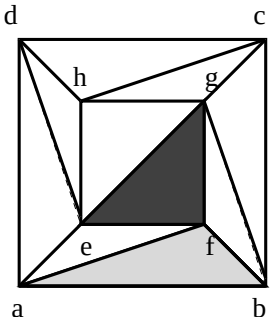


Figure 9.4: Given this graph, the concept 'remove triangle from graph' returns graphs resulting from removing triangle efg, or from removing triangle abf.

Session details

The student was given the initial concepts in the proof, in its input file. The teacher constructed the proof and sent it to the student, with the instruction to run for 100 steps and then perform lemma-incorporation on the proof. The student reconstructed the proof scheme as follows:

Global conjecture:

for all a: a is a polyhedron → a is a polyhedron & the euler characteristic of a is 2

Proof scheme:

1. for all a: a is a polyhedron → a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar)

2. for all a: a is a polyhedron & the euler char of a is 2 ↔ a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar & the euler char of graph b is 1)

3. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a is planar ↔ b is an integer & a is a graph & a is planar & exists c (c is a graph & c is the triangulated version of a & the euler char of the graph c is b)

4. for all a: a is a graph & a is triangulated ⇝

   a is a graph & ((b is a graph & b is a with one triangle removed) → (b is a graph & ((a is a graph & b is a with one edge and one face removed) or (a is a graph & b is a with two edges, one vertex and one face removed))))

5. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a is triangulated & -(a=egh[gh;he;eg]) ↔ b is an integer & a is a graph & exists c (c is a graph & c is a with one triangle removed & the euler char of the graph c is b)

6. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a=egh[gh;he;eg] ↔ b is an integer & a is a graph & the euler char of the graph a is b & a=egh[gh;he;eg] & b=1

The student found no global counterexamples, but identified faulty lemma (4), and the graph counterexample `abcdefgh[ab;bc;cd;da;ae;bf;cg;dh;ef;fg;gh;he;af;bg;ch;de;eg]`. The student then looked in its theory for a concept which covered only this counterexample, and found the concept of the graph *a* which is unchanged by having a triangle removed, i.e., the concept was ‘for all *a*: *a* is *a* with one triangle removed’. It then formed the new concept of being a triangulated graph and *not* being unchanged by having a triangle removed, and formed the new conjecture ‘for all *a*: *a* is a graph & ¬(*a* is *a* with one triangle removed) → *a* is a graph & *a* is triangulated’. Finally, the student replaced the old lemma (4) in the proof scheme with this new lemma, and sent the new proof scheme to the teacher. This modification is analogous to that performed in [Lakatos, 1976, pp 10-11], in which Cauchy’s proof of Euler’s conjecture is modified to state that the triangles which are removed from the network must be *boundary* triangles.

9.5.2 Discussion

Note that while in [Lakatos, 1976], the counterexample is considered to be the cube, in our implementation, the counterexample is considered to be the graph: `abcdefgh[ab;bc;cd;da;ae;bf;cg;dh;ef;fg;gh;he;af;bg;ch;de;eg]`. The student does not connect this with the cube. It would be possible to trace it up through the proof, but as it does not affect the role of the counterexample, we have not done this.

9.6 Given a counterexample which is both global and local

Algorithm 10 Algorithm for global and local lemma-incorporation

Require: Given a proof scheme where there are counterexamples to the global conjecture, and these counterexamples are also counterexamples to a lemma in the proof. Let the global conjecture be a near-implication $P \rightsquigarrow Q$, and the faulty lemma be a near-implication $R \rightsquigarrow S$, then:

1: form the concept C 'objects which satisfy the faulty lemma', by merging the two concepts R and S. This is done by applying the production rule 'compose' to R and S.
2: modify the global conjecture by making the new conjecture $C \rightarrow Q$.
3: replace the old global conjecture in the proof scheme with the modified version.
4: return the proof scheme.

9.6.1 Illustrative session

A student is given the hollow cube as an example of a polyhedron. When asked to perform lemma-incorporation, it finds that the hollow cube is a counterexample, both to the global conjecture, and the lemma that 'if we remove a face from any polyhedra, then we can stretch it flat onto a blackboard'. It makes the concept 'polyhedra which, if you remove a face, can be stretched flat on a board', and then limits the domain of the global conjecture to polyhedra of this type.

Initial information

A student was given the initial concepts in the proof, in its input file. The student was given the hollow cube as an example of a polyhedron.

Session details

The teacher constructed the proof and sent it to the student, with the instruction to run for 100 steps and then perform lemma-incorporation on the proof. The student reconstructed the proof scheme as follows:

Global conjecture:

for all a: a is a polyhedron $\rightsquigarrow$ a is a polyhedron & the euler char of a is 2

Proof scheme:

1. for all a: a is a polyhedron $\rightsquigarrow$ a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar)

2. for all a: a is a polyhedron & the euler char of a is 2 $\leftrightarrow$ a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar & the euler char of graph b is 1)

3. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a is planar $\leftrightarrow$ b is an integer & a is a graph & a is planar & exists c (c is a graph & c is the triangulated version of a & the euler char of the graph c is b)

4. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a is triangulated & -(a=egh[gh;he;eg]) $\leftrightarrow$ b is an integer & a is a graph & exists c (c is a graph & c is a with one triangle removed & the euler char of the graph c is b)

5. for all a b: b is an integer & a is a graph & the euler char of the graph a is b & a=`egh[gh;he;eg]` $\leftrightarrow$ b is an integer & a is a graph & the euler char of the graph a is b & a=`egh[gh;he;eg]` & b=1

The student found that the hollow cube is a counterexample to the global conjecture, and then looked to see if it is a counterexample to any of the lemmas as well, and found that it violates lemma 1. The student then applied the production rule *compose* to the two concepts in the faulty lemma: *(i) a is a polyhedron,* and *(ii) a is a polyhedron* & *exists b (b is a graph* & *remove a face from a and stretch it flat to get b* & *b is planar),* to produce: *a is a polyhedron* & *exists b (b is a graph* & *remove a face from a and stretch it flat to get b* & *b is planar).* That is, the student made the concept of objects which satisfy the faulty lemma. The student then modified the global conjecture to holding for objects of this type only, i.e., the global conjecture became 'for all a: a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar) $\rightarrow$ a is a polyhedron & the euler char of a is 2'. This has no counterexamples in the database. The student updated the proof scheme to contain the new global conjecture, and returned it. it currently

9.6.2 Discussion

Lakatos [1976] prescribes how to modify a global conjecture, given a faulty lemma. However, he does not make it clear what to do with the lemma. The relevant rule is:

> Rule 2: If you have a global counterexample discard your conjecture, add to your proof-analysis a suitable lemma that will be refuted by the counterexample, and replace the discarded conjecture by an improved one that incorporates that lemma as a condition. [Lakatos, 1976, p 50]

The problem is that these 'lemmas' in the proof are seen at different times as procedural steps to be carried out, and as declarative conjectures. If they are steps to be carried out, then the problem disappears: we have the conjecture, 'for any polyhedra which, by removing one face can be stretched flat onto a blackboard, $V-E+F=2$', and the step 'remove a face from the polyhedron and stretch it flat onto the blackboard', which can now, by definition, be carried out. However, if we regard them as conjectures, then we have three options. Firstly, we can keep the faulty local conjecture that 'all polyhedra can be stretched flat onto a blackboard, if we remove one face'. Secondly, we can replace it by the tautology that 'all polyhedra which, if we remove one face can be stretched flat onto a blackboard, can be stretched flat onto a blackboard if we remove one face'. The third option is to look for a concept which covers the objects that satisfy the faulty lemma.

In [Lakatos, 1976] the students discover the concept of 'polyhedra which you can inflate into a sphere' as one which is equivalent to the concept of 'polyhedra which, after removing one face, can be stretched flat onto a board'. This concept can then be used to modify the global conjecture to 'all polyhedra which you can inflate into a sphere have an Euler characteristic of 2', and the lemma in the proof step to 'for all polyhedra which you can inflate into a sphere, it is possible to remove a face and stretch it flat on the board'. The first two options above are clearly undesirable, while the third option – at least the lemma modification aspect of it – seems to be the most sensible. However, Lakatos does not discuss how we should modify a faulty lemma, if we have a counterexample which is both global and local. Even if a concept equivalent to the concept in the lemma is found, it is not used in the modification of the local lemmas (see [Lakatos, 1976, p. 34]).

9.6.3 An example in groupoid theory

In order to test our method, we invented an example of global and local lemma-incorporation in an algebraic domain (see the glossary of group theory terms on page 253).

Global conjecture: Let G be an algebra with one binary operator, and g, h, k be in G. Then $gh = gk \rightarrow h = k$.

Proof:
lemma 1: $gh = gk \rightarrow g^{-1}(gh) = g^{-1}(gk)$

lemma 2: $g^{-1}(gh) = g^{-1}(gk) \rightarrow (g^{-1}g)h = (g^{-1}g)k$

lemma 3: $(g^{-1}g)h = (g^{-1}g)k \rightarrow eh = ek$

lemma 4: $eh = ek \rightarrow h = k$

This conjecture is the left hand cancellation law, which is a theorem in group theory (note that a group must be associative and have a unique identity in the group, and a unique inverse for each element, in addition to being a groupoid). The proof is standard in group theory. We hypothesise that both the conjecture and proof hold for groupoids as well.

9.6.4 Illustrative session

We ran HRL with one student which had three groupoids:

$G1$ = the trivial algebra $< \{0\}, * >$,

$G2 = < \{0,1\}, * >$, where the * operator is defined as follows: $0*0 = 0$; $1*0 = 0$; $0*1 = 1$; and $1*1 = 0$, and

$G3 = < \{0,1,2\}, * >$, where the * operator is defined as follows: $0*0 = 0$; $0*1 = 1$; $0*2 = 2$; $1*0 = 1$; $1*1 = 0$; $1*2 = 0$; $2*0 = 2$; $2*1 = 0$; and $2*2 = 0$.

The first two algebras are groups, and the third, $G3$, is only a groupoid as it is not associative and does not have a unique inverse for each element. We represent its multiplication table in table 9.2 below.

	0	**1**	**2**
0	0	1	2
1	1	0	0
2	2	0	0

Table 9.2: The groupoid $G3$

The teacher started with the following proof scheme: global conjecture $\Leftarrow$ {lemma 1, lemma 2, lemma 3, lemma 4}, and asked the student to perform lemma-incorporation. The student found $G3$ as a counterexample to the global conjecture, and found that this was also a counterexample to lemma 2. It then formed the concept:

algebras with tuples g, h and k such that $(g^{-1}g)h = (g^{-1}g)k$,

and from this the student modified the global conjecture to:

for all algebras with elements g, h and k, $(g^{-1}g)h = (g^{-1}g)k \rightarrow (gh = gk \rightarrow h = k)$.

9.6.5 Discussion

It is encouraging that the method ran in a different domain to that used by Lakatos, and used the proof and counterexample to modify a false conjecture to a true theorem. However, the theorem is trivial as it only applies to $G1$, i.e. the trivial group. This is unnecessarily restricted. This criticism is not surprising, as the lemma-incorporation methods used exception-barring; in particular strategic withdrawal, which is subject to the criticism of retreating too far. Lakatos's comments on how to avoid this problem, by generalising the conjecture, are relevant here too. We discuss this in §13.1.

9.7 Given a global but not local counterexample

This has been one of the most difficult methods to implement, as it involves hidden assumptions, and in §9.2, we outlined various considerations which have influenced our implementation of it. As described in §9.2.2, in order to identify a problem lemma, there are two types of phenomena which cause surprise. The first is where it makes sense to use a concept to describe a given entity, but the entity gives a surprising result, as is the case when we remove the jacket from the cylinder, and it falls into two parts, leaving two disconnected circles, where all other entities had resulted in connected networks. The second type of surprise is where it only makes questionable sense to use a concept to describe a given entity, as the entity is not an example of the concept.

In order to implement the notion of surprise, we have considered previous ways of measuring surprise in automated theory formation. Colton et al. [1999] define the surprisingness of an equivalence conjecture as the number of concepts which appear in the construction path of one concept related by the conjecture, but not both. In AM, Lenat [1976] measured a concept as

surprising if it possessed a property not possessed by its parents. However, neither of these measures captures what we want, which is to measure the surprisingness of an *entity*, as opposed to a conjecture or concept. In Lakatos's example, it makes sense to apply the concept of removing a face from a polyhedron and stretching it flat, to the cylinder, but the result differs from what we get when we apply that concept to other polyhedra. Applying this concept to all of the other entities results in a graph which is connected, i.e. there is a property (or concept) which applies to everything except the surprising entity.

9.7.1 Modelling surprise

Recall that we are interested in two types of surprise: surprise caused by unexpected behaviour, and surprise caused by non-meaningful terms. We refer to these as type 1 and type 2 respectively. In this section, we define each of these types of surprise and describe our algorithm for each.

Surprise caused by unexpected behaviour
We define the first type of surprise as follows:

• surprisingness (type 1): an entity x is surprising with respect to a conjecture $P \rightarrow Q$ if there exists another conjecture C' such that x is the only counterexample to C' and C' has the form $P \rightarrow Q \wedge R$, for some concept R.

In order to identify the 'hidden assumption' in a conjecture, we have to break down the concepts in it, in particular the concept Q in the conjecture $P \rightarrow Q$. This is made easy for our purposes since for each of its concepts, HR records the construction path, i.e. the concepts to which production rules were applied to get the current concept. This ancestor list allows us to gradually dissect a concept until we find what we were looking for. Our algorithm for surprise caused by unexpected behaviour is shown in algorithm 11.

Surprise caused by non-meaningful terms
The second cause of surprise described in §9.2.2 concerns statements which are about non-meaningful terms. We have implemented this as:

• surprisingness (type 2): an entity m is **surprising** with respect to conjecture $P(x) \rightarrow Q(x)$ if $\neg P(m)$.

This is a simplification of the example in [Lakatos, 1976], in which the circle is argued to be a counterexample to the conjecture that 'all diagonals on all polygons create a new face'. To see this, consider representing this conjecture as: for all $p, e, diagonal(e, p) \rightarrow create\_new\_face(e, p)$,

where p is a polygon and e is an edge. The entity circle is considered to be surprising since if p = circle, there do not exist e such that $diagonal(e, p)$. While our definition does not capture predicates with more than one argument, it is clearly a simplification of such cases.

This type of surprise can be determined by testing to see whether the antecedent fails with respect to the global counterexample, i.e., whether, given counterexample m, the lemma is of the form $P(x) \rightarrow Q(x)$, and $\neg P(m)$. We describe an algorithm for finding a hidden lemma and generating an explicit faulty lemma, using surprise type 2, in algorithm 12. Note that this algorithm is only partially implemented.

Algorithm 11 Algorithm for finding a hidden lemma and generating an explicit faulty lemma, using surprise caused by unexpected behaviour (type 1)

Require: given a proof scheme and an entity which is a global but not local counterexample:

1: **for** $0 \leq i \leq$ length of proof scheme **do**
2: take each lemma $C_i = P(x) \rightarrow Q(x)$ in the proof scheme
3: take the ancestor list for the concept $Q(x)$
4: **for** $0 \leq j \leq$ length of ancestor list for $Q(x)$ **do**
5: take each concept $Q_j(x)$ in the ancestor list, starting with $Q(x)$ and going back to the core concept(s) from which all the ancestors were subsequently generated.
6: **for** $0 \leq k \leq$ length of concepts list in the theory **do**
7: for each concept $R_k(x)$ in the theory
8: perform the theory formation step "compose $Q_j(x) \wedge R_k(x)$"
9: **if** the outcome of the theory formation step is a concept **then**
10: **if** this concept has the same arity as $P(x)$ **then**
11: make the conjecture $C_i' = P(x) \rightarrow (Q_j(x) \wedge R(x))$
12: **else if** this concept has an arity which is greater than $P(x)$ by n **then**
13: apply the *exists* production rule with parameter [1] n times to the composed concept
14: when the arity of the concept $C(x)$, which results from the above step matches that of concept $P(x)$, make the conjecture $C_i' = P(x) \rightarrow C(x)$,
15: **end if**
16: **if** the entity is the only counterexample to C_i' **then**
17: return C_i as the hidden faulty lemma and C_i' as the explicit lemma.
18: **end if**
19: **end if**
20: **end for**
21: **end for**
22: **end for**

Note that in our implementation of surprise caused by unexpected behaviour, described in algorithm 11, it would not make sense to make some conjectures, for instance, between concepts with different types or arities. Therefore, before making the conjecture $C' = P(x) \rightarrow (Q_i(x) \wedge R(x))$, the student checks that $Q_i(x) \wedge R(x)$ is a concept, and if not, it moves onto the next concept. If $Q_i(x) \wedge R(x)$ actually *is* a concept, but has an arity which is larger than the arity of $P(x)$ by n, then the student applies the 'exists' production rule with parameter [1] n times to the composed concept, reducing the arity by 1 each time. When the arity of the composed concept $Q_i(x) \wedge R(x)$ matches that of concept $P(x)$, the student makes the conjecture $P(x) \rightarrow C(x)$, where $C(x)$ is the concept which results from repeated application of the 'exists' production rule to the concept $Q_i(x) \wedge R(x)$. This process in shown in stage *(v)* of the illustrative session below (§9.7.2).

Algorithm 12 Algorithm for finding a hidden lemma and generating an explicit faulty lemma, using surprise caused by non-meaningful terms (type 2)

Require: given a proof scheme and an entity m which is a global but not local counterexample:

1: **for** $0 \leq i \leq$ length of proof scheme **do**
2: take each lemma $C_i = P(x) \rightarrow Q(x)$ in the proof scheme
3: **if** C_i is of the form $P(x) \rightarrow Q(x)$, and $\neg P(m)$ **then**
4: generate the concept $P(m)$
5: generate the conjecture $C' = (P(x) \rightarrow Q(x)) \wedge P(m)$. The entity m is now a counterexample to C'
6: Return C_i as the hidden faulty lemma C, and C_i' as the explicit lemma.
7: **end if**
8: **end for**

9.7.2 Illustrative session

In this section, we describe a session with HRL which demonstrates our model of surprise caused by unexpected behaviour.

Initial information

A student was given the proof scheme with global conjecture "for all a: a is a polyhedron $\rightsquigarrow$ a is a polyhedron & the euler char of a is 2", and the cylinder entity, which is a counterexample.

Session details

The student reconstructed the following lemmas from the proof scheme which the teacher passed to it:

local lemmas:

1. for all a : a is a polyhedron → a is a polyhedron ∧ exists b (b is a graph ∧ remove a face from a and stretch it flat to get b ∧ b is planar)

2. for all a : a is a polyhedron ∧ the euler char of a is 2 ↔ a is a polyhedron ∧ exists b (b is a graph ∧ remove a face from a and stretch it flat to get b ∧ b is planar ∧ the euler char of graph b is 1)

3. for all a b : b is an integer ∧ a is a graph ∧ the euler char of the graph a is b ∧ a is planar ↔ b is an integer ∧ a is a graph ∧ a is planar ∧ exists c (c is a graph ∧ c is the triangulated version of a ∧ the euler char of the graph c is b)

4. for all a b : b is an integer ∧ a is a graph ∧ the euler char of the graph a is b ∧ a is triangulated ∧ ¬(a=`egh[gh;he;eg]`) ⇜⇝ b is an integer ∧ a is a graph ∧ exists c (c is a graph ∧ c is a with one triangle removed ∧ the euler char of the graph c is b)

5. for all a b : b is an integer ∧ a is a graph ∧ the euler char of the graph a is b ∧ a=`egh[gh;he;eg]` ↔ b is an integer ∧ a is a graph ∧ the euler char of the graph a is b ∧ a=`egh[gh;he;eg]` ∧ b=1

In order to evaluate whether the cylinder was surprising in the first sense, i.e., surprise caused by unexpected behaviour, with respect to any of the lemmas in the proof scheme, the student performed the steps below.

(i) It started cycling through all of the lemmas in the proof scheme. Firstly, it took lemma (1) as $C = P(x) \rightarrow Q(x)$, where $P(x)$ is the concept *for all a : a is a polyhedron*, and $Q(x)$ is the concept *a is a polyhedron ∧ exists b (b is a graph ∧ remove a face from a and stretch it flat to get b ∧ b is planar)*.

(ii) The student then found the ancestor list for $Q(x)$ [p31_0, p14_0, graph005, poly002] (shown in figure 9.5). We show the concepts below.

(iii) the student took the first concept in the ancestor list, $p31\_0$, and cycled through all of the concepts in its theory, attempting to compose them with this concept.

(iv) Stage (iii) failed, and so the student looked at the next concept in the ancestors list, $p14\_0$. When the student composed $p14\_0$ with the concept in its theory *graph002 = [a] : a is a graph ∧ a is a connected network*; it generated the new concept:

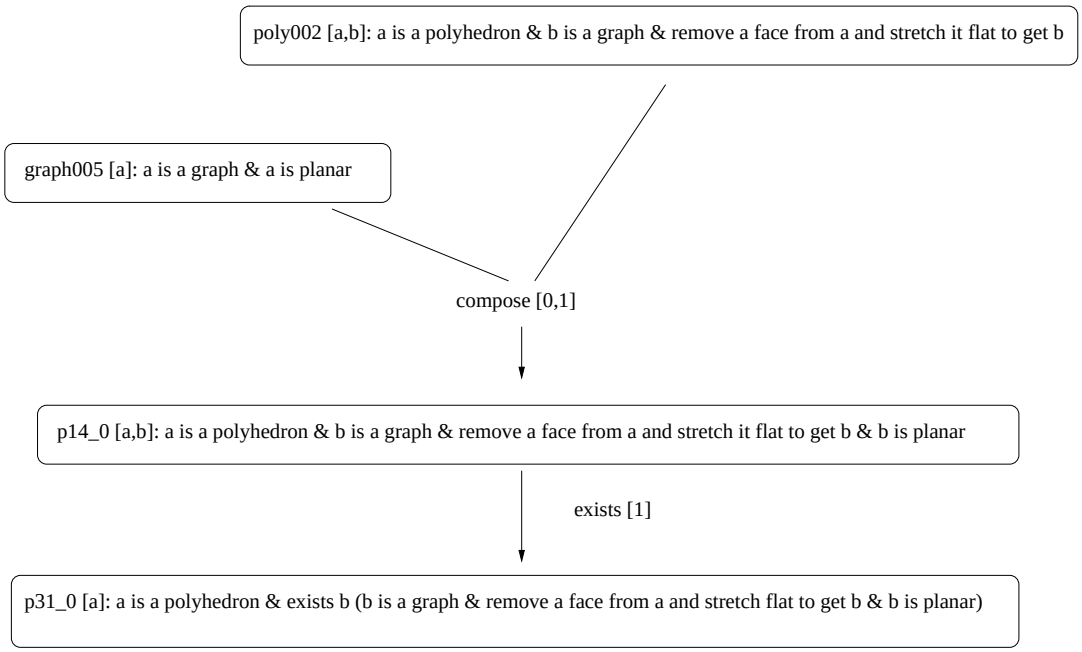


Figure 9.5: The ancestor tree for the right hand concept in lemma (1)

$p38\_0$ [a, b] a is a polyhedron $\wedge$ b is a graph $\wedge$ remove a face from a and stretch it flat to get b $\wedge$ b is planar $\wedge$ b is a connected network.

(v) Since this new concept was of a higher arity than $P(x)$, the student performed another theory formation step: $< p38\_0, exists[1] >$. This generated a further new concept:

[a] a is a polyhedron $\wedge$ exists b (b is a graph $\wedge$ remove a face from a and stretch it flat to get b $\wedge$ b is planar $\wedge$ b is a connected network)

(vi) Since this further new concept had the same arity as $P(x)$, the student made the conjecture C':

for all a : a is a polyhedron $\rightarrow$ a is a polyhedron $\wedge$ exists b (b is a graph $\wedge$ remove a face from a and stretch it flat to get b $\wedge$ b is planar $\wedge$ b is a connected network).

(vii) The student then found that this conjecture had exactly one counterexample: the cylinder. Therefore, the student evaluated that the cylinder *is* surprising with respect to lemma (1).

(viii) The student returns lemma (1) as the hidden faulty lemma, and C' as the explicit lemma.

9.7.3 Algorithm for hidden lemma-incorporation

For hidden lemma-incorporation, we need to find out whether there is a hidden lemma in the proof scheme and if so, identify it and generate an explicit lemma which the global counterexample *does* break. We do this by testing to see whether the global counterexample is surprising with respect to any lemma C in the proof scheme, where surprising is defined first as surprisingness 1, and then if unsuccessful, as surprisingness 2, described in §9.7.1. If so, then C is returned as the hidden faulty lemma. The student then generates an intermediate proof scheme in which the hidden faulty lemma is replaced by the explicit lemma, and then call global and local lemma-incorporation on the intermediate proof scheme, and return the result. We describe this procedure in algorithm 13. Note that this method is currently only implemented for implication conjectures in the proof scheme.

9.7.4 Illustrative session

We ran the agency with one student and a teacher. The student had the polyhedron cylinder, which was a counterexample to the global conjecture, but lemma 1 or lemma 2. In order to avoid the controversy over whether a global counterexample is local or not, as outlined in §9.2.3, we omitted lemma 3 from the proof when testing this method.

Initial information

The teacher started with one polyhedron, the cube, and the proof scheme in §9.7.2, where all lemmas and the global conjecture were proper, rather than near, implications and equivalences. The student had two polyhedra, the cube and the cylinder.

Session details

The teacher sent the proof scheme to the student and asked it to perform lemma-incorporation on it. The student reconstructed it as the proof scheme in §9.7.2. Since it had a global counterexample which was not also a local counterexample, it performed global but not local lemma-incorporation. It went through each of its lemmas in turn, to see whether the cylinder was surprising (type 1) with respect to it, and found that it was surprising with respect to *lemma 1*, as described in §9.7.2. It then generated the explicit lemma C':

for all a : a is a polyhedron $\rightarrow$ a is a polyhedron $\wedge$ exists b (b is a graph $\wedge$ remove a face from a and stretch it flat to get b $\wedge$ b is planar $\wedge$ b is a connected network).

The student then replaced *lemma 1* in its proof scheme with C', and performed local and global lemma-incorporation. The outcome was the following modified conjecture and proof scheme:

Algorithm 13 Algorithm for global but not local lemma-incorporation

Require: Given a proof scheme where there are counterexamples to the global conjecture, but these counterexamples are not counterexamples to any of the lemmas in the proof, and none of the lemmas have counterexamples which are related to the global counterexamples. Let the global conjecture be a near-implication $P \rightsquigarrow Q$. Then:

go through the proof scheme and take each lemma in turn, testing each to see whether the global counterexample is surprising in the first sense (type 1) with respect to the lemma
if so **then**
    return this lemma as the hidden faulty lemma, and generate another conjecture, which the global counterexample *is* a counterexample to, as the explicit lemma
else
    go through the proof scheme and take each lemma in turn, testing each to see whether the global counterexample is surprising in the second sense (type 2) with respect to the lemma
    if so **then**
        return this lemma as the hidden faulty lemma, and generate another conjecture, which the global counterexample *is* a counterexample to, as the explicit lemma
    else
        return the proof scheme unchanged
    end if
end if
if an explicit lemma has been found **then**
    generate an intermediate proof scheme in which the hidden faulty lemma is replaced by the explicit lemma
    perform explicit lemma-incorporation, i.e., global and local lemma incorporation on the intermediate proof scheme, and return the result
end if

Global conjecture:

for all a : a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar & b is a connected network) → a is a polyhedron & the euler char of a is 2

Proof scheme:

1. for all a : a is a polyhedron ⇝ a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar & b is a connected network)

2. for all a : a is a polyhedron & the euler char of a is 2 ↔ a is a polyhedron & exists b (b is a graph & remove a face from a and stretch it flat to get b & b is planar & the euler char of graph b is 1)

3. for all a b : b is an integer & a is a graph & the euler char of the graph a is b & a is planar ↔ b is an integer & a is a graph & a is planar & exists c (c is a graph & c is the triangulated version of a & the euler char of the graph c is b)

4. for all a b : b is an integer & a is a graph & the euler char of the graph a is b & a is triangulated & -(a=egh[gh;he;eg]) ⇜⇝ b is an integer & a is a graph & exists c (c is a graph & c is a with one triangle removed & the euler char of the graph c is b)

5. for all a b : b is an integer & a is a graph & the euler char of the graph a is b & a=egh[gh;he;eg] ↔ b is an integer & a is a graph & the euler char of the graph a is b & a=egh[gh;he;eg] & b=1

9.7.5 Discussion

Since this method involves surprise and implicit assumptions, it has been the most difficult of the lemma-incorporation methods to implement. The fact that HR records the construction history of each of its concepts has made it possible to trace concepts back, and identify hidden meanings. This has enabled us to write an algorithm which, although a simplified version, captures Lakatos's method. This is shown in our illustrative session in §9.7.2.

Note that although the global conjecture in the illustrative session in §9.7.4 now has no counterexamples, there are still known problems in lemmas 1 and 4. Lakatos did not specify exactly what to do with a faulty lemma once it had been used to modify the global conjecture, and we have not extended this aspect of Lakatos's theory in our implementation. It would be sensible to remove the faulty lemma from the proof: however, this would affect the proof structure and a new lemma should be generated.

9.8 The method of proofs and refutations

The method of proofs and refutations is the method of lemma incorporation "rechristened" [Lakatos, 1976, 49]. The only addition is that the method of finding counterexamples is made more explicit. This consists of using both the global conjecture and the local lemmas in a proof scheme to suggest counterexamples, by looking for entities which violate them. For the sake of completeness, we include an illustrative session below.

9.8.1 Illustrative session

We gave the global conjecture in the group theory example to the MACE model generator McCune [2001], as shown below:

for all g, h, $k \in G$, $(((g*h) = (g*k)) \rightarrow (h = k))$,

and asked it to generate models of size 1, size 2, etc. up to size 8, and to spend 10 seconds on each search. It found counterexamples in which the product was 0 for every pair of elements. We show examples of groups of size 2 and 3 in table 9.3. To see why the first of these is a counterexample, let g = 1; h = 1 and k = 0. Then $g*h = 1*1 = 0$, and $g*k = 1*0 = 0$, but $1 \neq 0$.

	0	**1**
0	0	0
1	0	0

	0	**1**	**2**
0	0	0	0
1	0	0	0
2	0	0	0

Table 9.3: Two groups generated by MACE as counterexamples to the conjecture 'for all $ghk \in G, (((g*h) = (g*k)) \rightarrow (h = k))$'

We have not implemented this aspect of the method of proofs and refutations within HRL, as it is beyond the scope of this project.

9.9 Related work

Bundy et al. [2004] have used Lakatos's rational reconstruction of Cauchy's proof to help them to provide a logic-based account of the way in which mathematicians discover and present proofs. They argue that mathematicians do not work within the Hilbertian notion of proof, as

proofs tend to be difficult to check and are often flawed, which would not be the case for a Hilbertian proof. Bundy et al use Lakatos's ideas to ask how it is possible for errors in a faulty proof to go unnoticed for years – even when counterexamples are known.

Their account of this phenomenon consists of the constructive ω-rule, which is a refinement of the ω-rule. The ω-rule states that we can infer $\phi(x)$ for all natural numbers x, so long as we can prove $\phi(1)$, $\phi(2)$, $\phi(3)$, $\phi(4)$,... The constructive ω-rule takes a variant of this as its first step – specifying that we must be able to find proofs $\phi(i)$ for a *finite* number of natural numbers. It has two further requirements. Step two is that each proof $\phi(i)$ must share a common structure, so that we can generalise and build a program $proof_\phi$ which takes a natural number n as input and outputs a proof $\phi(n)$. This function is a *schematic proof*, called $proof_\phi : \phi(n)$. Step three is that we can prove by meta-level mathematical induction that $proof_\phi : \phi(n)$ is valid for all natural numbers; i.e. we must be able to show *(i)* $proof_\phi : \phi(0)$, and *(ii)* $proof_\phi : \phi(n) \vdash proof_\phi : \phi(n+1)$.

Bundy et al argue that we can see Cauchy's proof in terms of the constructive ω-rule. An initial proof was offered for the cube (step one), and this type of proof was generalised to all polyhedra (step two). However, the third step, of verification, was omitted, and this is why the proof was not sound and counterexamples arose. Bundy et al suggest that omission of the third step is not uncommon, and they hypothesise that their constructive ω-rule provides a cognitive model of how some proofs are discovered and presented. In particular, they suggest that their notion of schematic proofs may account for errors in proofs – both why they arise at all, and why they can be difficult to find, in a way that the conventional logical account fails to do.

The constructive ω-rule has been implemented in two systems, [Baker, 1993] and [Jamnik, 2001b]. These take as input a conjecture and uniform proofs of examples from the domain (step one). They output a schematic proof for which both step two and step three have been performed.

Our works differs from Bundy et al's approach in that we have not attempted to answer the question of *how it is possible for errors in a faulty proof to go unnoticed?* Instead, our aim has been to provide a computational representation of Lakatos's answer to the question of *how can counterexamples be used to improve a faulty proof and conjecture?* In doing so, we have had to answer the following questions (see the relevant sections):

• how can we represent an informal proof in our system? (§8.3)

• how can a computer program uncover hidden assumptions in a proof? (§9.7)

• given a counterexample, how can a computer program determine whether it is global or local? (§9.4)

• how can a computer program perform local/global/hidden lemma incorporation? (sections 9.5, 9.6 and 9.7).

9.10 Summary

We have discussed Lakatos's method of lemma-incorporation, including local but not global, both global and local, and global but not local lemma-incorporation. We have described our implementation of all three variations of the method, and given illustrative sessions in each case. We have considered how the method may be applied to further mathematical domains, including geometry and group theory. We hold that the process of providing a formal representation of Lakatos' method of lemma-incorporation, which we have described in this chapter and the previous one, has enabled us to answer questions such as: how can we represent an informal proof? How can we uncover hidden assumptions in a proof? How can we modify a faulty conjecture or a faulty proof, given a counterexample? How can we formalise the surprise we feel when an example behaves in an unexpected manner in a proof? We have also briefly considered how we could extend HRL using the model generator MACE, in order to generate counterexamples in the way described in Lakatos's method of proofs and refutations. Finally, we have discussed work which is related to Lakatos's method of lemma-incorporation.

Chapter 10

Testing hypotheses in HRL

10.1 Introduction

In chapters 5 to 9, we described our implementation of each of Lakatos's methods, except for monster-adjusting which we leave as further work (see chapter 13). At the end of each chapter, we have presented illustrative sessions, and a discussion about ways in which our implementation has extended or clarified Lakatos's methods. In the following three chapters, we evaluate our project. We argue that the process of producing an algorithm and subsequent implementation of Lakatos's methods has not only enabled us to improve upon his theory, but also to evaluate both his theory and our extended theory. The three chapters are distinguished by the criteria to which we refer. In this chapter we use the interestingness measures in HRL to evaluate theories which it has produced, and we test hypotheses about our system. In chapter 11 we appeal to philosophical criteria of a good theory, and argue that our interpretation and extension of Lakatos's methods is useful from this perspective. Finally, in chapter 12, we argue that automating Lakatos's methods has useful application to the field of automated theorem proving.

Our primary aim in this chapter is to evaluate three main hypotheses:

• it is possible to fine-tune the method of surrender;

• it is possible to fine-tune the method of monster-barring;

• it is possible to fine-tune the method of exception-barring.

Our secondary aim is to evaluate the claim that Lakatos's methods apply both to other types of conjecture and to other domains.

In order to achieve these aims, we run HRL with different parameter settings, and present and discuss the output. The purpose of this process is threefold: firstly, we wish to provide support for, or to falsify our three hypotheses, that changing the parameters to HRL has an effect on its output. Secondly, we wish to investigate which parameter settings produce certain desired behaviours, in particular whether the 'Lakatos variable' (i.e. the variable which most closely mirrors an aspect of a method as Lakatos described it), or an extension, is better in different situations. Thirdly, we wish to show HRL using different methods and working in different domains, with different types of conjectures. This would demonstrate the generality of Lakatos's methods. Thus, we evaluate both Lakatos's methods and our extended theory of mathematical discovery and justification.

The chapter is organised as follows: firstly, in §10.2, we describe our experimental setup, recapping the variables in HRL at both the student and the agency level. We also state the default values and state where we have varied these. The following three sections; §10.3, 10.4 and 10.5 contain our evaluation of our three hypotheses, giving the particular experimental setup, evaluation criteria, results and discussion in each. We conclude the chapter in §10.6.

10.2 Experimental setup

In order to investigate our three hypotheses, and to achieve our secondary aim, we vary the method used, domain, and conjecture type which the teacher requests. We have also varied interestingness weightings between the students.

10.2.1 Variables in HRL

Within HRL we have continued Colton's methodology of allowing for empirical experimentation rather than making design decisions, where possible. This allows us to test our hypotheses by differing the variables. Below we show these variables, the values they take, and their default values where relevant.

10.2.2 Variables at the student level

These variables are flags which are set in the individual input file to each student, at the start of a session. Different students might have different values for each variable.

1. **Domain.** HRL can work in many different domains. We test it in group theory, an animals data set, and number theory.

2. **Method of generating false conjectures.** These are different ways of automatically generating false conjectures:

   (a) **near-conjectures:** these are either near-equivalences or near-implications and hold for x% of a student's entities, where $0 \leq x < 100$. The default is that students do *not* generate near-conjectures.

   (b) **data distribution:** we distribute the data by concept, by natural ordering or randomly between different students.

3. **Discussion.** These are variables which direct the length of discussion and the content which is taken from them. They consist of:

   (a) **number of independent work phases:** this is the number of independent work phases which are performed altogether in a session. This takes any value in Z^+, and has a default of 10.

   (b) **length of independent work phase:** this is the length of each independent phase, measured by the number of theory formation steps performed. This takes any value in Z^+, and has a default of 10.

   (c) **threshold to add conjecture to theory:** any conjecture which arises during the discussion phase must be more interesting than this threshold in order for the student to add it to its own theory after the discussion. This takes any non-negative real value, and has a default of 0.

   (d) **threshold to add concept to theory:** any concept which arises during the discussion phase must be more interesting than this threshold in order for the student to add it to its own theory after the discussion. This takes any non-negative real value, and has a default of 0.

4. **Lakatos Method.** These variables dictate which Lakatos method should be used and how it is to be applied.

   (a) **surrender:** these variables dictate when a student should perform surrender (see §10.3.1).

   (b) **monster-barring:** these variables dictate when, and how to perform monster-barring (see §10.4.1).

   (c) **exception-barring:** these variables dictate which type of exception-barring should be performed, and when (see §10.5.1).

10.2.3 Variables at the agency level

These variables are flags set at the start of a session which are universal to the agency.

1. **Conjecture request.** This is the type of conjecture which the teacher requests, and take the following values: conjecture, implication, near-implication, equivalence, near-equivalence, non-exists. The default conjecture is "conjecture".
2. **Number of students.** HRL consists of a variable number of students, and one teacher. There is no upper limit on the number of students. The default number is 2.
3. **Use communal piecemeal exclusion.** If this flag is set, then the class perform piecemeal exclusion together rather than individually. This takes either true or false as input, with the default being false.

10.2.4 Variable settings

Unless otherwise specified, the students were set to make near equivalence conjectures which held for 60% of their entitites. The data distribution is shown below, in §10.2.5. The number of independent work phases was set to 10, and the length of each independent work phase was set to 20.

In order to prevent conjectures which are dull from being discussed, we introduced a threshold for the purposes of the hypothesis testing. When a student received a request to send a certain type of conjecture, it sent the first conjecture it found in its theory of the appropriate type, whose interestingness is greater than 1.0.

There are many further variables and parameter values in HRL, since it encompasses all those in HR. For instance, the production rules and parameters, interestingness measures and search technique to be used in the individual theory formation are all variables to be set at the start of each session. Where relevant, we state these, otherwise we use default values. be found

10.2.5 Domains

Group Theory

We tested HRL on an algebra data-set. This consisted of fourteen groups, one semi-group and three groupoids. The core concepts were group, element, the binary operator, identity, inverse,

associativity, and whether the left hand cancellation law held. The first student worked in group theory, the second in semi-group theory, and the third in groupoid theory, as shown below.

Student 1 had 14 groups: one of size one, one of size two, one of size three, two of size four, one of size five, two of size six, one of size seven, and five of size eight.

Student 2 had one semi-group, SG1: a set with elements $\{0,1,2\}$, where the $*$ operator is defined in table 10.1. It is associative, but has no identity, and no inverse.

	0	**1**	**2**
0	0	2	2
1	2	1	2
2	2	2	2

Table 10.1: The $*$ operator for SG1

Student 3 had three groupoids. The first, GP1 is the trivial groupoid: a set with element $\{0\}$, where the $*$ operator is the first table defined in 10.2. This is associative, has identity 0, and inverse 0.

The second groupoid, GP2 is a set with elements $\{0,1,2\}$, where the $*$ operator is the second table defined in 10.2. This is not associative, has identity 0, and does not have a unique inverse for each element.

The third groupoid, GP3 is a set with elements $\{0,1,2\}$, where the $*$ operator is the third table defined in 10.2. This is not associative, has no identity, and therefore no inverse for any element.

	0
0	0

	0	**1**	**2**
0	0	1	2
1	1	0	0
2	2	0	0

	0	**1**	**2**
0	2	2	1
1	2	2	1
2	2	2	1

Table 10.2: The $*$ operator for GP1, GP2 and GP3

We set the interestingness weighting of a conjecture to:
interestingness = surprisingness * 0.5 + comprehensibility * 0.5 + applicability * 0.5 + plausibility * 0.5.

Animal taxonomy

We tested HRL on a machine learning data-set from Inductive Logic Programming. This consisted of the following 18 animals: dog, dolphin, platypus, bat, trout, herring, shark, eel, lizard, crocodile, t-rex, turtle, snake, eagle, ostrich, penguin, cat, and the dragon. The core concepts were: whether they were covered by hair, scales or feathers; the number of legs they have; whether they are homeothermic; whether they produce milk, lay eggs, or have gills; what sort of habitat they live in; and what class of animal they are: mammal, fish, reptile or bird. We distributed the data as shown below:

Student 1: all animals except the platypus. Interestingness measures were weighted towards conjectures which were surprising, as follows:

interestingness = surprisingness * 3 + comprehensibility * 0.5 + applicability * 0.5 + plausibility * 0.5

Student 2: all animals except the trout. Interestingness measures were weighted towards conjectures which were high in comprehensibility, as follows:

interestingness = surprisingness * 0.5 + comprehensibility * 3 + applicability * 0.5 + plausibility * 0.5

Student 3: all animals except the cat. Interestingness measures were weighted towards conjectures which had a high applicability, as follows:

interestingness = surprisingness * 0.5 + comprehensibility * 0.5 + applicability * 3 + plausibility * 0.5

Note that although we would normally consider the t-rex and the dragon to be odd, and thus liable to be barred as monsters, the domain file contains no information about whether the animals are now extinct, or are mythical, and so it would be unfair to penalise HRL for not identifying these two as potential monsters.

Number Theory

We tested HRL on an number theory data-set. This consisted of the integers 0 - 60, and the core concepts of integer, divisor, multiplication, and addition. The data was distributed as shown below:

Student 1: integers 0 – 20

Student 2: integers 20 – 40

Student 3: integers 40 – 60

We set the interestingness weighting of a conjecture to:
interestingness = surprisingness * 0.5 + comprehensibility * 0.5 + applicability * 0.5 + plausibility * 0.5

10.3 It is possible to fine-tune the method of surrender

There needs to be some limit on our system as to when it can apply each method. Conjectures which have been over-modified may become dull or too specific (for instance after repeated application of piecemeal exclusion). Resources spent discussing such conjectures will prevent the system from investigating more interesting paths. In this section, we experiment to see how best to measure when a conjecture is over-modified or dull. We consider the following five variables: the number of modifications already performed on a conjecture, the interestingness of the conjecture compared to a user-given threshold, the interestingness of the conjecture compared to the average interestingness of all the conjectures in the theory, the plausibility of the conjecture compared to a user-given threshold, and the domain of application of the conjecture compared to a user-given threshold.

10.3.1 Experimental setup

We investigate the five ways of testing whether to surrender a conjecture by testing the values as shown below. The value false indicates that the variable in question is not considered.

1. **the number of modifications:** 0 (a), 1 (b), 2 (c), 3 (d), false
2. **the interestingness threshold of the conjecture:** 0.5 (a), 0.75 (b), 1.0 (c), false
3. **whether to compare the interestingness of the conjecture with the average interestingness of all the conjectures in the theory:** true (a), false
4. **the plausibility threshold of the conjecture:** 0.7 (a), 0.8 (b), 0.9 (c), false
5. **the domain of application of the conjecture (which is divided by the number of entities in the student's database):** 0.4 (a), 0.6 (b), 0.8 (c), false

We ran the agency 42 times; 14 times each in group theory, animal taxonomy and number theory, each time with three students. The students ran for 20 theory formation steps in each

individual work phase, and they performed 10 individual work phases. For each session, the three students were set to perform surrender where the students had identical values for the surrender variables. They were also set to perform piecemeal exclusion. Students were set to make near-equivalences which held for 60% of their entities. The input values for the surrender variables, in terms of the key above, for sessions 1-14 are presented table 10.3.

Session	**student 1**	**student 2**	**student 3**
1	surr-1a	surr-1a	surr-1a
2	surr-1b	surr-1b	surr-1b
3	surr-1c	surr-1c	surr-1c
4	surr-1d	surr-1d	surr-1d
5	surr-2a	surr-2a	surr-2a
6	surr-2b	surr-2b	surr-2b
7	surr-2c	surr-2c	surr-2c
8	surr-3a	surr-3a	surr-3a
9	surr-4a	surr-4a	surr-4a
10	surr-4b	surr-4b	surr-4b
11	surr-4c	surr-4c	surr-4c
12	surr-5a	surr-5a	surr-5a
13	surr-5b	surr-5b	surr-5b
14	surr-5c	surr-5c	surr-5c

Table 10.3: Experiments on the surrender variables

10.3.2 Evaluation criteria

When evaluating the different variable settings in the method of surrender, we are interested in four types of conjecture:

(i) conjectures which should have been surrendered and were (true positives),
(ii) conjectures which should have been surrendered but were not (false negatives),
(iii) conjectures which should not have been surrendered but were (false positives), and
(iv) conjectures which should not have been surrendered and were not (true negatives).

(i) and *(ii)* involve dull, or uninteresting conjectures, and *(iii)* and *(iv)* conjectures which are interesting. We would like to find variables settings which maximise the number of conjectures of type *(i)* and *(iv)*, and minimise the number of conjectures of type *(ii)* and *(iii)*.

10.3.3 Results

In figure 10.1, we show the results in terms of the number of conjectures which each student either surrendered or considered surrendering but opted to modify, and the average interestingness of these conjectures as evaluated by the student. We show this for each domain, in terms of the four quadrants *(i)* to *(iv)* above, where we take 0.8 as our threshold for interestingness. This value is an average of the average interestingness values of all conjectures in all of the students' theories.

In figure 10.2 we show the results in terms of each of the five variables that we test. For each domain, we give the number of conjectures for each value of a given variable, which were in either of the quadrants which it is desirable to maximise (*(i)* and *(iv)*), shown above the *x*-axis; and the number of conjectures which were in either of the quadrants which it is desirable to minimise (*(ii)* and *(iii)*), shown below the *x*-axis. We have calculated this accumulatively for each student in each session. For instance, in session 1 in group theory, the second student surrendered two conjectures which it evaluated as uninteresting (in quadrant (*(i)*), and the third student surrendered four conjectures which it evaluated as uninteresting (also in quadrant (*(i)*). We show this in the first graph in figure 10.2, in which the *x*-axis denotes the number of modifications variable (set to 0 in session 1), by the coordinate $(0,6)$.

Recall that the results show the subjective values of the individual students. That is, in this set of experiments we are considering which conjectures the students themselves believe that they should surrender or modify.

10.3.4 Discussion

The best setting for the number of modifications was 0, as shown in the first graph in figure 10.2. This was a surprising result, as it amounted to the situation in which a student surrenders any conjecture for which it can find a counterexample, i.e., the default setting in which a student was set to perform surrender, but all five values are false. This was one of the best indicators of all of the variables that we tested. The reason was that only students working in the animal taxonomy domain found conjectures which they actually considered interesting enough to modify. This indicates that we should change the interestingness setting themselves in future experiments. The worst value of all those we tested, was setting the number of modifications to 3. This produced only conjectures in quadrants *(ii)* or *(iii)* in every domain. This was because no conjectures had been modified more than three times, and so it amounted to modifying every conjecture which was discussed, despite the fact that most were evaluated by the students as uninteresting. Again, before concluding that this value is ineffective, it would be useful to

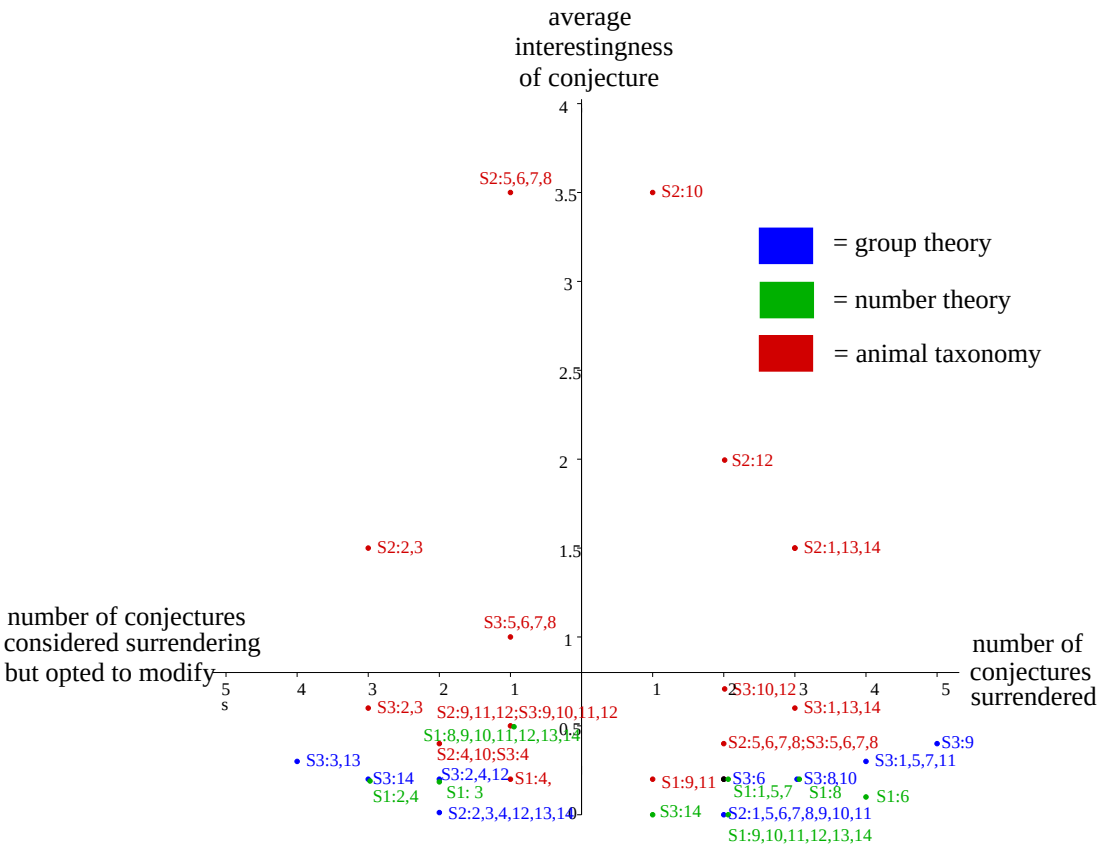


Figure 10.1: Results from testing the surrender variables. The numbers denote the sessions, and student one is represented as S1, etc.. The bottom right quadrant contains true positives; the top right quadrant contains false positives; the bottom left quadrant contains false negatives; and the top left quadrant contains true negatives.

experiment further, changing the interestingness settings of the students, as well as comparing them with an external evaluation of interestingness.

Using the interestingness threshold was the most successful variable of the five, and was the only variable to obtain conjectures only in quadrants *(i)* or *(iv)* (above the *x*-axis in figure 10.2). This was unsurprising, since the interestingness measure we used in presenting the results is that of the individual agents, rather an external measure. In future experiments, it would be useful to use an external measure. The results, however, were inconclusive as to which value the threshold should take.

The third variable, comparing the interestingness of the conjecture under discussion with the average interestingness of all of the conjectures in the theory, was very effective. The only domain in which setting this variable to true led to a student modifying rather than surrendering a conjecture, which the student evaluated as being uninteresting (quadrant *(ii)*), was in number theory. This was one of the most successful indicators which we tested.

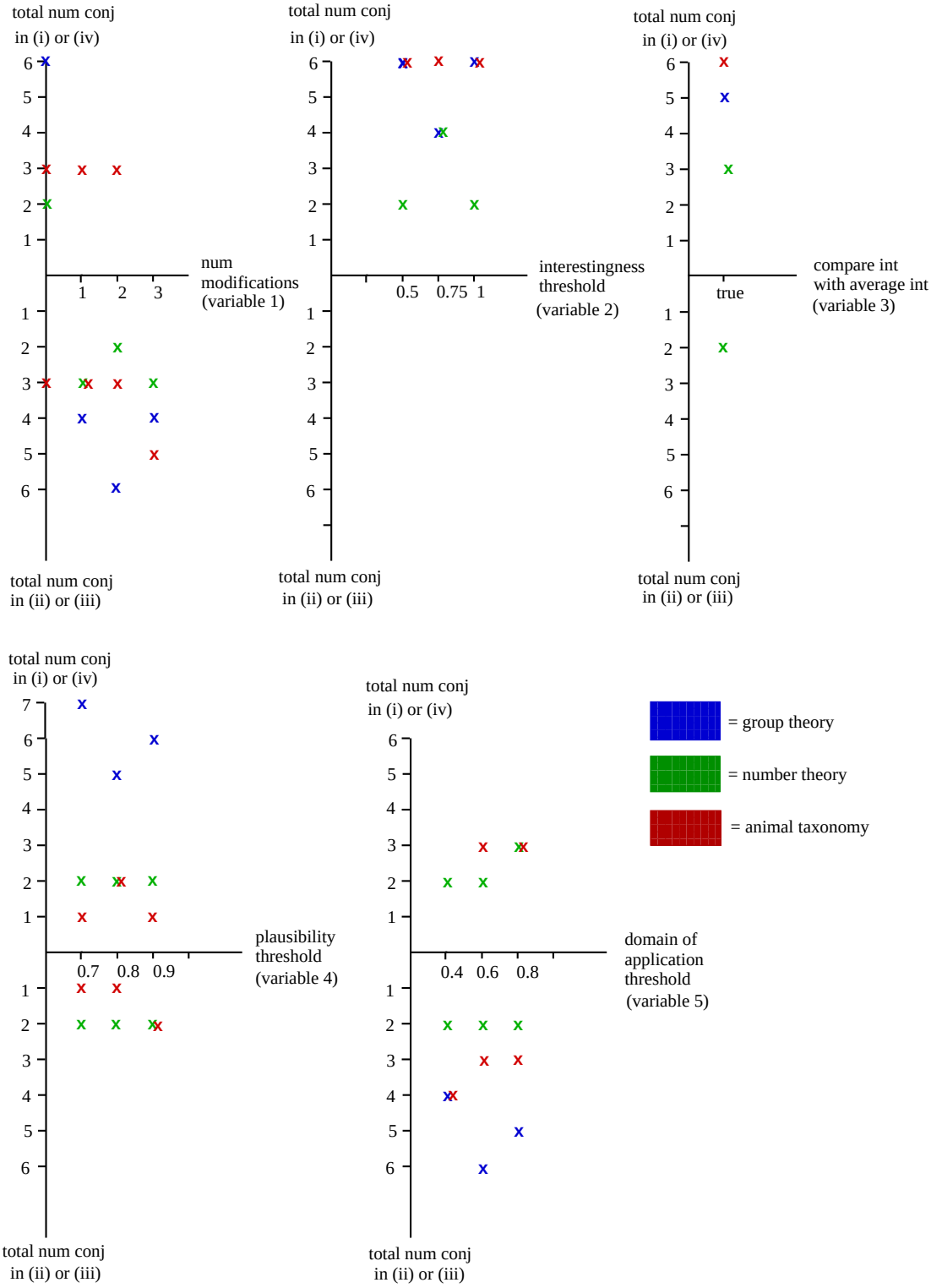


Figure 10.2: The number of conjectures for which each of the five surrender variables give us desirable (above the x-axis) or undesirable (below the x-axis) results

Testing the plausibility of the conjecture under discussion was a useful indicator, and the optimal setting in our experiments was 0.7.

Finally, using the domain of application was not a useful variable, with more conjectures being below the *x*-axis than above for every setting.

If we add the number of conjectures in quadrants *(i)* and *(iv)*, and subtract the number of conjectures in quadrants *(ii)* and *(iii)*, for each variable, then we obtain the following hierarchy of variables and their settings: 2a, 2b, 2c (14); 3a (12); 1a (8); 4a (7); 4b (6); 4c (5); 5c (-4); 5b (-6); 1c, 5a (-8); 1d (-12). This provides a useful guide as to which of the five variables, and which value of a given variable, enabled the students to modify those conjectures which they evaluated as interesting, and surrender those conjectures which they evaluated otherwise.

Many conjectures which were proposed were not considered interesting at all by some students. In these situations, the students either had very different entities in their databases, or very different interestingness measures. We could investigate adding a lower, as well as an upper threshold on the interestingness of a conjecture when considering whether to reject it or not. That is, define x and y such that $0 < y < x$, and if the interestingness of a conjecture lies between y and x then surrender the conjecture; if it is either less than y or greater than x, then further investigation may be productive. The type of investigation would depend on the interestingness: for instance, if it were less than y, then an agent might ask the student who proposed the conjecture why they thought it was interesting, and if greater than x, then the student might attempt to modify the conjecture alone. HRL currently uses only the upper threshold.

One further way of evaluating whether to surrender a conjecture or not might be to attempt to modify it, and, if successful, evaluate the modification. If an agent either could not modify a conjecture, or did so but evaluated the modification as dull, then it would surrender the original conjecture.

10.4 It is possible to fine-tune the method of monster-barring

10.4.1 Experimental setup

Monster-barring variables consist of determining when to propose to bar an object (variables 1 - 8 below), how to perform monster-barring (variable 9), and how to evaluate a proposal to bar an object (variables 2, 4 and 10). We tested the values shown below.

1. **use monster barring:** true (a); false (b)
2. **use breaks conj under discussion:** true (a); false (b)
3. **use percentage conjectures broken:** true (a); false (b)
4. **monster barring minimum:** 0 (a); 10 (b); 20 (c); 30 (d); 40 (e); 50 (f); 60 (g); 70 (h); 80 (i); 90 (j); 100 (k)
5. **use culprit breaker:** true (a), false (b)
6. **use culprit breaker on conjecture under discussion:** true (a), false (b)
7. **use culprit breaker on all conjectures:** true (a), false (b)
8. **monster barring culprit min:** 0 (a); 10 (b); 20 (c); 30 (d); 40 (e); 50 (f); 60 (g); 70 (h); 80 (i); 90 (j); 100 (k)
9. **monster barring type:** "vaguetovague" (a); "vaguetospecific" (b)
10. **accept strictest:** true (a), false (b)

We describe the experiments we ran in sections 10.4.2, 10.4.3 and 10.4.4 below.

10.4.2 When to propose to bar an object

We tested when to propose to bar an object by running the agency with three students: the first of whom took the values shown in variables 1-8, with the flag "use monster-barring" always set to true (1a); and the other two students were set to perform surrender, and to accept the strictest definition if a monster-barring proposal were made. Accepting the strictest definition is the Lakatos variable, i.e. the variable which is the closest interpretation of the earliest part of Lakatos's theory.

We set the students to run for 10 theory formation steps to each individual work phase, and to perform 10 individual work phases. The input data from 24 sessions are shown in table 10.4, and the results are discussed in §10.4.6.

10.4.3 How to perform monster-barring

The "monster barring type" variable concerns the two different ways of performing monster-barring: "vaguetovague", and "vaguetospecific".

Session	student 1	student 2	student 3
1	mb-2a	surr	surr
2	mb-3a-4a	surr	surr
3	mb-3a-4b	surr	surr
4	mb-3a-4c	surr	surr
5	mb-3a-4d	surr	surr
6	mb-3a-4e	surr	surr
7	mb-3a-4f	surr	surr
8	mb-3a-4g	surr	surr
9	mb-3a-4h	surr	surr
10	mb-3a-4i	surr	surr
11	mb-3a-4j	surr	surr
12	mb-3a-4k	surr	surr
13	mb-5a-6a	surr	surr
14	mb-5a-7a-8a	surr	surr
15	mb-5a-7a-8b	surr	surr
16	mb-5a-7a-8c	surr	surr
17	mb-5a-7a-8d	surr	surr
18	mb-5a-7a-8e	surr	surr
19	mb-5a-7a-8f	surr	surr
20	mb-5a-7a-8g	surr	surr
21	mb-5a-7a-8h	surr	surr
22	mb-5a-7a-8i	surr	surr
23	mb-5a-7a-8j	surr	surr
24	mb-5a-7a-8k	surr	surr

Table 10.4: Experiments to determine *when* to perform monster-barring

In sessions 1 - 24, this takes its default value “vaguetospecific”. To test this variable, we performed sessions 25 - 48, shown in table 10.5. These take the same variable values as sessions 1 - 24 except that the “monster barring type” flag is set to “vaguetovague”. We discuss the results from sessions 25 - 48 in §10.4.7.

10.4.4 How to evaluate a proposal to bar an object

The first variable concerning how to evaluate a proposal to bar an object is 2: “use breaks conjecture under discussion”, which is set to true (a) or false (b). For three students, the four different combinations for this variable are: *(i)* 2a (student 1), 2a (student 2), 2a (student 3); *(ii)* 2a, 2a, 2b; *(iii)* 2a, 2b, 2b; and *(iv)* 2b, 2b, 2b. In sessions 49 - 52 we test variable 2, with the results shown in table 10.6.

Session	student 1	student 2	student 3
25	mb-2a-9a	surr	surr
26	mb-3a-4a-9a	surr	surr
27	mb-3a-4b-9a	surr	surr
28	mb-3a-4c-9a	surr	surr
29	mb-3a-4d-9a	surr	surr
30	mb-3a-4e-9a	surr	surr
31	mb-3a-4f-9a	surr	surr
32	mb-3a-4g-9a	surr	surr
33	mb-3a-4h-9a	surr	surr
34	mb-3a-4i-9a	surr	surr
35	mb-3a-4j-9a	surr	surr
36	mb-3a-4k-9a	surr	surr
37	mb-5a-6a-9a	surr	surr
38	mb-5a-7a-8a-9a	surr	surr
39	mb-5a-7a-8b-9a	surr	surr
40	mb-5a-7a-8c-9a	surr	surr
41	mb-5a-7a-8d-9a	surr	surr
42	mb-5a-7a-8e-9a	surr	surr
43	mb-5a-7a-8f-9a	surr	surr
44	mb-5a-7a-8g-9a	surr	surr
45	mb-5a-7a-8h-9a	surr	surr
46	mb-5a-7a-8i-9a	surr	surr
47	mb-5a-7a-8j-9a	surr	surr
48	mb-5a-7a-8k-9a	surr	surr

Table 10.5: Experiments to determine *how* to perform monster-barring

The second relevant variable is the monster-barring minimum (4). We tested twelve combinations, where this flag was set to 30% (d), 60% (g) and 90% (j), for three students. These are shown in sessions 53 - 64, shown in table 10.6.

The final relevant variable is the "accept strictest" flag (10). This is set to true (a) or false (b). We test all combinations of this variable in sessions 65 - 68, shown in table 10.6.

The second and third students were set to perform surrender by default in sessions 49 - 68.

In order to enable all students to propose to monster-bar an object, in sessions 69 - 92, table 10.7, we gave all of the students the value of the monster-barring variable which first student had in sessions 1 - 24. This prevented the distribution of the data affecting which entities could be highlighted as monsters. We ran HRL in all three domains.

Session	student 1	student 2	student 3
49	mb-2a	mb-2a	mb-2a
50	mb-2a	mb-2a	mb-2b
51	mb-2a	mb-2b	mb-2b
52	mb-2b	mb-2b	mb-2b
53	mb-4d	mb-4d	mb-4d
54	mb-4d	mb-4d	mb-4g
55	mb-4d	mb-4d	mb-4j
56	mb-4d	mb-4g	mb-4g
57	mb-4d	mb-4g	mb-4j
58	mb-4d	mb-4j	mb-4j
59	mb-4g	mb-4d	mb-4g
60	mb-4g	mb-4g	mb-4g
61	mb-4g	mb-4g	mb-4j
62	mb-4g	mb-4j	mb-4j
63	mb-4j	mb-4d	mb-4j
64	mb-4j	mb-4j	mb-4j
65	mb-10a	mb-10a	mb-10a
66	mb-10a	mb-10a	mb-10b
67	mb-10a	mb-10b	mb-10b
68	mb-10b	mb-10b	mb-10b

Table 10.6: Experiments to determine how best to react to a proposal to bar an object

10.4.5 Evaluation criteria

The purpose of these experiments was to determine which entities were highlighted during each session as potential monsters. We then compared these to certain entities we expected to be highlighted, which were as follows:

Group theory

The semi-group SG1, and groupoids GP2 and GP3 are odd entities compared to standard groups. Therefore, of the seventeen different algebras in our group theory domain, we considered three to be odd.

Animal taxonomy

The platypus was first brought from Australia to Britain in the 19th century, and was initially thought to be a hoax played by taxidermists. It was described by Darwin as a ‘funny sort’ [Mozley Moyal, 2001], and [Burrell, 1927, p. 1] claims that “No animal has given rise to so much controversy among both layman and professed zoologists”. Therefore, of the eighteen

Session	student 1	student 2	student 3
69	mb-2a	mb-2a	mb-2a
70	mb-3a-4a	mb-3a-4a	mb-3a-4a
71	mb-3a-4b	mb-3a-4b	mb-3a-4b
72	mb-3a-4c	mb-3a-4c	mb-3a-4c
73	mb-3a-4d	mb-3a-4d	mb-3a-4d
74	mb-3a-4e	mb-3a-4e	mb-3a-4e
75	mb-3a-4f	mb-3a-4f	mb-3a-4f
76	mb-3a-4g	mb-3a-4g	mb-3a-4g
77	mb-3a-4h	mb-3a-4h	mb-3a-4h
78	mb-3a-4i	mb-3a-4i	mb-3a-4i
79	mb-3a-4j	mb-3a-4j	mb-3a-4j
80	mb-3a-4k	mb-3a-4k	mb-3a-4k
81	mb-5a-6a	mb-5a-6a	mb-5a-6a
82	mb-5a-7a-8a	mb-5a-7a-8a	mb-5a-7a-8a
83	mb-5a-7a-8b	mb-5a-7a-8b	mb-5a-7a-8b
84	mb-5a-7a-8c	mb-5a-7a-8c	mb-5a-7a-8c
85	mb-5a-7a-8d	mb-5a-7a-8d	mb-5a-7a-8d
86	mb-5a-7a-8e	mb-5a-7a-8e	mb-5a-7a-8e
87	mb-5a-7a-8f	mb-5a-7a-8f	mb-5a-7a-8f
88	mb-5a-7a-8g	mb-5a-7a-8g	mb-5a-7a-8g
89	mb-5a-7a-8h	mb-5a-7a-8h	mb-5a-7a-8h
90	mb-5a-7a-8i	mb-5a-7a-8i	mb-5a-7a-8i
91	mb-5a-7a-8j	mb-5a-7a-8j	mb-5a-7a-8j
92	mb-5a-7a-8k	mb-5a-7a-8k	mb-5a-7a-8k

Table 10.7: Experiments to determine when to perform monster-barring, when all three agents are able to perform it

different animals in our animal taxonomy, we considered one to be odd. Recall that although we would normally consider the t-rex and the dragon to be odd, and thus liable to be barred as monsters, the domain file contains no information about whether the animals are now extinct, or are mythical, and so we do not penalise HRL for not identifying these two as potential monsters.

Number theory

We discussed the history of number theory in §6.2, and argued that the numbers 0 and 1 were previously considered to be monsters. Therefore, of the sixty one numbers in our number theory domain, we considered only 0 and 1 to be odd.

We also assume in our evaluation that these entities should be monster-barred, rather than monster-accepted. This does not reflect history, as the platypus was considered to be odd, but

later on it was accepted as a valid animal, and similarly while the numbers 0 and 1 were initially rejected, they were later accepted as valid numbers. We use this categorisation only because the theories we evaluate from HRL are young, in terms of the number of theory formation steps which have been performed, and therefore might be considered to be in the initial conservative stage still. We show how the entities were classified whenever we give our evaluation of the percentage of entities which were correctly classified.

10.4.6 When to propose to bar an object: results and discussion

We show the results from sessions 1 - 24 in table 10.8, where we present which of the expected 'monsters' identified above, were proposed as monsters, and which other entities were similarly proposed. We calculate the percentage of correct classifications, where an entity is correctly classified if it is one which we identified as a monster and it was proposed as a monster (true positives), or if it was not one which we considered to be a monster, and was not proposed as a monster (true negatives). The results show that using the 'Lakatos variable', to propose to bar any entity which is a counterexample to a conjecture currently being discussed, produced optimal results in group theory and animal theory. Similarly, using the monster-barring minimum of up to 30 or 40% produced good results in both domains. Using a higher value for the monster-barring minimum, or any of the culprit breaker variables, proved to be too high a restriction, and while we did not get entities which we considered to be valid being proposed as monsters, nor did we get the entities which we wanted to see. This suggests that our restriction for the culprit breaker, namely that including the entity in a theory causes more than half of the entities in a theory to become counterexamples, is too high.

In number theory, the agency identified neither 0 nor 1 as monsters in any of the sessions 1 - 24, and hence scored 97% according to our scheme. This was disappointing. The reason was that the only student which was set to perform monster-barring was the student with the 'monsters' 0 and 1. Even though it was set to make near-equivalence conjectures which held for at least 60% of its entities, it did not raise any conjectures in the discussion to which either 0 or 1 were counterexamples.

In order to prevent the distribution of the data from influencing the monster-barring method in this way, we performed further experiments in sessions 69 - 92. In these sessions, all of the students were set to perform monster-barring if the right conditions arose. We show the results in table 10.9. This table shows which entities were proposed as monsters by one or more of the students, which were monster-barred by the group and which were monster-accepted by the group, and the degree to which we considered the classification to be correct, shown as a

percentage. Note that in table, 10.9, $*^1$ stands for all groups except the trivial group, and $*^2$ stands for a single group of size eight.

In group theory, the results in sessions 69 - 75, excluding session 73, were good, although not optimal. This was consistent with our results in table 10.8, describing our investigation into when a single agent should propose to bar an object. The students did not identify the semi-group as a monster in any of the sessions. This was because it did not arise as a counterexample to any of the conjectures under discussion. Although surprising, we conjecture that the reason for this was that the discussion focused more on the groupoids in the theory and therefore there was not time to discuss the semi-group. We hypothesise that if we ran the agency with mainly examples of groups, a few examples of semi-groups, and no groupoids, then the semi-groups would be highlighted as monsters. In session 70, we saw quite extreme behaviour, in which two semi-groups, and all of the groups except the trivial group were proposed as monsters. However, only one of the groups was barred by the students.

In animal theory, the platypus was the only animal to be highlighted as a monster, which was very encouraging. This was highlighted in sessions 69, 71 and 74, when the monster-barring threshold was either zero or very low. In the intermediate sessions, it did not arise as a counterexample before the session timed out. As in group theory, the agency tended to produce the behaviour we expected when the monster-barring minimum was set fairly low: between 0 and 10%.

In session 69 in number theory, the numbers 2 and 3 were also highlighted, and subsequently barred, as monsters, which was not behaviour we expected. The best session was 71, in which the monster-barring minimum was 10%, and both 0 and 1 were barred. We also saw a case of monster-accepting, in session 76. Although we evaluate this as a misclassification, it was interesting behaviour as it is analogous to the historical acceptance of the number zero. We also saw a case of monster-accepting, in session 75 in group theory, in which one of the groupoids was accepted as a valid member of the domain.

From these results, we hypothesise that using a monster-barring minimum of 10% for all students in the agency, produces the most interesting behaviour in this method.

10.4.7 How to perform monster-barring: results and discussion

The “monster barring type” variable concerns the two different ways of performing monster-barring: “vaguetovague”, and “vaguetospecific”. The interesting setting is “vaguetospecific”, since we then see suggested definitions for the domain of interest. When the variable is set

Session	monsters	other	% correct
Group theory	GP2; GP3; SG1		
1	✓✓✓	–	100%
2	✓✓–	–	94%
3	✓✓✓	–	100%
4	✓✓✓	–	100%
5	✓✓✓	–	100%
6	–✓✓	–	94%
7	–✓✓	–	94%
8 - 24	–––	–	82%
Animal theory	platypus		
1	✓	–	100%
2	✓	–	100%
3	✓	–	100%
4	✓	–	100%
5	✓	–	100%
6	✓	–	100%
7 - 24	–	–	94%
Number theory	0; 1		
1 - 24	–	–	97%

Table 10.8: Monster-barring results: proposing a monster

to “vaguetovague”, discussion continues with no agreement as to how the domain is defined, although the status of the entity in question has been settled.

In table 10.10 we show the different definitions which were suggested for the domain of interest, in our three different domains, when the monster barring type variable was set to “vaguetospecific”. Definitions 1 - 5 are from group theory, 6 and 7 from animal theory, and 8 - 11 from number theory.

The group theory definitions (1 - 5 in table 10.10) were the most satisfactory, with the proposed definitions starting to resemble the group axioms. For instance, the associativity criterion was introduced to prevent the groupoid GP2 from breaking conjectures. Similarly, the criterion of an inverse for at least one element in a group was introduced to prevent the semi-group SG1 from breaking conjectures.

The number theory definitions (8 - 11 in table 10.10) were more obscure, and concerned the way that the core concepts were calculated. For instance, one student defined the domain of interest as “a is an integer and $a \mid a$” as a way of excluding the number zero. This was because zero did not have any divisors, according to the way that it calculated divisors of a number.

Session	**proposed monsters**	**barred**	**accepted**	% correct
Group theory				
69	GP2; GP3	GP3	-	88%
70	GP2; GP3; $*^1$	GP2; GP3; $*^2$	-	88%
71	GP2; GP3	GP3	-	88%
72	GP2	-	-	82%
73	-	-	-	82%
74	GP3	GP3	-	88%
75	GP3	-	GP3	82%
76 - 92	-	-	-	82%
Animal theory				
69	platypus	platypus	-	100%
70	-	-	-	95%
71	platypus	platypus	-	100%
72	-	-	-	95%
73	-	-	-	95%
74	platypus	-	-	95%
75 - 92	-	-	-	95%
Number theory				
69	0; 1; 2; 3	0; 1; 2; 3	-	97%
70	0	0	-	97%
71	0; 1	0; 1	-	100%
72	0	0	-	98%
73	-	-	-	97%
74	0	0	-	98%
75	0	0	-	98%
76	0	-	0	97%
77 - 92	-	-	-	97%

Table 10.9: Monster-barring results: the decision of the agency about which monsters to bar, and which to accept, when all students are able to propose a monster

In animal theory however, the alternative suggested definitions (6 and 7 in table 10.10) were inappropriate. They were both introduced either to exclude, or to cover the platypus. However, clearly none of them do exclude the platypus. The reason for the lack of success in this domain was the way that concepts are represented in HRL. Data tables for new mathematical entities can be calculated: for instance, given a new algebra, HRL can determine whether it is associative or not. However, data tables for other types of new entities, such as a platypus, cannot be calculated by HRL. Each student considers its data table to be complete, and so if a platypus is not in the data table for “animals which live in habitat b”, then the student would consider making this concept a condition of the domain of interest which would exclude the platypus.

Def	Suggested definition
GT	
1	a is a group and the left hand cancellation law holds for a
2	a is a group and a is associative
3	a is a group and exists b c (b in a and c in a and inv(b)=c)
4	a is a group and exists b (b in a and b=id)
5	a is a group and exists b c d (b in a and c in a and d in a and b*c=d)
AT	
6	a is an animal and exists b (b is a habitat for animals and a lives in b)
7	a is an animal and exists b (b is an integer and a has b legs)
NT	
8	a is an integer and $a \mid a$
9	a is an integer and exists b (b is an integer and $b \mid a$)
10	a is an integer and exists b c (b is an integer and $b \mid a$ and c is an integer and $c \mid a$ and c*b=a and b*c=a)
11	a is an integer and exists b c (b is an integer and c is an integer and c+b=a and b+c=a)

Table 10.10: Alternative suggested definitions for the domain of interest

Clearly, it does not.

Our conclusions in this variable, therefore, are that the monster barring type variable should be set to “vaguetovague” unless it can calculate new rows in its data tables when new entities arise.

10.4.8 How to evaluate a proposal to bar an object: results and discussion

In experiments 49 - 68, we tested to see which variable setting was the most effective when students were asked to evaluate a proposal to bar an object. Entities which were proposed as monsters were: 0, 1, 2, 3, 13, 17, 19, 20, 40, 60 in number theory; SG1, GP2, GP3, and a group of size eight in group theory; and the platypus in the animal taxonomy. We were interested in how the students reacted to these proposals, and we evaluated objects as being correctly classified according to our argument in §10.4.5.

We show the results from our experiments in figures 10.3 and 10.4. The first of these shows the percentage of correct classifications, as an average of the each student’s vote for each proposed monster, for each session 49 - 68. We show each domain in a different colour. For instance, in session 49, students correctly classified 71% of the objects proposed as monsters in number theory, and 100% in group theory and the animal taxonomy. We show the details of this session

in table 10.11, where the number in brackets denoted the percentage of conjectures which the entity breaks in the given student's theory.

Session	monster	S1	S2	S3	% classification
GT					
49	GP3	bar (66)	bar (50)	bar (42)	9/9 = 100
49	GP2	bar (31)	bar (33)	bar (25)	
49	SG1	bar (24)	bar (3)	bar (52)	
AT					
49	platypus	bar (27)	bar (18)	bar (20)	3/3 = 100
NT					
49	0	bar (60)	bar (80)	bar (80)	15/21 = 71
49	1	accept (0)	accept (0)	bar (80)	
49	19	accept (0)	accept (0)	accept (0)	
49	17	bar (12)	accept (0)	accept (0)	
49	13	bar (15)	accept (0)	accept (0)	
49	3	bar (18)	accept (0)	accept (0)	
49	2	bar (26)	accept (0)	accept (0)	

Table 10.11: Monster-barring results for session 49: determining how to react to a proposal to monster-bar entities.

In figure 10.4, we present the average of the percentage of correct classifications across the three domains, and the result for each of the three variables we tested: "use breaks conjecture under discussion" (which was set to true or false); "accept strictest" flag (which was set to true or false), and the monster-barring minimum (set to 30%, 60%, or 90%). In the latter case, we add up the value of the monster-barring minimum of all three students in the agency. For instance, in session 49, whose details are presented in table 10.11, we have taken the average percentage of correctly classified entities over the three domains, as (71 + 100 + 100)/3 = 90%. The variable "use breaks conjecture under discussion" was set to true for each of the three students in the agency in this session; therefore this is represented by the brown cross in the left hand graph, at co-ordinates (3,90).

The results suggest that the most effective settings were to set the "use breaks conjecture under discussion" variable to true, or the "accept strictest" flag to false for most, or all of the students in the agency. The best results for the "monster-barring minimum" flag were when the total amount came to 120 or 150%, i.e., the average percentage for each student was either 40% or 50%. However, the former two flags produced the best results. The percentage of correct classifications decreased as the average value of the students' "monster-barring minimum" flag increased, and was lowest, 0% in session 64, when each of the students had a "monster-barring minimum" flag of 90%. This is unsurprising, since if the percentage of conjectures which a

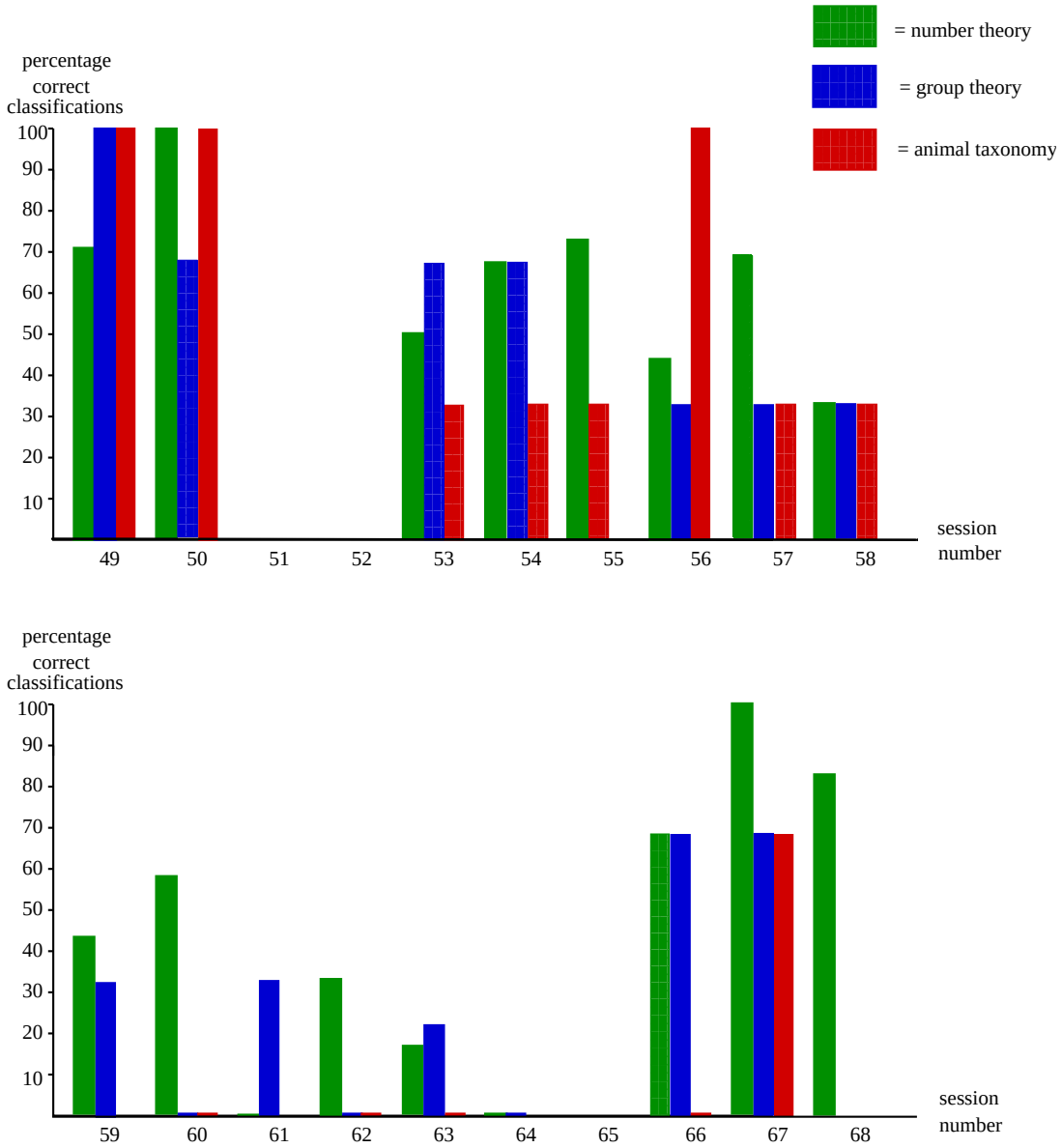


Figure 10.3: The percentage of correct classifications, given a proposed monster, for each domain: sessions 49 - 68

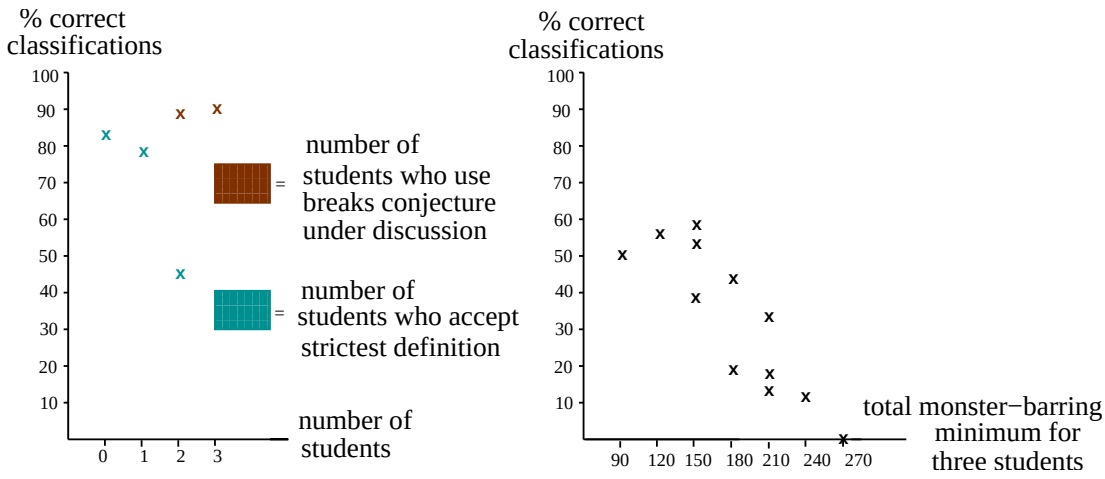


Figure 10.4: The average percentage of correct classifications, given a proposed monster, over the three domains

proposed monster must break in a given student's theory must be very high, before the student agrees to bar it, then many entities which we consider to be monsters will be accepted, as they do not break a sufficiently high proportion of the student's conjectures. We show the results for this variable in number theory, in table 10.12. The numbers given in brackets denote the percentage of conjectures which the entity breaks in the given student's theory. In the case of some sessions, for instance, 51, 52 and 65, no entities were proposed as monsters, and therefore we were unable to evaluate the students' responses.

10.5 It is possible to fine-tune the method of exception-barring

10.5.1 Experimental setup

We tested four variables associated with exception-barring: piecemeal exclusion, piecemeal exclusion with counterexample-barring, strategic withdrawal, and communal piecemeal exclusion. The first three are variables at the student level, whereas communal piecemeal exclusion is at the agency level.

1. **use piecemeal exclusion:** true (a); false (b)
2. **use piecemeal exclusion with counterexample-barring:** true (a); false (b)
3. **use strategic withdrawal:** true (a), false (b)
4. **use communal piecemeal exclusion:** true (a), false (b)

Session	monster	S1	S2	S3	% classification		
53	0	bar (50)	bar (66)	bar (62)	3/6	=	50
53	1	accept (6)	accept (17)	accept (18)			
54	0	bar (42)	bar (57)	accept (57)	2/3	=	67
55	0	bar (50)	bar (66)	accept (66)	11/15	=	73
55	20	accept (0)	accept (0)	accept (0)			
55	40	accept (0)	accept (11)	accept (0)			
55	60	accept (0)	accept (0)	accept (0)			
55	1	accept (5)	accept (16)	accept (17)			
56	0	bar (42)	accept (57)	accept (57)	4/9	=	44
56	1	accept (6)	accept (18)	accept (20)			
56	60	accept (22)	accept (10)	accept (5)			
57	0	bar (50)	bar (66)	accept (57)	8/12	=	67
57	1	accept (6)	accept (17)	accept (12)			
57	3	accept (12)	accept (0)	accept (4)			
57	2	accept (15)	accept (0)	accept (4)			
58	0	bar (42)	accept (57)	accept (57)	1/3	=	33
59	0	accept (42)	bar (57)	accept (57)	4/9	=	44
59	1	accept (6)	accept (18)	accept (20)			
59	60	accept (15)	accept (10)	accept (5)			
60	0	accept (42)	accept (57)	bar (62)	7/12	=	58
60	1	accept (5)	accept (18)	accept (20)			
60	3	accept (21)	accept (0)	accept (0)			
60	2	accept (25)	accept (0)	accept (0)			
61	0	accept (50)	accept (57)	accept (57)	0/3	=	0
62	0	accept (42)	accept (57)	accept (57)	3/9	=	33
62	1	accept (6)	accept (18)	accept (20)			
62	60	accept (15)	accept (10)	accept (5)			
63	0	accept (50)	bar (57)	accept (57)	1/6	=	17
63	1	accept (6)	accept(17)	accept (18)			
64	0	accept (42)	accept (57)	accept (57)	0/3	=	0

Table 10.12: Monster-barring results: determining how to react to a proposal to monster-bar entities in number theory

In session 1, we tested the communal piecemeal exclusion flag in each of our three domains. Since this is a variable at the agency, rather than student level, we did not test different combinations. This meant that fewer experiments were performed on this method than on other methods.

In sessions 2 - 9, we tested all combinations of the piecemeal exclusion and piecemeal exclusion with counterexample-barring methods, for three students. This was to investigate how different combinations affected the number and quality of the modifications made. In the

piecemeal exclusion and counterexample methods, the teacher requested non-existence conjectures. In tables 10.13, 10.14 and 10.15, we present the input variables for testing the piecemeal exclusion and counterexample-barring variables; combinations of piecemeal exclusion with counterexample-barring; and combinations of strategic withdrawal, respectively.

Session	**student 1**	**student 2**	**student 3**
2	pm	pm	pm
3	pm	pm	pm+cb
4	pm	pm+cb	pm
5	pm	pm+cb	pm+cb
6	pm+cb	pm	pm
7	pm+cb	pm	pm+cb
8	pm+cb	pm+cb	pm
9	pm+cb	pm+cb	pm+cb

Table 10.13: Testing the piecemeal exclusion and counterexample-barring variables

Session	**student 1**	**student 2**	**student 3**
10	pm+cb	pm+cb	pm+cb
11	pm+cb	pm+cb	surr
12	pm+cb	surr	pm+cb
13	pm+cb	surr	surr
14	surr	pm+cb	pm+cb
15	surr	pm+cb	surr
16	surr	surr	pm+cb

Table 10.14: Testing combinations of piecemeal exclusion with counterexample-barring and surrender

Session	**student 1**	**student 2**	**student 3**
17	sw	sw	sw
18	sw	sw	surr
19	sw	surr	sw
20	sw	surr	surr
21	surr	sw	sw
22	surr	sw	surr
23	surr	surr	sw

Table 10.15: Testing combinations of strategic withdrawal and surrender

10.5.2 Evaluation criteria

We evaluate the interestingness of the original conjecture and of any modifications produced using exception-barring methods, in terms of the interestingness measures described in 3.1.3 and 4.4. To recap:

• The *applicability* of a conjecture is the proportion of entities that the conjecture discusses. The applicability of an equivalence conjecture is the applicability of the concepts which are hypothesised to be equivalent. The applicability of an implication conjecture is the applicability of the antecedent concept). The applicability of a non-exists conjecture is zero.

• The *comprehensibility* of a conjecture is the reciprocal of the number of distinct concepts which appear in the construction path of the concepts discussed in the conjecture. Comprehensibility is not defined for non-existence conjectures.

• The *surprisingness* of an equivalence or an implication conjecture is the number of concepts which appear in the construction path of one concept, but not both. The surprisingness of a non-existence conjecture is measured by the applicability of its parent concept.

• The *plausibility* of a conjecture is (the number of entities the conjecture discusses - the number of counterexamples to the conjecture)/the number of entities the conjecture discusses.

10.5.3 Communal piecemeal exclusion results

Domain	**counterexample**	**concept found**
number	0	no
group	-	-
animals	platypus	a is an animal $\wedge$ a produces milk $\wedge$ a produces eggs

Table 10.16: Communal piecemeal exclusion results

The results in communal piecemeal exclusion are shown in table 10.16. While only three sessions were performed, the results suggest that either the independent work phase should be increased when using this method, in order to give students more time to form concepts, or that the communal aspect of the method does not add to its effectiveness. In the only case where an appropriate concept was found, it was a concept which the students who had the counterexample in their theories had already formed, and so they could have performed the modification independently. This would save time because it would decrease the amount of communication necessary for a single modification. We implemented piecemeal exclusion in

its communal form, with the idea that those students with counterexamples to a conjecture may not be the same as those with an appropriate concept which could be used in the modification, and thus collaboration would be useful. However, the results do not support this idea and more work would need to be done on this method for it to become useful.

10.5.4 Piecemeal exclusion and counterexample-barring results

We show our piecemeal exclusion and counterexample-barring results in terms of the interestingness of the original conjecture and of the modification, in table 10.17. We are only interested in the surprisingness and plausibility of the conjectures found using piecemeal exclusion, since the applicability of a non-existence conjecture will always be zero, and comprehensibility is not defined for non-existence conjectures. Recall that the surprisingness of a non-existence conjecture is measured by the applicability of its parent concept. This is based on the idea that it is more surprising if the parent concept has many examples than if the parent concept has only a few examples. For instance, conjecture 1 in number theory, shown in table 10.17 below, states that "$\nexists a$ such that (a is an integer $\wedge$ a $|$ a $\wedge$ a * a = a)". This has surprisingness 1, since its parent concept, the core concept given for multiplication, has applicability of 1 (since it applies to every entity in the theory). Using this measure, the surprisingness of a modified non-existence conjecture is the number of counterexamples to the original conjecture (which provides the parent concept) divided by the number of entities in a student's theory, i.e., the applicability of the concept in the original non-existence conjecture. The first conjecture in table 10.17 was modified by student 1, working with entities 0 - 20. The student found counterexample 1 to the conjecture, and modified it to become $\nexists a$ such that (a is an integer $\wedge$ a $|$ a $\wedge$ a * a = a $\wedge$ $\neg$(a = 1)). We measure the modification as having overall surprisingness 1/61, since the parent concept has 1 example, and there are 61 entities altogether in the theory.

In table 10.17 below, we refer to the overall surprisingness and plausibility of the original and the modified conjecture. In the penultimate column, we also give the average interestingness of the modified conjecture, calculated as (surprisingness + plausibility)/2. For instance, modification 1 in number theory has surprisingness 1/61 and plausibility 1, and so average interestingness of (1/61 + 1)/2 = 0.508. We give this figure to three decimal places. The final column shows whether piecemeal exclusion (pm) or counterexample-barring (cb) was used.

In figure 10.5, we also represent these results (to two decimal places), showing the number of modifications for each session, and the average interestingness of the modification. The latter is calculated by the interestingness given in table 10.17 (which is an average of the surprisingness and plausibility of each modification). For instance, session 6 achieved three modifications in

Modn	**surp**			**plaus**			**int**	**method**
	orig	→	modn	orig	→	modn		
NT								
1	1	→	1/61	60/61	→	1	0.508	cb
2	1	→	1/61	60/61	→	1/61	0.016	pm
3	1	→	1/61	60/61	→	1	0.508	cb
GT								
1	0.5	→	1/17	16/16	→	1	0.571	cb
2	1	→	1	0/17	→	15/17	0.941	pm
3	0.5	→	0/17	17/17	→	0/17	0	pm
4	1	→	3/17	14/17	→	3/17	0.176	cb
5	0.6	→	14/17	3/17	→	14/17	0.824	cb
6	1	→	3/17	14/17	→	14/17	0.5	cb
AT								
1	0.5	→	1/18	17/18	→	1	0.528	cb

Table 10.17: The interestingness of the original and modified conjectures, where the modification was performed using piecemeal exclusion and counterexample-barring

number theory (shown as 1-3) and therefore had an average interestingness of 0.508 + 0.016 + 0.508 = 0.344.

10.5.5 Discussion

The most noticeable aspect of figure 10.5 is that, of the three domains we tested, piecemeal exclusion works best in group theory. This is a particularly encouraging result since it indicates that HRL is not fine-tuned to only work in number theory. Another aspect to notice is that giving some, but not all agents counterexample-barring seems to improve the modifications. However, in general, while we had some interesting modifications, the number of modifications was too low to be able to draw conclusions about optimal parameter settings.

In number theory, conjectures 1 and 3 were similar, though not identical, to conjectures found in the illustrative sessions described in sections 6.6.1 and 6.6.3, in which the students were set to perform monster-barring, and they barred the numbers one and zero. When exception-barring methods are performed instead, conjecture 1: $\nexists a$ such that (a is an integer $\wedge$ a $|$ a $\wedge$ a * a = a) was modified to:

• $\nexists a$ such that (a is an integer $\wedge$ a $|$ a $\wedge$ a * a = a $\wedge$ $\neg$(a = 1))

Conjecture 3: $\nexists a$ such that (a is an integer $\wedge$ a + a = a) was modified to:

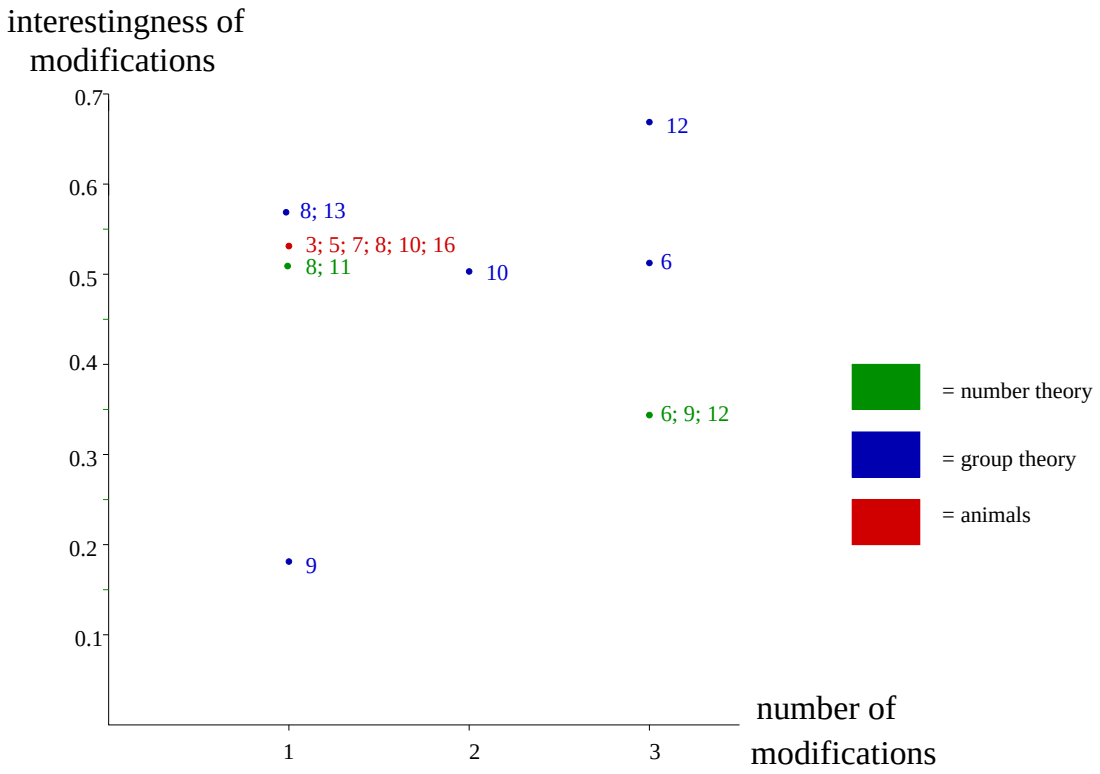


Figure 10.5: The exception-barring results for sessions 2 - 16. In those sessions which do not appear, no modifications were performed. Session numbers appear in the figure.

• $\nexists a$ such that (a is an integer $\wedge$ a + a = a $\wedge$ $\neg$(a = 0))

In group theory, we saw sequences of modifications which were sometimes poorly understood by one of the agents. For instance, the first conjecture in group theory in table 10.17 states that there does not exist a group in which every element is the identity. This was modified, in session 6, to the theorem that the only group in which every element is the identity, is the trivial group, phrased as:

• $\nexists a$ such that (a is a group $\wedge$ ((b $\in$ a) $\rightarrow$ (b $\in$ a $\wedge$ b = id)) $\wedge$ $\neg$(a = 0))

When this became the conjecture under discussion, the student without the trivial group could not reconstruct the concept of the trivial group, and instead "modified" the conjecture to saying that there are no groups (conjecture 3 in table 10.17). This in turn was modified to saying that there are no groups which are not associative (conjecture 2 in table 10.17). While the discussion did result in some interesting conjectures, these results suggest that we need to enable agents in HRL to question when they see a concept which they cannot reconstruct (since they lack either a core concept or the parameters necessary). This could be done in a similar way to that in which a student comes across an entity which is new to it. A further improvement would be

to measure the number of production rule applications that it takes to modify a conjecture, as multiple applications resulted in overly specialised conjectures.

There was only one example of a modified conjecture in the animal domain: $\nexists a$ such that (a is an animal $\wedge$ a produces milk $\wedge$ a produces eggs). This was modified to:

• $\nexists a$ such that (a is an animal $\wedge$ a produces milk $\wedge$ a produces eggs $\wedge \neg$(a = platypus))

This was an alternative reaction to the monster-barring technique of barring the platypus. This was also a very encouraging result. It was disappointing, however, that there were no other modifications. The reason for this was a lack of false conjectures: there was only one other conjecture suggested for which agents had counterexamples in their data sets. This led on directly from the above modification, and stated that there is no such animal as the platypus. The two agents with the platypus in their data sets attempted to modify this, but were unable to find a modification.

10.5.6 Strategic withdrawal results

In 2 of the 7 sessions, namely, 17 and 19, students successfully performed strategic withdrawal only in the group theory domain. In both sessions, the method was performed twice. In the first case, the flawed conjecture was that:

$$\text{a is a group} \leftrightarrow \text{a is associative.}$$

Student 2 found counterexamples GP2 and GP3, and only one positive example, GP1, the trivial group. The student looked for a concept to cover the trivial group, and found “a is a group $\wedge$ the left hand cancellation law holds for a”. It then modified the original conjecture in three ways:

• for all a: a is a group $\wedge$ the left hand cancellation law holds for a $\leftrightarrow$ a is a group $\wedge$ a is associative,

• for all a: a is a group $\wedge$ the left hand cancellation law holds for a $\rightarrow$ a is a group $\wedge$ a is associative,

• for all a: a is a group $\wedge$ the left hand cancellation law holds for a $\rightarrow$ a is a group.

The second flawed conjecture was “a is a group $\leftrightarrow$ a is a group $\wedge$ the left hand cancellation law holds for a”. This was also modified by student 2 in three ways:

• for all a: a is a group $\wedge$ a is associative $\leftrightarrow$ a is a group $\wedge$ the left hand cancellation law holds for a,

• for all a: a is a group $\wedge$ a is associative $\rightarrow$ a is a group $\wedge$ the left hand cancellation law holds for a,

• for all a: a is a group $\wedge$ a is associative $\rightarrow$ a is a group.

Considering the agency as a whole, and measuring the combined plausibility, all six modifications improved upon their original conjecture. However, the combined applicability decreased in all cases. This was unsurprising, as the method of strategic withdrawal narrows the domain of application. (Note that in measuring the applicability of near-equivalence conjectures, we use that of the concept with highest applicability.) The surprisingness and comprehensibility measures stayed constant in all cases. We show the values for these four measures of interestingness, in table 10.18.

int	**conj1**	**modn1**	**modn2**	**modn3**	**conj2**	**modn1**	**modn2**	**modn3**
plaus	15/17	16/17	1	1	14/17	16/17	16/17	1
appl	1	15/17	14/17	14/17	1	15/17	15/17	15/17
surp	2	2	2	2	2	2	2	2
comp	1/2	1/2	1/2	1/2	1/2	1/2	1/2	1/2

Table 10.18: Showing the effects of strategic withdrawal on the interestingness measures

10.5.7 Discussion

One reason for the failure to perform strategic withdrawal more often was that we have not implemented this method for non-existence conjectures, which was the form of some of the conjectures under discussion. Another reason was that students were often unable to find a concept which covered all of their positive entities. We might solve this problem by increasing the number of steps in the independent work phase, so that more concepts are available to the students. Another solution would be to allow the students to use a concept which covers a proper subset of the positives. This would result in weaker conjectures, but may be preferable to not being able to perform the method at all. Although there were not enough instances of strategic withdrawal being performed to be able to test our hypothesis, all of the modifications were interesting conjectures, and the method was also shown to work well in group theory.

10.6 Conclusions

In this chapter we have presented and discussed results from evaluating our three hypotheses: that it is possible to fine-tune the method of surrender; it is possible to fine-tune the method of monster-barring, and it is possible to fine-tune the method of exception-barring. This has taken the form of an investigation of which variables and parameter settings produced certain desired behaviour, as well as showing that by altering the parameters we alter the behaviour of the system.

We have presented results from experiments in which we test different parameter settings. These indicate which settings produce the desired behaviour and which do not. We have run all of our experiments in three domains: group theory, animal theory and number theory. Although we developed HRL in number theory and therefore might expect it to perform better in this domain than in others, it does not. This shows that our system is not fine-tuned. We have also shown results from experiments in which we focused on different types of conjecture: namely non-existence conjectures in piecemeal exclusion.

Although we have performed only preliminary tests, we have seen that the methods work in different domains as well as on different types of conjectures. We have also identified which variable settings lead to desirable conjectures and concepts. Our experimentation has also highlighted areas of future experimentation which may prove interesting, as well as aspects of the system which could be improved.

Chapter 11

Philosophical evaluation

Our project is to *(a)* give a computational reading of Lakatos's theory of mathematical discovery and justification, and *(b)* to show that it has been useful to do so. In this chapter we continue the argument which we began in the previous chapter, that it has been useful to do so, where 'useful' is defined by a set of criteria suggested by Thagard, Sloman, Popper and Lakatos, and described in §3.3.1. We adopt a shotgun approach with respect to these criteria: a theory is not either good or bad, but good with respect to certain criteria. We are evaluating two aspects of our project. Firstly, our computational approach suggests new ways in which to evaluate Lakatos's theory. Secondly, this approach has suggested ways in which to clarify and extend Lakatos's theory, hence embedding it within a broader theory.

This chapter is organised as follows: in §11.1 we recap criteria of a good theory, suggested by Thagard [1993], Sloman [1978] and Popper [1972], and described in §3.3.1. From these, we identify the two main criteria of consilience and simplicity which we use to evaluate our project in this chapter. In §11.2 we describe these two criteria in terms of sets, where we are interested in the set of all mathematical conjectures, the set of all mathematical conjectures which a theory accounts for, and the set of all mathematical conjectures which one attempts to account for with the theory (the inspiring set). In sections 11.3, 11.4, 11.5 and 11.6 we evaluate the methods of surrender, monster-barring, exception-barring and lemma-incorporation with reference to elements in the sets, and the two main criteria. Where relevant, we discuss sessions from experiments in the previous chapter. In §11.7 we argue that our approach has satisfied five further criteria of a good theory, suggested by Sloman: that a theory explain a range of possibilities, and a good theory be: definite, rigorous, economical, and extendable. In §11.8, we discuss our project with reference to Feferman's criticisms of Lakatos's theory, and we argue that the computational approach has suggested answers to some of these. Finally, in

§11.9 we discuss our project and Lakatos's project with reference to Lakatos's own criteria for evaluating scientific research programmes. We summarise the chapter in §11.10.

11.1 Recap of criteria of a good theory

We consider criteria suggested by Thagard [1993], Sloman [1978] and Popper [1972], described in §3.3.1. These consist of Thagard's notions of consilience, simplicity and analogy (§ 3.3.2); Sloman's criteria that a theory *(i)* explain a range of possibilities, and a good theory is *(ii)* definite, *(iii)* general, *(iv)* able to explain fine structure, *(v)* non-circular, *(vi)* rigorous, *(vii)* plausible, *(viii)* economical, *(ix)* rich in heuristic power, and *(x)* extendable (§3.3.3); and Popper's criteria of being independently testable, and rich in content (§3.3.4). These three sets of criteria overlap in the criteria shown below. A good theory should:

- be as general as possible (Thagard's notion of consilience, and Sloman's third generality criterion);
- explain more than it set out to explain (Thagard's notion of simplicity, and Popper's richness in content criterion);
- not assume what it sets out to explain (Sloman's fifth non-circularity criterion, and Popper's independently testable criterion).

We focus on the first and second of these criteria. We give further arguments that we have satisfied Sloman's first, second, sixth, eighth and tenth criterion. We do not consider Thagard's notion of analogy, Sloman's fourth, seventh and ninth criteria or Popper's independently testable criterion. This is because they are either not relevant, in the case of Thagard's notion of analogy, or it is difficult to show that our project has satisfied them, in the case of the remaining criteria. Since the criteria are considered to be desirable rather than necessary characteristics of a good theory, showing that our theory satisfies some of them is sufficient to support our argument that it is a good theory.

11.2 The criteria in terms of sets

Ritchie [2001] suggested that we evaluate the degree of creativity in a computer program by considering four sets within a universe of basic items (all possible instances of an intended class of artefacts). These consist of the set of inspiring examples used to guide the construction of

the creative program; the set of items generated by the program in a session; and two fuzzy sets: Typicality and Value, which respectively measure the novelty and quality of an item. Ritchie suggests a catalogue of fourteen formally expressed criteria, based on the relative proportions of the cardinality of the four sets. This methodology is continued in [Colton et al., 2001], in which the set of inspiring examples and the set of artefacts of high quality are among those considered to be relevant to an evaluation of creativity. Colton et al. [2001] also relate their work to the criteria outlined by Popper. We adopt this methodology of seeing programs and output as sets, and considering the content of relevant subsets. We therefore view the first and second of the overlapping criteria above, in terms of sets.

There are three relevant sets. Firstly, there is the set of all mathematical conjectures. We assume that this set is fixed, containing future as well as historical mathematical conjectures, and conjectures which may never be considered. This is the universal set, U[1]. Secondly, there is the set of all mathematical conjectures which a theory accounts for, E. Finally, we have the set of all mathematical conjectures which one attempts to account for with the theory. This is the inspiring set, I. We are interested in making the set E large with respect to U, and with varied and interesting members. This is the first criterion above. We are also interested in the difference between sets E and I, i.e., the intersection of E and the complement of I: $E \cap I'$, or $E \backslash I$. This is the second criterion above. In terms of our evaluation of Lakatos's theory, if I is Lakatos's inspiring set, containing Euler's conjecture and associated concepts, conjectures and proofs, i.e., the case study which Lakatos documented, then E contains further concepts, conjectures and proofs which can be explained by Lakatos's theory. In this chapter, we identify members of E. Furthermore, if the inspiring set for our extended theory is $I1$, then $E1$ contains all of the concepts, conjectures and proofs which can be explained by our theory. Figure 11.1 shows a Venn diagram which is a simplified representation of these five sets, assuming a subset relation between the sets. Our evaluation mainly consists of identifying elements in $E \backslash I$ and $I1 \backslash E$. Elements in $E \backslash I$ show both that Lakatos's theory is general (criterion 1), and that it explains more than the data it set out to explain, i.e., that it applies to more than the evolution of Euler's conjecture (criterion 2). Elements in $I1 \backslash E$ are the inspiring examples for our extended theory, and help us to see the gaps in Lakatos's theory. Elements in $E1 \backslash I1$ allow us to make the argument that our extended theory is general and that it explains more than the data it set out to explain.

In general, at the start of chapters 5 to 9, we have outlined Lakatos's theory and his inspiring examples from [Lakatos, 1976] in I (we assume that examples which are not in [Lakatos, 1976] are not in I). We have then suggested further inspiring examples – which are generally

[1] We would have liked to include concepts and other mathematical phenomena such as objects, theoretical statements, and proofs in this set. However, since this set would then itself be a concept of mathematical interest, we stay on the safe side of Russell and Zermelo's madness by limiting it to conjectures.

applications of the method being discussed – to number theory. These examples either fall into E, if they are strong applications, or $I1$, if they are extensions of Lakatos's theory. We have presented details of our implementation, and illustrative sessions, the results of which either fall into E, for a faithful implementation of Lakatos's theory, $I1$, if we were specifically replicating a theoretical example, or $E1 \backslash I1$ for further results. In chapter 10 we tested hypotheses in HRL by running it in various domains with various parameter settings, and produced further results in $E \backslash I$ and $E1 \backslash I1$.

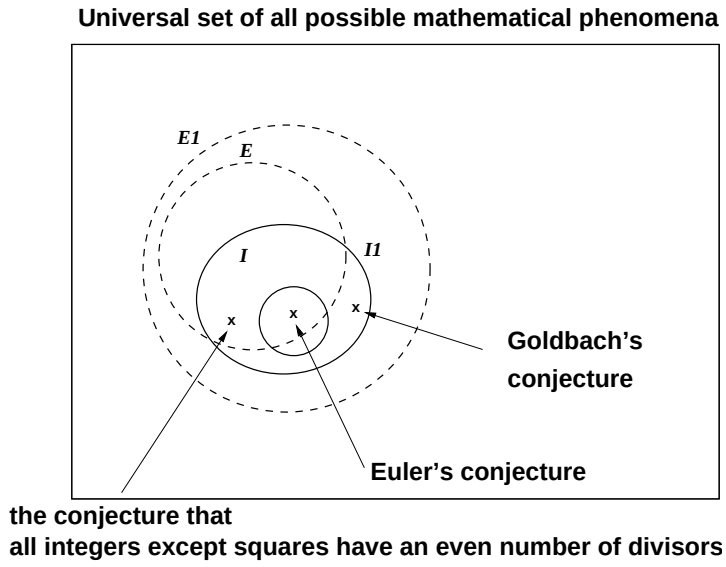


Figure 11.1: A simplified Venn diagram representation of Lakatos's inspiring set (I), the mathematical conjectures which his theory explains (E), our inspiring set ($I1$), and the mathematical conjectures which our theory explains ($E1$)

11.3 Evaluating the method of surrender

The method of surrender consists of using a counterexample to refute a conjecture. Lakatos's inspiring example in I was the rejection of Euler's conjecture, once the counterexample of the hollow cube had arisen.

We identified an application of this method in number theory, discussed in Burton [1985]. This is the conjecture that the *nth* perfect number P_n contains exactly n digits. This conjecture was rejected when the fifth perfect number, 33,550,336, which is a counterexample, was found. This example is in $E \backslash I$, i.e., it is explained by Lakatos's theory but was not one of the inspiring examples.

We show another application of this method to the domain of groupoids, which was a result

from our experiments in the previous chapter. Below, we give an extract from session 1 in group theory (table 10.3 on page 164), in which all three students in HRL were set to perform surrender whenever they had a counterexample to a conjecture. The students were discussing groupoids. Student 1, who had only examples of groups, conjectured that all groupoids have an identity. Student 3, who had examples of non-groups, found a counterexample of a groupoid which did not have an identity (the groupoid GP3 in table 10.2 on page 161), and rejected the conjecture. Note that we have implemented some natural language capabilities in HRL, in order to make the output easier to follow. There are some grammatical and punctuational errors in the output, which we include in the extracts.

> **Teacher:** Has anyone got any interesting conjectures to get the discussion started?
>
> **Student 1:** for all a : a is a groupoid $\leftrightarrow$ a is a groupoid $\wedge$ exists b (b in a $\wedge$ b=id)
>
> **Teacher:** Has anyone got any conjectures which modify the conjecture ' for all a : a is a groupoid $\leftrightarrow$ a is a groupoid $\wedge$ exists b (b in a $\wedge$ b=id)'? We want to perform individual methods
>
> **Student 3:** for all a : a is a groupoid $\nleftrightarrow$ a is a groupoid $\wedge$ exists b (b in a $\wedge$ b=id) is not worth modifying

This example is also in $E\backslash I$.

We extended Lakatos's theory by considering further when someone may decide to reject a conjecture, given a counterexample. We implemented the situations where a conjecture:

- has already been modified a certain number of times;
- is less interesting than a certain threshold;
- is less interesting than the average interestingness of someone's conjectures;
- is considered to be implausible, and
- has a domain of application which is below a user-set threshold.

In session 12, the students are set to consider the domain of application before performing surrender and rejecting a conjecture. The first student sent the following conjecture for discussion:

> **Student 1:** for all a : a is a group $\leftrightarrow$ a is a group $\wedge$ a is associative

The third student reconstructs this as a near-equivalence conjecture, since it has a counterexample. However, the student does not reject the conjecture, because the conjecture has a high domain of application. Therefore, the third student instead attempts to modify the conjecture. This example is in $E1\backslash I1$, and shows how someone working with groupoids may start to consider groups to be a more interesting domain.

11.4 Evaluating the method of monster-barring

The method of monster-barring consists of defending a conjecture from a counterexample by arguing that the counterexample is not valid. In [Lakatos, 1976], a series of definitions of polyhedron are suggested and negotiated. Students who want to defend a conjecture argue for definitions which *exclude* a proposed counterexample, or monster. Students who want to attack the conjecture argue for definitions which *include* a given counterexample, i.e., which would mean that the conjecture is false. The teacher resolves each such discussion by asking the class to accept the strictest definition, i.e., that which excludes the monster, leaving the conjecture open. The first inspiring example in *I* from [Lakatos, 1976] is the hollow cube, which is barred as a monster. Two rival definitions of polyhedra are raised: "a solid whose surface consists of polygonal faces", and "a surface consisting of a system of polygons". The teacher suggests that the class accept the second, stricter definition.

11.4.1 Ambiguity in theory formation

In order to implement monster-barring, we have had to model ambiguity. This forces us to answer questions about what can be ambiguous. In mathematical theories, at least two types of component may be ambiguous: objects of interest and concepts. An example from *I*1 is that it may be ambiguous whether the object of interest $\aleph_0$ (the size of the set of all integers, i.e., the first transfinite number) is really a number or not. Another example is that it may be ambiguous whether the definition of the concept of prime number is 'any number with exactly two divisors', or 'a number which is only divisible by itself and one' (the difference being whether we consider the number 1 to be prime or not).

In our implementation, objects of interest and core concepts can be ambiguous. Students can question whether the definition of a core concept includes a specific object or not, for instance whether the concept of number should include zero, or whether the concept of animal should include the platypus.

11.4.2 When to perform monster-barring

Implicit in the method of monster-barring is the fact that each party has a reason for wanting to define a concept in a certain way. This is a common phenomenon in everyday reasoning; for instance, politicians will define 'unemployment' or 'violent crime' differently, depending on whether they wish to argue that the figures have risen or fallen. In [Lakatos, 1976] the students

discuss what was their original, unexpressed, intended definition and whether it corresponds to the explicit definition they are currently defending. Very little time is spent on why an alternative definition is first proposed, or why it may eventually be accepted or rejected; in [Lakatos, 1976] the teacher always instructs the class to accept the monster-barring definition, i.e., that which excludes the monster. This is a gap in the theory; always accepting the strictest definition is an unrealistic way of settling the dispute. As Corfield says; "Mathematicians have an intuitive feeling of the behaviour of objects they try to define – the process of discovery involves the struggle to find a good or 'right' definition" [Corfield, 1997, p.113].

We have addressed this gap in our algorithm for the way in which students decide: *(i)* whether they want propose an alternative definition or not, and *(ii)* given a proposed new definition, whether they want to accept it or not. There are a range of motives which mathematicians have for rejecting or accepting a concept definition. In the context which Lakatos presents, the most obvious motivation is to defend or attack a given conjecture (i.e., Euler's conjecture). Another factor is the effect that choosing one of two competing definitions will have on the rest of a mathematician's theories or beliefs (this is known as the degree of *epistemic entrenchment* in belief revision [Gärdenfors, 1992]). In our implementation, we have extended Lakatos's theory to reflect this.

Suppose that a student is sent an object of interest which is a counterexample to a conjecture that the group is currently discussing, and the object is new to the student. If this object is presented as an example of a concept C which the student is familiar with, then it is clear that there is some ambiguity over the definition of C. For instance, suppose a student receives the 'number' 0 when it has only previously seen positive examples of number (1, 2, etc..). In this case, the concept 'number' is ambiguous. The student then has two ways to decide whether it wants to bar the object (where the user decides which of the two ways should be used, at the start of the session). The first way is to test whether the counterexample breaks more than a (user-defined) percentage of all the conjectures in the student's theory, and if so, it proposes to monster-bar the counterexample. The second way is to test to see whether the new object is a 'culprit breaker'. This means that not only is the object a counterexample to the conjecture under discussion, but if it is allowed into the student's theory, then it forces other objects in the theory which previously supported the conjecture, to be involved in a counterexample to the conjecture. For instance, suppose that the conjecture under discussion is the law of monotonicity itself, that for all numbers a, b and c, if $b < c$, then $a+b < a+c$. A mathematician who does not consider $\aleph_0$ to be a number may hold monotonicity to be true, and be opposed to accepting $\aleph_0$ as a number since not only is it a counterexample to the law of monotonicity – if $a = \aleph_0$ (for any finite b and c) – but also because it forces the numbers b and c into being part of a counterexample (the triple $[a,b,c]$), where previously they were supporting examples.

This example is in $I1$. Similarly, suppose that the conjecture under discussion is the conjecture that there do not exist integers a, b, c such that $a+b=c$ and $c|a$. A mathematician who does not consider 0 to be a number may make this conjecture. However, if the number 0 is proposed as a counterexample, then the mathematician will find that if 0 is to be accepted as a number, there is not just a single counterexample, 0, to this conjecture, but that the existence of 0 has forced all of the other objects to become counterexamples as well. For instance, if we take $b=0$ and $a=c$, then 1 is a counterexample since $1+0=1$ and $1|1$, and similarly 2 is also a counterexample since $2+0=2$ and $2|2$, etc. Therefore, allowing 0 into a theory containing the integers $1-10$ means that there are now 11 sets of counterexamples a, b and c. This example is also in $I1$. These examples inspired our implementation of the 'culprit breaker': an entity is monster-barred if it forces other entities into being counterexamples also. In our algorithm, if the object in question forces other objects into being counterexamples for a higher number of conjectures than a minimum, user defined, proportion, then the object is called a 'culprit breaker', and monster-barred.

Rather than the teacher instructing the students to use the narrowest definition, once a concept has been raised as being ambiguous and two definitions suggested, each student decides which definition they prefer and votes accordingly. The definition is then decided democratically, based on these votes. If the votes are equal, then we follow Lakatos's principle of taking the narrowest definition. The students make the decision based on the proportion of conjectures in their theories which still hold under each of the rival definitions. Clearly this way of determining a definition means that we have to be able to accept the 'monster'. In our algorithm, if the consensus between the students is to extend a definition, then the teacher asks them all to perform monster-*accepting* by agreeing on the new, wider definition. [Lakatos, 1976, p. 83 - 99] does raise this issue, calling it concept stretching. However, the discussion in this part of Lakatos's book principally concerns the semantics and methodology of monster-barring rather than reasons for initially proposing and accepting a rival definition.

For example, in session 1 in table 10.4 on page 170, the first student is set to perform monster-barring whenever another student suggests an entity which is a counterexample to a conjecture which the first student believes in. The following is an extract from the output of this session.

> **Teacher:** Has anyone got any interesting conjectures to get the discussion started?
>
> **Student 1:** for all a : a is an animal $\wedge$ a produces eggs $\leftrightarrow$ a is an animal $\wedge$ $\neg$(a produces milk)
>
> **Teacher:** Has anyone got any entities which break the conjecture ' for all a : a is an animal $\wedge$ a produces eggs $\leftrightarrow$ a is an animal $\wedge$ $\neg$(a produces milk)'? We want to perform communal piecemeal exclusion
>
> **Student 1:** No, sorry.

Student 2: platypus
Student 3: platypus
Student 1: platypus is not a animal.

By contrast, in session 7 in table 10.4 on page 170, the first student is set to perform monster-barring whenever another student suggests an entity which is also a counterexample to more than 50% of its conjectures. When the platypus is given as a counterexample in the situation above, the first student looks to see how many of its conjectures this entity breaks, and finds that it breaks 48%. Therefore the student does not propose to bar it, and instead accepts it and adds it to its theory. In the above dialogue, rather than the last message from student 1, that a "platypus is not a animal", the next message is:

Teacher: Has anyone got any concepts which cover platypus; ? We want to perform communal piecemeal exclusion

We see an example of how to react to a proposal to monster-bar an object in session 66 in table 10.4, the animal taxonomy domain. In this session, the first two students are set to accept the strictest definition. Given the proposal from student 1 above to bar the platypus, all three students vote to bar the entity, with the outcome being that they all downgrade it to a pseudo-entity.

11.4.3 Proposing a new definition

In [Lakatos, 1976], there are two types of concept definitions: an initial vague concept, which is not explicitly defined but for which some positive examples are known; and an explicit definition, for which the extension of the concept should be easier to determine. The two ways in which HR [Colton, 2002] can represent concepts bears some analogy to these: a core concept has no explicit definition, and a concept which HR has generated does.

The process of monster-barring in [Lakatos, 1976] might start with a vague definition and become more specific, or start with a specifically defined concept and by discussion reach agreement to define it in a different, yet still specific way. We have extended this by implementing the case which starts and finishes with a vague concept, i.e., a specific definition is not reached, but agreement is reached about whether a concept includes a given object or not. This is useful as explicit and precise definitions cannot always be reached nor agreed upon: even supposed specific definitions contain vague terms. For instance, defining a polyhedron as 'a solid whose surface consists of polygonal faces' [Lakatos, 1976, p.14] *is* being more explicit about what

is meant by the concept of a polyhedron, but there is still ambiguity in the sub-concepts solid, surface, polygon and face. We have implemented the process of generating a specific definition from a vague concept *C*, by finding all of the concepts which are conjectured to be equivalent to *C*, and then selecting the most interesting of these.

In session 66 in table 10.6 (on page 172), working in group theory, the first and second students are set to accept the strictest definition if controversy arises. The third student (working with groupoids) sends the groupoid GP2 in table 10.2 (on page 161) as a counterexample to a conjecture, and the second student sends the reply that GP2 is not a counterexample since the domain of interest concerns algebras which are associative.

All students then evaluate the proposal to monster-bar this object. The first two vote to bar it and accept the strictest definition, where associativity is required. Having counted the votes, the teacher asks everyone to downgrade GP2 to a pseudo-entity.

This is an example of how mathematicians might begin to develop group theory, although having started working with the more general theory of groupoids. It shows the situation in which a concept, which was previously conjectured to be equivalent to another concept, becomes a part of the definition of the latter concept. This example is in $E1\backslash I1$.

11.5 Evaluating the method of exception-barring

Exception-barring consists of generalising from a set of counterexamples and then excluding entities with this property from the domain of application of the conjecture, or generalising from a set of supporting examples and then limiting the domain of application of the conjecture to entities with this property. Lakatos's first inspiring example of the former was generalising from the hollow cube to polyhedra with cavities, and then excluding polyhedra of this type from Euler's conjecture. Lakatos's first inspiring example of the latter was generalising from cubes, octahedra, pyramids and prisms to convex polyhedra and then limiting the domain of application of Euler's conjecture to convex polyhedra. These examples are all in *I*.

We considered the application of the exception-barring methods to number theory. For instance, we considered Goldbach's conjecture that 'all even numbers except 2 are the sum of two primes', and the conjecture that 'all primes except 2 are odd'. These examples, in $I1$, led us to distinguish between counterexample and piecemeal exclusion. Introducing this distinction raised the question of *when* we should use concept-barring and when we should use counterexample-barring, which we explored in the previous chapter. We also considered extending the method to form equivalence conjectures. For instance, in §7.6.1 we described our

application of piecemeal exclusion to the conjecture that "all integers have an even number of divisors". Using piecemeal exclusion, HRL modified this to the equivalence conjecture "an integer is non-square if and only if it has an even number of divisors".

11.6 Evaluating the method of lemma-incorporation

The process of providing a formal representation of Lakatos's method of lemma-incorporation as a computer program has forced us to answer questions such as: how can we represent an informal proof? How can we uncover hidden assumptions in a proof? How can we modify a faulty conjecture or a faulty proof, given a counterexample? How can we formalise the surprise we feel when an example behaves in an unexpected manner in a proof?

Much of our work for this method consisted of defining and implementing formal procedures for modifying proofs and conjectures in the way specified by Lakatos, and we have not performed hypothesis testing in this method. Instead, we have produced a formal theory and shown that it is possible to derive the mathematical phenomena which Lakatos claims his theory explains. This provides evidence for Lakatos's argument, and shows that his method satisfies Sloman's first criterion for calling T a theory: that T must actually explain what it sets out to explain (see §11.7.1 below).

We have considered two examples of hidden lemma-incorporation, in different domains. In geometry, we considered Hilbert's theorem that for two points A and C, there always exists at least one point D on the line AC that lies between A and C. We analysed how Lakatos's method could help us to revise the proof of this theorem and the faulty conjecture statement. This is in part an application of Lakatos's method, and in part a further extension of his method, which inspired us when writing our algorithm. Therefore, aspects of this example are in $E\backslash I$, and aspects are in $I1\backslash E$. Our other inspiring example was an invented example in group theory: let G be a groupoid and g, h, k be in G. Then $gh = gk \rightarrow h = k$. As in the geometry case, this example was partially an application of Lakatos's method, and in part a further extension of his method, and therefore partly in $E\backslash I$, and partly in $I1\backslash E$.

We performed a session with HRL in which we tested an example of both global and local lemma-incorporation. The faulty conjecture was that the left hand cancellation law holds for groupoids, with the proof being the proof for the left hand cancellation law for groups. HRL modified the faulty conjecture to a true theorem, although one with a very limited domain of application. This example was neither an inspiring example for us nor for Lakatos, and is therefore in the set E, to the extent in which we were faithful to Lakatos's theory, and in the set $E1$, to the extent to which we have extended Lakatos's theory.

11.7 Further criteria of a good theory

Below we consider Sloman's first, second, sixth, eighth and tenth criteria in [Sloman, 1978], for what constitutes a good theory.

11.7.1 A theory should explain a range of possibilities

Sloman's first criterion for a theory says that the possibilities which the theory purports to explain must be validly derivable from the theory; that is, a theory must actually explain what it sets out to explain. By implementing Lakatos's theory as a computer program, we have created a way of testing this criterion. In terms of our sets above, this involves exploring I, and ensuring that Euler's conjecture and associated concepts, entities, conjectures and proofs can be generated by Lakatos's theory. We have shown this in the method of lemma-incorporation, which we modelled in HRL.

We described illustrative sessions in §9.5.1 (given a counterexample which is local but not global), §9.6.1 (given a counterexample which is both global and local), §9.7.2 (showing our model of surprise which is sometimes felt when given a global but not local counterexample), and §9.7.4 (given a counterexample which is global but not local). By generating the new proofs, concepts and conjectures shown in [Lakatos, 1976], these sessions show that Lakatos's method of lemma-incorporation does indeed explain those possibilities.

11.7.2 A good theory should be definite

Sloman's second criterion for a good theory says that it should be clear what the theory explains and what it does not explain. Lakatos was attempting to explain a phenomenon which is not deterministic, and his theory of mathematical discovery is neither deterministic nor predictive. By formalising Lakatos's theory as a program which can be tested empirically, we have provided a way of determining what the theory can explain. That is, we can run HRL with the Lakatos settings for the variables, in a variety of domains, and the conjectures, concepts, and proofs which are generated can be said to have been explained by Lakatos's theory (assuming that our model is faithful). It is more difficult to show that a possibility cannot be explained by the theory, i.e., that HRL would never generate it.

11.7.3 A good theory should be rigorous

Sloman's sixth criterion for a good theory says that the procedures by which possibilities are derived from the theory should be explicitly specified. We have clearly achieved this, both for Lakatos's theory and for our extended theory, by presenting a set of thirteen algorithms throughout the thesis.

In chapter 5 we give an algorithm which shows how a student is to determine when to perform the method of surrender (algorithm 1).

In chapter 6 we give three algorithms. Algorithm 2 shows how a student is to determine *when* to propose to bar an entity; algorithm 3 shows how a student is to *evaluate a proposal* to bar an entity; and algorithm 4 shows how a student is to determine *how* to perform monster-barring or monster-accepting.

In chapter 7 we give three algorithms. Algorithm 5 shows how to perform the method of piecemeal exclusion; algorithm 6 shows how to perform our extension to this in the method of counterexample-barring; and algorithm 7 shows how to perform the method of strategic withdrawal.

Finally, in chapter 9 we give six algorithms. Algorithm 8 shows how to determine which kind of lemma-incorporation to perform; algorithm 9 shows how to perform local but not global lemma-incorporation; algorithm 10 shows how to perform global and local lemma-incorporation; algorithm 11 shows how to find a hidden lemma and generate an explicit faulty lemma, using the first type of surprise which we identify; algorithm 12 shows how to find a hidden lemma and generate an explicit faulty lemma, using the second type of surprise which we identify; and algorithm 13 shows how to perform global but not local lemma-incorporation.

11.7.4 A good theory should be economical

Sloman's eighth criterion for a good theory says that it should not include elements which are not required to explain the possibilities. By reusing some of our code for the method of exception-barring in our implementation of the method of lemma-incorporation, we have progressed some way towards satisfying this criterion. Lakatos also made this point, that the method of lemma-incorporation is a "limiting case of the exception-barring method" [Lakatos, 1976, p.37].

11.7.5 A good theory should be extendable

Sloman's tenth criterion for a good theory says that it should be extendable. That is, the theory can be embedded within an improved, larger theory which explains more possibilities or more of the fine-structure of previously explained possibilities. We have suggested extensions to Lakatos's theory throughout the thesis: for instance, our analysis of when to surrender a conjecture, our method of counterexample-barring, new ways of determining when to reject or when to accept a 'monster' as a valid entity, and our analysis of different types of surprise which one might feel when faced with an unexpected counterexample. By doing this, we have shown that Lakatos's theory can be extended, both in terms of explaining more possibilities and more of the fine-structure. Our theory, in turn, could be extended, for instance, by implementing a more sophisticated interaction protocol between the agents (see §13.3).

11.8 Answers suggested by the computational approach

In chapter 2 we described ten criticisms of Lakatos [1976] which Feferman [1978] gave, and claimed that the process of providing a computational representation has enabled us to answer some of Feferman's questions, and has also raised more questions which we have answered (§2.4). These mainly consist of highlighting gaps in Lakatos's theory. We outline below the questions and criticisms of the theory, and ways in which our model has allowed, or might allow us, to answer them. We refer to the criticism number, from Feferman's ten criticisms in [Feferman, 1978, pp.316-320], in italic roman numerals.

11.8.1 The scope of the methods

Although Lakatos is praised for the extremely detailed and in-depth analysis of his case studies, in particular of Euler's conjecture, he has been criticised (see, for instance, [Feferman, 1978]) for only considering two examples. Certainly, it is difficult to claim to have found patterns general to mathematical and even other types of discovery from such a small sample. We see determining the scope of the methods as one of the major contributions of our work, and suggest an alternative to [Larvor, 1998, p.11] who claims that "this type of dispute [whether Lakatos's methods are typical or atypical of mathematical reasoning] can only be resolved by extensive historical research". Producing a computer model gives us an obvious way of testing the variety of domains to which the methods can be usefully applied.

It is worth noting that while Lakatos did claim that his methods are general enough to apply to other domains, both mathematical and non-mathematical, he did not believe them to be the

sole explanation of mathematical discovery. Indeed, as Larvor argues, Lakatos did not believe that there is a unique logic of mathematical discovery, much less that he had found it. (Larvor points out that it was the editors, rather than Lakatos, who gave the book the subtitle *the* logic of mathematical discovery.) Instead, in his original thesis at least, he states the more modest aim of pointing out 'some tentative rules which may help us to avoid some deeply entrenched wrong heuristic habits' (Thesis, p. 75; quoted in [Larvor, 1998, p.12]). This view, that there are many ways of practising mathematics is echoed in the philosophy of science; for instance, Bird [Bird, 1998, chap 8] and Feyerabend [1975] argue that there is no such thing as *the* scientific method.

Applying the methods to other areas of mathematics

Feferman [1978] argues that Lakatos's methods only explain a small fraction of mathematical reasoning. For instance, he argues that the main method, the method of proofs and refutations, fails to account for foundational changes before 1847 *(i)*. Feferman also questions Lakatos's claim that the method of proofs and refutations is most appropriate to young, growing theories *(ii)*. He argues that *(a)* Lakatos's main example of the method, Euler's conjecture, was not a young, growing theory, and *(b)* there are examples of young, growing theories, such as continuous probability measures, which progressed without recourse to counterexamples and therefore without recourse to Lakatos's method. Hacking [1981] argues that since Lakatos's reasoning assumes the hypothetico-deductive model, its relevance is restricted to this type of knowledge, and there are other styles of knowledge such as Crombie's six styles of reasoning (see [Crombie, 1994]). He warns us not to let "the eternal verities depend on a mere episode in the history of human knowledge" [Hacking, 1981, p.143].

Our implementation strategy has been to develop the methods in other mathematical domains, mainly number theory. In this way, we can help to ensure that we implement them in a general way. This also lets us investigate whether the methods are sufficiently general to produce interesting mathematics in areas other than topology, vector algebra and real analysis.

Applying the methods to empirical sciences

[Lakatos, 1976] is often seen as Lakatos's attempt to apply Popper's philosophy of science to mathematics. There are both methodological and epistemological parallels: the view that mathematics advances by studying refutations and therefore practitioners should focus on finding counterexamples and anomalies, and the belief that there is no certain knowledge: both mathematics and science are fallibilist. We have already noted one key difference between Popper and Lakatos, the discovery/justification distinction. Another difference (pointed out by Larvor [1998]) is what we should do with the refutations once we have found them. Popper's

naive falsificationism tells us to reject the hypothesis, whereas this reaction is the first and most naive method in [Lakatos, 1976], the method of surrender, to which Lakatos devotes only one out of the total one hundred and twenty pages in the book. More sophisticated methods use the counterexample to refine the conjecture and concepts in it.

Lakatos partially addresses this problem by inheriting Kuhn's ideas on demarcation. Kuhn [1970] argued that the boundaries between scientific and non-scientific knowledge are not sharp; and Lakatos thought that the degree to which mathematics and science are the same type of (empirical) knowledge, corresponds to the degree to which his methods apply to science as well as mathematics. Lakatos claimed that "mathematical heuristic is very like scientific knowledge – not because both are inductive, but because both are characterised by conjectures, proofs, and refutations. The – important – difference lies in the nature of the respective conjectures, proofs (or, in science, explanations), and counterexamples" [Lakatos, 1976, p.74]. He credits Polya's stress on the similarities between scientific and mathematical heuristic as one of the most important contributions of his work (see [Lakatos, 1976, p.74], footnote 1).

Feferman [1978] also asks *(x) what is distinctive about mathematics?* He argues that Lakatos's methods do generalise to other domains, that his logic of mathematical discovery is really a logic of *rational* discovery. However, Feferman sees this as a shortcoming, claiming that they would then be overly general and "could account only for a few gross features of the actual growth of mathematics" [Feferman, 1978, p.320].

11.8.2 Applying Lakatos's methods to other types of conjecture

Feferman [1978] points out *(vii)* that all the examples of conjectures given by Lakatos are of the form $\forall x\ [A(x) \rightarrow B(x)]$ and gives examples of other types of conjecture found in mathematics and the form their refinement might take. This is also an aspect which has arisen in our implementation and we have suggested and implemented ways of applying the methods to other types of conjecture.

A related criticism is that, with the exception of strategic withdrawal, Lakatos's methods are only applicable to the type of conjecture which could be falsified by counterexample [Feferman, 1978]; *(ii)* and *(iii)*. This corresponds to the criticism of Popper's falsificationism (for example, see [Bird, 1998]), that it only applies to scientific hypotheses which are generalisations. This excludes, for instance statistical hypotheses (nothing can falsify a probabilistic hypothesis). While we cannot avoid this criticism, by running our model on different domains, and considering the conjectures it generates, we have to a certain extent investigated how limiting this is.

11.8.3 Comparing the methods

In both Lakatos's and commentators' writings, for example [Corfield, 1997], there is a clear hierarchy of methods, where they are presented as being increasingly sophisticated. Therefore, much work focuses purely on the final method to be described: proofs and refutations. Indeed, to the best of our knowledge, the first method – of surrendering a conjecture as false when faced with a counterexample – is not mentioned in any book on Lakatos. Lakatos wrote that "Mere 'falsification' (in Popper's sense) must not imply rejection" [Lakatos, 1981, p.116]. He called exception-barring, monster-barring and monster-adjusting 'conventionalist strategems' [Lakatos, 1981, p.117], and thought that they are *ad hoc* in the sense that once applied, the new conjecture has no more empirical content than the old one. Therefore they cannot form part of a progressive research programme (i.e., one which successfully predicts novel facts). Commentators usually make scant reference to these methods, with the assumption often being made that [Lakatos, 1976] is solely about the final method.

By running our system on different combinations of the methods and evaluating the resulting mathematical theories, we have been able to investigate these views. For instance, we hold that surrender can be useful in preventing resources from being wasted on uninteresting conjectures. The challenge is to find ways of knowing when to use a counterexample to surrender a conjecture, and when to use it as a catalyst for refining the conjecture. Similarly, we hold that the method of exception-barring has enabled us to generate interesting conjectures. Moreover, by allowing us to explore concepts more fully, monster-barring can generate interesting discussion and highlight entities which are in some way odd.

11.9 Applying Lakatos's MSRP to this project

The question of whether we can apply Lakatos's methodology of scientific research programmes (MSRP) to mathematical research programmes has been raised by very few philosophers. Koetsier [1991] was one exception, arguing that the analogy between science and mathematics is not close enough to merit the cross-over. This was in contrast to Hallett [1979a,b], and Oliveri [2006], the latter of which used mathematical case studies to rebut Koetsier's objections. We could view the case study of Euler's conjecture in Lakatosian terms by including Euler's conjecture that for any polyhedron, $V - E + F = 2$, in the hard core, and the various monster-barring definitions of polyhedron, face, edge, etc. in the protective belt (monster-barrers such as *Delta* may characterise it thus). Alternatively, the hard core may consist in the more general hypothesis that there is a relationship between the number of edges, faces and vertices of a polyhedron, analogous to the two dimensional case. Viewed in this way, Euler's conjecture

and subsequent modifications would be in the protective belt. (Clearly the question of which elements are in the hard core of a programme is subjective.) In this case we might interpret the mathematical analogy of "predicting novel facts" to be generating new open conjectures, and "some of the excess content being corroborated" as generating new theorems. However, in this section we are more concerned with whether we could apply Lakatos's criteria of a good theory, or a progressive research programme, to his philosophy of mathematical progress, and to our extended model, than whether, or how, it may be applied to mathematical examples (we leave this latter question to [Koetsier, 1991], [Hallett, 1979a,b] and [Oliveri, 2006]).

We can see the hard core of Lakatos's research programme as comprising the following hypotheses:

1. it is possible to write a rational reconstruction of mathematical progress in terms of the theory of the interaction of counterexample and conjecture, proof and concept definitions;
2. informal, quasi-empirical mathematics does not grow through a monotonous increase of the number of indubitably established theorems, but through the incessant improvement of guesses by speculation and criticism, by the logic of proof and refutations; and
3. the context of discovery and justification are conjoined.

The protective belt of Lakatos's research programme by nature changes throughout the programme. It includes the following hypotheses:

1. the methods of surrender monster-barring, piecemeal exclusion, strategic withdrawal, and lemma-incorporation help to describe the progress made in examining Euler's conjecture and connected proofs, definitions and entities; and
2. we can distinguish between local and global counterexamples and use this distinction to help to determine what to do when faced with a counterexample.

This is modified as the different methods evolve and form a hierarchy (for example, monster-barring turns into monster-stretching and lemma-incorporation into proofs and refutations).

According to Lakatos's criteria, this was a progressive research programme in the sense that it retained its coherence throughout the programme, i.e., the assumptions in its hard core were unchanged. The question of whether it generated any novel predictions is less clear, although he did predict that presenting mathematical research and textbooks using his heuristic style

rather than the typical deductivist style would benefit both experts and students. We could also argue that Lakatos's research programme led to the prediction, or suggestion, that it could be represented computationally, although it does not necessarily imply this. The programme was non-scientific to the extent that it failed this criterion, which we would expect as it was a philosophical programme.

Our programme shares the same hard core as Lakatos's programme, with the additional hypothesis that *it is possible to give a computational reading of mathematical progress*. This process of giving a computational representation of Lakatos's theory has led us to extend and modify the protective belt by further considering the following:

- when to use the method of surrender;
- when and how to use the method of monster-barring;
- how we can distinguish different types of exception-barring ;
- how we can determine when to use each type of exception-barring;
- how the methods might apply to different types of conjecture;
- how we can represent an informal proof;
- how we can uncover hidden assumptions in a proof;
- how we can modify a faulty conjecture or a faulty proof, given a counterexample;
- how we can formalise the surprise we feel when an example behaves in an unexpected manner in a proof; and
- which other domains Lakatos's theory might apply to.

Our extended programme has given rise to predictions such as *it is possible to fine-tune the methods of surrender, monster-barring and exception-barring*, and *we can usefully apply Lakatos's exception-barring methods to the field of automated theorem proving*, which we have tested in chapters 10 and 12. Additionally, it suggests the following hypotheses:

- Lakatos's method of surrender can be seen as comparable to Kuhn's ideas on paradigm revolution;
- it is useful to base the selection of a concept definition during the method of monster-barring by protecting definitions which are either very young (and thus need time to prove themselves) or very established; or

- we can evaluate the interestingness of an entity by considering the diversity of conjecture which it breaks;
- the method of monster-adjusting can be seen as a special case of monster-barring;
- a model of Lakatos's theory would be an aid to scientists;
- a model of Lakatos's theory would be an aid to students;
- we can use Lakatos's theory to develop a meta theory, for instance, about which is the most useful method to use in a given situation;
- we can develop a way of using the exception-barring methods to aid theorem proving by splitting a theorem into different conjectures which are then easier to prove, and finally
- by developing and running our model in a variety of mathematical and non-mathematical domains we can determine their domain of application.

These and other hypotheses which our research programme suggests are considered in chapter 13.

Building, running and evaluating our model enable us to test predictions in a way which was less obvious under Lakatos's original research programme. Therefore, we argue that not only is it progressive according to Lakatosian criteria, but that our extended programme is scientific where Lakatos's was not.

11.9.1 Falsification

Recall that a theory T is falsified if and only if another theory T' has been proposed with the following three criteria:

(i) T' predicts novel facts, i.e., phenomena which was not predicted by T (this is a sign of a theoretically progressive research programme);

(ii) T' explains all of the confirming instances that T explained; and

(iii) some of the excess content of T' is corroborated (this is a sign of an empirically progressive research programme).

We can see Lakatos's theory as T and our extended theory as T', and interpret the philosophical analogy of "predicting novel facts" to be suggesting new methods by which mathematics progresses, and "some of the excess content being corroborated" as these new methods explaining

new areas of mathematics. Then we can argue that our theory T' has predicted novel facts, such as the counterexample-barring method, which has been corroborated by explaining how, for instance, Goldbach's conjecture, might be thought of. We cannot argue, however, that our extended model has falsified Lakatos's theory (in a Lakatosian sense), since our computational model fails the second criterion. For instance, we have not implemented Lakatos's method of monster-adjusting (see section 13).

11.10 Summary

We have evaluated our project from a philosophical perspective, using criteria from Thagard [1993], Sloman [1978] and Popper [1972]. The main two criteria which we have considered are that a good theory should be general, and that it explain more than it set out to explain. We have used these criteria to evaluate Lakatos's methods of surrender, monster-barring, exception-barring and lemma-incorporation, and our implementation of them. We have described these criteria and our evaluation in terms of five sets: I is the set of examples which inspired Lakatos's theory, E is the set of possibilities which Lakatos's theory explains, $I1$ is a larger set of examples which inspired us in our extended computational theory, $E1$ is the set of possibilities which our theory explains, and U is the set of all possible mathematical conjectures. The first criterion considers the size and type of elements in E and $E1$. We have identified elements in both of these sets, and in particular elements in $E1 \backslash E$, which suggests that our theory is more general than Lakatos's theory. The second criterion considers the size of $E \backslash I$ and $E1 \backslash I1$ and, again, we have identified elements in both of these sets. Where relevant, we have presented output from HRL. We have also considered five other criteria suggested by Sloman [1978]: that a theory explain a range of possibilities, and a good theory be definite, rigorous, economical, and extendable, and we argue that our approach has satisfied these criteria. We have discussed our project with reference to Feferman's criticisms of Lakatos's theory, and argued that the computational approach has suggested answers to some of these. Finally, we have discussed our project and Lakatos's project with reference to Lakatos's own criteria for evaluating scientific research programmes.

Chapter 12

Application to automated theorem proving

In this chapter we further substantiate our claim that it is useful to automate Lakatos's work, by describing the application of his exception-barring methods to the field of automated theorem proving[1]. The chapter is organised as follows: firstly, in §12.1, we describe a spin-off system, Theorem Modifier, or TM, in which we have automated the method of exception-barring. We then describe experiments which we have performed in order to test the hypothesis that TM can find meaningful modifications to non-theorems, and discuss the results, in §12.2. We conclude in §12.3.

12.1 The Theorem Modifier System

Colton and Pease [2004] have developed a spin-off system, Theorem Modifier, or TM, which uses Lakatos's methods to modify faulty conjectures into true ones. Given a conjecture, the system uses the Otter theorem prover [McCune, 1990] to first try and prove the conjecture. If it fails, the system uses the MACE model generator [McCune, 2001] to produce examples which support the conjecture, and examples which falsify the conjecture. We then use concept barring and strategic withdrawal methods implemented within HRL, to find concepts covering a subset of the falsifying examples and/or concepts covering a subset of the supporting examples. The system uses the first type of concept to perform piecemeal exclusion, and the second type of concept to perform strategic withdrawal, again to specialise the conjecture. A set of modified

[1] Note that the work in this chapter was jointly carried out by Colton and Pease. In particular, Colton did much of the programming and all of the interfacing to Otter and MACE, while Pease did much of the testing. Analysis of the results was carried out jointly, with Colton directing the project.

conjectures is generated this way, and each is tested for theorem-hood by Otter. The user is shown only those modified theorems which Otter has proved.

TM works by taking in a conjecture of the form $A \Rightarrow C$ where A is a conjoined set of axioms, and C is the conjecture a user wishes to prove, modify or disprove. We limited TM to algebraic domains; for instance group theory, where there is a single binary operator which satisfies the three axioms of associativity, identity, and inverse.

Stage 1: Given a conjecture, TM first performs two preliminary checks to see whether it is worth modifying. These are:
(i) if Otter can prove in a user-specified period of time that the conjecture is true, i.e., $A \Rightarrow C$, then TM reports this and returns the proof
(ii) TM negates the conjecture and invokes Otter to try to prove the negation. If the negation is true, then TM will not be able to modify it, hence this ends the session.

Stage 2: If unsuccessful, then TM performs a further check before invoking MACE and HR – whether the conjecture is true if and only if we limit the objects in the domain to (a) the trivial group, i.e., if $A \Rightarrow ((\forall\, a,b\ (a = b)) \Leftrightarrow C)$, or (b) non-trivial groups, i.e., if $A \Rightarrow ((\exists\, a,b\ (a \neq b))) \Leftrightarrow C$. This check is a type of modification inspired by Lakatos's exception-barring methods, where (a) the only supporting example is the trivial group, so we limit the domain to this (a type of strategic withdrawal), and (b) the only counterexample is the trivial group, so we exclude it from the conjecture (a type of piecemeal exclusion). We apply these methods separately at this stage as it is often the case that a theorem is true *only* for the trivial algebra, in which case the theorem is usually uninteresting. The opposite case, that the theorem is true for everything *but* the trivial algebra is rare.

Stage 3: If the conjecture gets beyond these checks, then TM has the chance to modify it. To do this;
(i) TM invokes MACE to generate two sets of algebras; the first containing those which support the conjecture, and the second containing those which contradict it.
(ii) These sets are then passed to HR as objects of interest, as well as the conjecture, from which it extracts the core concepts. It uses this input to produce a theory, for a user-specified number of steps.
(iii) TM then identifies all the different types of group which have been defined, i.e., specialisation concepts (such as Abelian or self-inverse groups) which HR has invented. TM extracts those which describe only the algebras which support the conjecture (or a subset of them). For each extracted specialisation, M, TM forms the modified conjecture $(A \wedge M) \Rightarrow C$ by adding M to the axioms.
(iv) Otter is invoked to see which of these modifications can be proved, and any which are

proved are presented to the user (after stage 4).

Stage 4: Finally, TM evaluates whether its modifications are likely to be interesting to the user. It does this by testing whether:
(i) the only example to satisfy M is the trivial algebra, in which case TM invokes Otter to check whether $A \Rightarrow (M \Leftrightarrow (\forall\, a,b\ (a = b)))$;
(ii) the concept is a redefinition of the conjecture statement, for instance, M is the condition that a group is Abelian, when the original conjecture was *all groups are Abelian*, i.e., the modification is that *all Abelian groups are Abelian*. If *every* supporting example has the property prescribed by M, then TM uses Otter to try to prove: (a) $M \Leftrightarrow C$, (b) $A \Rightarrow (M \Leftrightarrow C)$, (c) $M \Rightarrow C$. TM marks these as probably uninteresting, though still reports them to the user, as it may be that the equivalence of C and M is surprising and non-trivial, and hence the modified conjecture interesting.

This process of modifying conjectures is an implementation of Lakatos's strategic withdrawal method. However, since TM instructs HR to use its negate production rule, for every specialisation M, the negation $\neg M$ will also be produced. Hence, if the examples of M contained all the *falsifying* examples for the conjecture, then $\neg M$ would describe a subset of the supporting examples, and hence would be used in a modification attempt. Therefore, TM also uses piecemeal exclusion to form the modifications.

12.2 Experiments and results

We have tested the hypothesis that TM can find meaningful modifications to non-theorems by using the TPTP library [Sutcliffe and Suttner, 1998]. From theorems in this library, we have generated non-theorems by removing axioms, changing or removing quantifiers, altering variables and constants, and altering bracketing. We have also tested TM on some of the non-theorems in TPTP. We set Otter and MACE to run for 10 seconds and HR for 3000 theory formation steps. From 98 invented non-theorems, TM produced valid modifications for 83% of them, and found an average of 3.1 modifications per non-theorem. When we tested it on 9 non-theorems already in TPTP, it found modifications to 7 of them (78%). For instance, we gave TM the non-theorem (RNG031-6 in TPTP) that the following property, P, holds for all rings: $\forall\, w,x\ ((((w*w)*x)*(w*w)) = id)$ where id is the additive identity element (see glossary). MACE found 7 supporting examples for this, and 6 falsifying examples. HR produced a single specialisation concept which was true of 3 supporting examples: $\nexists\, b\ (b*b = b \wedge -(b+b = b))$. Otter then proved that P holds in rings for which HR's invented property holds. Hence, while TM couldn't prove the original theorem, it did prove that, in rings for which $\forall x, (x*x = x \Rightarrow x+x = x)$, property P does hold.

12.3 Conclusion

Our results show that TM can be used to produce theorems from conjectures which are either false or open, and we hold that these modified theorems are interesting. This argument, as presented in [Pease and Colton, 2004] and [Colton and Pease, 2004], provides support for our argument that automating Lakatos's methods has applications to the field of automated theorem proving. By adding more robustness and flexibility to automated theorem proving, we increase the potential to aid mathematicians, as well as producing more intelligent behaviour. Furthermore, we have demonstrated the application of Lakatos's work to another algebraic domain: ring theory.

Chapter 13

Further work

This project is an initial implementation of Lakatos's theory of mathematical discovery and justification. We have argued that it is both possible and useful to give a computational reading of Lakatos's theory. There are many ways in which our work could be extended. These include improving our implementation of Lakatos's methods and implementing further aspects, described in §13.1; generating an initial problem and an initial proof scheme, described in §13.2; increasing the sophistication of the agency in terms of interaction and autonomy, described in §13.3; making the parameter-settings flexible, described in §13.4; giving a cognitively plausible notion of mathematical concepts, described in §13.5, and investigating applications of our system, described in §13.6. In this chapter, we discuss these improvements and estimate whether each improvement would be a long or short term project.

13.1 Improving our implementation of Lakatos's methods

13.1.1 Extending the method of surrender

We claimed in chapter 5 that the two questions concerned with the method of surrender are: 1) *when* should we give up on a conjecture?, and 2) what should we do next? We considered the first, which partly connected to how specialised the conjecture had become. The answer to the second question concerns specialisation. Lakatos calls it the *problem of content*, and it concerns the situation where a conjecture has been specialised to such an extent that its domain of application is severely reduced.

The methods of proofs and refutations, lemma incorporation, exception-barring and monster-barring may increase the certainty of the theorem being true, but they decrease content, as each

new lemma, or condition we add reduces the domain of the theorem. [Lakatos, 1976, p 57] compares this to throwing the baby out with the bath water, and argues that a proof and theorem should explain *all* of the supporting examples, rather than just exclude the counterexamples. Sometimes this is possible, while sometimes there may not be a single theorem which explains all of the supporting examples. If this is the case, then we need to surrender the conjecture, and consider what to do next. One possibility would be to generate an *initial problem* (see §13.2.1) and to return to this once a conjecture had been surrendered.

Although surrender is presented as a naive and unproductive reaction to a counterexample, it can be of great use, and in particular the question of what to do next is interesting. There are parallels here to Kuhn's work on the patterns of science in [Kuhn, 1970]. Kuhn's *normal science* period is where the other methods are at play. His notion of *crisis* corresponds to a counterexample which cannot be ignored nor absorbed as a modification. The old discipline (or conjecture) is increasingly unable to solve pressing anomalies. The *revolution* concerns the shift from one paradigm (conjecture) to another. Finally, the *new normal science* is concerned with new problems (or conjectures). These new problems may or may not be better than the previous approach. Seen in this light, surrender is no longer a weak and premature rejection of a useful idea leaving one with no further recourse, but rather a triumphant overturning of old dogma.

13.1.2 Extending the method of monster-barring

Larvor[1] has suggested that our current algorithm in monster-barring for determining whether a concept definition is good or not, by testing to see how many conjectures an entity within the concept definition breaks, is too conservative. It may be the case that a new concept breaks current conjectures, but shows the promise of exciting new theories. We have partially addressed Larvor's point by allowing the students in the agency to select what they evaluate as interesting from the discussion vector, and to continue working with personal definitions which other students may not have accepted. This evaluation is based on the interestingness measures described in §3.1.3. One further aspect we could implement would be to use the number of theory formation steps as a confidence measure, so that young theories could be protected and allowed time to develop before having to prove themselves, and similarly concepts which have been accepted and form an important part of an older theory could be made much more difficult to change (this would fit with Lakatos's concept of a hardcore belt in science [Lakatos, 1970]).

More sophisticated methods we would like to implement include comparing the interestingness of the conjectures the entity in question either supports or break. Another way of evaluating

[1] Personal communication

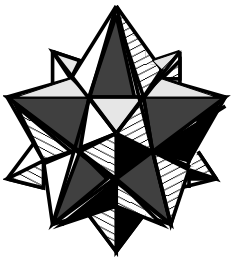
an entity would be to consider proofs of theorems in the theory. If the entity is used in proofs, then that would be a further reason for a student to accept the entity or keep it in the student's theory. It would also be interesting to consider the diversity of conjectures which are broken by the entity. Once disagreement over a concept definition has arisen, an agent might suggest one and justify why it is good (for example it may be used in many of the agent's conjectures). This may then lead another agent to re-evaluate the definition and rate it more highly. This is relevant to work by [Jennings et al., 1998], which we outlined in §3.2.4.

13.1.3 Implementing the method of monster-adjusting

For time reasons, we omitted implementing the method of monster-adjusting. Like the method of monster-barring, this method also exploits ambiguity in concepts, but reinterprets an object in such a way that it is no longer a counterexample. The example in [Lakatos, 1976] concerns the star polyhedron (see figure 13.1). This entity is raised as a counterexample since, it is claimed, it has 12 faces, 12 vertices and 30 edges (where a single face is seen as a star polygon), and thus an Euler characteristic of -6. This is contested, and it is argued that it has 60 faces, 32 vertices and 90 edges (where a single face is seen as a triangle), and thus an Euler characteristic of 2. The argument then turns to the definition of 'face'.

This method can be seen as a type of monster-barring, where the concept in question may be the right hand concept in an implication or equivalence conjecture, rather than the domain. Let us formalise monster-barring as follows: from conjecture $\forall x, P(x) \rightarrow Q(x)$, and (known) counterexample m such that $P(m)$ and $\neg Q(m)$, (re)define either P or Q so that for the m in question, either $\neg P(m)$ or $Q(m)$ is true. Monster-adjusting can be seen as a case of this formalisation, where the concept under debate is Q rather than P.

Modelling this ambiguity could be related to work carried out in humour research, for instance, [Ritchie, 2006], on reinterpretation and viewpoints, which exploits the ambiguity in language in order to make jokes. In this paper, Ritchie observes that many jokes rely on listeners reinterpreting the initial part of a joke once the punchline has been delivered. He outlines how these reinterpretations may happen in various ways. There is a clear analogy with the method of monster-adjusting. Implementing this method would be a rather open ended project: it could be a medium, or a long term project, depending on the level of detail.



The method of monster−adjustment is
used when someone brings up the star polyhedron,
with 12 vertices, 30 edges and 12 faces, as a counterexample.
Later others object to this interpretation, saying that this is
the case only if each face is a pentagon. This interpretation
can be 'adjusted' to seeing each face as a triangle, in which case
the equation becomes 32 − 90 + 60 =2 and therefore it is no
longer a counterexample.

Figure 13.1: The star polyhedron

13.2 Generating an initial problem and an initial proof scheme

13.2.1 Generating an initial problem

Lakatos distinguishes between an initial problem and an initial conjecture. In the Euler case study, the initial *problem* is to find out whether there is a relationship between the number of edges, vertices and faces on a polyhedron which is analogous to the relation which holds for polygons, that is, the number of vertices is equal to the number of edges. The initial *conjecture* is that for any polyhedron, $V-E+F=2$. After surrendering an initial conjecture, the initial problem should be reconsidered and a new initial conjecture, such as $V-E+F=2-2(n-1)+\sum_{k=1}^{F} e_k$ for an *n*-spheroid polyhedra, found and discussed.

One way in which this could be implemented in HRL would be to run two agents in two different domains. Either of the agents could send a conjecture which it evaluated as interesting to the other agent and a request to find an analogous relation in its domain. (Part of the interestingness evaluation could include whether it has been proved as a theorem by third party software, such as Otter [McCune, 1990].) This would constitute the initial problem. The second agent would need a way of determining whether it already has analogous conjectures and concepts, and a way of constructing new ones if necessary. Two concepts could be defined to be analogous if:

• they have been constructed with the same production rules and parameters, although the initial core concepts may be different (as the two agents are working in a different domain).

Two conjectures could be defined to be analogous if:

• they combine analogous concepts (as well as possibly other concepts).

If the agent did not already have concepts which shared the same construction history as the concepts in the conjecture then it would attempt to construct them from core concepts. The

agent would then have a list of analogous concepts and would check its list of conjectures to see whether it already had any which combined some combination of these concepts. If so, these are candidate initial conjectures and if they are later rejected then this list can be updated to find a new initial conjecture. If not, then combining the analogous concepts could become a gold standard which would guide the search for future conjectures. This would guide both the concept and conjecture formation.

13.2.1.1 Case study

Suppose that we run the agency with one agent working in the polygon domain and a second in the polyhedra domain. In table 13.1 we show a reduced version of a polygon domain in which a triangle, with *vertices a*, *b* and *c* and *edges e*1, *e*2 and *e*3, is an example of a *polygon*. Other concepts in the polygon domain would include *vertex-on-edge(P,V,E)* as well as core concepts *integer*, *addition* etc. from number theory. In table 13.2 we show a reduced version of a polyhedron domain, in which a tetrahedron, with *vertices a*, *b*, *c* and *d* and *edges e*1, *e*2, *e*3, *e*4, *e*5 and *e*6, is an example of a *polyhedron*. Other concepts in the polyhedron domain would include *vertex-on-edge(P,V,E)* and *face-on-edge(P,F,E)* as well as the same core concepts from number theory.

The first agent might construct the concepts *number of edges of a polygon (E)* and *number of vertices of a polygon (V)* respectively, as follows:

• [polygon002,size,[1]], and

• [polygon003,size,[1]].

This would then automatically form the conjecture that for all polygons, $E = V$. Suppose that this agent then sent this conjecture to the second agent, working with polyhedra, with the request for an analogous relation in that domain. The second agent would then check in its theory to see whether it had any concepts which shared the same construction history as the concepts in the conjecture.

The second agent would find, or generate, the concepts *number of edges of a polyhedron (E)*, and *number of vertices of a polyhedron (V)* and *number of faces of a polyhedron (F)* by performing the same production rule with the same parameters on its core concepts of *vertex*, *edge* and *face*. It would then look for a conjecture which combined these three concepts, and if it did not have one, it would use them as a gold standard to guide its search.

polygon001
polygon(P) "P" is a polygon polygon(triangle).

polygon002
vertex(P,V) "V" is a vertex of "P" vertex(P,V) → polygon(P) vertex(triangle,a). vertex(triangle,b). vertex(triangle,c).

polygon003
edge(P,E) "E" is an edge of "P" edge(P,E) → polygon(P) edge(triangle,e1). edge(triangle,e2). edge(triangle,e3).

Table 13.1: Partial representation of the triangle polygon

13.2.1.2 Polya's work on Induction and Analogy in Mathematics

Lakatos refers the reader to [Polya, 1954, chapter 3] for the initial phase of conjecturing and testing. Polya describes the process which mathematicians might go though in order to generate a conjecture which is worthy of a proof attempt, drawing parallels with natural science. He splits the process up into the stages shown below. We look at each of these steps and suggest ways in which they might be implemented. A general model would then link up all of these stages.

1. Examine empirical evidence.
2. If necessary, change the representation in order to better see patterns in the data.
3. Find a regularity which holds for all considered instances. Conjecture that it holds in all cases (i.e., use scientific induction).
4. Do more tests, and look at more examples. Do they satisfy the conjecture? If so, go on to 5. If not, go back to 2.
5. We need a worthwhile way of testing the conjecture. Another supporting example won't show much at this stage. Look at the original data again, and search for a deeper pattern. Try to show that the conjecture holds for a general type (possibly an infinite set).
6. Submit your conjecture to a severe, searching test that stands a good chance of refuting it.

Stages 1 and 3 could easily be implemented using the standard HR mechanism, by looking at data tables for given and generated concepts and generating conjectures about similarities between the data tables. The initial steps of finding a counterexample could also be easily implemented: for instance, asking other agents in the agency for a counterexample, or appealing to third party software, such as MACE [McCune, 2001], for a counterexample. The fifth

polyhedron001
polyhedron(P)
“P” is a polyhedron
polyhedron(tetrahedron).

polyhedron002
vertex(P,V)
“V” is a vertex of “P”
vertex(P,V) → polyhedron(P)
vertex(tetrahedron,a).
vertex(tetrahedron,b).
vertex(tetrahedron,c).
vertex(tetrahedron,d).

polyhedron003
edge(P,E)
“E” is an edge of “P”
edge(P,E) → polyhedron(P)
edge(tetrahedron,e1).
edge(tetrahedron,e2).
edge(tetrahedron,e3).
edge(tetrahedron,e4).
edge(tetrahedron,e5).
edge(tetrahedron,e6).

polyhedron004
face(P,F)
“F” is a face of “P”
face(P,F) → polyhedron(P)
face(P,F) → face(F)
face(tetrahedron,f1).
face(tetrahedron,f2).
face(tetrahedron,f3).
face(tetrahedron,f4).

Table 13.2: Partial representation of the tetrahedron polyhedron

stage of testing a conjecture and looking for a deeper pattern could be implemented by using exception-barring in order to form case splits of the conjecture, and then using Otter to try to prove some (or all) of the splits. We envisage that the second stage of re-representing the data and the final stage of submitting the conjecture to a severe test would be the most difficult to implement, constituting a medium or long term project. Work by [Karmiloff-Smith, 1990] on representational change may be relevant to the second stage.

A model of how we form initial problems might provide the beginnings of an answer to the question “where do problems come from in real mathematical activities?”.

13.2.2 Generating an initial proof scheme

We have not addressed the automatic generation of an initial proof scheme for a conjecture, since [Lakatos, 1976] starts with an initial proof. Doing so, however, would make a more complete cycle of mathematical discovery and justification. It should be possible to integrate HRL with a system which can generate schematic proofs, such as those systems described in [Baker, 1993] or [Jamnik, 2001b]. This would automate the process of finding a general, yet flawed proof of a conjecture, given some example proofs. To automate the full cycle, however, the difficult stage of generating example proofs automatically would be necessary. We show a

potential integration in figure 13.2. We see this as a medium term project.

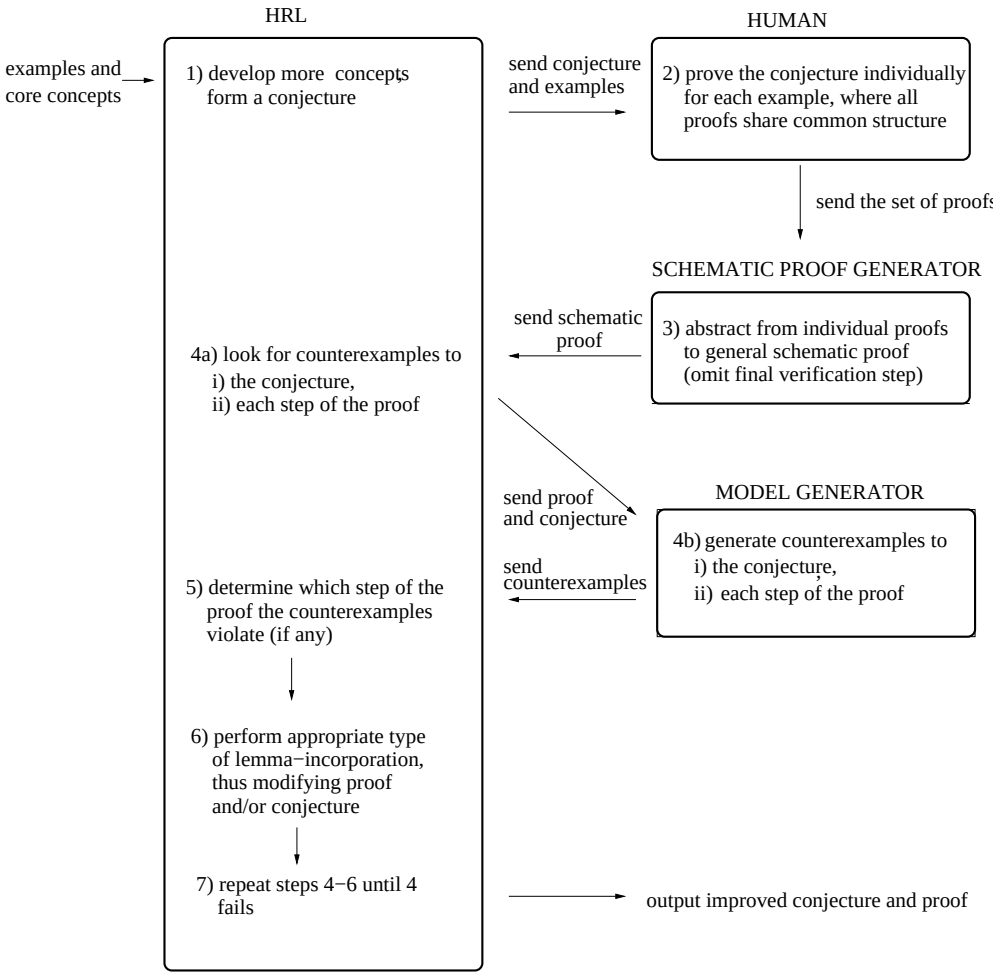


Figure 13.2: Building an interface between a HRL, a schematic proof generator and a model generator would make a more complete cycle of mathematical discovery and justification. Automating step two would be a further project.

13.3 A more sophisticated agency

We would like to improve the agency, both in terms of the interaction protocol in HRL and making the students more autonomous. The global theory is the collection of conjectures, concepts, objects of interest (both those provided by the user and those discovered as counterexamples to conjectures), and 'proofs' which the group discuss and accept. Three main factors can influence the production of the global theory. These are the user, the teacher and the students, and the complexity of the system increases respectively with the degree of influence that these have.

We can distinguish five different layers of complexity in the interaction between agents, shown below. Our approach has been to implement these sequentially, each time building on the last. This follows that of Brooks [1991], in which an architecture is built layer by layer, with increasing complexity. An advantage of this approach is that we can compare global theories in different types of system, to see which is the best.

1. Maximum user input (a puppet show). The user has control over which method is used by the students, and what should be added to the group agenda (and in which order).

2. Teacher/user using students as references (a dictatorship). The teacher plays the role of the user above.

3. Teacher allowing students to make decisions: the students decide which method they want to use on a particular faulty conjecture.

4. Students decide the group agenda (a democracy). Students decide the order in which responses are inserted into the agenda (for example, they evaluate all responses and vote on the order of importance).

5. Students are allowed class discussion: they can interact with each other directly and are not tied to responding to the teacher's requests. They can also send requests for conjectures etc. themselves.

HRL is currently at level two in some regards, and three in other regards. For instance, a student who has a counterexample to a conjecture under discussion will try to perform piecemeal exclusion if it is set to by the user, regardless of whether it evaluates the conjecture as interesting or not, or whether it can find an appropriate concept to exclude. In this regard, it is at the second level. However, students can also decide whether to perform monster-barring or counterexample-barring, based on aspects of their theories. In this regard, therefore, students are at the third level. Increasing the capabilities of HRL to level four would be a simple, short term task. Achieving the fifth level would be a significant improvement to HRL, and would be a longer task. This would be a worthwhile goal, since the type of discussion held in *Proofs and Refutations* was far more complex than we have modelled, and interaction between the students is an integral part of Lakatos's theory. One way of achieving this might be to make the teacher's role more analogous to the role of a blackboard.

13.4 Flexible parameter settings

Another way of increasing the sophistication of the agents would be to make the parameter settings flexible. Our current method of setting parameter values at the start of a session, which are then fixed throughout the session, is unsatisfactory as a way of capturing differences between mathematicians. Additionally, it is inflexible in that students cannot learn from their application of Lakatos's methods, or develop preferences during a run. In future versions of this system we would like to include a meta-level functionality. We envisage that this meta-level would work as a theory formation in which the input background knowledge would be current parameters in HRL: for instance, the Lakatos methods, the way in which they are applied, the number of individual theory formation steps, the percentage of entities for which a near-conjecture should hold, as well as parameters in HR, such as the production rules used in the generation of new concepts and the interestingness measures themselves.

Conjectures would be made about the way in which theories are being formed, such as *exception-barring always produces interesting results*, *the concepts which monster-barring produces are not interesting*, or *concepts which are surprising do not lead to interesting conjectures*. This is turn would affect the parameter values which students use when they operate once more on the object-level. Students would be able to change from object to meta-level; when and how often they do this might be one further parameter which is open to change. How to generate interesting theories might also be a topic of discussion within the agency, for instance one student might communicate its conjecture that *exception-barring always produces interesting results*, a second student might find a counterexample, i.e., a conjecture or a concept which was produced by this method but not considered by the second student to be a good result (the disagreement may arise if the first student does not know of the counterexample, or it may know it but evaluate it differently). Students could then perform one of Lakatos's methods on the conjecture: for instance, monster-barring might lead to analysis of the definition of "interesting".

Lakatos's dialectic itself evolves during the dialectic: even the methods themselves undergo a refinement, with monster-barring turning into monster-stretching and lemma-incorporation into proofs and refutations. Adapting parameter values in the way we have described would be a first step to capturing this subtlety. We anticipate that this would be a medium term project.

Colton [2001b] has already demonstrated that HR works on the meta-level. This worked very successfully as HR was able to form meta-theories in the same way as object-theories. Colton outlined ideas for using HR's meta-level reasoning functionality in a multiagent setting, in which one agent would produce a theory while another agent, working in near-parallel, would produce a theory about the way in which the first was producing its theory. The first agent

would be able to access this meta-theory and use it to alter its search accordingly. These ideas, however, have not yet been implemented. Other examples of automated reasoning on the meta-level include Buchanan's program Meta-DENDRAL, which reasoned about the program DENDRAL [Buchanan and Feigenbaum, 1978], and Lenat's program Eurisko, which reasoned about the AM program [Lenat, 1983].

13.5 A cognitively plausible notion of mathematical concepts

In this thesis we make no claim that the way in which concepts are represented in the input data is cognitively plausible: it clearly is not, especially in the case of the concepts which are purely defined extensionally. While our representation was sufficient for our goal of modelling Lakatos's theory, it would be interesting to replace the current way in which we input data with techniques which better characterise the way in which humans represent, use and store mathematical concepts. Very little work has been carried out in this area; a deficiency which led cognitive scientists Lakoff and Núñez to claim in 2001 that "there was still no discipline of mathematical idea analysis"[Lakoff and Núñez, 2001, p.*XI*]

Sloman [Sloman, 1978, Chapter 8] also argues that we are not yet able to answer questions such as "What are number concepts?", "How is it possible for them to be learnt?", "How is it possible for them to be used?", "How is it possible to discover non-empirical facts about them?", but does offer a preliminary exploration of some of the issues. He suggests that methods of conceptual analysis, traditionally used by philosophers and linguists, can also be important for a psychological account how we understand mathematical concepts. Conversely, a new psychological account could feed back into old philosophical problems about the nature of numbers. Sloman presents a series of increasingly complex data structures as possible ways of representing this knowledge. For instance, in order to grasp the number *three*, a child must build associations between different representations of it (both written and oral) and information such as the fact that it is a number name, that its successor is four and its predecessor is two, the fact that it is a prime number, that it is odd, its addition table, i.e., $(0+3)$, $(3+0)$, $(2+1)$, $(1+2)$, $(1+1+1)$, its multiplication table, and so on. A theory which purports to explain how mathematical concepts are represented must specify a structure which is can be gradually built up, extending the structure so as to contain more explicit information about itself, how it can be partially modified or replaced, how complex information is held an efficiently accessible form, and how someone might learn new procedures for doing things with the structure. Sloman also highlights the educational potential inherent in a better understanding of how we build and use representations of a mathematical concept such as *number*. One advantage would be an increased respect for the abilities of children to grasp such concepts.

Lakoff and Núñez have also addressed the deficiency. They claim that mathematics is about ideas and understanding, rather than formal proofs from formal axioms and definitions, which could be seen as a cognitive counterpart to Lakatos's philosophical ideas. In particular, they argue that mathematical concepts develop by means of conceptual metaphors which are rooted in our biology and are best understood in light of the embodied mind.

Lakoff and Núñez claim that we are born with some minimal innate arithmetic. This includes capacity for subitizing (immediately recognising how many objects there are in a small collection), simple adding and subtracting of small numbers, and making rough estimates of the number of objects in a group.

In order to form more complex mathematical ideas, we need to be able to form two types of conceptual metaphor between innate arithmetic and the more complex arithmetic of natural numbers. Firstly, we need to be able to make *grounding metaphors*. These allow us to project from everyday experiences onto abstract concepts. For instance, we make the metaphor between putting physical objects into groups, and the abstract concept of addition. Lakoff and Núñez identify four grounding metaphors: forming collections, putting objects together, using measuring sticks, and moving through space. The second type of metaphor that we need to be able to make is a *linking metaphor*. This consists of blending different metaphors, for instance to blend subitizing with counting, and enables us to link arithmetic, for example, to other branches of mathematics. (It can be seen as forming metaphors where the source and target domain are themselves metaphors.) Lakoff and Núñez argue that much of the abstraction of higher mathematics is the consequence of this type of systematic layering of metaphor upon metaphor and show where mathematical concepts and laws come from, in terms of these metaphors.

In each of the grounding metaphors, there is a relationship between our physical world and mathematics. There are collections of objects, complex objects with parts, physical segments and locations, in the physical world. Lakoff and Núñez call this relationship the *Numbers are Things in the World* metaphor and argue that the grounding metaphors induce this more general metaphor. They use this general metaphor to derive the mathematical concept of *closure*, which they hold, does not derive from innate mathematics. (For example, they claim that we can subitize three objects and we can subitize four objects, but we cannot subitize seven objects.) Instead, they argue, from experience we notice the general principle that "an operation on physical things yields a physical thing of the same kind"[Lakoff and Núñez, 2001, p.81] and, therefore, we can use the *Numbers are Things in the World* metaphor to conclude that *an operation on numbers yields a number of the same kind*. This central concept of mathematics has led to the extension of the number system in order to achieve closure with respect to

different arithmetic operations. For instance, the operation of subtraction led to zero and negative numbers, division led to rational numbers, and taking roots led to irrational and imaginary numbers.

Another example of metaphor being central to our mathematical concepts is what Lakoff and Núñez term the *Basic Metaphor of Infinity*. This is a single general conceptual metaphor in which processes that go on indefinitely are conceptualised as having an end and an ultimate result. Special cases of this general metaphor include infinite sets, points at infinity, limits of infinite series, infinite intersections and least upper bounds.

To our knowledge, the work by Sloman and Lakoff and Núñez has not yet been explicitly represented computationally. However, a cognitively plausible account of the way in which we represent and manipulate mathematical concepts is vital to any model which purports to explain human mathematical reasoning. For instance, in §6.1 we refer to the fact that mathematicians had formed conjectures about polyhedra and even formulated detailed proof plans of conjectures which involved complicated manipulations of polyhedra, without ever explicitly defining what a polyhedron is (highlighted by [Lakatos, 1976]). Clearly, theorems can be meaningful even without an explicit definition of the concepts involved. In Euler's example the concept of a polyhedron is evidently embedded within a lot of knowledge about two and three-dimensional Euclidean space, as shown by Cauchy's initial proof. How that intuitive information is acquired, how it is represented, and how it is used, are important questions, to which Sloman [1978], and Lakoff and Núñez [2001] have started to give theoretical answers. A model of their ideas might entail combining a model of an embodied world (either simulated or real) with work from the areas of metaphor and conceptual blending. It might be the case that the fields of robotics and of research into metaphors are both sufficiently developed for such a relationship to be established. Work in the former includes [Almeida e Costa and Rocha, 2005], while Pereira and Cardoso [2003] and Veale and O'Donoghue [2000] who present computational representations of conceptual blending, extending work by [Fauconnier and Turner, 1998], are examples of work in the latter.

We see an implementation of these ideas as being a long term project, of particular interest.

13.6 Applications of our system

An aid to scientists

We could use Lakatos's ideas to help us to develop new techniques which aid scientists in their work [Langley, 2002]. As with much work in the automated reasoning field, HR [Colton,

2002] was originally developed for this purpose, i.e., to help mathematicians discover new results, and we consider that our extended version of automated theory formation contributes to this purpose. Research, for instance [Fielder, 2001] and [Langley, 2002], has shown that scientists prefer to know the background of a claim made by a computer program, rather than take it on trust. This fits perfectly with Lakatos's philosophy of presenting the history of a result rather than isolated results with no explanation as to the thinking behind them.

Bundy [2006] has addressed the problem of why mathematicians tend not to accept automated proofs, despite computer proofs of theorems which mathematicians have been trying to prove for years. Examples of such theorems include the Kepler problem on the densest packing of spheres, and the four-colour theorem that any map in a plane can be coloured using four colours in such a way that regions sharing a common boundary (other than a single point) do not share the same colour. Bundy argued that mathematicians normally give two reasons for not accepting automated proofs; the proof may not be correct, and the proof is not human understandable. Of these, Bundy suggested that the first was not the real issue since non-fatal errors are tolerated elsewhere in mathematics, for instance in computer algebra systems used as teaching aids (such as Maple), and in human proofs where errors may go undetected for hundreds of years. He argues that the main reason that mathematicians are reluctant to accept computer proofs is the latter, that computer generated proofs are inaccessible to humans. Using computers to make proofs more accessible, therefore, is a current challenge within the field of computational mathematical reasoning. A system which showed the evolution of a proof, the statement which it proved, and the concepts in the statement and the proof, could make the final proof much more accessible. Our implementation of Lakatos's theory would be one way of addressing this challenge.

An aid to students

Lakatos's work had a strong pedagogical flavour. He hypothesised that students of mathematics would benefit from a more realistic presentation of the history behind concept definitions, theorem statements, etc., since giving the motivation for defining a certain set of entities would aid understanding rather than seeming arbitrary. As Kadvany [2001] argues, "pedagogy is a main topic in Lakatos's philosophy, and compliments his role as an educator and pedagogical philosopher" [Kadvany, 2001, p. 8].

Lakatos's theory can be seen as a theory of 'debugging' applied to mathematics[2]. Counterexamples indicate bugs of some sort, in concepts, conjectures, proofs, etc., and Lakatos's heuristic methods can be seen as different debugging strategies. The role of debugging in learning is demonstrated by Sussman [1975], in which he discusses various kinds of bugs that can occur,

[2]Thanks to Aaron Sloman for pointing this out.

techniques for dealing with them, and kinds of learning that prevent similar bugs from being written. Polya [1945] also discusses ways in which a teacher can set challenges which are at an appropriate level for a given student, and ways in which a student can make mistakes which then aid the learning process.

[Lakatos, 1976] is the story of a class of individual students learning, whose progress is an analogy for Lakatos's rational reconstruction of the progress of mathematical thought in this area. Kadvany [2001] points out that this uses Haeckel's biogenetic law that "ontogeny recapitulates phylogeny". If it is true that the progress in mathematical thought of an individual in some way mirrors historical progress, then a tool which modelled progress through a subject would have an obvious application as an intelligent tutor. Conversely, developing the model as an intelligent tutor would enable us to test Lakatos's claims about its value. Such an application of HRL would be a long term project.

Finding knowledge of interest to experts

One motivation behind computational philosophy of science is to develop new techniques which are useful in finding new knowledge of interest to experts [Langley, 1999]. To this end, it would be interesting to develop both HRL and our spin-off system, TM, so that mathematicians could use it to discover new, proved theorems. This would be a long term project.

13.7 Conclusion

[Lakatos, 1976] is a highly detailed account of how mathematics evolves, and while we hope to have achieved our goals of showing that it is both possible and useful to provide a computational reading of his ideas, our work is only a preliminary reading, and there are many future directions which the project could take. In this chapter, we have discussed six such directions.

Further possibilities, which would be valuable and interesting endeavours to pursue, include the following:

(i) extending the domain of application of the methods, and working with larger data-sets;

(ii) further areas of improvement of the system, as well as further experiments, which we identified in chapter 10;

(iii) using Lakatos's methods to develop a meta theory, for instance, about which was the most useful method to use in a given situation. Lakatos's dialectic itself evolves during the dialectic:

even the methods themselves undergo a refinement, with monster-barring turning into monster-stretching and lemma-incorporation into proofs and refutations;

(iv) developing a way of using the exception-barring methods to aid theorem proving by splitting a theorem into different conjectures which are then easier to prove, as described by Colton et al. [2005]. For instance, Colton et al. [2005] show how strategic withdrawal might be used to narrow the domain of application of an open conjecture about all groups, to all Abelian groups. This modified conjecture might then be proved, and another conjecture about the negation of the concept, in this case non-Abelian groups, would then be the new open conjecture. This may be possible to prove, or the process repeated and strategic withdrawal applied again, and so on until all of the different cases, and therefore the original conjecture, have been proven; and

(v) implementing other theories of scientific discovery, for instance, Popper [1972] and Kuhn [1970], and comparing the theories produced by each system. This would be an ambitious research project of great interest to computational philosophers of science.

Chapter 14

Conclusions

Lakatos attacked the view that mathematical knowledge is a priori and infallible, arguing instead that mathematics is a quasi-empirical subject, and mathematical knowledge evolves via analysis of conjectures, concepts and proofs, which are refined through interaction with counterexamples. Lakatos's work is still relevant today, as evidenced by its inclusion in a recent work on new directions in the philosophy of mathematics [Tymoczko, 1998].

We hold that Lakatos's work in the philosophy of mathematics is of interest to researchers in artificial intelligence, since it provides an account of the evolution of a mathematical conjecture, which is rare in its richness of detail and historical attention. We also hold that the field of artificial intelligence provides us with the tools to gain a novel perspective on Lakatos's philosophical work, since the process of implementing an idea as an algorithm forces one to clarify the idea, raises questions which otherwise may not arise, and results in a model which provides a new way of evaluating the idea. This thesis is the story of our exploration of the symbiotic relationship between the two fields.

Our central thesis is that it is possible to give a computational reading of Lakatos's theory. Our secondary thesis is that it is useful to do so, both from the philosophy of mathematics perspective, and the AI perspective. In §14.1 we argue that we have provided strong evidence for both of these positions. In §14.2 we discuss the contributions of this thesis. We conclude the discussion of our project in §14.3.

14.1 Have we achieved our aims?

In §1.2, we stated that the aims of this project were to:

(i) provide a computational reading of Lakatos's theory;
(ii) clarify and extend Lakatos's methods;
(iii) test the domains to which the methods apply;
(iv) evaluate the methods; and to
(v) evaluate our implementation.

Achieving the first of these aims would provide evidence for our primary thesis, i.e., we show that it is *possible* to give a computational reading of Lakatos's theory by doing so. The four remaining aims suggest ways in which providing such a reading would be a useful endeavour to undertake.

We have achieved our first aim by developing a computational model of Lakatos's theory: HRL. This is a multiagent system in which 'students' discuss their theories with each other via a 'teacher', which is based on the dialogue format which Lakatos uses to present his theory. Each agent is able to perform the method of surrender, monster-barring, exception-barring and lemma-incorporation. We describe our representation of each of these methods in chapters 5, 6, 7 and 9 respectively.

The process of developing this model has raised questions concerning how ambiguity arises and how it can be resolved; in what way we are surprised when faced with an unusual counterexample in a proof; and when we should surrender a conjecture. Writing algorithms which are then translated into a computer program has forced us to provide answers to these questions, and thus clarify some of Lakatos's methods. We have also extended Lakatos's theory by considering the application of his methods to other types of conjecture; in particular to equivalence, and non-existence conjectures. Implementing counterexample-barring and monster-accepting is a further extension of Lakatos's theory. In this way, we have achieved our second aim.

We achieved our third aim by testing the domains to which the methods apply: considering their application to further mathematical domains of number theory and group theory, as well as the non-mathematical domain of animal taxonomy.

Our system HRL has enabled us to evaluate the methods, thus achieving our fourth aim, by performing empirical experiments and analysing the output. This is described in chapter 10. In particular, we have tested three hypotheses:

• *it is possible to fine-tune our system*;

• *the methods are of general use*; and

• *the methods have application as AI techniques*.

Finally, in chapter 11, we have performed a philosophical evaluation of Lakatos's theory and our extended theory, based on criteria suggested by Thagard [1993], Sloman [1978] and Popper [1972]. Hence our fifth and final aim has been achieved.

14.2 Contributions

This is the first systematic automated realisation of Lakatos's theory. We have contributed to three areas: computational philosophy of science and philosophy of mathematics, automated theory formation, and automated theorem proving.

14.2.1 Computational philosophy of science

We contribute to computational philosophy of science by providing a novel perspective on Lakatos's theory. The computational perspective has suggested new ways to clarify, extend and evaluate Lakatos's theory.

14.2.2 Automated theory formation

We contribute to the field of automated theory formation by extending the capabilities of a state of the art automated theory formation program HR [Colton, 2002]. HRL can modify its conjectures, discuss and reject objects which are controversial, and represent and improve proof plans. These are all important processes in theory formation, and automating them has advanced the state of the art in this field.

14.2.3 Automated theorem proving

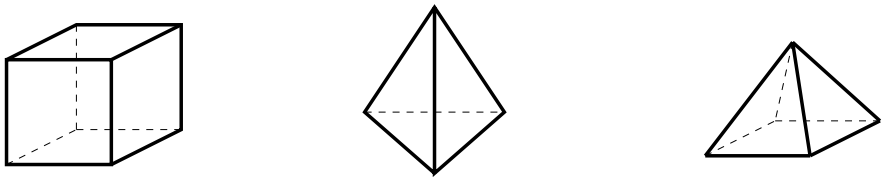
We contribute to the field of automated theorem proving by implementing a system, which is based on Lakatos's methods. This system is able to take in a conjecture, try to prove it and, if unsuccessful, produce modified versions of the conjecture which it *can* prove. This improves upon current automated theorem provers, which are inflexible in that if they are passed a conjecture which is either false or too difficult to prove within the allocated time, they simply fail.

14.3 Conclusions

Understanding the phylogeny and the ontogeny of mathematical thought is a challenge to researchers in the field of artificial intelligence, as well as to philosophers of mathematics and psychologists. Lakatos has shed some light on the problem, and we have used his insights to develop an automated theory formation system within which agents are able to discuss and refine conjectures, raise counterexamples, negotiate meaning of concepts, and improve upon a faulty proof. Sloman argues that "*the development of science, the learning of a child, and the mechanisms necessary for an intelligent robot all involve computational processes, which build up and deploy knowledge of the form and contents of the world. This is one of several points at which bridges can be built between philosophy of science, developmental psychology, and artificial intelligence.*" [Sloman, 1978, p. 61]. We have successfully built one such bridge.

Appendix A

Polyhedra and their Euler characteristic



CUBE
8 − 12 + 6 = 2

TETRAHEDRON
4 − 6 + 4 = 2

OCTAHEDRON
5 − 8 + 5 = 2

Figure A.1: Regular polyhedra

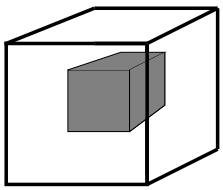


Figure A.2: The hollow cube: V - E + F = 16 - 24 + 12 = 4

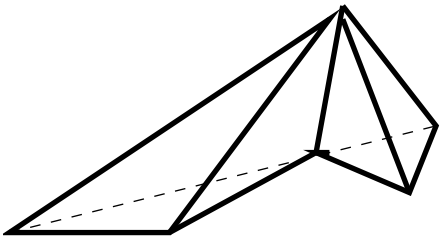


Figure A.3: The twin tetrahedra: V - E + F = 6 - 11 + 8 = 3

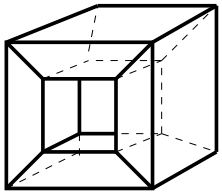


Figure A.4: The picture frame: V - E + F = 16 - 32 + 16 = 0

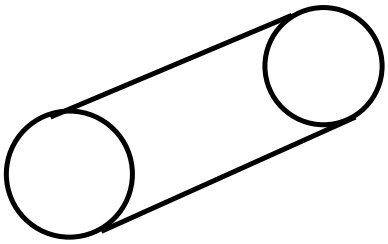


Figure A.5: The cylinder: V - E + F = 0 - 2 + 3 = 1

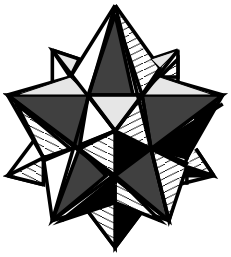


Figure A.6: The star polyhedron: V - E + F = 12 - 30 + 12 = -6 (if each face is a pentagon); or 32 - 90 + 60 = 2 (if each face is a triangle)

Appendix B

Mathematical proofs

B.1 Proof of Euler's conjecture

Theorem: If G is a connected plane graph, then $v - e + f = 2$

Proof: (By induction on e)

Base case:
(i) Let $e = 0$. Then $v = 1$ and $f = 1$. $1 - 0 + 1 = 2$
(ii) Let $e = 1$. Then we either have a circle, or a line, as shown in figure B.1.

Figure B.1: The two possible connected plane graphs consisting of one edge, and their associated Euler characteristics

So the theorem holds for $e \leq 1$.

Inductive hypothesis: Assume that $e \geq 1$, and that the theorem holds for all graphs with $e - 1$ edges.
Case 1: If G is a tree, then $f = 1$, and $v = e + 1$. So $v - e + f = (e + 1) - e + 1 = 2$.
Case 2: If G is not a tree, then remove an edge e belonging to some circuit. We now have planar graph H with v vertices and $e - 1$ edges, and $f - 1$ faces (i.e., we have removed an edge and a face). By the base case, we know that H satisfies Euler's conjecture, i.e.,

$v-(e-1)+(f-1)=2$
$v-e+1+f-1=2$
$v-e+f=2$. □

B.2 The Diagonalisation proof

Theorem: The set of real numbers is uncountable[1]

Proof:[2] Assume that the set of real numbers ($\mathcal{R}$) is countable. Then any subset of $\mathcal{R}$ is also countable. Take the interval (0,1). If it is countable then there exists a function f such that $f : \mathcal{Z} \rightarrow$ (0,1) and f is one-to-one onto. Then we can list all of the elements in (0,1) as they are mapped to $\mathcal{Z}^+$, say in $\mathcal{S}$. But then we can construct $b \in (0,1)$ such that b differs from every number in $\mathcal{S}$ by letting $b = 0.b_1b_2b_3...b_i...$ where b_i is 1 if the the ith position of the ith integer in $\mathcal{S}$ is 2, and 2 otherwise. Therefore $b \notin \mathcal{S}$. But $\mathcal{S}$ is a list of all elements, i.e., $a \in (0,1) => a \in \mathcal{S}$. This is a contradiction. Therefore $\mathcal{R}$ is uncountable. □

B.3 Hilbert's proof

We present the relevant axioms from [Hilbert, 1901] and Hilbert's proof of the theorem that for two points A and C there always exists at least one point D on the line AC that lies between A and C.

Axioms of incidence:

Axiom (I,3): There exist at least two points on a line. There exist at least three points that do not lie on a line.

Axioms of order:

Axiom (II,2): For two points A and C, there always exists at least one point B on the line AC such that C lies between A and B.

Axiom (II,3): Given any three points A, B, C of a line, one and only one of the points is between the other two

[1] An infinite set is countable if there exists a one-to-one onto correspondence between the set and the integers. Otherwise it is uncountable.

[2] This proof is a sketch only.

Axiom (II,4): Let A, B C be three points that do not lie on a line and let *a* be a line (in the plane ABC) which does not meet any of the points A, B, C. If the line *a* passes through a point of the segment AB, is also passes through a point of the segment AC, or through a point of the segment BC.

Theorem: For two points A and C there always exists at least one point D on the line AC that lies between A and C.

Proof: By Axiom (I,3) there exists a point E outside the line AC, and by Axiom (II,2) there exists on AE a point F such that E is a point of the segment AF. By the same axiom and by Axiom (II,3) there exists on FC a point G, that does not lie on the segment FC. By Axiom (II,4) the line EG must then intersect the segment AC at a point D. □

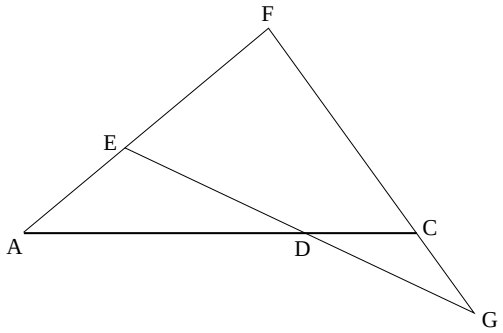


Figure B.2: Hilbert's proof of the theorem that for two points A and C there always exists at least one point D on the line AC that lies between A and C.

Appendix C

Output from HRL

C.1 Lemma-incorporation

This example shows HRL performing global and local lemma incorporation in the algebra domain (see §9.6.3).

Global conjecture:

∀ a b c d : a is a group ∧ b in a ∧ c in a ∧ d in a ⇝ a is a group ∧ b in a ∧ c in a ∧ d in a ∧ b * c = b * d implies c = d; i.e., the left hand cancellation law holds for a

Proof scheme:

1. ∀ a b c d : a is a group ∧ b in a ∧ c in a ∧ d in a ∧ b * c = b * d → a is a group ∧ b in a ∧ c in a ∧ d in a ∧ inv(b) * (b * c) = inv(b) * (b * d)
2. ∀ a b c d : a is a group ∧ b in a ∧ c in a ∧ d in a ∧ inv(b) * (b * c) = inv(b) * (b * d) ⇝ a is a group ∧ b in a ∧ c in a ∧ d in a ∧ (inv(b) * b) * c = (inv(b) * b) * d
3. ∀ a b c : a is a group ∧ b in a ∧ c in a ∧ exists d (d in a ∧ (inv(d) * d) * b = (inv(d) * d) * c) → a is a group ∧ b in a ∧ c in a ∧ exists d (d=id ∧ d * b = d * c)
4. ∀ a b c : a is a group ∧ b in a ∧ c in a ∧ exists d (d=id ∧ d * b = d * c) → a is a group ∧ b in a ∧ c in a ∧ b=c

The algebra 012100200 is a counterexample to both the global conjecture and to lemma 2.

After performing local and global lemma-incorporation, HRL modifies the global conjecture to:

$\forall$ a b c d : a is a group $\wedge$ b in a $\wedge$ c in a $\wedge$ d in a $\wedge$ (inv(b) * b) * c = (inv(b) * b) * d $\rightarrow$ a is a group $\wedge$ b in a $\wedge$ c in a $\wedge$ d in a $\wedge$ b * c = b * d implies c = d; i.e., the left hand cancellation law holds for a.

Appendix D

Further details of HR

In this appendix we extend the technical details on HR [Colton, 2002], which we briefly discussed in chapter 3.

D.1 Production rules

We motivate and describe the production rules in HR which we refer to in the thesis. We outline how they work, and show an example in terms of taking an old data table and generating a new one. We describe the parameters in standard PROLOG notation, and also include the HR notation for completeness. For details on how the new definitions are generated, see [Colton, 2002, chapter 6]. Most of the motivating and illustrative examples are taken from [Colton, 2002, chapter 6].

The match production rule

The match production rule is motivated by concepts such as a *square number*, where the predicate is multiplication, and the sub-objects are two divisors which are equal. Other motivating examples include *self inverse groups* in group theory, and *loops* in graph theory, where a node is adjacent to itself.

The match production rule extracts those rows from the data table where the entries in certain columns are equal. In PROLOG notation, it takes parameters $[X_1, X_2, X_3, ...X_n]$, where n is the arity of the input concept, and where the nth entry takes the value of the specified column X_i. For instance, the parameters $[X, Y, Y]$ would take a data table with arity three (i.e. there are three columns in the data table) and keep those rows of the data table where the entry in the third

column is equal to the entry in the second column. We show an example in table D.1, where the input concept is multiplicative pairs, and the output concept is squares with their integer square roots. In HR notation this is $< 1,2,2 >$.

Input

int	div	div
1	1	1
2	1	2
2	2	1
3	1	3
3	3	1
4	1	4
4	2	2
4	4	1
.	.	.
.	.	.
.	.	.
10	1	10
10	2	5
10	5	2
10	10	1

→

Intermediate

int	div	div
1	1	1
4	2	2
9	3	3

→

Output

int	div
1	1
4	2
9	3

Table D.1: An example of the match production rule, with parameters $[X,Y,Y]$. Here, any tuple from the input concept where the the entry in the second column is equal to the entry in the third column is entered into an intermediate data table. The output data table is generated from the intermediate data table by omitting the final column, i.e. by taking all tuples $[X,Y]$. In this example the input concept is *multiplicative pairs* and the output concept is*squares with their integer square roots*

The negate production rule

The negate production rule is motivated by concepts such as *non-squares*, *non-central elements* in groups, and *closed graphs* – which have no endpoints. It finds the complement of a concept, i.e. those entities with a certain property, such as integer, which do not satisfy the predicate of an input concept, such as square.

This rule does not require a parameterisation. We show an example in table D.2, where the input concept is squares, and the output concept is non-squares.

The size production rule

The size production rule is motivated by concepts such as the τ function in number theory, which counts the number of divisors of an integer, and the concept of an *order* of a group,

Input
integer
1
4
9

→

Integers
integer
1
2
3
4
5
6
7
8
9
10

→

Output
integer
2
3
5
6
7
8
10

Table D.2: An example of the negate production rule It takes an input concept and then finds the data table for the objects of interest in the theory in which it is working. It then produces a new data table consisting of all of the tuples in the the objects of interest data table which are *not* in the data table for the input concept. In this example the input concept is *square numbers*, the object of interest data table represents *integers* and the output concept is *non-squares*. This rule takes no parameters

which counts the number of elements in a group.

The size production rule counts the number of tuples of sub-objects which satisfy the definition of a concept. It takes parameters which specify which columns should be counted, and then counts each tuple in the specified column for each entry, and records this number. For instance, given parameterisation $[X]$ (or $< 1 >$ in HR notation), it counts the number of tuples for each entry in column one and records this figure. We show an example in table D.3, where the input concept is divisors of an integer, and the output concept is the number of divisors of an integer.

The split production rule

The split production rule is motivated by concepts such as a *prime number*, where the number of divisors is exactly 2. Another example if that of a *symmetric group*, which has exactly one central element.

The split production rule extracts rows from a data table where specified columns have specified values. For example, given parameterisation $[X, 2, Y]$ for a concept of arity three, the split rule will extract those rows whose second column has value 2. (This is represented as $< 2 = 2 >$ in HR notation.) We show an example in table D.4, where the input concept is the number of divisors of an integer, i.e. the τ function, and the output concept is integers which have exactly two divisors, i.e. prime numbers.

Input	
integer	divisor
1	1
2	1
2	2
3	1
3	3
.	.
.	.
.	.
10	1
10	2
10	5
10	10

→

Intermediate
integer
1
2
2
3
3
4
4
4
.
.
.
10

→

Output	
integer	number
1	1
2	2
3	2
4	3
5	2
6	4
7	2
8	4
9	3
10	4

Table D.3: An example of the size production rule, with parameters $[X]$. In this example, the size rule counts the number of tuples for each entry in column one of the input concept. It represents each entry the appropriate number of times in an intermediate data table. The rule then generates a new data table from the intermediate data table by counting the number of times an entry appears in the data table and recording this number next to the entry. In this example, the input concept is *divisors of an integer*, and the output concept is *the number of divisors of an integer*.

The compose production rule

The compose production rule was motivated by the *composition* of two functions. For instance, given functions $f(x)$ and $g(x)$, it was designed to construct the function $f(g(x))$.

This production rule takes two concepts, a primary and a secondary concept, as input. It then 'overlaps' the rows of the primary concept with those of the secondary concept in a way which has been specified: both *(i) which* pairs of tuples to overlap and *(ii) how* to overlap them. For instance, suppose we have primary concept P and secondary concept S, both of arity 3. We might specify that *(i)* we are to overlap all tuples $[P1, P2, P3]$ from P with any tuples $[P1, P3, S3]$ from S, to produce *(ii)* a new concept of arity 4 with tuples $[P1, P2, P3, S3]$. A simpler example would be to take two concepts P and S both of arity 1, and overlap any tuple $[P1]$ from P with any tuple $[P1]$ from S, to produce a new concept of arity 1 with tuples $[P1]$. The parameterisation in the latter example would be $[X]$ (or $< 1 >$). In table D.5 we show an example of this case, where the primary input concept is *odd numbers*, which has arity 1, and the secondary concept is *non-square numbers*, also of arity 1. Given the parameterisation $[X]$, the compose production rule will take any tuple $[P1]$ from the first concept and see whether there are matching tuples $[P1]$ from the second concept. If so, the tuple will be included in the

Input	
integer	number
1	1
2	2
3	2
4	3
5	2
6	4
7	2
8	4
9	3
10	4

→

Intermediate	
integer	number
2	2
3	2
5	2
7	2

→

Output
integer
2
3
5
7

Table D.4: An example of the split production rule, with parameters $[X, 2]$. The split rule extracts from the input data table those rows whose second column has value 2, and represents these in an intermediate data table. The rule then extracts each entry and omits the second column, $[X]$, in the data table for the output concept. In this example the input concept is the *number of divisors of an integer*, i.e. the τ function, and the output concept is *integers which have exactly two divisors*, i.e. prime numbers.

data table for the new concept. If not, the tuple will be omitted. The resulting new concept in this example is *odd, non-square numbers*.

Input (primary concept)
integer
1
3
5
7
9

$+$

Input (secondary concept)
integer
2
3
5
6
7
8
10

$\rightarrow$

Output
integer
3
5
7

Table D.5: An example of the compose production rule, with parameters $[X]$. The input is two concepts: a primary concept *odd numbers* and a secondary concept *non-square numbers*. Compose takes any tuples from the data table for the primary concept which also match tuples in the data table for the secondary concept, and adds them to the data table for the new concept. Any tuples which do not match are omitted. The resulting output concept is *odd, non-square numbers*

Glossary of Philosophical terms

Philosophers of mathematics are usually concerned with what kind of knowledge mathematical knowledge is, what sort of entity a mathematical entity is, whether mathematical truths correspond to reality, and whether mathematical truths exist independently of humans. Below we briefly describe traditional perspectives and proponents[1] .

Constructivism – the view that mathematical entities are mental constructions and do not exist independently of the mind. On this view we invent, as opposed to discover, theorems. Kant held this viewpoint.

Empiricism – the view that mathematical truths are based on sensory experience of the external world. All numbers must be numbers *of* something, such as bodies, sounds, or beatings of the pulse. Mill [1973] was an empiricist.

Game formalism – see *mathematical formalism*.

Logicism – the view that all of mathematics is reducible to logic. Frege and Russell were logicists.

Mathematical formalism – the view that there are no mathematical objects, only uninterpreted formal systems. Hilbert was a mathematical formalist. There are two types of formalism; *term formalism* and *game formalism*. The first is the view that mathematics is about linguistic expressions, that is, we identify mathematical entities, such as 0, with their names, '0'. Game formalism likens the practice of mathematics to a game played with linguistic characters. These symbols and expressions are devoid of reference, and so it would make no sense to ask, for example, what 0 refers to. Mathematics is seen as the manipulation of meaningless symbols, or a game played with arbitrary or conventional rules. Hilbert suggested one such rule in his definition of proof as "a sequence of formulae each of which is either an axiom or follows from earlier formulae by a rule of inference" Hilbert [1901]. Formal logic and Hilbertian view of proof is largely a twentieth century invention.

[1] We are grateful to Jeffery Ketland for his lecture notes on the philosophy of mathematics, from which we have drawn in this appendix.

Mathematical realism – the view that mathematical entities exist independently of the mind. Mathematical statements have a truth value which is independent of our knowing it, and humans *discover*, rather than invent theorems. Statements such as $2+2=4$ are true because they *correspond* to facts. Plato was a proponent of this view.

Term formalism – see *mathematical formalism*.

Glossary of Mathematical terms[2]

Abelian group: a group on which the defined binary operation is commutative; that is, if a and b are members of an Abelian group, $a * b = b * a$.

algebra: a system, such as a ring, group or field, endowed with finitary operations with specific properties.

algebraic number: any number that is the root of a polynomial equation with coefficients drawn from a given field, in particular the rationals. We denote this by $\mathbb{A}$

Euler characteristic: the following equation of a graph or polyhedron:
number of vertices - number of edges + number of faces

group: a set which is closed under an associative binary operation with respect to which there exists a unique identity element within the set and every element has an inverse within the set.

groupoid: a set together with a binary operation under which it is closed.

integer: a number that may be expressed as the sum or difference of two natural numbers; a member of the set $\{..., -3, -2, -1, 0, 1, 2, 3, ...\}$, usually denoted by $\mathbb{Z}$.

irrational number: any real or complex number that cannot be expressed as the ratio of two integers.

quasi-group: a groupoid in which every element has a unique left inverse and a unique right inverse, which need not be equal unless the associative law holds. If a quasi-group is commutative, each element has at least one inverse, but it need not be unique.

rational number: any number that can be expressed as a ratio, $\frac{a}{b}$, of two integers, of which the latter may not be zero. Usually denoted by $\mathbb{Q}$.

real number: any rational or irrational number, usually denoted by $\mathbb{R}$.

[2]Extracts from Borowski and Borwein [1989]

ring: a non-empty set endowed with two binary operations, usually called addition and multiplication, such that the set is an Abelian group under the addition and a semi-group under the multiplication, the latter being both left and right distributive over addition. If, furthermore, the ring has a multiplicative identity element, it is said to be a ring with identity.

semi-group: a set endowed with an associative binary operation, usually called addition, under which it is closed.

transcendental number: a number that is real but not algebraic. We denote this by $\mathbb{T}$.

Bibliography

Almeida e Costa, F. and Rocha, L. e. (2005). *Artificial Life. Special Issue on Embodied and Situated Cognition*, 11(1-2).

Aristotle (1955). *On sophistical refutations*. Harvard University Press, USA.

Aristotle (1957). *Aristotle's prior and posterior analytics: a revised text*. Clarendon Press, Oxford.

Aristotle (1976). *Rhetoric*. Berolini.

Baker, S. (1993). *Aspects of the Constructive Omega Rule within Automated Deduction*. PhD thesis, University of Edinburgh, Edinburgh, UK.

Barnard, T. and Neill, H. (1996). *Mathematical Groups: Teach Yourself*. Hodder Headline Plc., London.

Beaudoin, L. P. (1995). *Goal processing in autonomous agents*. PhD thesis, Faculty of Science, University of Birmingham.

Bernays, P. (1967). *'Hilbert, David', in The Encyclopedia of Philosophy – Edwards, P. (ed)*, volume 3. New York, Macmillan Publishing Co. and the Free Press.

Bird, A. (1998). *Philosophy of Science*. Routledge, London.

Borowski, E. J. and Borwein, J. M. (1989). *Dictionary of Mathematics*. Harper Collins, Glasgow.

Brooks, R. A. (1991). Intelligence without representation. In Haugeland, J., editor, *Mind Design II*, pages 395 – 420. MIT Press, Cambridge, MA.

Buchanan, B. and Feigenbaum, E. (1978). Dendral and Meta-Dendral: Their applications dimension. *AI*, 11.

Bundy, A. (1990). What kind of field is AI? In Partridge, D. and Wilks, Y., editors, *The foundations of artificial intelligence : a sourcebook*, pages 213–222. CUP, Cambridge.

Bundy, A. (2006). A very mathematical dilemma. *The Computer Journal*, 49(4):480–486. Delivered as Boole Lecture, Cork, Ireland, February 2006.

Bundy, A., Jamnik, M., and Fugard, A. (2004). What is a proof? *Submitted to Royal Society*, Discussion meeting: "The nature of mathematical proof" 18th - 19th October 2004.

Burrell, H. (1927). *The Platypus*. Angus and Robertson Ltd., Sydney.

Burton, D. (1985). *The History of Mathematics*. Allyn and Bacon, Inc, Boston, USA.

Carbogim, D., Robertson, D., and Lee, J. (2000). Argument-based applications to knowledge engineering. *The Knowledge Engineering Review*.

Cauchy, A. L. (1813). Recherches sur les polyèdres. *Journal de l'École Polytechnique*, 9:68–86.

Cauchy, A. L. (1821). *Cours d'Analyse de l'École Royale Polytechnique*. de Bure, Paris.

Church, A. (1932). A set of postulates for the foundation of logic. *Annals of Mathematics*, 33:pp. 346–66.

Cohen, L. (1962). *The diversity of meaning*. Methuen & Co Ltd, London.

Colton, S. (2001a). *Automated Theory Formation in Pure Mathematics*. PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh.

Colton, S. (2001b). Experiments in meta-theory formation. In Wiggins, G., editor, *Proceedings of the AISB'01 Symposium on Artificial Intelligence and Creativity in Arts and Science*.

Colton, S. (2002). *Automated Theory Formation in Pure Mathematics*. Springer-Verlag.

Colton, S., Bundy, A., and Walsh, T. (1999). Automatic concept formation in pure mathematics. In *Proceedings IJCAI-99*.

Colton, S., Bundy, A., and Walsh, T. (2000a). Agent based cooperative theory formation in pure mathematics. In *Proceedings of the AISB-00 Symposium on Creative and Cultural Aspects and Applications of AI and Cognitive Science*.

Colton, S., Bundy, A., and Walsh, T. (2000b). On the notion of interestingness in automated mathematical discovery. *International Journal of Human Computer Studies*, 53(3):351–375.

Colton, S., Hoermann, F., Sutcliffe, G., and Pease, A. (2005). Machine learning case splits for theorem proving. In *Proceedings of the Automated Reasoning Workshop, Edinburgh, 2005*.

Colton, S. and Pease, A. (2004). The TM system for repairing non-theorems. In *Workshop on Disproving, Proceedings of IJCAR'04*, pages 13–26.

Colton, S., Pease, A., and Ritchie, G. (2001). The effect of input knowledge on creativity. In *Technical Reports of the Navy Center for Applied Research in Artificial Intelligence*.

Corfield, D. (1997). Assaying Lakatos's philosophy of mathematics. *Studies in History and Philosophy of Science*, 28(1):99–121.

Crawshay-Williams, R. (1957). *Methods of Criteria of reasoning: An Inquiry into the Structure of Controversy*. London: Routledge and Kegan Paul.

Crelle, A. L. (1826-1827). *Lehrbuch der Elemente der Geometrie*, volume 1,2. Reimer, Berlin.

Crombie, A. C. (1994). *Styles of Scientific Thinking in the European Tradition: the History of Argument and Explanation especially in the Mathematical and Biomedical Sciences and Arts*. Gerald Duckworth & Co, Ltd.

Dennett, D. C. (Feb. 25 1971). Intentional systems. *The Journal of Philosophy*, 68(4):pp. 87–106.

Dunmore, C. (1992). Meta-level revolutions in mathematics. In Gillies, D., editor, *Revolutions in Mathematics*, pages 209–225. Clarendon Press, Oxford.

Elvang-Goransson, M., Krause, P., and Fox, J. (1993). Dialectical reasoning with inconsistent information. In *Proceedings of Uncertainty in AI*.

Euler, L. (1758a). Demonstratio nonnullarum insignium proprietatus quibus solida hedris planis inclusa sunt praedita. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4:140–160.

Euler, L. (1758b). Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4:109–140.

Fauconnier, G. and Turner, M. (1998). Conceptual integration networks. *Cognitive Science*, 22(2):133–187.

Feferman, S. (1978). The logic of mathematical discovery vs. the logical structure of mathematics. In Asquith, P. D. and Hacking, I., editors, *Proceedings of the 1978 Biennial Meeting of the Philosophy of Science Association*, volume 2, pages 309–327. Philosophy of Science Association, East Lansing, Michigan.

Feyerabend, P. (1975). *Against Method*. Verso, London.

Fielder, A. (2001). Dialog-driven adaptation of explanations of proofs. In Nebel, B., editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1296–1300, Seattle, WA. Morgan Kaufmann.

Fourier, J. (1808). Mémoire sur la propagation de la chaleur dans les corpe solides (extrait). *Nouveau Bulletin des Sciences, par la Société Philomathique de Paris*, 1:pp. 112 – 16.

Frege, G. (1879). *Begriffsschrift, Eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. English translation in From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931 - Jean van Heijenoort (ed)*. Harvard University Press, Harvard.

Frege, G. (1893/1903). *Grundgesetze der Arithmetik, Vols I and II*. Reclam Philipp Jun.

Freund, Y. and Schapire, R. E. (September, 1999). A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14(5):pp. 771 – 780.

Gärdenfors (1992). *Belief revision*. Cambridge University Press, Cambridge.

Gillies, D. (1996). *Artificial Intelligence and Scientific Method*. Oxford University Press, Oxford.

Gillies, D. (2002). Logicism and the development of computer science. In Kakas, A. C. and Sadri, F., editors, *Computational Logic. Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408 of *Lecture Notes in Artificial Intelligence*, pages 588–604. Springer.

Hacking, I. (1981). Lakatos's philosophy of science. In Hacking, I., editor, *Scientific Revolutions*, pages 128 – 1443. Oxford University Press, Oxford.

Haggith, M. (1996). *A meta-level argumentation framework for representing and reasoning about disagreement*. PhD thesis, Dept. of Artificial Intelligence, University of Edinburgh.

Hallett, M. (1979a). Towards a theory of mathematical research programmes (i). *British Journal for the Philosophy of Science*, 30:1 – 25.

Hallett, M. (1979b). Towards a theory of mathematical research programmes (ii). *British Journal for the Philosophy of Science*, 30:135 – 159.

Hallett, M. and Majer, U., editors (2004). *David Hilbert's Lectures on the Foundations of Geometry: 1891-1902*. Springer-Verlag, Berlin Heidelberg.

Hamblin, C. L. (1970). *Fallacies*. Methuen, London.

Hardy, G. H. (1928). Mathematical proof. *Mind*, 38:pp. 11–25.

Hayes-Roth (1983). Using proofs and refutations to learn from experience. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach*, pages 221–240. Tioga Publishing Company, Palo Alto, CA.

Hayes-Roth, B. (1985). A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321.

Hempel, C. G. (1945). Studies in logic and confirmation. *Mind*, 54:pp. 1–26.

Hilbert, D. (1901). *The Foundations of Geometry. English translation by Townsend, E. J.* The Open Court Company, 1st edition.

Huhns, M, N. and Stephens, L. M. (2001). Multiagent systems and societies of agents. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages pp. 79 – 120. The MIT Press, Cambridge, Massachusetts.

Jamnik, M. (2001a). *Mathematical Reasoning with Diagrams: From Intuition to Automation*. CSLI Press, Stanford University, Stanford, CA.

Jamnik, M. (2001b). *Mathematical Reasoning with Diagrams: From Intuition to Automation*. CSLI Press, Stanford, CA.

Jennings, N. R., Parsons, S., Noriega, P., and Sierra, C. (1998). On argumentation-based negotiation. In *Proceedings of the International Workshop on Multi-Agent Systems*, Boston, USA.

Jonquières, E. (1890). Note sur un Point Fondamental de la Théorie des Polyèdres. *Comptes Rendus des Séances de l'Académie des Sciences*, 110:pp. 110–115.

Kadvany, J. (2001). *Imre Lakatos and the Guises of Reason*. Duke University Press, Durham and London.

Karmiloff-Smith, A. (1990). Constraints on representational change: Evidence from children's drawing. *Cognition*, (34):57 – 83.

Kerber, M. (1992). *On the Representation of Mathematical Concepts and their Translation into First Order Logic*. PhD thesis, Fachbereich Informatik, Universität Kaiserslautern.

Koetsier, T. (1991). *Lakatos's Philosophy of Mathematics. A Historical Approach*. Amsterdam, North-Holland.

Kuhn, T. (1970). *The Structure of Scientific Revolutions*. The University of Chicago Press, Chicago, USA.

Lakatos, I. (1970). Falsification and the methodology of scientific research programmes. In Lakatos, I. and Musgrave, A., editors, *Criticism and the growth of knowledge*, pages 91–195. CUP, Cambridge.

Lakatos, I. (1976). *Proofs and Refutations*. CUP, Cambridge, UK.

Lakatos, I. (1978a). Infinite regress and foundations of mathematics. In Worral, J. and Currie, G., editors, *Mathematics, Science and Epistemology*, pages 3 – 23. Cambridge University Press, Cambridge.

Lakatos, I. (1978b). A renaissance of empiricism in the recent philosophy of mathematics. In Worral, J. and Currie, G., editors, *Mathematics, Science and Epistemology*, pages 24 – 42. Cambridge University Press, Cambridge.

Lakatos, I. (1981). History of science and its rational reconstructions. In Hacking, I., editor, *Scientific Revolutions*, pages 107 – 127. Oxford University Press, Oxford.

Lakatos, I. (August, 1963b). Proofs and refutations (II). *The British Journal for the Philosophy of Science*, 14:120–139.

Lakatos, I. (February, 1964). Proofs and refutations (IV). *The British Journal for the Philosophy of Science*, 14:296–342.

Lakatos, I. (May, 1963a). Proofs and refutations (I). *The British Journal for the Philosophy of Science*, 14:1–25.

Lakatos, I. (November, 1963c). Proofs and refutations (III). *The British Journal for the Philosophy of Science*, 14:221–245.

Lakoff, G. and Núñez, R. (2001). *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books Inc., U.S.A.

Langley, P. (1999). The computer-aided discovery of scientific knowledge. In *Proceedings of the First International Conference on Discovery Science*, Fukuoka, Japan. Springer.

Langley, P. (2002). Lessons for the computational discovery of scientific knowledge. In *Proceedings of First International Workshop on Data Mining Lessons Learned*, pages 9–12, Sydney.

Langley, P., Simon, H., Bradshaw, G., and Żytkow, J. (1987). *Scientific Discovery*. Cambridge, MA: MIT Press/Bradford Books.

Larvor, B. (1998). *Lakatos: An Introduction*. Routledge, London.

Lenat, D. (1976). *AM: An Artificial Intelligence Approach to Discovery in Mathematics*. PhD thesis, Stanford University.

Lenat, D. (1983). Eurisko: A program which learns new heuristics and domain concepts. *Artificial Intelligence*, 21.

Matthiessen, L. (1863). Über die Scheinbaren Einschränkungen des Euler'schen Satzes von den Polyedern. *Zeitschrift für Mathematik und Physik*, 8:pp. 1449–450.

McCune, W. (1990). The otter user's guide. Technical Report ANL/90/9, Argonne National Laboratories.

McCune, W. (2001). MACE 2 reference manual. Technical Report ANL/MCS-TM-249,, Argonne National Laboratories.

McNeill, F., Bundy, A., and Walton, C. (2004). Facilitating agent communication through detecting, diagnosing and refining ontological mismatch. In *Proceedings of the KR2004 Doctoral Consortium*. AAAI Technical Report.

Meikle, L. and Fleuriot, J. (2003). Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Theorem Proving in Higher Order Logics*, volume 2758, pages 319–334. Springer.

Mill, J. (1973). In Robson, J. M., editor, *A System of Logic: The Collected Works of John Stuart Mill*, volume 7. University of Toronto Press, Toronto.

Mitchell, T. (1997). *Machine Learning*. The McGraw-Hill Companies, Inc.

Mozley Moyal, A. (2001). *Platypus: The Extraordinary Story of How a Curious Creature Baffled the World*. Smithsonian Institution Press.

Muggleton, S. and Feng, C. (1992). Efficient iduction of logic programs. In Muggleton, S., editor, *Inductive Logic Programming*, pages 231–98. Academic Press.

Naess, A. (1953). *Interpretation and Preciseness: A Contribution to the theory of Communication*. Oslo: Skrifter utgitt ar der norske videnskaps academie.

Oliveri, G. (2006). Mathematics as a quasi-empirical science. *Foundations of Science*, 11:41 – 79.

Pease, A. and Colton, S. (2004). Automatic conjecture modification. In *Proceedings of the Automated Reasoning Workshop, Leeds, 2004*.

Pease, A., Colton, S., Smaill, A., and Lee, J. (2002). Semantic negotiation: Modelling ambiguity in dialogue. In *Proceedings of Edilog 2002, the 6th Workshop on the semantics and pragmatics of dialogue*, Edinburgh, UK.

Pease, A., Colton, S., Smaill, A., and Lee, J. (2004). A model of Lakatos's philosophy of mathematics. *Computing and Philosophy (ECAP)*.

Pease, A., Winterstein, D., and Colton, S. (2001). Evaluating machine creativity. In Weber, R. and von Wangenheim, C. G., editors, *Case–Based Reasoning: Papers from the Workshop Programme at ICCBR'01*, pages 129–137. Washington, DC: Naval Research Laboratory, Navy Centre for Applied Research in Artificial Intelligence.

Pereira, F. C. and Cardoso, A. (2003). The horse-bird creature generation experiment. *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour(AISBJ)*, 1(3):257–280.

Plato (1993). *The Republic*. OUP, Oxford.

Polya, G. (1945). *How to solve it*. Princeton University Press.

Polya, G. (1954). *Mathematics and plausible reasoning*, volume Vol. 1, Induction and analogy in mathematics. Princeton University Press.

Polya, G. (1962). *Mathematical Discovery*. John Wiley and Sons, New York.

Popper, K. R. (1972). *Objective Knowledge*. OUP, Ely House, London.

Quinlan, J. R. (1979). Discovering rules by induction from large collections of examples. In Michie, D., editor, *Expert System in the Micro-Electronic Age*, pages 168 – 201. Edinburgh University Press, Edinburgh.

Quinlan, J. R. (1986). Induction of decision trees. *Mchine Learning*, 1:86–106.

Rissland, E. L. and Skalak, D. B. (1991). Cabaret: Statutory interpretation in a hybrid architecture. *International Journal of Man-Machine Studies*, 34:pp. 839–887.

Ritchie, G. (2001). Assessing creativity. In Wiggins, G., editor, *Proceedings of the AISB'01 Symposium on AI and Creativity in Arts and Science*, pages 3 – 11. SSAISB.

Ritchie, G. (2005). On transformational creativity. In *Proceedings of the Computational Creativity Workshop, IJCAI*, Edinburgh.

Ritchie, G. (2006). Reinterpretation and viewpoints. *HUMOR special issue on Cognitive Linguistics (To appear)*.

Romacker, M. and Hahn, U. (2001). Context-based ambiguity management for natural language processing. In *Proceedings of the Third International Conference on Modelling and Using Context*, pages 184 – 197, Dundee, Scotland.

Russell, B. (1971). *Logic and Knowledge: Essays 1901-1950*. George Allen and Unwin Ltd., London.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: a Modern Approach*. Prentice-Hall, Englewood Cliffs, New Jersey.

Ryle, G. (1949). *The Concept of Mind*. Hutchinson's University Library, London.

Sartor, G. (June, 1993). A simple computational model for nonmonotonic and adversarial legal reasoning. In *Proceedings of the Fourth International Conference on Artificial Intelligence and Law*, Amsterdam, The Netherlands.

Schapire, R. (2002). The boosting approach to machine learning: An overview. In *MSRI Workshop on Nonlinear Estimation and Classification*.

Shapiro, S. (2000). *Thinking about Mathematics: The philosophy of mathematics*. OUP, Oxford.

Simon, H. (1997). Machine discovery. *Foundations of Science*, 2:pp.171 – 200.

Skalak, D. B. and Rissland, E. L. (1991). Argument moves in a rule-guided domain. In *Proceedings of the 3rd International Conference on Artificial Intelligence and Law*, pages 1 – 11, New York, USA. ACM Press.

Sloman, A. (1978). *The Computer Revolution in Philosophy*. The Harvester Press, Ltd.

Sussman, G. (1975). *A computational model of skill acquisition*. American Elsevier.

Sutcliffe, G. and Suttner, C. (1998). The TPTP problem library: CNF release v1.2.1. *Journal of Automated Reasoning*, 2(21):177–203.

Thagard, P. (1993). *Computational Philosophy of Science*. MIT Press, Cambridge, Mass.

Thagard, P. (1998). Computation and the philosophy of science. *The Digital Phoenix: How Computers are Changing Philosophy*, pages pp. 48 – 61.

Toulmin, S. (1958). *The uses of argument*. Cambridge University Press.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59:433–460.

Tymoczko, T., editor (1998). *New directions in the philosophy of mathematics*. Princeton University Press, Princeton, New Jersey.

Veale, T. and O'Donoghue, D. (2000). Computation and blending. *Cognitive Linguistics, Special Issue on Conceptual Blending*, 11(3/4):253–281.

Wells, D. (1997). *The Penguin Dictionary of Curious and Interesting Numbers*. Penguin Books Ltd., London.

Whitehead, A. N. and Russell, B. (1910, 1912, 1913). *Principia mathematica: 3 vols*. Cambridge University Press, Cambridge.

Wilder, R. (1944). The nature of mathematical proof. *The American Mathematical Monthly*, 52:pp. 1309–323.

Winterstein, D. (2004). *Using Diagrammatic Reasoning for Theorem Proving in a Continuous Domain*. PhD thesis, University of Edinburgh.

Wooldridge, M. (2001). Intelligent agents. In Weiss, G., editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages pp. 27–77. The MIT Press, Cambridge, Massachusetts.

Wooldridge, M. (2002). *An Introduction to MultiAgent Systems*. John Wiley a Sons, Ltd, Chichester, West Sussex.